ESCUELA DE INGENIERÍA DE BILBAO

UNIVERSIDAD DEL PAÍS VASCO

MÁSTER UNIVERSITARIO EN INGENIERÍA DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA

TRABAJO FIN DE MÁSTER

APLICACIÓN DE PICK AND PLACE CON ROBOT PARALELO MINI DELTA INTEGRANDO VISIÓN ARTIFICIAL

Estudiante: Roberto Alonso Mijangos

Directoras: Itziar Cabanes Axpe

Aitziber Mancisidor Barinagarrementeria

Curso académico: 2021 / 2022

Agradecimientos

Me gustaría agradecer a Itziar Cabanes y Aitziber Mancisidor la cooperación e interés que ha mostrado en este trabajo, por haber sido mi guía, haber mantenido la misma energía y paciencia durante todas las fases del proyecto.

También quiero agradecer al Departamento de Ingeniería de Sistemas y Automática de la Escuela de Ingeniería de Bilbao, por darme la oportunidad de realizar este proyecto, a su gente por la ayuda ofrecida y a los técnicos César y Raúl por haber contribuido con su ayuda en hacer posible este proyecto.

Después de un duro y largo camino para llegar hasta aquí, quiero agradecer a mi madre, a mi padre, a mi hermana y a mi pareja el apoyo y ánimo transmitido durante todo este tiempo, ya que sin ellos no lo hubiera conseguido.

Para conseguir superar este máster, no existen atajos, tan solo sirve el esfuerzo, trabajo y sacrificio, realizado durante dos intensos años. En estos años, he crecido como persona como nunca lo había hecho antes, he aprendido que la gente maravillosa escasea y no voy a separarme de las que he encontrado. Con esto, quiero agradecer a todos los amigos que me han ayudado y han estado en los momentos difíciles.

Aplicación de pick and place con un robot paralelo Mini Delta integrando visión artificial

Resumen

La evolución que se está produciendo en la industria durante la Cuarta Revolución Industrial está llevando a las empresas a la búsqueda de la automatización y la creación de fábricas inteligentes. En este ámbito han surgido los robots paralelos como respuesta a aquellas aplicaciones que se quieren realizar con una gran precisión y alta velocidad.

Paralelamente se está dotando a estos robots de cámaras que, junto con programas de visión artificial, hacen que estas células robotizadas sean inteligentes y tomen decisiones a partir del entorno que perciben.

Este Trabajo Fin de Master tiene como objetivo el diseño e implementación de una célula robotizada con un sistema de visión artificial para operaciones pick and place.

La célula robotizada contiene un robot paralelo tipo Delta (de la empresa Omron), cuya arquitectura de control abierta permite implementar estrategias de control avanzada. Para la programación del robot y sus distintas tareas, se va a utilizar el entorno de programación de Sysmac Studio.

La aplicación que se ha implementado con la célula robotizada consiste en formar palabras mediante cubos con letras serigrafiadas en la cara superior. De manera que, el usuario desde una interfaz gráfica selecciona la palabra que desea formar y se utiliza un sistema de visión artificial que detecta donde se encuentra cada una de las letras en cada momento, enviará esos datos codificados al robot, y este cogerá y colocará los cubos, con las letras necesarias para formar la palabra.

Para su monitorización y control se ha realizado una interfaz gráfica que permite el control de la palabra que va a formar el robot y de su velocidad, al mismo tiempo que informa del estado de la conexión TCP y de las posiciones de los cubos que va a recoger y depositar en cada plataforma.

Este trabajo se ha realizado exclusivamente en la Universidad del País Vasco, concretamente en la Escuela de Ingeniería de Bilbao.

Palabras clave: Automatización, Robótica, Robot paralelo, Visión Artificial, Pick and Place.

Pick and place application with Mini Delta parallel robot integrating artificial vision

Abstract

The evolution that is taking place in industry during the Fourth Industrial Revolution is leading companies to seek automation and the creation of smart factories. Parallel robots have emerged in this field as a response to applications that need to be carried out with high speed and precision.

At the same time, these robots are being equipped with cameras that, together with artificial vision programmes, make these robotic cells intelligent and make decisions based on the environment they perceive.

The objective of this Master's Thesis Project is the design and implementation of a robotic cell with an artificial vision system for pick and place operations.

This robotic cell contains a parallel robot type Delta (from the company Omron), whose open control architecture allows the implementation of advanced control strategies. For the programming of the robot and its various tasks, the Sysmac Studio programming environment will be used.

The application to be implemented with the robot cell consists of forming words by means of cubes with letters screen-printed on the upper face. In this way, the user will use a graphic interface to select the word he/she wishes to form and an artificial vision system will be used to detect where each of the letters is at any given moment, send this coded data to the robot, and the robot will pick up and place the cubes with the letters necessary to form the word.

For its monitoring and control, a graphic interface has been created that allows control of the word that the robot is going to form and its speed, at the same time as it reports the status of the TCP connection, the positions of the cubes that it is going to pick up and deposit on each platform.

This work has been carried out exclusively at the University of the Basque Country, specifically at at the Faculty of Engineering from Bilbao.

Keywords: Automation, Robotics, Parallel Robot, Artificial Vision, Pick and Place.

Pick and place aplikazioa Mini Delta robot paraleloa eta ikusmen artifiziala erabiliz

Laburpena

Laugarren Industria Iraultzaren barnean, enpresak prozesu desberdinak automatizatzen eta lantegi adimendunak sortzen ari dira. Honetako, robot paraleloak oso egokiak izan daitezkeela ikusi da, zehaztasun eta abiadura handiz lan egiteko ahalmena baitute.

Aldi berean, azken urteetan, robot horiei kamerak jartzen hasi dira, ikusmen artifizialean oinarrituz, ingurunearen egoeraren arabera erabakiak hartzeko gaitasuna duten zelula robotizatu adimendunak sortuz.

Master Amaierako Lan honen helburua, pick and place operazioetarako, ikusmen artifiziala duen zelula robotizatu bat diseinatzea eta inplementatzea da.

Zelula robotizatua, Omron enpresako Delta robot paralelo batean oinarrituta dago. Robot honen kontrol irekiko arkitekturak, kontrol aurreratuko estrategia desberdinak ezartzea ahalbidetzen du. Robota eta haren zereginak programatzeko, Sysmac Studioko programazio-ingurunea erabiliko da.

Zelula robotizatua erabiliz inplementatuko den aplikazioa, goiko aldean letra serigrafiatuak dituzten kuboen bidez hitzak osatzean datza. Behin aplikazioa diseinatu eta inplementatuta, erabiltzaileak, interfaze grafikoan osatu nahi duen hitza hautatuko du. Aukeraketa hau egin ondoren, ikusmen artifizialeko sistema bat erabiliko da, letra bakoitza une bakoitzean non dagoen antzemateko. Ikusmen artifizialeko sistemak, datu kodetu horiek robotari bidaliko dizkio, eta honek, behar diren letrak dituzten kuboak mugituko ditu aukeratutako hitza osatuz.

Interfaze grafiko bat garatuko da, aplikazioa hau monitorizatu eta kontrolatzeko. Interfaze honek, robotak osatuko duen hitza eta mugimenduen abiadura aukeratzea ahalbidetzeaz gainera, TCP konexioaren egoeraren eta kuboen posizioen berri emango du.

Lan hau Euskal Herriko Unibertsitateko Bilboko Ingeniaritza Eskolan burutu da.

Hitz gakoak: Automatizazioa, Robotika, Robota Paraleloa, Ikusmen Artifiziala, Pick and Place.

Índice

Índice de figuras

Índice de tablas

Acrónimos

3D	Tres dimensiones
Mm	Milímetros
Cm	Centímetros
PLC	Controlador lógico programable
S	Segundos
Min	Minutos
CPU	Unidad Central de Procesamiento
Gr.	Gramos
GDL	Grado de libertad
º	Grados angulares
Pick and Place	Cogida y dejada
º/s	Grados/segundo
Kg	Kilogramos
M	Métrica
UNE	Una Norma Española
ISO	Organización Internacional de Normalización
Ø	Diámetro
N/mm2	Newton/milímetro^2
IoT	Internet of Things (Internet de las Cosas)
E/S	Entradas/Salidas
TCP	Transmission Control Protocol
UDP	User datagram protocol

Capítulo 1:
INTRODUCCIÓN

1. Introducción.

La robótica nació llena de promesas para el futuro con un desarrollo tan rápido e intenso que, en pocos años, habría alcanzado metas que en aquellos momentos correspondían al ámbito de la ciencia ficción. El continuo desarrollo junto a las novedosas metodologías de la inteligencia artificial, permitían imaginar robots con una gran flexibilidad y capacidad de adaptación al entorno, que invadirían todos los sectores productivos de forma rápida e imparable.

Actualmente, se está asistiendo a uno de los momentos más interesantes dentro de la industria, ya que, la irrupción de la digitalización está modificando las máquinas y los sistemas de automatización de los procesos productivos, haciéndose denominar esta evolución como “Industria 4.0”.

Los robots industriales ocupan un gran lugar dentro de la automatización de la producción y su papel continúa consolidándose en los últimos años. Por tanto, la automatización y la robótica se han convertido en uno de los grandes precursores de la economía de las grandes potencias, ya que, éstas permiten producir una mayor cantidad de producto con mayor calidad y mejores precios.

Ante esta realidad, el grupo de investigación de Sensorización Virtual para Bioingeniería perteneciente a la Universidad del País Vasco, quiere hacer una aportación con este Trabajo Fin de Máster que consiste en el diseño e implementación de una célula robotizada con un sistema de visión artificial para operaciones pick and place. Dicha célula contiene un robot paralelo tipo Mini Delta de Omron cuyas acciones serán distintas en función de las operaciones de pick and place que se quieran ejecutar.

Para su realización, será fundamental documentarse acerca del contexto actual existente en el ámbito industrial y de la robótica, cuya evolución en los últimos años ha sido determinante. La profundización en estos conceptos permitirá la definición de una serie de objetivos que aseguren que el proyecto es realizado correctamente acorde con la denominada Industria 4.0.

Una vez se hayan definido las bases sobre las que se va a fundamentar este proyecto, se procederá a analizar detalladamente las diferentes alternativas valoradas para la célula robotizada a diseñar, destacando, como no podría ser de otra manera, el robot paralelo que será integrado. Esta elección resulta fundamental, ya que, de ella va a depender la elección del escenario a desarrollar en la célula.

Tras analizar todas las alternativas en cada tarea a realizar se aplicará la que se considere más óptima. Para la realización de este diseño resulta imprescindible la utilización de herramientas de simulación para llevar a cabo la programación del robot.

Por último, se implementará físicamente el escenario diseñado, incluyendo el montaje de un cuadro eléctrico al cual se cablearán las E/S del robot y los elementos auxiliares que forman la célula robotizada.

1.1. Historia de los robots industriales.

En la actualidad se conocen varias definiciones para el término robot. Según la Asociación de Industrias Robóticas (RIA) un robot es un manipulador reprogramable multifuncional diseñado para manipular y/o transportar material a través de movimientos programados para la realización de tareas diversas, es decir, es un dispositivo mecánico que puede ser programado para llevar a cabo instrucciones y realizar tareas que son complicadas para el ser humano [1].

Los robots industriales han sufrido una rápida evolución desde la aparición de la primera generación de robots industriales a principios de los 60 [2]. La compañía estadounidense Unimation, se encargó de dirigir el primer robot (ver Figura 1.1) de transferencia programable para el levantamiento de grandes piezas de metal caliente, para depositarlo posteriormente en líquidos refrigerantes, tarea que era muy peligrosa para los operarios.

Figura 1.1. Primer robot industrial instalado en General Motors llamado Unimate.

La mayoría de los robots de primera generación empleaban actuadores neumáticos y eran controlados mediante puertas lógicas, como relés, que activaban y desactivaban válvulas actuando como reguladores automáticos. El primer robot industrial, llamado Unimate, fue creado por George Charles Devol en 1954. Las funciones de los primeros robots se limitaban a simples funciones de carga/descarga y estaban diseñados para soportar grandes cargas.

La segunda generación de robots (1968-1977) coincidió con la Tercera Revolución Industrial. En 1969, Víctor Scheinman desarrolló el Standford Arm (ver Figura 1.2) que fue el primer robot ligero en forma de brazo multiprogramable para diversas tareas [3].

Figura 1.2. Standford Arm desarrollado por Víctor Scheinman.

Estos robots de segunda generación empezaban a reconocer algunos elementos del entorno y su sistema de controlador empleaba microprocesadores o Controladores Lógicos Programables (PLC).

La tercera generación de robots nace en 1978. Esta generación se caracteriza por la invención de robots con diferentes morfologías, como los robots SCARA, de 4 grados de libertad, o los robots DELTA, robots paralelos de 3 grados de libertad (ver Figura 1.3).

Figura 1.3. Robot tipo SCARA (izquierda) y Robot tipo DELTA (derecha).

1.2. Contexto.

El desarrollo del TFM se va a realizar utilizando la célula robotizada del robot paralelo Delta del departamento de Ingeniería de Sistemas y Automática que se puede observar en la figura 1.4.

Figura 1.4. Célula con el espacio superior y el espacio inferior.

Para este trabajo, se parte de una célula robotizada con un chasis rectangular de cuatro caras. En su interior, en el espacio superior, se encuentra el robot delta con el que se va a trabajar y en el espacio inferior, se encuentran los múltiples dispositivos y componentes necesarios para su funcionamiento.

En la figura 1.5, se muestra con mayor detalle la parte superior, donde se encuentra el robot Mini Delta. Se trata de un robot paralelo compuesto por una base fija y una base móvil, ambas unidas por tres brazos con la misma longitud.

Figura 1.5. Espacio superior de la célula robotizada.

1.3. Estructura del trabajo.

En este apartado, se va a exponer la estructura del presente Trabajo Fin de Master que se divide en diez capítulos y siete anexos:

1. En el capítulo 1, se presenta una introducción del trabajo y de la célula robotizada con la que se va a trabajar.
2. En el capítulo 2, se van tratan los objetivos principales y secundarios de este proyecto junto con el alcance del mismo.
3. En el capítulo 3, se plantea un pequeño resumen acerca del estado actual de la robótica y de las tecnologías que la rodean para poder ubicar en contexto este trabajo.
4. En el capítulo 4, se exponen las razones por las que se ha seleccionado el robot, las características técnicas de este y los componentes de la célula robotizada.
5. En el capítulo 5, se realiza el análisis de alternativas de cada una de las tareas a realizar.
6. En el capítulo 6, se explican las tareas realizadas junto con los resultados obtenidos en cada tarea.
7. En el capítulo 7, se realiza una exposición de la metodología empleada para realizar este proyecto, con cada una de sus fases y el tiempo invertido en cada una de ellas.
8. En el capítulo 8, se define el presupuesto necesario para poder llevar a cabo este proyecto.
9. En el capítulo 9, se exponen las conclusiones que se han obtenido durante la ejecución de este trabajo y los trabajos futuros.
10. En el capítulo 10, se publica la bibliografía empleada para documentar este proyecto.

I. En el anexo 1, se muestran los planos de las piezas 3D diseñadas.

II. En el anexo 2, se presenta el código del controlador del robot desarrollado en Sysmac Studio.

III. En el anexo 3, se indica el código implementado para el funcionamiento de la interfaz gráfica en Matlab.

IV. En el anexo 4, se muestra el código diseñado para la comunicación en Python.

V. En el anexo 5, se expone el manual de usuario junto con la guía de instalación.

VI. En el anexo 6, se muestra información adicional sobre ciertos dispositivos que permiten el funcionamiento del robot y son de interés.

VII. En el anexo 7, se indican los pasos a seguir para realizar la calibración mecánica y hardware del robot.

Capítulo 2:
OBJETIVOS Y ALCANCE DEL TRABAJO

2. Objetivos y alcance del trabajo.

Recientemente, hay un creciente interés por incorporar la robótica paralela en los procesos de producción como respuesta a la demanda de producción de hoy en día.

Los tiempos y la calidad de producción juegan un papel muy importante donde las empresas quieren obtener la mayor ventaja posible. Con este objetivo, los centros de producción buscan la manera de automatizar los procesos, de manera que se pueda fabricar de manera continua, sin interrupciones, logrando la mayor flexibilidad posible en la producción y consiguiendo satisfacer las demandas personalizadas de los clientes.

La robótica puede abordar estos retos de producción, pero si además se consigue trasladar al humano fuera de tareas repetitivas, rápidas y tediosas, se obtendrá un beneficio por partida doble.

2.1. Objetivos del proyecto.

El objetivo principal de este proyecto es el diseño e implementación de una célula robotizada para operaciones de pick and place con un sistema de visión artificial para el reconocimiento de los objetos del entorno de trabajo.

Para el control de esta célula se ha implementado una interfaz gráfica de usuario que permita al usuario controlar dicha célula garantizando la seguridad tanto del robot como del operario.

Además, se han analizado varias alternativas que permitan solucionar cada problemática y así, escoger la más adecuada para cada problema. Los objetivos para poder completar este proyecto son:

- Realizar un análisis exhaustivo de las tecnologías actuales para este tipo de aplicaciones y robots.
- Adquirir un nivel básico de conocimientos que permita trabajar con las distintas herramientas software necesarias para cubrir todas las partes del proyecto. Las herramientas son; Sysmac Studio, Matlab, Fusion 360 y Python.
- Analizar las especificaciones del robot como, por ejemplo; la cinemática del robot, el espacio de trabajo y diseñar el entorno del trabajo del mismo.
- Diseñar un sistema de visión artificial capaz de extraer la información necesaria sobre el escenario y los objetos que se encuentran en él, para poder realizar las distintas tareas del robot.
- Aplicar un protocolo de comunicación que permita enviar la información deseada al robot y utilizar la que mejor se adapte al problema.

- Programar la aplicación de pick and place con el programa de control del robot.
- Diseñar e implementar la interfaz gráfica para qué el usuario pueda decidir que tarea quiere que realice el robot.
- Diseñar e implementar las piezas 3D que componen el entorno de trabajo dentro de la célula robotizada.
- Configurar y calibrar el robot.

2.2. Alcance del trabajo.

El alcance de este TFM comprende la integración de una interfaz gráfica, junto con un sistema de visión artificial dentro de una célula robotizada compuesta por un robot paralelo. Para ello, se han llevado a cabo distintas simulaciones que han permitido completar el proyecto.

Para este fin, es necesario llevar a cabo un análisis de los distintos métodos de extracción de características de una imagen, de la calibración, disposición e iluminación de la cámara. Para ello, resulta imprescindible realizar un análisis de los distintos protocolos de comunicación que permiten desarrollar esta aplicación.

Por último, es necesario analizar y definir el comportamiento del control de la aplicación de pick and place del robot para que cumpla satisfactoriamente la tarea, a tiempo y con seguridad.

Para este proyecto, cada parte se lleva a cabo mediante simulaciones para posteriormente realizar la implementación física con el robot.

Capítulo 3:
ESTADO ACTUAL DE LA TECNOLOGÍA

3. Estado actual de la tecnología.

Desde la creación del primer robot programable en 1954, la robótica ha evolucionado hasta convertirse en una parte indispensable de nuestras vidas. Hoy en día, existen robots en prácticamente cualquier ámbito de la sociedad, y cada día su número y sus capacidades aumenta, haciendo que sea más fácil la vida con ellos.

Desde la aparición de las primeras máquinas de trabajo, el ser humano ha buscado la forma de disminuir el esfuerzo físico y aumentar la eficiencia lo máximo posible. Esta búsqueda es la que ha llevado a la tecnología hasta el estado actual, pudiendo agruparse este desarrollo en cuatro grandes revoluciones industriales.

3.1. Revoluciones industriales.

Cada una de las cuatro revoluciones industriales fue impulsada por un conjunto de tecnologías que transformaron el mundo en su momento, marcando una antes y un después.

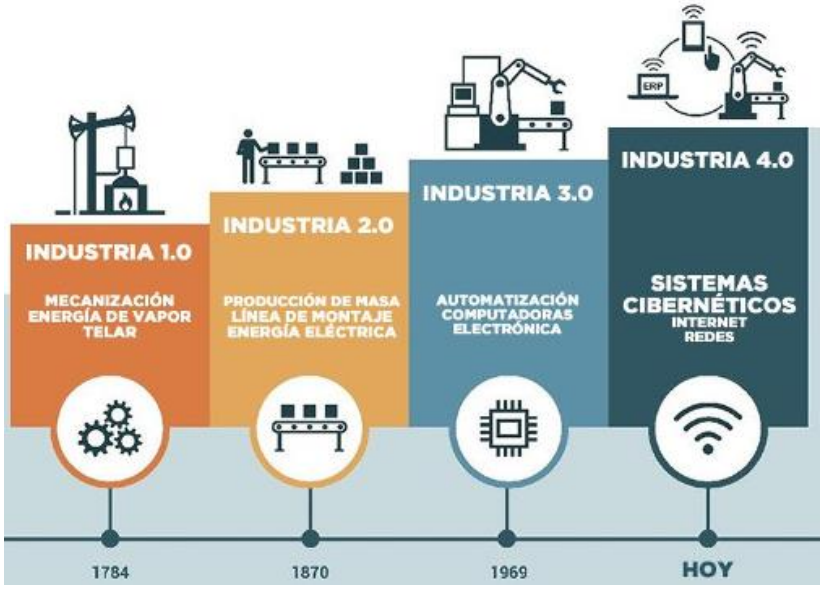


Figura 3.1. Revoluciones industriales.

La Primera Revolución Industrial surgió a mediados del siglo XVIII y tuvo su origen en Inglaterra, país que se convirtió durante mucho tiempo en el primer productor de bienes industriales del mundo y generó consecuencias económicas, sociales, culturales e incluso, ambientales [4].

La máquina de vapor, cuyo combustible era el carbón mineral, fue la base sobre la que se asentó todo el desarrollo. Esto permitió la sustitución de las viejas herramientas artesanales por maquinaria más productiva.

La Segunda Revolución Industrial iniciada entre los años 1850 y 1870, se extiende hasta 1914, coincidiendo con el inicio de la Primera Guerra Mundial. En esta época comienzan a aparecer nuevas formas de energía, como el petróleo, que reemplazó al carbón, y la electricidad [5].

La modernización de los medios de transporte fue otra de las importantes consecuencias tecnológicas. El ferrocarril fue el medio de transporte símbolo de esta nueva época. Asimismo, los barcos a vapor se hicieron cada vez más grandes y veloces. Estos medios facilitaron el transporte de mayor cantidad de personas y mercaderías a grandes distancias y en menos tiempo.

La Tercera Revolución Industrial comienza en la segunda mitad del siglo XX y se asienta sobre nuevas tecnologías de la información y la comunicación, en especial con la aparición de Internet. Nunca antes se había llegado a niveles tan altos de interactividad e intercomunicación. Paralelamente se produce el desarrollo de las energías renovables [6].

La Cuarta Revolución Industrial comienza en 2011 y continua actualmente. El elemento clave de esta Revolución Industrial es la innovación para una mayor reconfiguración a las necesidades de la producción y una mejora en la eficiencia de los recursos.

3.2. Industria 4.0 y robótica industrial.

La Industria 4.0 implica la promesa de una nueva revolución que combina técnicas avanzadas de producción y operaciones con tecnologías inteligentes que se integran en las organizaciones, las personas y los activos.

Esta revolución está marcada por la aparición de nuevas tecnologías como la robótica, la analítica, la inteligencia artificial, las tecnologías cognitivas, la nanotecnología y el Internet of Things (IoT), entre otras [7].

Las organizaciones deben identificar las tecnologías que mejor satisfacen sus necesidades para invertir en ellas. Si las empresas no comprenden los cambios y oportunidades que trae consigo la Industria 4.0, corren el riesgo de desaparecer.

La integración digital de la información desde diferentes fuentes y localizaciones permite llevar a cabo negocios en un ciclo continuo. A lo largo de este ciclo, el acceso en tiempo real a la

información está impulsado por el continuo y cíclico flujo de información y acciones entre los mundos físicos y digitales.

La base de la industria 4.0 está en la creación de fábricas inteligentes (Smart Factories), mediante la introducción del concepto de “internet de las cosas”. El internet de las cosas o

IoT (Internet of Things) es un sistema de dispositivos de computación interrelacionados, que tienen identificadores únicos y capacidad de transferir datos a través de una red, sin necesidad de interacciones humanas [8], [9].

El objetivo de las Smart Factories es satisfacer demandas dinámicas de manera eficiente y automática, sin detener el proceso de producción cada vez que haya que fabricar un producto diferente. Este proceso de producción inteligente se conoce como Smart Manufacturing [10], [11].

La implementación de robots industriales para automatizar los procesos productivos, ha generado una verdadera revolución que sin duda alguna está cambiando el panorama de la producción industrial. Fabricantes de todo el mundo están llevando a cabo la automatización de sus aplicaciones industriales con la finalidad de ser más eficientes, seguros e incrementar su productividad.

Las ventajas que los robots industriales aportan a las empresas y son clave de su éxito son:

- Fiabilidad y calidad mejorada. Todos los brazos robóticos que son distribuidos por las marcas fabricantes de robots industriales deben pasar por un proceso previo de estudio y pruebas de repetitividad y carga útil, con la finalidad de garantizar que cuando estos salgan al mercado puedan realizar sus trabajos con precisión y eficiencia.
- Reducción de costes de producción. Un retorno a la inversión (ROI) rápido, permite poder recuperar los costes de la inversión inicial. Instalar robots industriales genera un impacto positivo en la producción, ya que se incrementan los tiempos de ciclos.
- Incremento de la producción. Una línea de fabricación adecuada es fundamental para incrementar la eficiencia. Los humanos tienen ciertas limitaciones que los robots no, como el hecho de trabajar a una velocidad constante sin descansos, lo que finalmente se expresa en mayores ganancias para las empresas y una mejora de su competitividad.
- Mejor utilización de los espacios de la empresa. Al reducir el espacio ocupado por un área de trabajo a través de la automatización de algunas partes de la línea de producción, se puede disponer de esos espacios para otro tipo de operaciones.
- Mayor calidad en los productos. Los robots industriales están diseñados para realizar trabajos efectivos y precisos. Al implementar estos en la línea de operaciones de una empresa se reducen el número de defectos en los productos, disminuyendo así los costes de los desperdicios de productos.
- Incremento de la seguridad en el lugar de trabajo. El incremento de la seguridad y sobre todo de la ergonomía, es una ventaja importante que ofrece la automatización

de los procesos industriales, al lograr desplazar a los trabajadores de las zonas peligrosas y de los trabajos repetitivos.

Los robots industriales disponen de hasta 7 ejes que pueden ser empleados para la realización de múltiples trabajos, que pueden ir desde la soldadura robotizada, el paletizado de cajas y de sacos, dispensar materiales y productos, cortar láminas o el transporte de materiales entre otros.

Teniendo en cuenta su estructura, los robots se pueden clasificar en dos tipos:

- Robots serie.
- Robots paralelos o manipuladores.

En la figura 3.2, a la izquierda se muestra la estructura cinemática de un robot serie (cadena abierta) y, a la derecha, un robot paralelo (cadena cerrada).

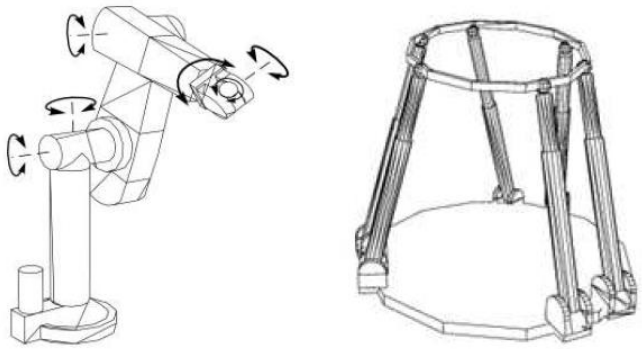


Figura 3.2. Estructura cinemática de un robot serie y paralelo.

Los robots tipo serie están formados por una cadena cinemática abierta, con una estructura similar a un brazo humano. Su principal ventaja frente a los robots paralelos se encuentra en la cinemática y en el espacio de trabajo:

- La cinemática de un robot es el estudio de los movimientos de un robot. En un análisis cinemático la posición, velocidad y aceleración de cada uno de los elementos del robot son calculados sin considerar las fuerzas que causan el movimiento. Su configuración de cadena abierta permite el desacoplo cinemático de sus articulaciones, lo que hace que el control sea más sencillo que para el caso de los paralelos y permita calcular un modelo genérico.
- El espacio de trabajo es mucho más grande en comparación con los robots paralelos y su cálculo mucho más sencillo.

Un robot paralelo es un mecanismo formado por una plataforma móvil que está conectada a una base por al menos dos series de cadenas cinemáticas independientes. Esta configuración de cadena cinemática cerrada, otorga a los robots paralelos, ciertas ventajas con respecto a

los robots serie, en términos de rigidez, velocidad, precisión e inercia en movimiento [12]. La figura 3.2, muestra la estructura cinemática de un robot serie y un robot paralelo.

Sus principales ventajas respecto a los robots serie son:

- La relación carga/potencia es alta, ya que, los accionamientos de potencia conectan directamente la base del robot al efector final, sirviendo de elementos estructurales que actúan de manera simultánea, permitiéndoles manipular cargas superiores a su propio peso.
- Presentan una alta rigidez, lo cual permite mayores precisiones respecto a un robot tipo serie.
- Su arquitectura les permite alcanzar mayores velocidades y aceleraciones que los serie, lo cual les permite realizar tareas industriales de manera más eficiente.
- Al tener los motores sobre una base fija, sus inercias son menores respecto a un robot serie y permite un control más rápido y preciso.

Estos robots destacan por sus aplicaciones como máquina-herramienta, como simuladores espaciales, en tareas de automoción, empaquetadoras de alimentos y ensambladores de microchips, entre otros. La razón por la que se emplean estos robots, es debido a su excelente desempeño dinámico para completar operaciones en las líneas de producción para una clasificación rápida, un agarre preciso y un ensamble seguro [13].

3.3. Robot paralelo tipo Delta.

Dentro de los robots paralelos, cabe destacar los robots tipo Delta (ver Figura 3.3), que se caracterizan por una gran velocidad, un posicionamiento seguro y preciso, costes bajos de mantenimiento y una alta eficiencia, lo que permite amortizar la inversión en periodos de un año [14].

El robot tipo Delta nace de la necesidad de los sectores de manufactura y producción, con el objetivo de utilizar manipuladores que trabajen a alta velocidad y con una alta precisión, en todas las tareas realizadas.

El robot Delta es un tipo de robot paralelo de tres grados de libertad conformado por dos bases unidas por tres cadenas cinemáticas basadas en el uso de paralelogramos. La base superior se encuentra fija, mientras que la base inferior, donde se ubica el efector final, es móvil y siempre está paralela a la base fija. Para su construcción se pueden utilizar actuadores rotacionales o lineales según la aplicación para la cual quiera usarse.

El uso de actuadores montados en la base y eslabones de poca masa permiten a la plataforma móvil alcanzar aceleraciones de hasta 50G (50 veces la aceleración de la

gravedad) en ambientes experimentales y 12G en aplicaciones industriales. Es uno de los robots más veloces, gracias a que puede alcanzar velocidades de hasta 500 m/s [15].

Los robots Delta se utilizan frecuentemente para aplicaciones de empaquetado e impresión 3D, gracias a las altas velocidades que alcanzan.

Figura 3.3. Diferentes robots tipo Delta.

Los robots Delta más actuales pueden llegar a disponer de hasta cinco grados de libertad y tener hasta seis ejes. Su característica más destacable es que pueden ser capaces de ejecutar hasta 300 movimientos por minuto.

En síntesis, los robots Delta están constituidos por (ver Figura 3.4):

- El cuerpo o base que integra todas las articulaciones principales, los servomotores, los reductores planetarios y la electrónica.
- Los brazos mecánicos que actúan como bielas y se encargan de desplazar el efector final sobre la superficie, siempre de forma paralela.
- El efector final en el extremo de los brazos.
- Los gripper o pinzas de vacío ubicados en el efector final que realizan las tareas de recolección.

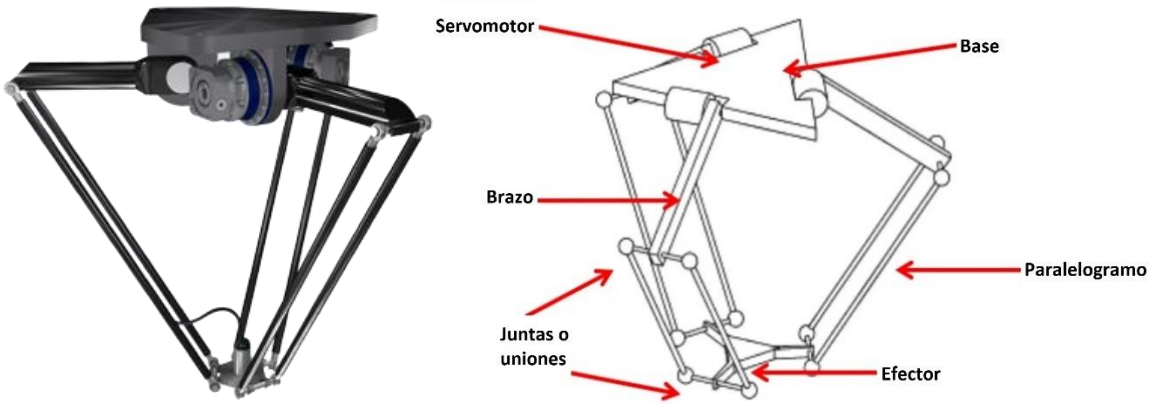


Figura 3.4. Robot paralelo y partes que lo componen.

3.4. Aplicaciones del robot tipo Delta.

La aplicación principal de este tipo de robot es la operación de Pick and place, que consiste en recoger un objeto de una posición para posteriormente depositarlo en otra. La rapidez de esta operación es clave en ciertas partes de las cadenas de producción, donde cada objeto se recoge selectivamente en base a unas características y se depositan en otro lugar, con una colocación u orientación cambiante [16], [17].

Un ejemplo de ello es el área de paletización, donde los objetos se recogen generalmente desde un almacén o una cinta transportadora y se cargan en pallets con unas dimensiones fijas. Esto implica cargar los objetos con distintas colocación y orientaciones para optimizar al máximo su espacio [18].

Por lo tanto, los robots Delta están diseñados especialmente para aplicaciones de baja carga y son muy demandados en la industria alimentaria, electrónica, cosmética y farmacéutica, ya que no hay ningún otro robot que pueda moverse sin sobrecalentarse a velocidades parecidas.

Estos robots en la mayoría de los casos suelen ir acompañados de sistemas de visión artificial. Estos permiten aplicar métodos automatizados e inteligentes de forma precisa y detallada, procesar y analizar las imágenes de los productos en las líneas de montaje. Gracias a los diferentes softwares instalados en los sistemas y en las cámaras de visión artificial, las máquinas automatizadas pueden detectar y/o identificar anomalías en los productos. Por esta razón, la visión artificial es una herramienta imprescindible para inspeccionar y detectar cualquier error en los procesos de producción [19].

La implementación de los sistemas de visión artificial en las líneas de montaje de las industrias y las empresas trae consigo múltiples beneficios, ya que permite solucionar los problemas de control de proceso de fabricación y con ello propiciar el aumento de la producción [20].

En la figura 3.5, se pude observar un robot tipo Delta en operaciones de selección de pimientos. Concretamente, en esta instalación, se dispone un conjunto de robots paralelos colocados en serie, cada uno con su célula y se encargan de recoger de su almacén un pimiento para depositarlo sobre una cinta trasportadora que contiene pimientos alternando sus colores.

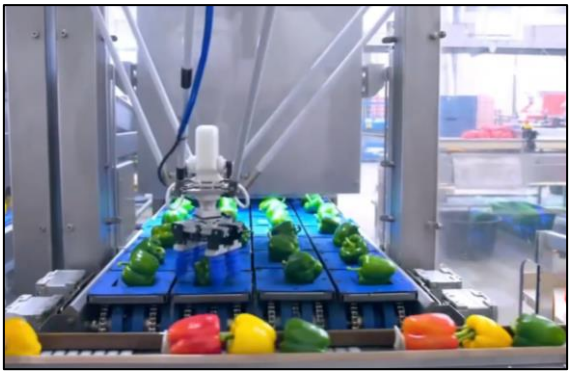


Figura 3.5. Robots tipo Delta en la industria alimentaria.

Los fabricantes de alimentos tienen que cumplir con estándares de lavado de equipos y evitar la exposición del producto a contaminantes potenciales como lubricantes, fragmentos de metal, plástico o polvo. Debido a la arquitectura del robot delta, los motores se pueden aislar bien en gabinetes, y muchos modelos están disponibles con una clasificación de protección de ingreso IP69K, lo que permite que el robot sea explotado por lavado a alta presión y alta temperatura.

Estos robots permiten a estas cadenas de producción trabajar de forma continua las 24 horas a temperaturas bajas, cumpliendo todos los estándares de seguridad alimentaria, sin presentar peligros microbiológicos y acelerando posteriormente el proceso de envasado.

Además, con el paso del tiempo, las mejoras realizadas a los robots Delta hacen que estos ya no se limiten solo a tareas de embalaje y ensamblaje, sino que sean empleados para aplicaciones de ultra precisión (en escala de nanómetros), tales como cirugía robótica remota, relojería y mecanizado de precisión; asimismo, pueden desempeñarse para simuladores de vuelo, manipuladores de grandes cargas y posicionamiento de antenas.

También se pueden encontrar aplicaciones de estos robots en las plataformas de movimiento de los simuladores de vuelo, conducción y en aparatos de posicionamiento para herramientas de cirugía de alta precisión, La figura 3.6, muestra un robot paralelo para simulación en conducción de un vehículo [21].

Figura 3.6. Robot paralelo como simulador para la conducción.

3.5. Trabajos relacionados con el robot tipo Delta.

En este apartado se exponen distintas líneas de investigación en los que actualmente se está trabajado con robots paralelos.

En [22] se plantea un método de planificación de trayectorias de tiempo óptimo en un escenario con dos cintas transportadoras, donde el robot tiene que recoger los discos de una cinta y depositarlos en otra, como se puede ver en la figura 3.7.

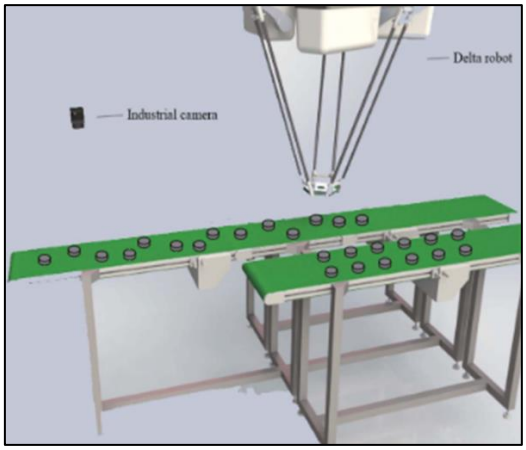


Figura 3.7. Robot paralelo junto a las cintas transportadoras.

Así, este trabajo propone una planificación de trayectorias basado en curvas pitagóricas quinticas para realizar la operación suave y estable a alta velocidad, permitiendo optimizar las trayectorias para reducir el tiempo de ciclo de la operación de recogida y colocación, como se muestra en la figura 3.8.

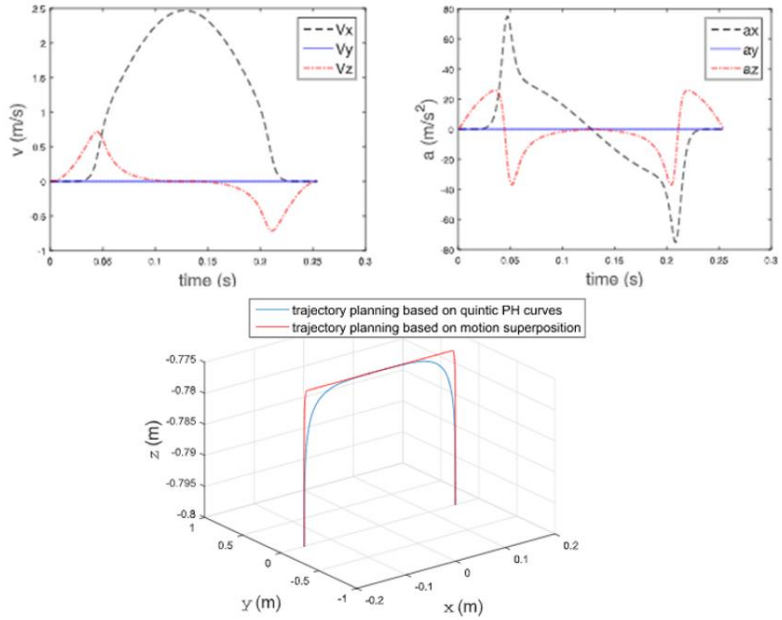


Figura 3.8. Resultados obtenidos mediante la planificación de las trayectorias.

Otros trabajos se centran en la dinámica y los espacios singulares de estos robots. Por ejemplo, algunos trabajos [23]–[25] proponen un control completo de impedancia del espacio de tareas con dinámica inversa usando cuaterniones duales para el modelado cinemático. Además, se integra una red neuronal para componer una representación libre de singularidades y adecuada para el cálculo en tiempo real. Esta red es capaz de calcular la cinemática directa más de 150 veces más rápido que un algoritmo de resolución de ecuaciones numéricas, con un error de estimación promedio de menos de 0.5mm.

En diferente ámbito, existen varios trabajos acerca de robots tipo Delta dedicados a jugar al tenis de mesa. Para ello emplean distintas técnicas de control visual, donde cabe destacar el realizado por la compañía OMRON con el robot FORPHEUS, [26], que se puede ver en la figura 3.9.

Figura 3.9. Robot FORPHEUS.

Se trata de un robot paralelo diseñado para ser tutor de tenis de mesa equipado con inteligencia artificial, capacidad de sentir, moverse y responder adecuadamente a las acciones del oponente, e incluso cuenta con diferentes niveles de habilidad.

Este robot no solo juega, sino que también ayuda a mejorar la habilidad del jugador humano, lo que lo convierte en un tutor en lugar de una simple máquina para golpear la pelota.

Figura 3.10. Imágenes obtenidas por el sistema de visión del robot.

La capacidad de predicción del sistema de visión artificial (ver Figura 3.10) se ha mejorado mediante el uso de cuatro cámaras y una cámara estéreo de procesamiento de imágenes de

alta velocidad, que permite detectar simultáneamente la forma, habilidad, posición y velocidad del jugador de la raqueta y la trayectoria de la pelota. De hecho, es capaz de predecir la dirección en que el jugador devolverá la pelota antes de que sea golpeada.

A modo de conclusión, en este trabajo se aborda el diseño y desarrollo que reproduce situaciones industriales de gran utilidad, como son la clasificación de piezas mediante visión artificial y las correspondientes tareas de pick and place.

Con esta intención, y poniendo el punto de mira en la rapidez para clasificar diferentes piezas mediante movimientos pick and place, se pretende abordar la aplicación de técnicas de visión artificial en una célula robotizada donde se incorporará un robot paralelo.

Capítulo 4:
SELECCIÓN DEL ROBOT

4. Selección del robot.

En este capítulo se tratan los distintos robots valorados para este proyecto, sus características y las razones por las que se ha elegido. Posteriormente, se realiza una caracterización técnica del robot seleccionado, los elementos que lo componen, sus principales propiedades y los elementos de los que consta la célula robotizada.

En el mercado existen gran variedad de robots tipo Delta, pero su selección se realiza en base a las especificaciones de trabajo, al proceso que se quiere automatizar y las características de producción deseadas. Con estos datos, los principales parámetros en los que se basa la selección del robot son:

- Número de grados de libertad. El número total de grados de libertad de un robot está dado por la suma de GDL de las articulaciones que lo componen.
- Espacio de trabajo. Es el conjunto de puntos del espacio accesibles al punto terminal, que depende de la configuración geométrica del manipulador.
- Capacidad de posicionamiento del punto terminal. Se concreta en tres magnitudes fundamentales: resolución espacial, precisión y repetibilidad, que miden el grado de exactitud en la realización de los movimientos de un manipulador al realizar una tarea programada.
- Capacidad de carga. Es el peso que puede transportar el elemento terminal del manipulador.
- Velocidad. Es la máxima velocidad que alcanzan el TCP y las articulaciones.
- Sistema de impulsión. Está relacionado con la capacidad de carga y la velocidad, donde si se quieren movimientos rápidos, el sistema de impulsión adecuado será el eléctrico, si se quiere mover cargas pesadas, el más adecuado es el hidráulico y para sistemas en las que el peso y la velocidad no sea un problema, el mejor sistema es el neumático que se suele utilizar para controlar las pinzas de los robots.

Atendiendo a estas características, los robots paralelos candidatos valorados para esta aplicación son:

- ABB IRB 360 1.
- Yaskawa MPP3S.
- Fanuc M-3iA/6A.
- Omron CR_UGD4MINI-NR.

4.1. ABB IRB 360 1.

En la figura 4.1 se muestra el IRB 360, uno de los robots industriales más rápido del mundo que ofrece una alta fiabilidad y precisión. Es un robot FlexPicker de ABB que ha sido el líder en tecnología robótica de pick and place de alta velocidad y última generación durante los últimos 20 años [27].

Figura 4.1. Robot de ABB IRB 360 FlexPicker.

ABB IRB 360 FlexPicker es capaz de realizar aplicaciones de preparación de pedidos muy rápidas y está optimizado para aplicaciones de empaquetado. El robot tiene un rendimiento de movimiento sobresaliente con tiempos de ciclo muy cortos y una precisión muy elevada.

El modelo escogido de este robot soporta cargas útiles de hasta 1kg con alcances de hasta 1130mm. Además, puede ser de tres o cuatro ejes, permitiendo este último la rotación de la pinza. Sus tiempos de ciclo varían desde 0.3 segundos a 0.36 segundos para cargas que van desde 0.1kg a 1kg.

4.2. Yaskawa MPP3S.

El robot Yaskawa MPP3S [28] (ver Figura 4.2) es un robot diseñado para operaciones de pick and place que permite realizar envasados de alta velocidad de hasta 230 ciclos por minuto.

Figura 4.2. Robot Yaskawa MPP3S.

El robot Yaskawa MPP3S consta de cuatro ejes (con rotación de la pinza) de alta velocidad, es capaz de mover cargas de hasta 3 kg a altas velocidades. Cuenta con 150 ciclos por minuto con una carga útil de 3 kg y 185 ciclos por minuto con una carga útil de 1 kg.

Además, debido a su reducido tamaño (diámetro inferior a 700 mm) permite ser implementado en instalaciones de alta densidad. Este robot realiza una carrera útil de 300 mm con un rango de trabajo de 800 mm y una repetibilidad de 0.1mm.

4.3. Fanuc M-3iA/6A.

El robot Fanuc M-3iA/6A (ver Figura 4.3) consta de seis ejes de alta velocidad perfecto para tareas de picking, embalaje y ensamblaje [29]. Su muñeca de tres ejes permite girar las piezas y herramientas de acuerdo con los requisitos de las aplicaciones de ensamblaje.

Figura 4.3. Robot Fanuc M-3iA/6A.

Cuenta con un alcance máximo de 1350mm y puede transportar una carga útil de hasta 6 kg con una repetibilidad de 0.1mm.

4.4. Omron CR_UGD4MINI-NR.

El robot CR_UGD4MINI-NR (ver Figura 4.4) de OMRON [30], es un robot de 3 ejes (sin rotación de la pinza) de alta velocidad, capaz de mover cargas de hasta 1 kg a velocidades altas y hasta 200 ciclos por minuto con una carga útil de hasta 1 kg. Debido a sus pequeñas dimensiones cuenta con un rango de trabajo de 500 mm, una repetibilidad de 0.2mm y una protección IP65.

Figura 4.4. Robot tipo Delta de Omron CR-UGD4MINI-NR.

Para este proyecto se ha seleccionado este robot debido principalmente a sus pequeñas dimensiones, ya que, el criterio de selección principal, es un robot cuya área de trabajo no supere los 400mm y tenga una repetibilidad menor de 0.2mm a altas velocidades para poder alcanzar los objetos con una buena precisión.

Además, las piezas con las que va a trabajar en ningún caso superarán los 50 gramos y tampoco se necesita la rotación de la pinza como en modelos anteriores, por lo que su tamaño y sencillez colocan al robot CR_UGD4MINI-NR de OMRON como el mejor candidato para esta tarea.

4.5. Caracterización técnica del robot.

En este apartado, se procede a exponer las características técnicas del robot seleccionado. El robot CR_UGD4MINI-NR Delta es un robot de alta velocidad constituido por fibra de carbono y unos servodrivers que pueden ser utilizados en las aplicaciones de pick and place más exigentes.

Está diseñado como un sistema cinemático Delta cuyas características principales son:

- Muy bajo mantenimiento.
- 3 grados de libertad.
- Diseño compacto.
- Nivel de ruido bajo <68 dB.
- Rango de trabajo de hasta 500 mm (175mm en el eje Z).
- Carga máxima de 1 kg.

La placa base del robot lleva una etiqueta de identificación con los datos importantes, entre los cuales se encuentran:

- Tipo de robot.
- Peso total del robot.
- Año de producción.
- Número de serie, importante para pedir repuestos.

El robot consta de tres ejes colocados radialmente que dan al TCP libertad para moverse en tres direcciones, ‘X’, ‘Y’ y ‘Z’. Los brazos primarios del robot están construidos con aluminio anodizado y montados directamente en una caja de cambios de dos etapas para garantizar una alta rigidez.

Los brazos secundarios están montados con cojinetes de rótula de acero inoxidable a los brazos primarios y el TCP, esto garantiza un bajo desgaste con lo que el mantenimiento es mínimo. A continuación, en la figura 4.5 se muestran las partes del robot.

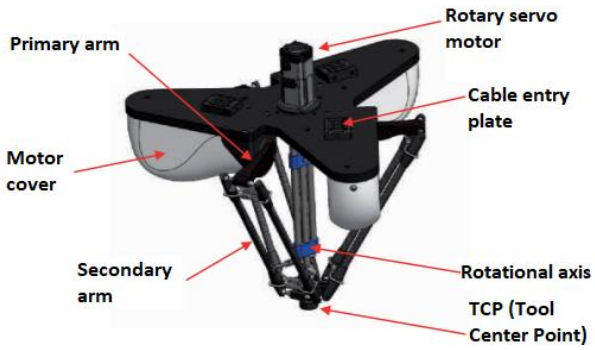


Figura 4.5. Partes del robot Delta CR-UGD4MINI-NR.

En la tabla 4.1 se recogen las especificaciones del rendimiento.

Robot	CR_UGD4MINI
Rango de trabajo nominal	Ø 500 x 155 mm
Carga útil (máxima)	1000g
Acciones Pick and Place	max 200 p/min
Repetibilidad de posición X, Y, Z	+/- 0.2 mm

Tabla 4.1. Especificaciones del rendimiento del robot.

Por otro lado, en cuanto a las velocidades de la ruta del robot para aplicaciones de pick and place, se muestra la tabla 4.2 asociada a la figura 4.6 de la ruta del robot.

Esta figura representa una ruta Pick and place del robot y dependiendo de la carga, el tiempo de ciclo para recorrer dicha ruta varia como muestra la tabla 4.2.

Ruta Z1 x Y x Z2	Carga útil	Tiempo del ciclo
25 x 305 x 25	0.1 kg	0.30 s
25 x 305 x 25	0.5 kg	0.40 s
25 x 305 x 25	1 kg	0.50 s

Tabla 4.2. Velocidades Pick and Place del robot.

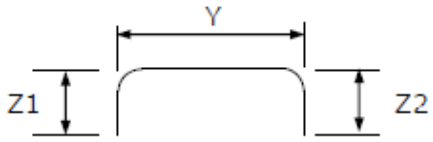


Figura 4.6. Ruta del robot.

Para más información sobre las características técnicas del robot CR-UGD4MINI-NR [31], [32] consultar cualquiera de los dos documentos técnicos de la bibliografía.

4.6. Cinemática del robot.

En este apartado se expone la cinemática del robot con el que se va a trabajar. Como se muestra en la figura 4.7, la geometría de este robot Delta, está formada por la parte fija (superior), tres brazos, uno con cada motor, tres brazos móviles y la conexión de los tres brazos móviles. La rotación de estos tres motores permite el movimiento en las direcciones ‘X’, ‘Y’ y ‘Z’.

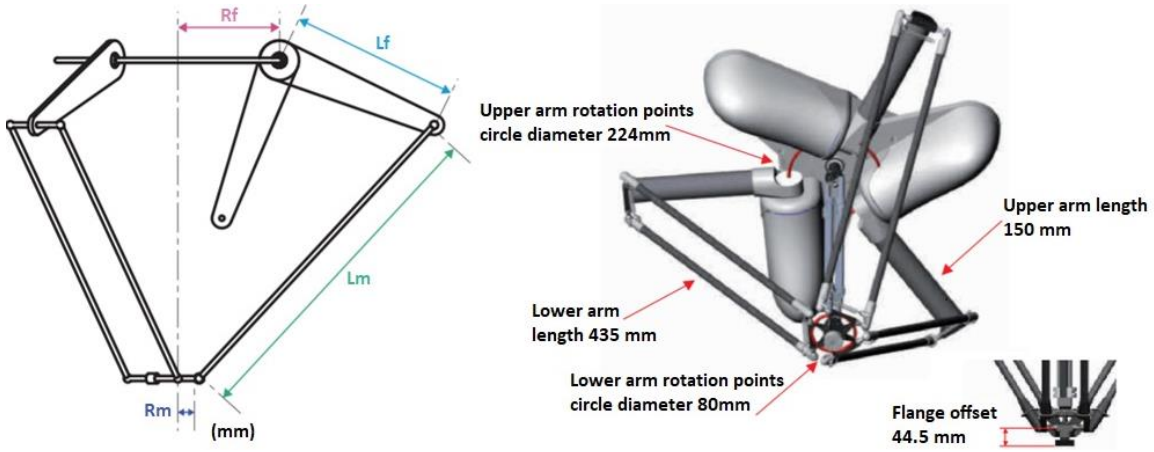


Figura 4.7. Parámetros cinemáticos del robot.

Donde las variables de la figura anterior hacen referencia a:

- Rf = 112 mm. Distancia (radio) desde el centro del marco fijo hasta el motor del eje.
- Rm = 40 mm. Distancia (radio) desde el centro del marco móvil hasta el punto de conexión del enlace 2.
- Lf = 150 mm. Longitud del enlace 1.
- Lm = 435 mm. Longitud del enlace 2.

4.7. Espacio de trabajo.

En este apartado, se trata el espacio de trabajo del robot CR-UGD4MINI-NR, el cual, utiliza tres servomotores paralelamente para poder manejar la posición del extremo del robot. Dicho extremo se puede mover en un volumen determinado, denominado espacio de trabajo.

Este volumen de trabajo es un cilindro y un cono cortado, como se muestra en la figura 4.8. La estructura que encierra el robot tiene que disponer de un espacio libre superior a estos valores.

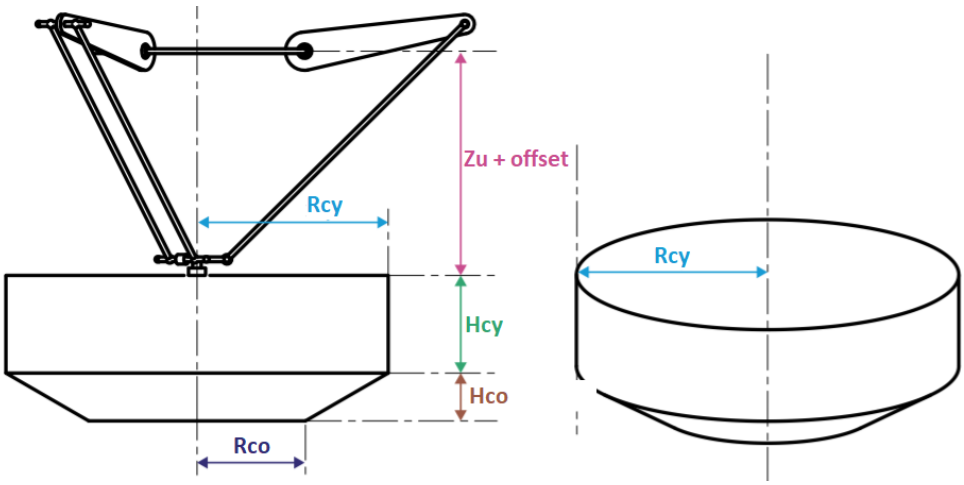


Figura 4.8. Espacio de trabajo del robot.

Donde las variables de la anterior figura hacen referencia a:

- Desplazamiento Zu +offset. -372.5 mm sin eje de rotación. Distancia desde la posición de origen del eje Z hasta la brida de la herramienta.
- Rcy. 250 mm Radio del cilindro.
- Hcy. 155 mm sin eje de rotación. Altura del cilindro.
- Rco. 172 mm sin eje de rotación. Radio del cono troncocónico de la parte inferior.
- Hco. 45 mm de altura del cono troncocónico.

4.8. Limitaciones Software.

En esta sección se muestran algunas de las limitaciones software que tiene este robot, entre las cuales, las más importantes son la amplitud máxima del movimiento que pueden alcanzar las articulaciones de los motores. Su rango de trabajo, va desde -40º hasta 90º y se refieren a los ángulos máximos y mínimos de cada cadena serie o pata.

Los límites de software para el robot CR-UGD4MINI-NR se muestran en la figura 4.9.

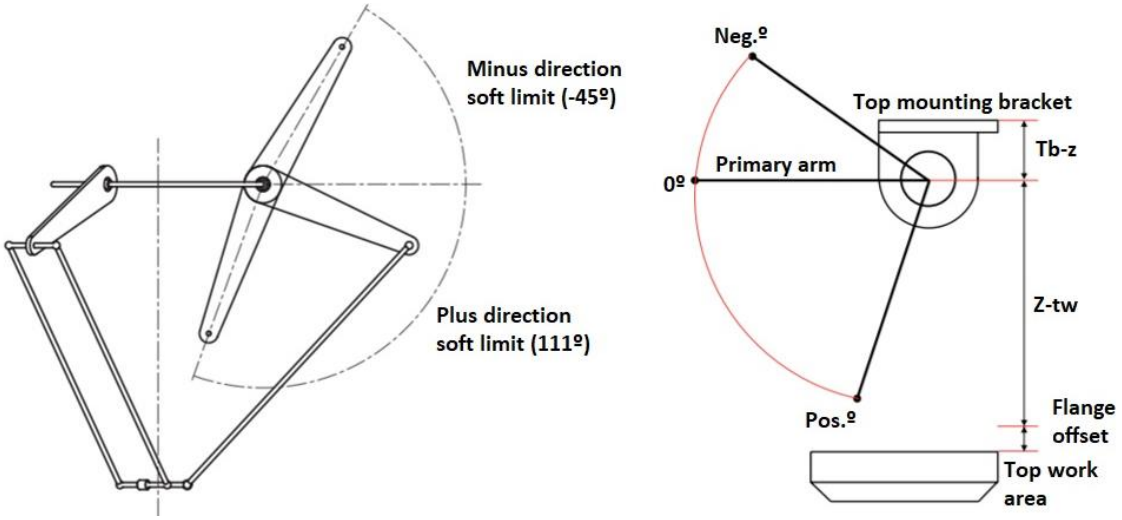


Figura 4.9. Limitaciones software del robot.

4.9. Estructura externa de la célula robotizada.

La célula robotizada consta de una estructura de seguridad que proporciona al robot Delta sujeción. Consta de dos zonas, una para colocar los dispositivos de control (en la parte inferior) y otra en la que se plantea el escenario y donde se pueden realizar sus operaciones de pick and place.

La estructura tiene forma de prisma rectangular y está formada por perfiles de aluminio con una sección de 80 X 80 mm^2.

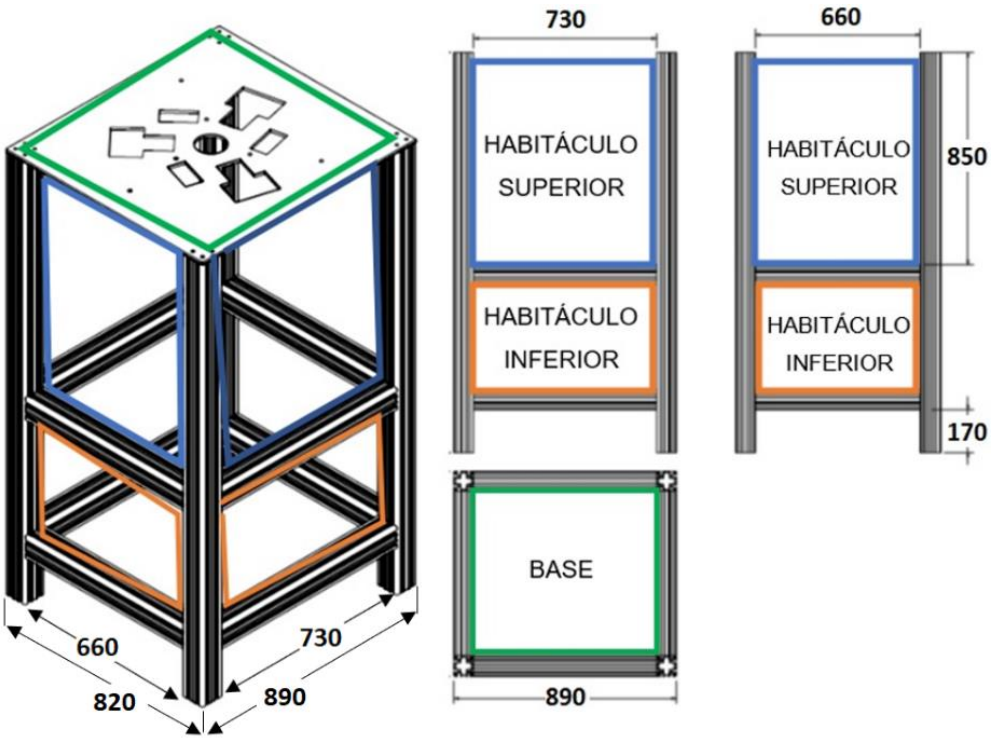


Figura 4.10. Estructura y dimensiones de la célula robotizada.

La célula (ver Figura 4.10) tiene las siguientes dimensiones y está dividida de la siguiente manera:

- Estructura de la célula. El tamaño de la estructura completa es de 1600 mm de alto y 820x890 mm, en el que incluye la zona de trabajo (superior) y el cuadro eléctrico (inferior).
- Habitáculo superior. Donde se encuentra la zona de trabajo y tiene unas dimensiones de 850mm de alto y 820x890 mm de base, en el que se encuentra el robot en la parte superior y los elementos para desarrollar la aplicación sobre la base.
- Habitáculo inferior. Que contiene el conexionado de todos los dispositivos del robot y tiene unas dimensiones de 500mm de alto y 820x890 mm de base. Dentro de se encuentran todos los elementos necesarios para el funcionamiento del robot Delta y sus dispositivos.

4.10. Cuadro eléctrico.

En esta sección, se muestra el cableado de la célula junto con los elementos de protección y control de los que dispone. El cuadro eléctrico es la parte de la instalación eléctrica en la que se encuentran los elementos de protección y control de la energía necesarios para el correcto funcionamiento del robot Delta. Este robot tiene que estar conectado a una red eléctrica de 230 V y 50 Hz con el esquema que aparece en la figura 4.11.

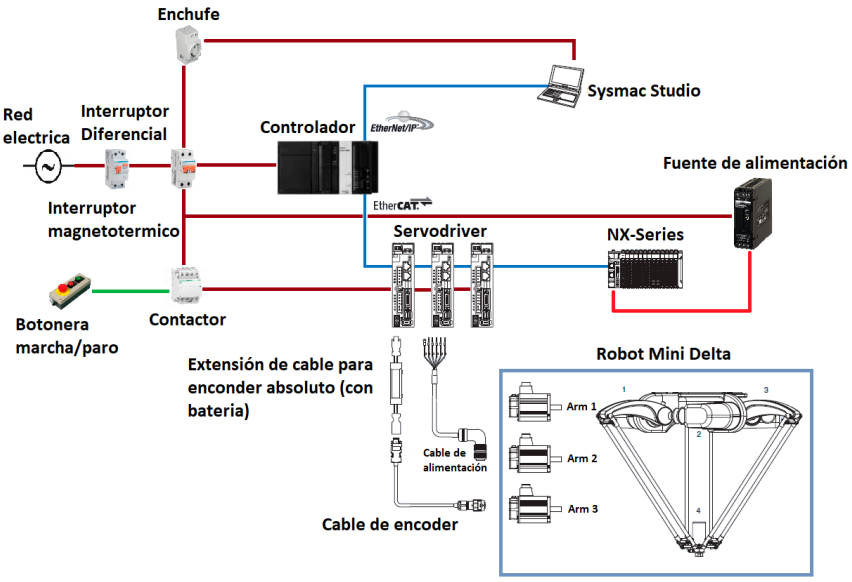


Figura 4.11. Diagrama de conexionado de los componentes del robot Delta.

4.10.1. Alimentación y protecciones.

En este subapartado se exponen los elementos de alimentación, seguridad y protección de los que dispone la célula robotizada en su habitáculo inferior, donde se pueden observar los siguientes componentes:

- Interruptor diferencial. El interruptor diferencial (ID) es el elemento encargado de detectar fugas de corriente dentro de una instalación eléctrica. Se trata de un interruptor imprescindible para salvaguardar al robot de daños provocados por cortocircuitos, así como para proteger a las personas ante descargas eléctricas.

  En el momento en el que el ID detecta una fuga de corriente, el interruptor baja y provoca temporalmente un corte en el suministro. El interruptor diferencial utilizado es de 30mA de corriente de fuga, y de 25A de corriente nominal (ver Figura 4.12).

Figura 4.12. Interruptor diferencial clase A.

- Interruptor magnetotérmico. Es un dispositivo eléctrico que interrumpe la corriente en un circuito cuando supera cierto valor umbral. Se emplea para proteger la instalación eléctrica del robot de cortocircuitos y sobrecargas, evitando daños en los elementos que hay conectados.

  El interruptor magnetotérmico que se utiliza es uno de uso industrial de 25A de corriente nominal como el que muestra la figura 4.13.

Figura 4.13. Interruptor magnetotérmico.

- Contactor de potencia. El contactor (ver Figura 4.14) es un aparato eléctrico de mando a distancia, que puede cerrar o abrir circuitos, ya sea en vacío o en carga. Es la pieza clave del automatismo en el motor eléctrico.

  Para la instalación se necesita un contactor de potencia para permitir el paso de corriente a los servomotores, y de esta manera, parar/activar los servomotores mediante la botonera.

Figura 4.14. Contactor de potencia.

4.10.2. Dispositivos de mando, control y seguridad.

En este subapartado, se exponen los elementos de mando, control y protección que dispone la célula robotizada, que se encuentran fuera del chasis:

- Botonera de emergencia. La función de la botonera de parada de emergencia sirve para prevenir situaciones que puedan poner en peligro a las personas y para evitar daños en la máquina. El robot cuenta con un botón para realizar una parada de emergencia como la que se muestra en la figura 4.15.

Figura 4.15. Botonera de emergencia.

- Botonera de activación. Una botonera de activación permite al usuario cambiar un ajuste entre dos estados. El robot cuenta con dos botones para activar y desactivar la alimentación de la red hacia los motores del robot. La figura 4.16 muestra la botonera de activación instalada en el lateral de la célula robotizada.

Figura 4.16. Botonera de activación.

- Botones auxiliares programables. El robot cuenta con un panel de tres luces led junto a un selector de tres posiciones (ver Figura 4.17), cuyas tres posiciones pueden ser personalizadas por el programador para realizar distintas acciones. La primera posición se activa colocando el selector apuntando a la parte superior izquierda, y lanza el programa de inicialización del robot.

  La segunda posición se activa colocando el selector en posición horizontal y hace que el robot se encuentre en pausa. Por último, la tercera posición se activa colocando el selector apuntando a la parte inferior izquierda y permite activar el programa de movimiento del robot.

Figura 4.17. Botonera auxiliar programable.

- Circuito de seguridad. El armario del robot cuenta con dos puertas de seguridad, una superior y otra inferior, las cuales tienen instaladas unos sensores (ver Figura 4.18) para detectar cuando se encuentran abiertas o cerradas, de manera que el robot se pare de forma inmediata cuando estas se abran. De esta forma, se garantiza la seguridad del operario evitando posibles accidentes.

Figura 4.18. Circuito de seguridad.

4.11. Dispositivos del robot.

En el espacio inferior de la célula robotizada se encuentran los dispositivos de los que se compone el robot para que este pueda funcionar correctamente. En los siguientes subapartados se exponen las características y el desempeño de cada uno de ellos.

4.11.1. Compresor.

El compresor se encarga de proporcionar aire comprimido al circuito neumático de la pinza para que esta pueda abrirse y cerrarse. El compresor utilizado es un Sparmax TC-610H Plus Air (ver Figura 4.19) con un flujo de aire de entre 23 y 28 litros/minuto.

Para más información sobre el compresor Sparmax TC-610H Plus Air consultar el documento técnico de la bibliografía [33].

Figura 4.19. Compresor Sparmax TC-610H Plus Air

4.11.2. Controlador.

La controladora utilizada encargada de gobernar esta célula robotizada es un PLC modelo NJ501-4300 de Omron (ver Figura 4.20). Se trata del controlador donde se van a cargar todos los programas permiten el correcto funcionamiento del robot. El modelo NJ 501-4300 tiene las siguientes características:

- El número máximo unidades que se pueden configurar es de 40.
- La capacidad máxima de E/S es de 2560 esclavos en EtherCAT.
- Tiene una capacidad de programación de 20MB
- Puede controlar el movimiento de 16 ejes y hasta 5 robots.
- Tiene un consumo de 5Vdc y 1.9A.

Figura 4.20. Controlador NJ501-4300 de Omron.

Para más información sobre el controlador NJ501-4300 consultar el documento técnico de la bibliografía [34] o acceder al anexo (*VI.I. Controlador*.).

4.11.3. Fuente de alimentación del controlador.

Para alimentar al controlador se necesita una fuente de alimentación. Para ello, se utiliza el modelo NJ-PA3001 (ver Figura 4.21) que cuenta con las siguientes características:

- La tensión de alimentación es de 100 a 240 voltios de alterna.
- Frecuencia de trabajo 50/60Hz.
- Puede alimentar a 24Vdc y 1A, o 5Vdc y 6A.
- Consumo total de 30W.
- Peso 470 gr.

Figura 4.21. Fuente de alimentación NJ-PA3001.

Para más información sobre la fuente de alimentación del PLC NJ-PA3001 consultar el documento técnico de la bibliografía [35] o acceder al anexo (*VI.II. Fuente de alimentación del controlador.).*

4.11.4. Servodrivers y servomotores.

Junto al PLC se encuentran unidos los tres servodrivers, uno por cada servomotor que contiene el robot delta. Su función es controlar la velocidad y posición del servomotor. Los servodrivers que utiliza el robot Mini Delta son el R88D-KN04H-ECT (ver Figura 4.22). El modelo R88D-KN04H-ECT tiene las siguientes características:

- Una tensión monofásica de 200 a 240 Vac a 50/60 Hz.
- Una corriente nominal de 4.1/2.4A.
- El circuito de control tiene la misma tensión que el circuito principal.
- La capacidad máxima que se aplica al servomotor es de 400W.
- Tiene un peso de 1100 gr.

Figura 4.22. Servodriver modelo R88D-KN04H-ECT.

Para más información sobre los servodriver R88D-KN04H-ECT consultar el documento técnico de la bibliografía [36] o acceder al anexo (*VI.III. Servodrivers.*).

Los servomotores que utiliza el robot Mini Delta son el R88M-K40030T-BS2 (ver Figura 4.23) y tienen las siguientes características:

- Un consumo nominal de 400W.
- Una tensión de 24Vdc (±10%) y consumo de 0.36A.
- Una corriente nominal de 2.4A (eficaces).
- Un par nominal de 1.3N.m.
- Una velocidad de rotación nominal de 3000rpm y máxima de 6000rpm.
- Tiene un encoder absoluto.

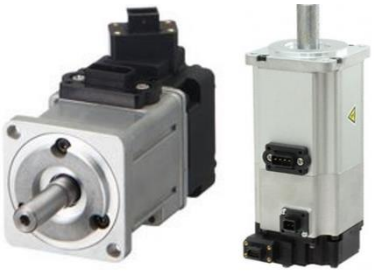


Figura 4.23. Servomotores del robot Mini Delta R88M-K40030T-BS2.

Para más información sobre los servomotores R88M-K40030T-BS2 consultar el documento técnico de la bibliografía [37].

4.11.5. Acoplador EtherCAT.

El acoplador detecta todos los módulos de E/S conectados y crea una imagen de proceso local. Esta imagen de proceso puede incluir una disposición mixta de módulos analógicos y digitales. Es el enlace entre la red de control de la máquina EtherCAT y el NX-serie de E/S.

EtherCAT es un sistema Ethernet en tiempo real diseñado para aplicaciones de automatización industrial que permite sustituir la costosa topografía de estrella Ethernet por una estructura más simple.

También realiza la función alimentar las unidades NX-serie de E/S. Se utiliza el modelo NX-ECC201 (ver Figura 4.24) con las siguientes características:

- El ciclo de comunicación es de 0.25 a 4ms.
- Admite la conexión de un máximo de 63 unidades NX.
- El suministro de alimentación que necesita el acoplador EtherCAT es de 24Vdc y 0.75A.
- Potencia de salida es de 10W a 5.8V.
- La fuente de alimentación de E/S es de 24Vdc y 4A.

Figura 4.24. Acoplador EtherCAT NX-ECC201.

Para más información sobre el acoplador EtherCAT NX-ECC201 consultar el documento técnico de la bibliografía [38] o acceder al anexo (*VI.IV. Acoplador EtherCAT*.).

4.11.6. Entradas y salidas digitales.

La función de las entradas digitales es leer los datos de una señal digital. Para la aplicación a desarrollar se necesitan un mínimo de 10 entradas, por lo que se utilizan dos bloques del modelo NX-ID4342 (ver Figura 4.25) que tiene las siguientes características:

- Tiene 8 puntos de entrada.
- La tensión nominal de entrada tiene que ser de 24Vdc y de al menos 3.5mA.
- El tiempo de respuesta de ON/OFF es como máximo de 0.4ms.

Figura 4.25. Entradas digitales NX-ID4342.

Por otro lado, la función de las salidas digitales es escribir los datos en una señal digital. Para la aplicación a desarrollar se necesitan de un mínimo de 5 salidas, por lo que se utilizan dos (aunque con una sería suficiente) del modelo NX-OD4256 (ver Figura 4.26) con las siguientes características:

- Tiene 8 puntos de entrada.
- La tensión nominal de salida es de 24Vdc y el valor máximo de la corriente de carga de 0.5A por punto.
- El tiempo de respuesta de ON/OFF es como máximo de 1ms.

Figura 4.26. Salidas digitales NX-OD4256.

Para más información sobre el NX-ID4342 y NX-OD4256 consultar el documento técnico de la bibliografía [39].

4.11.7. Fuente de alimentación.

Se necesita una fuente de 24Vdc para alimentar el acoplador EtherCAT y las E/S digitales. En el peor de los casos se necesita que suministre hasta 8A, ya que dispone de dos salidas digitales con un consumo máximo de 4A cada una, pero no se van a utilizar para esta aplicación todas las salidas y el consumo baja, por lo que con 2,5A es suficiente.

La fuente de alimentación utilizada es el modelo S8VK-G06024 (ver Figura 4.27) que tiene las siguientes características:

- La tensión de alimentación es de 100 a 240Vac 50/60Hz y 1.3A.
- Proporciona una tensión de 24Vdc con 2.5A, y un consumo de 60W.
- Tiene una eficiencia del 88%.

Figura 4.27. Fuente de alimentación S8VK-G06024.

Para más información sobre la fuente de alimentación S8VK-G06024 consultar el documento técnico de la bibliografía [40].

4.12. Nuevos dispositivos añadidos.

En este apartado se exponen los nuevos dispositivos que se han añadido a la célula robotizada para abordar el objetivo definido en este TFM. Las células suelen estar equipadas con sistemas de visión artificial 3D que hacen que los robots puedan tomar decisiones en función del entorno que perciben haciendo que sean totalmente autónomos porque son capaces de distinguir objetos y formas presentes en el escenario y que interaccionarán con el robot. De esta forma, permitirá al robot leer códigos y localizar objetos de cualquier tamaño, forma, posición o color.

Por tanto, las ventajas más relevantes de una célula robotizada son:

- Optimización de los tiempos de ciclo.
- Aumento de la productividad y de la repetibilidad de la calidad del producto.
- Amplia gama de aplicaciones industriales.
- Flexibilidad, reprogramación y convertibilidad.
- Alta autonomía.
- Salud y seguridad para los operarios y el medio ambiente.
- Rápido retorno de la inversión.

Para este TFM se ha instalado una webcam que permite implementar un sistema de visión artificial. Esta webcam se conecta al PC (mediante un cable USB) que se encarga del procesamiento de la imagen.

Por otro lado, para obtener unos niveles de iluminación constantes y un mejor procesado de la imagen se necesita una buena iluminación, para lo cual se va a utilizar unas tiras led blancas que funcionan a 5Vdc y 100mA.

Para poder implementar el programa de captura y procesar la imagen, analizarla y extraer información de la misma se necesita un PC que se ubica en el habitáculo inferior. Por otro lado, para que el usuario pueda interactuar con la interfaz se ha añadido un monitor táctil de LG con una resolución de 720p como el que muestra la figura 4.28.

Figura 4.28. Monitor táctil de 17" de LG modelo 17MB15T-B-.

Capítulo 5:
ANÁLISIS DE ALTERNATIVAS

5. Análisis de alternativas.

En este capítulo se van a exponer cada una de las alternativas valoradas para las tareas a realizar junto con las razones de selección de la alternativa elegida.

5.1. Selección del programa de diseño de piezas.

Para el diseño de las piezas 3D se han valorado principalmente dos herramientas de diseño, que son:

- Tinkercad.
- Fusion 360.

A continuación, se va a exponer las características de cada una de ellas y las razones de la selección.

5.1.1. Tinkercad.

Tinkercad [41] es un software gratuito de diseño y modelado 3D muy sencillo de usar, pero con una cantidad de herramientas limitada para el diseño de piezas complejas. Permite diseñar cualquier objeto con volumen de forma intuitiva y después llevarlo a la realidad mediante una impresora 3D.

Figura 5.1. Logo de la aplicación de Tinkercad.

Entre las principales ventajas de esta herramienta destaca que es online y gratuita, por lo que se puede usar desde cualquier dispositivo con acceso a internet y permite compartir diseños entre diferentes usuarios. Además, cuenta con una gran cantidad de tutoriales explicando paso a paso las funciones básicas del programa.

5.1.2. Fusion 360.

Fusion 360 [42] es un software CAD de circuitos impresos de modelado 3D basado en la nube para el diseño y la manufactura de productos.

Tiene una gran libertad de diseño y es perfecto para la impresión 3D ya que permite el diseño de piezas con gran precisión, por esta razón, se ha convertido en el software de diseño 3D

de moda. También permite el diseño colaborativo de piezas, compartiendo proyecto con otros usuarios en línea.

Además, cuenta con múltiples herramientas de diseño que permite reducir el coste de los prototipos y obtener una solución completa.

Figura 5.2. Logo de la herramienta de diseño 3D.

Teniendo en cuenta estas características y atendiendo a la tabla 5.1 se ha decidido utilizar la herramienta de Fusion 360 de Autodesk para el diseño de las piezas 3D que van a componer el entorno de trabajo del robot.

	FUSION 360	TINKERCAD
Fácil de aprender	NO	SI
Pensado para el ámbito educativo	NO	SI
Pensado para el ámbito profesional	SI	NO
Limitado por catálogo de formas	NO	SI
Potentes herramientas de modelado	SI	NO
El árbol de historial facilita la edición	SI	NO

Tabla 5.1. Comparación de características entre Tinkercad y Fusion 360.

5.2. Selección del programa de control del robot.

Para el diseño del programa de control del robot con el PLC que se dispone, solo se pueden utilizar dos herramientas:

- CX-Programmer.
- Sysmac Studio.

A continuación, se exponen las características de cada uno de ellos y las razones de la selección.

5.2.1. CX-Programmer.

CX-Programmer [43] es el programador de los autómatas programables de Omron. Permite programar todos los modelos, desde micro-PLC hasta la serie CS de gama alta.

Esta herramienta permite construir complejos sistemas de múltiples dispositivos aplicando lenguajes en diagrama de relés y/o de listas de instrucciones. Además de un entorno de programación exhaustivo, CX-Programmer proporciona todas las herramientas necesarias para proyectar, probar y depurar cualquier sistema de automatización.

Figura 5.3. Logo de CX-Programmer.

5.2.2. Sysmac Studio.

Sysmac Studio [44] es un entorno de desarrollo integrado que permite aumentar la productividad de un proceso, ya que se trata de un entorno de desarrollo integrado (IDE) que integra lógica, movimientos, robótica, HMI, visión, detección, seguridad y simulación en 3D en una única plataforma.

Figura 5.4. Logo Sysmac Studio.

Esta herramienta es la nueva versión de CX-Programmer, ya que las nuevas series NJ/NX de Omron se han diseñado a partir del estándar IEC61131-3 que cada vez es más demandado. Los lenguajes de programación definidos por este estándar son cinco, pero esta herramienta tan solo soporta dos:

- LD lenguaje diagrama de contactos (Ladder).
- ST texto estructurado (variante de Pascal).

Para el desarrollo de esta aplicación, se ha realizado el código con diagrama de bloques de función (LD) y texto estructurado (ST).

La programación basada en IEC61131-3 se apoya básicamente en tres conceptos:

- POU (Program Organization Unit). Son los programas que ha de ejecutar el controlador en función del estado del sistema y las condiciones requeridas.
- Tareas (Tasks). Los POU son agregados a tareas que se ejecutan de forma periódica o por eventos.
- Variables y tipos de datos. Son los identificadores que leen y escriben programas.

Teniendo en cuenta que Sysmac Studio es la nueva versión de CX-Programmer y atendiendo a la tabla 5.2, se ha decidido que la herramienta para el diseño del programa de control del robot sea Sysmac Studio.

	Sysmac Studio	CX-Programmer
Estándar IEC-61131-3	SI	NO
Lenguajes LD y ST	SI	SI
Simulación 3D	SI	NO
Funciones Robótica	SI	NO
HMI	SI	NO

Tabla 5.2. Comparación de características entre Sysmac Studio y CX-Programmer.

5.3. Selección del protocolo de comunicación.

La siguiente tarea a realizar consiste en la selección del protocolo de comunicación más adecuado para este proyecto. Las comunicaciones industriales son los mecanismos que realizan un intercambio de información entre dos o más partes, en los que se busca ejercer cierto control.

Las redes de comunicación industrial pueden utilizarse en los sistemas de control para pasar datos entre los dispositivos de campo y los PLCs, entre diferentes PLCs, o entre los PLCs y PCs. Un protocolo de comunicación son todas las reglas que definen dicho intercambio de datos.

Un sistema de comunicación efectivo y que se adapte a las características de cada proceso resulta esencial para mejorar la productividad, ya que permite un mejor control de los procesos, la vigilancia de la línea de producción y una mejor coordinación entre los procesos.

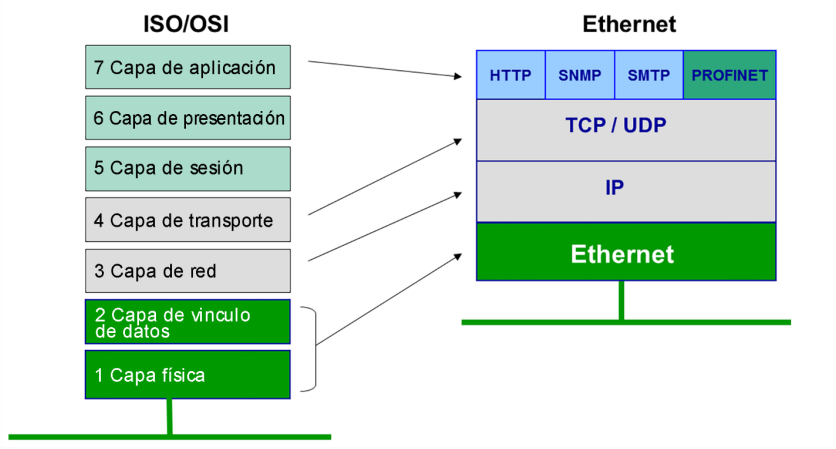


Figura 5.5. Esquema del modelo OSI y del modelo de Ethernet.

Para este TFM hay que implementar una comunicación entre un PLC y un PC, y aunque existen múltiples protocolos de comunicación, es importante tener en cuenta que solo se puede implementar un tipo de comunicación si los dispositivos a comunicar entre sí permiten dicho protocolo.

En este caso, el PLC modelo NJ501-4300 de Omron soporta comunicaciones UDP, TCP y Modbus TCP. Por lo que, a continuación, se va a exponer cada uno de estos protocolos para elegir el que mejor se ajusta a este TFM.

5.3.1. Protocolo de comunicación UDP.

El protocolo UDP (Protocolo de Datagramas de usuario) [45] es uno de los protocolos fundamentales en Internet, permite que las aplicaciones puedan comunicarse con garantías independientemente de las capas inferiores del modelo TCP/IP.

El protocolo UDP permite el envío de datagramas sin necesidad de establecer previamente una conexión, tan solo es necesario tener abierto un socket en el destino para que acepte los datagramas del origen.

UDP es un protocolo no orientado a la conexión, es decir, no ocurre como en TCP donde hay una fase de establecimiento de la conexión, aquí directamente se envían sin establecimiento previo.

Este protocolo no proporciona ningún tipo de control de flujo, si un equipo es más rápido que otro y envía información, es muy posible que se pierda información debido a que colapsará al más lento, y habrá que reenviar la información.

Un detalle importante es que la gestión de reenvío de los datagramas la realiza la capa de transporte, ya que UDP es muy simple y no dispone de mecanismos de control de reenvío de datagramas por haberse perdido.

UDP tampoco proporciona ningún tipo de control de congestión, si hay congestión en la red, se podrían perder paquetes, y no se va a encargar de reenviarlos como sí ocurre con TCP.

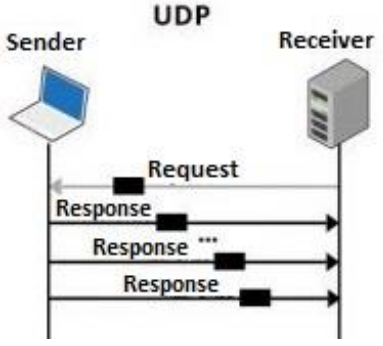


Figura 5.6. Sucesión de la comunicación entre el emisor y el receptor en UDP.

Por tanto, UDP al no disponer de control de congestión, control de flujo ni control de errores, se podría decir que UDP es un protocolo no fiable.

Además, tampoco proporciona orden en los datagramas enviados, ni información si un datagrama ha llegado correctamente, ya que no hay confirmación ni de entrega ni de recepción. Cualquier tipo de garantías para la transmisión de la información deben ser implementadas en capas superiores.

5.3.2. Protocolo de comunicación Modbus TCP.

Modbus [46] es un protocolo de comunicación abierto, utilizado para transmitir información a través de redes en serie entre dispositivos electrónicos. El dispositivo que solicita la información se llama maestro Modbus y los dispositivos que suministran la información son los esclavos Modbus.

Esto significa que un dispositivo esclavo no puede ofrecer información, debe esperar a que se le pida. El maestro escribirá datos en los registros de un dispositivo esclavo y leerá los datos de los registros de un dispositivo esclavo.

Por lo tanto, en una red Modbus estándar, hay un maestro y hasta 247 esclavos, cada uno con una dirección de esclavo única de 1 a 247. El maestro también puede escribir información a los esclavos.

Existen varios tipos de versiones en el protocolo Modbus para el puerto serie y Ethernet, que se utilizan para atender las necesidades específicas de los sistemas de automatización industrial en las empresas. Las más comunes son:

- Modbus RTU.
- Modbus TCP.
- Modbus ASCII.
- Modbus Plus.

Modbus TCP se introdujo para aprovechar las infraestructuras LAN actuales. A su vez, aumentó el número de unidades que podían conectarse a la misma red. Este sistema engloba los bloques de datos de solicitud y respuesta del Modbus RTU en un bloque TCP transmitido a través de redes estándar de Ethernet.

En este contexto, el esclavo se convierte en el servidor y el maestro en el cliente. Puede haber más de un cliente que obtenga datos de un servidor, es decir, puede haber múltiples maestros, así como múltiples esclavos.

5.3.3. Protocolo de comunicación TCP.

Tanto TCP como UDP son dos protocolos fundamentales para las comunicaciones a través de Internet, ya que estos dos protocolos se sitúan en la capa de transporte del modelo TCP/IP,

y es la primera capa donde origen y destino se comunican directamente, ya que las capas inferiores (capa de red y capa de acceso al medio) no realizan esta función [45].

Debido a que TCP sirve a una gran cantidad de protocolos de la capa de aplicación, es fundamental que los datos (segmentos) lleguen correctamente al destinatario, sin errores y en orden.

Si en la transmisión de los segmentos, se corrompiesen o perdiesen, automáticamente el protocolo TCP inicia la retransmisión, sin intervención de la capa de aplicación. De esta manera, se garantiza que los datos llegan al destinatario sin errores, ya que este protocolo se encarga de solucionar cualquier tipo de problema.

Si TCP detecta un error, iniciará la retransmisión automáticamente sin que la capa de aplicación tenga que hacer absolutamente nada.

Otra característica muy importante es que la información que viaja desde un origen hasta un destino, llegue en orden, es decir, en el mismo orden que fueron emitidos, ya que el protocolo IP es un protocolo best-effort, hace todo lo que puede para que los paquetes lleguen en orden y correctos, pero no es confiable ya que no garantiza nada.

TCP dispone de una ventana deslizante en el emisor y en el receptor, de tal forma que, si recibimos un segmento que no está en orden, automáticamente «esperará» hasta que llegue el segmento que falta, o si no, pedirá una retransmisión únicamente del segmento que falte.

Con cada segmento recibido por el receptor, se enviará un ACK indicando al emisor que todo está llegando correctamente, no obstante, en la vida real las implementaciones de TCP permiten que se envíe un ACK para confirmar la recepción de varios segmentos simultáneamente, con el objetivo de no saturar la red de tantas confirmaciones.

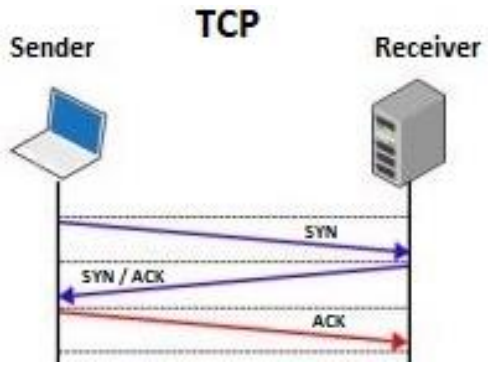


Figura 5.7. Sucesión de la comunicación entre el emisor y el receptor en TCP.

TCP también dispone de control de congestión, esto permite que no se pierdan paquetes en Internet porque haya congestión en los routers. Si el router no es capaz de procesar o reenviar paquetes al ritmo que los recibe, el propio router los descartará y se perderá, ya que su buffer se llenará.

En resumen, este protocolo de comunicación permite realizar una comunicación en la que se asegura que todos los paquetes llegan correctamente al destino y se recibe una confirmación por parte del receptor, lo cual es fundamental para la aplicación que se quiere implementar.

Atendiendo a estas características y la tabla 5.3, el protocolo de comunicación para el intercambio de información entre el PC y el PLC es TCP.

	UDP	**TCP**	**Modbus TCP**
Orientado a la conexión	NO	SI	SI
Velocidad	Rápido	Rápido (más lento que UDP)	Rápido (más lento que UDP)
Fiabilidad de datos	NO	SI	SI
Confirmación de recepción	NO	SI	SI
Control de flujo	NO	SI	SI
Comprobación de errores	NO	SI	SI
Dificultad	BAJA	BAJA	MEDIA

Tabla 5.3. Comparación de características entre UDP, TCP y Modbus TCP.

5.4. Selección de la webcam.

Para cumplir con el objetivo de ubicación de los cubos e identificación de las letras, se necesita una cámara de alta resolución para poder asegurar que todas las letras de los cubos se van a poder obtener con una resolución suficiente para poder trabajar con ellas.

Entre los diferentes modelos de cámaras del mercado, se han valorado dos principales productos:

- Logitech c270.
- Logitech Brio.

A continuación, se va a exponer las características de ambas y las razones de la selección.

5.4.1. Logitech c270.

La webcam Logitech c270 [47] (ver Figura 5.8) es una webcam de gama media que destaca por su relación calidad-precio.

Figura 5.8. Logitech c270.

Permite grabar con una resolución de 720p en pantalla panorámica, y cuenta con una cámara de 3 megapíxeles que permite realizar fotos con una calidad aceptable a distancias cortas y deficiente a distancias medias. Cuenta con ajuste automático de la imagen para mejorarla en condiciones de iluminación escasa.

Entre los principales inconvenientes de esta cámara destaca que no permite realizar un ajuste de la distancia a la cual se quiere enfocar la imagen, por lo que una parte de la plataforma aparecerá desenfocada y no permitirá realizar una buena identificación.

5.4.2. Logitech Brio.

La webcam modelo Logitech Brio 4k UltraHD [48] (ver Figura 5.9) es una webcam de gama alta que destaca por la buena calidad que se obtiene de sus imágenes.

Figura 5.9. Logitech Brio 4k UltraHD.

Permite grabar con una resolución de hasta 4k a 30fps y cuenta con una cámara de 13MP. Esta webcam permite configurar la cámara para distintas condiciones de iluminación y así poder obtener buenos resultados en condiciones de iluminación cambiante.

Además, se puede configurar la distancia a la que enfoca la cámara, lo cual es muy importante para este TFM, ya que la distancia desde la webcam hasta la plataforma es mucho mayor de la distancia para las que están enfocadas por defecto este tipo de cámaras.

Debido a los constantes cambios en la iluminación que puede sufrir el entorno de trabajo y atendiendo a la tabla 5.4, se va a implementar el sistema de visión artificial con la cámara Logitech Brio.

	Logitech c270	Logitech Brio
Resolución	720p	4K
Cámara	3MP	13MP
Permite configurar la exposición	NO	SI
Permite configurar la distancia de enfoque	NO	SI

Tabla 5.4.Comparación de características entre Logitech c270 y Brio.

5.5. Selección del método de diferenciación de objetos.

En este apartado se tratan los tres principales métodos planteados para la diferenciación de las letras a través del sistema de visión artificial para extraer la información deseada de la imagen. Para este TFM se han valorado tres posibles herramientas:

- Red neuronal.
- Reconocimiento de patrones.
- Extracción de características geométricas.

5.5.1. Red neuronal.

Dentro de las múltiples potencialidades de este complejo y completo sistema, el Deep Learning destaca en los procesos de análisis, reconocimiento y clasificación de imágenes, a partir de un modelo entrenado (visión artificial).

Para esta aplicación las redes neuronales convolucionales (CNN, ver Figura 5.10) permiten detectar un objeto dentro de una imagen, ya que, a diferencia de las redes neuronales convencionales y otros algoritmos de clasificación de imágenes, procesan la información de manera rápida y sencilla [49].

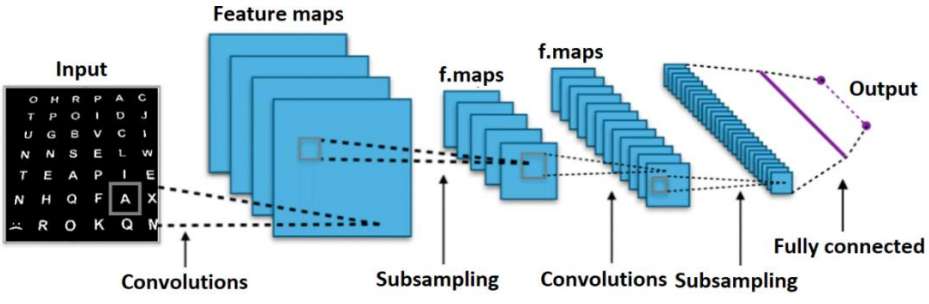


Figura 5.10. Representación de la red neuronal convolucional.

Su sistema de trabajo se basa en la localización de patrones en áreas reducidas de la imagen y no en su conjunto, es decir, detectaría ciertos patrones en las formas de cada una de las letras para poder diferenciarlas.

Para ello, el algoritmo toma un pequeño cuadrado de píxeles (entrada) que pasarán por diversas capas ocultas organizadas para una ejecución analítica gradual, es decir, las primeras capas de la red detectarán características simples, como líneas, curvas, bordes… siendo las capas más profundas capaces de reconocer las formas complejas de cada letra.

En consecuencia, a mayor número de capas mejor será la 'capacidad de visión de la red', y más certera su salida, que muestra como resultado del análisis la probabilidad de que cada letra pertenezca a un tipo de letra, que la red está entrenada para reconocer.

Algo a tener en cuenta es que para esta red se utilizaría un aprendizaje supervisado donde se partiría de un conjunto de datos etiquetado previamente, es decir, se necesitaría una gran cantidad de imágenes etiquetadas para entrenar a la red, lo cual implica mucho tiempo de toma y clasificación de las fotos.

5.5.2. Reconocimiento de patrones.

El reconocimiento exacto de objetos, también conocido como pattern matching, siempre ha sido uno de los principales problemas con los que ha tenido que lidiar la visión artificial y que actualmente sigue en desarrollo.

Los sistemas de reconocimiento de patrones son una herramienta automatizada de visión artificial empleada para tareas de localización e inspección de productos, como por ejemplo la detección de las matrículas de los coches [50].

En comparación con otras herramientas de reconocimiento, el empleo de sistemas de correspondencia de patrones destaca por su complejidad, ya que tiene en cuenta muchas variables que alteran la forma de un objeto y las características que pueden ser reconocidas.

Por tanto, usando este sistema de visión artificial se busca la forma de que las herramientas de reconocimiento de visión actúen acorde a patrones, identificando y clasificando productos según este esquema predeterminado, obviando otras características físicas que pueden alterar la forma, pero no la naturaleza del objeto.

Para comparar cada una de las letras, se utiliza una librería (patrones) de cada una de las letras con distintos tamaños y formas y se comparan una a una hasta encontrar la que más se aproxime. La figura 5.11 muestra un ejemplo de una librería.

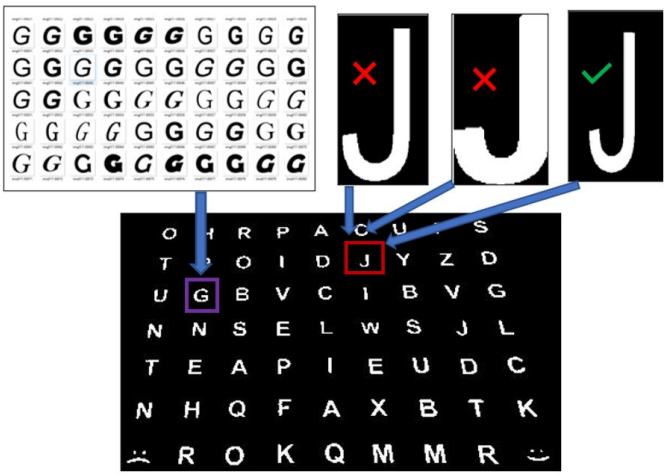


Figura 5.11. Ejemplo de imágenes a comparar con la técnica de pattern machine.

Para entrenar al sistema con este método se necesitan en torno a 1.000 imágenes de letras para que el sistema sea capaz reconocer los patrones de los distintos números y letras de las matrículas.

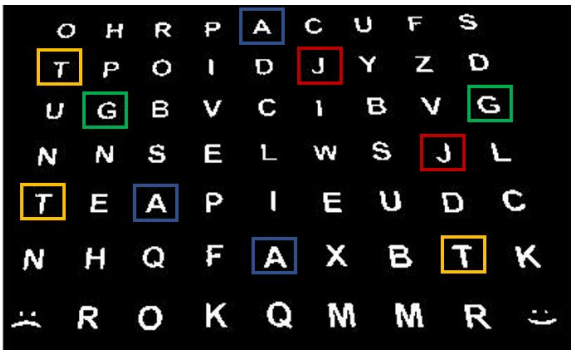


Figura 5.12. Ejemplo de reconocimiento automático de patrones.

Por lo que, al igual que ocurre con el método de la red neuronal, se necesitan una gran cantidad de imágenes de letras etiquetadas para poder hacer las comparaciones y tiene el inconveniente de que el coste computacional es demasiado alto.

5.5.3. Extracción de características geométricas.

Este método consiste en la obtención de distintas características geométricas de cada letra y mediante estas características establecer unas relaciones que permiten diferenciar cada una de estas letras. Entre estas características geométricas, destacan:

- Área.
- Circularidad.
- Anchura.
- Altura.
- Perímetro.

Atendiendo a las características de la tabla 5.5 y teniendo en cuenta que no se dispone de una librería de imágenes, este ha sido el método elegido para diferenciar cada letra.

	Red Neuronal	**Reconocimiento de patrones**	**Características geométricas**
Librería de imágenes	SI	SI	NO
Tiempo de entrenamiento	SI	SI	NO
Complejidad	ALTA	ALTA	MEDIA
Coste computacional	ALTO	ALTO	MEDIO

Tabla 5.5. Comparación entre la red neuronal, el reconocimiento de patrones y las características geométricas.

5.6. Selección de la herramienta para la Interfaz gráfica.

En este apartado se tratan las herramientas planteadas para la implementación de la interfaz gráfica, juntos con las características más relevantes de cada una de ellas y las razones de la aplicación seleccionada.

La Interfaz gráfica de usuario o GUI (Graphic User Interface) es un entorno visual de imágenes y objetos mediante el cual una máquina y un usuario interactúan. El objetivo principal de las GUIs es hacer más sencilla la comunicación entre una máquina/sistema operativo y un usuario.

Actualmente, existen múltiples herramientas para diseñar interfaces, pero en función de los requisitos de la aplicación algunas son más adecuadas que otras. Para este proyecto se ha valorado realizar dicha interfaz en tres plataformas diferentes:

- LabVIEW.
- Matlab App Designer.
- Matlab Guide.

5.6.1. LabVIEW.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) [51] es un ambiente para el desarrollo de aplicaciones de propósito general que permite controlar dispositivos y realizar análisis de datos.

También utiliza un lenguaje de programación gráfica llamado lenguaje G, el cual a través de diagramas de bloques brinda una solución gráfica a un problema de programación.

Figura 5.13. Logo de LabVIEW.

Este software permite visualizar resultados mediante interfaces de usuario de “clic-y-arrastre” y visualizadores de datos integrados. Este software garantiza la compatibilidad con otras herramientas ya que puede interactuar o reutilizar bibliotecas de otro software y lenguajes de fuente abierta como Arduino.

LabVIEW cuenta con dos interfaces; un panel frontal y un diagrama de bloques. Estas cuentan con paletas que contienen los objetos necesarios para implementar y desarrollar tareas.

El principal inconveniente de esta herramienta se encuentra en que presenta una capacidad de procesamiento mucho menor que cualquiera de las otras dos opciones de Matlab.

5.6.2. App Designer.

App Designer [52] es un entorno de desarrollo gráfico de Matlab introducido a partir de la versión R2016a que surge de la evolución del entorno Guide de Matlab. App Designer permite crear apps profesionales arrastrando y colocando los componentes visuales para crear el diseño de la interfaz gráfica de usuario (GUI).

Figura 5.14. Logo del App Designer de Matlab.

La principal ventaja que tiene App Designer sobre el entorno Guide de Matlab aparte de disponer de un nuevo entorno de trabajo reside en que dispone de más herramientas especializadas respecto a su predecesor. Pero las herramientas nuevas implementadas en esta versión no se van a utilizar para este trabajo.

5.6.3. Matlab Guide.

Matlab dispone de una app con interfaces gráficas (Matlab Guide) que permite el control de programas que necesiten un ingreso continuo de datos. Esta herramienta contiene todas las características básicas de todos los programas visuales como Visual Basic o Visual C++.

Figura 5.15. Logo de Matlab Guide.

Atendiendo a la tabla 5.6, Matlab Guide [53] es la herramienta que se ha escogido, ya que cuenta con todas las herramientas necesarias para implementar la interfaz gráfica deseada, al mismo tiempo que permite exportar la interfaz final a una aplicación independiente.

	LabVIEW	**App Designer**	**Matlab GUIDE**
Permite comunicaciones	SI	SI	SI
Potencia computacional	MEDIA	ALTA	ALTA
Herramientas de edición	MEDIA	ALTA	MEDIA

Tabla 5.6. Comparación de características entre LabVIEW, App Designer y Matlab Guide.

Capítulo 6:
INTEGRACIÓN DE LA CÉLULA ROBOTIZADA

6. Integración de la célula robotizada.

En este apartado se describe el proceso seguido para la realización del proyecto. El objetivo final es conseguir que el usuario, desde una interfaz gráfica mostrada a través de una pantalla táctil, escriba una palabra y el robot que cuenta con un sistema de visión artificial, reconozca cada una de las letras impresas en las piezas cúbicas que están depositadas de forma aleatoria en la plataforma grande y las coloque en la segunda, escribiendo la palabra deseada por el usuario.

Este capítulo que se divide en siete partes y contiene las tareas a realizar para este TFM, que son:

1. Fabricación de componentes del escenario. El primer apartado consta del diseño de las piezas 3D junto con cada una de las piezas diseñadas, sus medidas y su resultado dentro de la célula robotizada.
2. Configuración del robot. En la segunda sección se expone el programa de control del robot junto con la configuración, calibración y selección de coordenadas del robot.
3. Comunicación TCP. En la tercera parte se trata el establecimiento de la conexión TCP, mostrando el intercambio de datos entre los dispositivos.
4. Visión artificial. El cuarto apartado explica la disposición, iluminación y calibración de la cámara junto con la extracción de información de la imagen.
5. Normativa de seguridad. La quinta sección plantea la normativa actual referente a la seguridad de los sistemas robóticos y el diseño de la seguridad y espacio de trabajo de la celda en base a esta normativa.
6. Interfaz gráfica de usuario. En la sexta parte se trata el diseño e implementación de la interfaz gráfica de usuario y el funcionamiento de cada una de sus partes.
7. Arquitectura de control. En la séptima parte se expone la arquitectura de control y los esquemas de conexiones realizados.

6.1. Fabricación de componentes del escenario.

En este capítulo se expone el diseño de las piezas 3D desarrolladas para el TFM junto con sus planos y la aplicación utilizada para ello. Estas piezas conforman el entorno de trabajo del robot el cual está compuesto por tres tipos de piezas:

- Plataforma grande.
- Plataforma pequeña.
- Cubos.

6.1.1. Plataforma grande.

En este apartado se define la primera tarea a realizar y es que, en el entorno de trabajo del robot, es necesario disponer de una plataforma de la que se puedan recoger los cubos con letras.

Para poder formar la mayor cantidad posible de palabras se necesita una cantidad importante de cubos por lo que se ha decidido diseñar una plataforma con 63 huecos, para poder colocar 63 cubos.

Teniendo en cuenta que los cubos tienen 25mm de lado, se han creado unos huecos en la plataforma de 27mm, para tener un margen de 2mm cuando el robot recoja y deposite dichos cubos, como muestra la figura 6.1.

Además, si el robot no sitúa de forma muy precisa el cubo en el hueco, se ha añadido un chaflan de 3mm por lado, para que el cubo se deslice y caiga correctamente dentro del hueco.

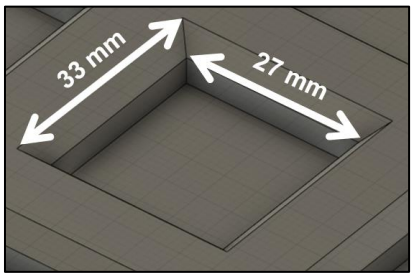


Figura 6.1. Dimensiones de los huecos de la plataforma.

Para disponer de los 63 huecos en el espacio más reducido posible, debido al espacio de trabajo del robot, la plataforma cuenta con 7 filas y 9 columnas de huecos, cuya separación entre cubo y cubo es de 10mm.

Esta separación es necesaria, ya que para coger cada uno de los cubos, se utiliza una pinza neumática de dos dedos y cada uno de esos dedos tiene una anchura de 4.11mm. En la figura 6.2, en la parte izquierda se puede observar un plano 3D del adaptador de la pinza sin las patas y en la parte derecha se muestra la pinza en la realidad.

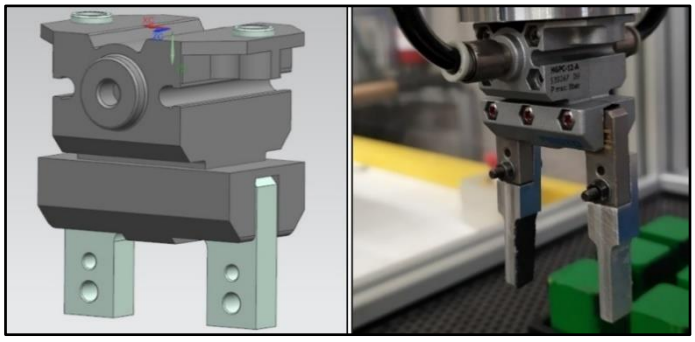


Figura 6.2. Adaptador de la pinza del robot.

De esta manera, se dispone de un margen de 3 mm a cada lado de las patas de la pinza para al recoger un cubo no impactar con el que se encuentra a su lado. En el anexo de planos de la pinza (I.II. Planos Pinza.) se observa el plano de las patas de la pinza, donde se muestra la anchura de las mismas anteriormente comentada.

Teniendo en cuentas estas dimensiones, para conseguir una plataforma con 63 huecos, se ha diseñado una plataforma de 381mm de largo y 295mm de ancho, como se puede observar en la figura 6.3.

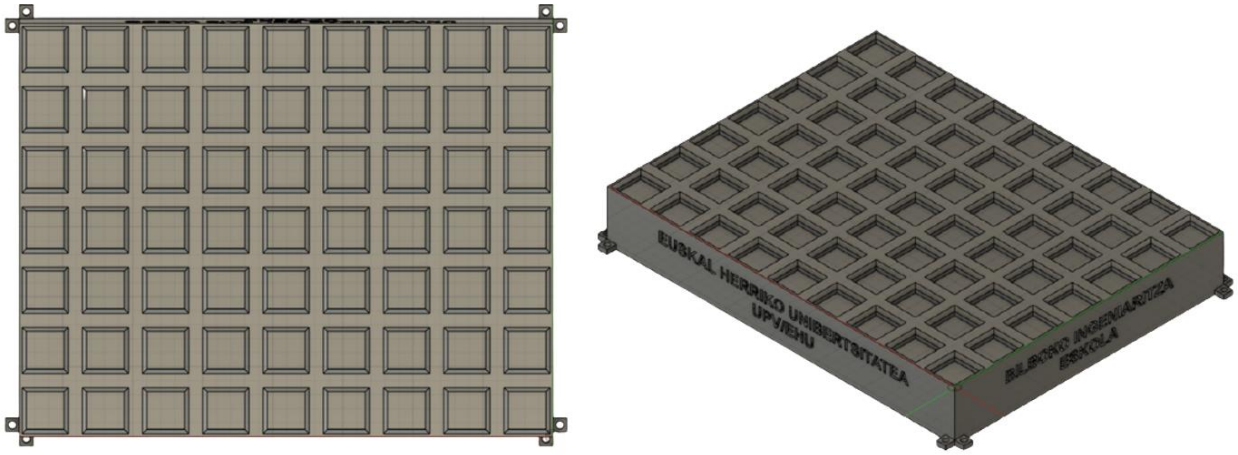


Figura 6.3. Plataforma grande completa.

Por otro lado, se han añadido ocho puntos de anclaje, dos por cada esquina, para atornillar la plataforma a la base como puede verse en los planos de la plataforma. Para ver el plano completo acceder a su anexo (I.I. Plano plataforma grande.).

Otro dato a tener en cuenta para el diseño de esta pieza, es el volumen de trabajo del robot, el cual se puede observar en la figura 6.4, donde se puede ver que dicho volumen está formado por un cilindro y un cono cortado en su parte inferior.

El radio inferior del cono es de 132.75mm, distancia insuficiente para alcanzar todas las posiciones de la plataforma grande, ya que esta tiene una anchura de 381mm, por lo que es imprescindible trabajar en el volumen de trabajo del cilindro del robot que es de 250mm de radio y así poder alcanzar todas las posiciones.

Para ello, la altura de la plataforma respecto de la base tiene que ser superior a 45mm, por lo que se ha definido la altura de la pieza a 55mm para trabajar recogiendo y colocando piezas de forma segura dentro del volumen del cilindro, como se muestra en la figura 6.4.

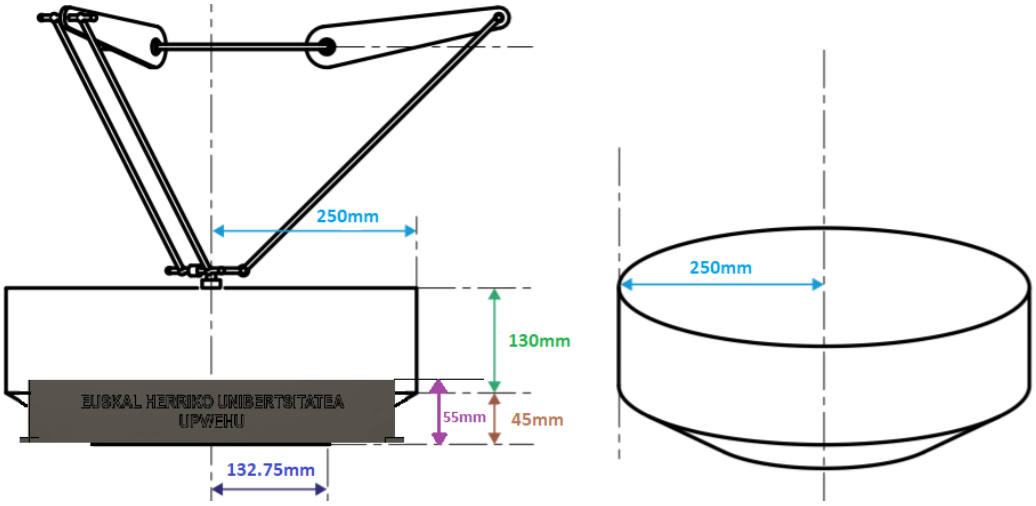


Figura 6.4. Representación de las dimensiones de la plataforma grande del espacio de trabajo del robot.

Por último, se ha serigrafiado en los laterales de la plataforma el nombre de la Universidad en castellano y en euskera, como se puede ver en la figura 6.5.

Figura 6.5. Plataforma grande en la realidad.

6.1.1.1. División de la plataforma grande.

Debido a las dimensiones de esta plataforma (381mm de largo y 295mm de ancho) y sabiendo que el área máxima de impresión de la impresora 3D que se dispone es de 250mmx250mm, ha sido necesario dividir la plataforma en cuatro partes y añadir unas pequeñas prolongaciones para posteriormente unirlas como si de un puzle se tratara.

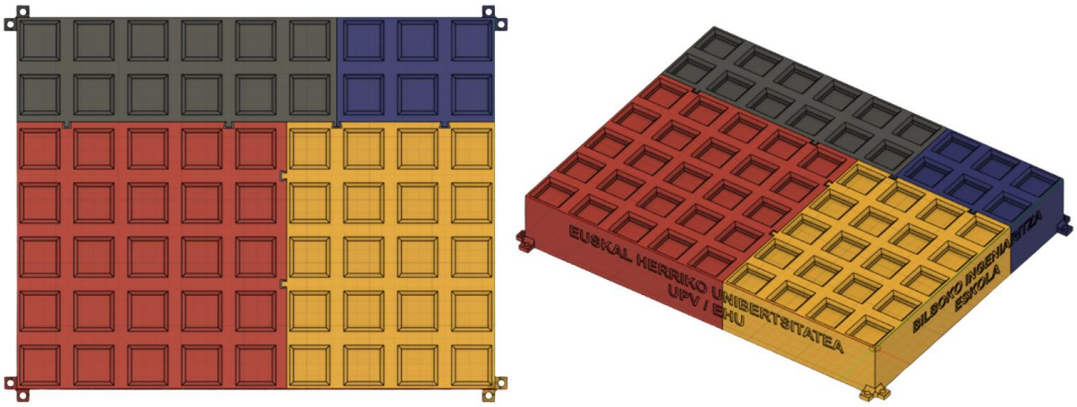


Figura 6.6. Divisiones de la plataforma grande para su impresión 3D.

6.1.2. Plataforma pequeña.

En este apartado se define la segunda pieza a realizar y al igual que se ha diseñado una plataforma grande de donde recoger los cubos, es necesario disponer de otra plataforma donde colocar los cubos formando la palabra deseada. Para ello, se ha diseñado una plataforma pequeña de 9 huecos, por lo que la longitud máxima de las palabras que se pueden formar es de 9 letras, donde colocar los cubos de la palabra deseada.

Para el diseño de esta plataforma, se han tenido en cuenta las mismas condiciones que para el diseño de la plataforma grande, por lo que se mantienen las dimensiones de los huecos y del espacio entre huecos.

Teniendo esto en cuenta, se ha diseñado una plataforma con dos filas de 9 huecos, que posee 381mm de largo y 80mm de ancho, como se puede observar en la parte izquierda de la figura 6.7.

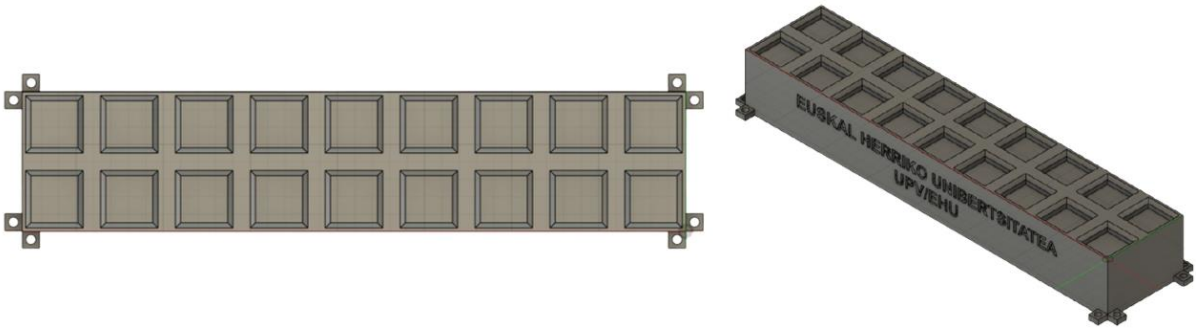


Figura 6.7. Plataforma pequeña completa.

Por otro lado, se han añadido ocho puntos de anclaje, dos por cada esquina, para atornillar la plataforma a la base. Para ver el plano completo acceder a su anexo (I.III. Plano plataforma pequeña.).

Al igual que ocurría con el diseño de la plataforma grande, para el diseño de esta pieza se ha tenido que tener en cuenta el espacio de trabajo del robot, el cual obliga a trabajar dentro del volumen del cilindro para alcanzar todas las posiciones, por lo que la altura de esta plataforma también será de 55mm.

Además, tanto la plataforma grande como la pequeña, han sido diseñadas para que ambas puedan convivir dentro del volumen del cilindro y así tener disponibles todos los huecos de las plataformas como se muestra en la figura 6.8, que muestra una representación en planta del área del cilindro y la disposición de las plataformas dentro de él.

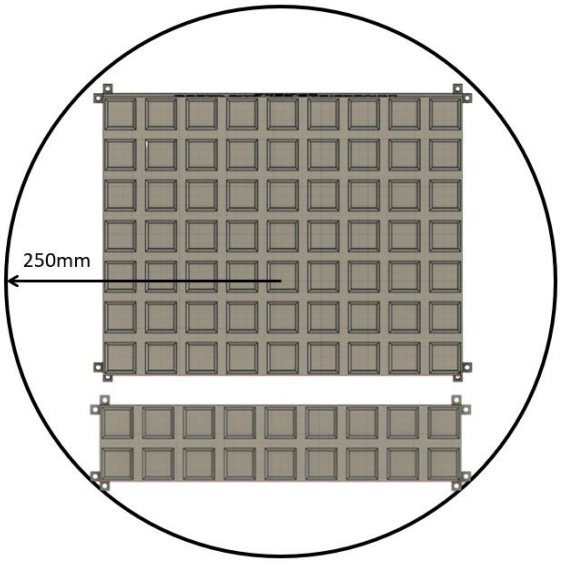


Figura 6.8. Vista en planta del área de trabajo del cilindro del robot.

Al igual que se ha hecho en la plataforma grande, se ha serigrafiado en los laterales el nombre de la Universidad en castellano y en euskera. El resultado de la plataforma pequeña en la realidad puede observarse en la figura 6.9.

Figura 6.9. Plataforma pequeña en la realidad.

6.1.2.1. División de la plataforma pequeña.

Debido a las dimensiones de esta plataforma (381mm de largo y 80mm de ancho) y sabiendo que el área máxima de impresión de la impresora 3D que se dispone es de 250mmx250mm, ha sido necesario dividir la plataforma en dos partes y añadir una pequeña prolongación para posteriormente unirlas como un puzle.

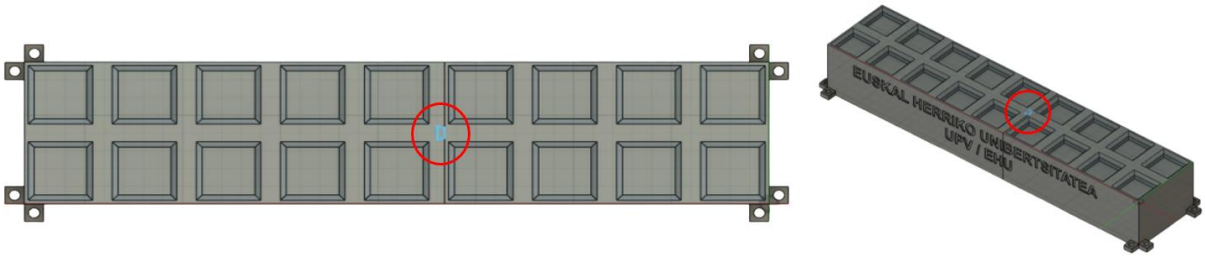


Figura 6.10. Divisiones de la plataforma pequeña para su impresión 3D.

6.1.3. Cubos.

En este apartado se expone el diseño de los cubos que el robot va a coger y dejar en cada uno de los huecos de la plataforma. Estos cubos tienen 25mm por cada lado y en su cara superior se ha serigrafiado una letra del abecedario, como se puede observar en la figura 6.11. La única letra que no se ha diseñado, ya que no se va a tener en cuenta en la programación, es la letra ñ. Para ver el plano completo de los cubos ver el anexo (I.IV. Plano cubos.).

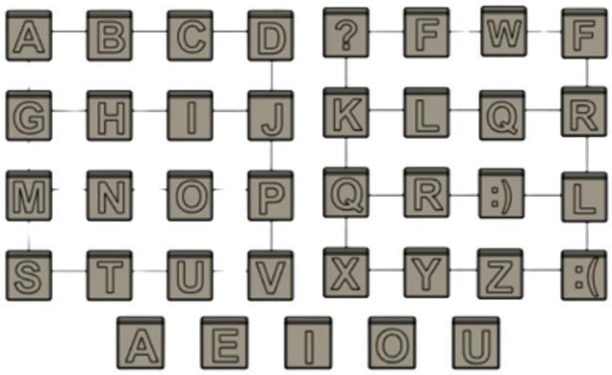


Figura 6.11. Cubos con letras serigrafiadas.

El resultado de los cubos finales puede observarse en la figura 6.12.

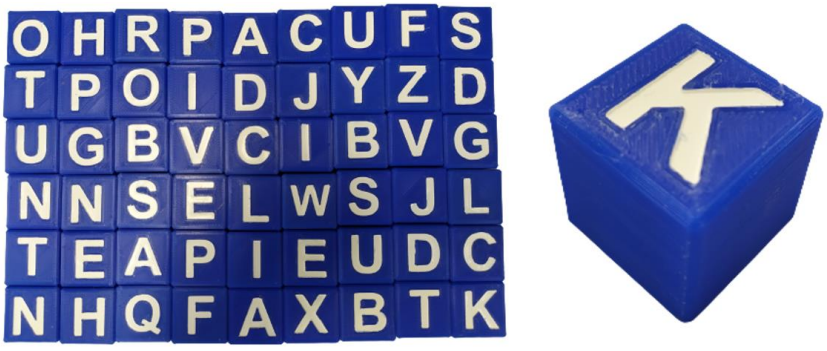


Figura 6.12. Cubos con cada letra serigrafiada en la parte superior.

Figura 6.13. Resultado de colocar los cubos sobre la plataforma en simulación y en la realidad.

6.1.4. Escenario de la célula robotizada.

En esta sección, se muestra el diseño del escenario de la célula robotizada, que es la distribución de todos los elementos dentro de la célula del robot. En este primer caso de estudio, se consideran únicamente los elementos necesarios para la aplicación que se quiere realizar, buscando una ubicación para todos ellos que minimice la longitud de los desplazamientos y que se encuentre siempre dentro del espacio de trabajo del robot.

En la figura 6.14, se muestra el resultado de la distribución planteada, con las dos plataformas, el robot, el monitor táctil y la webcam.

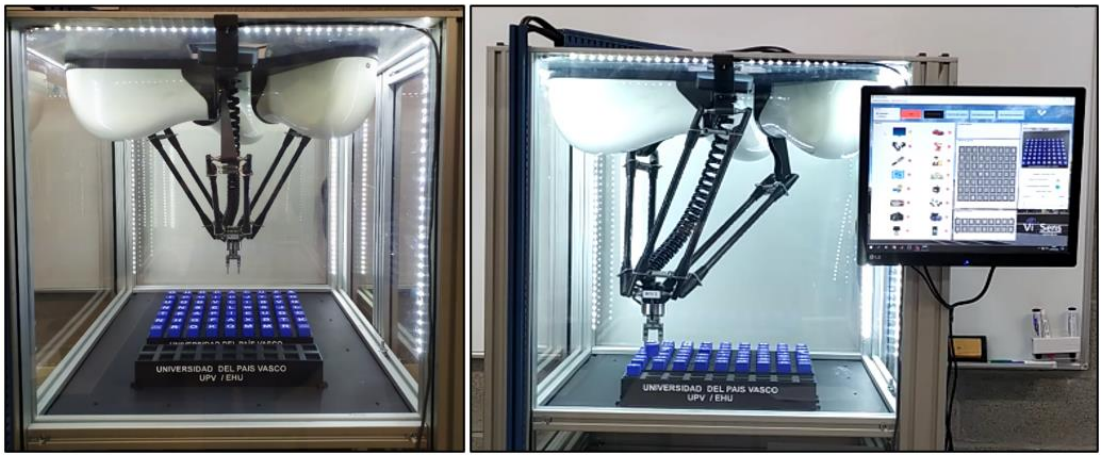


Figura 6.14. Distribución del habitáculo superior del robot en la realidad.

6.2. Configuración del robot.

En este capítulo, se expone la configuración del robot mediante la herramienta Sysmac Studio junto con el programa de control del robot.

Antes de comenzar con la programación del robot en Sysmac Studio, la primera tarea a realizar es la configuración de la red EtherCAT y la configuración de los ejes, [54] que hay que cargar en el PLC.

6.2.1. Configuración de la red EtherCAT.

En primer lugar, hay que configurar los módulos del controlador en base a la configuración física que existe. Posteriormente hay que definir la red EtherCAT y pasarle el nombre y el número de nodo que se le ha asignado a cada módulo, como se tienen tres servomotores (uno para cada eje del robot):

- El servodriver 1 R88D-KN04h-ECT tiene el número de nodo 1.
- El servodriver 2 R88D-KN04h-ECT tiene el número de nodo 2.

- El servodriver 3 R88D-KN04h-ECT tiene el número de nodo 3.
- El acoplador EtherCAT NX-ECC201 tiene el número de nodo 4.

En la figura 6.15 se muestra la configuración de la red EtherCAT.

Figura 6.15. Configuración de la red EtherCAT.

6.2.2. Configuración de los ejes.

El robot tiene tres servomotores; uno para el eje X, otro para el eje Y y otro para el eje Z. En el apartado configuración, se tiene que crear un eje por cada servomotor del robot Mini Delta como se muestra en la figura 6.16, ya que es necesario realizar un control en cada eje para poder alcanzar la posición deseada correctamente.

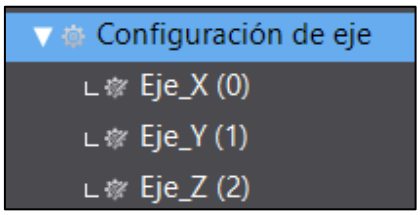


Figura 6.16. Configuración de ejes.

Para la configuración de cada eje, hay que realizar los siguientes pasos:

1. Realizar la configuración básica del eje, como se puede ver la figura 6.17.

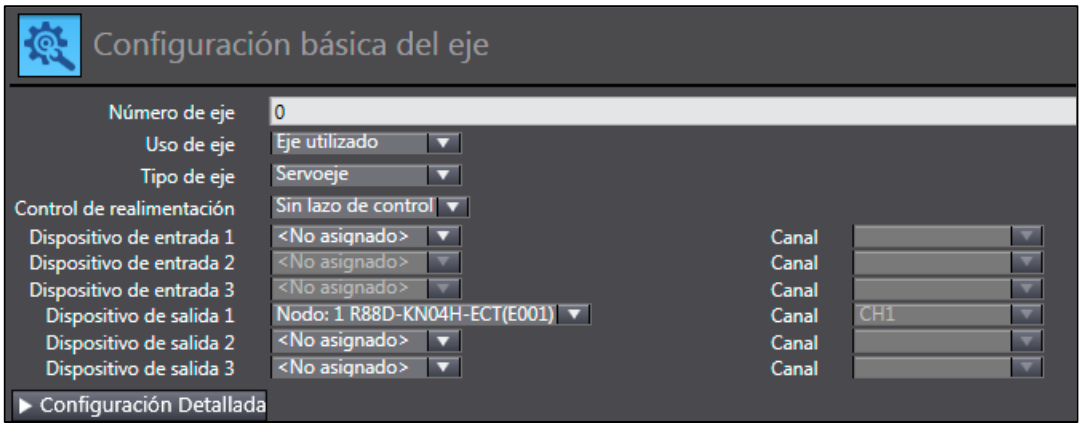


Figura 6.17. Configuración básica del eje.

2. La configuración de la conversión de unidades. El motor Accurax G5 posee un encoder absoluto con una resolución de 17 bits, $2^{17} = 131072$, con una relación de reducción 1:25 y un contador de impulsos de comando por rotación de $131072 \text{ x } 25 = 3276800$.

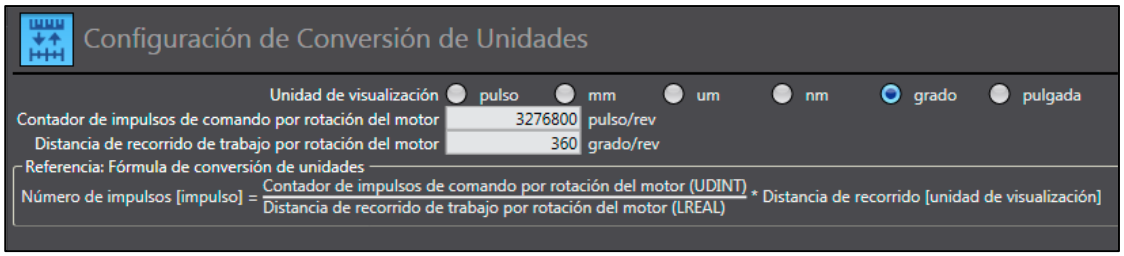


Figura 6.18. Configuración de la conversión de unidades.

3. El siguiente paso es la configuración de las velocidades, aceleración, deceleración y par del robot como ilustra la figura 6.19.

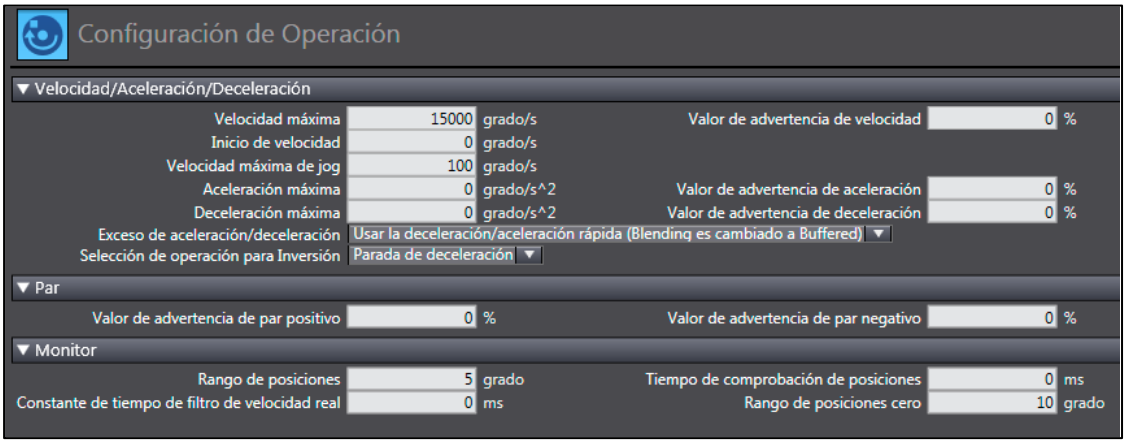


Figura 6.19. Configuración de operación.

4. Posteriormente, se realizan otras configuraciones como se puede observar en la figura 6.20.

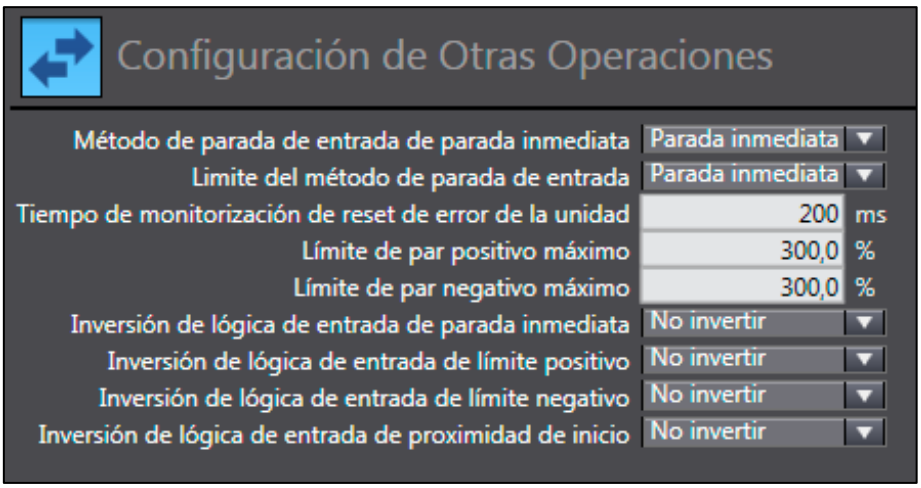


Figura 6.20. Configuración de otras operaciones.

5. Tal y como se ha comentado en el apartado de las limitaciones software del robot (*4.8. Limitaciones Software.*) y por la seguridad de este, es necesario limitar por software el rango de movimiento de cada servomotor.

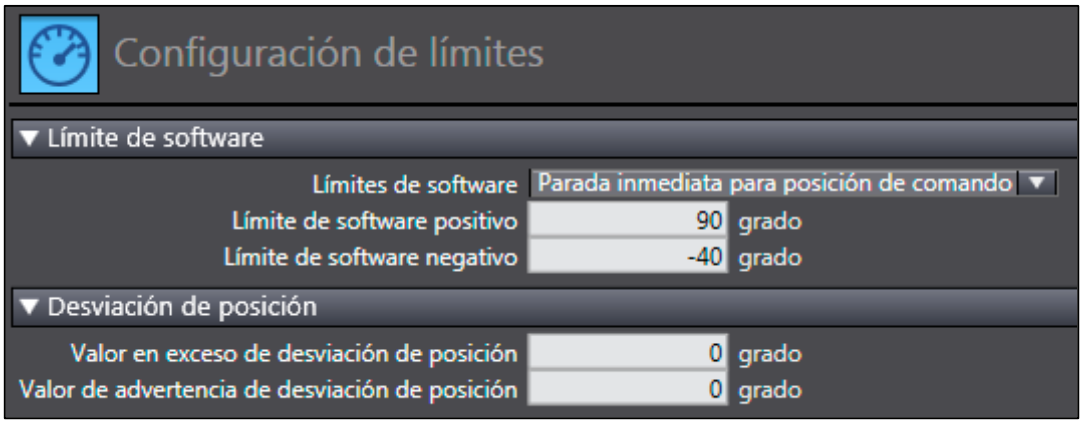


Figura 6.21. Configuración de los límites de operación.

6. El siguiente paso es establecer el método de Homing.

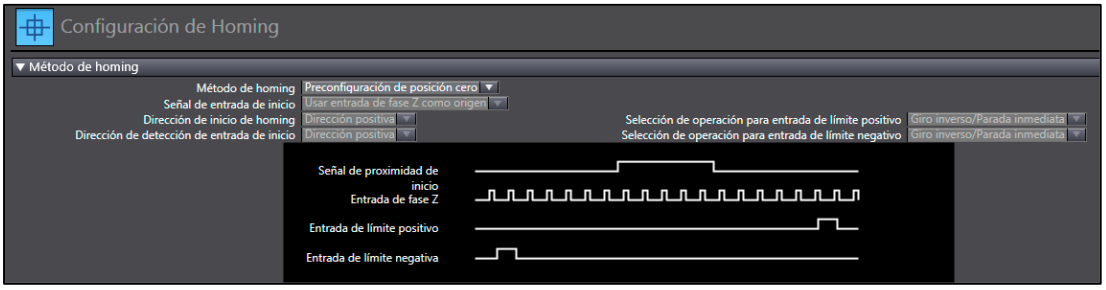


Figura 6.22. Configuración de Homing.

7. Por último, se realiza la configuración del contador lineal y del encoder absoluto.

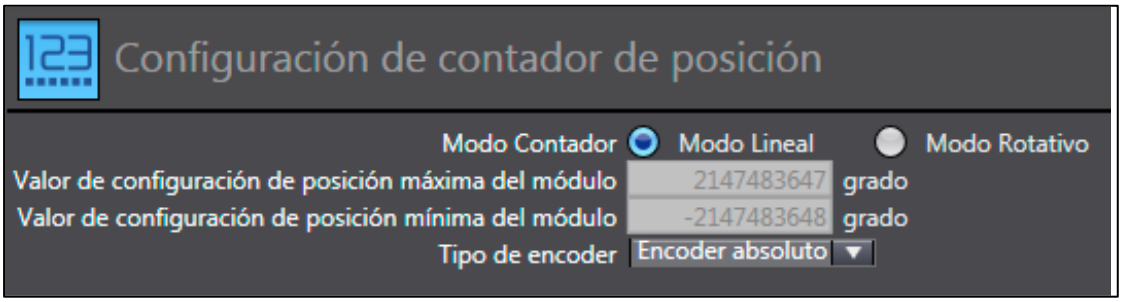


Figura 6.23. Configuración de contador de posición.

6.2.3. Configuración de grupo de ejes.

Una vez definidos los ejes, hay que crear el grupo con los ejes X, Y y Z para formar el grupo Mini Delta, que permite el control del grupo de motores de forma coordinada, como se puede observar en la figura 6.24.

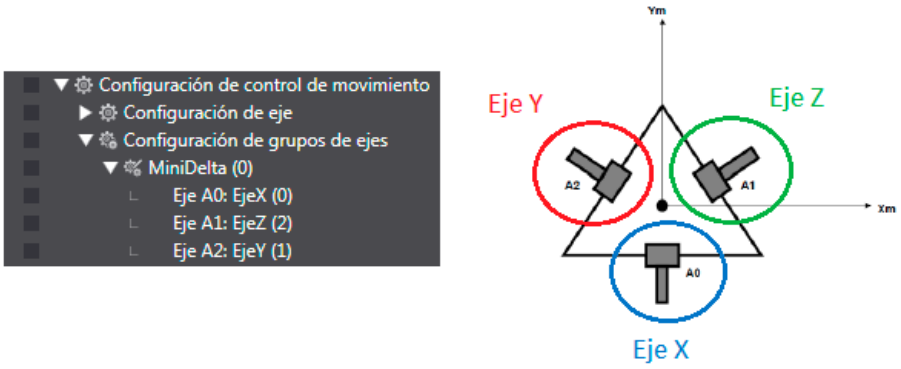


Figura 6.24. Grupo de ejes.

El siguiente paso es realizar la configuración básica del grupo de ejes, en la que se asigna el control de cada eje lógico a cada eje de composición, tal y como se puede observar en la figura 6.25.

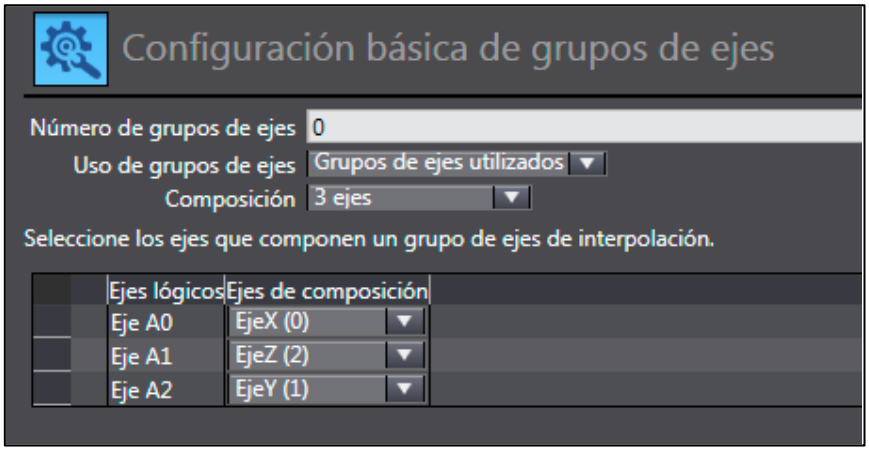


Figura 6.25. Configuración básica de grupo de ejes.

6.2.4. Configuración de tareas.

Sysmac Studio funciona con un sistema de prioridad de tareas, teniendo que configurar al menos, la tarea principal. El periodo de tiempo que se quiere para que se repita la tarea principal es de 1ms (ver Figura 6.26).

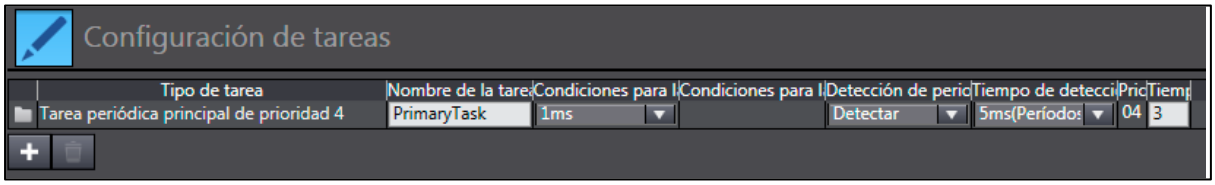


Figura 6.26. Configuración de tareas.

Se ha definido como tarea periódica de prioridad 4 para que el programa la considere como la única tarea obligatoria que tiene que realizar, donde se encuentran las funciones, programas y bloques de función implementados.

Esta prioridad de tarea es la más alta como se puede ver en la figura 6.27, indicando que es la principal tarea periódica a realizar, dentro de una escala que va desde 4 hasta 48.

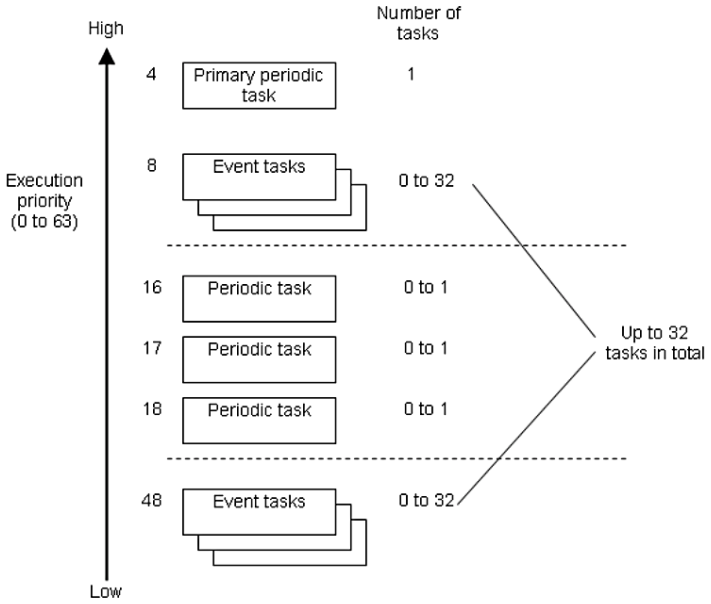


Figura 6.27. Funcionamiento de las tareas y sus prioridades en Sysmac Studio.

Tras realizar la configuración hardware del robot, es necesario realizar una calibración mecánica del robot y mapeo del espacio de trabajo del robot a coordenadas cartesianas que puede verse en el anexo (VII.I. Método de calibración del robot Mini Delta.).

6.2.5. Programa de control del robot.

Una importante tarea a realizar es el programa de control del robot que se divide en siete partes. Para ello, se utilizan las funciones de la librería “Motion Control” que permiten manejar los motores del robot y posicionar el extremo en la coordenada deseada y/o realizar una trayectoria determinada a una velocidad o tiempo establecido. A continuación, se muestran algunas de las funciones más utilizadas:

- Reset. Elimina los errores que pueda tener el eje.
- Home. Establece como casa (0º) la posición en la que se encuentre el eje.
- Power. Alimenta el servomotor para que esté listo para funcionar.
- GroupEnable. Habilita un grupo de ejes.
- GroupSyncMoveAbsolute. Emite cíclicamente la posición de destino especificados para los ejes.
- Move. Realiza el movimiento relativo o absoluto del eje.
- MoveLinear. Realiza la interpolación lineal.
- MoveCircular2D. Realiza la interpolación circular de dos ejes.

- MoveVelocity. Realiza el control de velocidad.
- SetTorqueLimit. Limita el par de salida del servomotor.
- Gearin. especifica la relación de transmisión entre el eje maestro y el eje esclavo, y se inicia la operación de sincronización.
- DefineCoordSystem. Establece el sistema de coordenadas de un robot.
- GroupMon. Lee la posición y velocidad actual de un robot.
- InverseKin. Realiza la cinemática inversa.
- MoveTimeAbsolute. Crea valores de comandos para llegar a una posición de destino en un tiempo determinado.
- SetKinTransform. Establece la cinemática de control del robot y verifica el área de trabajo para un grupo de ejes.

Para más información sobre el software de control Sysmac Studio [55] consultar el documento técnico de la bibliografía.

Para el correcto funcionamiento de la aplicación, se han desarrollado cuatro programas, tres funciones y un bloque de función, que se ejecutan de forma secuencial y se pueden dividir en las siguientes dos partes:

1. Inicialización.
2. Movimiento del robot.

6.2.5.1. Inicialización.

Para arrancar el robot, hay que presionar el pulsador verde de la botonera de activación del robot y colocar el selector de tres posiciones en la parte superior. Tras esto, comenzará a ejecutarse el programa desarrollado para la inicialización del robot (ver anexo II.I. Inicialización del robot.). Es necesario asegurarse previamente de que el robot no se encuentre arrancado.

Durante el tiempo que el robot se encuentre en este estado, parpadeará el led verde de la botonera auxiliar programable y se ejecutarán internamente las siguientes etapas:

1. Se borra la memoria de los errores detectados relacionados con el eje y el variador para que esté disponible para nuevos mensajes de error. Si la fase de arranque está deshabilitada con la respuesta de error del variador, se puede volver a habilitar si se ha corregido la causa del error detectado.

   También se comprueba si el PLC tiene algún error. Si hubiera algún error en el PLC, en el eje o en el variador, se encenderá el piloto rojo y los servomotores no se podrán alimentar hasta que se solucione el problema.

2. Después de que pasen 5 segundos desde que se aprieta el pulsador de arranque, se alimentan los servomotores y se lleva cada eje a la posición de reposo del robot.
3. Tras verificar que el robot se encuentra en la posición de reposo, se define la cinemática y el espacio de trabajo del robot.
4. Se convierte el sistema de coordenadas de la máquina (MCS) al sistema de coordenadas del usuario (UCS).
5. Cuando se haya completado la inicialización, el robot se moverá a la posición de reposo, el piloto verde se iluminará de forma continua y se colocará el selector de tres posiciones de la botonera en la posición horizontal (pausa).

6.2.5.2. Movimiento del robot.

Con la inicialización completada y tras colocar el selector de tres posiciones de la botonera en la posición inferior izquierda (RUN), comenzará a ejecutarse el programa implementado para el movimiento del robot en el PLC (ver anexo II.II. Movimiento del robot.).

El robot comienza moviéndose a la posición de reposo que se ha definido para poder obtener imágenes de la webcam sin que las patas del robot interfieran en dicha imagen.

En ese momento, lo primero que se realiza es la comunicación entre el PC y el PLC la cual se explicará más adelante (6.3.). El PLC se comporta como el servidor de la comunicación, por lo que en un primer momento el PLC acepta la petición de comunicación del PC, se comprueba que el estado de la conexión sea correcto y se produce el intercambio de información.

Tras recibir la información, se acondiciona para poder utilizarla correctamente y el robot comienza a recoger el primer cubo de la plataforma grande para depositarlo en la plataforma pequeña.

Cada aproximación del robot a un cubo se realiza mediante una trayectoria obtenida con un polinomio de tercer grado. Para que no se produzcan colisiones con los cubos, todas las trayectorias del robot se realizan 30mm por encima del punto medio de lo que sobre sale cada cubo, es decir, cada cubo sobresale 17mm de la plataforma.

De manera que cada vez que se quiere recoger un cubo se realiza un movimiento a las coordenadas ‘X’ e ‘Y’, pero 30mm por encima en la coordenada ‘Z’.

Una vez se alcanza dicho punto el robot abre la pinza, desciende 30mm y cierra la pinza para recoger el cubo. Con el cubo atrapado y para evitar colisiones se asciende otra vez verticalmente 30mm y se realiza la siguiente trayectoria a la posición donde se va a depositar dicho cubo.

Cuando se ha alcanzado el punto donde se va a depositar el cubo, se desciende verticalmente 30mm, se abre la pinza y se asciende 30mm para recoger el siguiente cubo y así con todos los cubos que tenga la palabra.

En la figura 6.28 se muestra una representación de las trayectorias explicadas anteriormente.

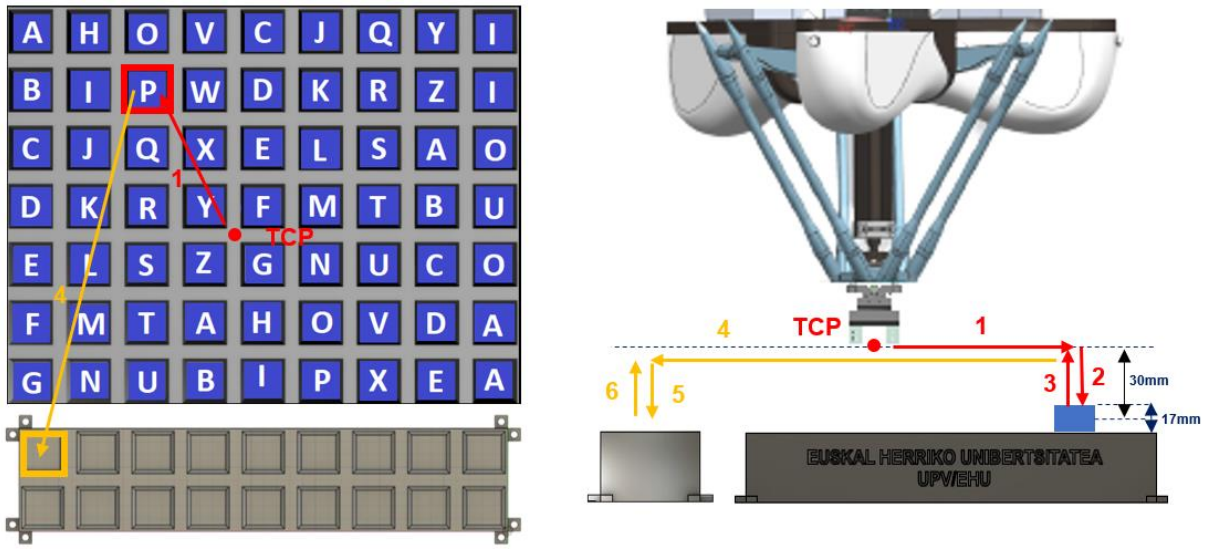


Figura 6.28. Representación de las trayectorias del robot.

Cuando haya completado la palabra, volverá a la posición de reposo y se quedará estático durante tres segundos para poder observar la palabra formada correctamente.

Cuando se termine ese tiempo, el robot volverá a moverse para recoger cada cubo de la palabra formada y depositarlo en la misma posición de donde fue extraída. Al terminar de recoger todos los cubos, el robot volverá a la posición de reposo, se cerrará la comunicación, indicando al cliente que el proceso se ha completado y el servidor volverá a estar a la escucha para formar otra palabra.

Debido a la extensión del diagrama del flujo del programa del control del robot se ha dividido en secciones más pequeñas detallando a continuación cada paso:

1. Inicialización.

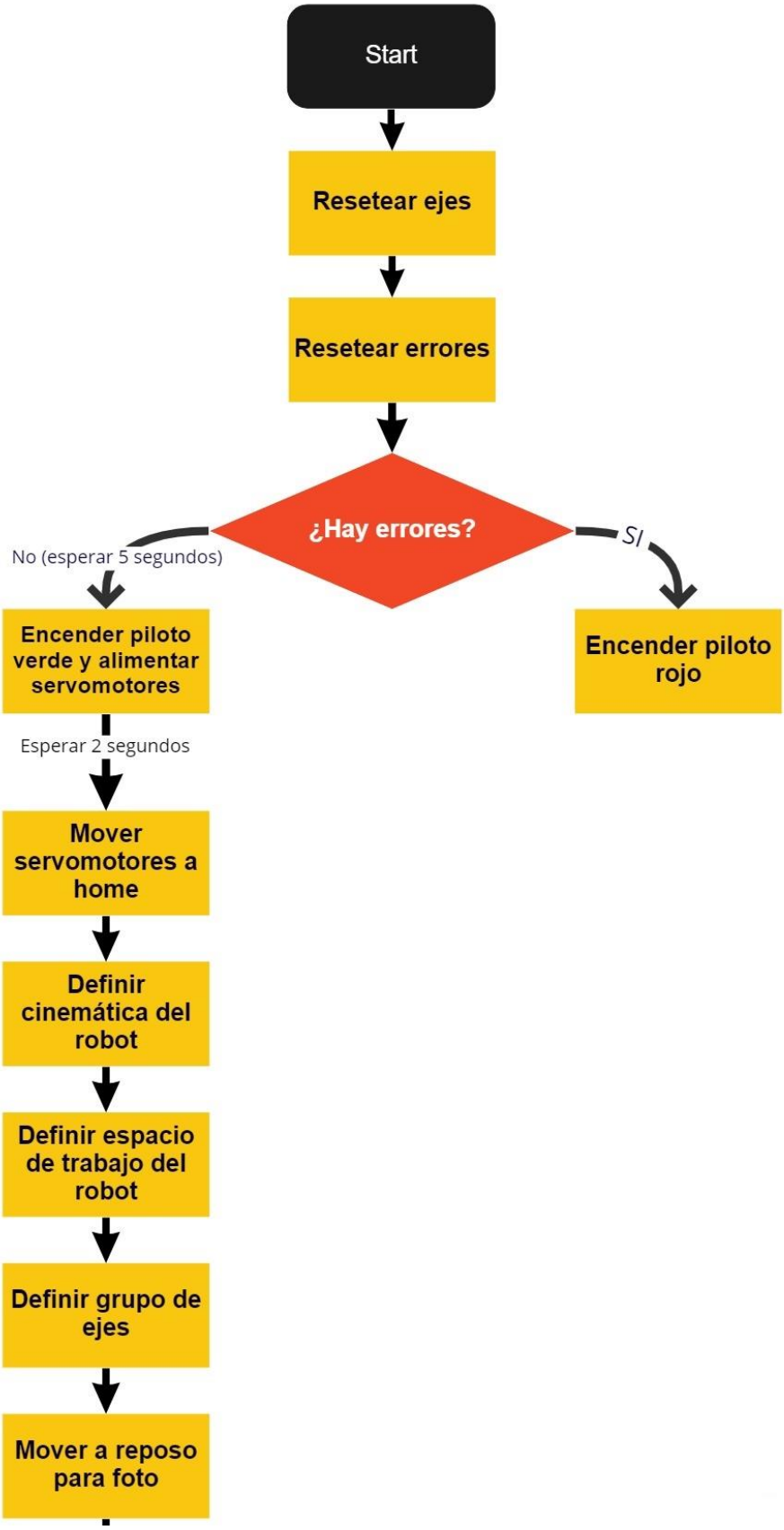


Figura 6.29. Diagrama de flujo de la inicialización.

2. Comunicación y acondicionamiento.

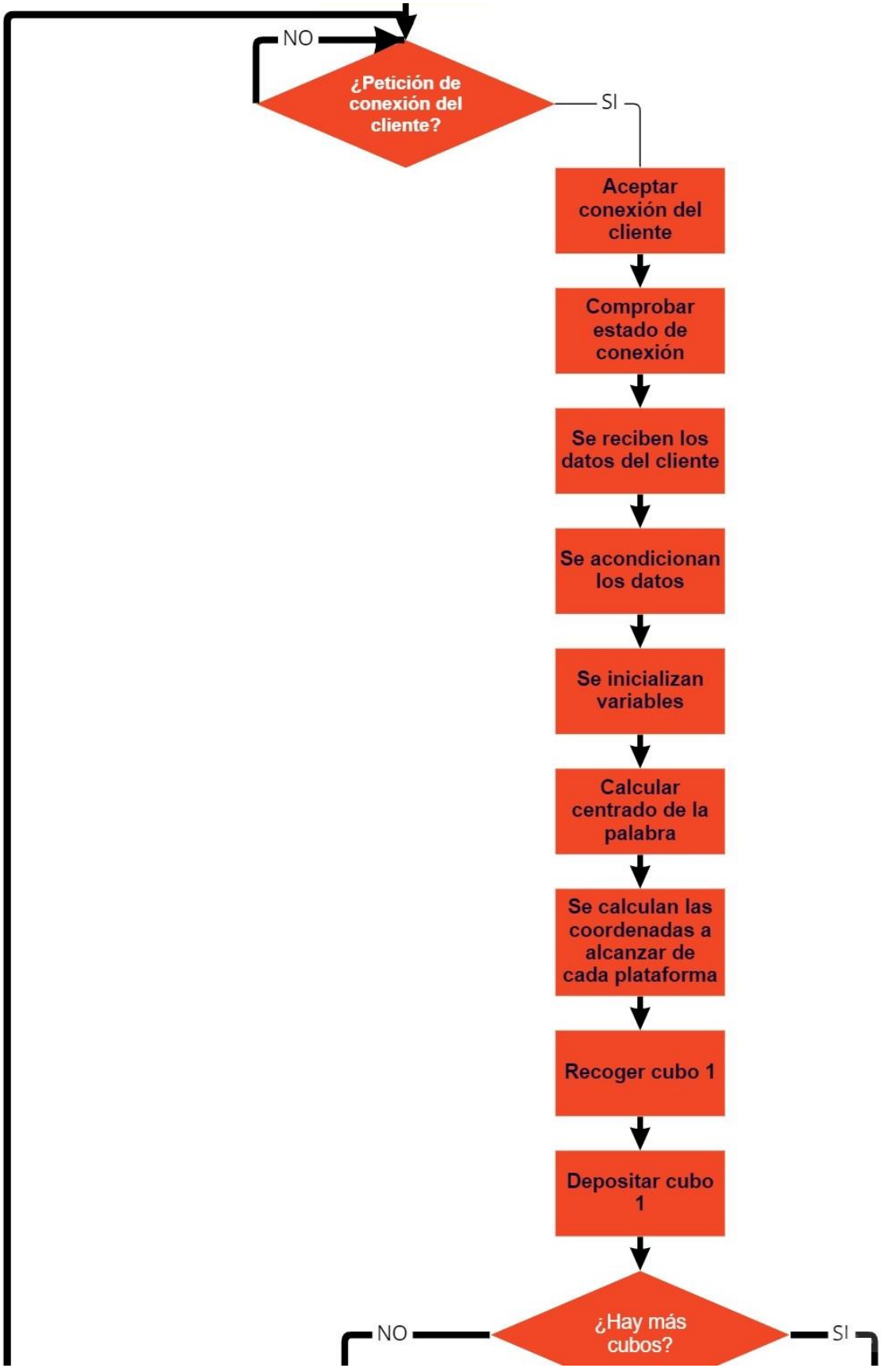


Figura 6.30. Diagrama de flujo de la comunicación y del acondicionamiento.

3. Extracción de cubos.

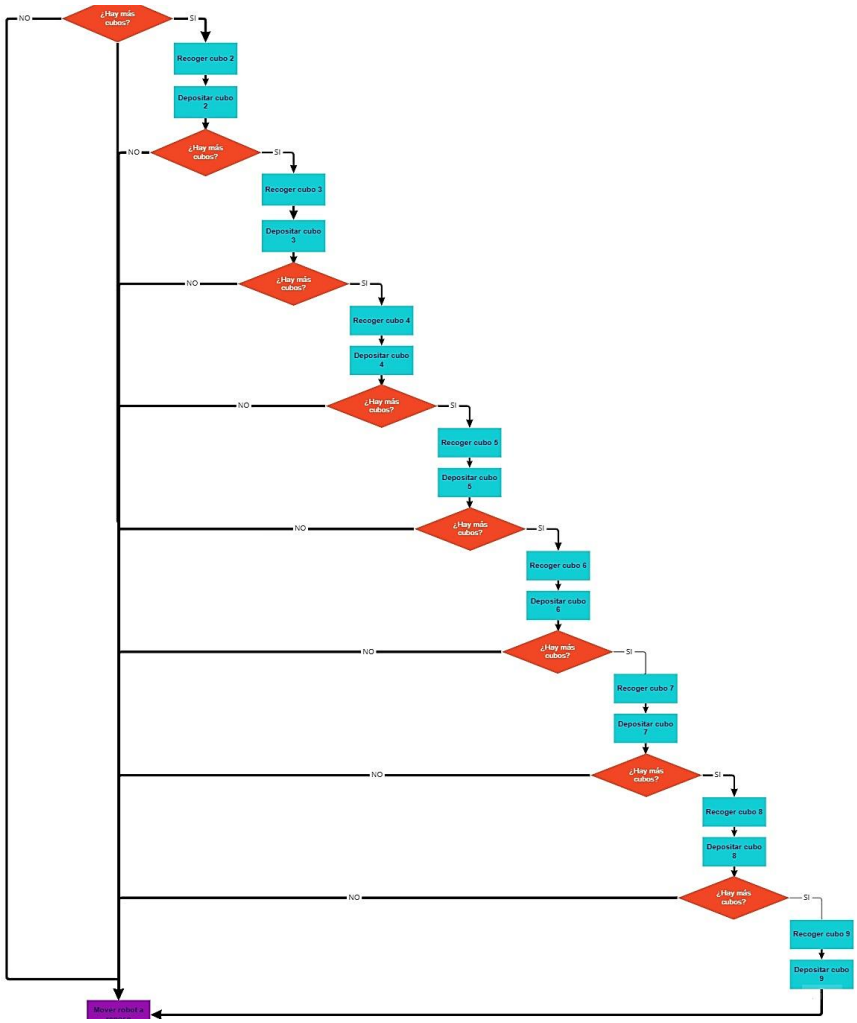


Figura 6.31. Diagrama de flujo de la extracción de cubos.

4. Recogida de cubos y cierre de la comunicación.

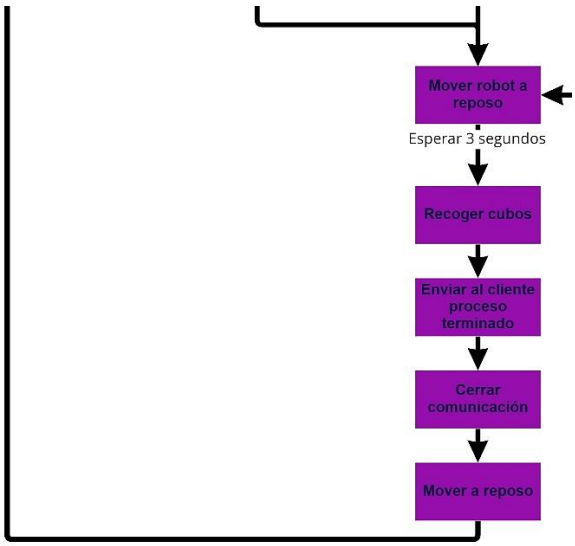


Figura 6.32. Diagrama de flujo de la recogida de cubos y cierre de la comunicación.

6.3. Comunicación TCP.

La siguiente tarea a realizar consiste en implementar un protocolo de comunicación entre un PLC y un PC que permita el intercambio de información entre ambos dispositivos.

El socket es el conjunto IP – Puerto y una dirección IP es un número de 32 bits. Las direcciones IP normalmente se expresan en formato decimal punteado, con cuatro números separados por puntos, como 192.168.137.100.

Una dirección IP tiene dos partes; la primera parte de una dirección IP se usa como dirección de red y la segunda parte como dirección host. Para el PC, se ha definido que la dirección IP a utilizar es la 192.168.137.101. Si se divide en dos partes, se obtiene que la red es 192.168.137 y el host es 101, es decir, 192.168.137.0 es la dirección de red y 0.0.0.101 es la dirección de host.

El segundo elemento, que es necesario para que la comunicación TCP/IP funcione, es la máscara de subred. El protocolo TCP/IP usa la máscara de subred para determinar si un host está en la subred local o en una red remota.

En TCP/IP, las partes de la dirección IP que se usan como direcciones de red y host no son fijas. Esta información se proporciona en otro número de 32 bits denominado máscara de subred. La máscara de subred para este trabajo es 255.255.255.0.

Por lo que para que se encuentren dentro de la misma subred la dirección IP del PLC se ha definido como 192.168.137.100. Tanto el PLC como el PC van a recibir y enviar datos desde el puerto 8080 (los puertos TCP van desde el 0 hasta el 65535), por lo que el socket queda definido como:

- PC - 192.168.137.101:8080.
- PLC - 192.168.137.100:8080.

En la figura 6.33, se muestra la configuración del puerto Ethernet utilizado para el PLC.

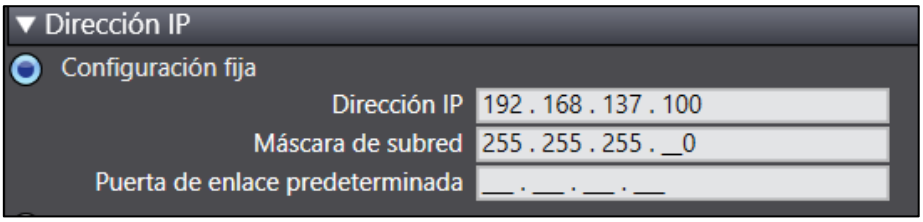


Figura 6.33. Dirección IP y mascara de subred del PLC.

Con la dirección IP del PLC definida, el siguiente paso es definir la dirección IP del PC, para ello se teclea desde el “cmd” el comando “ipconfig” y se obtiene el resultado de la figura 6.34,

donde se puede observar la dirección IPV4 del equipo actual, y esa es la dirección IP que hay que modificar a 192.168.137.101.

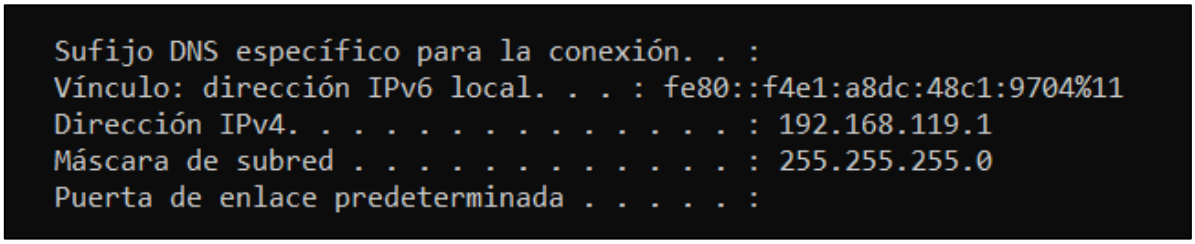


Figura 6.34. Dirección IP del PC original.

Para cambiar la dirección IP hay que ir a conexiones de red, seleccionar la red Ethernet que se quiere modificar, pinchar en propiedades, seleccionar el Protocolo de Internet versión 4 (TCP/IPv4) y poner la dirección IP y la máscara de subred comentada anteriormente, como muestra la figura 6.35.

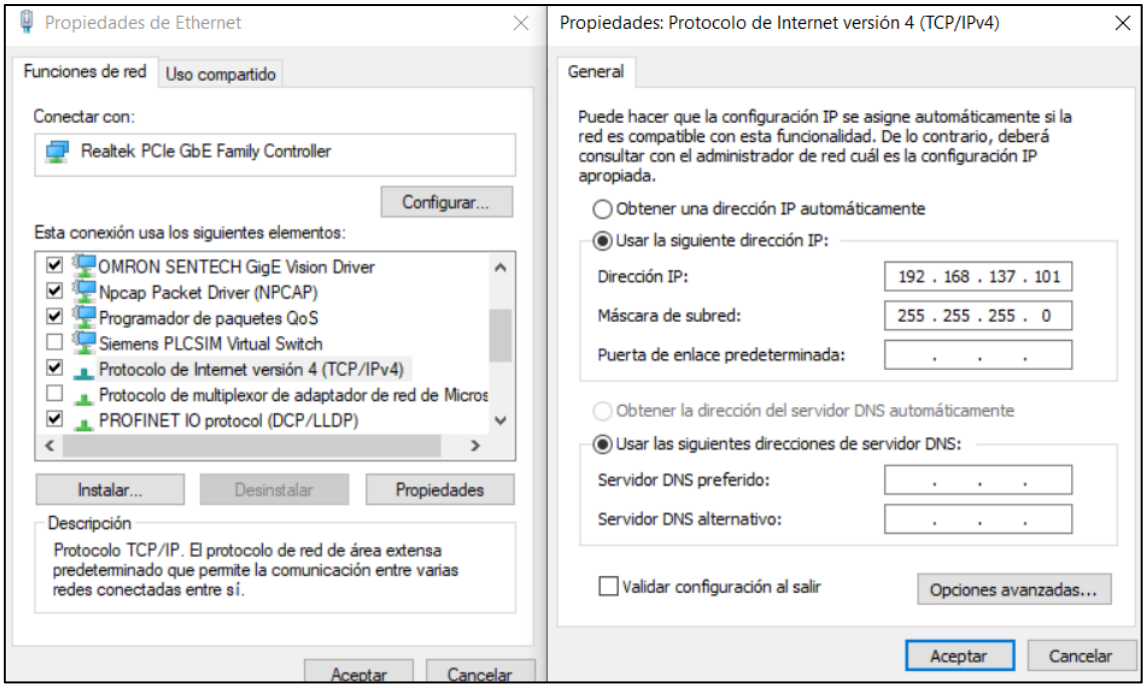


Figura 6.35.Dirección IP del PC modificada.

Con las direcciones IP del PLC y PC definidas, el siguiente paso es establecer la comunicación y una característica muy importante es el tiempo que se tarda en realizar dicha conexión, ya que realizando una comunicación entre el PLC (Sysmac Studio) y el PC (Matlab), se obtienen unos tiempos de respuesta de en torno a unos 10 segundos.

Pero si esa comunicación se realiza entre el PLC (Sysmac Studio) y el PC (Python) la conexión es inmediata, por lo que para comunicar estos dos dispositivos se va a utilizar un programa de Python como pasarela de comunicación entre el PLC y PC, como muestra el esquema de la figura 6.36.

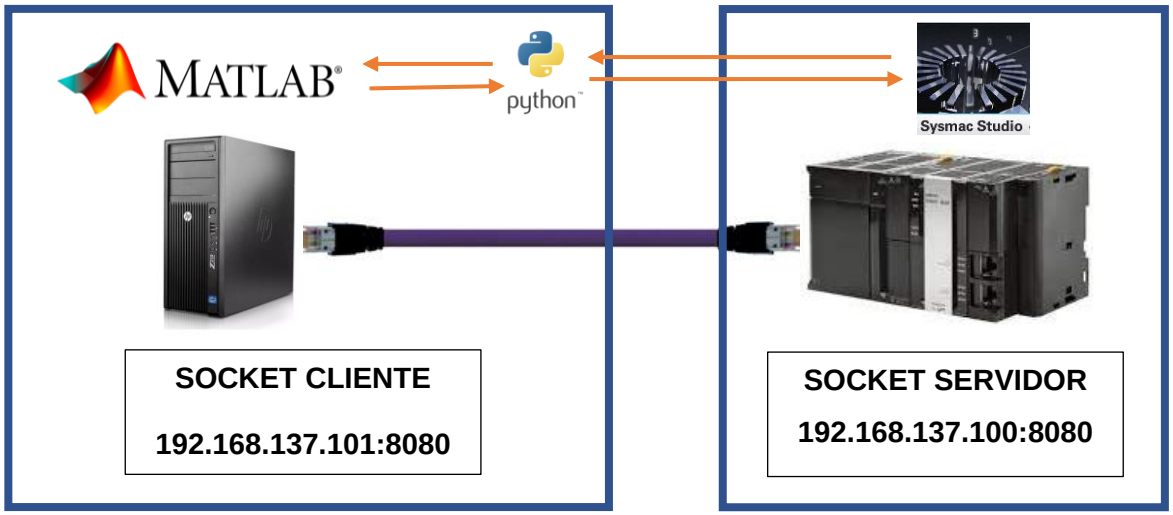


Figura 6.36. Esquema de la comunicación entre el PLC y el PC.

6.3.1. Establecimiento de la conexión con TCP.

La principal característica del protocolo TCP es que es un protocolo orientado a conexión y para poder establecer una conexión entre cliente y servidor es necesario establecer una conexión previa con dicho servidor.

Esta conexión previa se denomina 3-way handshake, y consiste básicamente en que el cliente (el que inicia la conexión) envía un mensaje SYN al servidor (el que recibe la conexión).

Posteriormente, el servidor le enviará un mensaje de tipo SYN-ACK, indicando que puede empezar a enviar información, finalmente, el cliente envía un ACK indicando que lo ha recibido correctamente, y es entonces cuando se puede comenzar a enviar toda la información entre cliente y servidor de manera bidireccional, como se puede observar en la figura 6.37.

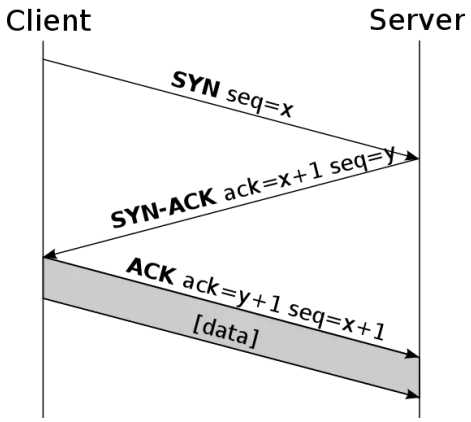


Figura 6.37. Establecimiento de la conexión entre el cliente y el servidor.

Para finalizar la conexión, el que quiere finalizar la conexión envía un mensaje FIN, y el host que lo recibe enviará un mensaje ACK junto con otro mensaje FIN, de tal forma que, el equipo

que ha iniciado la finalización de la conexión, le envíe un último ACK y se cerrará el socket abierto, como se muestra en la figura 6.38.

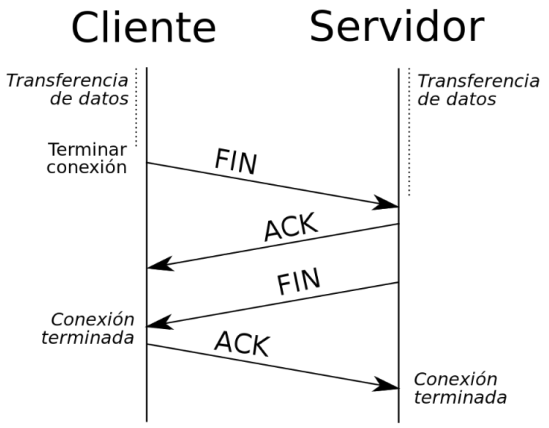


Figura 6.38. Finalización de la conexión entre el cliente y el servidor.

Para poder ver este establecimiento de la conexión e intercambio de datos se ha utilizado el programa Wireshark [56] cuyo principal objetivo es el análisis de tráfico y la resolución de problemas de red.

Figura 6.39. Logo de la aplicación para capturar el tráfico de red.

Dentro de la aplicación, para poder ver esta comunicación hay que seleccionar el método de conexión, el puerto y escribir en el filtro de visualización “tcp.port==8080” para observar el tráfico de datos de la comunicación TCP en el puerto 8080.

	Time	Source	Destination	Protocol	Length	Info
143	135.123880	192.168.137.101	192.168.137.100	TCP	66	50070 → 8080 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=2
144	135.125919	192.168.137.100	192.168.137.101	TCP	66	8080 → 50070 [SYN, ACK] Seq=0 Ack=1 Win=9000 Len=0 MSS=
145	135.126003	192.168.137.101	192.168.137.100	TCP	54	50070 → 8080 [ACK] Seq=1 Ack=1 Win=131328 Len=0
146	135.126336	192.168.137.101	192.168.137.100	TCP	82	50070 → 8080 [PSH, ACK] Seq=1 Ack=1 Win=131328 Len=28
147	135.330889	192.168.137.100	192.168.137.101	TCP	60	8080 → 50070 [ACK] Seq=1 Ack=29 Win=10220 Len=0
200	232.759112	192.168.137.100	192.168.137.101	TCP	60	8080 → 50070 [PSH, ACK] Seq=1 Ack=29 Win=10220 Len=1
201	232.759667	192.168.137.101	192.168.137.100	TCP	54	50070 → 8080 [FIN, ACK] Seq=29 Ack=2 Win=131328 Len=0
202	232.761090	192.168.137.100	192.168.137.101	TCP	60	8080 → 50070 [ACK] Seq=2 Ack=30 Win=10220 Len=0
203	232.761090	192.168.137.100	192.168.137.101	TCP	60	8080 → 50070 [FIN, ACK] Seq=2 Ack=30 Win=10220 Len=0
204	232.761174	192.168.137.101	192.168.137.100	TCP	54	50070 → 8080 [ACK] Seq=30 Ack=3 Win=131328 Len=0

Figura 6.40. Tráfico de datos de la comunicación TCP.

En la figura 6.40, se muestra el tráfico de datos. En la primera columna se define el instante en el que se produce el mensaje. La segunda columna define la fuente, es decir, la dirección IP del emisor del mensaje, y la tercera contiene la dirección IP del receptor del mensaje.

La cuarta columna muestra el protocolo de comunicación utilizado, la quinta columna contiene la longitud del mensaje enviado y la sexta posee la información donde se puede ver el tipo de mensaje enviado.

Atendiendo a la figura 6.40, el recuadro en color rojo muestra los mensajes para establecer la conexión. Se puede observar como en el instante 135.123 el cliente envía un mensaje al servidor (tipo SYN) para establecer una comunicación entre los dos dispositivos, a lo que el servidor contesta con un tipo ACK para confirmar la recepción del mensaje en 2ms.

Como se puede comprobar, para cada mensaje recibido, se obtiene una confirmación de que ese paquete ha llegado al destino.

El recuadro verde de la figura 5.70 muestra la parte de la comunicación en la que se envía el vector de datos al PLC, comienza en el instante 135.126 y termina en el instante 232.759, que es cuando el servidor manda un mensaje al cliente informando que se ha completado el trabajo del robot (el robot ha formado una palabra en 97.663 segundos).

Tras recibir este mensaje, se procede a cerrar la comunicación cuyo intercambio de mensajes se encuentran dentro del recuadro azul.

Otra propiedad importante es la cabecera de TCP, que tiene la forma que se muestra en la figura 6.41.

Offsets Octeto		0								1								2								3							
Octeto	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Puerto de origen																Puerto de destino															
4	32	Número de secuencia																															
8	64	Número de acuse de recibo (si ACK es establecido)																															
12	96	Longitud de Cabecera				Reservado				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Tamaño de Ventana														
16	128	Suma de verificación																Puntero urgente (si URG es establecido)															
20 ...	160 ...	Opciones (Si la Longitud de Cabecera > 5, relleno al final con "0" bytes si es necesario) ...																															

Figura 6.41. Cabecera TCP.

TCP añade 20 bytes de cabecera en cada segmento como mínimo, ya que se dispone de un campo de «opcionales». En esta cabecera TCP se encuentra el puerto de origen y puerto de destino de la conexión (socket), también se encuentra el número de secuencia, número de ACK, y los diferentes FLAGS de TCP como SYN, ACK, RST, FIN, URG y otros.

6.3.2. Intercambio de información en la comunicación TCP.

En una estructura cliente-servidor el cliente es el que inicia la conexión solicitándoselo al servidor, el cual siempre está a la escucha de un cliente.

Cuando desde la interfaz el usuario seleccione una palabra a escribir y el algoritmo del PC calcule las coordenadas de los cubos que tiene que recoger de la plataforma grande, solicitará establecer la comunicación y una vez el servidor la acepte, le pasará un vector de información.

Una vez se envíe dicha información, el PC (cliente) se quedará a la espera de que el PLC (servidor) le indique que se ha terminado de formar la palabra mediante un bit y en ese momento se cerrará la comunicación y el servidor (PLC) volverá a estar a la escucha, como muestra la figura 6.42.

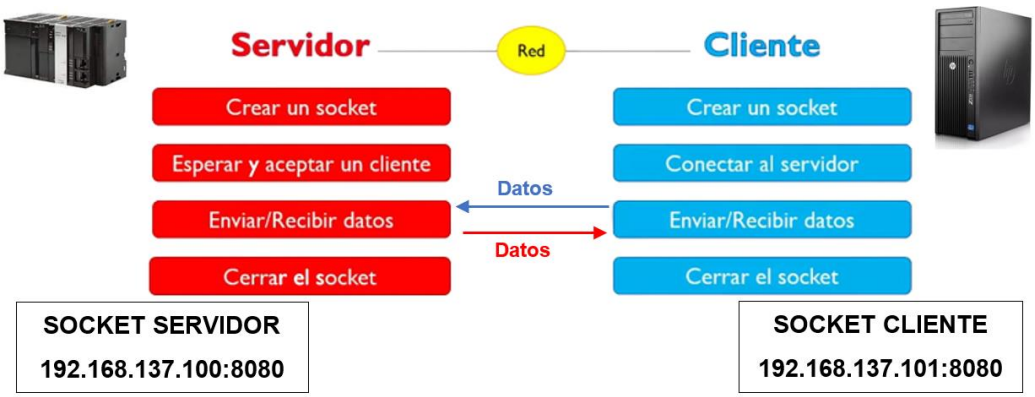


Figura 6.42. Secuencia e intercambio de datos entre el servidor y el cliente.

Para esta comunicación TCP, la información tiene que pasarse en formato de bytes. Un byte está compuesto de 8 bits, entre 0 y 255 en decimal o entre 00 y FF en hexadecimal.

El rango de coordenadas que tiene el robot dentro de su espacio de trabajo va de -170 a +170, pero en formato bytes no se pueden declarar números negativos por lo que cada coordenada va acompañada de un tercer valor que funciona a modo de interprete para que desde el programa del PLC se pueda obtener la coordenada real, como muestra la tabla 6.1.

	COORDENADA X	COORDENADA Y	INTÉRPRETE
Valor	**0 - 255**	**0 - 255**	**0 - 3**

Tabla 6.1. Rango de valores de cada variable.

Los valores del interprete aportan la siguiente información.

- Si vale 0, tanto la coordenada X como la coordenada Y son positivas.
- Si vale 1, la coordenada X es positiva y la coordenada Y es negativa.
- Si vale 2, la coordenada X es negativa y la coordenada Y es positiva.
- Si vale 3, tanto la coordenada X como la coordenada Y son negativas.

Como la longitud máxima de la palabra que se puede escribir es de nueve letras, si se realiza esto para cada letra, el vector de información que se pasa en la comunicación es un vector

de 27 posiciones, más una posición más (28) donde se define la velocidad del robot. Dicho vector se muestra en la figura 6.43.

X1	Y1	I1	X2	Y2	I2	X3	Y3	I3	X4	Y4	I4	X5	Y5	I5	X6	Y6	I6	X7	Y7	I7	X8	Y8	I8	X9	Y9	I9	V

Leyenda:

- **X. Coordenada X.**
- **Y. Coordenada Y.**
- **I. Intérprete.**
- **V. Velocidad.**

Figura 6.43. Esquema del vector de comunicación TCP.

6.4. Visión artificial.

La tecnología de visión artificial proporciona a los equipos industriales la capacidad de "ver" lo que están haciendo y tomar decisiones rápidas en función de lo que ven. Los usos más comunes de la visión artificial son la inspección visual y la detección de defectos, la colocación y medición de piezas, además de identificar, clasificar y rastrear productos.

La visión artificial es una de las tecnologías básicas de la automatización industrial. Ha ayudado a mejorar la calidad de los productos, a acelerar la producción y a optimizar la fabricación y la logística durante décadas. Y ahora, se fusiona con la inteligencia artificial para liderar la transición hacia la industria 4.0.

Para este TFM se ha implementado una aplicación de visión artificial capaz de detectar cada uno de los cubos de la plataforma y distinguir cada una de las letras serigrafiadas en la cara superior.

6.4.1. Disposición de la cámara.

Una decisión importante a la hora de utilizar una cámara para capturar un proceso industrial es la disposición de la misma, ya que hay que procurar capturar toda la zona en la que se quiere trabajar.

En algunas ocasiones no es posible capturar toda la zona deseada con una sola cámara y será necesario colocar varias cámaras y que cada una de ella trate una parte del proceso.

En otras ocasiones no se puede obtener toda la zona de trabajo con una sola imagen, y una solución es tomar varias imágenes con una única cámara eliminando los objetos molestos que se colocan en el medio entre captura y captura para posteriormente juntarlas, tratarlas computacionalmente y obtener la imagen del entorno completa.

Para este trabajo, un gran problema ha sido la colocación de la cámara, ya que, apenas existen posiciones que permitan capturar todos los cubos de la plataforma grande con una sola imagen al encontrarse el robot en el medio, como se muestra en la figura 6.44.

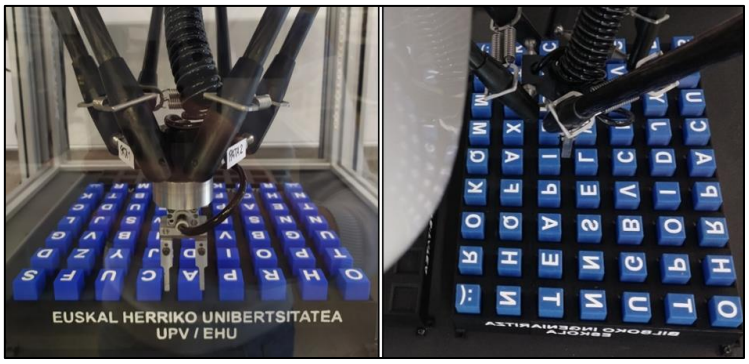


Figura 6.44. Zona de visión de la cámara con el robot interfiriendo.

La disposición ideal de la cámara sería vertical al centro de la plataforma para poder ver sin distorsiones ni sombras todas las letras de la plataforma, pero esto no es posible, ya que tanto las patas del robot como la pinza impiden poder tomar una foto de toda la plataforma ya que se encuentran en mitad del camino.

Una posible solución para utilizar esta disposición sería mover el robot para tomar varias fotografías, combinarlas y obtener una imagen completa, y así capturar todos los cubos. Pero esta opción ha sido desechada debido a que el tiempo de obtención de cada imagen, más el procesamiento, más el de espera de cada movimiento del robot harían que el tiempo de obtención de una imagen procesada con todos los cubos fuera demasiado grande.

En la figura 6.45 se puede comprobar la vista de la cámara desde la parte superior del robot, donde no se puede evitar que alguna pata oculte ciertos cubos de la plataforma.

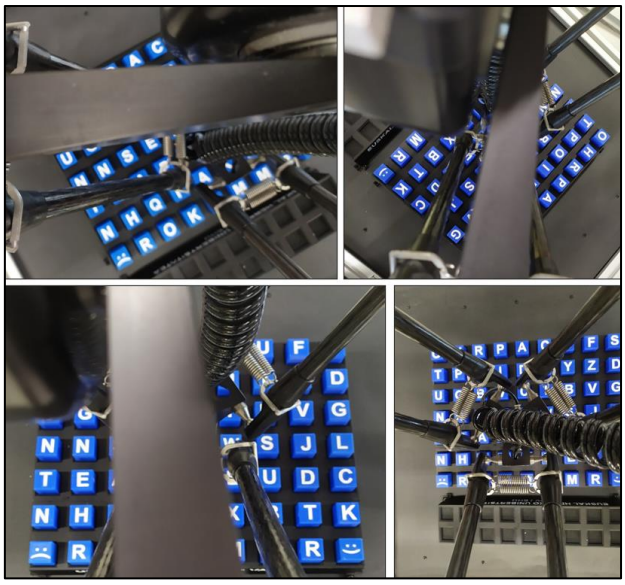


Figura 6.45. Vista de la cámara colocada verticalmente a la plataforma.

Después de varias pruebas, solo se ha encontrado un lugar y una posición que permite obtener con una sola imagen la plataforma grande completa, lo cual reduce enormemente el tiempo de procesamiento respecto a las anteriores opciones comentadas anteriormente.

La disposición de la cámara va a ser oblicua respecto a la base de la plataforma, con un ángulo de 43º y para poder ver toda la plataforma con una sola imagen, después de que el robot termine de formar cada palabra, se moverá a una posición de reposo (no coincide con la posición de reposo del robot) que permita ver toda la plataforma. Dicha posición puede verse en la figura 6.46.

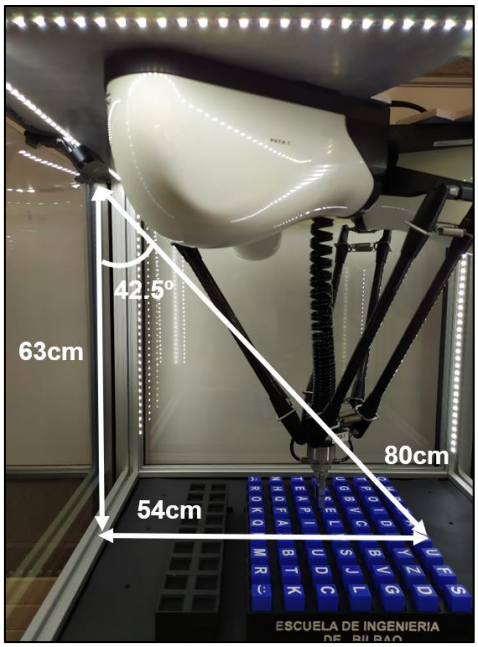


Figura 6.46. Distancias entre la cámara y la plataforma.

En la figura 6.47 se observa la zona de trabajo que se consigue con esta disposición de la cámara y a su derecha la imagen recortada por el programa que identifica la zona de trabajo, que es la plataforma grande.

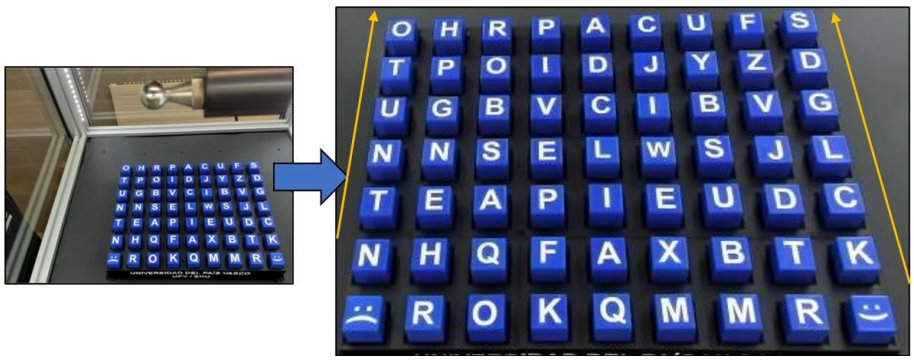


Figura 6.47. Zona de visión del robot original vs zona de trabajo.

La posición oblicua en la que se encuentra la cámara genera cierta distorsión (flechas amarillas), la cual se tratará adecuadamente más adelante.

6.4.2. Iluminación.

La siguiente tarea a realizar consiste en el principal problema de las aplicaciones de visión en un entorno industrial, la iluminación.

Esto es debido a que las condiciones no son nunca homogéneas debido al nivel de luz ambiente, ya que la luz varía a lo largo del día por la iluminación natural o porque en ciertas horas del día se encienden luces de apoyo. Esto hace que la luz varíe en un porcentaje que, de una manera u otra, afecta a los programas de las cámaras de visión artificial.

Para que una aplicación de visión artificial obtenga resultados robustos, los objetos deben iluminarse de la manera más uniforme y constante posible en cualquier momento del día.

Entre los diversos tipos de iluminación para aplicaciones de visión artificial, cabe destacar la iluminación LED, que se ha convertido en el estándar universal para aplicaciones de imágenes. Por esta razón, en este proyecto se han introducido unas tiras led autoadhesivas con longitud adaptable como la que se encuentra en la figura 6.48.

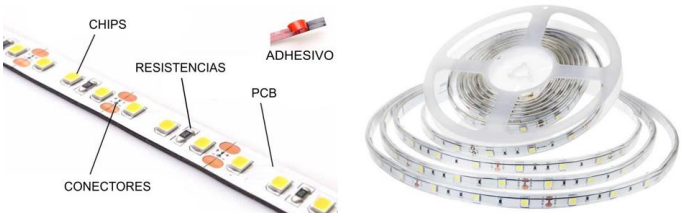


Figura 6.48. Tiras led empleadas.

Observando el esquema de la figura 6.49, se han colocado 8 tiras led en la parte superior del chasis de la célula robotizada para conseguir una iluminación lo más uniforme posible, cuatro de ellas con una longitud de 50cm y otras cuatro con una longitud de 70cm.

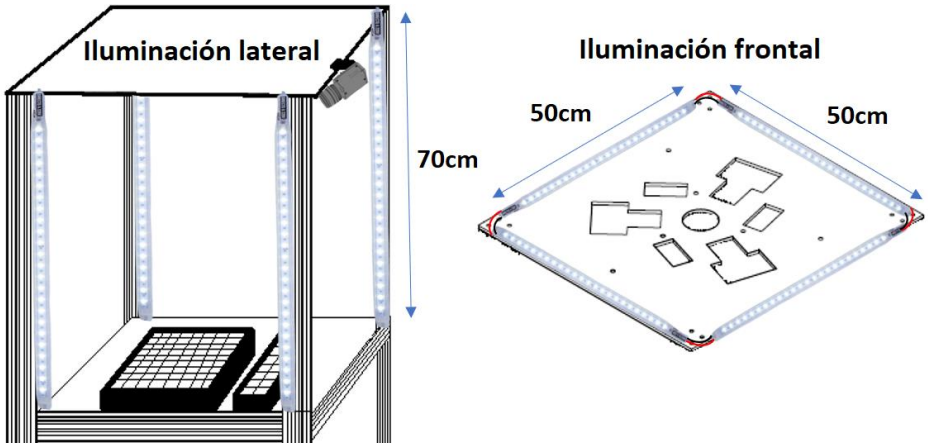


Figura 6.49. Esquema de las tiras led dentro de la célula.

A continuación, en la figura 6.50, se muestra un esquema de conexión de cada tira led para ser alimentada con la fuente de alimentación.

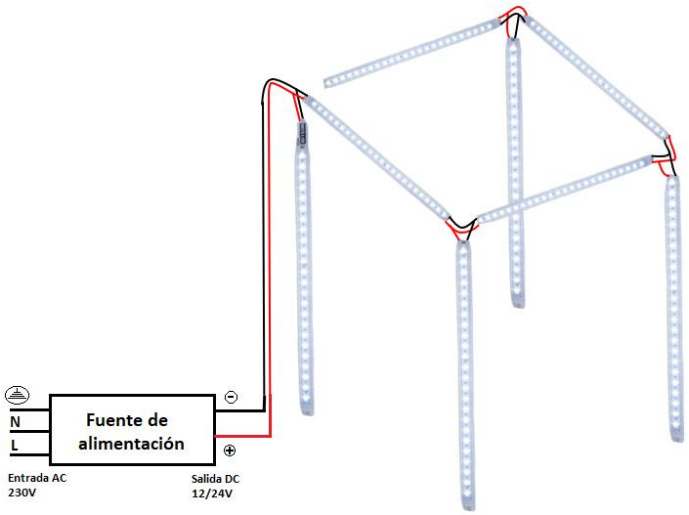


Figura 6.50. Esquema de conexiones de cada tira led.

En la figura 6.51 se puede observar el resultado real, donde se muestra en color rojo los leds de la iluminación frontal y en color naranja los leds de la iluminación lateral.

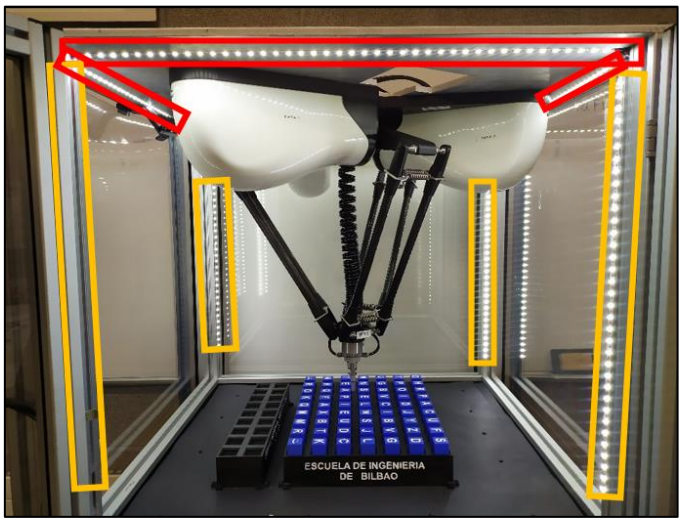


Figura 6.51. Iluminación real de la célula robotizada.

El propósito de un sistema de iluminación en aplicaciones de visión no es otro que controlar la forma en la que la cámara verá el objeto. Por esta razón y tal y como se puede observar en la figura 6.52, se ha optado por una iluminación frontal y lateral.

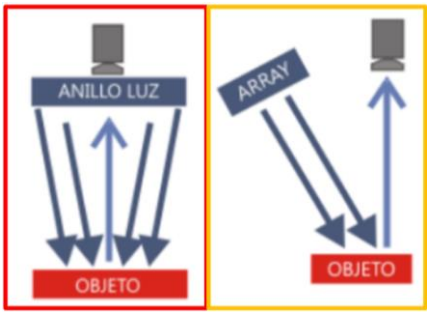


Figura 6.52. Tipo de iluminación propuesto.

En la iluminación frontal, la cámara se posiciona mirando al objeto en la misma dirección que la luz. Esto reduce las sombras, suaviza las texturas y minimiza la influencia de rayas, polvo e imperfecciones que pueda tener el objeto. La cámara recibe la luz reflejada del objeto.

La gran ventaja que aporta es que elimina sombras y se puede utilizar a grandes distancias cámara/objeto como es este caso. Pero tiene un gran inconveniente y es que produce intensos reflejos sobre superficies reflectantes y la base donde se apoya la plataforma es de aluminio, lo cual produce grandes reflejos y brillos.

Por lo que, para solucionar este problema se ha colocado como fondo una lámina negra mate que elimina dicho brillo.

Esta iluminación frontal, se ha combinado con una iluminación lateral cuya principal ventaja es resaltar los bordes, rayas y fisuras que, para esta aplicación, permite resaltar con más claridad las letras.

Finalmente, con este sistema de iluminación, se consiguen eliminar todas las zonas oscuras del espacio de trabajo de la cámara que dificultan la detección y diferenciación de las letras de los cubos y paralelamente, se solucionan los problemas de brillos producidos por los rayos del sol en los días soleados que incidían dentro de la célula y provocaban brillos indeseados en la cámara.

En la figura 6.53 se muestra en la parte izquierda la imagen que obtiene la cámara sin iluminación en su interior y en la parte derecha con la iluminación instalada.

Figura 6.53.. Imagen de la webcam sin iluminación y con iluminación.

6.4.3. Calibración de la cámara.

Con la posición de la cámara fija y la iluminación corregida, la siguiente tarea consiste en realizar la calibración de la cámara. Para la calibración se suele emplear patrones ya preestablecidos y para este proyecto se ha optado por utilizar un tablero de ajedrez debido a su sencillez.

La imagen a modo de patrón que se utiliza es un tablero de ajedrez (figura 6.54) que se ha colocado en la base de la plataforma y se conoce cuál es el tamaño de la cuadrícula del tablero (30x30mm).

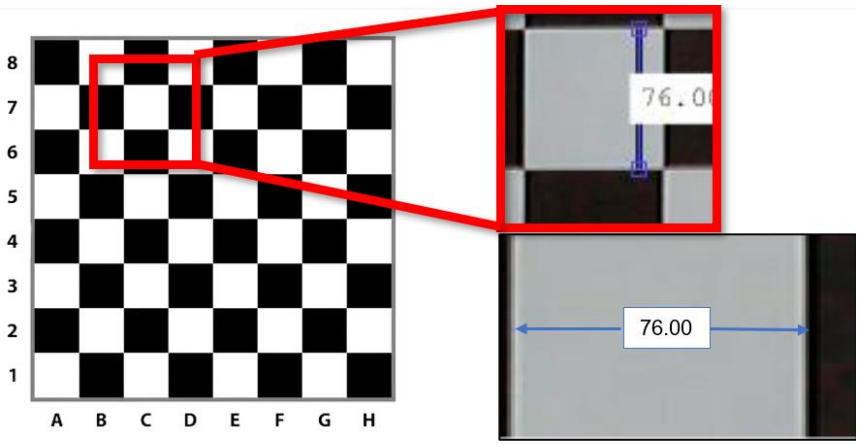


Figura 6.54. Tablero de ajedrez como patrón con medidas de pixeles.

Del programa se extrae el ancho y alto en píxeles de cada cuadricula, que tienen que ser iguales y conociendo los milímetros de cada cuadricula y los pixeles que conforman esa cuadricula, se puede obtener un parámetro para conocer la relación mm/píxel, como muestra en la expresión 6.1.

$$\text{alfa} = \frac{30\text{mm}}{76\ \text{píxeles}} = 0.3947 \qquad (6.1)$$

Para los dos últimos apartados *(6.4.3. Calibración de la cámara.* y *6.4.4. Preprocesado de la imagen.*) de iluminación y calibración existen gran cantidad de empresas que se dedican única y especialmente a estas dos actividades debido a su gran dificultad y su importancia para que un sistema de visión artificial pueda funcionar correctamente.

6.4.4. Preprocesado de la imagen.

En este apartado se tratan los distintos modelos de color utilizados para obtener la imagen con la que trabajar y las operaciones morfológicas y de segmentación utilizadas para obtener una imagen final óptima con la que trabajar.

Con la calibración terminada, la siguiente tarea a realizar ha sido obtener una imagen con la que poder trabajar y extraer las características deseadas a partir de la imagen capturada por la cámara que se encuentra en formato RGB.

El término RGB se refiere a un espacio de color que reproduce todos los colores visibles para los humanos mediante la mezcla aditiva de los tres colores primarios. El espacio de color RGB recibe, por tanto, el nombre de estos tres colores; rojo, verde y azul y su modelo de color se puede ver en la figura 6.55.

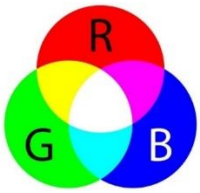


Figura 6.55. Modelo de color RGB.

El siguiente paso es convertir la imagen actual que se encuentra en el espacio de trabajo RGB a escala de grises para posteriormente convertirla a una imagen binaria en la que el espacio de colores es blanco y negro (binario) y es donde se realiza la diferenciación de cada una de las letras.

Una imagen en escala de grises indica el valor que cada píxel posee equivalente a una graduación de gris. Es una imagen cuyos pixeles van del negro (valor 0) al blanco (valor 255) pasando por varios tonos del gris.

Los pixeles únicamente contienen intensidades de luz, es decir, se ha eliminado toda la presencia del color, como muestra la figura 6.56, que es el modelo de color de escala de grises.

Figura 6.56. Modelo de escala de grises.

En la figura 6.57 se muestra la imagen en escala de grises junto a su histograma que es una representación gráfica en forma de barras, que simboliza la distribución del valor de cada pixel de la imagen. Sirve para obtener un panorama de la distribución de la población.

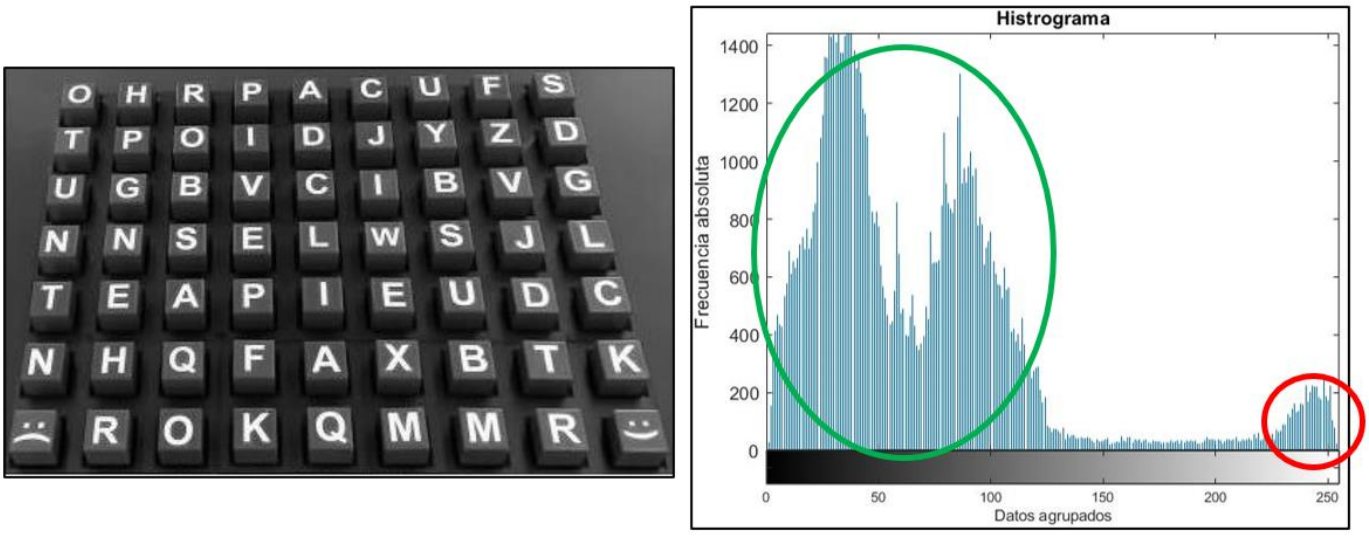


Figura 6.57. Imagen de la plataforma en escala de grises junto con su histograma.

Analizando la imagen y el histograma, se puede observar que las letras que están escritas en blanco (color rojo) que es lo que se quiere diferenciar del resto de la imagen (background, color verde) se encuentran agrupadas en la parte derecha del histograma.

Con el objetivo de poder diferenciar las letras del resto de la imagen, se trabaja en el espacio de colores binario.

Una imagen binaria es una imagen digital que tiene únicamente dos valores posibles para cada píxel, siendo estos, negro (valor 0) y blanco (calor 1), y así poder separar cada letra del resto de la imagen. De esta manera el fondo se mantendrá de color negro y las letras en color blanco.

6.4.4.1. Segmentación y operaciones morfológicas.

La segmentación se lleva a cabo empleando el método de umbralización por histograma. Para ello, se halla el umbral de forma automática y se procede a eliminar los datos que corresponden al fondo, de forma que solo queden los datos del objeto.

La imagen resultante (ver Figura 6.58) dista mucho de la imagen que se quiere, ya que contiene ruido y pérdida de información, por lo que es necesario hacer un acondicionamiento de la misma mediante operaciones morfológicas para corregirla.

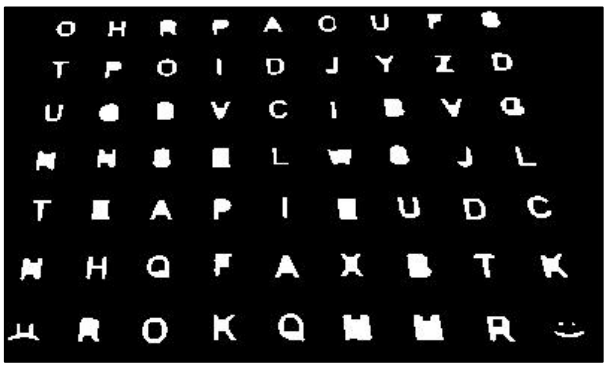


Figura 6.58. Imagen binaria original de la plataforma.

El objetivo de las transformaciones morfológicas es la extracción de estructuras geométricas en los conjuntos sobre los que se opera, mediante la utilización de otro conjunto de forma conocida denominado elemento estructurante. El tamaño y la forma de este elemento se escoge de acuerdo a la extracción de formas que se desean obtener. Estas operaciones permiten:

- Preprocesamiento de imágenes (supresión de ruido, simplificación de formas).
- Resaltar la estructura de los objetos (extraer el esqueleto, detección de objetos, ampliación, reducción, ...).
- Descripción de objetos (área, perímetro, ...).

Las operaciones morfológicas primarias son la erosión y la dilatación y a partir de estas, se componen otro tipo de operaciones como la apertura y cierre.

- Dilatación. La dilatación es la transformación morfológica que combina dos vectores utilizando la suma y agrega píxeles a los límites de los objetos en una imagen.

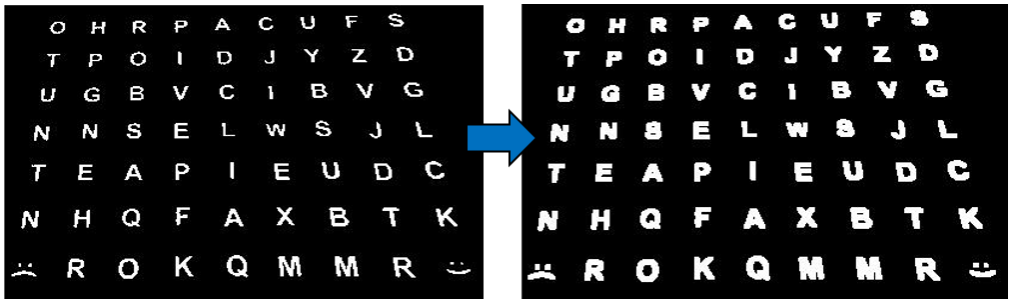


Figura 6.59. Representación de la operación morfológica de dilatación.

- Erosión. Es la operación contraria a la dilatación. La erosión es la transformación morfológica que combina dos vectores utilizando la resta, es decir, elimina píxeles en los límites de los objetos

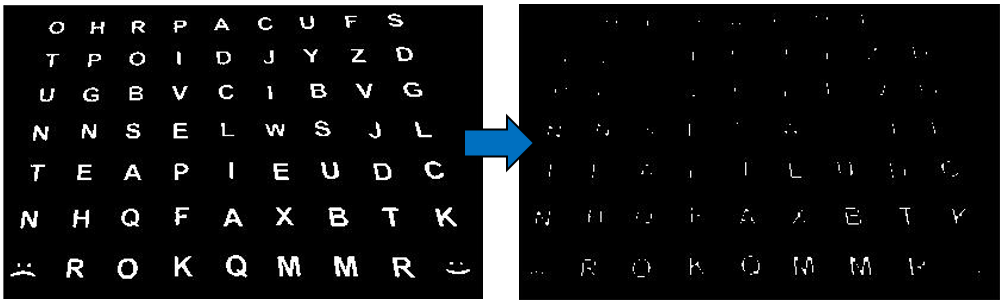


Figura 6.60. Representación de la operación morfológica de erosión.

- Apertura. La apertura es la dilatación de la erosión de un conjunto, es decir, una apertura consta de una erosión y posteriormente una dilatación.

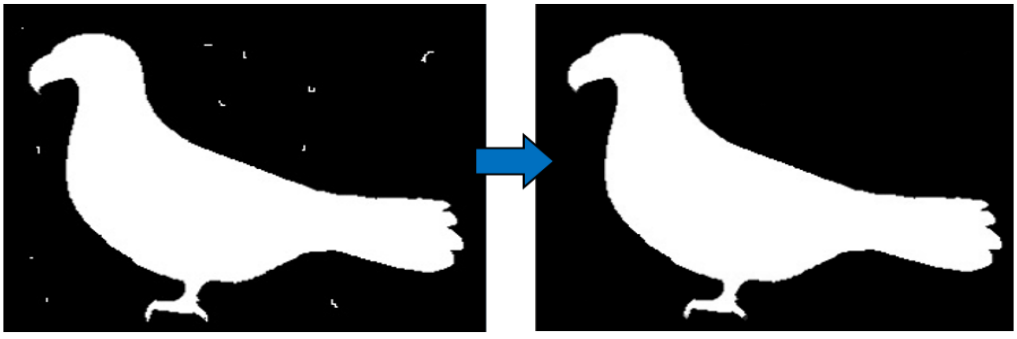


Figura 6.61. Representación de la operación morfológica de apertura.

- Cierre. El cierre es una operación morfológica que permite redondear las concavidades importantes y unir elementos cercanos. Consiste en una dilatación seguida de una erosión.

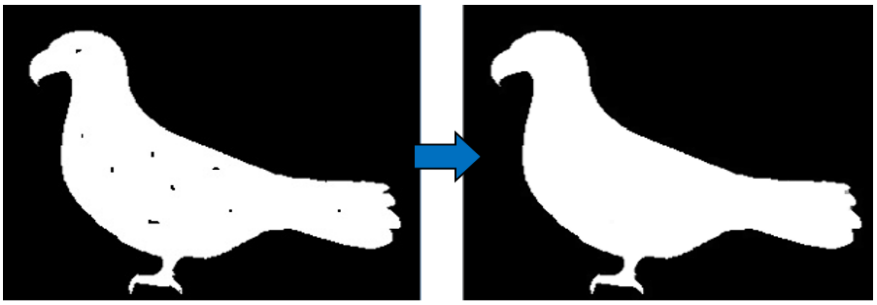


Figura 6.62. Representación de la operación morfológica de cierre.

Tras una serie de transformaciones morfológicas (ver anexo, III.XI. Visión artificial.) donde se han realizado dos aperturas, una erosión y una dilatación, se consigue obtener cada una de las letras claras y como un único objeto, se obtiene el resultado de la figura 6.63 en forma de imagen binaria, donde en la parte izquierda se puede observar la imagen original sin transformaciones y en la parte derecha la imagen con las transformaciones.

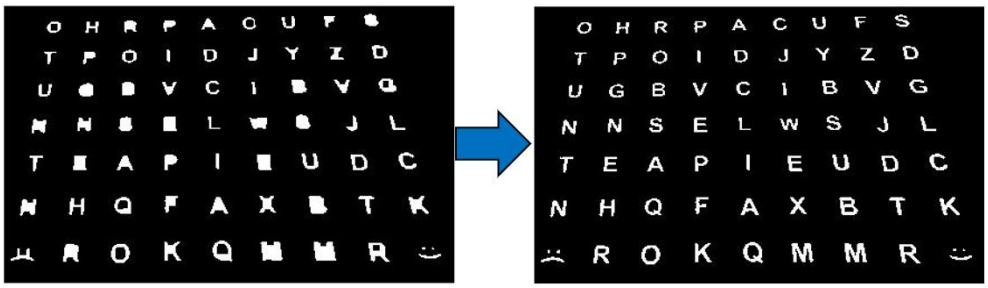


Figura 6.63. Imagen binaria original y resultante obtenida.

6.4.5. Extracción de características geométricas.

En este apartado se trata el reconocimiento de las letras, que es la siguiente tarea a realizar para extraer la información deseada de la imagen

Por lo que la siguiente tarea a realizar ha sido un programa capaz de extraer diversas características geométricas de cada letra. Este programa de visión artificial está implementado como una función dentro de la interfaz, de manera que cada vez que se quiera formar una palabra, se saque una foto mediante la webcam y se detecte donde se encuentra cada cubo y letra.

Para poder diferenciar correctamente cada letra se han calculado las siguientes características geométricas de cada objeto, las cuales consiguen establecer unas relaciones que permiten diferenciar cada una de las letras de la plataforma:

- Área.
- Centroide.
- Circularidad.

- Anchura.
- Altura.
- Perímetro.

Lo primero que se calcula es el centroide de cada letra, con su coordenada ‘X’ e ‘Y’ asociada a cada objeto. Al haber 63 cubos, tienen que reconocerse 63 objetos.

Figura 6.64. Representación del centroide de cada letra.

En la figura 6.64 se muestra el centroide de cada letra sobre la imagen y en la tabla 6.2 se muestra las coordenadas ‘X’ e ‘Y’ del centroide de cada objeto.

Conociendo el centroide es posible definir la posición que va a ocupar cada letra dentro de la plataforma, ya que esta se puede asemejar a una matriz de 9 columnas y 7 filas.

Si no se produjera la distorsión, cada cubo de la misma columna tendría un valor de ‘X’ similar, pero no es así, por lo que, para poder colocar cada elemento dentro de la matriz, cada elemento se ha ordenado por filas, ya que en este caso sí que tiene una coordenada ‘Y’ similar y dentro de los elementos que comparten esa coordenada ‘Y’, se ordenan de izquierda a derecha a medida que va aumentando la coordenada ‘X’ y de esta manera se consigue rellenar la matriz de letras.

Una vez se tiene definido cada objeto dentro de la matriz, el siguiente paso es extraer la información de cada objeto para diferenciarlo del resto de letras y así obtener la letra de ese cubo.

Object	Coordenada X	Coordenada y
1	16,94782609	222,1913043
2	22,44736842	180,6403509
3	28,54716981	139,5471698
4	31,59550562	107,741573
5	37,01694915	76,3559322
6	41,35294118	45,76470588
7	45,49019608	19,50980392
8	60,512	221,144
9	65,81188119	179,5445545
10	67,55952381	141,1547619
11	72,48780488	106,5243902
12	74,07042254	76,14084507
13	76,05882353	45,8627451
14	79,39622642	19,47169811
15	104,593985	222,1503759
16	107,2761905	179,5142857
17	109,0352941	141,3294118
18	109,5789474	106,4736842
19	113,015873	45,41269841
20	111,597561	74,6097561
21	113,5322581	18,09677419
22	148,6607143	73,71428571
23	148,0188679	17,39622642
24	148,1724138	106,1149425
25	149,1216216	139,0135135
26	148,5	176,525641
27	150,3902439	220
28	148,03125	44,84375
29	183,75	17,19230769
30	185,2295082	44,31147541
31	186,8275862	73,44827586
32	186,4285714	105,7142857
33	193	179,7425743
34	196,6052632	220,1184211
35	192	139,5
36	219,6545455	16,32727273
37	224,9459459	44,86486486
38	229,2236842	105,3421053
39	224,5517241	74,06896552
40	232,9693878	140,4285714
41	236,4343434	178,2323232
42	240,197861	219,6096257
43	256	15,37931034
44	259,4782609	42,06521739
45	264,6091954	72,56321839
46	268,5822785	104,5822785
47	275,3877551	139,9183673
48	281,1830065	178,6013072
49	287,8088235	219,9705882
50	290,8205128	13,66666667
51	298,4915254	43,44067797
52	304,4637681	72,07246377
53	314,1632653	106,8367347
54	319,0098039	140,4901961
55	326,202381	176,3571429
56	329,7936508	13,38095238
57	335,2689655	220,1655172
58	337,0759494	41,64556962
59	343,9411765	71,95294118
60	351,3035714	106,875
61	361,3541667	139,2604167
62	371,2892562	178,7520661
63	381,859375	221,2265625

Tabla 6.2. Coordenadas de los centroides de cada letra.

Para ello, se utilizan las características geométricas comentadas anteriormente cuyo resultado para cada objeto se encuentra en la tabla 6.3.

Object	Area	MajorAxisLength	MinorAxisLength	Circularity	Perimeter
1	115	24,73125202	12,55260714	0,247421143	76,425
2	114	17,4830826	15,57156241	0,2586984	74,415
3	53	16,94016222	8,159029799	0,350760711	43,575
4	89	15,96685997	13,29506476	0,274513552	63,829
5	59	16,24067212	13,37570686	0,25098444	54,351
6	34	13,96899132	7,479917995	0,295355306	38,034
7	51	15,64634508	12,2671483	0,597526859	32,75
8	125	18,36505503	16,73706487	0,38675153	63,73
9	101	19,10660542	14,72575633	0,239032822	72,868
10	84	17,15859594	13,04934992	0,206606683	71,478
11	82	15,76454676	12,51680549	0,259812343	62,977
12	71	16,05946726	14,00694175	0,179536016	70,495
13	51	15,72917705	9,079222547	0,52968905	34,784
14	53	16,01727043	10,8629677	0,253490927	51,258
15	133	21,14215177	19,83882463	0,640233947	51,093
16	105	18,93540919	17,36876891	0,541826262	49,348
17	85	14,58756198	13,838418	0,490781582	46,652
18	76	15,32825833	12,69978624	0,203869703	68,444
19	63	16,62369792	13,30125997	0,602801099	36,24
20	82	14,62981686	13,42721463	0,674257394	39,093
21	62	13,41451895	11,33518146	0,441065856	42,029
22	56	13,54955462	12,49855957	0,323227589	46,66
23	53	14,18512255	8,755980356	0,600183868	33,312
24	87	16,95150433	13,40632697	0,213441674	71,569
25	74	16,40018798	12,083106	0,515630288	42,467
26	78	17,23027525	10,96986165	0,289901085	58,147
27	123	19,10433657	16,88610449	0,236312333	80,875
28	32	12,38909297	3,430858783	0,793542637	22,511
29	52	12,31271975	10,52153279	0,507697678	35,876
30	61	16,02471954	12,51378157	0,607566451	35,52
31	58	15,54124416	14,06134469	0,234856575	55,708
32	35	16,21997542	7,283436818	0,313514986	37,455
33	101	17,11083138	15,2703066	0,464738924	52,259
34	152	23,20977067	19,04545733	0,568312122	57,974
35	40	15,51294473	3,408109813	0,65293371	27,746
36	55	14,52349232	12,61034448	0,260691439	51,49
37	37	14,10364058	7,155336423	0,414455165	33,494
38	76	18,39290478	10,95492491	0,183723612	72,099
39	29	13,51774605	3,20952661	0,651877454	23,644
40	98	18,38427183	13,44439865	0,213851656	75,886
41	99	18,16396748	14,17375403	0,254495137	69,917
42	187	22,51687724	18,54071262	0,169127967	117,874
43	58	15,93010551	12,10769936	0,282485349	50,795
44	46	12,19226037	10,15663481	0,420152734	37,092
45	87	16,28070755	13,86531849	0,645575771	41,152
46	79	16,56523811	12,82964859	0,177727246	74,738
47	98	19,27298792	15,58558303	0,275867789	66,814
48	153	20,54736128	16,01902431	0,679070363	53,21
49	204	23,2206393	18,56539476	0,182423539	118,544
50	39	11,71941187	8,45216424	0,288469991	41,218
51	59	15,39043587	11,09197147	0,238623034	55,741
52	69	15,2044619	12,10918502	0,333874362	50,961
53	49	15,45573094	8,590181188	0,374808884	40,532
54	102	20,17483411	16,34290794	0,595686096	46,387
55	84	19,79981384	10,57106509	0,383603226	52,457
56	63	15,14250269	10,65265764	0,190255282	64,507
57	145	21,50684748	17,07717224	0,34006012	73,2
58	79	17,09060122	14,06386309	0,630512395	39,68
59	85	19,23323393	13,98822176	0,19742602	73,555
60	56	19,87226542	8,402777662	0,35988224	44,22
61	96	21,09451152	14,71154547	0,261870758	67,873
62	121	19,7914359	15,40791409	0,258419474	76,707
63	128	21,17886244	14,35652316	0,273134301	76,74

Tabla 6.3. Características geométricas de cada letra.

Tras muchas simulaciones y pruebas se ha conseguido obtener una relación de parámetros que permite esta diferenciación. Por ejemplo, para diferenciar la letra 'I' sobre las demás, esta es la letra que tiene menor ancho (Major Axis Length).

Esta misma regla se aplica para diferenciar la 'L', 'T' y 'J' combinando el ancho y largo de la letra (Minor Axis Length).

Para letras con cierta circularidad como la 'O', la 'Q', la 'D', la 'C', la 'S' o la 'B' se utiliza la propiedad 'Circularity'. La circularidad mide la redondez de los objetos (para un círculo perfecto, el valor de circularidad es la unidad), y se calcula a partir de la expresión 6.1:

$$\text{Circularidad} = \frac{(4 * \text{Área} * \text{pi})}{\text{Perimetro}^2} \tag{6.1}$$

Las letras 'W' y 'M' tienen un perímetro mucho más grande que las demás y se pueden clasificar atendiendo a este parámetro. Otras letras se diferencian atendiendo a un parámetro o la mezcla de varios parámetros para identificarlas y diferenciarlas.

De esta manera se comprueba letra por letra si se encuentra dentro de ese rango de características geométricas que define una letra, para así diferenciarlas todas.

El concepto de esta técnica se puede asemejar a la técnica de reconocimiento de patrones, pero con la diferencia de que, en vez de comparar imágenes o partes de imágenes, lo que se comparan en este caso son valores numéricos asociados a características geométricas que permite conseguir un coste computacional menor.

Utilizando este método se consigue diferenciar cada una de las letras y en la figura 6.66 se muestran los resultados teniendo en la parte izquierda la imagen original obtenida por la webcam y en la parte derecha la imagen procesada con el programa de visión artificial que muestra una pegatina roja con cada letra que detecta para cada cubo.

Paralelamente, en este programa de visión, se utiliza el parámetro de conversión de pixeles a milímetros para mostrar por pantalla (figura 6.65), la clase de letra detectada y la posición de su centroide en milímetros.

```
--------------------------------
    Class: G
    Position: (136, 28) mm
--------------------------------
--------------------------------
    Class: L
    Position: (139, 42) mm
--------------------------------
--------------------------------
    Class: C
    Position: (143, 55) mm
--------------------------------
--------------------------------
    Class: K
    Position: (147, 71) mm
--------------------------------
--------------------------------
    Class: X
    Position: (151, 87) mm
--------------------------------
```

Figura 6.65. Mensaje mostrado por pantalla con la clase de letra y su posición en milímetros.

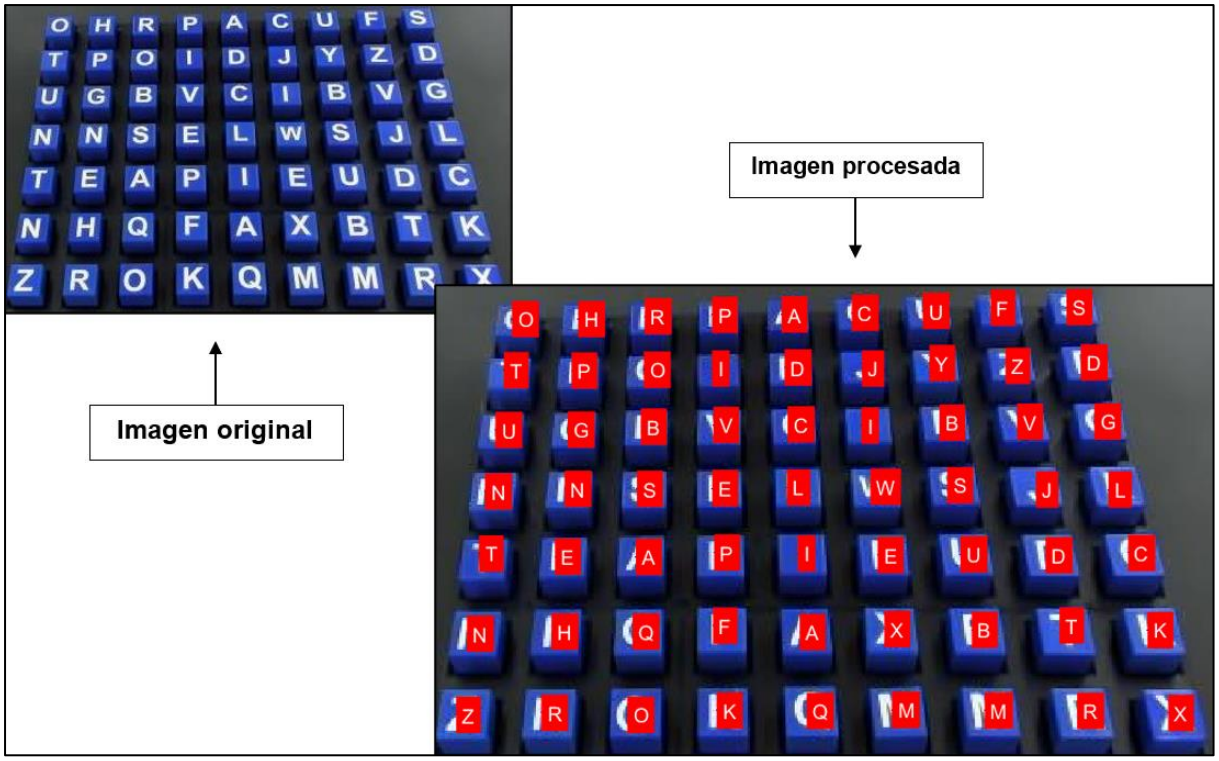


Figura 6.66. Resultado obtenido mediante el programa de visión artificial.

Para demostrar que el resultado es robusto, se muestran los resultados obtenidos con distintas configuraciones de letras en las figuras 6.67 y 6.68. Y como se puede comprobar en cada una de ellas, las pegatinas de la imagen procesada muestran la letra identificada correctamente.

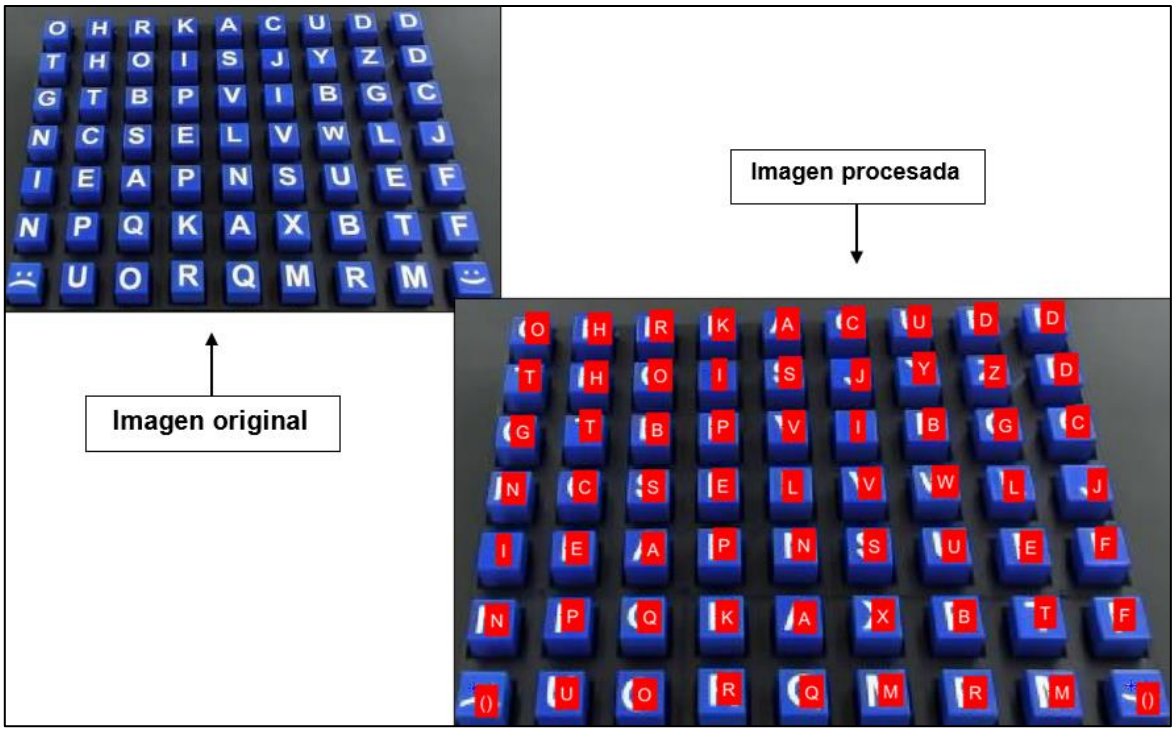


Figura 6.67. Resultado obtenido mediante el programa de visión artificial para otras distribuciones de letras.

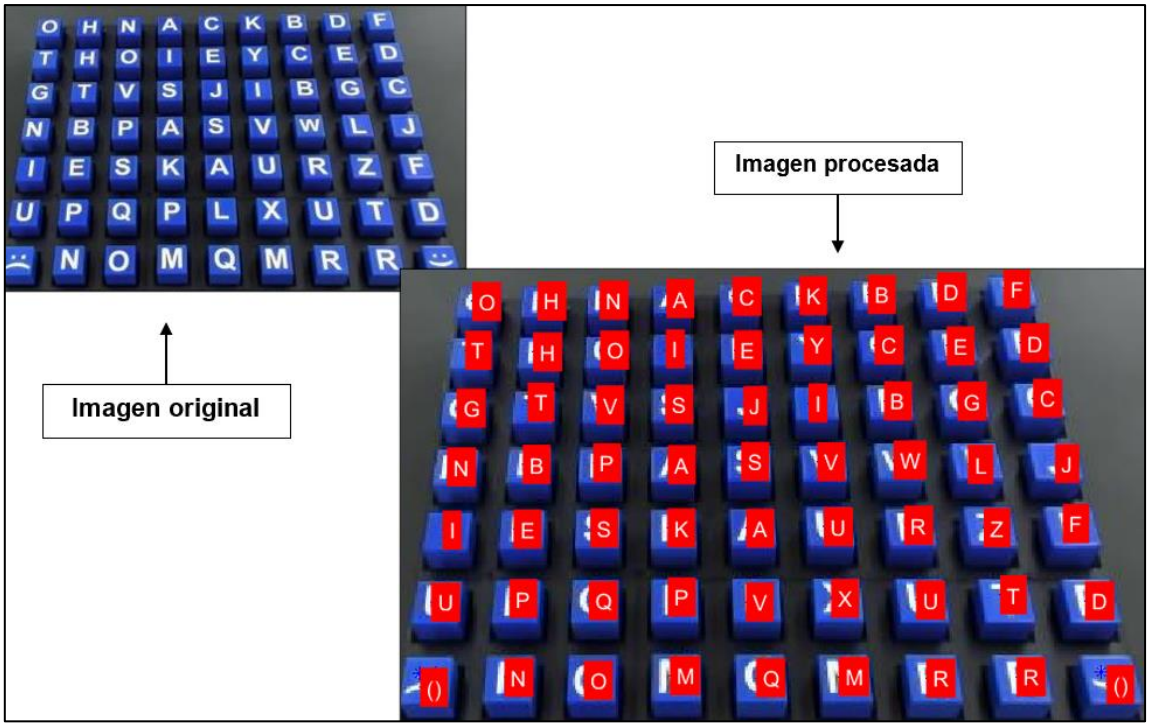


Figura 6.68. Resultado obtenido mediante el programa de visión artificial para otras distribuciones de letras.

6.5. Extracción de cubos y comunicación.

La siguiente tarea a realizar consiste en realizar el algoritmo que es capaz de extraer a partir de los resultados obtenidos del programa de visión artificial las coordenadas donde se encuentra cada cubo que hay que extraer para formar la palabra deseada.

Para ello, se hace un barrido fila por fila de la matriz, hasta encontrar la letra deseada. Por ejemplo, si se quiere formar la palabra ‘robot’, se empezará buscando la letra ‘r’, cuando se encuentre, se guardará la posición fila y columna que ocupa en la matriz y se pasará a buscar la siguiente letra hasta completar la palabra.

Para este ejemplo y con la distribución de letras de la figura 6.66, se obtiene que las letras a recoger se encuentran en las coordenadas de la tabla 6.4 y al mismo tiempo la interfaz muestra visualmente en la figura 6.69, donde se encuentra cada letra que va a recoger de la plataforma grande y como va a quedar expuesta en la plataforma pequeña.

	LETRA 1 (R)	LETRA 2 (O)	LETRA 3 (B)	LETRA 4 (O)	LETRA 5 (T)
FILA	7	7	3	2	2
COLUMNA	8	3	3	3	1

Tabla 6.4. Posición de cada letra dentro de la matriz a recoger.

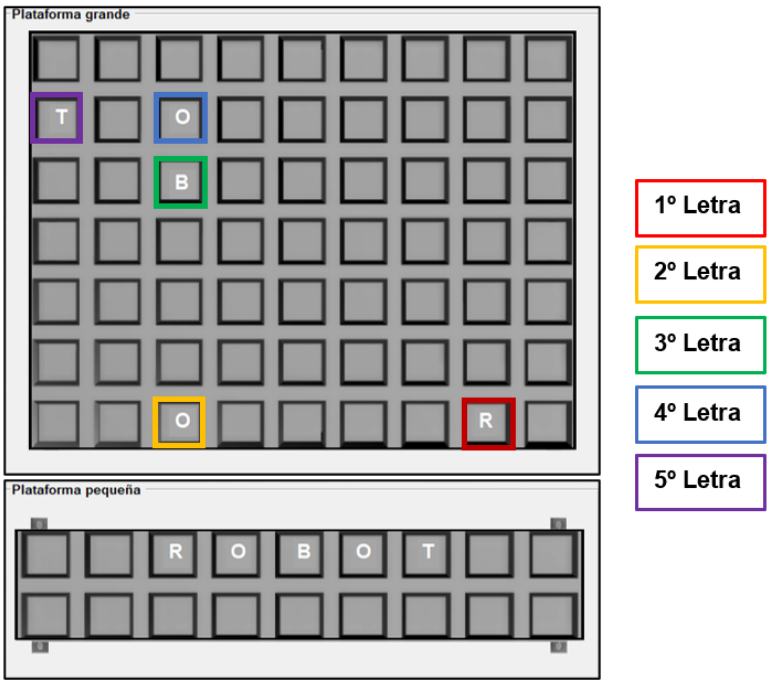


Figura 6.69. Representación del lugar donde se va a recoger y depositar cada letra en la interfaz.

Con las posiciones identificadas, el siguiente paso es convertir dichas posiciones a las coordenadas del robot y mandárselas al PLC que gobierna al robot mediante la comunicación TCP. Para lo cual es necesario convertir esas coordenadas a datos en formato byte que posteriormente el PLC acondiciona para obtener la coordenada real deseada.

Como se ha comentado anteriormente en el apartado de la comunicación TCP, el programa que permite comunicar el PC con el PLC se encuentra desarrollado en Python, ya que esta herramienta permite que la comunicación sea inmediata, al contrario que ocurría empleando Matlab que el tiempo medio de conexión se situaba en 10 segundos.

Python [57] es un lenguaje de programación de alto nivel, orientado a objetos, con una semántica dinámica integrada, principalmente para el desarrollo web y de aplicaciones informáticas.

Figura 6.70. Icono de Python.

De manera que cuando se quieren enviar los datos, Matlab ejecuta una función de Python que envía el vector de comunicación explicado anteriormente, a la dirección y puerto del PLC y hasta que no se active un bit que envía el PLC cuando se ha completado el proceso del robot no se completará dicha función, es decir, el programa se queda a la espera de que el PLC indique que el proceso ha terminado para cerrar la comunicación y poder formar otra palabra.

En ocasiones no se pueden escribir todas las palabras, ya que depende de la cantidad de letras de cada tipo que se haya colocado, es decir, si se necesitan más letras de un tipo de las que tiene la plataforma, la interfaz avisará de que no es posible formar dicha palabra y permitirá escribir otra.

Otra limitación es el número de letras, ya que no se pueden formar palabras que tengan más de nueve letras ni tampoco palabras que contengan la letra 'ñ'.

A modo de easter egg, se ha programado una función que utiliza las caras triste y sonriente de la plataforma y las coloca al final de las palabras para un conjunto de palabras determinadas. Esta función puede verse en el anexo (*III.III. Escoger cara.*).

6.6. Normativa de seguridad.

En este apartado se plantea la normativa actual referente a la seguridad ISO aplicada a los sistemas robóticos en el marco de la Unión Europea para posteriormente demostrar que tanto los sistemas de seguridad de la célula como diseño del tamaño del chasis teniendo en cuento el espacio de trabajo del robot cumplen la normativa.

Se comienza explicando la norma UNE-ISO 10218. Esta norma trata de dar respuesta a los riesgos que presentan los robots y los sistemas robóticos industriales, dando así unas directrices y criterios a seguir, para así minimizar al máximo los peligros que puede suponer la aplicación de uno de estos sistemas en un entorno industrial.

El número y tipo de riesgos producidos por estos sistemas están relacionados con el proceso de automatización (tipo de robot y función) y con la manera en que está instalado, programado, operado y mantenido.

Haciendo así, que para cada sistema robótico existan diferentes riesgos a tener en cuenta. Es por todo ello, que esta primera norma se divide en dos partes, las cuales se trataran en los dos siguientes apartados.

6.6.1. UNE-ISO 10218-1:2012.

En esta primera parte de la norma [58], se especifican los requisitos e indicaciones a seguir para garantizar la seguridad en el diseño y construcción del robot. También, se describen los riesgos asociados a los robots y se dan las pautas, medidas de protección e información de uso, para eliminar o reducir dichos riesgos.

Esta norma está destinada a los fabricantes de robots, a los cuales proporciona una herramienta muy útil para evaluar todos los peligros que podría suponer el robot

6.6.2. UNE-ISO 10218-2:2011.

La segunda parte de la norma [59] trata los peligros derivados del diseño e integración de los sistemas robóticos industriales en celdas industriales robotizadas y líneas de producción. Igual que en la primera parte, también se proporcionan directrices, pero en este caso enfocadas a garantizar la seguridad del personal hacía todo lo que tiene que ver con la integración del robot.

Mientras la primera parte estaba más enfocada a los fabricantes de robots, ésta segunda, afecta a todos los "participantes" en la integración de un sistema de este tipo. En la misma norma, se especifican como "participantes": el fabricante, el suministrador, el integrador y el usuario final.

Según la norma UNE-ISO 10218-2:2011, una célula robotizada es aquella que se compone de uno o más sistemas robóticos incluyendo la maquinaria correspondiente, el espacio de seguridad y las medidas de protección correspondientes.

Se puede describir la celda robotizada como aquella área restringida donde trabaja el robot y quedan recogidos todos los elementos necesarios para que trabaje. Tal y como indica la definición dada en la norma, la celda incluye el espacio de seguridad y los elementos de protección.

Este aspecto es muy importante, ya que dentro de la celda el robot ejecuta movimientos a alta velocidad, con lo cual, cualquier contacto con una persona podría resultar fatal (aplastamiento, golpes, proyección a alta velocidad, etc.). Por lo tanto, el límite físico de la celda robotizada se corresponde con el espacio de seguridad del robot.

6.6.3. Diseño del sistema de seguridad de la celda.

La siguiente tarea a realizar consiste en demostrar que todo el diseño de la célula y su seguridad cumple con las normativas referentes sistemas robóticos en el marco de la Unión Europea.

Para este proyecto ha sido muy importante que, tanto durante todo el proceso de trabajo como para el futuro uso del robot, este sea seguro ante condiciones adversas, es por ello que la célula robotizada cuenta con un circuito de seguridad.

Tal y como se ha comentado anteriormente, una celda robotizada no se podría entender sin su espacio de seguridad y las medidas de protección correspondientes, ya que el robot puede suponer un peligro potencial para las personas y para el entorno.

Todas estas medidas deben cumplir estrictamente las diferentes normativas recogidas en el estándar UNE-ISO 10218-2:2011, ya que es la única manera de asegurar que se han evaluado todos los peligros existentes, y que, por lo tanto, se cumplen con todos los requisitos de seguridad.

Para el diseño del sistema de seguridad de las celdas, es necesario tener en cuenta tanto los requisitos del sistema (de modo que se puedan satisfacer lo máximo posible), como el cumplimiento de las normas.

Aunque el espacio de trabajo del robot este formado por un cilindro y un cono, se define que el espacio de trabajo del robot como todo el lugar del habitáculo superior que puede ocupar.

A partir de aquí, su protección está formada por una cabina formada por un chasis de perfiles de aluminio recuadrados en color rojo en la figura 6.71 y unos paneles de metacrilato en cada pared, señalados en color naranja.

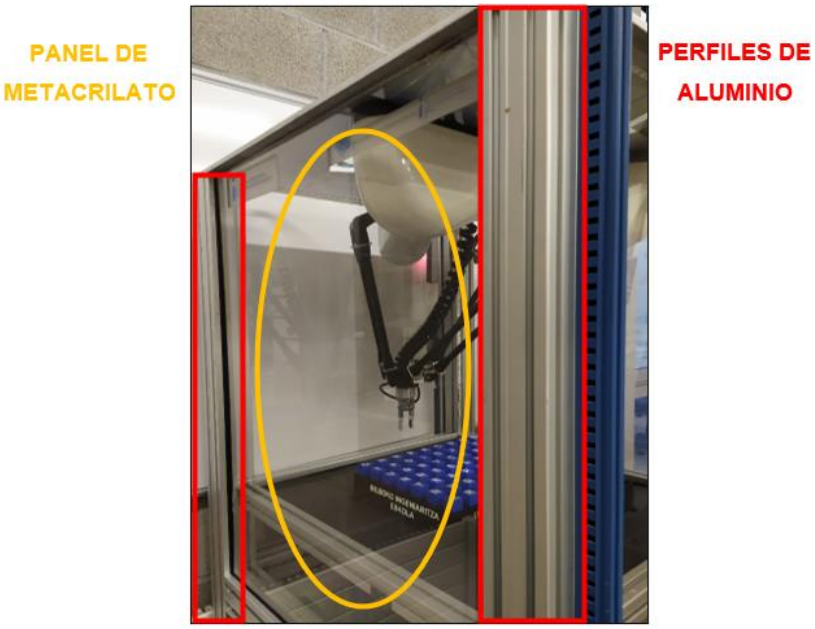


Figura 6.71. Chasis de aluminio de la célula robotizada y paneles de metacrilato.

A continuación, hay que definir el acceso al espacio de seguridad, que se ha realizado mediante dos puertas; una para acceder a la zona superior y otra para acceder a la zona inferior. Ambas zonas de acceso disponen de los elementos de seguridad en caso de que se abran dichas puertas y así asegurar la integridad de la persona en cuestión.

Las puertas que permiten acceder al habitáculo superior e inferior del robot cuentan con unos sensores (ver Figura 6.72) para detectar cuando se encuentran abiertas o cerradas de manera que el robot se pare inmediatamente en el momento en que una puerta se abra para garantizar la seguridad del usuario y evitar posibles accidentes.

Figura 6.72. Sensores de las puertas del circuito de seguridad.

El sistema de seguridad dispone de unos indicadores lumínicos que indican el estado de la celda y si está en funcionamiento o no. La aplicación ha sido programada con gran seguridad por lo que, si durante la marcha del robot se detecta desde el PLC algún valor anómalo, este parará el robot y encenderá el piloto rojo de la botonera auxiliar. Y para poder volver a arrancar el robot es necesario realizar el proceso de arranque completo.

En la figura 6.73, se puede observar la botonera auxiliar programable, que cuenta con un panel de tres luces led junto a un selector de tres posiciones con enclavamiento.

En caso de que el robot esté en funcionamiento o preparado para funcionar se encenderá el piloto verde y si el robot se encuentra parado, sin alimentación o con un algún error se encenderá el piloto rojo.

Figura 6.73. Panel de la botonera auxiliar programable junto con sus pilotos.

Para definir las limitaciones del movimiento del robot y el vallado perimetral mediante los paneles de metacrilato es necesario analizar la norma UNE-ISO 10218-2:2011 que se ha comentado anteriormente.

En primer lugar, se identifican los espacios de la celda a diseñar. El espacio de seguridad debe ser al menos, el espacio de trabajo del robot, sumando su espacio máximo de trabajo más la longitud del elemento terminal y una pieza de trabajo.

Este espacio se considera el espacio restringido, y en este caso, se define que el espacio de seguridad deberá ser al menos 10 cm mayor que el restringido, como muestra la figura 6.74.

Donde en la parte izquierda de la figura se muestra la pinza que tiene una anchura de 50mm, pero solo puede sobre salir sobre el área de trabajo la mitad es decir 25mm por cada lado más el espacio restringido de 100mm.

Por debajo, la longitud de la pinza desde el TCP del robot son 45.1mm y el cubo que es de 25mm, como máximo puede sobresalir por debajo de la pinza 12.5mm, más el área de seguridad inferior de 100mm.

En la parte derecha de la figura, se puede ver en color rojo el espacio de seguridad mínimo que debe tener la celda del robot y que cumple nuestra célula de seguridad.

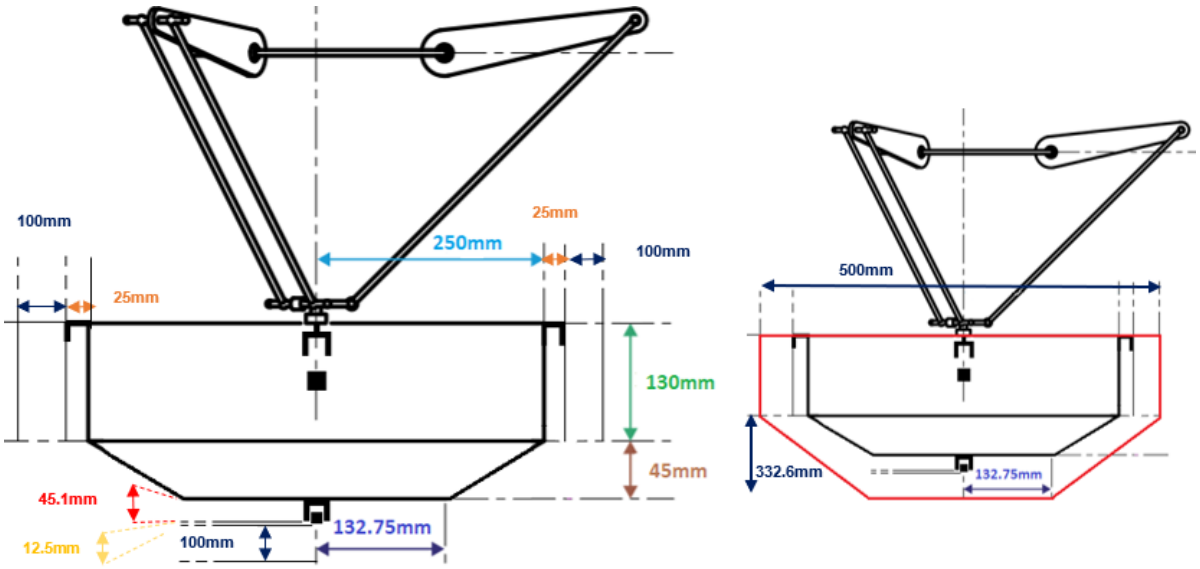


Figura 6.74. Espacio de seguridad mínimo del robot.

Si durante el funcionamiento del robot ocurre algo que puede peligrar la seguridad del operario o del propio robot, la célula robotizada cuenta con una botonera de emergencia que al apretarla quita toda la alimentación de la célula.

Figura 6.75. Botonera de emergencia.

Paralelamente, se cuenta con botonera de activación que permite al usuario mediante dos botones activar y desactivar la alimentación de la red hacia los motores del robot. La figura 6.76 muestra la botonera de activación que se encuentra instalada en el lateral de la célula robotizada.

Figura 6.76. Botonera de activación.

Apretando el pulsador verde se permitirá el paso de corriente a los servomotores y apretando el pulsador rojo se impide el paso de dicha corriente para parar el robot.

Con tal de cumplir con la normativa seleccionada, las partes de los sistemas de control encargadas de la seguridad (SRP/CS), deben estar diseñadas de manera que un único fallo en cualquiera de estas partes no conlleve una pérdida de la función de seguridad. En otras palabras, si algún elemento falla, no puede derivar en un fallo general del sistema de seguridad de la celda.

6.7. Interfaz gráfica de usuario.

Para alcanzar el objetivo principal del TFM, que es el diseño e implementación de una célula robotizada con un sistema de visión artificial para operaciones pick and place, es necesario desarrollar una interfaz gráfica que dé respuesta a las necesidades del usuario.

Dado que la aplicación consiste en formar palabras, elegidas a través de una pantalla táctil por el usuario, mediante cubos con letras serigrafiadas en la cara superior, las acciones a programar para la nueva interfaz son las siguientes:

- Comunicación de la interfaz con el robot.
- Selección por parte del usuario de la palabra que se desea formar, bien escogiendo entre las propuestas o bien escribiendo en pantalla la que desee.
- Comunicación de la interfaz con el sistema de visión artificial para que detecte donde se encuentra cada una de las letras en cada momento, visualizando en pantalla las letras que serán manipuladas por el robot y el movimiento del mismo, permitiendo controlar su velocidad desde la interfaz.
- Información del estado de la conexión TCP, de las posiciones de los cubos que va a recoger y depositar en cada plataforma y permitiendo diferentes idiomas (castellano y euskera)

Todo ello será implementado en Matlab, utilizando la herramienta Matlab Guide y para su control se utiliza el monitor táctil de la célula robotizada.

6.7.1. Inicio de la interfaz.

En este apartado se explica y se muestra el diseño y las funciones que se pueden realizar al iniciar la interfaz (ver Figura 6.77), donde aparece una imagen con el fondo de la escuela y dos botones para arrancar la interfaz (en color verde) o para cerrarla (en color negro).

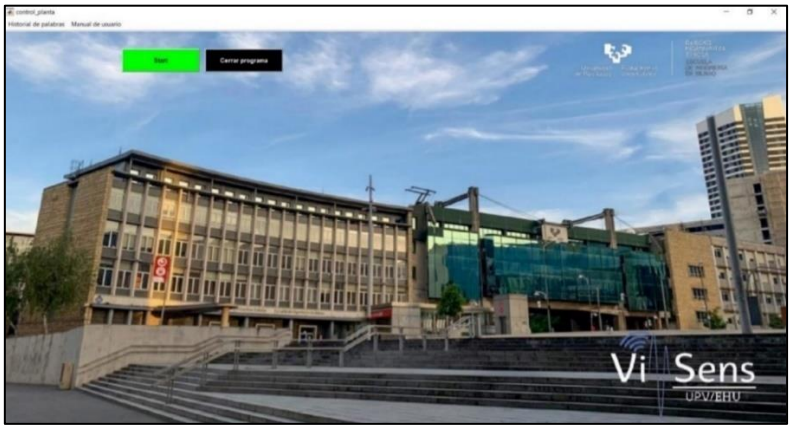


Figura 6.77. Inicio de la interfaz.

La interfaz desarrollada cuenta con las siguientes partes que se pueden ver en la figura 6.78:

- Panel de selección de idioma.
- Panel de palabras predefinidas.
- Panel de comprobar palabra.
- Panel de la plataforma grande.
- Panel de la plataforma pequeña.
- Panel del robot.
- Pulsador de arranque y paro de la interfaz.
- Pulsador de cierre de la interfaz.
- Pulsador del panel de comprobar palabra.
- Pulsador del panel de la plataforma grande.
- Pulsador del panel de la plataforma pequeña.
- Pestaña de historial de palabras.
- Pestaña del manual de usuario.

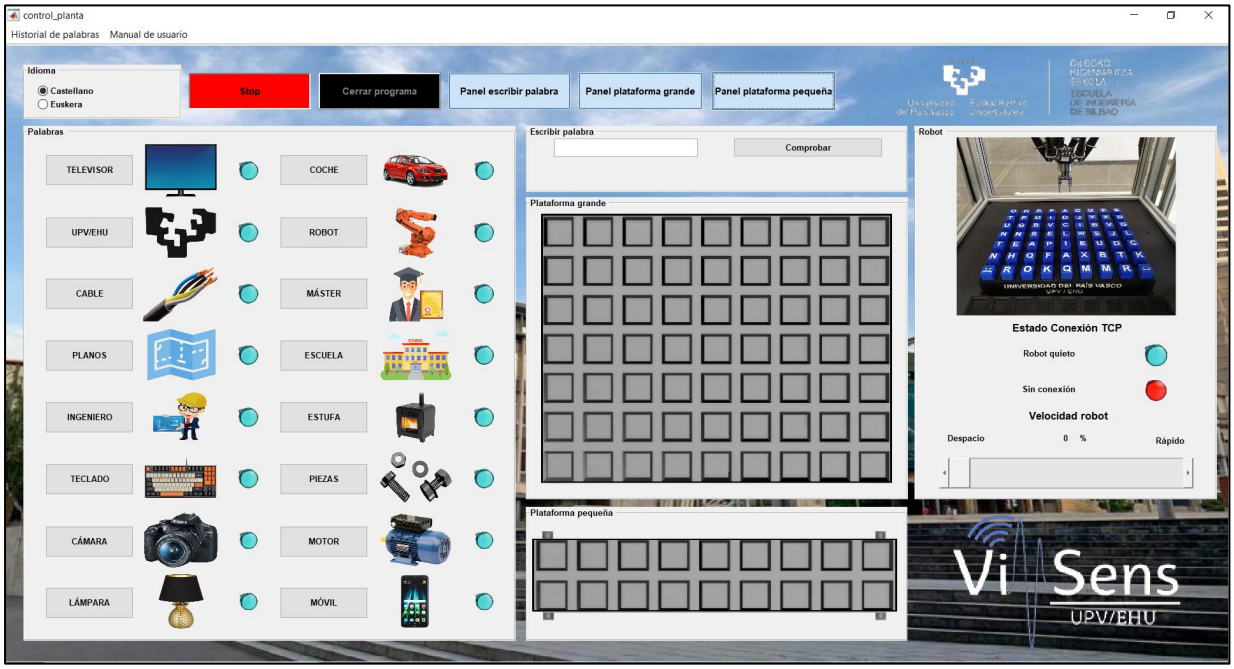


Figura 6.78. Distribución de los paneles de la interfaz.

6.7.2. Panel de selección de idioma.

En esta sección se explica la función encargada de cambiar el idioma de las palabras de la interfaz, ya que se ha implementado en la parte superior izquierda (recuadrado en color azul en la figura 6.96) un panel de idioma, que permite cambiar el idioma de la interfaz entre castellano y euskera.

Figura 6.79. Panel de selección de idioma.

Para poder implementar el cambio de idioma, se ha desarrollado una función que, según el idioma seleccionado, permite cambiar todos los textos de la aplicación de un idioma a otro.

6.7.3. Panel de palabras.

En este apartado se trata el panel de 'Palabras' (recuadrado en color verde en la figura 6.96) que se encuentra justo debajo del panel anterior que contiene 16 pulsadores con 16 palabras que el robot puede escribir. Cada una de estas palabras tiene a su lado derecho una imagen de la palabra y unos pilotos verdes que indican que la palabra está disponible para escribirla.

Estos pilotos pueden encontrarse de tres colores como muestra la figura 6.80:

- Piloto verde. Indica que la palabra asociada a ese piloto está disponible para escribirse.
- Piloto amarillo. Indica que la palabra se encuentra en proceso de formación por el robot. Este piloto aparece al pulsarse una palabra para escribir.
- Piloto rojo. Indica que la palabra que no está disponible para escribir. Cuando se pulsa una palabra para escribir, todas las demás se bloquean hasta que se termine de formar y recoger la seleccionada.

Figura 6.80. Pilotos de la interfaz.

En este panel se encuentran las palabras predefinidas que el robot siempre podrá escribir con la distribución de letras de la plataforma. En la figura 6.81, se muestra el panel de palabras, donde en la parte izquierda se muestra el panel por defecto y en la parte izquierda se muestra el panel cuando se pulsa una palabra para escribir, en este caso 'Televisor'.

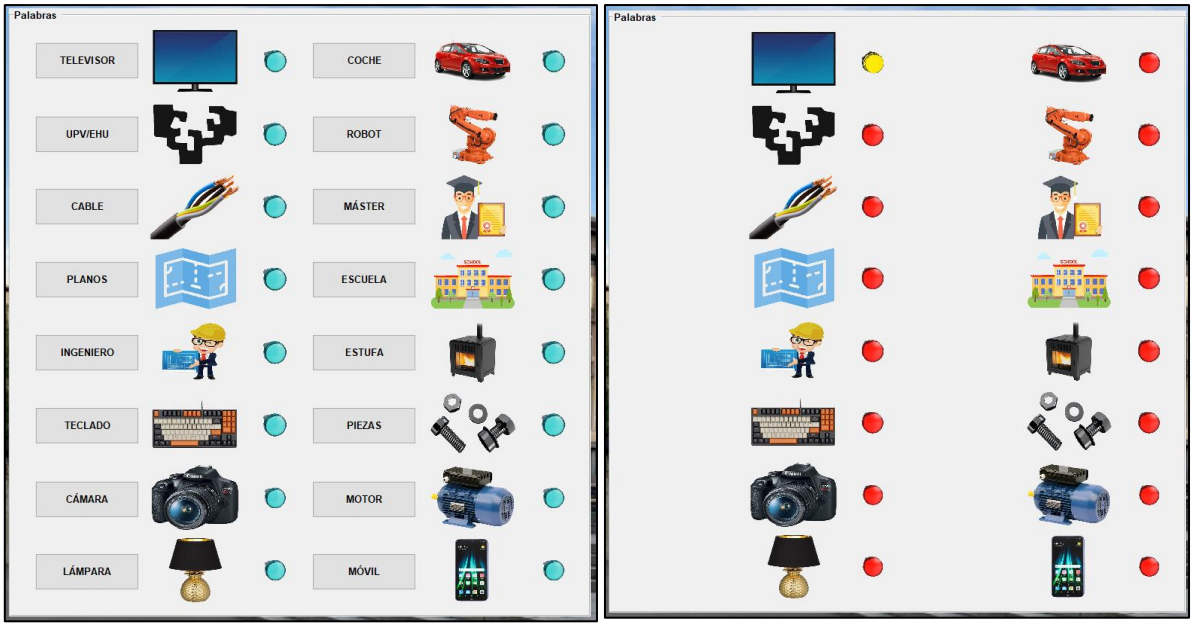


Figura 6.81. Panel de palabras.

Para que este panel funcione correctamente existen varias (ver anexo *III.IX. Secuencia panel palabras.*) funciones que trabajan de forma coordinada para que, al apretar un pulsador, se pase la información de la palabra que se quiere formar a las funciones que se encargan del cálculo y la obtención de coordenadas.

Posteriormente, otra función se encarga de ocultar todos los pulsadores (ver anexo *III.XIV. Ocultar pulsadores., III.XIII. Mostrar pulsadores.*), definir cada piloto del color adecuado (ver anexo*, III.IV. Control color pilotos., III.V. Actualizar color pilotos.*) e inhabilitar el panel de escritura de otras palabras para no provocar conflictos de escritura de palabras.

6.7.4. Pulsadores de arranque y paro de la interfaz.

En esta sección se tratan los pulsadores de arranque y paro de la interfaz (ver Figura 6.82) que se encuentran en la parte superior central (recuadrado en color naranja en la figura 6.96).

El primero permite parar la interfaz, cuando la interfaz esté arrancada, aparecerá en color rojo con la palabra ‘Stop’ para parar la interfaz y se pondrá en color verde cuando la interfaz está parada y se quiera arrancar. El segundo pulsador, que aparece en color negro permite cerrar la interfaz, pero este botón se encuentra deshabilitado si la interfaz está en marcha. Si se quiere cerrar la interfaz, primero hay que pararla, pulsando en ‘stop’ y posteriormente pulsa en ‘cerrar programa’.

Figura 6.82. Pulsadores de arranque y paro de la interfaz.

El estado de marcha, paro y cierre de la aplicación se controla desde la función principal de la interfaz (ver anexo *III.I. Función principal.*) que al mismo tiempo funciona como director de orquesta de todas las demás funciones, ejecutando cada una según la acción demandada del usuario.

6.7.5. Panel de comprobar palabra.

En esta división se explica el panel de comprobar palabra (recuadrado en color amarillo en la figura 6.96) que se encuentra en la parte central de la interfaz y permite comprobar si es posible escribir una palabra fuera de las que están predefinidas.

Dentro de este panel (ver Figura 6.83) se habilita un recuadro de texto, donde el usuario puede escribir cualquier palabra que quiera que el robot escriba. Posteriormente, es necesario pulsar el botón de comprobar para verificar si esa palabra se puede escribir y si es posible, se comenzará a ello.

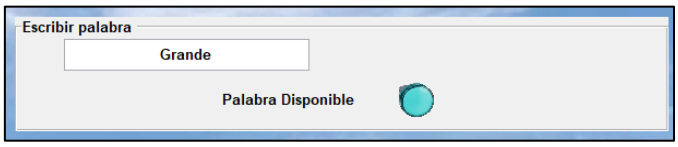


Figura 6.83. Palabra disponible para escribir.

Para esto, se ha desarrollado una función (ver anexo *III.X. Secuencia panel comprobar palabras.*) capaz de conocer la cantidad de letras de cada tipo que contiene la plataforma grande a través del programa de visión artificial y comprobar letra por letra si se puede escribir dicha palabra.

En ocasiones no se pueden escribir todas las palabras, porque depende de la cantidad de letras de cada tipo que se hayan colocado, si la palabra tiene más de 9 letras o si contiene la letra ‘ñ’.

En estos casos, al apretar el botón de comprobar, aparecerá un piloto rojo y un mensaje de que la palabra no se puede escribir y pasados 3 segundos, permitirá escribir otra, como se puede ver en la figura 6.84.

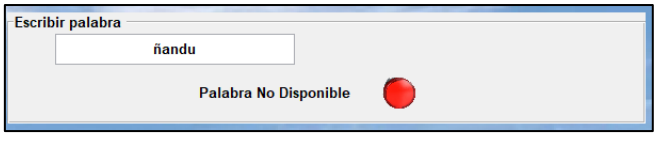


Figura 6.84. Palabra no disponible para escribir.

Este panel cuenta con un pulsador (recuadrado en color amarillo en la figura 6.96) que permite desplegar u ocultar el panel. En la figura 6.85, se muestra el panel en la parte izquierda y en la parte derecha el pulsador asociado.

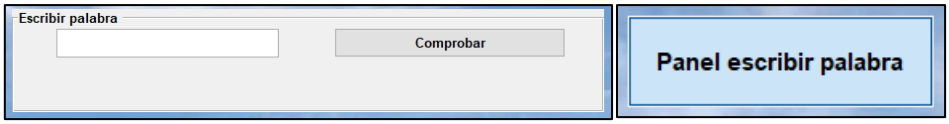


Figura 6.85. Panel de comprobar palabra.

6.7.6. Panel de la plataforma grande.

En este apartado se expone el panel de la plataforma grande (recuadrado en color morado en la figura 6.96) que se encuentra en la parte central de la interfaz y muestra la plataforma grande y el cuadrante de donde el robot va a recoger cada letra para formar la palabra. Este panel cuenta con un pulsador que permite desplegar u ocultar el panel.

En la figura 6.86, se muestra el panel vacío en la parte izquierda, el panel con los cuadrantes donde se va a recoger cada cubo para la palabra 'televisor' en la parte derecha y en la parte inferior el pulsador asociado.

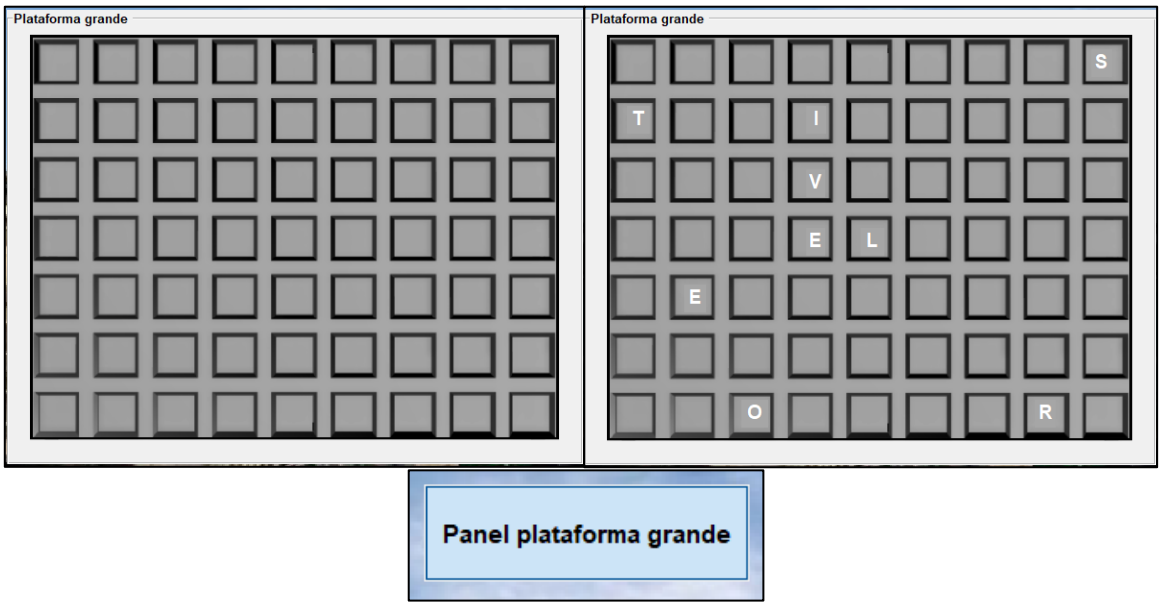


Figura 6.86. Panel de la plataforma grande.

Para este panel, se ha implementado una función (ver anexo *III.XVII. Panel plataforma grande.*) que a partir de la matriz de letras que se obtiene del programa de visión artificial, permite obtener las posiciones de cada letra que el robot va a extraer de dicha matriz para mostrarlo gráficamente.

6.7.7. Panel de la plataforma pequeña.

En esta sección se trata el panel de la plataforma pequeña que se encuentra en la parte inferior central de la interfaz y permite

ver la plataforma pequeña y el lugar donde el robot va a depositar cada cubo para formar la palabra. Este panel cuenta con un pulsador que permite desplegar u ocultar este panel.

En la figura 6.87, se muestra el panel vacío en la parte izquierda, el panel con el lugar donde se va a depositar cada cubo para la palabra 'televisor' en la parte derecha y el pulsador asociado en la parte inferior.

Figura 6.87. Panel de la plataforma pequeña.

Para este panel se ha desarrollado una función (ver anexo III.XIX. *Panel plataforma pequeña.*), que conociendo la palabra que se desea escribir y su longitud, es capaz de obtener la posición de la plataforma pequeña donde va a ir cada cubo.

6.7.8. Panel del robot.

En esta sección se explica el último panel de la interfaz que es el panel del robot (recuadrado en color rosa en la figura 6.96) que se encuentra en la parte central derecha de la interfaz y muestra lo que ve el robot a través de la webcam (ver Figura 6.88). Al mismo tiempo se muestra el estado de la conexión TCP.

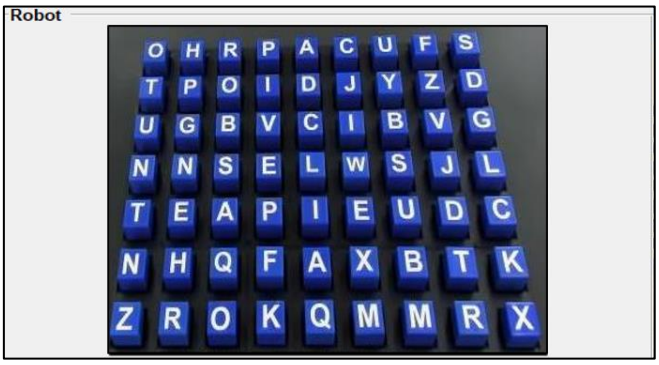


Figura 6.88. Visión de la webcam.

Donde la interfaz informa al usuario de si el robot se encuentra quieto o en movimiento, a través de un mensaje y un piloto verde en caso de que este parado y con una señal de advertencia en caso de que esté en movimiento. En la figura 6.89, se observa en la parte izquierda los avisos con el robot quieto y en la parte derecha con el robot en movimiento.

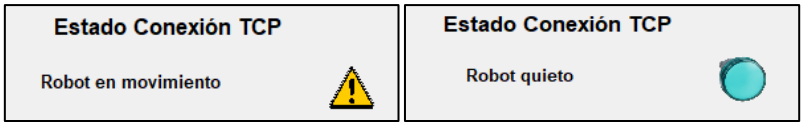


Figura 6.89. Avisos del estado del robot.

Al mismo tiempo se informa del estado de la conexión TCP mediante un mensaje y un piloto de color. En la figura 6.90, se puede ver en la parte izquierda los avisos sin establecer la conexión TCP y en la parte derecha con la conexión TCP establecida.

Figura 6.90. Estado de la conexión del robot.

Además, en este panel se permite regular la velocidad a la que se quiere que se mueva el robot mediante una barra deslizante que se puede llevar desde el 0% al 100%. Este rango de velocidades del robot no se corresponde con la velocidad mínima y máxima del robot sino con un rango de velocidad segura, tanto para el robot como para el usuario. En la figura 6.91, se muestra diferentes velocidades del robot.

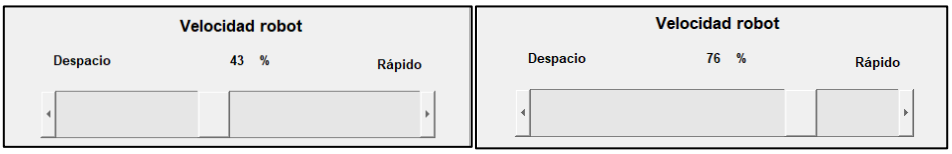


Figura 6.91. Barra deslizante para regular la velocidad del robot.

Es importante saber que antes de seleccionar la palabra que se quiere formar, se seleccione a la velocidad a la cual se quiere que se mueva el robot. Finalmente, en la figura 6.92, se puede observar el panel completo del robot.

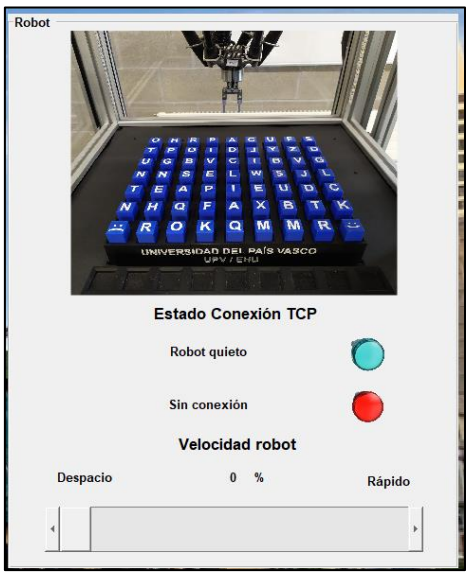


Figura 6.92. Panel completo del robot.

Para este panel, se han implementado varias funciones cuyas tareas son mostrar el panel del robot junto con su imagen (ver anexo *III.XXII. Velocidad robot.*), mostrar el estado de la conexión TCP, definir la velocidad del robot y declarar el vector de comunicación (ver anexo *II.IV. Acondicionamiento del vector TCP.*, *III.II. Comunicación TCP.*) para mandar la información al robot.

6.7.9. Pestañas habilitadas.

En este apartado se exponen las dos pestañas habilitadas que se encuentran en la parte superior izquierda de la interfaz. La primera pestaña muestra un documento con el histórico de las palabras que se han escrito con la fecha y hora de cada una de ellas (ver anexo *III.VIII. Exportar fichero palabras.*), como se puede ver en la figura 6.93.

```
Fecha: 28-Mar-2022 10:07:39  --->  Palabra escrita: camara
Fecha: 28-Mar-2022 10:09:10  --->  Palabra escrita: lampara
Fecha: 28-Mar-2022 10:10:59  --->  Palabra escrita: robot
Fecha: 28-Mar-2022 10:12:18  --->  Palabra escrita: master
Fecha: 28-Mar-2022 10:14:07  --->  Palabra escrita: escuela
Fecha: 28-Mar-2022 10:15:50  --->  Palabra escrita: estufa
```

Figura 6.93. Parte del documento de historial de palabras.

También se ha implementado una pestaña pensada para ayudar al usuario a utilizar correctamente la interfaz. Esta pestaña se llama 'Manual de usuario' y al pinchar en ella despliega por pantalla un documento con una interfaz, como se muestra en la figura 6.94.

```
---------------------------------- MANUAL DE USUARIO PARA EL ROBOT PARALELO ----------------------------

---------------------------------- ALIMENTAR ROBOT Y ENCENDER INTERFAZ ----------------------------

PASO 1. Apretar el pulsador verde para alimentar el robot.

PASO 2. Esperar 5 segundos y colocar el selector en la posición superior (3 INIT), cuando el piloto
ver se encienda, bajar el selector a la posición inferior (4 RUN).

PASO 3. Ejecutar la aplicación del escritorio de Windows IROBOT.

PASO 4. Una vez se cargue la aplicación, aparecerá la ventana de la interfaz donde habrá que pulsar
en 'START' para comenzar a utilizar la interfaz gráfica.

PASO 5. Después de pulsar 'START', pulsar en tres los paneles para desplegarlos ('panel escribir
palabra', 'panel plataforma grande', 'panel plataforma pequeña').

Los paneles de las plataformas muestran los huecos de donde el cubo va a recoger cada de las letras
para formar la palabra.

En la pestaña Idioma, se puede seleccionar el idioma de las palabras de la interfaz para escribir
las palabras en euskera.

------------------- REPETIR LOS PASOS 6, 7 Y 8 PARA ESCRIBIR LA SIGUIENTE PALABRA -----------------

PASO 6. Primero habrá que ajustar la velocidad del robot (0 – 100%) deseada de la barra de la
derecha del panel robot deslizándola.

PASO 7. Con la velocidad seleccionada hay que seleccionar la palabra que se desea escribir, para
ello hay dos paneles; el panel 'palabras' donde pulsando sobre los botones de cualquier palabra el
robot comenzará a escribir esa palabra.
La segunda forma es desde el panel 'escribir palabra' donde se escribirá la palabra deseada en el
cuadro en blanco y se pulsará en comprobar. Si la palabra se puede escribir con las letras de la
plataforma del robot, este comenzará a moverse, en cambio, si no hay suficientes letras para
escribir dicha palabra, mostrará un mensaje de 'palabra no disponible' y permitirá escribir otra.

PASO 8. Cuando el robot termine de recoger los cubos después de escribir una palabra, esperar 3
segundos y colocar el selector en la posición superior (3 INIT), esperar un segundo y bajar el
selector a la posición inferior (4 RUN).

Paso 9. Para cerrar la interfaz, es necesario pulsar en 'Stop', esperar 3 segundos y posteriormente
se puede pulsar el boton de 'Cerrar programa' para apagar la aplicación.
```

Figura 6.94. Manual de usuario de la interfaz.

En la figura 6.95, se puede observar las pestañas de la interfaz.

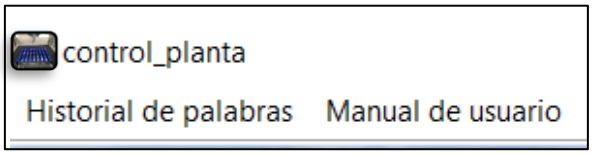


Figura 6.95. Pestañas de la interfaz gráfica.

En la figura 6.96, puede verse la interfaz completa con todos los paneles desplegados que se acaban de explicar.

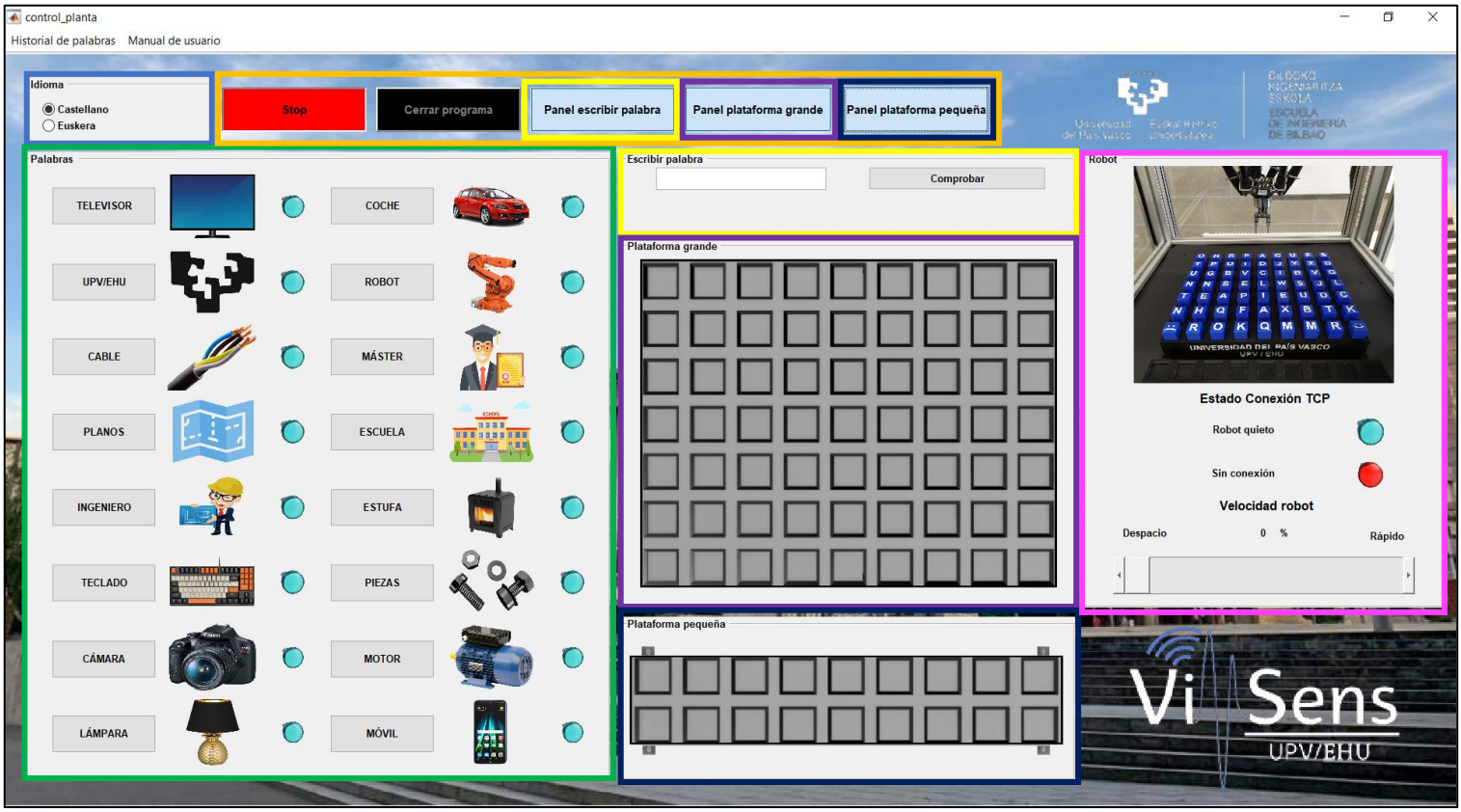


Figura 6.96. Interfaz con cada uno de los paneles recuadrados.

6.7.10. Aplicación.

Una vez se ha completado el programa y todas las funciones, en esta sección se detalla la creación de la aplicación autónoma IROBOT para instalar el programa y la interfaz en cualquier equipo sin necesidad de tener Matlab instalado.

Esta aplicación puede instalarse en equipos cuyo sistema operativo no sea Windows, ya que también soporta macOS y Linux. Para ello se ha utilizado la herramienta de Matlab Compiler que permite compartir programas de Matlab como aplicaciones independientes.

Para crear la aplicación hay que seguir cuatro pasos:

1. Se lanza el compilador de aplicaciones.

2. Se especifica el archivo principal de la aplicación que se va a implementar, en este caso, el archivo principal que guía todo el desarrollo del programa se llama 'control_planta'.

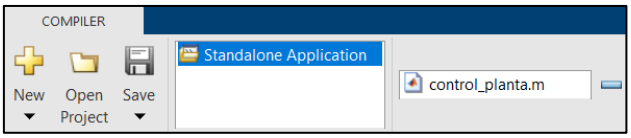


Figura 6.97. Archivo principal del programa.

3. Se define el instalador de Matlab Runtime, ya que si se quiere que la aplicación funcione sin tener Matlab instalado es necesario generar un instalador que incluye el instalador de Matlab Runtime.

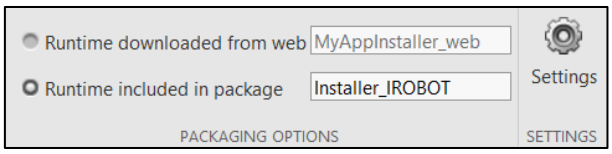


Figura 6.98. Instalador con Matlab Runtime.

4. Se personaliza la aplicación empaquetada y su apariencia con los siguientes datos, que se pueden ver en la figura 6.99:
   - Logo.
   - Nombre de la aplicación.
   - Fondo de precarga.
   - Autor.
   - Correo electrónico del desarrollador.
   - Nombre de la empresa.
   - Descripción de la aplicación.
   - Paquetes de la aplicación.
   - Funciones de las que se compone la aplicación.
   - Imágenes.

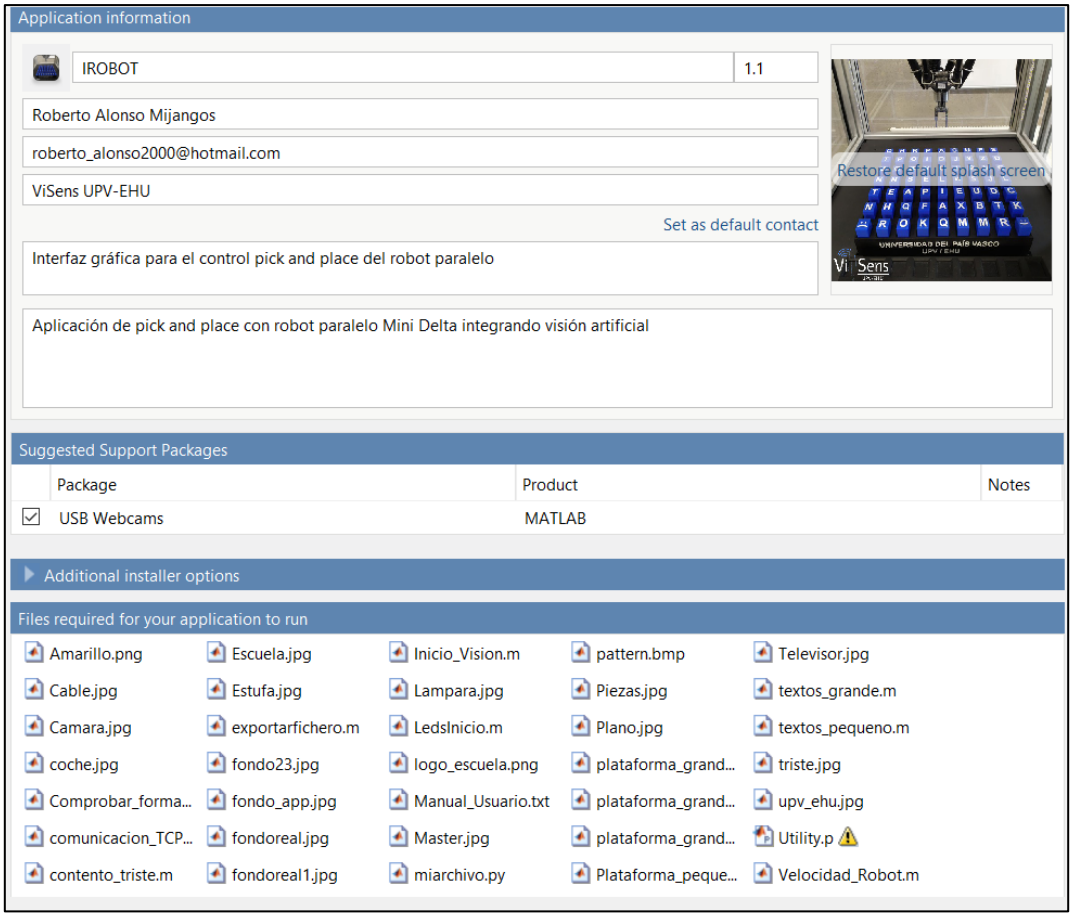


Figura 6.99. Información cargada en la aplicación.

La aplicación se ha llamado IROBOT y el logo desarrollado para la aplicación puede ver en la figura 6.100.

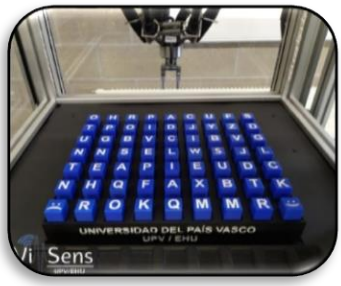


Figura 6.100. Logo de la aplicación.

6.7.10.1. Instalación de la aplicación.

Para instalar la aplicación, se proporciona un instalador dentro de la carpeta del programa como muestra la figura 6.101.

Figura 6.101. Instalador de la aplicación.

Para instalarlo aparecerá una ventana con la información y versión de la aplicación (ver Figura 6.102a). Después se elige el directorio de instalación (ver Figura 6.102b) y comenzará la instalación (ver Figura 6.102c). Al finalizar, el programa ya se encuentra instalado y se genera un acceso directo en el escritorio que permite abrir la aplicación (ver Figura 6.102d).

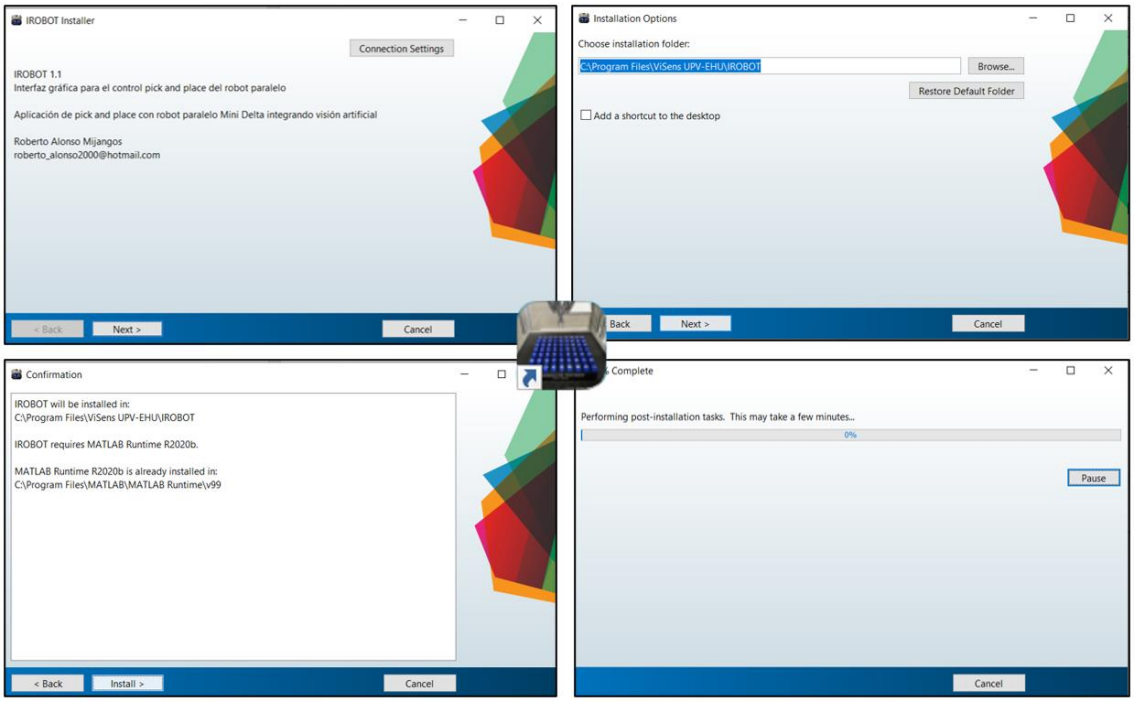


Figura 6.102. Pasos de la instalación de la aplicación.

La carpeta del programa cuenta con tres elementos:

- Un instalador. Un ejecutable que instala el programa en el ordenador del cliente.
- Un ejecutable para realizar pruebas y modificaciones en el código que no afectan a la versión final.
- La aplicación propiamente dicha.

6.8. Arquitectura de control.

En esta sección se expone la tarea desarrollada de conexiones entre los diferentes dispositivos que permiten el funcionamiento de este trabajo, así como su arquitectura de control y unos esquemas de conexiones.

Para que la aplicación funcione correctamente, además de tener la aplicación instalada es necesario que el cableado este correctamente conexionado. Es necesario conectar la webcam por USB al equipo donde está instalada la interfaz y, por otro lado, es imprescindible que se conecte un cable de red de Ethernet entre el PC y el PLC para que se pueda producir la comunicación TCP, como se muestra en la figura 6.103.

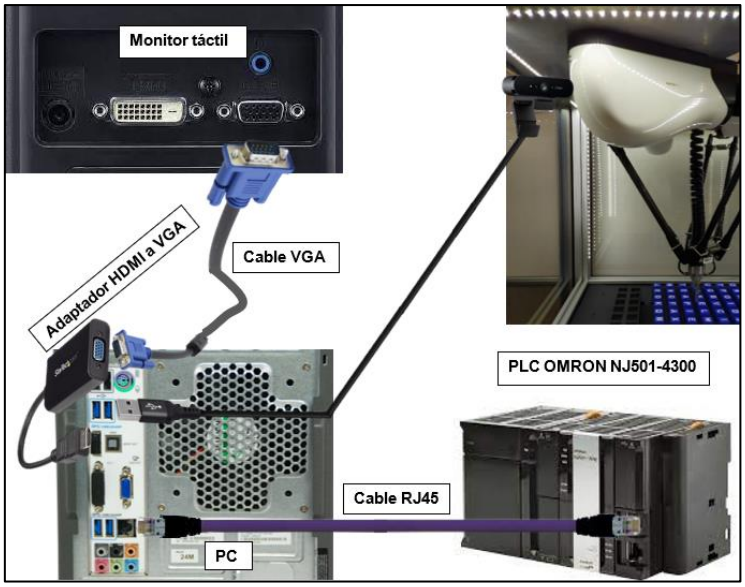


Figura 6.103. Esquema del cableado entre el ordenador y el PLC.

En esta figura se puede ver además la conexión entre el equipo y el monitor táctil, que se ha realiza mediante un cable VGA a partir de un adaptador de HDMI a VGA desde el equipo. En la figura 6.104, se muestran los cuatro elementos que tienen que ir alimentados a la red.

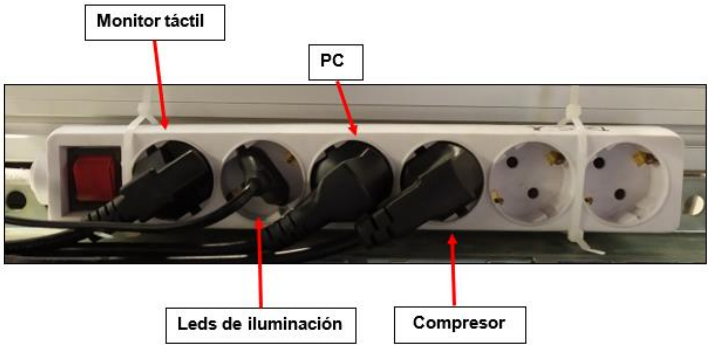


Figura 6.104. Elementos alimentados a la tensión de la red.

Por otro lado, la instalación realizada permite alimentar a la pinza neumática de aire comprimido mediante dos fuentes; bien a través de la red de aire comprimido del laboratorio, conectando la tubería de aire comprimido desde la toma del laboratorio hasta la llave de paso de la célula o bien a través del compresor como muestra el esquema de la figura 6.105.

Para cambiar la fuente de aire, se deberá soltar la tubería azul de la llave de paso y conectar la otra. La razón por la que se permiten estas dos fuentes, es para hacer que la célula sea completamente independiente y portable, y así poder transportarla a cualquier lugar donde tan solo será necesario alimentarla de la red.

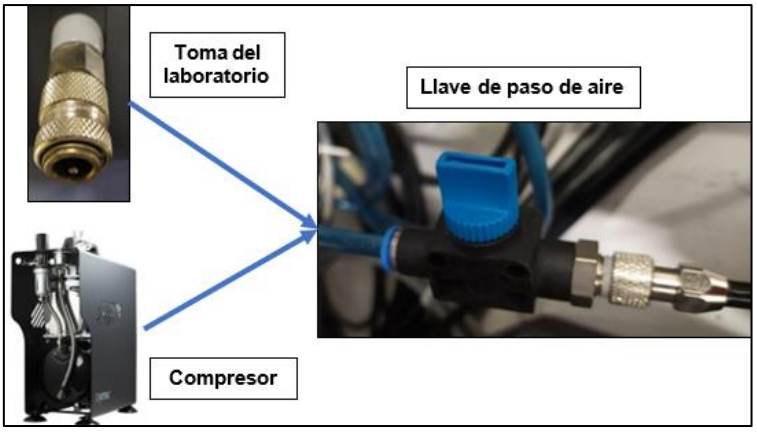


Figura 6.105. Esquema de alimentación del aire comprimido de la pinza

El esquema de control propuesto está formado por un PLC, que intercambia datos con el PC y muestra el resultado en el HMI con la interfaz gráfica desarrollada. Al mismo tiempo, el PLC se comunica con los servo-drivers para mandar al robot a la posición deseada y paralelamente obtiene información del entorno a través de los módulos de entrada y salidas digitales que se encuentran conectados a las botoneras de control del robot. La estructura de control jerarquizada se muestra en la figura 6.106.

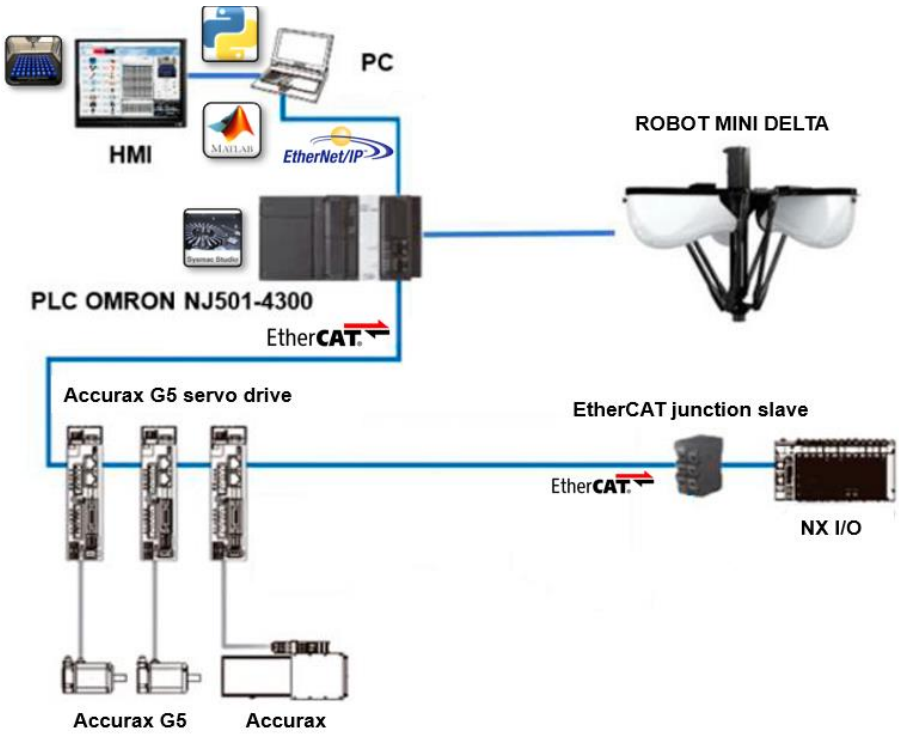


Figura 6.106. Esquema del conexionado de los componentes del robot.

6.8.1. Diagrama de flujo de la interfaz.

En esta sección se muestra el diagrama de flujo utilizado por la aplicación de la interfaz que se muestra en la figura 6.107.

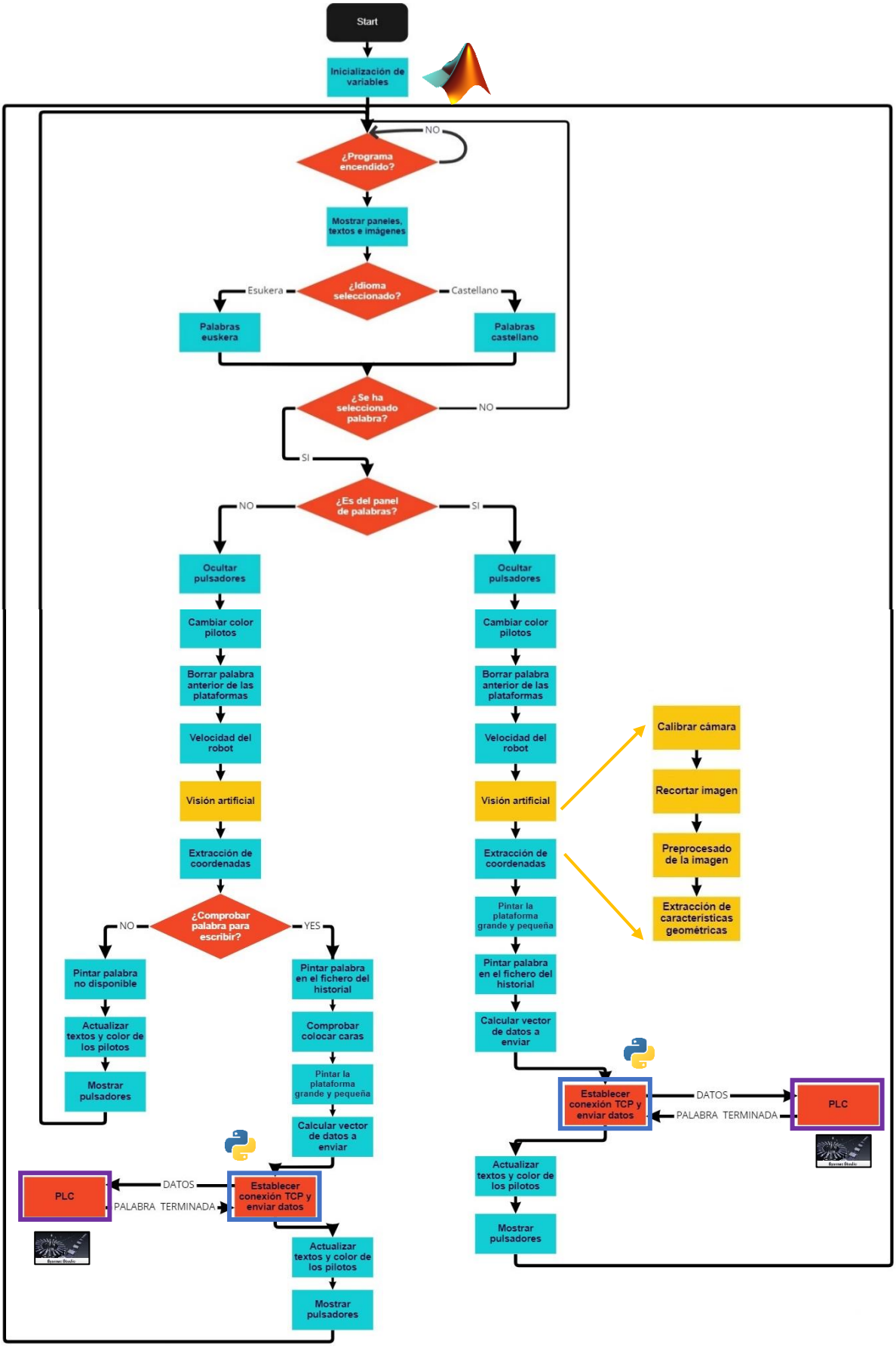


Figura 6.107. Diagrama de flujo de la interfaz

Capítulo 7:
METODOLOGÍA SEGUIDA PARA EL DESARROLLO DEL TRABAJO

7. Metodología seguida para el desarrollo del trabajo.

En este apartado se describen las tareas realizadas durante el transcurso del trabajo. Se incluyen también los medios materiales y humanos utilizados en cada una de las tareas, que servirán como base para el desarrollo del presupuesto. Las tareas realizadas son las siguientes:

1. Búsqueda de la información.
2. Diseño de la célula robotizada.
3. Selección de la instrumentación utilizada.
4. Diseño de las piezas 3D.
5. Aprendizaje de uso del software Sysmac Studio.
6. Impresión de piezas en 3D.
7. Programación, configuración y calibración del movimiento del robot.
8. Diseño de la iluminación y cableado.
9. Programación del sistema de visión artificial.
10. Diseño y programación de la interfaz gráfica de usuario.
11. Diseño y programación de la comunicación.
12. Cableado de la arquitectura de control.
13. Pruebas, solución de errores y validación.
14. Elaboración del informe.

En los siguientes apartados, se procede a describir cada una de estas tareas con mayor detalle.

T1. Búsqueda de la información.

En esta tarea se ha realizado una detallada búsqueda de información de los tipos de robots paralelos existentes en la actualidad, buscando en el mercado cuales son los comercializados por los principales fabricantes del sector.

También se incluye en esta tarea una búsqueda de información del contexto tecnológico actual para intentar adecuar la solución adoptada a las últimas tecnologías presentes en la actualidad.

- Duración: 2 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador.

T2. Diseño de la célula robotizada.

Una vez se ha obtenido toda la información de la tarea interior, se procede a la selección del robot paralelo que mejor se ajusta a los requisitos de la aplicación.

Además de la selección del robot, también se hará un diseño preliminar de la célula, detallando cuales son los componentes adicionales necesarios para el correcto funcionamiento de la misma, para en tareas posteriores hacer un diseño en detalle.

- Duración: 2 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador.

T3. Selección de la instrumentación utilizada.

Tras hacer el diseño preliminar de la célula y determinar cuál va a ser la instrumentación necesaria, se procede a buscar en el mercado la instrumentación que mejor se adapte a los requisitos de la misma.

- Duración: 1 semana.
- Medios humanos: Ingeniero Junior y Responsable de Proyecto.
- Medios materiales: Ordenador.

T4. Diseño de las piezas.

En esta tarea se ha realizado en diseño de detalle de las piezas 3D en base al espacio de trabajo del robot.

- Duración: 2 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador.

T5. Aprendizaje de uso del software Sysmac Studio.

Dado que cada fabricante de robots dispone de su propio software y lenguaje de programación, ha sido necesario realizar un proceso de aprendizaje en la herramienta de Sysmac Studio apoyándose en los recursos existentes en la red.

- Duración: 2 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador.

T6. Impresión de piezas en 3D.

Esta tarea incluye el tiempo de espera transcurrido en el proceso de impresión de las piezas mediante impresoras 3D.

- Duración: 2 semanas.
- Medios humanos: Ingeniero Junior, Técnico de impresión 3D.
- Medios materiales: Ordenador, set de herramientas.

T7. Programación, configuración y calibración del movimiento del robot.

En esta tarea se ha realizado la programación del control de movimiento del robot después de realizar la configuración hardware y la calibración.

- Duración: 4 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador, set de herramientas.

T8. Diseño de la iluminación y cableado.

En esta tarea se ha detallado el diseño e implementación del sistema de iluminación del robot y su cableado para el correcto funcionamiento del sistema de visión artificial.

- Duración: 1 semana.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador, set herramientas, soldador, cables, tiras led.

T9. Programación del sistema de visión artificial.

En esta tarea se ha realizado la programación completa del sistema de visión artificial que extrae la información de las imágenes obtenidas por la webcam.

- Duración: 4 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador, webcam.

T10. Diseño y programación de la interfaz gráfica.

En esta tarea se ha ejecutado el diseño y la programación completa de la interfaz gráfica de usuario.

- Duración: 6 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador, monitor táctil.

T11. Diseño y programación de la comunicación.

En esta tarea se ha llevado a cabo el diseño, selección y la programación del protocolo de comunicación entre el PLC y el PC.

- Duración: 3 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador.

T12. Cableado de la arquitectura de control.

Esta tarea consiste en el diseño y el montaje del cableado de todos los dispositivos añadidos para que la aplicación funcione correctamente y sea móvil.

- Duración: 1 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Taladro, set de herramientas.

T13. Pruebas, solución de errores y validación.

Una vez comprobado el funcionamiento individual de cada uno de los elementos, se han realizado las pruebas de integración, probando el funcionamiento del conjunto, solucionando los errores encontrados.

- Duración: 2 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador portátil.

T14. Elaboración del informe.

Esta tarea incluye el tiempo necesario para realizar un informe elaborado que exponga cada una de las tareas realizadas.

- Duración: 4 semanas.
- Medios humanos: Ingeniero Junior.
- Medios materiales: Ordenador.

T15. Diagrama de Gantt.

En este apartado, se muestra el diagrama de Gantt de las tareas desarrolladas en el apartado anterior, obteniendo una duración total del proyecto de 32 semanas, comenzando el 15 de septiembre de 2021 y finalizando el 15 de mayo de 2022.

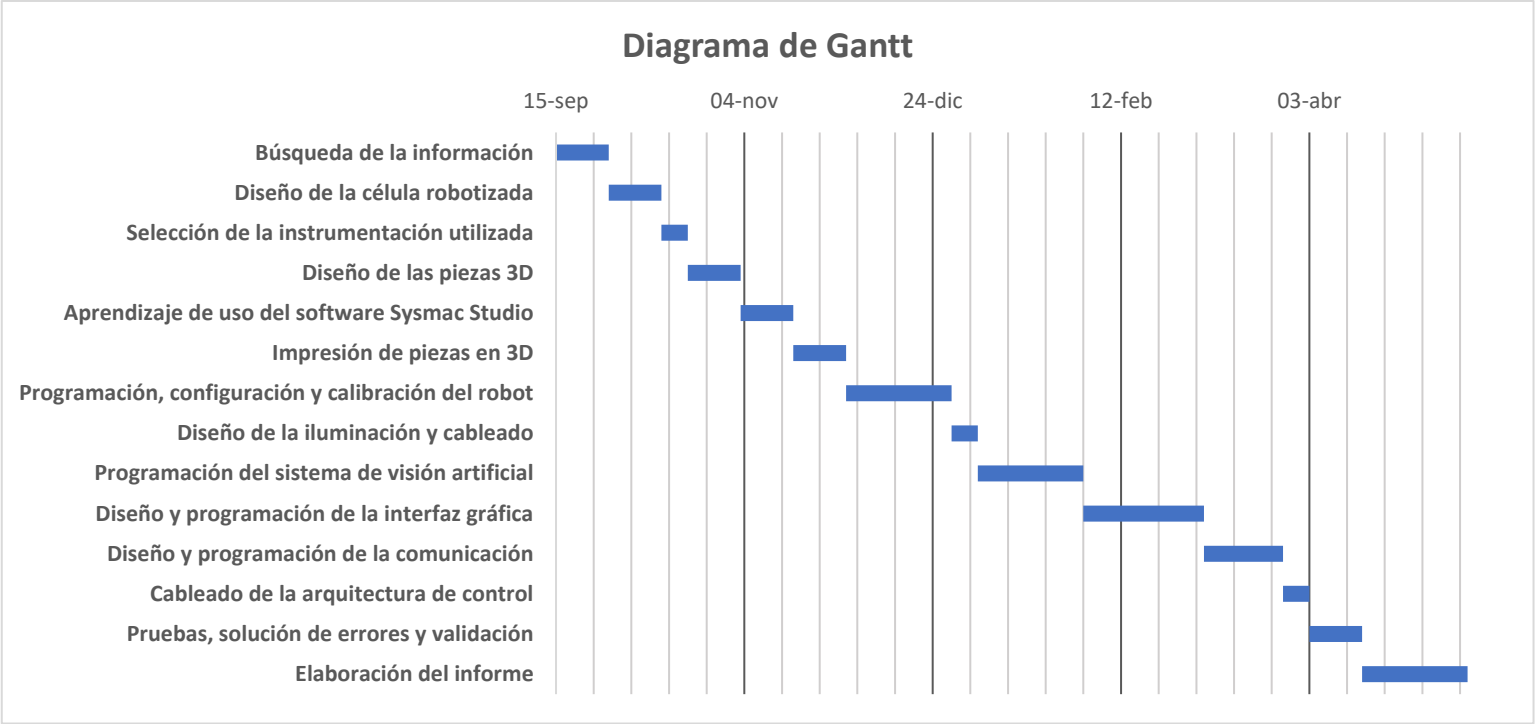


Figura 7.1. Diagrama de Gantt.

Capítulo 8:
ASPECTOS ECONÓMICOS

8. Aspectos económicos.

En este apartado se desarrolla el estudio económico necesario para llevar a cabo este trabajo. Se desglosa por actividades y se valoran cada uno de los costes.

8.1. Costes materiales totales.

En este apartado se muestra la tabla 8.1 que contiene la identificación de cada producto, la cantidad, el precio por unidad y total de los componentes utilizados para realizar el proyecto.

REFERENCIA	CANTIDAD	DESCRIPCIÓN	PVP
CR-UGD4MINI-NR	1	Robot Delta	23.890,00 €
R88D-KN04H-ECT	3	Servodrivers Accurax G5	2.880,00 €
R88A-CAKA005SR-E	3	Cables de potencia XYZ	540,00 €
R88A-CRKA005CR-E	3	Cables de encoder	360,00 €
R88A-CAKA005BR-E	3	Cables de freno	110,00 €
R88A-CRGD0R3C-BS	3	Batería y cables encoder	540,00 €
R88A-CSK003S-E	3	Cables de seguridad	119,00 €
R88A-CNW01C	3	Conectores de E/S (26 pines)	105,00 €
XS6W6LSZH8SS50CMY	3	Cables EtherCAT 0,5m	18,00 €
NJ501-4500	1	Controlador NJ	8.000,00 €
NJ-PA3001	1	Fuente controlador NJ	320,00 €
CP1W-CN221	1	Cable programación NJ	12,00 €
NX-ECC201	1	Cabecera NX	220,00 €
XS6W6LSZH8SS30CMY	1	Cable EtherCAT a cabecera	6,00 €
NX-EC0142	2	Tarjeta encoder NX	668,00 €
E6B2-CWZ1X2000PR05MOMS	2	Encoder	478,00 €
Logitech Brio Webcam 4k	1	Webcam	163,30 €
LG 17MB15T-B-	1	Monitor táctil de 17''	252,89 €
Tiras Led	1	Tiras led de 7 metros	21,99 €
PC	1	Ordenador	700,00 €
Adaptador HDMI a VGA	1	Adaptador PC a monitor	9,34 €
Sparmax TC-610H Plus Air	1	Compresor	269,00 €
R88A-BAT01G 3.6V 2700mAh	3	Batería Litio servodrivers	47,85 €
		SUBTOTAL	**39.730,00 €**

Tabla 8.1. Costes materiales.

8.2. Coste mano de obra directa.

En la tabla 8.2 se muestran los costes de personal que se desglosan en las horas necesarias para la realización de cada parte del proyecto. Con un coste de 30 euros por hora trabajada para un ingeniero.

PUESTO	€/HORA	Nº HORAS	SUBTOTAL
Estudio del estado del arte	30	20	600,00 €
Diseño del proceso	30	20	600,00 €
Diseño de piezas	30	20	600,00 €
Desarrollo del programa	30	300	9.000,00 €
Desarrollo de la interfaz gráfica	30	100	3.000,00 €
Validación de resultados	30	40	1.200,00 €
Elaboración del informe	30	100	3.000,00 €
Horas totales		**600**	
SUBTOTAL		**18.000,00 €**	

Tabla 8.2. Costes mano de obra directa.

8.3. Costes de equipo y software.

Los principales programas utilizados para este proyecto necesitan una licencia que conlleva cierto coste que se expone en la tabla 8.3.

CONCEPTO	SUBTOTAL
Licencia Fusion 360	530,00 €
Licencia Labview	1.489,00 €
Licencia Sysmac Studio	2.200,00 €
SUBTOTAL	**4.219,00 €**

Tabla 8.3. Costes de equipo y software.

8.4. Costes totales.

El coste total es la suma de los costes materiales totales, de los costes de mano de obra directa y de los costes de equipo y software, que se puede observar en la tabla 8.4.

CONCEPTO	SUBTOTAL
Costes materiales totales	39.730,37 €
Coste mano de obra directa	18.000,00 €
Costes de equipo y software	4.219,00 €
SUBTOTAL	**61.949,37 €**

Tabla 8.4. Costes totales.

8.5. Gastos generales y beneficio industrial.

Se estima un 15% para los gastos generales, y un 6% para el beneficio industrial sobre el coste de ejecución material, como se muestra en la tabla 8.5.

CONCEPTO	SUBTOTAL
Coste de ejecución material	61.949,37 €
Gastos generales (15%)	9.292,41 €
Beneficio industrial (6%)	3.716,96 €

Tabla 8.5. Gastos generales y beneficio industrial.

8.6. Presupuesto de ejecución por contrata.

El presupuesto de ejecución por contrata es la suma de los costes de ejecución material, los gastos generales y el beneficio industrial, como se puede comprobar en la tabla 8.6.

CONCEPTO	SUBTOTAL
Coste de ejecución material	61.949,37 €
Gastos generales y beneficio industrial	13.009,37 €
SUBTOTAL FINAL	**74.958,74 €**

Tabla 8.6. Presupuesto de ejecución por contrata.

8.7. Importe total del presupuesto.

Por último, se muestra el precio final de ejecución del proyecto en la tabla 8.7. Todos los precios mostrados anteriormente, llevan incluido el 21% de IVA.

CONCEPTO	SUBTOTAL
Importe final	74.958,74 €

Tabla 8.7. Importe total del presupuesto.

Capítulo 9:
CONCLUSIONES

9. Conclusiones.

La evolución que se está produciendo en la industria durante la Cuarta Revolución Industrial está llevando a las empresas a la búsqueda de la automatización y la creación de fábricas inteligentes. En este ámbito han surgido los robots paralelos como respuesta a aquellas aplicaciones que se quieren realizar con una gran velocidad y precisión.

Paralelamente se está dotando a estos robots de cámaras que junto con programas de visión artificial hacen que estas células robotizadas sean inteligentes y tomen decisiones a partir del entorno que perciben.

En este TFM se ha llevado a cabo la implementación de una célula robotizada con un sistema de visión artificial para operaciones de pick and place utilizando visión artificial para el reconocimiento de objetos y permitiendo su control desde una interfaz gráfica. Todo ello implementado dentro de la aplicación IROBOT.

Concretamente, se ha diseñado el escenario de trabajo del robot mediante las plataformas grandes y pequeñas atendiendo al espacio de trabajo del robot, permitiendo maximizar la cantidad de cubos disponibles para poder formar la mayor cantidad de palabras. Para ello se ha empleado la herramienta software Fusion 360.

En cuanto al sistema robótico, se ha diseñado una aplicación mediante Sysmac Studio capaz de configurar, calibrar y realizar la aplicación de pick and place mediante el programa de control del robot.

Al estar integrado el robot en una célula con un sistema de visión, se ha implementado un protocolo de comunicación TCP que permite el intercambio de información entre el PC (gestionará los aspectos de visión) y el PLC (gestionará las acciones del robot) consiguiendo que la comunicación sea fiable y segura. Para ello se han empleado las herramientas software Python y Sysmac Studio.

Adicionalmente, con el objetivo de integrar en la célula un buen sistema de visión, se ha implementado un sistema de iluminación que permite obtener una iluminación constante y uniforme en el habitáculo superior de la célula robotizada. En cuanto a la aplicación de visión, se ha desarrollado mediante el software de Matlab, una solución capaz de identificar cada uno de los objetos detectados y conseguir extraer sus características geométricas.

A modo de conclusión, el sistema robotizado es capaz de interaccionar con el usuario formando las palabras de 9 letras (como máximo) en tiempos por debajo de 1 minuto.

9.1. Trabajos futuros.

A partir del trabajo realizado en este proyecto, se ha abierto la oportunidad de complementar y analizar distintas soluciones a los problemas encontrados, implementarlos y comparar los resultados obtenidos.

En el ámbito del sistema de visión artificial, sería interesante comparar los resultados obtenidos con los otros dos métodos de extracción de características de la imagen mencionado en el TFM, es decir, mediante las redes neuronales convolucionales y el reconocimiento de patrones. De esta manera, se podrá confirmar cual es más robusto ante entornos de iluminación cambiantes.

En el campo de los protocolos de comunicación sería atractivo implementar esta aplicación con otros protocolos y analizar con cuales se obtienen unos mejores tiempos de respuesta y fiabilidad en el envío y recepción de los datos.

Respecto al programa con el que se ha implementado la interfaz gráfica, sería interesante replicar dicha aplicación en Python o LabVIEW y comparar el rendimiento obtenido en dichas plataformas.

Capítulo 10:
BIBLIOGRAFÍA

[1] M. A. K. Bahrin, M. F. Othman, N. H. N. Azli, and M. F. Talib, “Industry 4.0: A review on industrial automation and robotic,” *Jurnal Teknologi*, vol. 78, no. 6–13, pp. 137–143, 2016, doi: 10.11113/JT.V78.9285.

[2] A. Gasparetto, L. Scalera, A. Gasparetto, and L. Scalera, “A Brief History of Industrial Robotics in the 20th Century,” *Advances in Historical Studies*, vol. 8, no. 1, pp. 24–35, Jan. 2019, doi: 10.4236/AHS.2019.81002.

[3] J. Ros, “APLICACIONES ACTUALES DE LOS ROBOTS PARALELOS,” 2007. [Online]. Available: https://www.researchgate.net/publication/240618328

[4] “Desarrollo tecnológico en la Primera Revolución Industrial - Dialnet.” https://dialnet.unirioja.es/servlet/articulo?codigo=1158936 (accessed May 24, 2022).

[5] M. Huberman, C. M. Meissner, and K. Oosterlinck, “Technology and Geography in the Second Industrial Revolution: New Evidence from the Margins of Trade,” *The Journal of Economic History*, vol. 77, no. 1, pp. 39–89, Mar. 2017, doi: 10.1017/S0022050717000018.

[6] “La tercera revolución industrial | Universitas Humanística.” https://revistas.javeriana.edu.co/index.php/univhumanistica/article/view/9908 (accessed May 24, 2022).

[7] G. Beier, A. Ullrich, S. Niehoff, M. Reißig, and M. Habich, “Industry 4.0: How it is defined from a sociotechnical perspective and how much sustainability it includes – A literature review,” *Journal of Cleaner Production*, vol. 259, p. 120856, Jun. 2020, doi: 10.1016/J.JCLEPRO.2020.120856.

[8] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, “Intelligent Manufacturing in the Context of Industry 4.0: A Review,” *Engineering*, vol. 3, no. 5, pp. 616–630, Oct. 2017, doi: 10.1016/J.ENG.2017.05.015.

[9] K. D. Thoben, S. A. Wiesner, and T. Wuest, “‘Industrie 4.0’ and smart manufacturing-a review of research issues and application examples,” *International Journal of Automation Technology*, vol. 11, no. 1, pp. 4–16, 2017, doi: 10.20965/IJAT.2017.P0004.

[10] F. Xia, L. T. Yang, L. Wang, and A. Vinel, “Internet of Things,” *International Journal of Communication Systems*, vol. 25, no. 9, pp. 1101–1102, Sep. 2012, doi: 10.1002/DAC.2417.

[11] “(85) Diseño Industrial DISEÑO Y AUTOMATIZACIÓN INDUSTRIAL | emerson juniors campos arce - Academia.edu.”

https://www.academia.edu/9144786/Dise%C3%B1o_Industrial_DISE%C3%91O_Y_AUTOMATIZACI%C3%93N_INDUSTRIAL (accessed May 24, 2022).

[12] T. Su, L. Cheng, Y. Wang, X. Liang, J. Zheng, and H. Zhang, “Time-Optimal Trajectory Planning for Delta Robot Based on Quintic Pythagorean-Hodograph Curves,” *IEEE Access*, vol. 6, pp. 28530–28539, Apr. 2018, doi: 10.1109/ACCESS.2018.2831663.

[13] G. Wu, “Kinematic Analysis and Optimal Design of a Wall-mounted Four-limb Parallel Schönflies-motion Robot for Pick-and-place Operations,” *Journal of Intelligent & Robotic Systems*, vol. 85, no. 3–4, pp. 663–677, Mar. 2016, doi: 10.1007/S10846-016-0377-5.

[14] V. Noppeney, T. Boaventura, and A. Siqueira, “Task-space impedance control of a parallel Delta robot using dual quaternions and a neural network,” *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 43, no. 9, pp. 1–11, Sep. 2021, doi: 10.1007/S40430-021-03157-4/FIGURES/7.

[15] L. R. Douat, I. Queinnec, G. Garcia, M. Michelin, F. Pierrot, and S. Tarbouriech, “Identification and vibration attenuation for the parallel robot par2,” *IEEE Transactions on Control Systems Technology*, vol. 22, no. 1, pp. 190–200, Jan. 2014, doi: 10.1109/TCST.2013.2249515.

[16] Q. Meng, F. Xie, and X. J. Liu, “Conceptual design and kinematic analysis of a novel parallel robot for high-speed pick-and-place operations,” *Frontiers of Mechanical Engineering 2017 13:2*, vol. 13, no. 2, pp. 211–224, Nov. 2017, doi: 10.1007/S11465-018-0471-4.

[17] B. Belzile, P. K. Eskandary, and J. Angeles, “Workspace Determination and Feedback Control of a Pick-And-Place Parallel Robot: Analysis and Experiments,” *IEEE Robotics and Automation Letters*, vol. 5, no. 1, pp. 40–47, Jan. 2020, doi: 10.1109/LRA.2019.2945468.

[18] K. Harada, T. Tsuji, K. Nagata, N. Yamanobe, and H. Onda, “Validating an object placement planner for robotic pick-and-place tasks,” *Robotics and Autonomous Systems*, vol. 62, no. 10, pp. 1463–1477, Oct. 2014, doi: 10.1016/J.ROBOT.2014.05.014.

[19] W. Ding, J. Gu, S. Tang, Z. Shang, E. A. Duodu, and C. Zheng, “Development of a calibrating algorithm for Delta Robot’s visual positioning based on artificial neural network,” *Optik (Stuttg)*, vol. 127, no. 20, pp. 9095–9104, Oct. 2016, doi: 10.1016/J.IJLEO.2016.06.126.

[20] X. Zhang, “Control Technology of Terminal Attitude Adjustment of Traction Parallel Robot Based on Visual Perception,” *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering, ICBAIE 2021*, pp. 1045–1049, Mar. 2021, doi: 10.1109/ICBAIE52039.2021.9390031.

[21] “41 AVALOS CUB conduccion”.

[22] T. Su, L. Cheng, Y. Wang, X. Liang, J. Zheng, and H. Zhang, “Time-Optimal Trajectory Planning for Delta Robot Based on Quintic Pythagorean-Hodograph Curves,” *IEEE Access*, vol. 6, pp. 28530–28539, Apr. 2018, doi: 10.1109/ACCESS.2018.2831663.

[23] Y. Li, T. Huang, and D. G. Chetwynd, “An approach for smooth trajectory planning of high-speed pick-and-place parallel robots using quintic B-splines,” *Mechanism and Machine Theory*, vol. 126, pp. 479–490, Aug. 2018, doi: 10.1016/J.MECHMACHTHEORY.2018.04.026.

[24] G. Wu, “Kinematic Analysis and Optimal Design of a Wall-mounted Four-limb Parallel Schönflies-motion Robot for Pick-and-place Operations,” *Journal of Intelligent & Robotic Systems 2016 85:3*, vol. 85, no. 3, pp. 663–677, May 2016, doi: 10.1007/S10846-016-0377-5.

[25] M. Özdemir, “Dynamic analysis of planar parallel robots considering singularities and different payloads,” *Robotics and Computer-Integrated Manufacturing*, vol. 46, pp. 114–121, Aug. 2017, doi: 10.1016/J.RCIM.2017.01.005.

[26] “[PDF] The Ping Pong Robot to Return a Ball Precisely | Semantic Scholar.” https://www.semanticscholar.org/paper/The-Ping-Pong-Robot-to-Return-a-Ball-Precisely-Kyohei-Masamune/2fbcc5e2a2a3da8451116c5c1f7003a13a8d36ce (accessed May 24, 2022).

[27] “Datos del IRB 360 - IRB 360 (Robots industriales) | ABB.” https://new.abb.com/products/robotics/es/robots-industriales/irb-360/datos (accessed May 24, 2022).

[28] “MPP3S.” https://www.yaskawa.es/productos/robots/packaging/productdetail/product/mpp3s_746 (accessed May 24, 2022).

[29] “Robot delta FANUC M-3 iA 6A.” https://www.fanuc.eu/es/es/robots/p%C3%A1gina-filtro-robots/delta-robots/serie-m3/m-3ia-6a (accessed May 24, 2022).

[30] “Item # CR-UGD4MINI-NR, 3 Max Axes Delta Robot On Omron Automation Americas.” http://products.omron.us/item/sysmac-robots/mini-delta-robot/cr-ugd4mini-nr (accessed May 24, 2022).

[31] “ZX-T Series CR_UGD4MINI Series USER´S MANUAL Delta 3+1 Robot IP65 protection class CONTENTS CR_UGD4MINI User’s Manual Chapter 1 Introduction.”

[32] “CR_UGD4MINI Manual 2”.

[33] “TC-610H User’s Manual,” 2018. [Online]. Available: www.SPARMAXair.com

[34] Omron, “NJ-Series.” [Online]. Available: www.ia.omron.com

[35] “NJ-PA/PD 2 General Specification.”

[36] “57 Accurax G5 servo drive Accurax G5 servo drive.”

[37] “R88M-K40030T-BS2”.

[38] “NX-ECC 2 System Configuration System Configuration of Slave Terminals.” [Online]. Available: www.ia.omron.com

[39] “NX-ID/IA/OD/OC/MD 2 System Configurations Connected to a CPU Unit.”

[40] “S8VK-G 2 Estructura de la referencia.”

[41] M. de Uso, “Nivel Básico Diseño 3D”.

[42] “Manual Fusion 360 Español | PDF | Point and Click | Diseño.” https://es.scribd.com/document/396877544/Manual-Fusion-360-Espanol (accessed May 25, 2022).

[43] “El CD-ROM de CX-One / CX-Programmer contiene el manual de usuario del archivo PDF”.

[44] Omron, “Sysmac Studio Software Version 1.18 Operation Manual”.

[45] “Protocolos TCP y UDP: características, uso y diferencias.” https://www.redeszone.net/tutoriales/internet/tcp-udp-caracteristicas-uso-diferencias/ (accessed May 25, 2022).

[46] “Modbus: Qué es y cómo funciona | Comunicaciones Industriales.” https://www.cursosaula21.com/modbus-que-es-y-como-funciona/ (accessed May 25, 2022).

[47] “C270 HD WEBCAM Complete Setup Guide Guide d’installation complet CONTENTS”.

[48] “Complete Setup Guide Guide d’installation complet BRIO ULTRA HD BUSINESS WEBCAM CONTENTS”.

[49] J. Durán Suárez, A. Del, R. Torres, and D. Suárez, “Redes Neuronales Convolucionales en R Reconocimiento de caracteres escritos a mano Redes Neuronales Convolucionales en R Reconocimiento de caracteres escritos a mano Redes Neuronales Convolucionales en R”.

[50] S. Enrique Ceballos Jiménez, “IDENTIFICACIÓN Y RECONOCIMIENTO DE MATRÍCULAS DE AUTOMÓVILES CON MATLAB”.

[51] E. Julian and A. Almidon, “Manual de progradation LabVIEW 9.0,” *Labview*, no. October, p. 79, 2018, Accessed: May 25, 2022. [Online]. Available: https://zenodo.org/record/2557815

[52] G. C. Espinosa, “Dionisio Ramírez Prieto”.

[53] D. Orlando and B. Guerrero, “Manual de Interfaz Gráfica de Usuario en Matlab”, Accessed: May 25, 2022. [Online]. Available: www.matpic.com

[54] “20131011_GR_Robot_DELTA”.

[55] “Automation Software Sysmac Studio Ver.1.@@ Sysmac Studio for machine creators.”

[56] R. Sharpe, E. Warnicke, and U. Lamping, “Wireshark User’s Guide Preface Foreword”, Accessed: May 25, 2022. [Online]. Available: https://gitlab.com/wireshark/wireshark/wikis/.

[57] “El tutorial de Python — documentación de Python - 3.10.4.” https://docs.python.org/es/3/tutorial/ (accessed May 25, 2022).

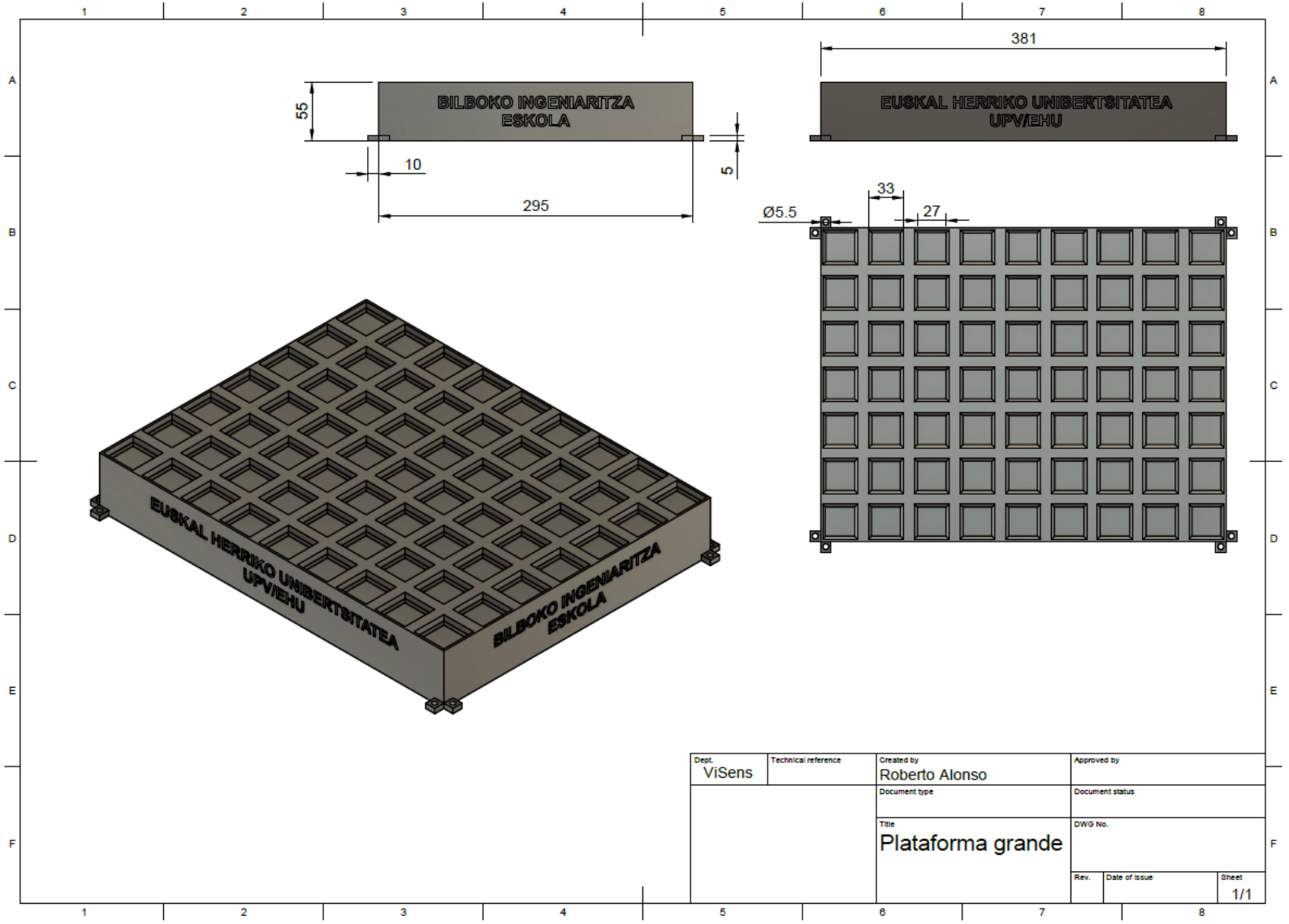
[58] “UNE-EN ISO 10218-1:2012 Robots y dispositivos robóticos. Requi...” https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0049289 (accessed May 25, 2022).

[59] “UNE-EN ISO 10218-2:2011 Robots y dispositivos robóticos. Requi...” https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0048668 (accessed May 25, 2022).

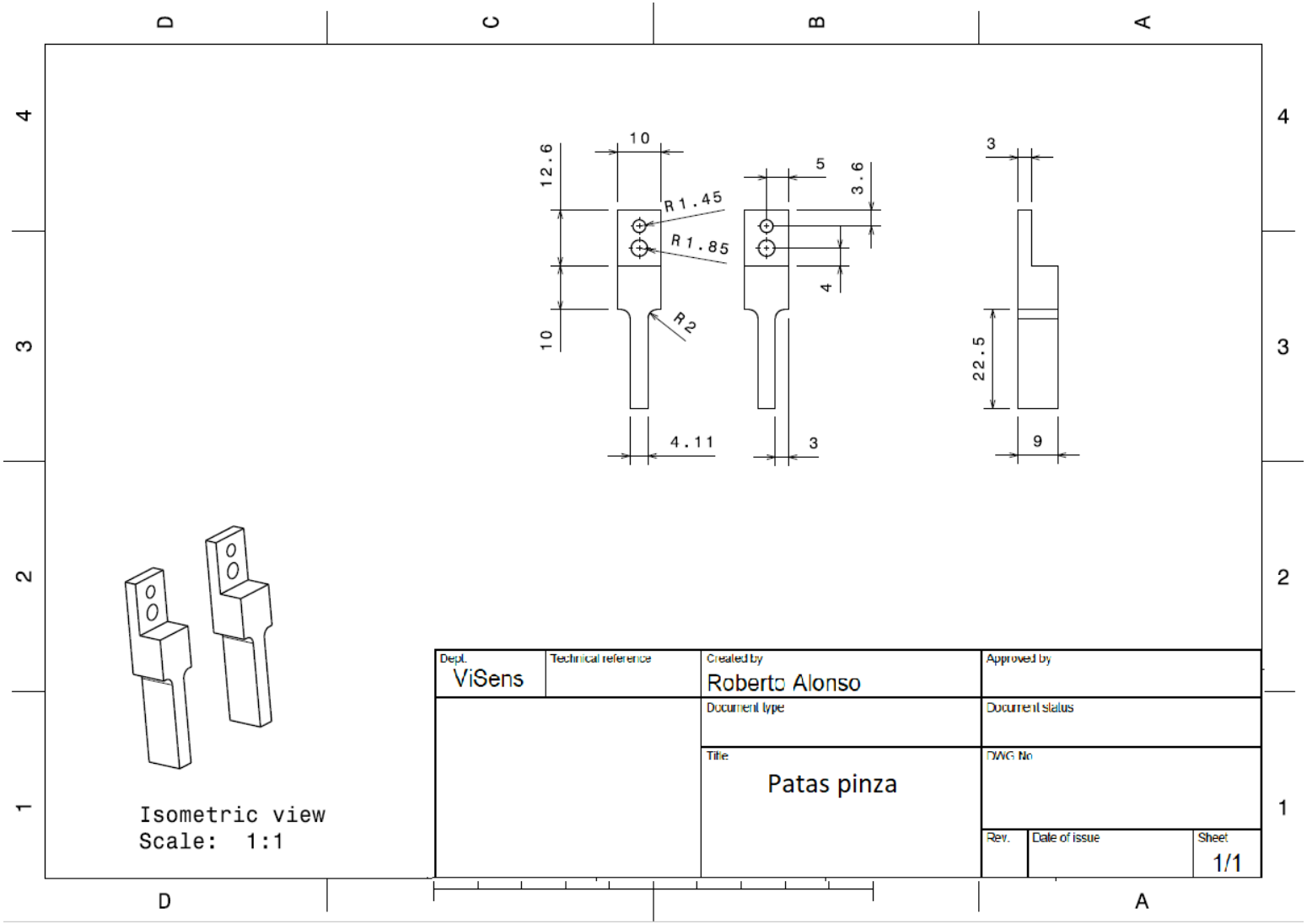
ANEXO I:

PLANOS

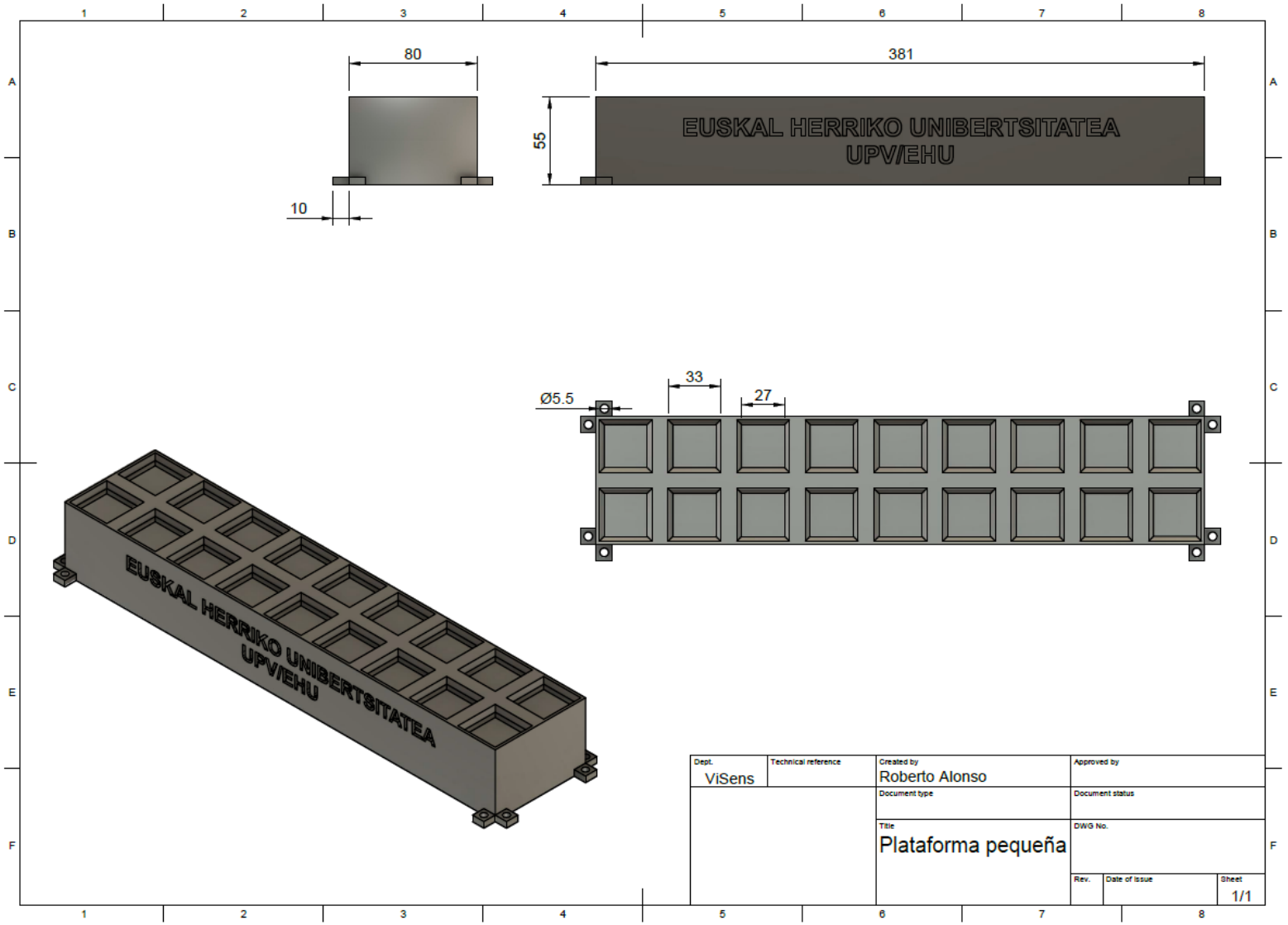
I.I. Plano plataforma grande.



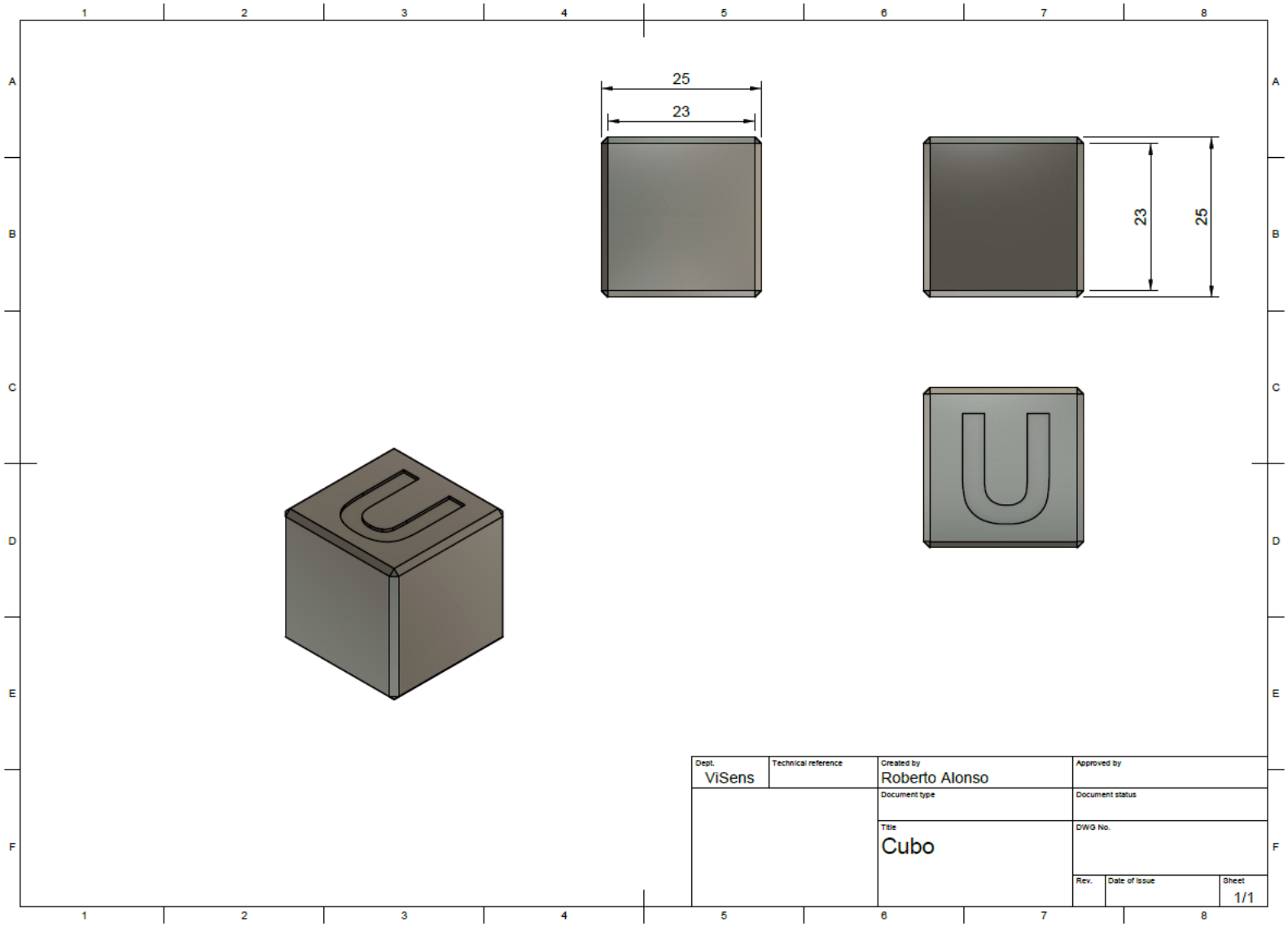
I.II. Plano patas pinza.



I.III. Plano plataforma pequeña.



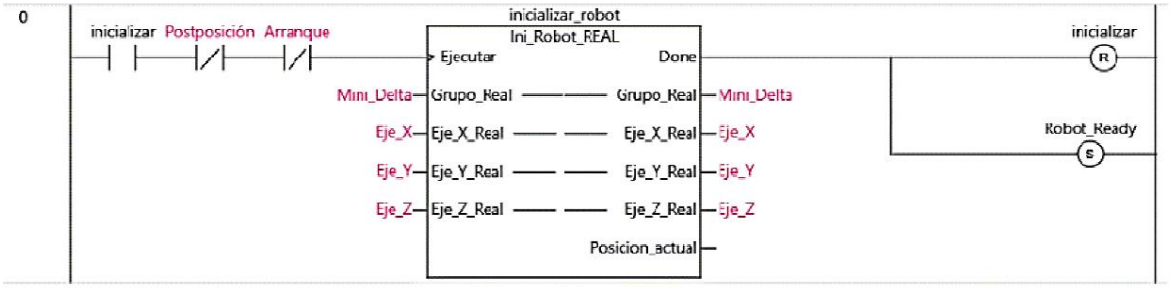
I.IV. Plano cubos.



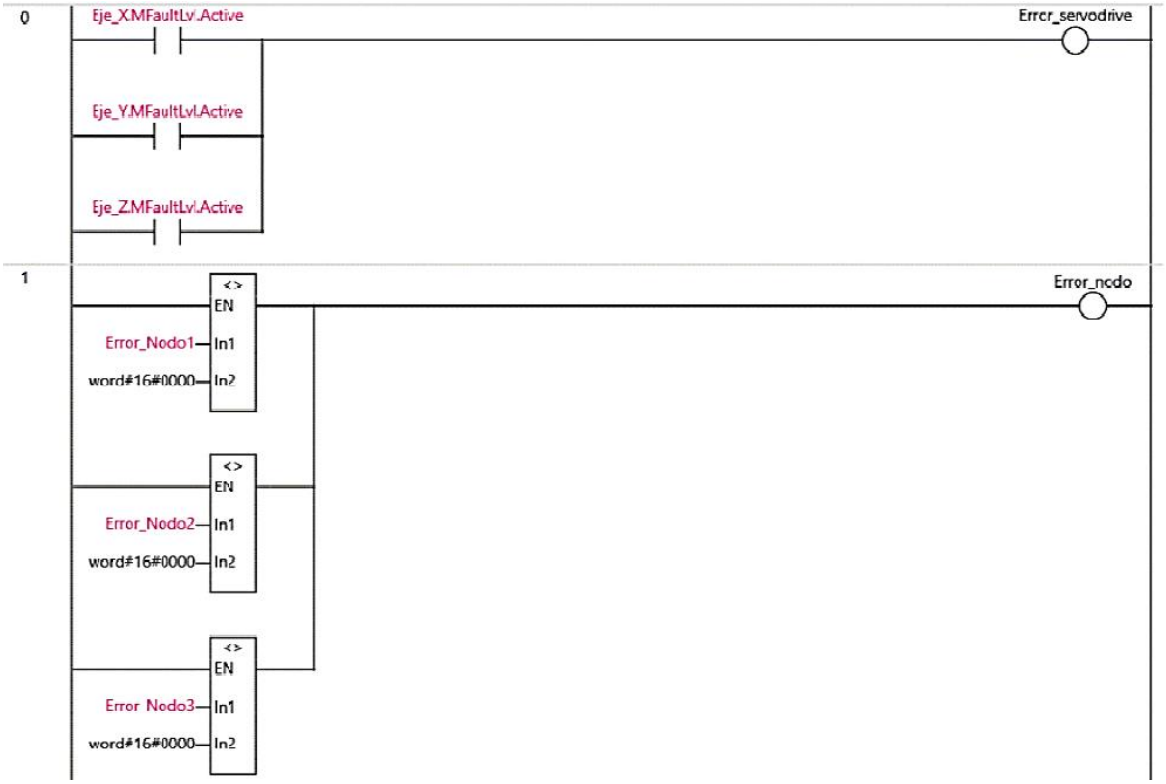
Anexo II:

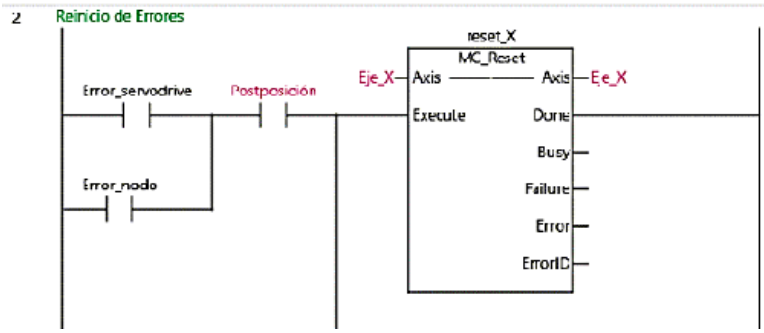
CÓDIGO SYSMAC STUDIO

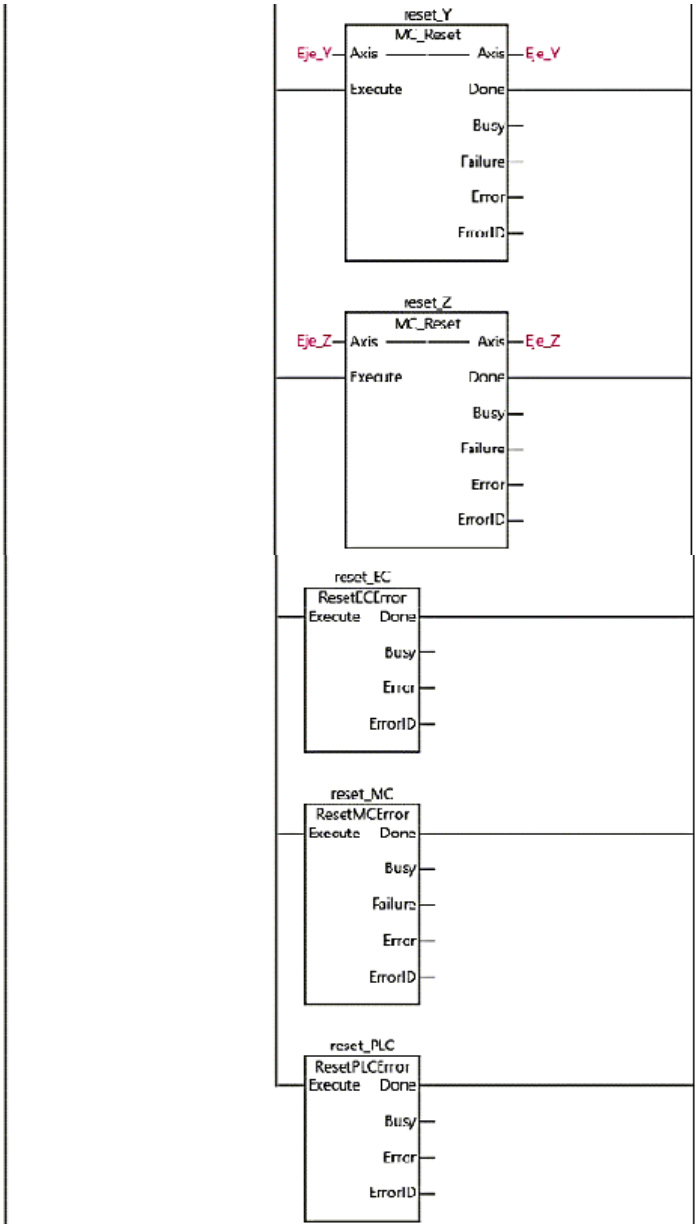
II.I. Inicialización del robot.



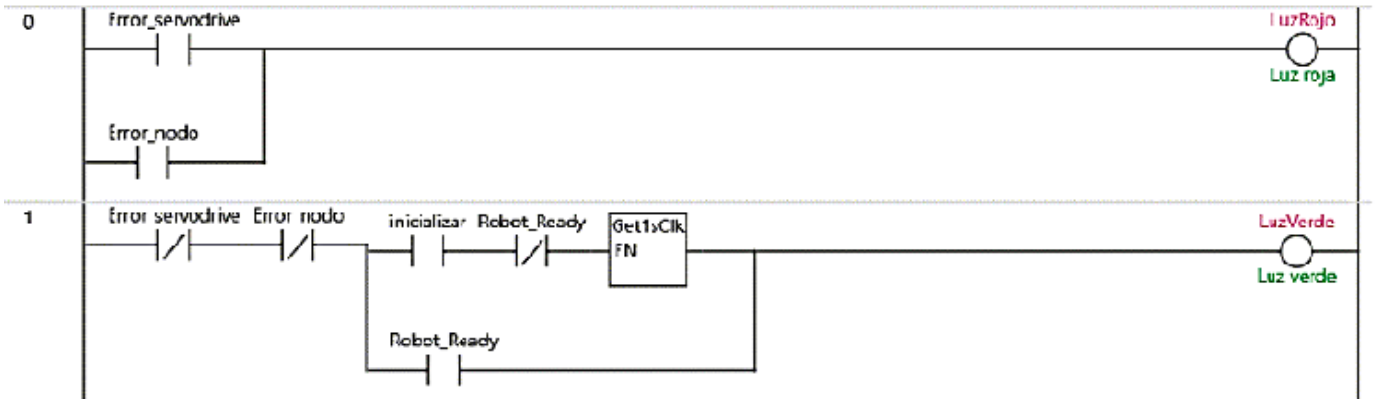
II.I.I. Reset de errores del robot.



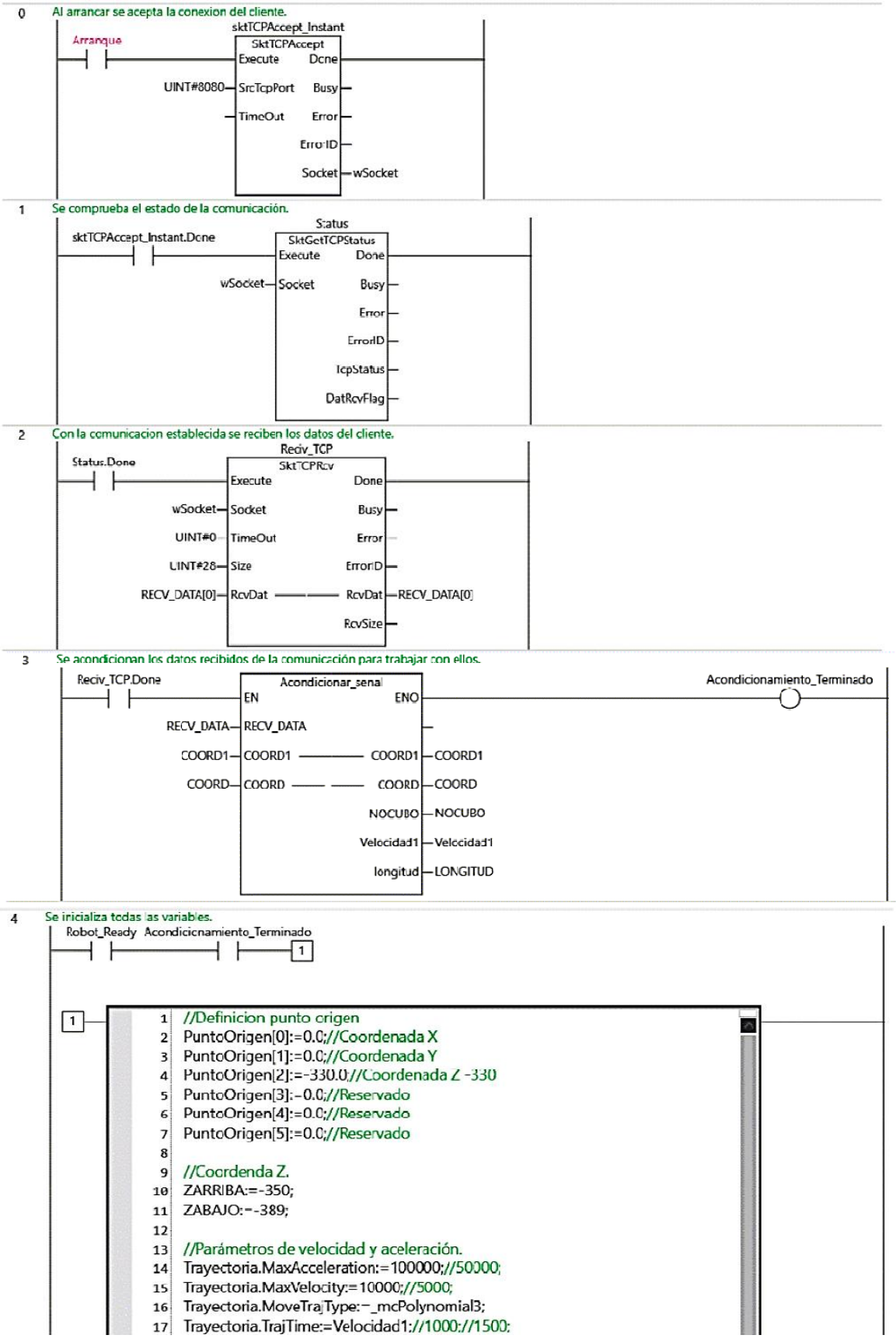




II.I.II. Iluminación de la botonera.



II.II. Movimiento del robot.



```
18
19  // Coordenadas cubos de la plataforma grande.
20
21  //Cubo 1.
22  //Arriba.
23  Grande1_arriba[0]:=COORD[0];
24  Grande1_arriba[1]:=COORD[1];
25  Grande1_arriba[2]:=ZARRIBA;
26  Grande1_arriba[3]:=0;
27  Grande1_arriba[4]:=0;
28  Grande1_arriba[5]:=0;
29
30  //Abajo.
31  Grande1_abajo[0]:=COORD[0];
32  Grande1_abajo[1]:=COORD[1];
33  Grande1_abajo[2]:=ZABAJO;
34  Grande1_abajo[3]:=0;
35  Grande1_abajo[4]:=0;
36  Grande1_abajo[5]:=0;
37
38  //Cubo 2.
39  //Arriba.
40  Grande2_arriba[0]:=COORD[3];
41  Grande2_arriba[1]:=COORD[4];
42  Grande2_arriba[2]:=ZARRIBA;
43  Grande2_arriba[3]:=0;
44  Grande2_arriba[4]:=0;
45  Grande2_arriba[5]:=0;
46
47  //Abajo.
48  Grande2_abajo[0]:=COORD[3];
49  Grande2_abajo[1]:=COORD[4];
50  Grande2_abajo[2]:=ZABAJO;
51  Grande2_abajo[3]:=0;
52  Grande2_abajo[4]:=0;
53  Grande2_abajo[5]:=0;
54
55  //Cubo 3.
56  //Arriba.
57  Grande3_arriba[0]:=COORD[6];
59  Grande3_arriba[2]:=ZARRIBA;
60  Grande3_arriba[3]:=0;
61  Grande3_arriba[4]:=0;
62  Grande3_arriba[5]:=0;
63
64  //Abajo.
65  Grande3_abajo[0]:=COORD[6];
66  Grande3_abajo[1]:=COORD[7];
67  Grande3_abajo[2]:=ZABAJO;
68  Grande3_abajo[3]:=0;
69  Grande3_abajo[4]:=0;
70  Grande3_abajo[5]:=0;
71
72
73  //Cubo 4.
74  //Arriba.
75  Grande4_arriba[0]:=COORD[9];
76  Grande4_arriba[1]:=COORD[10];
77  Grande4_arriba[2]:=ZARRIBA;
78  Grande4_arriba[3]:=0;
79  Grande4_arriba[4]:=0;
80  Grande4_arriba[5]:=0;
81
82  //Abajo.
83  Grande4_abajo[0]:=COORD[9];
84  Grande4_abajo[1]:=COORD[10];
```

```
85  Grande4_abajo[2]:=ZABAJO;
86  Grande4_abajo[3]:=0;
87  Grande4_abajo[4]:=0;
88  Grande4_abajo[5]:=0;
89
90
91  //Cubo 5.
92  //Arriba.
93  Grande5_arriba[0]:=COORD[12];
94  Grande5_arriba[1]:=COORD[13];
95  Grande5_arriba[2]:=ZARRIBA;
96  Grande5_arriba[3]:=0;
97  Grande5_arriba[4]:=0;
98  Grande5_arriba[5]:=0;
99
100 //Abajo.
101 Grande5_abajo[0]:=COORD[12];
102 Grande5_abajo[1]:=COORD[13];
103 Grande5_abajo[2]:=ZABAJO;
104 Grande5_abajo[3]:=0;
105 Grande5_abajo[4]:=0;
106 Grande5_abajo[5]:=0;
107
108
109 //Cubo 6.
110 //Arriba.
111 Grande6_arriba[0]:=COORD[15];
112 Grande6_arriba[1]:=COORD[16];
113 Grande6_arriba[2]:=ZARRIBA;
114 Grande6_arriba[3]:=0;
115 Grande6_arriba[4]:=0;
116 Grande6_arriba[5]:=0;
117
118 //Abajo.
119 Grande6_abajo[0]:=COORD[15];
120 Grande6_abajo[1]:=COORD[16];
121 Grande6_abajo[2]:=ZABAJO;
122 Grande6_abajo[3]:=0;
123 Grande6_abajo[4]:=0;
124 Grande6_abajo[5]:=0;
125
126
127 //Cubo 7.
128 //Arriba.
129 Grande7_arriba[0]:=COORD[18];
130 Grande7_arriba[1]:=COORD[19];
131 Grande7_arriba[2]:=ZARRIBA;
132 Grande7_arriba[3]:=0;
133 Grande7_arriba[4]:=0;
134 Grande7_arriba[5]:=0;
135
136 //Abajo.
137 Grande7_abajo[0]:=COORD[18];
138 Grande7_abajo[1]:=COORD[19];
139 Grande7_abajo[2]:=ZABAJO;
140 Grande7_abajo[3]:=0;
141 Grande7_abajo[4]:=0;
142 Grande7_abajo[5]:=0;
143
144
145 //Cubo 8.
146 //Arriba.
147 Grande8_arriba[0]:=COORD[21];
148 Grande8_arriba[1]:=COORD[22];
149 Grande8_arriba[2]:=ZARRIBA;
150 Grande8_arriba[3]:=0;
151 Grande8_arriba[4]:=0;
152 Grande8_arriba[5]:=0;
```

```
153
154 //Abajo.
155 Grande8_abajo[0]:=COORD[21];
156 Grande8_abajo[1]:=COORD[22];
157 Grande8_abajo[2]:=ZABAJO;
158 Grande8_abajo[3]:=0;
159 Grande8_abajo[4]:=0;
160 Grande8_abajo[5]:=0;
161
162
163 //Cubo 9.
164 //Arriba.
165 Grande9_arriba[0]:=COORD[24];
166 Grande9_arriba[1]:=COORD[25];
167 Grande9_arriba[2]:=ZARRIBA;
168 Grande9_arriba[3]:=0;
169 Grande9_arriba[4]:=0;
170 Grande9_arriba[5]:=0;
171
172 //Abajo.
173 Grande9_abajo[0]:=COORD[24];
174 Grande9_abajo[1]:=COORD[25];
175 Grande9_abajo[2]:=ZABAJO;
176 Grande9_abajo[3]:=0;
177 Grande9_abajo[4]:=0;
178 Grande9_abajo[5]:=0;
179
180
181 //Coordenadas cubos de la plataforma pequeña.
182 //Cubo 1.
183 //Arriba.
184 Pequena1_arriba[0]:=-175;
185 Pequena1_arriba[1]:=153;
186 Pequena1_arriba[2]:=ZARRIBA;
187 Pequena1_arriba[3]:=0;
188 Pequena1_arriba[4]:=0;
189 Pequena1_arriba[5]:=0;
190
191 //Abajo.
192 Pequena1_abajo[0]:=-175;
193 Pequena1_abajo[1]:=153;
194 Pequena1_abajo[2]:=ZABAJO;
195 Pequena1_abajo[3]:=0;
196 Pequena1_abajo[4]:=0;
197 Pequena1_abajo[5]:=0;
198
199
200 //Cubo 2.
201 //Arriba.
202 Pequena2_arriba[0]:=-172;
203 Pequena2_arriba[1]:=109;
204 Pequena2_arriba[2]:=ZARRIBA;
205 Pequena2_arriba[3]:=0;
206 Pequena2_arriba[4]:=0;
207 Pequena2_arriba[5]:=0;
208
209 //Abajo
210 Pequena2_abajo[0]:=-172;
211 Pequena2_abajo[1]:=109;
212 Pequena2_abajo[2]:=ZABAJO;
213 Pequena2_abajo[3]:=0;
214 Pequena2_abajo[4]:=0;
215 Pequena2_abajo[5]:=0;
216
217
218 //Cubo 3.
```

```
219  //Arriba.
220  Pequena3_arriba[0]:=-170;
221  Pequena3_arriba[1]:=70;
222  Pequena3_arriba[2]:=ZARRIBA;
223  Pequena3_arriba[3]:=0;
224  Pequena3_arriba[4]:=0;
225  Pequena3_arriba[5]:=0;
226
227  //Abajo.
228  Pequena3_abajo[0]:=-170;
229  Pequena3_abajo[1]:=70;
230  Pequena3_abajo[2]:=ZABAJO;
231  Pequena3_abajo[3]:=0;
232  Pequena3_abajo[4]:=0;
233  Pequena3_abajo[5]:=0;
234
235
236  //Cubo 4.
237  //Arriba.
238  Pequena4_arriba[0]:=-169;
239  Pequena4_arriba[1]:=30;
240  Pequena4_arriba[2]:=ZARRIBA;
241  Pequena4_arriba[3]:=0;
242  Pequena4_arriba[4]:=0;
243  Pequena4_arriba[5]:=0;
244
245  //Abajo.
246  Pequena4_abajo[0]:=-169;
247  Pequena4_abajo[1]:=30;
248  Pequena4_abajo[2]:=ZABAJO;
249  Pequena4_abajo[3]:=0;
250  Pequena4_abajo[4]:=0;
251  Pequena4_abajo[5]:=0;
252
253
254  //Cubo 5.
255  //Arriba.
256  Pequena5_arriba[0]:=-167;
257  Pequena5_arriba[1]:=-10;
258  Pequena5_arriba[2]:=ZARRIBA;
259  Pequena5_arriba[3]:=0;
260  Pequena5_arriba[4]:=0;
261  Pequena5_arriba[5]:=0;
262
263  //Abajo.
264  Pequena5_abajo[0]:=-167;
265  Pequena5_abajo[1]:=-10;
266  Pequena5_abajo[2]:=ZABAJO;
267  Pequena5_abajo[3]:=0;
268  Pequena5_abajo[4]:=0;
269  Pequena5_abajo[5]:=0;
270
271
272  //Cubo 6.
273  //Arriba.
274  Pequena6_arriba[0]:=-167;
275  Pequena6_arriba[1]:=-46;
276  Pequena6_arriba[2]:=ZARRIBA;
277  Pequena6_arriba[3]:=0;
278  Pequena6_arriba[4]:=0;
279  Pequena6_arriba[5]:=0;
280
281  //Abajo.
282  Pequena6_abajo[0]:= 167;
283  Pequena6_abajo[1]:=-46;
284  Pequena6_abajo[2]:=ZABAJO;
285  Pequena6_abajo[3]:=0;
286  Pequena6_abajo[4]:=0;
287  Pequena6_abajo[5]:=0;
```

```
288
289
290 //Cubo 7.
291 //Arriba.
292 Pequena7_arriba[0]:=-167;
293 Pequena7_arriba[1]:= 84;
294 Pequena7_arriba[2]:=ZARRIBA;
295 Pequena7_arriba[3]:=0;
296 Pequena7_arriba[4]:=0;
297 Pequena7_arriba[5]:=0;
298
299 //Abajo.
300 Pequena7_abajo[0]:=-167;
301 Pequena7_abajo[1]:=-84;
302 Pequena7_abajo[2]:=ZABAJO;
303 Pequena7_abajo[3]:=0;
304 Pequena7_abajo[4]:=0;
305 Pequena7_abajo[5]:=0;
306
307
308 //Cubo 8.
309 //Arriba.
310 Pequena8_arriba[0]:=-167;
311 Pequena8_arriba[1]:=-124;
312 Pequena8_arriba[2]:=ZARRIBA;
313 Pequena8_arriba[3]:=0;
314 Pequena8_arriba[4]:=0;
315 Pequena8_arriba[5]:=0;
316
317 //Abajo.
318 Pequena8_abajo[0]:=-167;
319 Pequena8_abajo[1]:=-124;
320 Pequena8_abajo[2]:=ZABAJO;
321 Pequena8_abajo[3]:=0;
322 Pequena8_abajo[4]:=0;
323 Pequena8_abajo[5]:=0;
324
325
326 //Cubo 9.
327 //Arriba.
328 Pequena9_arriba[0]:=-168;
329 Pequena9_arriba[1]:=-169;
330 Pequena9_arriba[2]:=ZARRIBA;
331 Pequena9_arriba[3]:=0;
332 Pequena9_arriba[4]:=0;
333 Pequena9_arriba[5]:=0;
334
335 //Abajo.
336 Pequena9_abajo[0]:=-168;
337 Pequena9_abajo[1]:=-169;
338 Pequena9_abajo[2]:=ZABAJO;
339 Pequena9_abajo[3]:=0;
340 Pequena9_abajo[4]:=0;
341 Pequena9_abajo[5]:=0;
342
343 //Inicializacion.
344 HACER1:=FALSE;
345 HACER2:=FALSE;
346 HACER3:=FALSE;
347 HACER4:=FALSE;
348 HACER5:=FALSE;
349 HACER6:=FALSE;
350 HACER7:=FALSE;
351 HACER8:=FALSE;
352 HACER9:=FALSE;
353
354 ALTO1:=FALSE;
355 ALTO2:=FALSE;
```

```
356  ALTO3:=FALSE;
357  ALTO4:=FALSE;
358  ALTO5:=FALSE;
359  ALTO6:=FALSE;
360  ALTO7:=FALSE;
361  ALTO8:=FALSE;
362  ALTO9:=FALSE;
363
364  RECOGER1:=FALSE;
365  RECOGER2:=FALSE;
366  RECOGER3:=FALSE;
367  RECOGER4:=FALSE;
368  RECOGER5:=FALSE;
369  RECOGER6:=FALSE;
370  RECOGER7:=FALSE;
371  RECOGER8:=FALSE;
372  RECOGER9:=FALSE;
373
374  REPOSO:=FALSE;
375  FIN1:=FALSE;
376  STOP:=FALSE;
377  RECOGER_FIN:=FALSE;
378  EMPEZAR_RECOGER:=FALSE;
379  PROCESO_COMPLETADO:=FALSE;
380
381  //Posicion de reposo.
382  Posicion_reposo[0]:=200;
383  Posicion_reposo[1]:=0;
384  Posicion_reposo[2]:=ZARRIBA;
385  Posicion_reposo[3]:=0;
386  Posicion_reposo[4]:=0;
387  Posicion_reposo[5]:=0;
388
389  //dato de proceso completado.
390  SEND_DATA[0]:=1;
391  INI1_1:=TRUE;
```

5 Se definen las coordenadas en funcion de la longitud de la palabra para que esta se escriba centrada.

Robot_Ready Acondicionamiento_Terminado INI1_1 → 1

1

```
1   // Coordenadas cubos de la plataforma pequeña.
2   // Para una palabra de una letra.
3   IF LONGITUD=8 THEN
4
5     //Empiezo en el 5.
6     //Arriba.
7     Pequena1_arriba[0]:=-167;
8     Pequena1_arriba[1]:=-10;
9     Pequena1_arriba[2]:=ZARRIBA;
10    Pequena1_arriba[3]:=0;
11    Pequena1_arriba[4]:=0;
12    Pequena1_arriba[5]:=0;
13
14    //Abajo.
15    Pequena1_abajo[0]:=-167;
16    Pequena1_abajo[1]:=-10;
17    Pequena1_abajo[2]:=ZABAJO;
18    Pequena1_abajo[3]:=0;
19    Pequena1_abajo[4]:=0;
20    Pequena1_abajo[5]:=0;
21
22  ELSIF LONGITUD=7 THEN
23
24    //Empiezo en el 4.
25    //Arriba.
26    Pequena1_arriba[0]:=-169;
27    Pequena1_arriba[1]:=-30;
28    Pequena1_arriba[2]:=ZARRIBA;
```

```
29    Pequena1_arriba[3]:=0;
30    Pequena1_arriba[4]:=0;
31    Pequena1_arriba[5]:=0;
32
33    //Abajo.
34    Pequena1_abajo[0]:=-169;
35    Pequena1_abajo[1]:=30;
36    Pequena1_abajo[2]:=ZABAJO;
37    Pequena1_abajo[3]:=0;
38    Pequena1_abajo[4]:=0;
39    Pequena1_abajo[5]:=0;
40
41
42    //Cubo 2.
43    //Arriba.
44    Pequena2_arriba[0]:=-167;
45    Pequena2_arriba[1]:=-10;
46    Pequena2_arriba[2]:=ZARRIBA;
47    Pequena2_arriba[3]:=0;
48    Pequena2_arriba[4]:=0;
49    Pequena2_arriba[5]:=0;
50
51    //Abajo.
52    Pequena2_abajo[0]:=-167;
53    Pequena2_abajo[1]:=-10;
54    Pequena2_abajo[2]:=ZABAJO;
55    Pequena2_abajo[3]:=0;
56    Pequena2_abajo[4]:=0;
57    Pequena2_abajo[5]:=0;
59  ELSIF LONGITUD=6 THEN
60
61    //Empiezo en el 4.
62    //Arriba.
63    Pequena1_arriba[0]:=-169;
64    Pequena1_arriba[1]:=30;
65    Pequena1_arriba[2]:=ZARRIBA;
66    Pequena1_arriba[3]:=0;
67    Pequena1_arriba[4]:=0;
68    Pequena1_arriba[5]:=0;
69
70    //Abajo.
71    Pequena1_abajo[0]:=-169;
72    Pequena1_abajo[1]:=30;
73    Pequena1_abajo[2]:=ZABAJO;
74    Pequena1_abajo[3]:=0;
75    Pequena1_abajo[4]:=0;
76    Pequena1_abajo[5]:=0;
77
78
79    //Cubo 2.
80    //Arriba.
81    Pequena2_arriba[0]:=-167;
82    Pequena2_arriba[1]:=-10;
83    Pequena2_arriba[2]:=ZARRIBA;
84    Pequena2_arriba[3]:=0;
85    Pequena2_arriba[4]:=0;
86    Pequena2_arriba[5]:=0;
87
88    //Abajo.
89    Pequena2_abajo[0]:=-167;
90    Pequena2_abajo[1]:=-10;
91    Pequena2_abajo[2]:=ZABAJO;
92    Pequena2_abajo[3]:=0;
93    Pequena2_abajo[4]:=0;
94    Pequena2_abajo[5]:=0;
95
96
97    //Cubo 3.
98    //Arriba.
99    Pequena3_arriba[0]:=-167;
100   Pequena3_arriba[1]:=-46;
101   Pequena3_arriba[2]:=ZARRIBA;
```

```
    Pequena3_arriba[3]:=0;
    Pequena3_arriba[4]:=0;
    Pequena3_arriba[5]:=0;

    //Abajo.
    Pequena3_abajo[0]:=-167;
    Pequena3_abajo[1]:=-46;
    Pequena3_abajo[2]:=ZABAJO;
    Pequena3_abajo[3]:=0;
    Pequena3_abajo[4]:=0;
    Pequena3_abajo[5]:=0;

ELSIF LONGITUD=5 THEN

    //Empiezo en el 3.
    //Arriba.
    Pequena1_arriba[0]:=-170;
    Pequena1_arriba[1]:=70;
    Pequena1_arriba[2]:=ZARRIBA;
    Pequena1_arriba[3]:=0;
    Pequena1_arriba[4]:=0;
    Pequena1_arriba[5]:=0;

    //Abajo.
    Pequena1_abajo[0]:=-170;
    Pequena1_abajo[1]:=70;
    Pequena1_abajo[2]:=ZABAJO;
    Pequena1_abajo[3]:=0;
    Pequena1_abajo[4]:=0;
    Pequena1_abajo[5]:=0;


    //Cubo 2.
    //Arriba.
    Pequena2_arriba[0]:=-169;
    Pequena2_arriba[1]:=30;
    Pequena2_arriba[2]:=ZARRIBA;
    Pequena2_arriba[3]:=0;
    Pequena2_arriba[4]:=0;
    Pequena2_arriba[5]:=0;

    //Abajo.
    Pequena2_abajo[0]:=-169;
    Pequena2_abajo[1]:=30;
    Pequena2_abajo[2]:=ZABAJO;
    Pequena2_abajo[3]:=0;
    Pequena2_abajo[4]:=0;
    Pequena2_abajo[5]:=0;


    //Cubo 3.
    //Arriba.
    Pequena3_arriba[0]:=-167;
    Pequena3_arriba[1]:=-10;
    Pequena3_arriba[2]:=ZARRIBA;
    Pequena3_arriba[3]:=0;
    Pequena3_arriba[4]:=0;
    Pequena3_arriba[5]:=0;

    //Abajo.
    Pequena3_abajo[0]:=-167;
    Pequena3_abajo[1]:=-10;
    Pequena3_abajo[2]:=ZABAJO;
    Pequena3_abajo[3]:=0;
    Pequena3_abajo[4]:=0;
    Pequena3_abajo[5]:=0;

```

```
169
170     //Cubo 4.
171     //Arriba.
172     Pequena4_arriba[0]:=-167;
173     Pequena4_arriba[1]:= 46;
174     Pequena4_arriba[2]:=ZARRIBA;
175     Pequena4_arriba[3]:=0;
176     Pequena4_arriba[4]:=0;
177     Pequena4_arriba[5]:=0;
178
179     //Abajo.
180     Pequena4_abajo[0]:=-167;
181     Pequena4_abajo[1]:=-46;
182     Pequena4_abajo[2]:=ZABAJO;
183     Pequena4_abajo[3]:=0;
184     Pequena4_abajo[4]:=0;
185     Pequena4_abajo[5]:=0;
186
187 ELSIF LONGITUD=4 THEN
188
189     //Empiezo en el 3.
190     //Arriba.
191     Pequena1_arriba[0]:=-170;
192     Pequena1_arriba[1]:=70;
193     Pequena1_arriba[2]:=ZARRIBA;
194     Pequena1_arriba[3]:=0;
195     Pequena1_arriba[4]:=0;
196     Pequena1_arriba[5]:=0;
197
198     //Abajo.
199     Pequena1_abajo[0]:=-170;
200     Pequena1_abajo[1]:=70;
201     Pequena1_abajo[2]:=ZABAJO;
202     Pequena1_abajo[3]:=0;
203     Pequena1_abajo[4]:=0;
204     Pequena1_abajo[5]:=0;
205
206
207     //Cubo 2.
208     //Arriba.
209     Pequena2_arriba[0]:=-169;
210     Pequena2_arriba[1]:=30;
211     Pequena2_arriba[2]:=ZARRIBA;
212     Pequena2_arriba[3]:=0;
213     Pequena2_arriba[4]:=0;
214     Pequena2_arriba[5]:=0;
215
216     //Abajo.
217     Pequena2_abajo[0]:=-169;
218     Pequena2_abajo[1]:=30;
219     Pequena2_abajo[2]:=ZABAJO;
220     Pequena2_abajo[3]:=0;
221     Pequena2_abajo[4]:=0;
222     Pequena2_abajo[5]:=0;
223
224
225     //Cubo 3.
226     //Arriba.
227     Pequena3_arriba[0]:=-167;
228     Pequena3_arriba[1]:=-10;
229     Pequena3_arriba[2]:=ZARRIBA;
230     Pequena3_arriba[3]:=0;
231     Pequena3_arriba[4]:=0;
232     Pequena3_arriba[5]:=0;
233
234     //Abajo.
```

```
235     Pequena3_abajo[0]:=-167;
236     Pequena3_abajo[1]:=-10;
237     Pequena3_abajo[2]:=ZABAJO;
238     Pequena3_abajo[3]:=0;
239     Pequena3_abajo[4]:=0;
240     Pequena3_abajo[5]:=0;
241
242
243     //Cubo 4.
244     //Arriba.
245     Pequena4_arriba[0]:=-167;
246     Pequena4_arriba[1]:=-46;
247     Pequena4_arriba[2]:=ZARRIBA;
248     Pequena4_arriba[3]:=0;
249     Pequena4_arriba[4]:=0;
250     Pequena4_arriba[5]:=0;
251
252     //Abajo.
253     Pequena4_abajo[0]:=-167;
254     Pequena4_abajo[1]:=-46;
255     Pequena4_abajo[2]:=ZABAJO;
256     Pequena4_abajo[3]:=0;
257     Pequena4_abajo[4]:=0;
258     Pequena4_abajo[5]:=0;
259
260
261     //Cubo 5.
262     //Arriba.
263     Pequena5_arriba[0]:=-167;
264     Pequena5_arriba[1]:=-84;
265     Pequena5_arriba[2]:=ZARRIBA;
266     Pequena5_arriba[3]:=0;
267     Pequena5_arriba[4]:=0;
268     Pequena5_arriba[5]:=0;
269
270     //Abajo.
271     Pequena5_abajo[0]:=-167;
272     Pequena5_abajo[1]:=-84;
273     Pequena5_abajo[2]:=ZABAJO;
274     Pequena5_abajo[3]:=0;
275     Pequena5_abajo[4]:=0;
276     Pequena5_abajo[5]:=0;
277
278
279 ELSIF LONGITUD=3 THEN
280
281     //Empiezo en el 2.
282     //Arriba.
283     Pequena1_arriba[0]:=-172;
284     Pequena1_arriba[1]:=109;
285     Pequena1_arriba[2]:=ZARRIBA;
286     Pequena1_arriba[3]:=0;
287     Pequena1_arriba[4]:=0;
288     Pequena1_arriba[5]:=0;
289
290     //Abajo.
291     Pequena1_abajo[0]:=-172;
292     Pequena1_abajo[1]:=109;
293     Pequena1_abajo[2]:=ZABAJO;
294     Pequena1_abajo[3]:=0;
295     Pequena1_abajo[4]:=0;
296     Pequena1_abajo[5]:=0;
297
298
```

```
299    //Cubo 2.
300    //Arriba.
301    Pequena2_arriba[0]:=-170;
302    Pequena2_arriba[1]:=70;
303    Pequena2_arriba[2]:=ZARRIBA;
304    Pequena2_arriba[3]:=0;
305    Pequena2_arriba[4]:=0;
306    Pequena2_arriba[5]:=0;
307
308    //Abajo.
309    Pequena2_abajo[0]:=-170;
310    Pequena2_abajo[1]:=70;
311    Pequena2_abajo[2]:=ZABAJO;
312    Pequena2_abajo[3]:=0;
313    Pequena2_abajo[4]:=0;
314    Pequena2_abajo[5]:=0;
315
316
317    //Cubo 3.
318    //Arriba.
319    Pequena3_arriba[0]:=-169;
320    Pequena3_arriba[1]:=30;
321    Pequena3_arriba[2]:=ZARRIBA;
322    Pequena3_arriba[3]:=0;
323    Pequena3_arriba[4]:=0;
324    Pequena3_arriba[5]:=0;
325
326    //Abajo.
327    Pequena3_abajo[0]:=-169;
328    Pequena3_abajo[1]:=30;
329    Pequena3_abajo[2]:=ZABAJO;
330    Pequena3_abajo[3]:=0;
331    Pequena3_abajo[4]:=0;
332    Pequena3_abajo[5]:=0;
333
334
335    //Cubo 4.
336    //Arriba.
337    Pequena4_arriba[0]:=-167;
338    Pequena4_arriba[1]:=-10;
339    Pequena4_arriba[2]:=ZARRIBA;
340    Pequena4_arriba[3]:=0;
341    Pequena4_arriba[4]:=0;
342    Pequena4_arriba[5]:=0;
343
344    //Abajo.
345    Pequena4_abajo[0]:=-167;
346    Pequena4_abajo[1]:=-10;
347    Pequena4_abajo[2]:=ZABAJO;
348    Pequena4_abajo[3]:=0;
349    Pequena4_abajo[4]:=0;
350    Pequena4_abajo[5]:=0;
351
352
353    //Cubo 5.
354    //Arriba.
355    Pequena5_arriba[0]:=-167;
356    Pequena5_arriba[1]:=-46;
357    Pequena5_arriba[2]:=ZARRIBA;
358    Pequena5_arriba[3]:=0;
359    Pequena5_arriba[4]:=0;
360    Pequena5_arriba[5]:=0;
361
362    //Abajo.
363    Pequena5_abajo[0]:=-167;
364    Pequena5_abajo[1]:=-46;
365    Pequena5_abajo[2]:=ZABAJO;
```

```
366     Pequena5_abajo[3]:=0;
367     Pequena5_abajo[4]:=0;
368     Pequena5_abajo[5]:=0;
369
370
371     //Cubo 6.
372     //Arriba.
373     Pequena6_arriba[0]:=-167;
374     Pequena6_arriba[1]:=-84;
375     Pequena6_arriba[2]:=ZARRIBA;
376     Pequena6_arriba[3]:=0;
377     Pequena6_arriba[4]:=0;
378     Pequena6_arriba[5]:=0;
379
380     //Abajo.
381     Pequena6_abajo[0]:=-167;
382     Pequena6_abajo[1]:=-84;
383     Pequena6_abajo[2]:=ZABAJO;
384     Pequena6_abajo[3]:=0;
385     Pequena6_abajo[4]:=0;
386     Pequena6_abajo[5]:=0;
387
388 ELSIF LONGITUD=2 THEN
389
390     //Empiezo en el 2.
391     //Arriba.
392     Pequena1_arriba[0]:=-172;
393     Pequena1_arriba[1]:=109;
394     Pequena1_arriba[2]:=ZARRIBA;
395     Pequena1_arriba[3]:=0;
396     Pequena1_arriba[4]:=0;
397     Pequena1_arriba[5]:=0;
398
399     //Abajo.
400     Pequena1_abajo[0]:=-172;
401     Pequena1_abajo[1]:=109;
402     Pequena1_abajo[2]:=ZABAJO;
403     Pequena1_abajo[3]:=0;
404     Pequena1_abajo[4]:=0;
405     Pequena1_abajo[5]:=0;
406
407
408     //Cubo 2.
409     //Arriba.
410     Pequena2_arriba[0]:=-170;
411     Pequena2_arriba[1]:=70;
412     Pequena2_arriba[2]:=ZARRIBA;
413     Pequena2_arriba[3]:=0;
414     Pequena2_arriba[4]:=0;
415     Pequena2_arriba[5]:=0;
416
417     //Abajo.
418     Pequena2_abajo[0]:=-170;
419     Pequena2_abajo[1]:=70;
420     Pequena2_abajo[2]:=ZABAJO;
421     Pequena2_abajo[3]:=0;
422     Pequena2_abajo[4]:=0;
423     Pequena2_abajo[5]:=0;
424
425
426     //Cubo 3.
427     //Arriba.
428     Pequena3_arriba[0]:=-169;
429     Pequena3_arriba[1]:=30;
430     Pequena3_arriba[2]:=ZARRIBA;
431     Pequena3_arriba[3]:=0;
432     Pequena3_arriba[4]:=0;
433     Pequena3_arriba[5]:=0;
```

```
435   //Abajo.
436   Pequena3_abajo[0]:=-169;
437   Pequena3_abajo[1]:=30;
438   Pequena3_abajo[2]:=ZABAJO;
439   Pequena3_abajo[3]:=0;
440   Pequena3_abajo[4]:=0;
441   Pequena3_abajo[5]:=0;
442
443
444   //Cubo 4.
445   //Arriba.
446   Pequena4_arriba[0]:=-167;
447   Pequena4_arriba[1]:=-10;
448   Pequena4_arriba[2]:=ZARRIBA;
449   Pequena4_arriba[3]:=0;
450   Pequena4_arriba[4]:=0;
451   Pequena4_arriba[5]:=0;
452
453   //Abajo.
454   Pequena4_abajo[0]:=-167;
455   Pequena4_abajo[1]:=-10;
456   Pequena4_abajo[2]:=ZABAJO;
457   Pequena4_abajo[3]:=0;
458   Pequena4_abajo[4]:=0;
459   Pequena4_abajo[5]:=0;
460
461
462   //Cubo 5.
463   //Arriba.
464   Pequena5_arriba[0]:=-167;
465   Pequena5_arriba[1]:= 46;
466   Pequena5_arriba[2]:=ZARRIBA;
467   Pequena5_arriba[3]:=0;
468   Pequena5_arriba[4]:=0;
469   Pequena5_arriba[5]:=0;
470
471   //Abajo.
472   Pequena5_abajo[0]:=-167;
473   Pequena5_abajo[1]:=-46;
474   Pequena5_abajo[2]:=ZABAJO;
475   Pequena5_abajo[3]:=0;
476   Pequena5_abajo[4]:=0;
477   Pequena5_abajo[5]:=0;
478
479
480   //Cubo 6.
481   //Arriba
482   Pequena6_arriba[0]:=-167;
483   Pequena6_arriba[1]:= 84;
484   Pequena6_arriba[2]:=ZARRIBA;
485   Pequena6_arriba[3]:=0;
486   Pequena6_arriba[4]:=0;
487   Pequena6_arriba[5]:=0;
488
489   //Abajo.
490   Pequena6_abajo[0]:=-167;
491   Pequena6_abajo[1]:=-84;
492   Pequena6_abajo[2]:=ZABAJO;
493   Pequena6_abajo[3]:=0;
494   Pequena6_abajo[4]:=0;
495   Pequena6_abajo[5]:=0;
496
497
```

```
498     //Cubo 7.
499     //Arriba
500     Pequena7_arriba[0]:=-167;
501     Pequena7_arriba[1]:=-124;
502     Pequena7_arriba[2]:=ZARRIBA;
503     Pequena7_arriba[3]:=0;
504     Pequena7_arriba[4]:=0;
505     Pequena7_arriba[5]:=0;
506
507     //Abajo.
508     Pequena7_abajo[0]:=-167;
509     Pequena7_abajo[1]:=-124;
510     Pequena7_abajo[2]:=ZABAJO;
511     Pequena7_abajo[3]:=0;
512     Pequena7_abajo[4]:=0;
513     Pequena7_abajo[5]:=0;
514
515 ELSIF LONGITUD=1 THEN
516
517     //Empezamos en el 1.
518     //Arriba.
519     Pequena1_arriba[0]:=-175;
520     Pequena1_arriba[1]:=153;
521     Pequena1_arriba[2]:=ZARRIBA;
522     Pequena1_arriba[3]:=0;
523     Pequena1_arriba[4]:=0;
524     Pequena1_arriba[5]:=0;
525
526     //Abajo.
527     Pequena1_abajo[0]:=-175;
528     Pequena1_abajo[1]:=153;
529     Pequena1_abajo[2]:=ZABAJO;
530     Pequena1_abajo[3]:=0;
531     Pequena1_abajo[4]:=0;
532     Pequena1_abajo[5]:=0;
533
534
535     //Cubo 2.
536     //Arriba.
537     Pequena2_arriba[0]:=-172;
538     Pequena2_arriba[1]:=109;
539     Pequena2_arriba[2]:=ZARRIBA;
540     Pequena2_arriba[3]:=0;
541     Pequena2_arriba[4]:=0;
542     Pequena2_arriba[5]:=0;
543
544     //Abajo.
545     Pequena2_abajo[0]:=-172;
546     Pequena2_abajo[1]:=109;
547     Pequena2_abajo[2]:=ZABAJO;
548     Pequena2_abajo[3]:=0;
549     Pequena2_abajo[4]:=0;
550     Pequena2_abajo[5]:=0;
551
552
553     //Cubo 3.
554     //Arriba.
555     Pequena3_arriba[0]:= 170;
556     Pequena3_arriba[1]:=70;
557     Pequena3_arriba[2]:=ZARRIBA;
558     Pequena3_arriba[3]:=0;
559     Pequena3_arriba[4]:=0;
```

```
560     Pequena3_arriba[5]:=0;
561
562     //Abajo.
563     Pequena3_abajo[0]:=-170;
564     Pequena3_abajo[1]:=70;
565     Pequena3_abajo[2]:=ZABAJO;
566     Pequena3_abajo[3]:=0;
567     Pequena3_abajo[4]:=0;
568     Pequena3_abajo[5]:=0;
569
570
571     //Cubo 4.
572     //Arriba.
573     Pequena4_arriba[0]:= 169;
574     Pequena4_arriba[1]:=30;
575     Pequena4_arriba[2]:=ZARRIBA;
576     Pequena4_arriba[3]:=0;
577     Pequena4_arriba[4]:=0;
578     Pequena4_arriba[5]:=0;
579
580     //Abajo.
581     Pequena4_abajo[0]:=-169;
582     Pequena4_abajo[1]:=30;
583     Pequena4_abajo[2]:=ZABAJO;
584     Pequena4_abajo[3]:=0;
585     Pequena4_abajo[4]:=0;
586     Pequena4_abajo[5]:=0;
587
588
589     //Cubo 5.
590     //Arriba.
591     Pequena5_arriba[0]:=-167;
592     Pequena5_arriba[1]:= 10;
593     Pequena5_arriba[2]:=ZARRIBA;
594     Pequena5_arriba[3]:=0;
595     Pequena5_arriba[4]:=0;
596     Pequena5_arriba[5]:=0;
597
598     //Abajo.
599     Pequena5_abajo[0]:=-167;
600     Pequena5_abajo[1]:=-10;
601     Pequena5_abajo[2]:=ZABAJO;
602     Pequena5_abajo[3]:=0;
603     Pequena5_abajo[4]:=0;
604     Pequena5_abajo[5]:=0;
605
606
607     //Cubo 6.
608     //Arriba.
609     Pequena6_arriba[0]:=-167;
610     Pequena6_arriba[1]:=-46;
611     Pequena6_arriba[2]:=ZARRIBA;
612     Pequena6_arriba[3]:=0;
613     Pequena6_arriba[4]:=0;
614     Pequena6_arriba[5]:=0;
615
616     //Abajo.
617     Pequena6_abajo[0]:=-167;
618     Pequena6_abajo[1]:=-46;
619     Pequena6_abajo[2]:=ZABAJO;
620     Pequena6_abajo[3]:=0;
621     Pequena6_abajo[4]:=0;
622     Pequena6_abajo[5]:=0;
```

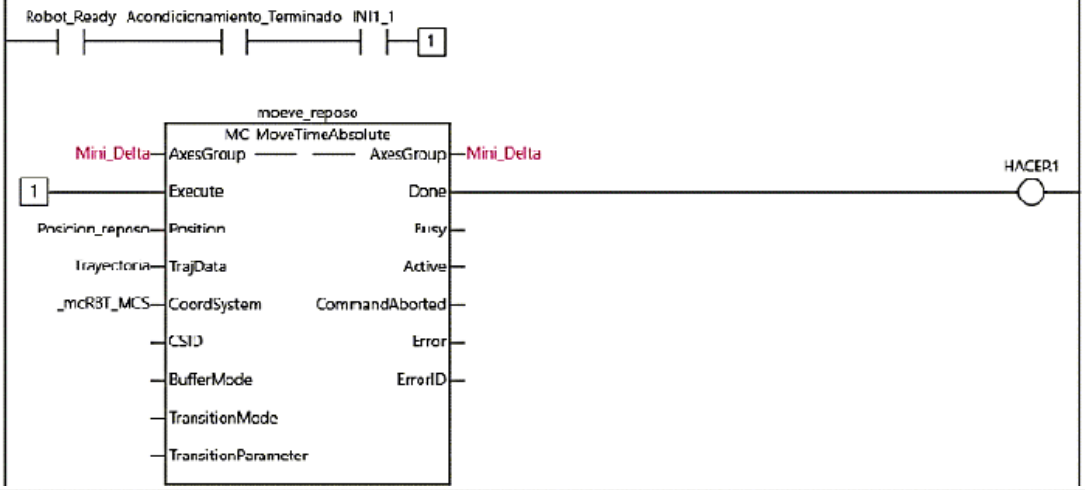
```
623
624
625     //Cubo 7.
626     //Arriba.
627     Pequena7_arriba[0]:=-167;
628     Pequena7_arriba[1]:=-84;
629     Pequena7_arriba[2]:=ZARRIBA;
630     Pequena7_arriba[3]:=0;
631     Pequena7_arriba[4]:=0;
632     Pequena7_arriba[5]:=0;
633
634     //Abajo.
635     Pequena7_abajo[0]:=-167;
636     Pequena7_abajo[1]:=-84;
637     Pequena7_abajo[2]:=ZABAJO;
638     Pequena7_abajo[3]:=0;
639     Pequena7_abajo[4]:=0;
640     Pequena7_abajo[5]:=0;
641
642
643     //Cubo 8.
644     //Arriba.
645     Pequena8_arriba[0]:=-167;
646     Pequena8_arriba[1]:=-124;
647     Pequena8_arriba[2]:=ZARRIBA;
648     Pequena8_arriba[3]:=0;
649     Pequena8_arriba[4]:=0;
650     Pequena8_arriba[5]:=0;
651
652     //Abajo.
653     Pequena8_abajo[0]:=-167;
654     Pequena8_abajo[1]:=-124;
655     Pequena8_abajo[2]:=ZABAJO;
656     Pequena8_abajo[3]:=0;
657     Pequena8_abajo[4]:=0;
658     Pequena8_abajo[5]:=0;
659
660
661 ELSIF LONGITUD=0 THEN
662
663     //Empezamos en el 1.
664     //Cubo 1.
665     //Arriba.
666     Pequena1_arriba[0]:=-175;
667     Pequena1_arriba[1]:=153;
668     Pequena1_arriba[2]:=ZARRIBA;
669     Pequena1_arriba[3]:=0;
670     Pequena1_arriba[4]:=0;
671     Pequena1_arriba[5]:=0;
672
673     //Abajo.
674     Pequena1_abajo[0]:= 175;
675     Pequena1_abajo[1]:=153;
676     Pequena1_abajo[2]:=ZABAJO;
677     Pequena1_abajo[3]:=0;
678     Pequena1_abajo[4]:=0;
679     Pequena1_abajo[5]:=0;
680
681
682     //Cubo 2.
683     //Arriba.
684     Pequena2_arriba[0]:=-172;
685     Pequena2_arriba[1]:=109;
```

```
686     Pequena2_arriba[2]:=ZARRIBA;
687     Pequena2_arriba[3]:=0;
688     Pequena2_arriba[4]:=0;
689     Pequena2_arriba[5]:=0;
690
691     //Abajo.
692     Pequena2_abajo[0]:=-172;
693     Pequena2_abajo[1]:=-109;
694     Pequena2_abajo[2]:=ZABAJO;
695     Pequena2_abajo[3]:=0;
696     Pequena2_abajo[4]:=0;
697     Pequena2_abajo[5]:=0;
698
699
700     //Cubo 3.
701     //Arriba.
702     Pequena3_arriba[0]:=-170;
703     Pequena3_arriba[1]:=70;
704     Pequena3_arriba[2]:=ZARRIBA;
705     Pequena3_arriba[3]:=0;
706     Pequena3_arriba[4]:=0;
707     Pequena3_arriba[5]:=0;
708
709     //Abajo.
710     Pequena3_abajo[0]:=-170;
711     Pequena3_abajo[1]:=70;
712     Pequena3_abajo[2]:=ZABAJO;
713     Pequena3_abajo[3]:=0;
714     Pequena3_abajo[4]:=0;
715     Pequena3_abajo[5]:=0;
716
717
718     //Cubo 4.
719     //Arriba.
720     Pequena4_arriba[0]:=-169;
721     Pequena4_arriba[1]:=30;
722     Pequena4_arriba[2]:=ZARRIBA;
723     Pequena4_arriba[3]:=0;
724     Pequena4_arriba[4]:=0;
725     Pequena4_arriba[5]:=0;
726
727     //Abajo.
728     Pequena4_abajo[0]:=-169;
729     Pequena4_abajo[1]:=30;
730     Pequena4_abajo[2]:=ZABAJO;
731     Pequena4_abajo[3]:=0;
732     Pequena4_abajo[4]:=0;
733     Pequena4_abajo[5]:=0;
734
735
736     //Cubo 5.
737     //Arriba.
738     Pequena5_arriba[0]:=-167;
739     Pequena5_arriba[1]:=-10;
740     Pequena5_arriba[2]:=ZARRIBA;
741     Pequena5_arriba[3]:=0;
742     Pequena5_arriba[4]:=0;
743     Pequena5_arriba[5]:=0;
744
745     //Abajo.
746     Pequena5_abajo[0]:=-167;
747     Pequena5_abajo[1]:=-10;
```

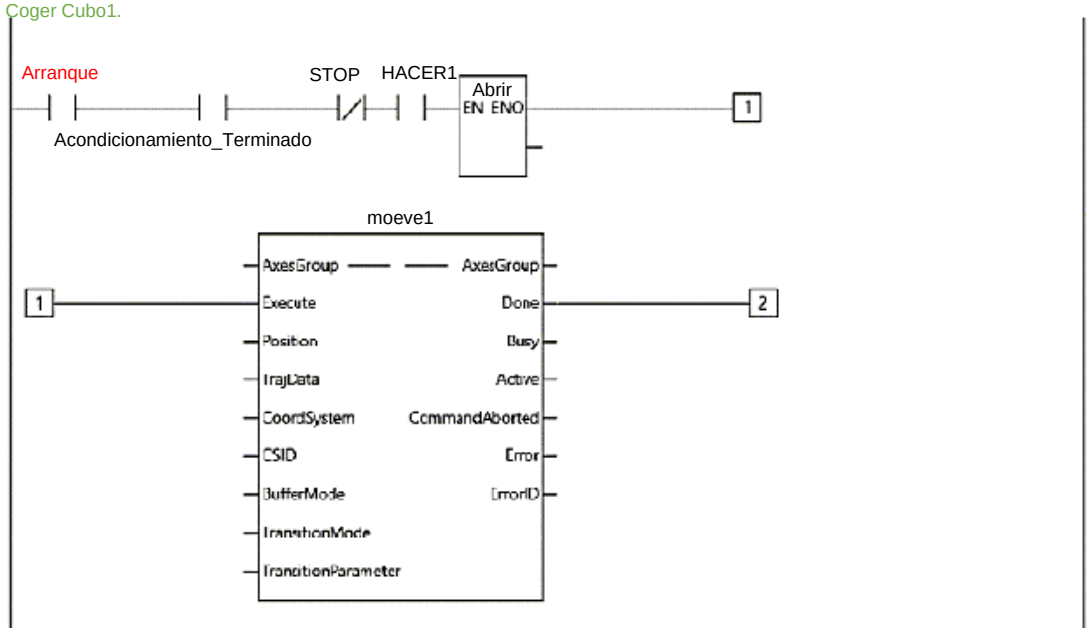
```
748     Pequena5_abajo[2]:=ZABAJO;
749     Pequena5_abajo[3]:=0;
750     Pequena5_abajo[4]:=0;
751     Pequena5_abajo[5]:=0;
752
753
754     //Cubo 6.
755     //Arriba.
756     Pequena6_arriba[0]:=-167;
757     Pequena6_arriba[1]:=-46;
758     Pequena6_arriba[2]:=ZARRIBA;
759     Pequena6_arriba[3]:=0;
760     Pequena6_arriba[4]:=0;
761     Pequena6_arriba[5]:=0;
762
763     //Abajo.
764     Pequena6_abajo[0]:=-167;
765     Pequena6_abajo[1]:=-46;
766     Pequena6_abajo[2]:=ZABAJO;
767     Pequena6_abajo[3]:=0;
768     Pequena6_abajo[4]:=0;
769     Pequena6_abajo[5]:=0;
770
771
772     //Cubo 7.
773     //Arriba.
774     Pequena7_arriba[0]:=-167;
775     Pequena7_arriba[1]:=-84;
776     Pequena7_arriba[2]:=ZARRIBA;
777     Pequena7_arriba[3]:=0;
778     Pequena7_arriba[4]:=0;
779     Pequena7_arriba[5]:=0;
780
781     //Abajo.
782     Pequena7_abajo[0]:=-167;
783     Pequena7_abajo[1]:= 84;
784     Pequena7_abajo[2]:=ZABAJO;
785     Pequena7_abajo[3]:=0;
786     Pequena7_abajo[4]:=0;
787     Pequena7_abajo[5]:=0;
788
789
790     //Cubo 8.
791     //Arriba.
792     Pequena8_arriba[0]:=-167;
793     Pequena8_arriba[1]:=-124;
794     Pequena8_arriba[2]:=ZARRIBA;
795     Pequena8_arriba[3]:=0;
796     Pequena8_arriba[4]:=0;
797     Pequena8_arriba[5]:=0;
798
799     //Abajo.
800     Pequena8_abajo[0]:=-167;
801     Pequena8_abajo[1]:=-124;
802     Pequena8_abajo[2]:=ZABAJO;
803     Pequena8_abajo[3]:=0;
804     Pequena8_abajo[4]:=0;
805     Pequena8_abajo[5]:=0;
806
807
808     //Cubo 9.
809     //Arriba.
810     Pequena9_arriba[0]:=-168;
```

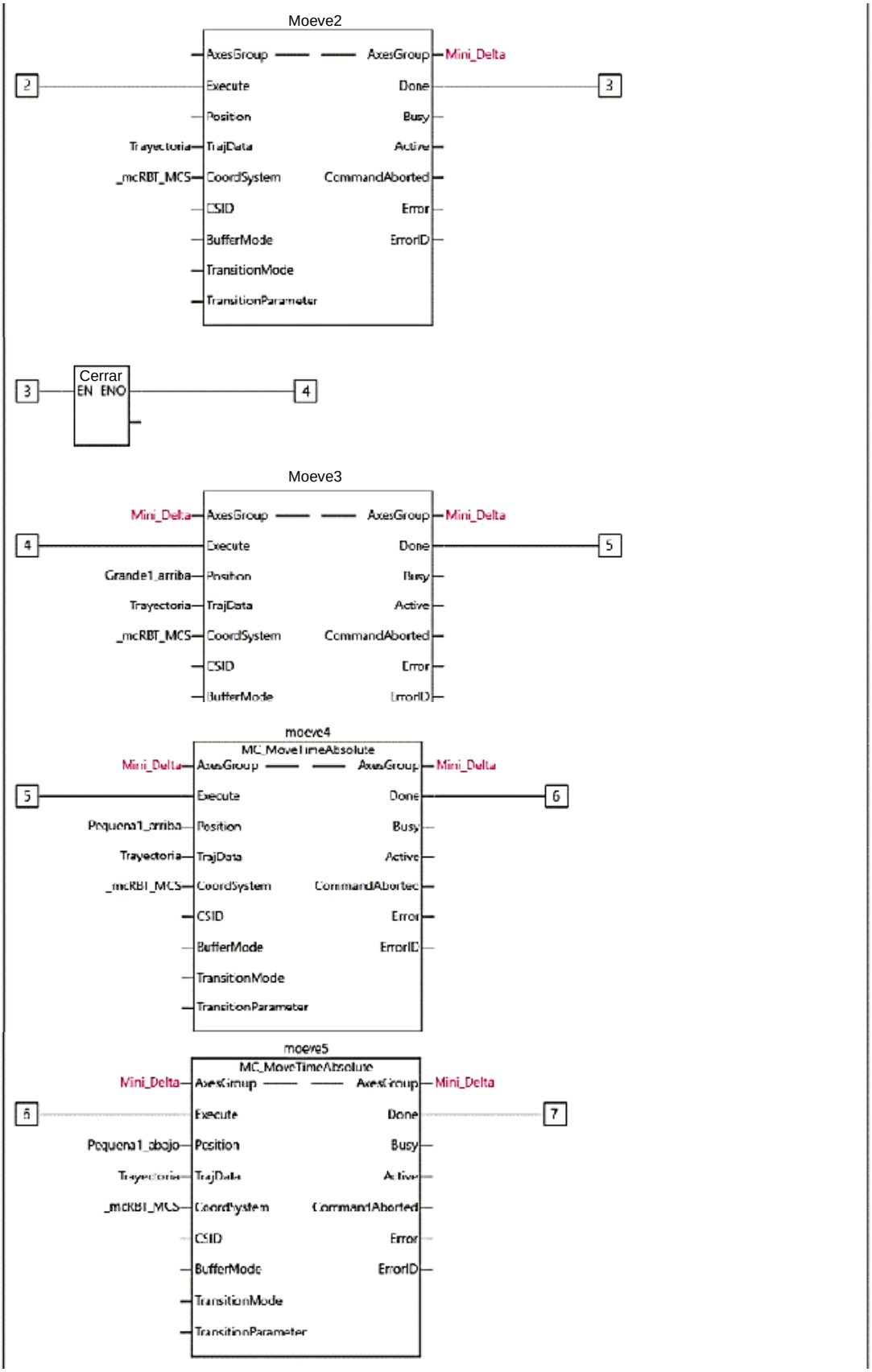
```
811     Pequena9_arriba[1]:=-169;
812     Pequena9_arriba[2]:=ZARRIBA;
813     Pequena9_arriba[3]:=0;
814     Pequena9_arriba[4]:=0;
815     Pequena9_arriba[5]:=0;
816
817     //Abajo.
818     Pequena9_abajo[0]:=-168;
819     Pequena9_abajo[1]:=-169;
820     Pequena9_abajo[2]:=ZABAJO;
821     Pequena9_abajo[3]:=0;
822     Pequena9_abajo[4]:=0;
823     Pequena9_abajo[5]:=0;
824
825 END_IF;
```

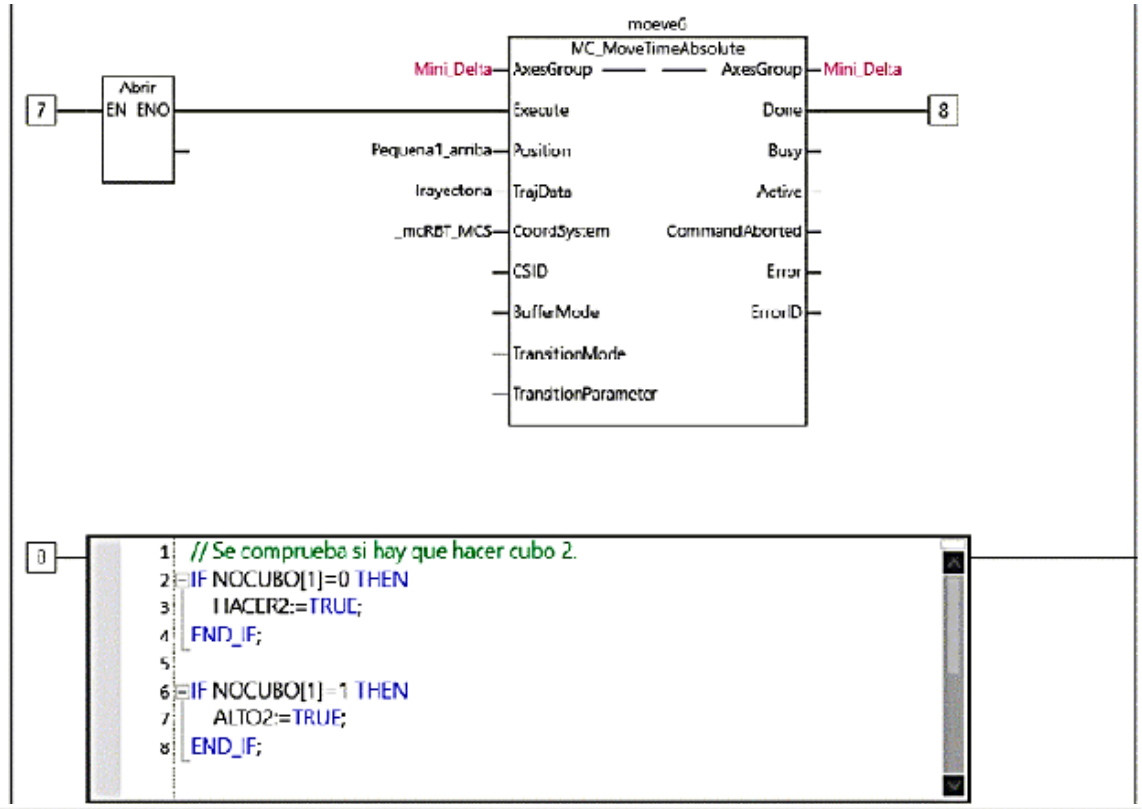
6 Se mueve el robot a reposo y se comienza a coger cubos.



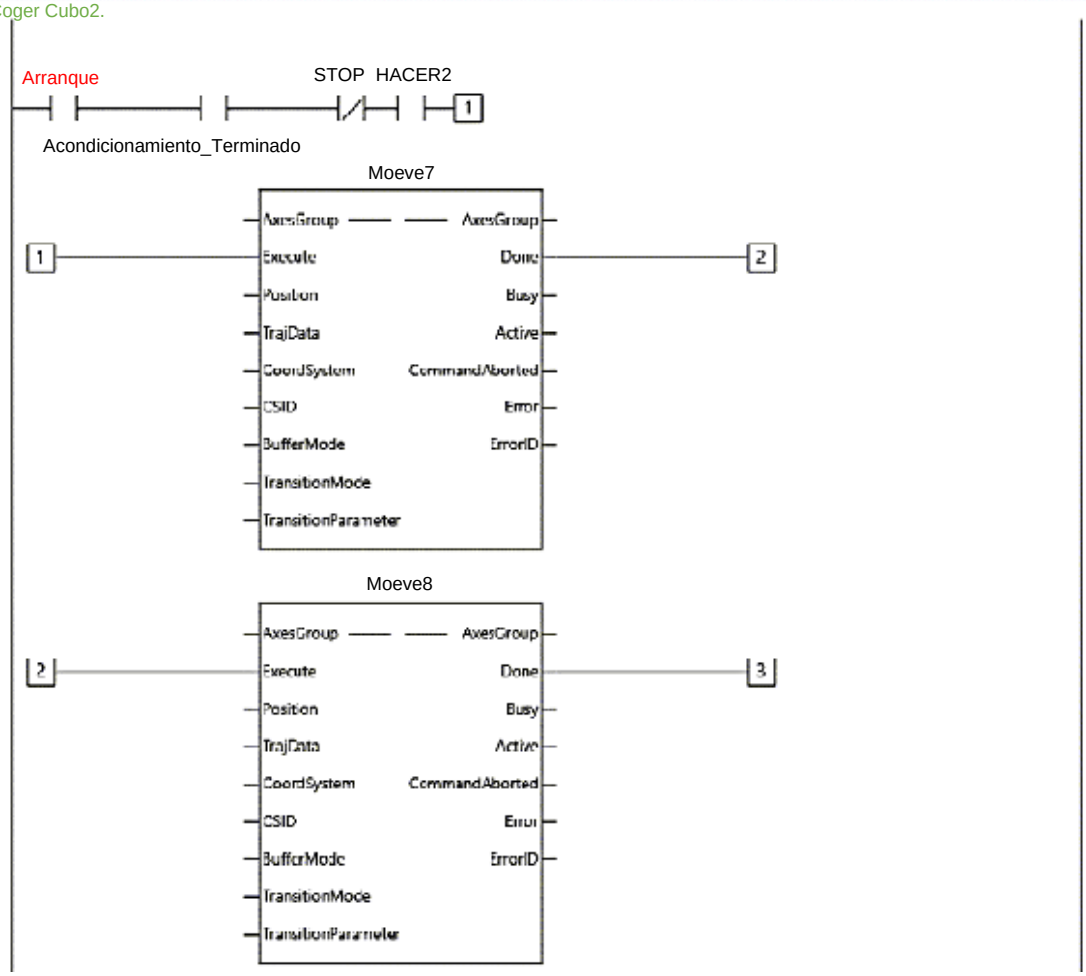
7 Coger Cubo1.

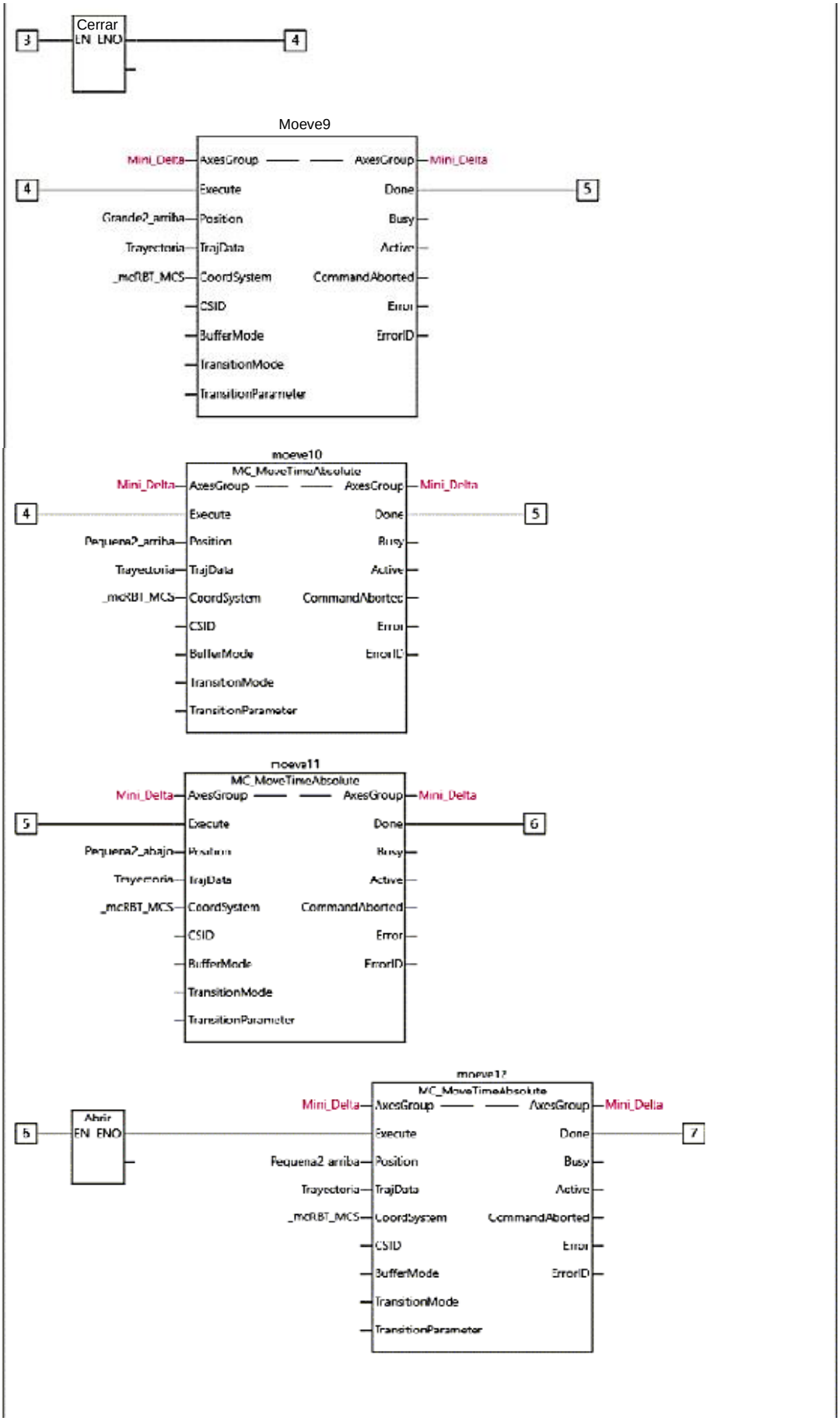


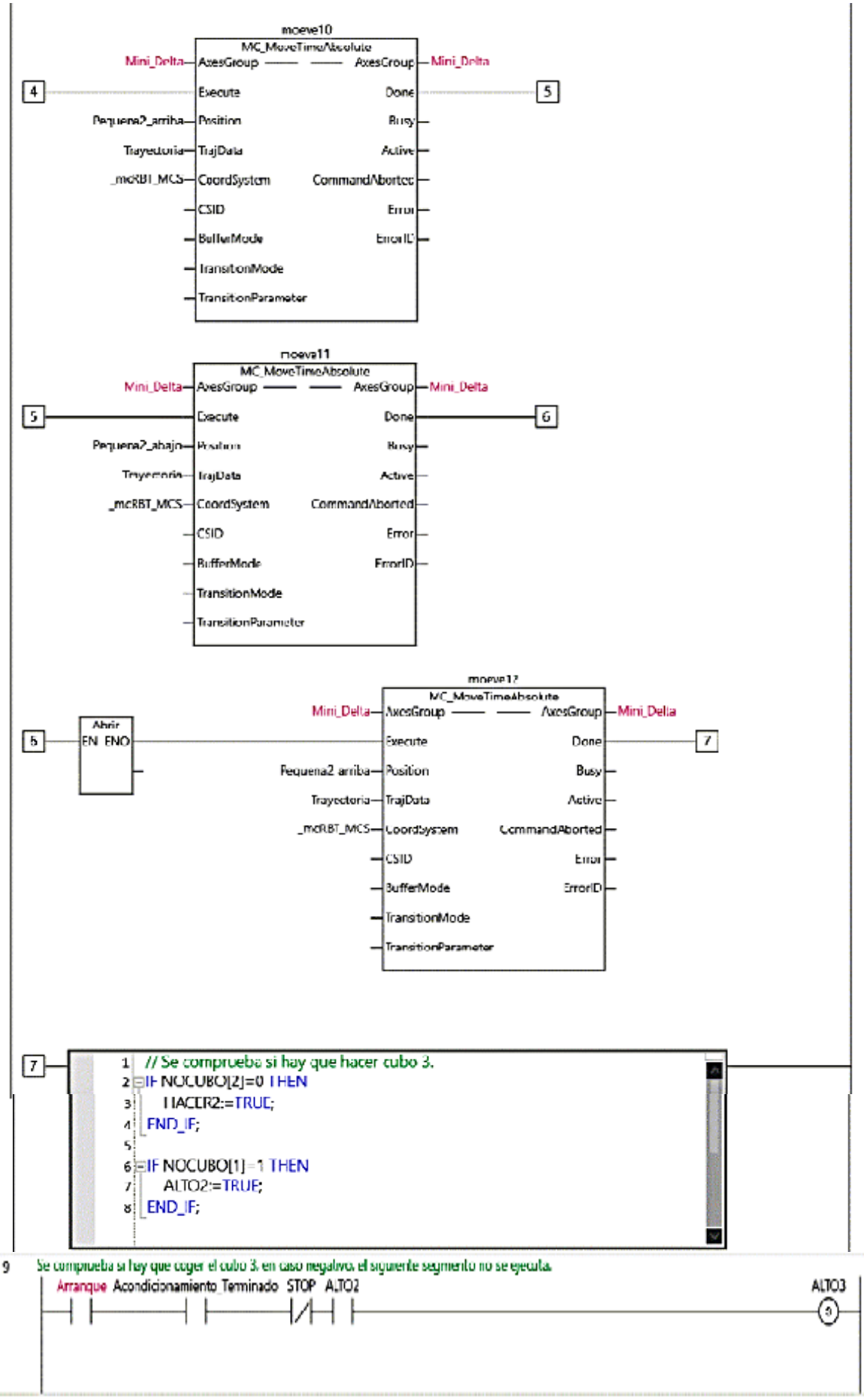

Moeve2
AxesGroup
AxesGroup
Mini_Delta
2
Execute
Done
3
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
3
Cerrar
EN ENO
4
Moeve3
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
4
Execute
Done
5
Grande1_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
moeve4
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Pequena1_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve5
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Pequena1_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter



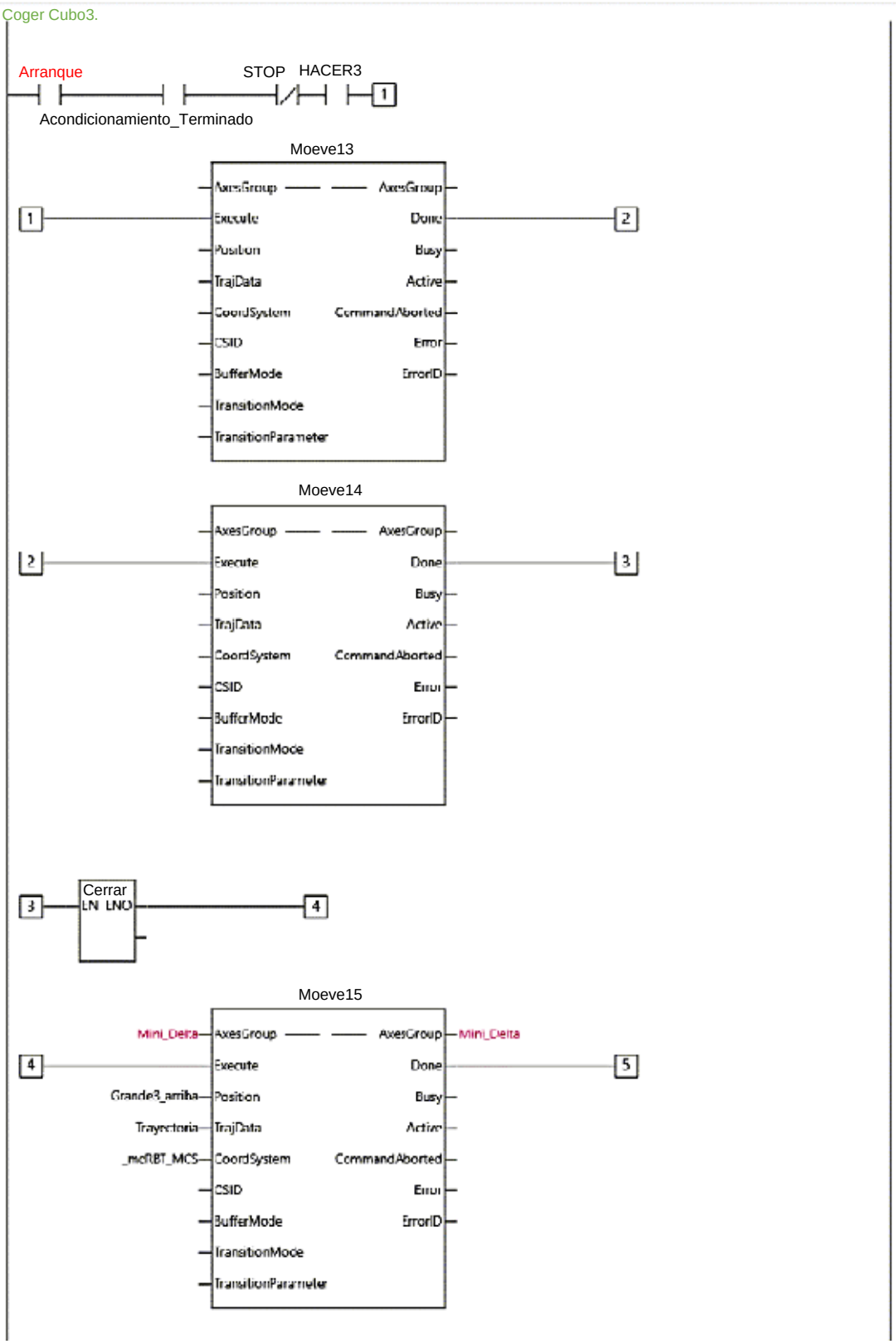
8 Coger Cubo2.

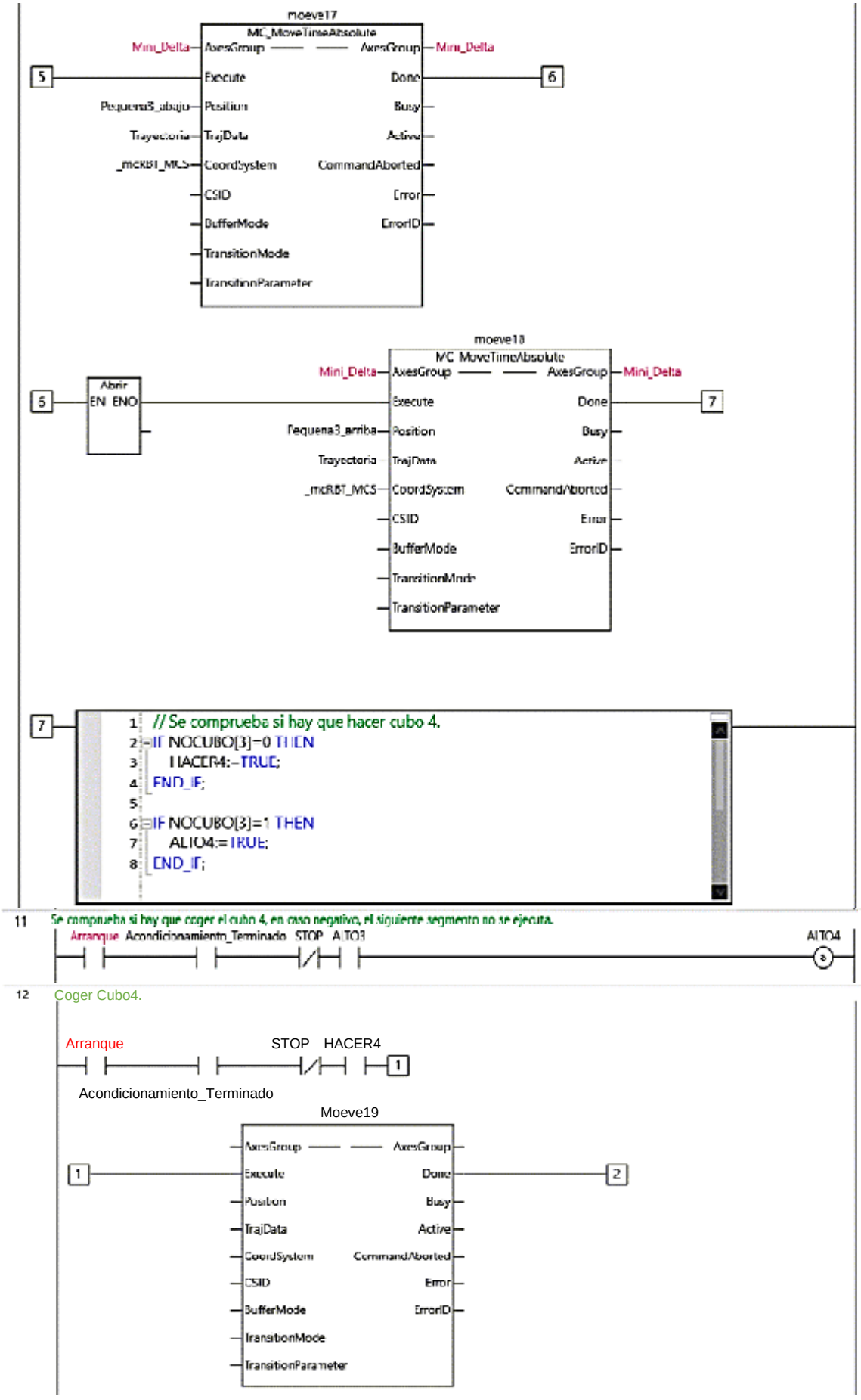


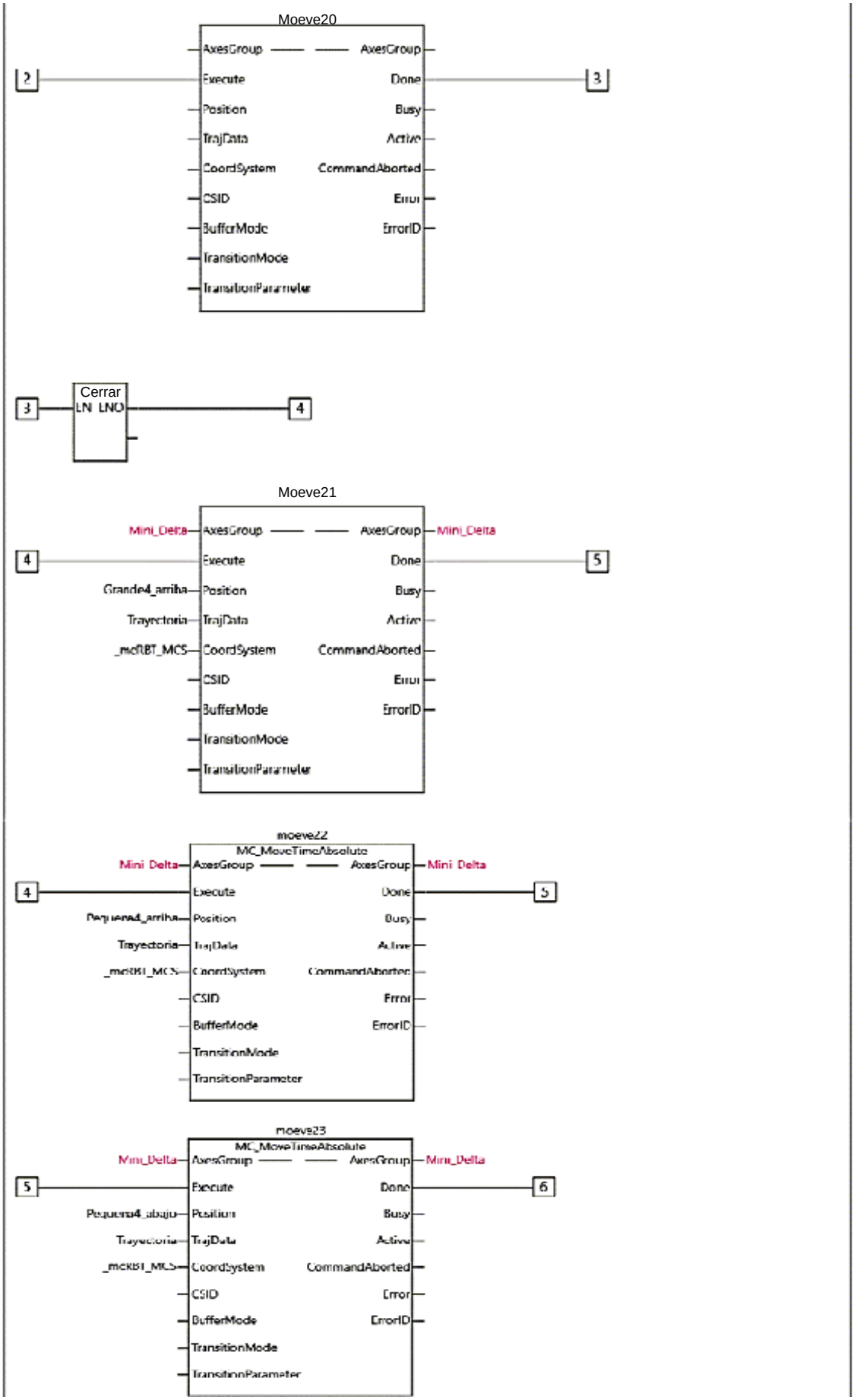

3
Cerrar
EN ENO
4
Moeve9
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
4
Execute
Done
5
Grande2_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve10
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
4
Execute
Done
5
Pequena2_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve11
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Pequena2_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve12
MC_MoveTimeAbsolute
Abrir
EN ENO
5
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
Execute
Done
7
Pequena2 arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter

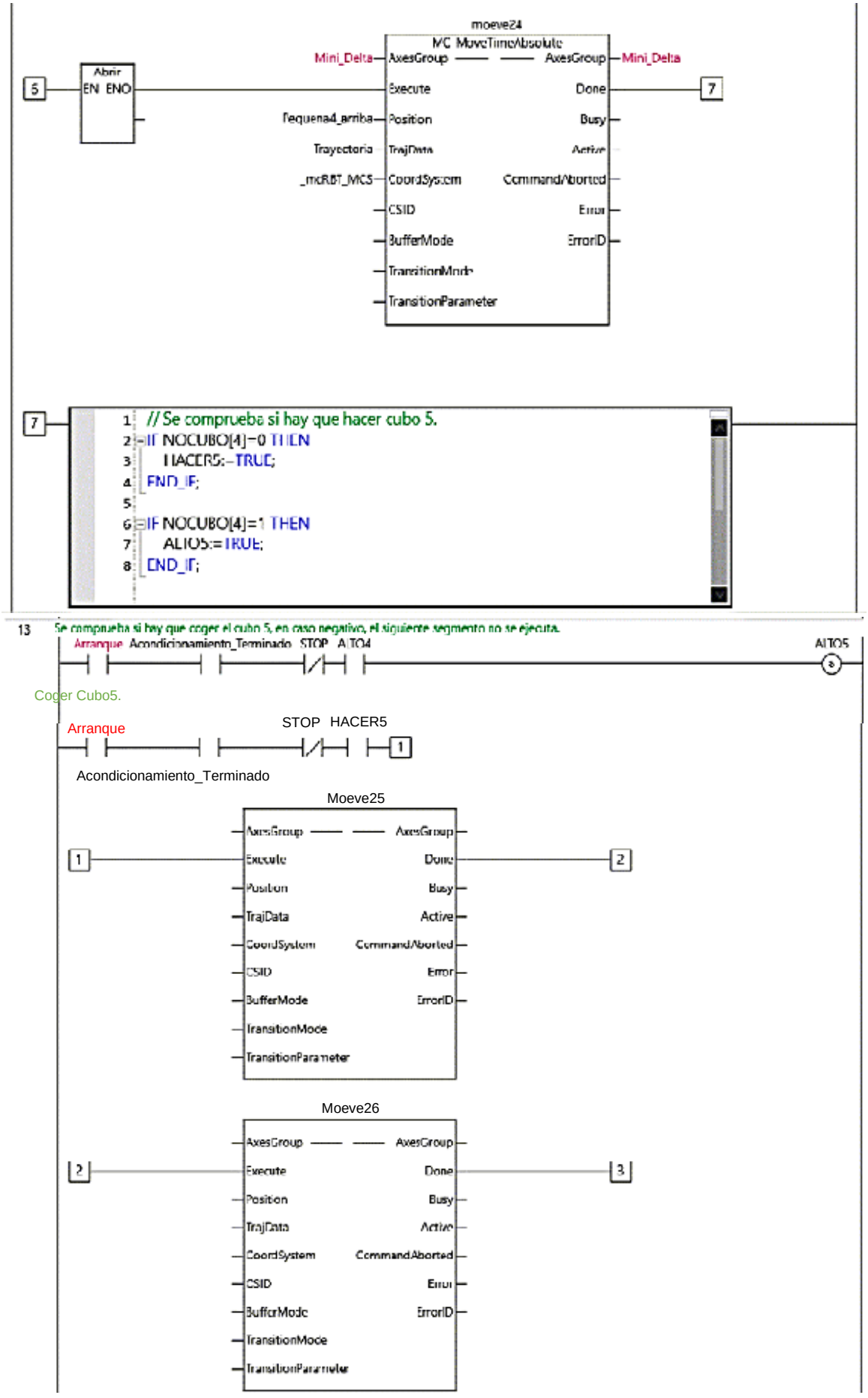

moeve10
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
Trayectoria
TrajData
_mcRBT_MCS
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
4
5
moeva11
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
TrajData
_mcRBT_MCS
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
5
6
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
Trayectoria
TrajData
_mcRBT_MCS
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Abrir
EN ENO
5
7
7
1 // Se comprueba si hay que hacer cubo 3.
2 IF NOCUBO[2]=0 THEN
3 HACER2:=TRUE;
4 END_IF;
5
6 IF NOCUBO[1]=1 THEN
7 ALTO2:=TRUE;
8 END_IF;
9
Se comprueba si hay que coger el cubo 3, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
ALTO2
ALTO3

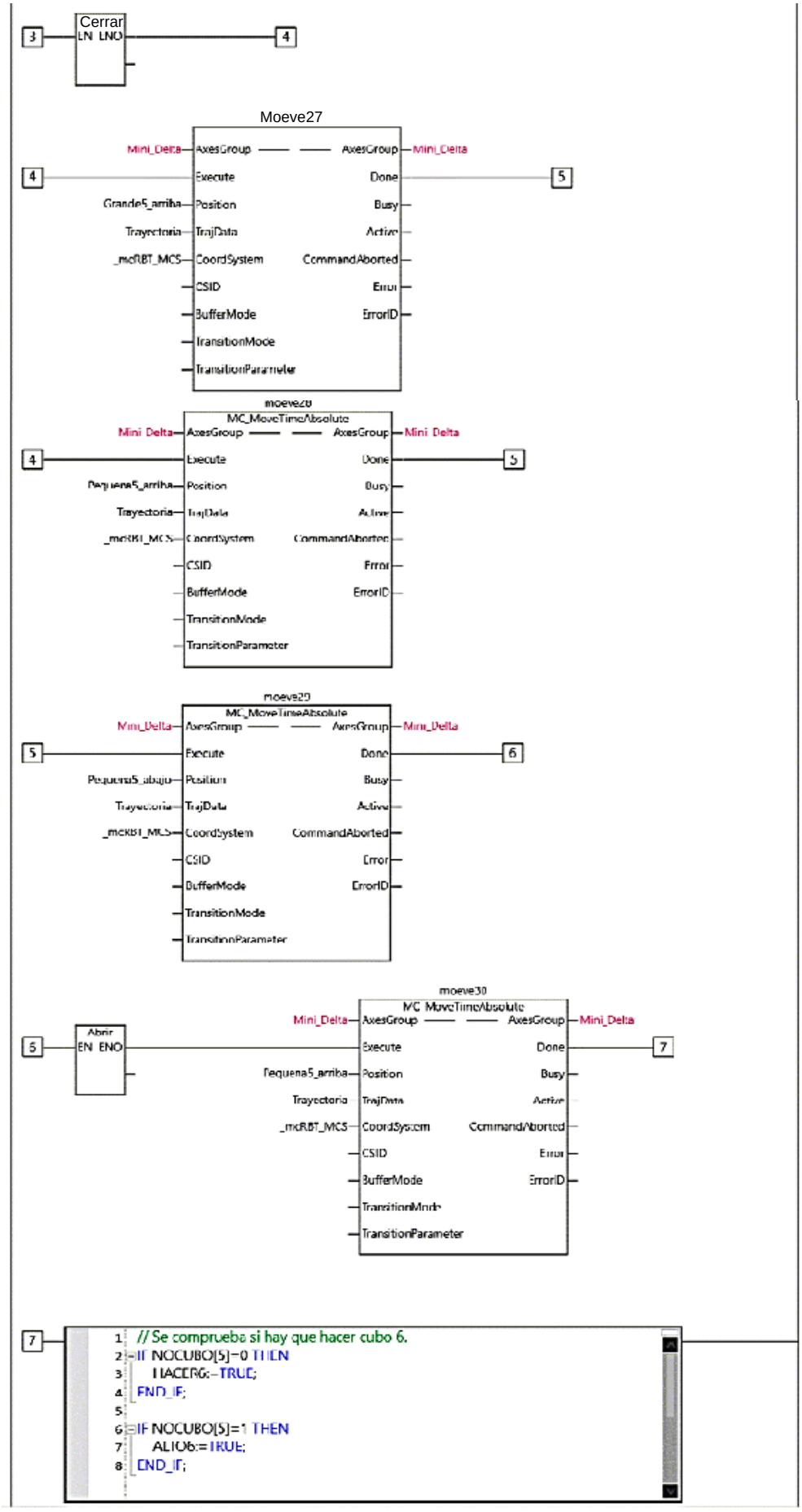
10 Coger Cubo3.

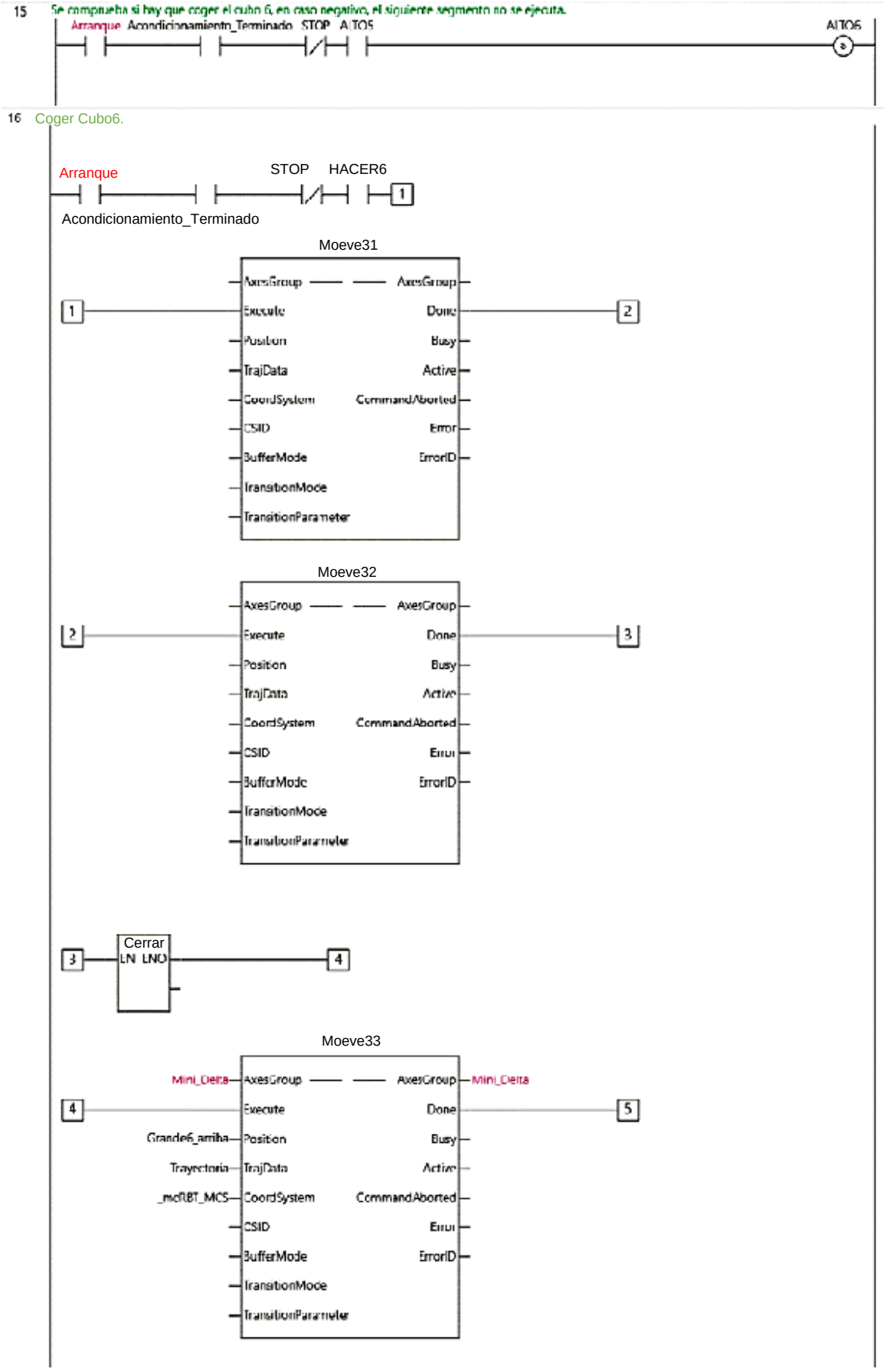


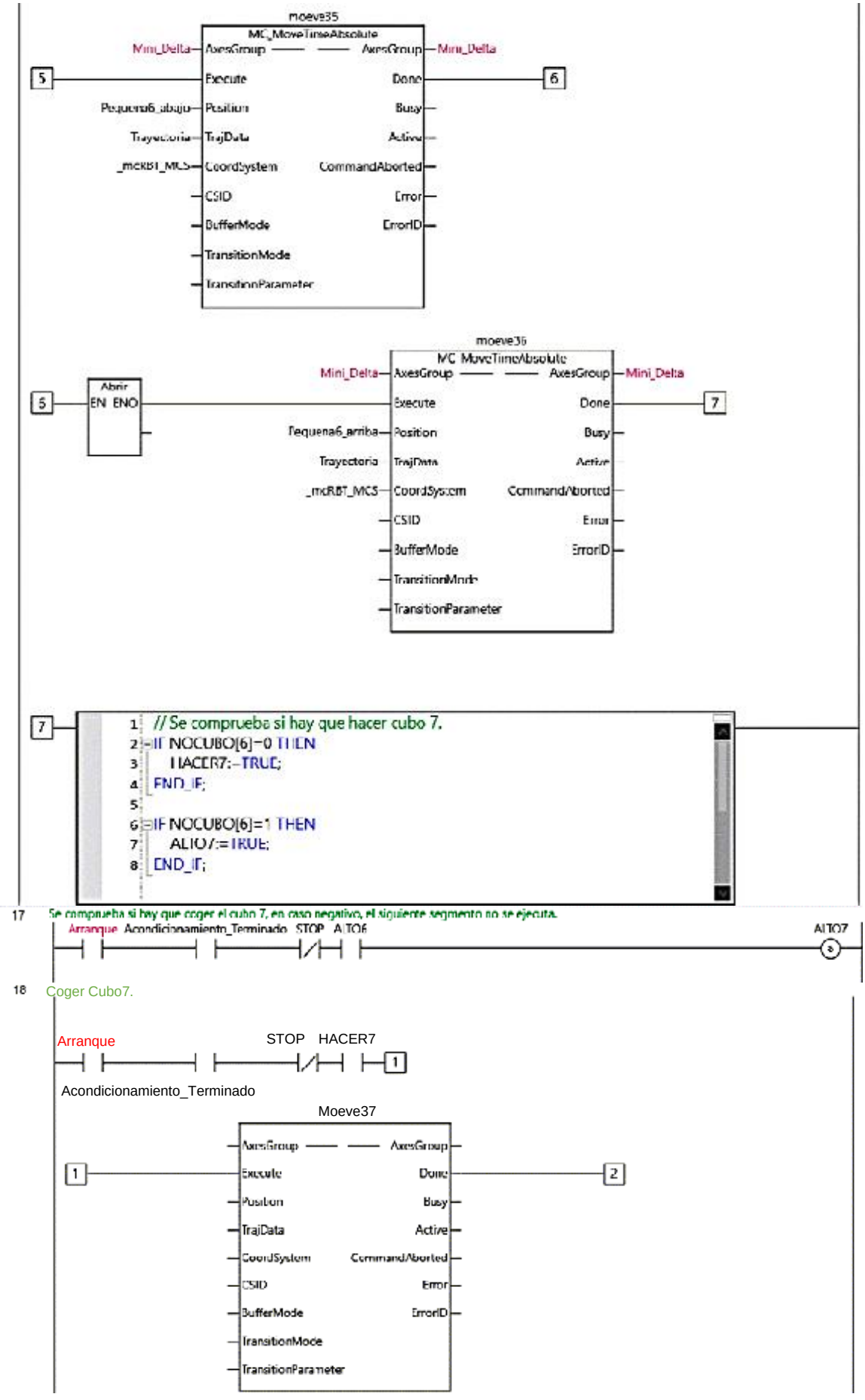

moeve17
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
5
6
Pequena3_abajo
Trayectoria
_mcRBT_MCS
moeve18
MC_MoveTimeAbsolute
Abrir
EN ENO
7
Pequena3_arriba
// Se comprueba si hay que hacer cubo 4.
IF NOCUBO[3]=0 THEN
HACER4:=TRUE;
END_IF;
IF NOCUBO[3]=1 THEN
ALTO4:=TRUE;
END_IF;
11
Se comprueba si hay que coger el cubo 4, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
ALTO3
ALTO4
12
Coger Cubo4.
Arranque
STOP
HACER4
Acondicionamiento_Terminado
Moeve19

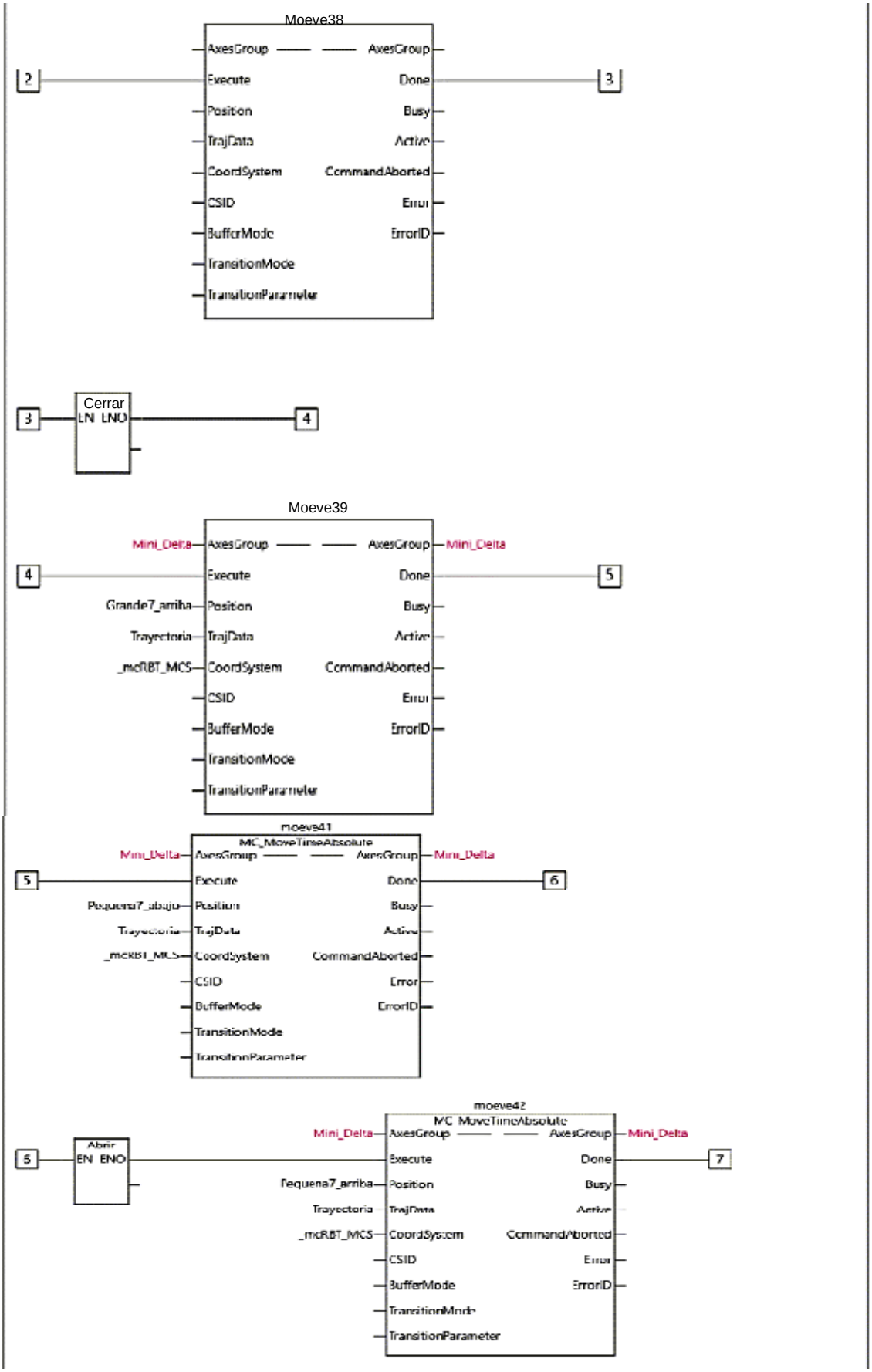

Moeve20
AxesGroup
AxesGroup
2
Execute
Done
3
Position
Busy
TrajData
Active
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Cerrar
LN LNO
3
4
Moeve21
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
4
Execute
Done
5
Grande4_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcKBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve22
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
4
Execute
Done
5
Position
Busy
Trayectoria
TrajData
Active
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve23
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Position
Busy
TrajData
Active
_mcKBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter

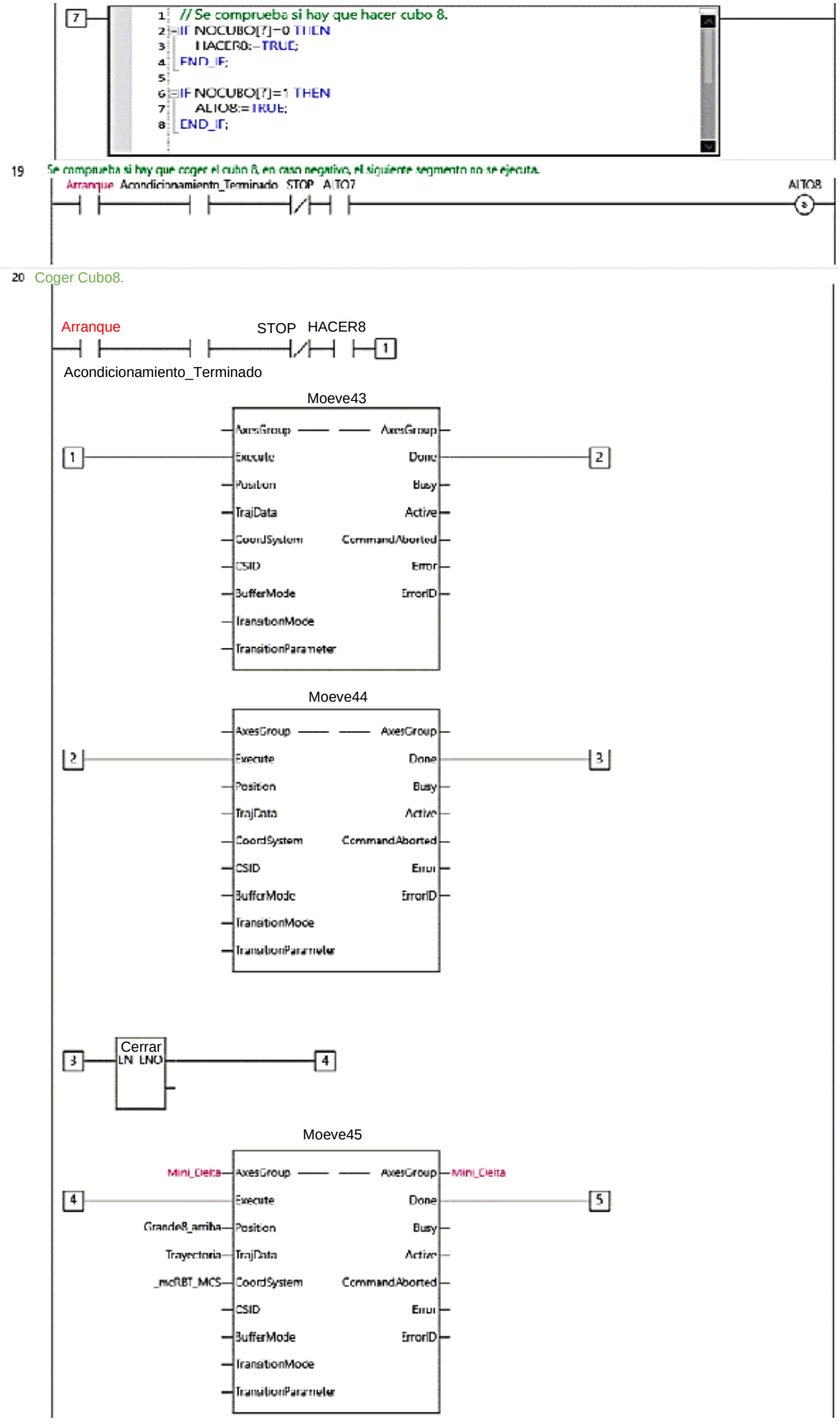

moeve24
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
Trayectoria
_mcRBT_MCS
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Abrir
EN ENO
5
7
// Se comprueba si hay que hacer cubo 5.
IF NOCUBO[4]=0 THEN
HACER5:=TRUE;
END_IF;
IF NOCUBO[4]=1 THEN
ALTO5:=TRUE;
END_IF;
13
Se comprueba si hay que coger el cubo 5, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
ALTO4
ALTO5
Coger Cubo5.
Arranque
STOP
HACER5
Acondicionamiento_Terminado
Moeve25
Moeve26

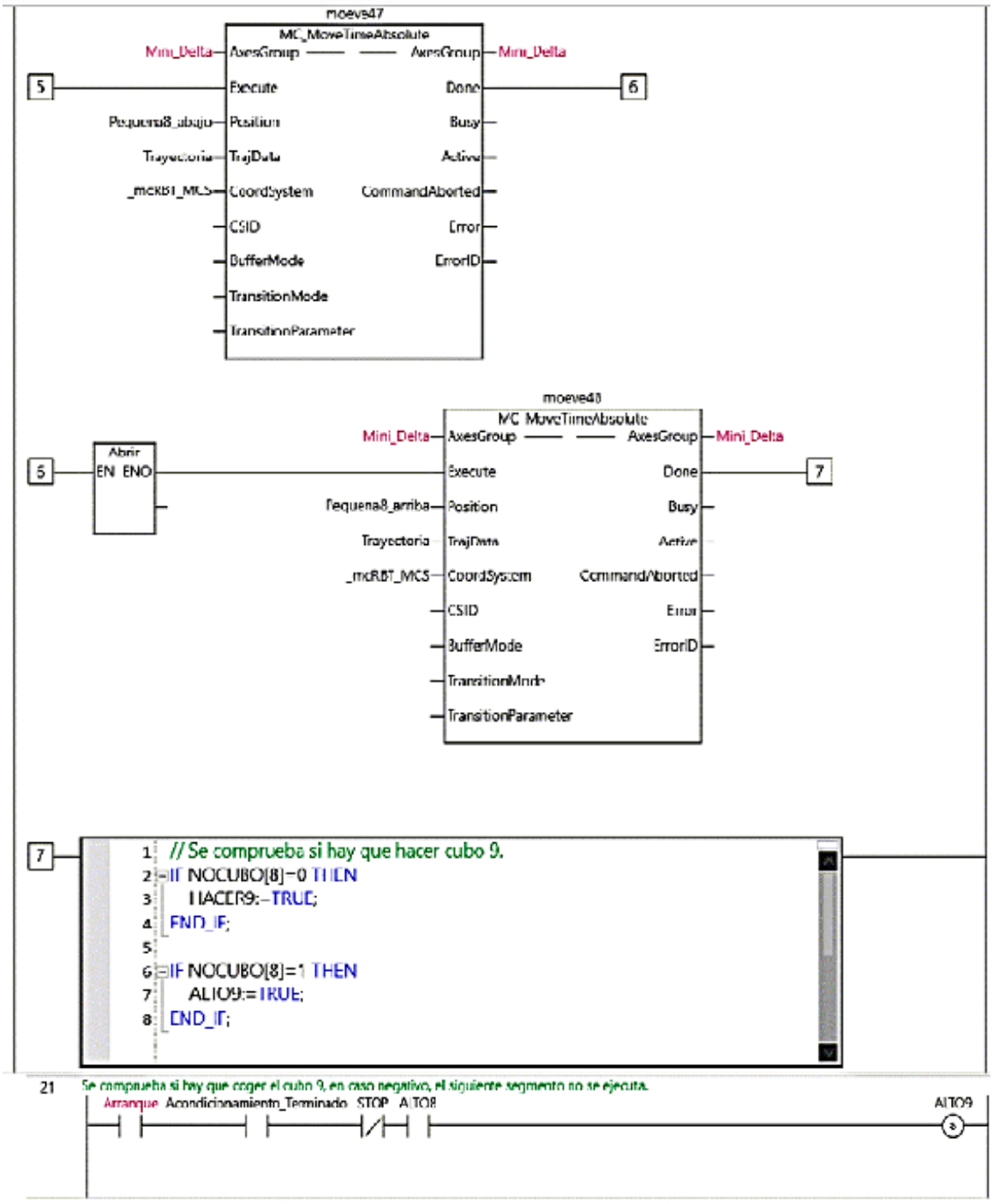

3
Cerrar
EN ENO
4
Moeve27
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
4
Execute
Done
5
Grande5_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve28
MC_MoveTimeAbsolute
Mini Delta
AxesGroup
AxesGroup
Mini Delta
4
Execute
Done
5
Pequena5_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve29
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Pequena5_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve30
MC_MoveTimeAbsolute
Abrir
EN ENO
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
7
Pequena5_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
7
1 // Se comprueba si hay que hacer cubo 6.
2 IF NOCUBO[5]=0 THEN
3 HACERG:=TRUE;
4 END_IF;
5
6 IF NOCUBO[5]=1 THEN
7 ALTO6:=TRUE;
8 END_IF;


15
Se comprueba si hay que coger el cubo 6, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
AITOS
AITO6
16
Coger Cubo6.
Arranque
STOP
HACER6
Acondicionamiento_Terminado
Moeve31
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Moeve32
Cerrar
LN
LNO
Moeve33
Mini_Delta
Grande6_arriba
Trayectoria
_mcfUBT_MCS

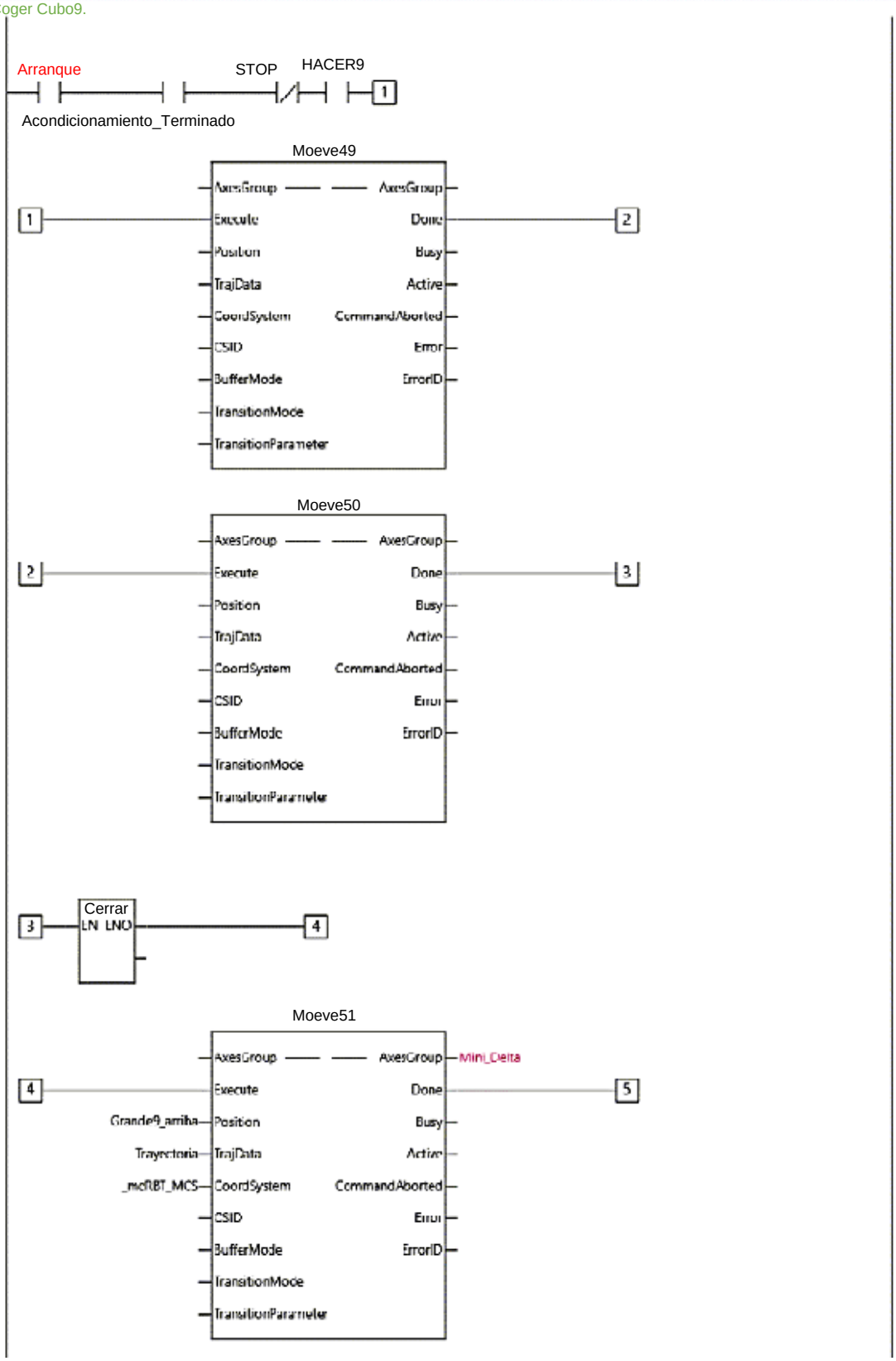

moeve35
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
5
6
moeve36
Abrir
EN ENO
7
Pequena6_arriba
Trayectoria
_mcRBT_MCS
1 // Se comprueba si hay que hacer cubo 7.
2 IF NOCUBO[6]=0 THEN
3 HACER7:=TRUE;
4 END_IF;
5
6 IF NOCUBO[6]=1 THEN
7 ALTO7:=TRUE;
8 END_IF;
17
Se comprueba si hay que coger el cubo 7, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
ALTO6
ALTO7
18
Coger Cubo7.
Arranque
STOP
HACER7
Acondicionamiento_Terminado
Moeve37
1
2

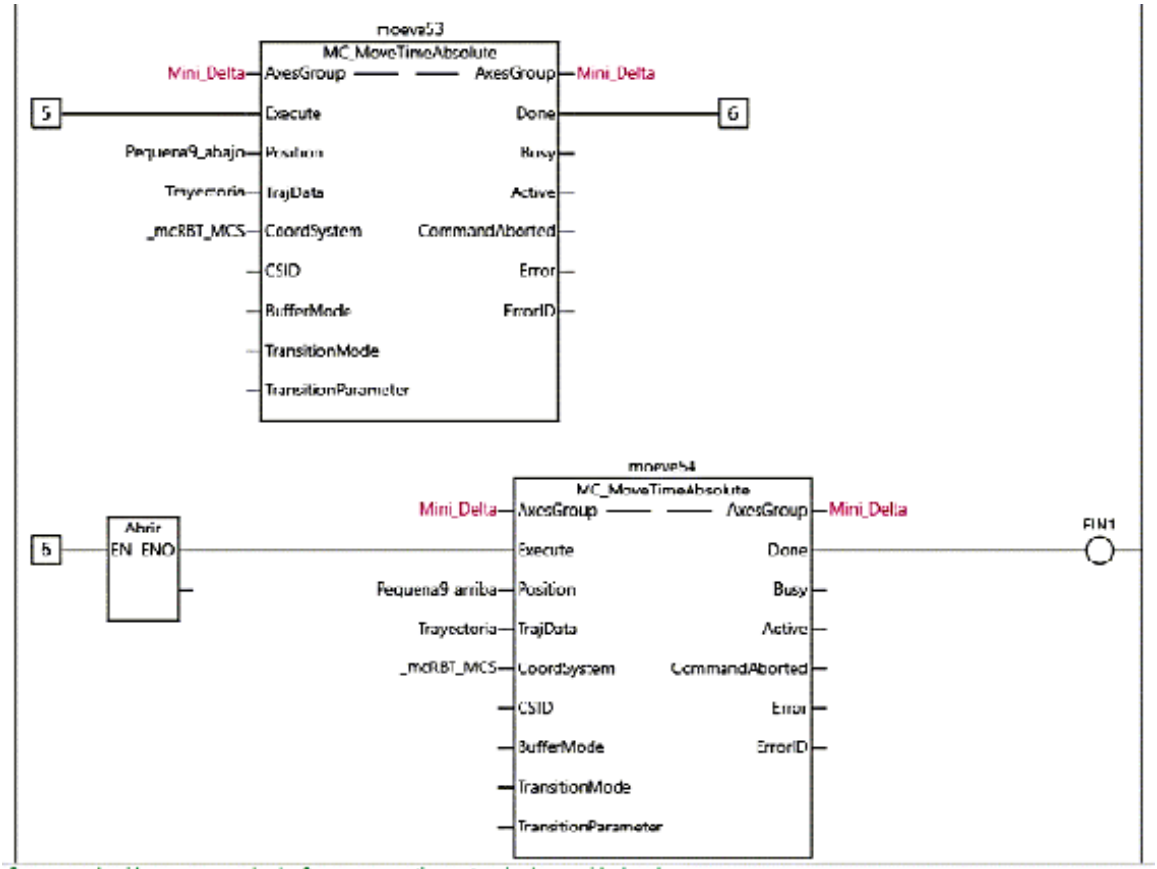

Moeve38
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
2
3
Cerrar
EN ENO
LN LNO
3
4
Moeve39
Mini_Delta
Grande7_arriba
Trayectoria
_mcRBT_MCS
5
moeve41
MC_MoveTimeAbsolute
Pequeña7_abajo
6
moeve42
Abrir
Pequeña7_arriba
7

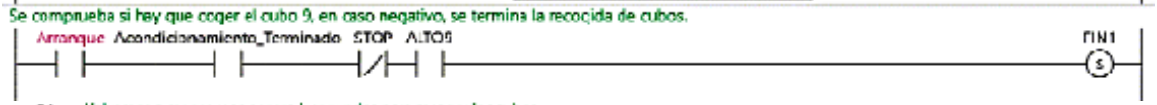

7
// Se comprueba si hay que hacer cubo 8.
IF NOCUBO[7]=0 THEN
HACER8:=TRUE;
END_IF;
IF NOCUBO[7]=1 THEN
ALTO8:=TRUE;
END_IF;
19
Se comprueba si hay que coger el cubo 8, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
ALTO7
ALTO8
20
Coger Cubo8.
Arranque
STOP
HACER8
1
Acondicionamiento_Terminado
Moeve43
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
1
2
Moeve44
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
2
3
3
Cerrar
LN LNO
4
Moeve45
Mini_Delta
AxesGroup
Execute
Grande8_arriba
Position
Trayectoria
TrajData
_mcRBT_MCS
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Mini_Delta
4
5

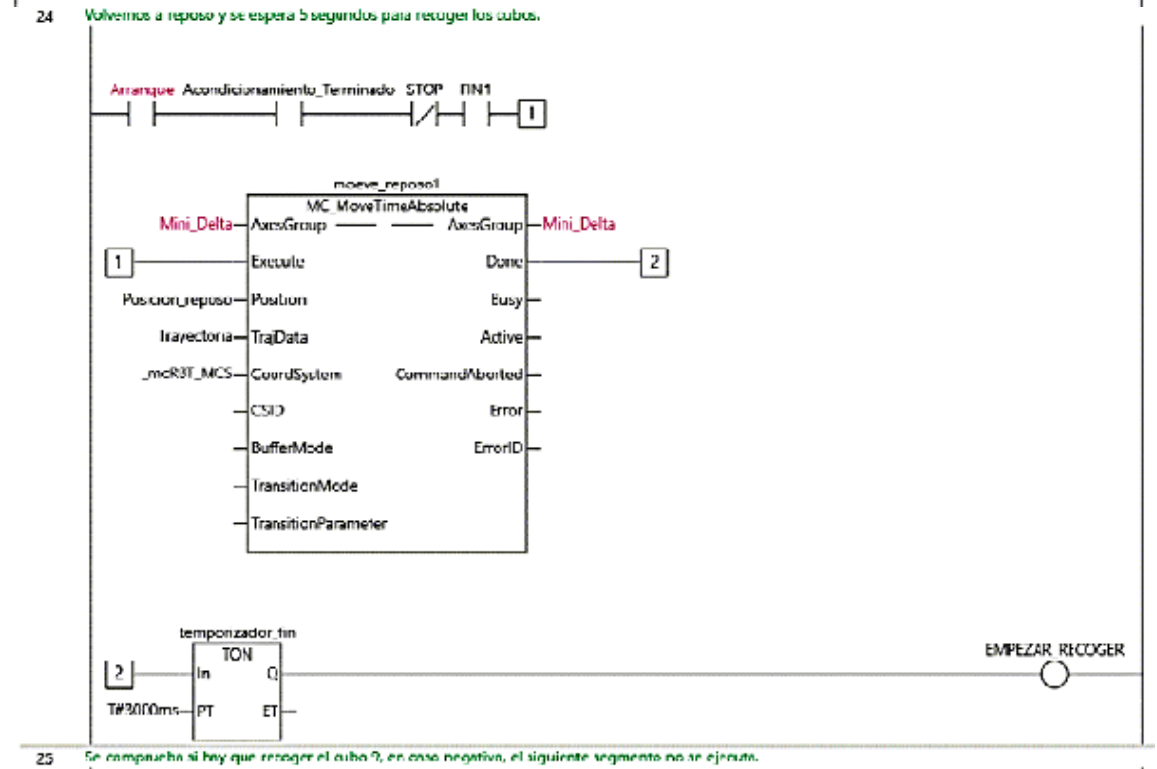

moevs47
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
5
6
Pequena8_abajo
Trayectoria
_mcRBT_MCS
moeve48
MC_MoveTimeAbsolute
Abrir
EN ENO
7
Pequena8_arriba
Trayectoria
_mcRBT_MCS
// Se comprueba si hay que hacer cubo 9.
IF NOCUBO[8]=0 THEN
HACER9:=TRUE;
END_IF;
IF NOCUBO[8]=1 THEN
ALTO9:=TRUE;
END_IF;
21
Se comprueba si hay que coger el cubo 9, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
ALTO8
ALTO9

22 Coger Cubo9.

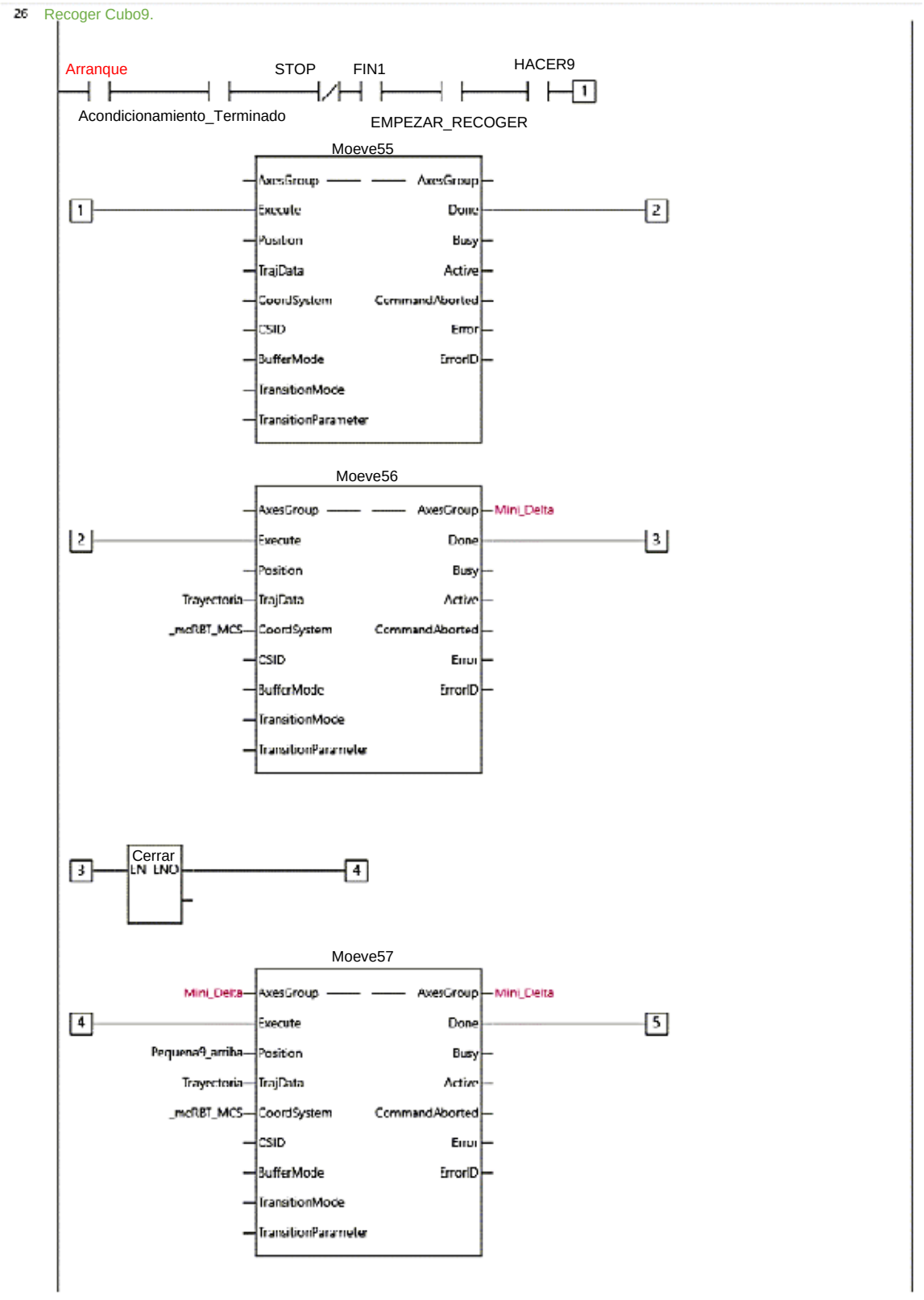


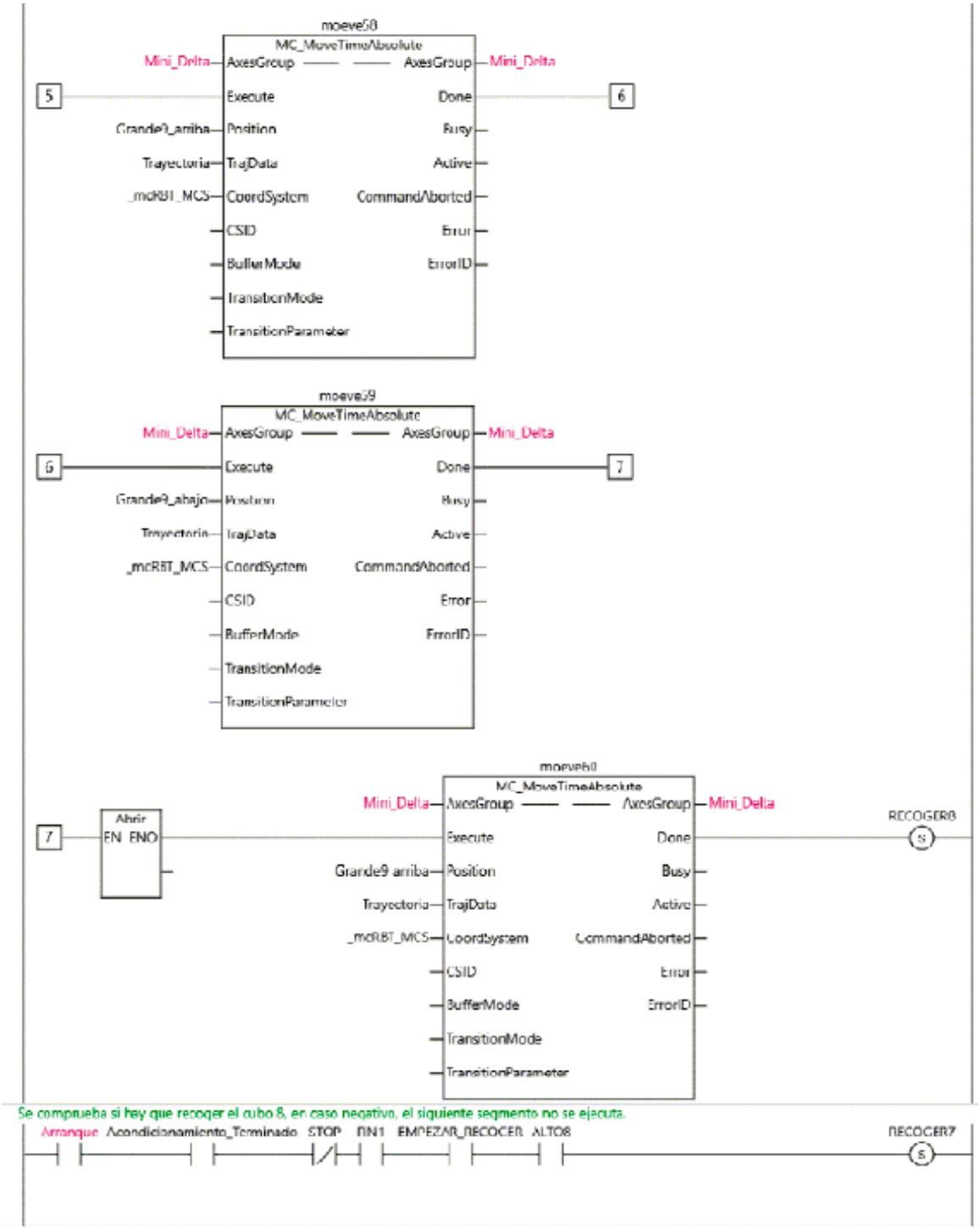

moeve53
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
5
6
Pequena9_abajo
Trayectoria
_mcRBT_MCS
moeve54
Abrir
EN ENO
b
Pequena9 arriba
FIN1


Se comprueba si hay que coger el cubo 9, en caso negativo, se termina la recogida de cubos.
Arranque Acondicionamiento_Terminado STOP ALTO9
FIN1
S

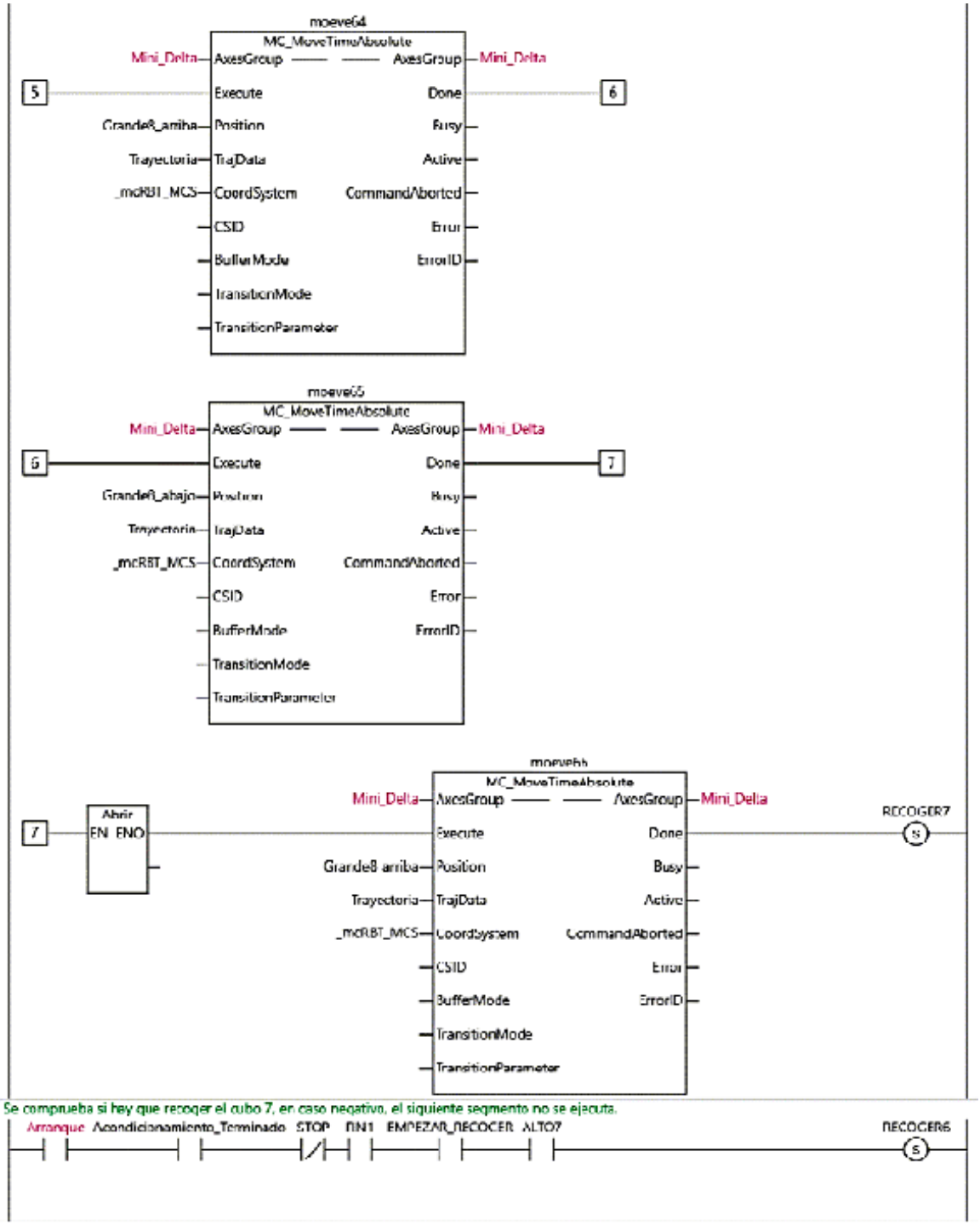

24
Volvemos a reposo y se espera 5 segundos para recoger los cubos.
Arranque Acondicionamiento_Terminado STOP FIN1
1
moeve_reposo1
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
2
Posicion_reposo
Trayectoria
_mcRBT_MCS
temporizador_fin
TON
In
PT
Q
ET
T#3000ms
EMPEZAR_RECOGER

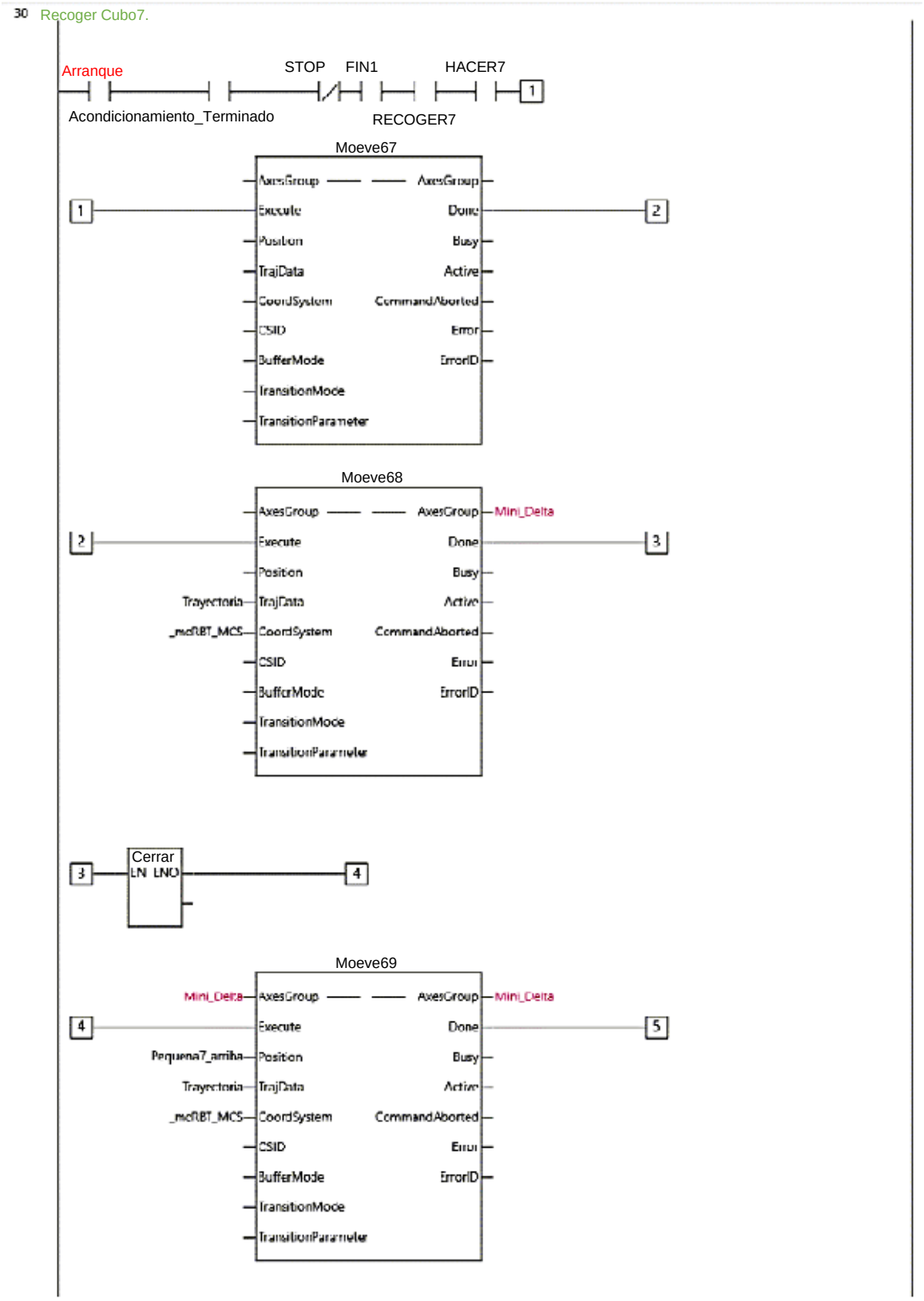

25
Se comprueba si hay que recoger el cubo 9, en caso negativo, el siguiente segmento no se ejecuta.
Arranque Acondicionamiento_Terminado STOP FIN1 EMPEZAR_RECOGER ALTO9
RECOGER8
S

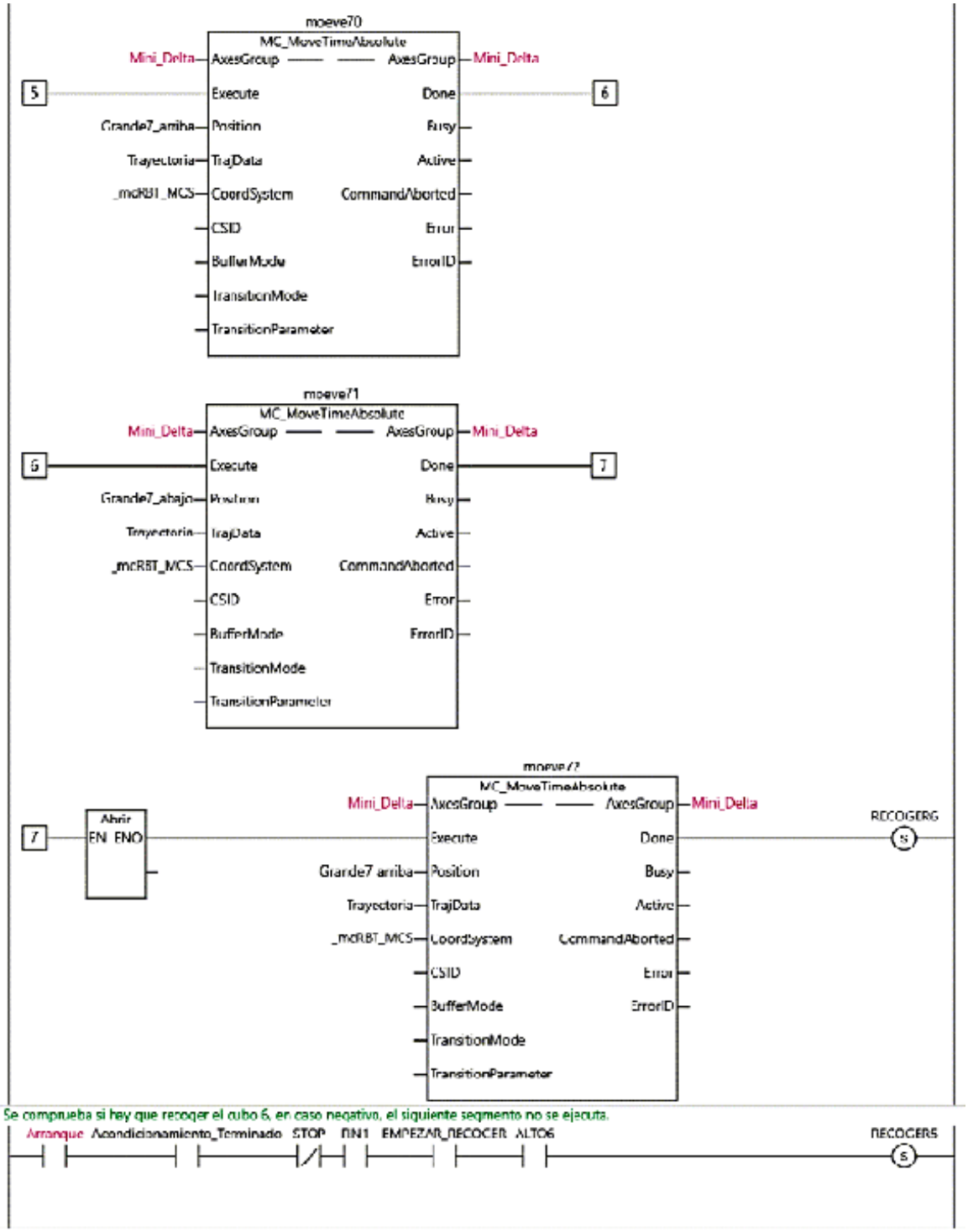

26 Recoger Cubo9.
Arranque
STOP
FIN1
HACER9
Acondicionamiento_Terminado
EMPEZAR_RECOGER
Moeve55
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Moeve56
Mini_Delta
Trayectoria
Cerrar
LN LNO
Moeve57
Pequena9_arriba


moeve68
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Grande9_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve69
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Grande9_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve60
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
7
Abrir
EN ENO
Execute
Done
RECOGER8
S
Grande9_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Se comprueba si hay que recoger el cubo 8, en caso negativo, el siguiente segmento no se ejecuta.
Arranque Acondicionamiento_Terminado STOP FIN1 EMPEZAR_RECOGER ALTO8
RECOGER7
S

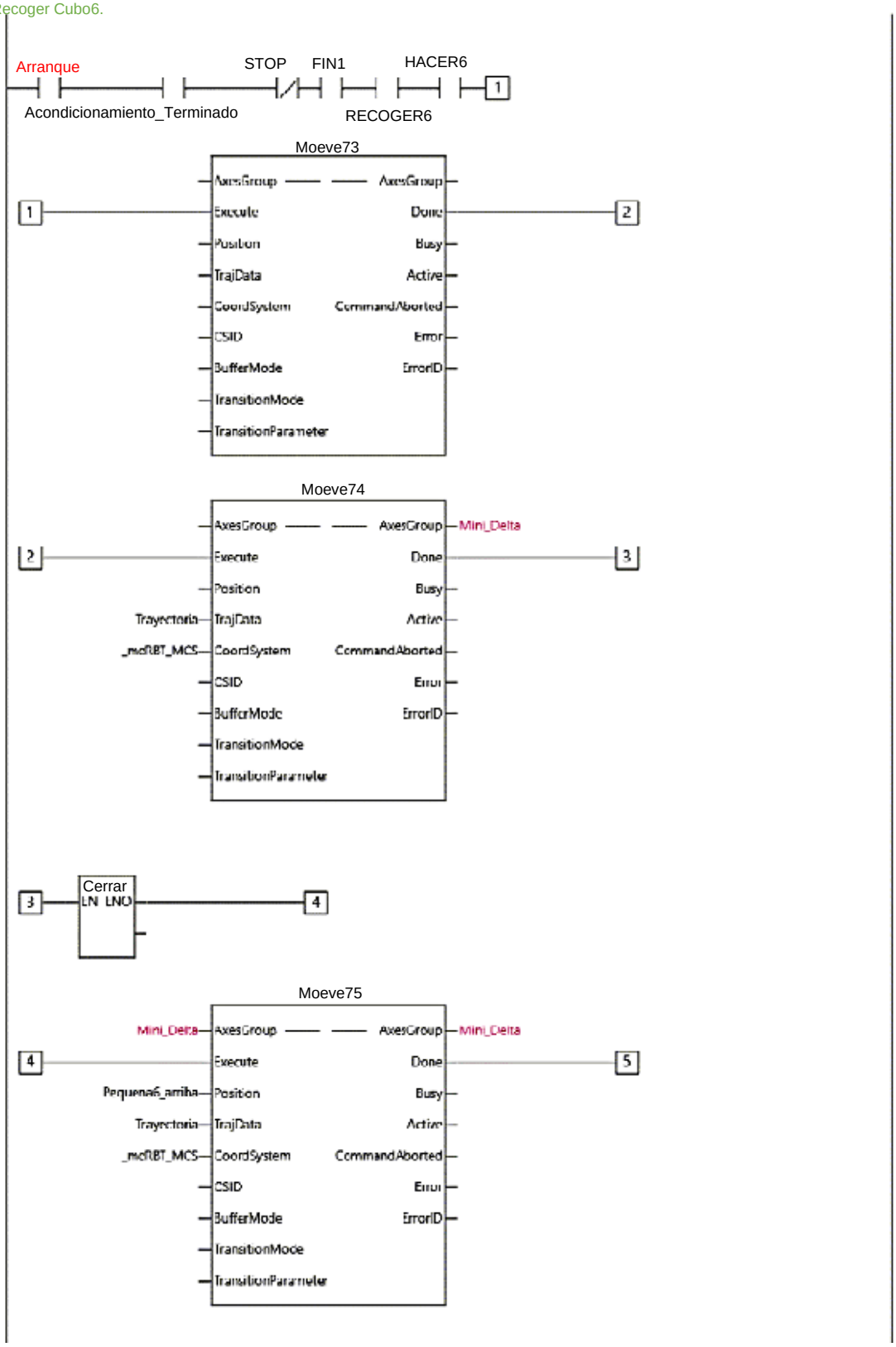
28 Recoger Cubo8.

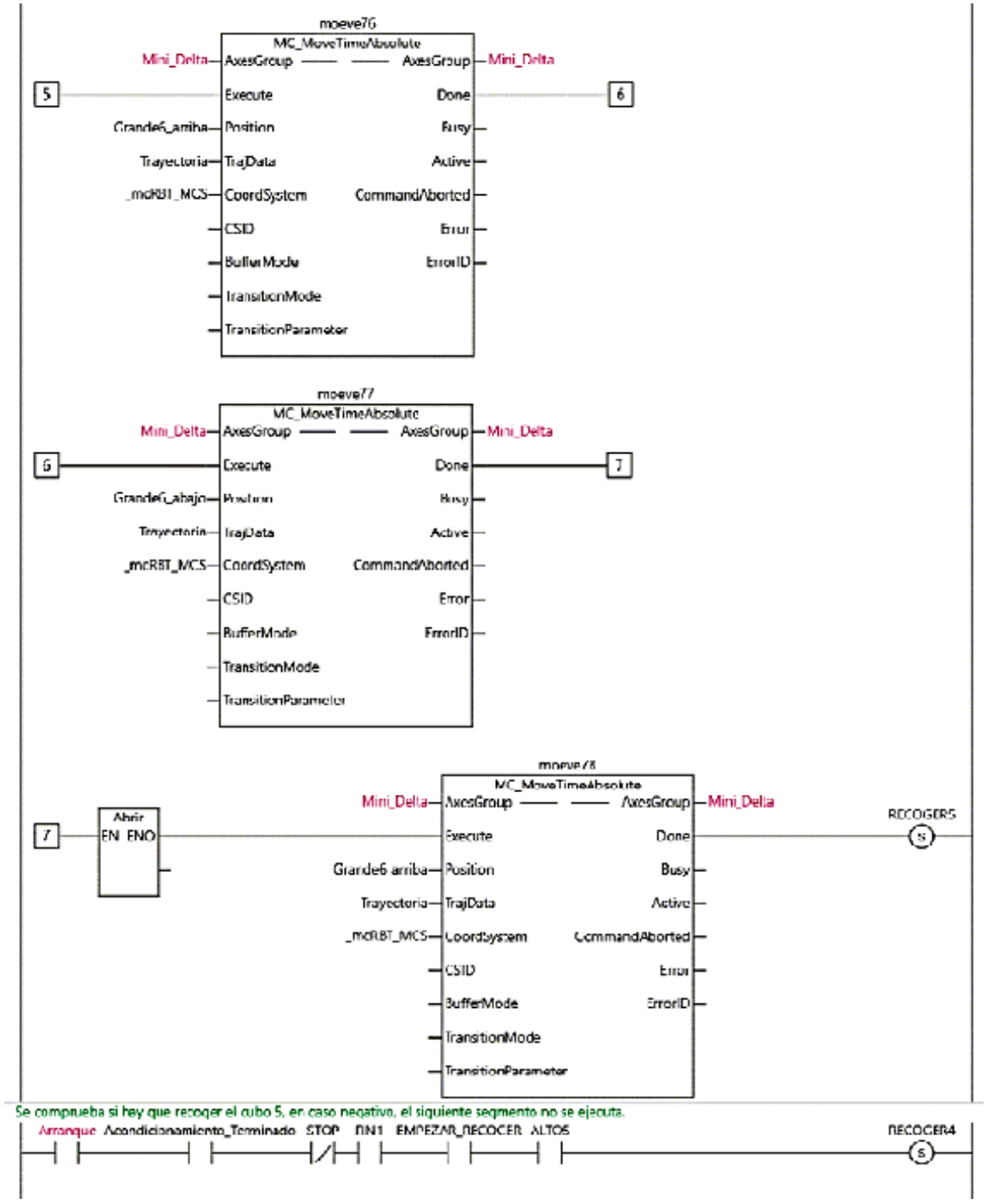

moeve64
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Grande8_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve65
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Grande8_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve66
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
7
Abrir
EN ENO
Execute
Done
RECOGER7
S
Grande8 arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Se comprueba si hay que recoger el cubo 7, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
FIN1
EMPEZAR_RECOGER
ALTO7
RECOGER6
S


30 Recoger Cubo7.
Arranque
STOP
FIN1
HACER7
Acondicionamiento_Terminado
RECOGER7
Moeve67
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Moeve68
Mini_Delta
Trayectoria
_mcfUBT_MCS
Cerrar
LN LNO
Moeve69
Pequena7_arriba

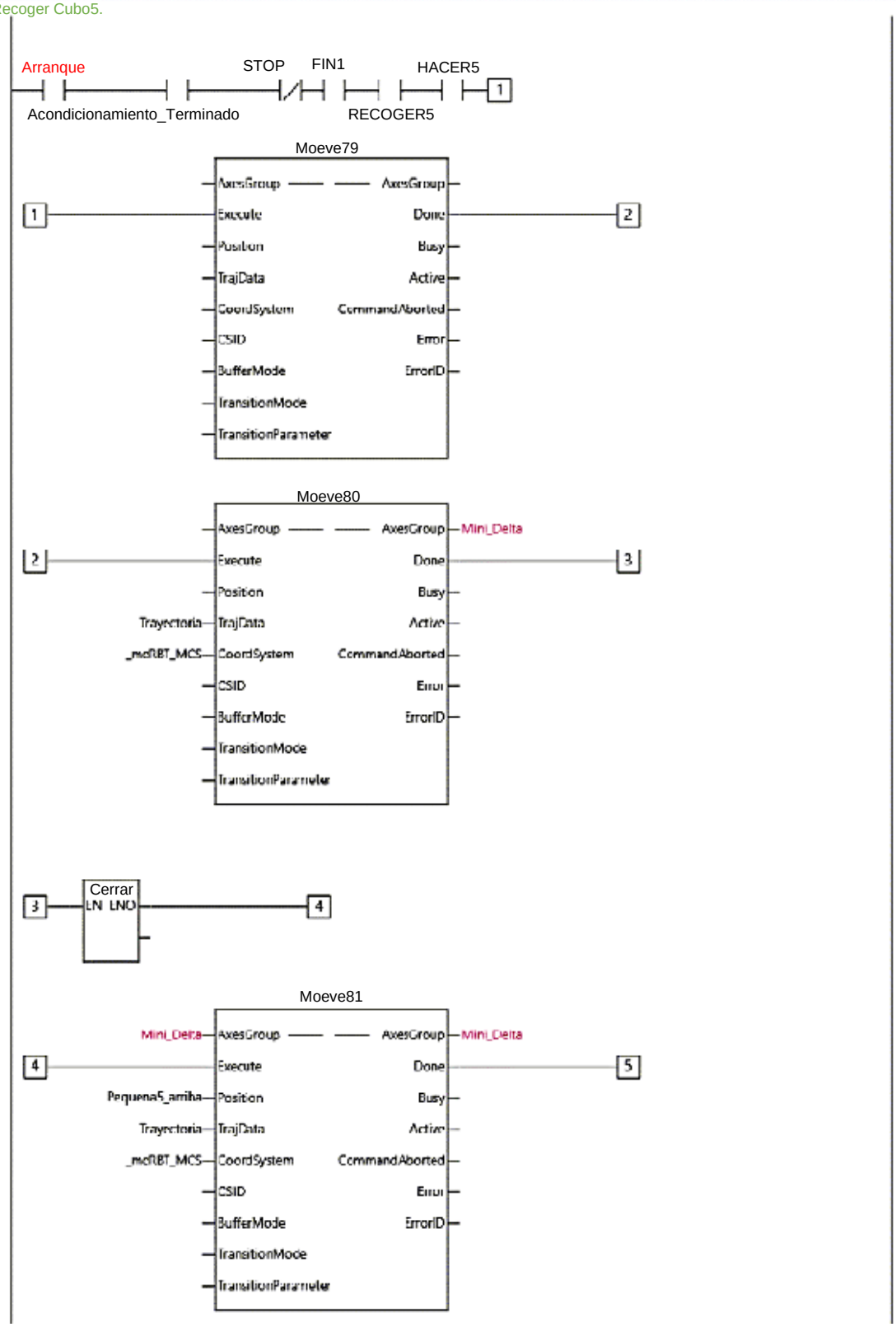

moeve70
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Grande7_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve71
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Grande7_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve72
MC_MoveTimeAbsolute
Abrir
7
EN ENO
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
Execute
Done
RECOGERG
Grande7 arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Se comprueba si hay que recoger el cubo 6, en caso negativo, el siguiente segmento no se ejecuta.
Arranque Acondicionamiento_Terminado STOP FIN1 EMPEZAR_RECOGER ALTO6
RECOGER5

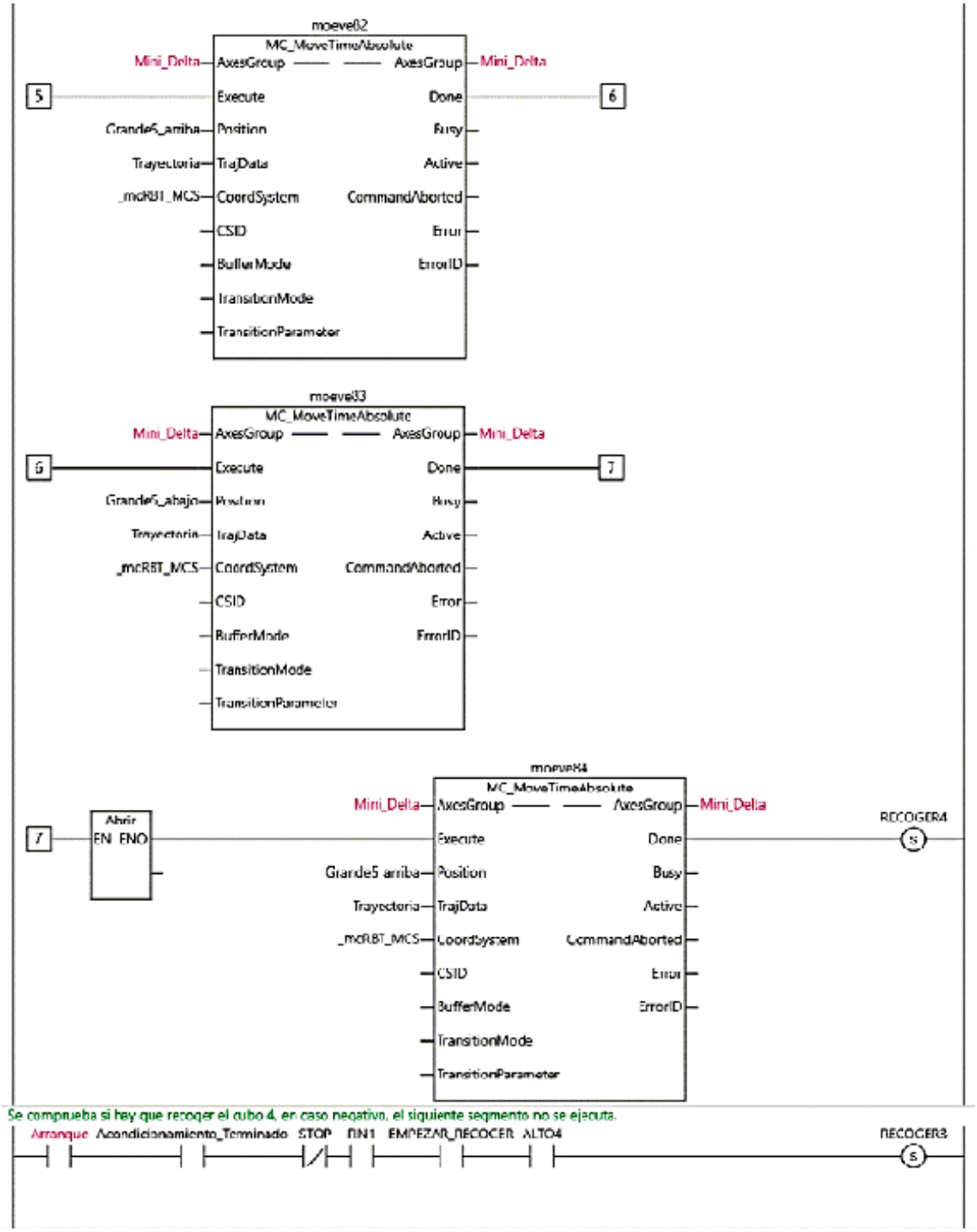
32 Recoger Cubo6.

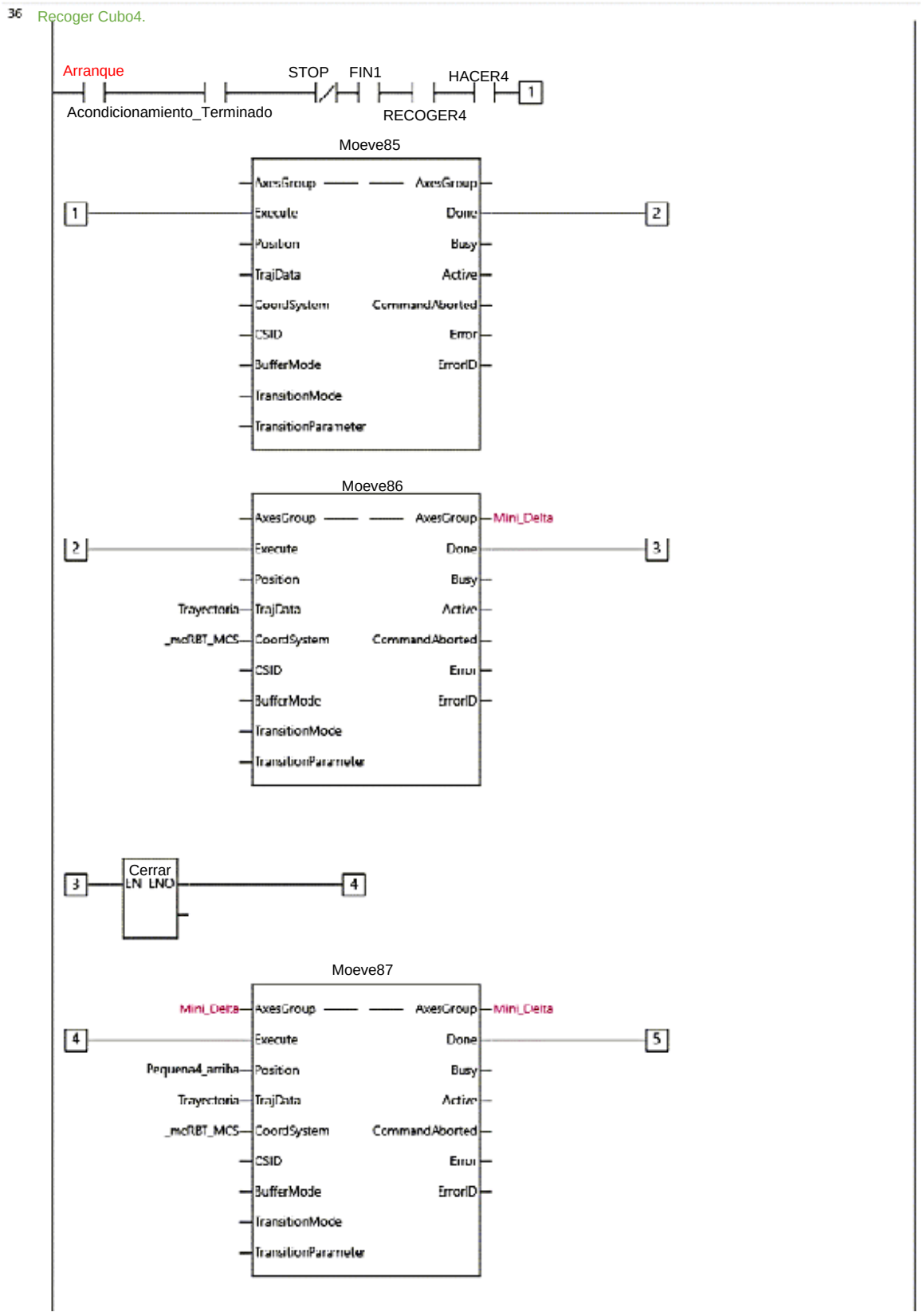


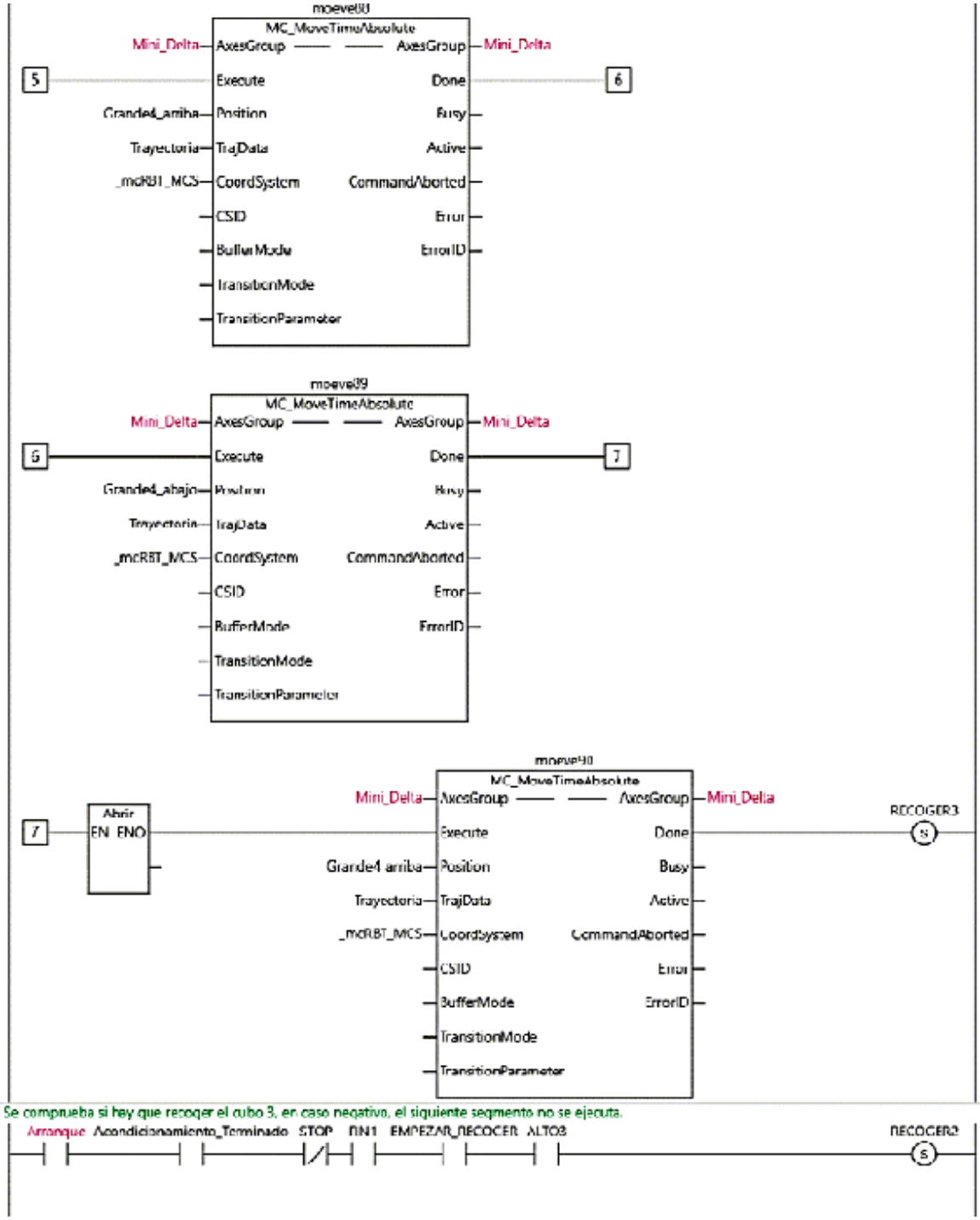

moeve76
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Grande6_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve77
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Grande6_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve78
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
7
Abrir
EN ENO
Execute
Done
RECOGER5
S
Grande6 arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Se comprueba si hay que recoger el cubo 5, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
BIN1
EMPEZAR_RECOGER
ALTOS
RECOGER4
S

34 Recoger Cubo5.

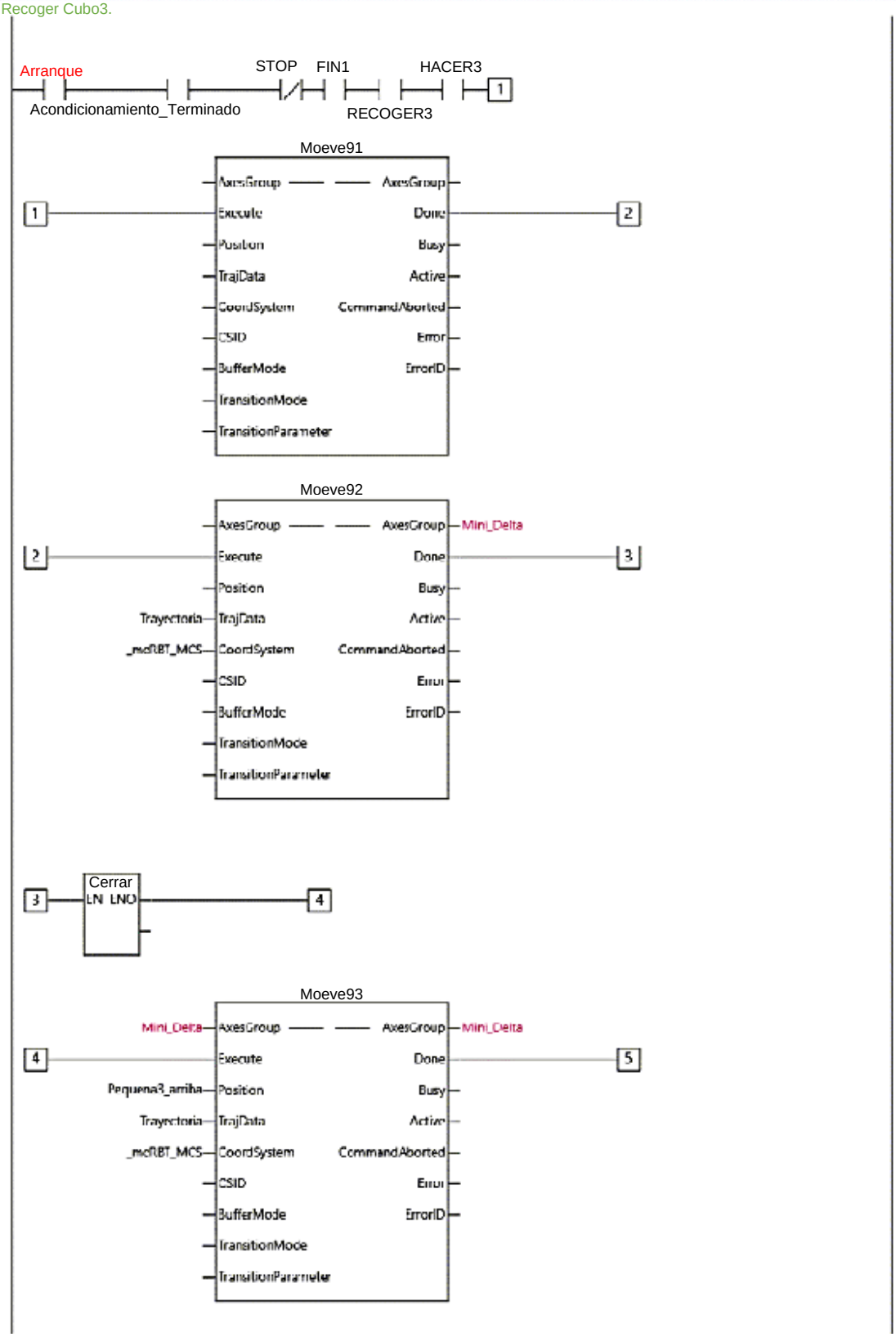


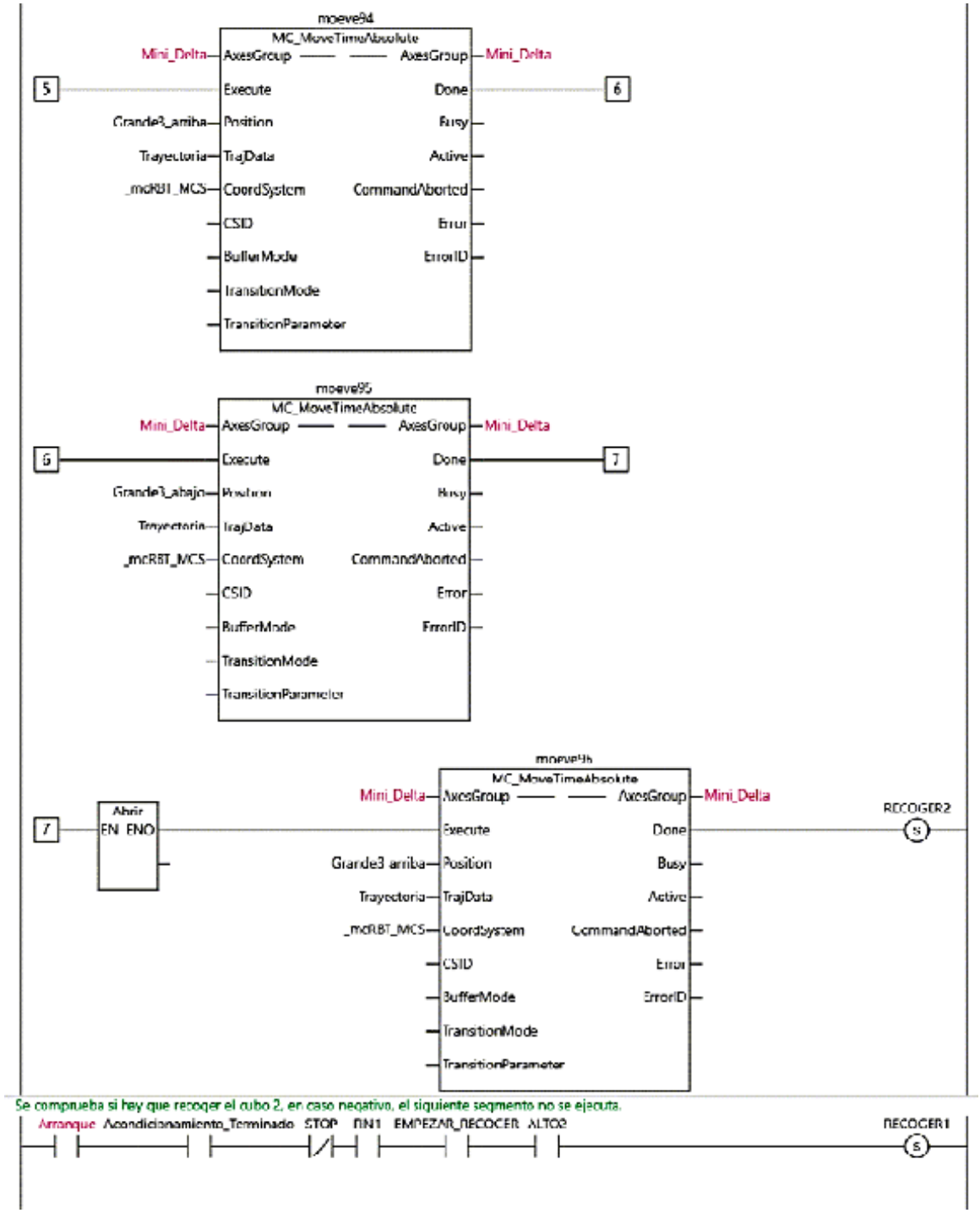

moeve02
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Grande5_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve03
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Grande5_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve04
MC_MoveTimeAbsolute
7
Abrir
EN ENO
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
Execute
Done
RECOGER4
S
Grande5 arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Se comprueba si hay que recoger el cubo 4, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
FIN1
EMPEZAR_RECOGER
ALTO4
RECOGER3
S

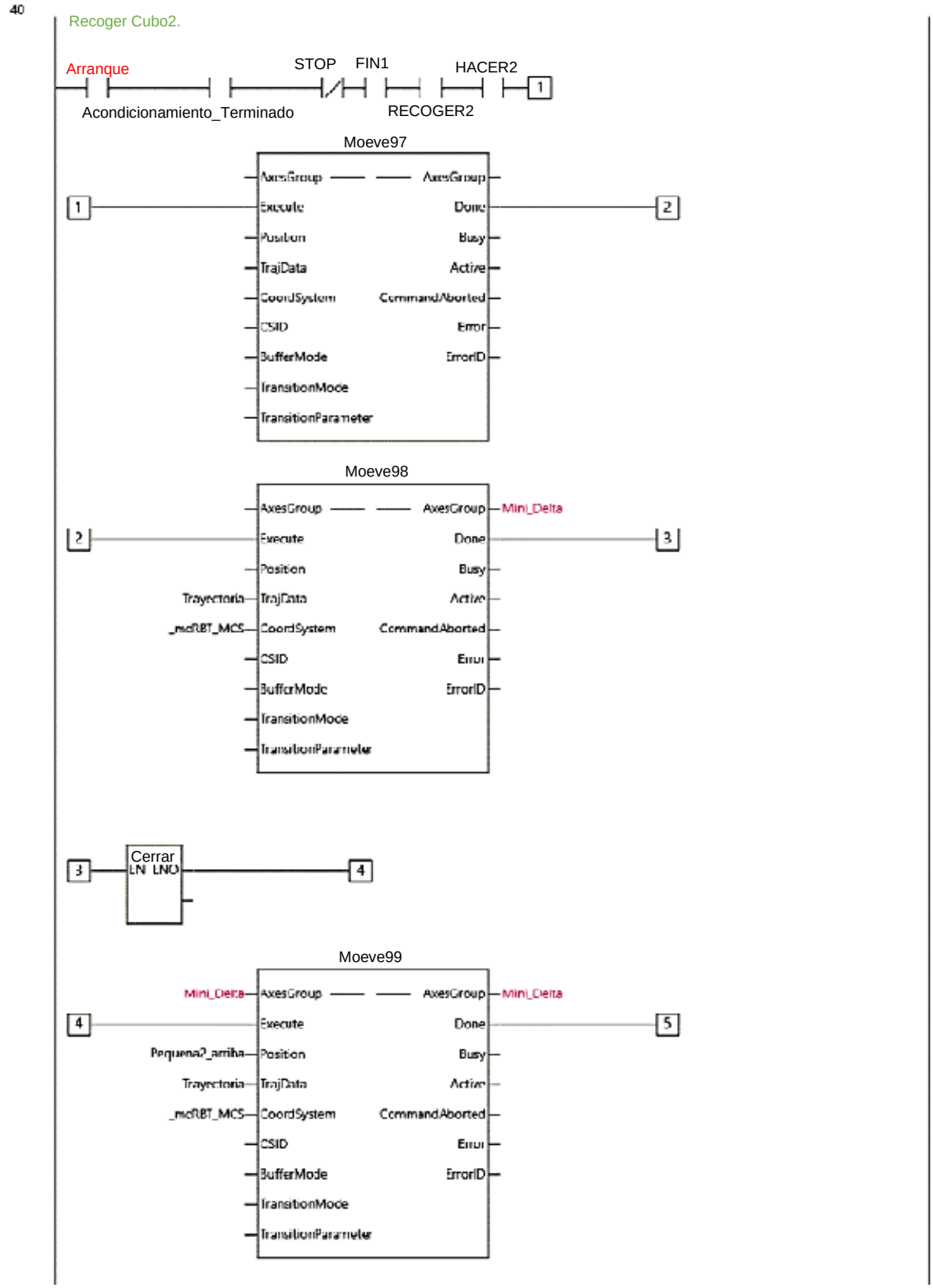

36
Recoger Cubo4.
Arranque
Acondicionamiento_Terminado
STOP
FIN1
HACER4
RECOGER4
1
Moeve85
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
2
Moeve86
Mini_Delta
Trayectoria
_mcfUBT_MCS
3
Cerrar
LN LNO
4
Moeve87
Pequena4_arriba
5

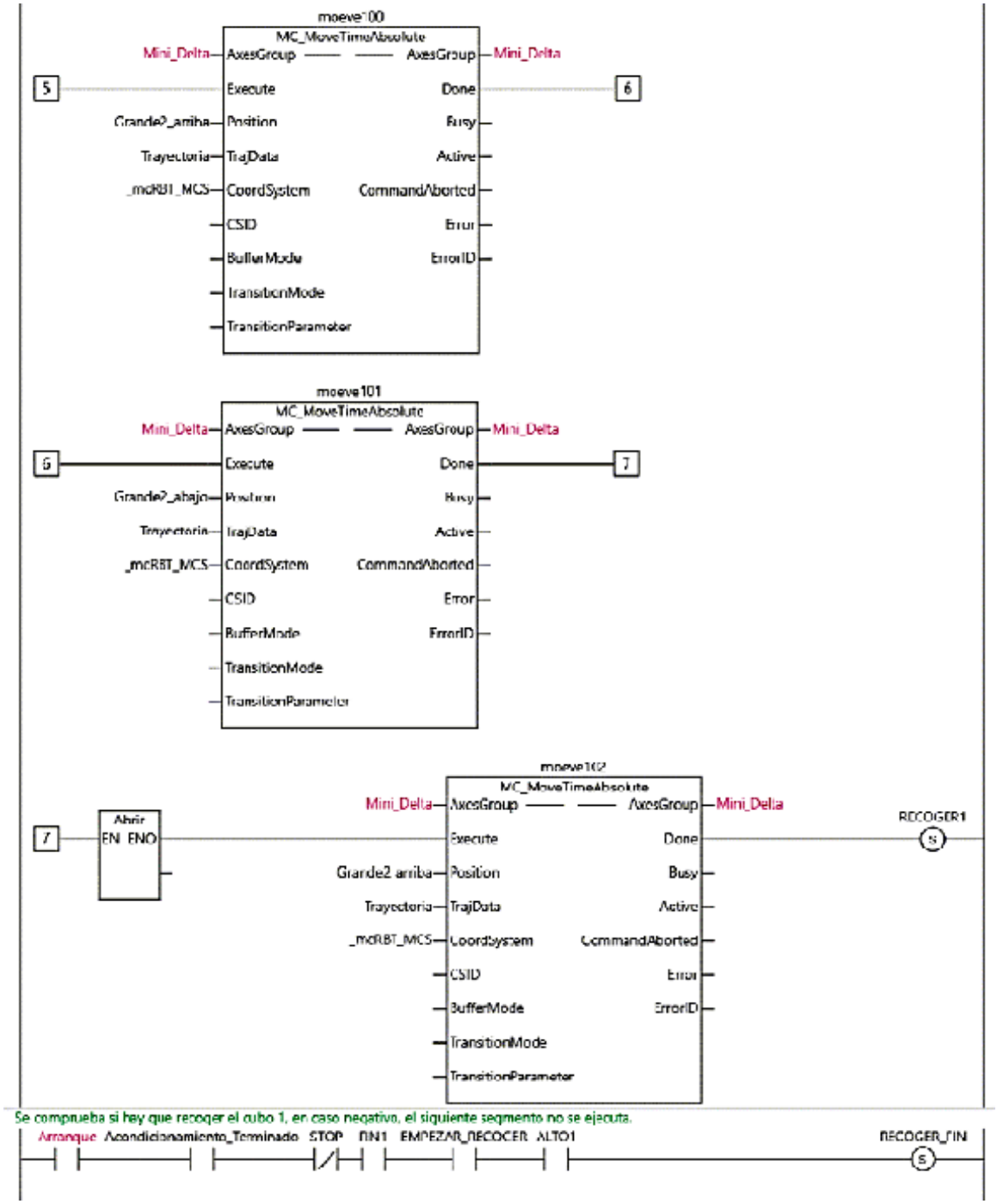

moevel00
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Grande4_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moevel09
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Grande4_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moevel10
MC_MoveTimeAbsolute
7
Abrir
EN ENO
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
Execute
Done
RECOGER3
S
Grande4 arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Se comprueba si hay que recoger el cubo 3, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
DIN1
EMPEZAR_RECOGER
ALTO3
RECOGER2
S

38 Recoger Cubo3.

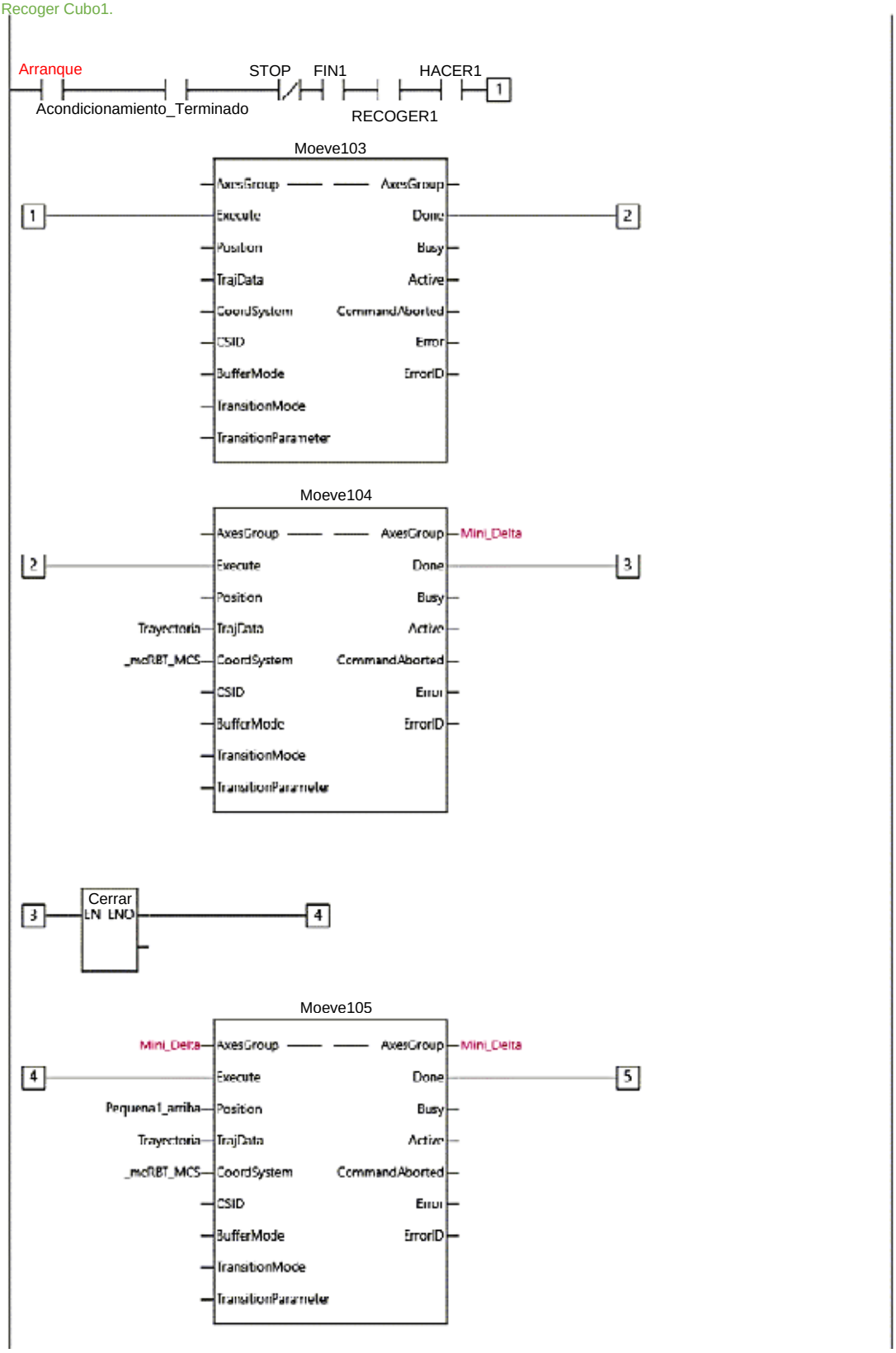


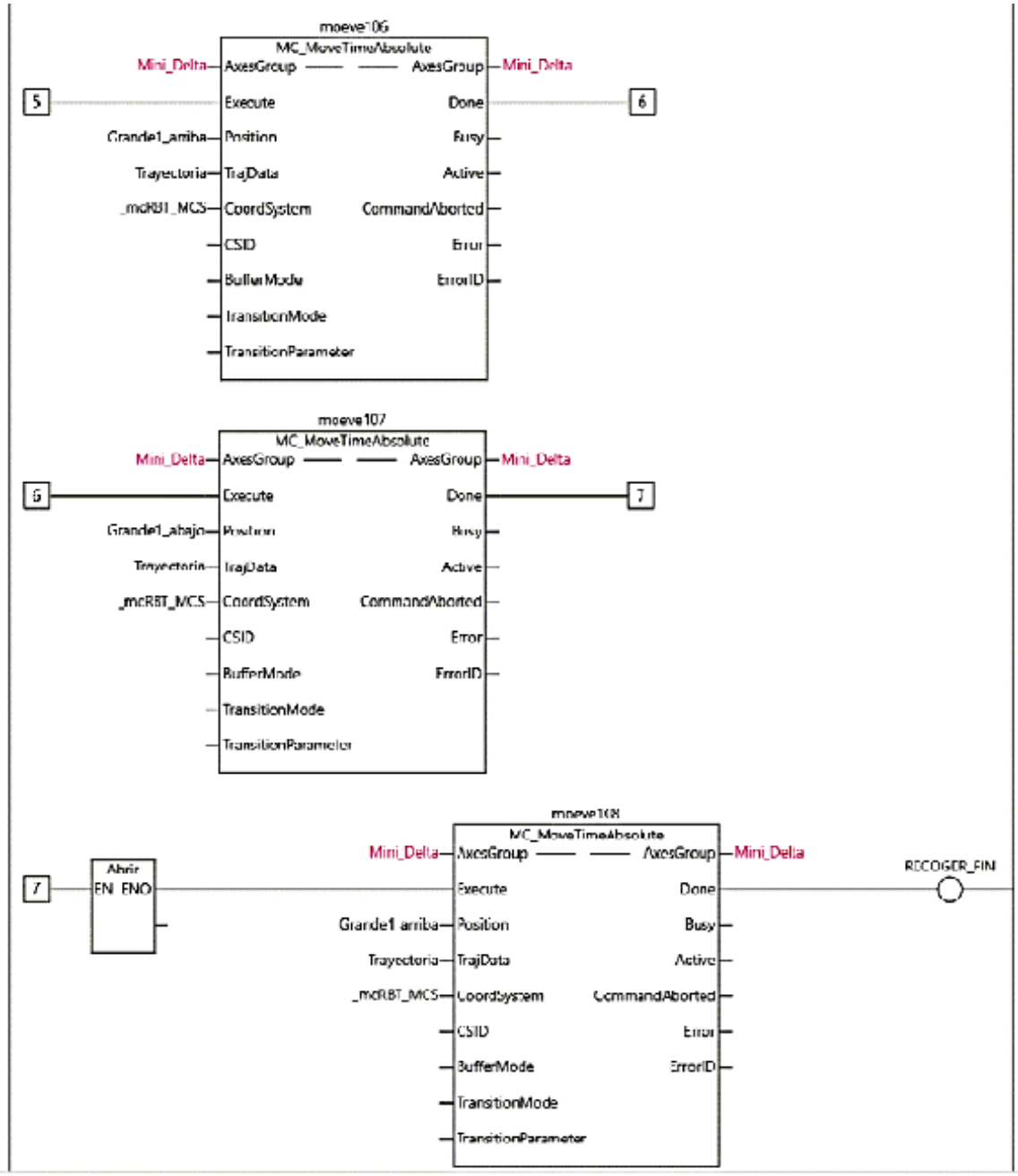

moeve84
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Grande3_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve85
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Grande3_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve86
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
7
Abrir
EN ENO
Execute
Done
RECOGER2
S
Grande3 arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Se comprueba si hay que recoger el cubo 2, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
DIN1
EMPEZAR_RECOGER
ALTO2
RECOGER1
S

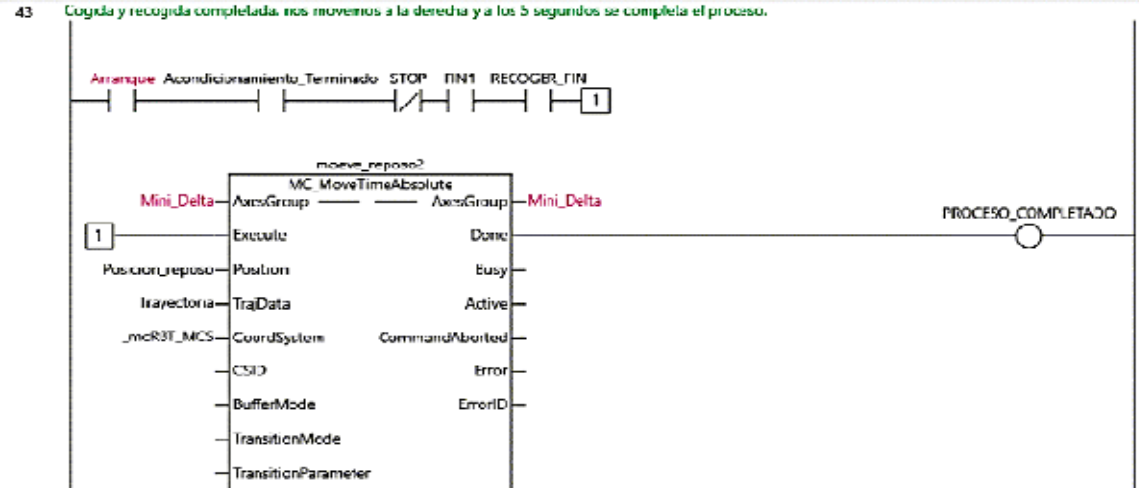

40
Recoger Cubo2.
Arranque
STOP
FIN1
HACER2
1
Acondicionamiento_Terminado
RECOGER2
Moeve97
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
1
2
Moeve98
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Mini_Delta
Trayectoria
_mcfUBT_MCS
2
3
Cerrar
LN LNO
3
4
Moeve99
AxesGroup
Execute
Position
TrajData
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
Mini_Delta
Pequena2_arriba
Trayectoria
_mcfUBT_MCS
4
5

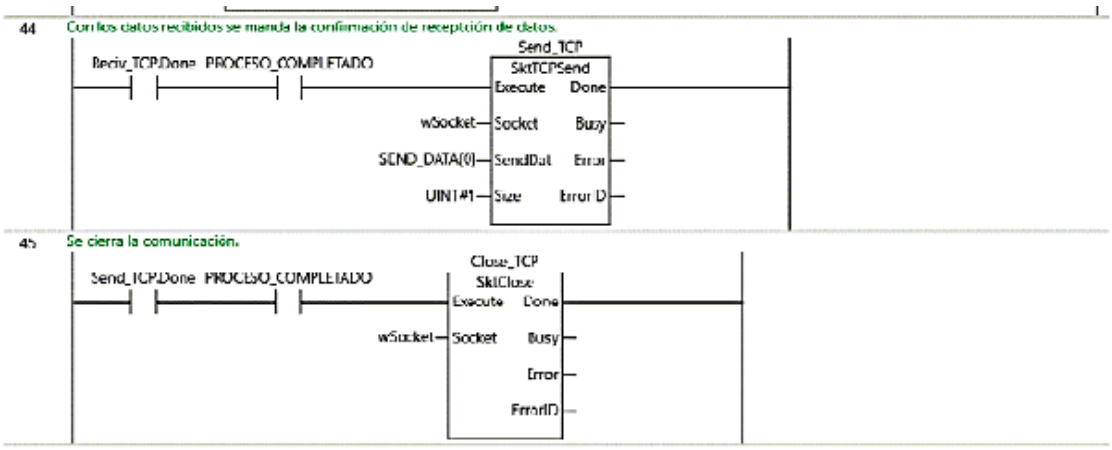

moeve100
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
5
Execute
Done
6
Grande2_arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve101
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
6
Execute
Done
7
Grande2_abajo
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
moeve102
MC_MoveTimeAbsolute
Abrir
7
EN
ENO
Mini_Delta
AxesGroup
AxesGroup
Mini_Delta
RECOGER1
S
Execute
Done
Grande2 arriba
Position
Busy
Trayectoria
TrajData
Active
_mcRBT_MCS
CoordSystem
CommandAborted
CSID
Error
BufferMode
ErrorID
TransitionMode
TransitionParameter
Se comprueba si hay que recoger el cubo 1, en caso negativo, el siguiente segmento no se ejecuta.
Arranque
Acondicionamiento_Terminado
STOP
FIN1
EMPEZAR_RECOGER
ALTO1
RECOGER_FIN
S

42 Recoger Cubo1.



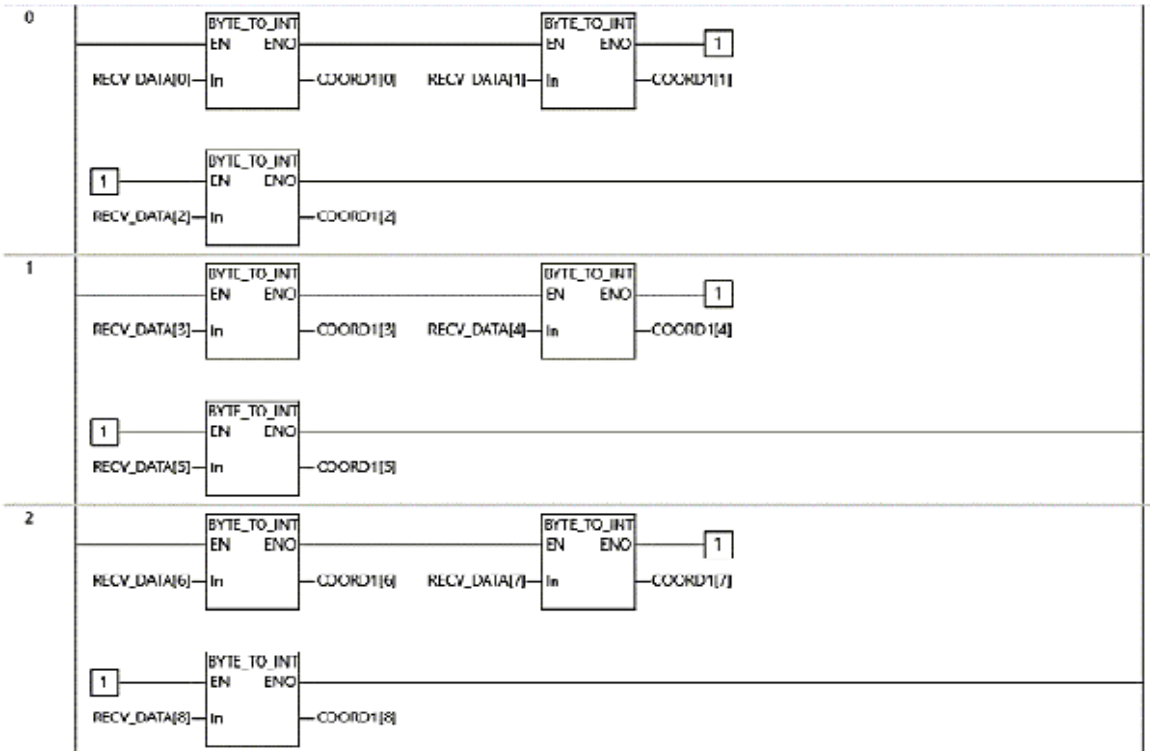

moeve106
MC_MoveTimeAbsolute
Mini_Delta
AxesGroup
Execute
Grande1_arriba
Position
Trayectoria
TrajData
_mcRBT_MCS
CoordSystem
CSID
BufferMode
TransitionMode
TransitionParameter
Done
Busy
Active
CommandAborted
Error
ErrorID
5
6
moeve107
Grande1_abajo
7
8
Abrir
EN ENO
RECOGER_FIN

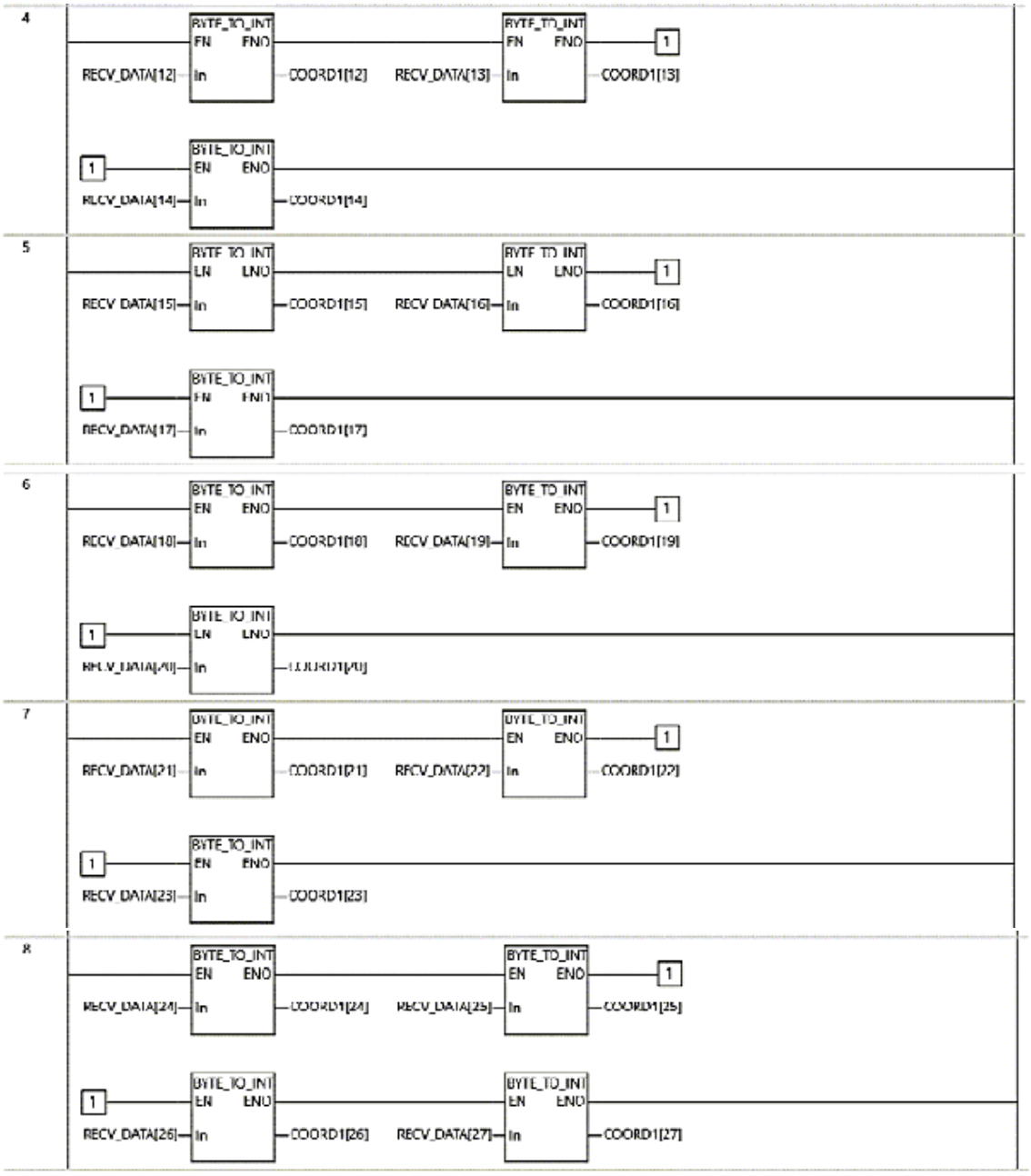

43
Cogida y recogida completada. Los movemos a la derecha y a los 5 segundos se completa el proceso.
Arranque
Acondicionamiento_Terminado
STOP
FIN1
RECOGER_FIN
1
moeve_reposo2
Posicion_reposo
PROCESO_COMPLETADO



II.III. Control de la pinza.

II.IV. Acondicionamiento del vector TCP.




4
BYTE_TO_INT
EN ENO
RECV_DATA[12]
In
COORD1[12]
BYTE_TO_INT
EN ENO
RECV_DATA[13]
In
COORD1[13]
1
BYTE_TO_INT
EN ENO
RECV_DATA[14]
In
COORD1[14]
5
BYTE_TO_INT
EN ENO
RECV_DATA[15]
In
COORD1[15]
BYTE_TO_INT
EN ENO
RECV_DATA[16]
In
COORD1[16]
1
BYTE_TO_INT
EN ENO
RECV_DATA[17]
In
COORD1[17]
6
BYTE_TO_INT
EN ENO
RECV_DATA[18]
In
COORD1[18]
BYTE_TO_INT
EN ENO
RECV_DATA[19]
In
COORD1[19]
1
BYTE_TO_INT
EN ENO
In
7
BYTE_TO_INT
EN ENO
RECV_DATA[21]
In
COORD1[21]
BYTE_TO_INT
EN ENO
RECV_DATA[22]
In
COORD1[22]
1
BYTE_TO_INT
EN ENO
RECV_DATA[23]
In
COORD1[23]
8
BYTE_TO_INT
EN ENO
RECV_DATA[24]
In
COORD1[24]
BYTE_TO_INT
EN ENO
RECV_DATA[25]
In
COORD1[25]
1
BYTE_TO_INT
EN ENO
RECV_DATA[26]
In
COORD1[26]
BYTE_TO_INT
EN ENO
RECV_DATA[27]
In
COORD1[27]

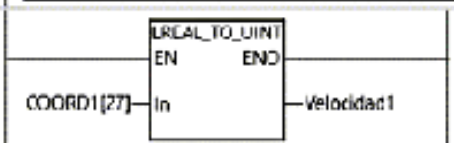
9

```
//Coordenadas reales del primer cubo.
IF RECV_DATA[2]=0 THEN
   COORD[0]:=COORD1[0];
   COORD[1]:=COORD1[1];
ELSIF RECV_DATA[2]=1 THEN
   COORD[0]:=COORD1[0];
   COORD[1]:= COORD1[1];
ELSIF RECV_DATA[2]=2 THEN
   COORD[0]:=-COORD1[0];
   COORD[1]:=COORD1[1];
ELSIF RECV_DATA[2]=3 THEN
   COORD[0]:=-COORD1[0];
   COORD[1]:=-COORD1[1];
END_IF;

//Coordenadas reales del segundo cubo.
IF RECV_DATA[5]=0 THEN
   COORD[3]:=COORD1[3];
   COORD[4]:=COORD1[4];
ELSIF RECV_DATA[5]=1 THEN
   COORD[3]:=COORD1[3];
   COORD[4]:=-COORD1[4];
ELSIF RECV_DATA[5]=2 THEN
   COORD[3]:= COORD1[3];
   COORD[4]:=COORD1[4];
ELSE
   COORD[3]:=-COORD1[3];
   COORD[4]:=-COORD1[4];
END_IF;

//Coordenadas reales del tercer cubo.
IF RECV_DATA[8]=0 THEN
   COORD[6]:=COORD1[6];
   COORD[7]:=COORD1[7];
ELSIF RECV_DATA[8]=1 THEN
   COORD[6]:=COORD1[6];
   COORD[7]:=-COORD1[7];
ELSIF RECV_DATA[8]=2 THEN
   COORD[6]:= COORD1[6];
   COORD[7]:=COORD1[7];
ELSIF RECV_DATA[8]=3 THEN
   COORD[6]:=-COORD1[6];
   COORD[7]:=-COORD1[7];
END_IF;

//Coordenadas reales del cuarto cubo.
IF RECV_DATA[11]=0 THEN
   COORD[9]:=COORD1[9];
   COORD[10]:=COORD1[10];
ELSIF RECV_DATA[11]=1 THEN
   COORD[9]:=-COORD1[9];
   COORD[10]:=-COORD1[10];
ELSIF RECV_DATA[11]=2 THEN
   COORD[9]:=-COORD1[9];
   COORD[10]:=COORD1[10];
ELSIF RECV_DATA[11]=3 THEN
   COORD[9]:=-COORD1[9];
   COORD[10]:=-COORD1[10];
END_IF;

//Coordenadas reales del quinto cubo.
IF RECV_DATA[14]=0 THEN
```

```
    COORD[12]:=COORD1[12];
    COORD[13]:=COORD1[13];
ELSIF RECV_DATA[14]=1 THEN
    COORD[12]:=COORD1[12];
    COORD[13]:=-COORD1[13];
ELSIF RECV_DATA[14]=2 THEN
    COORD[12]:=-COORD1[12];
    COORD[13]:=COORD1[13];
ELSIF RECV_DATA[14]=3 THEN
    COORD[12]:= COORD1[12];
    COORD[13]:= COORD1[13];
END_IF;

//Coordenadas reales del sixto cubo.
IF RECV_DATA[17]=0 THEN
    COORD[15]:=COORD1[15];
    COORD[16]:=COORD1[16];
ELSIF RECV_DATA[17]=1 THEN
    COORD[15]:=COORD1[15];
    COORD[16]:=-COORD1[16];
ELSIF RECV_DATA[17]=2 THEN
    COORD[15]:=-COORD1[15];
    COORD[16]:=COORD1[16];
ELSIF RECV_DATA[17]=3 THEN
    COORD[15]:=-COORD1[15];
    COORD[16]:= COORD1[16];
END_IF;

//Coordenadas reales del septimo cubo.
IF RECV_DATA[20]=0 THEN
    COORD[18]:=COORD1[18];
    COORD[19]:=COORD1[19];
ELSIF RECV_DATA[20]=1 THEN
    COORD[18]:=COORD1[18];
    COORD[19]:= COORD1[19];
ELSIF RECV_DATA[20]=2 THEN
    COORD[18]:=-COORD1[18];
    COORD[19]:=COORD1[19];
ELSIF RECV_DATA[20]=3 THEN
    COORD[18]:=-COORD1[18];
    COORD[19]:=-COORD1[19];
END_IF;

//Coordenadas reales del octavo cubo.
IF RECV_DATA[23]=0 THEN
    COORD[21]:=COORD1[21];
    COORD[22]:=COORD1[22];
ELSIF RECV_DATA[23]=1 THEN
    COORD[21]:= COORD1[21];
    COORD[22]:= COORD1[22];
ELSIF RECV_DATA[23]=2 THEN
    COORD[21]:=-COORD1[21];
    COORD[22]:=COORD1[22];
ELSIF RECV_DATA[23]=3 THEN
    COORD[21]:=-COORD1[21];
    COORD[22]:=-COORD1[22];
END_IF;

//Coordenadas reales del noveno cubo.
IF RECV_DATA[26]=0 THEN
    COORD[24]:=COORD1[24];
    COORD[25]:=COORD1[25];
ELSIF RECV_DATA[26]=1 THEN
```

```
    COORD[24]:=COORD1[24];
    COORD[25]:=-COORD1[25];
ELSIF RECV_DATA[26]=2 THEN
    COORD[24]:=-COORD1[24];
    COORD[25]:=COORD1[25];
ELSIF RECV_DATA[26]=3 THEN
    COORD[24]:=-COORD1[24];
    COORD[25]:=-COORD1[25];
END_IF;

//Velocidad robot.
COORD1[27]:=COORD1[27]*6;
```

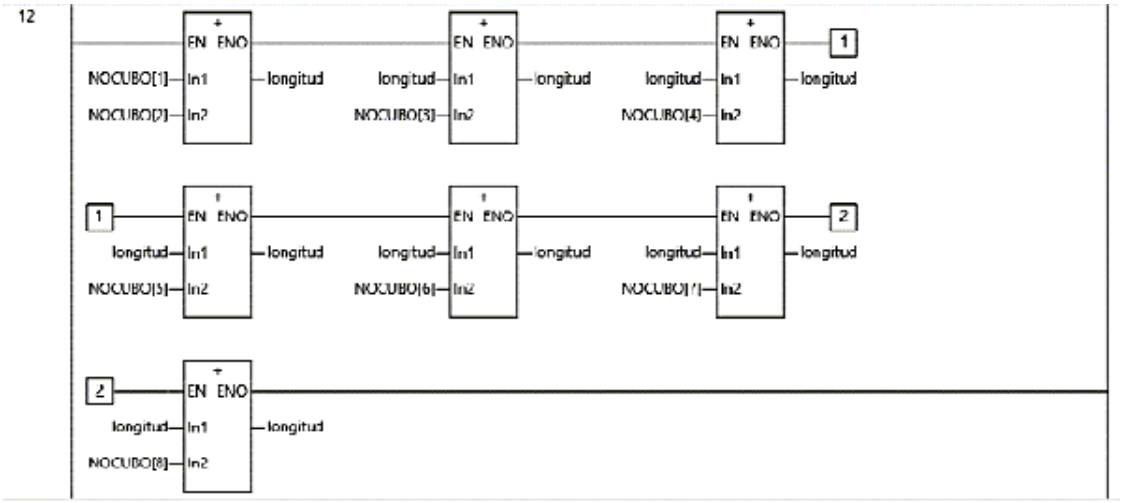
10



11

```
// Se comprueba si tiene longitud 255.
//Coordenadas cubo 1.
IF COORD1[0]=255 THEN
    NOCUBO[0]:=1;
ELSE
    NOCUBO[0]:=0;
END_IF;

 //Coordenadas cubo 2.
IF COORD1[3]=255 THEN
    NOCUBO[1]:=1;
ELSE
    NOCUBO[1]:=0;
END_IF;

//Coordenadas cubo 3.
IF COORD1[6]=255 THEN
    NOCUBO[2]:=1;
ELSE
    NOCUBO[2]:=0;
END_IF;

//Coordenadas cubo 4.
IF COORD1[9]=255 THEN
    NOCUBO[3]:=1;
ELSE
    NOCUBO[3]:=0;
END_IF;

//Coordenadas cubo 5.
IF COORD1[12]=255 THEN
    NOCUBO[4]:=1;
ELSE
    NOCUBO[4]:=0;
END_IF;

//Coordenadas cubo 6.
IF COORD1[15]=255 THEN
```

```
    NOCUBO[5]:=1;
ELSE
    NOCUBO[5]:=0;
END_IF;

//Coordenadas cubo 7.
IF COORD1[18]=255 THEN
    NOCUBO[6]:=1;
ELSE
    NOCUBO[6]:=0;
END_IF;

//Coordenadas cubo 8.
IF COORD1[21]=255 THEN
    NOCUBO[7]:=1;
ELSE
    NOCUBO[7]:=0;
END_IF;

//Coordenadas cubo 9.
IF COORD1[24]=255 THEN
    NOCUBO[8]:=1;
ELSE
    NOCUBO[8]:=0;
```



Anexo III:

CÓDIGO MATLAB

III.I. Función principal.

```
function varargout = control_planta(varargin)
% PLANTA MATLAB code for Planta.fig
%      PLANTA, by itself, creates a new PLANTA or raises the existing
%      singleton*.
%
%      H = PLANTA returns the handle to a new PLANTA or the handle to
%      the existing singleton*.
%
%      PLANTA('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PLANTA.M with the given input arguments.
%
%      PLANTA('Property','Value',...) creates a new PLANTA or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Planta_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Planta_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Planta

% Last Modified by GUIDE v2.5 30-Mar-2022 10:08:30

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @control_planta_OpeningFcn, ...
                   'gui_OutputFcn',  @control_planta_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Planta is made visible.

function control_planta_OpeningFcn(hObject, ~, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Planta (see VARARGIN)
```

```
% Choose default command line output for Planta
handles.output = hObject;

%INICIALIZACIÓN DE VARIABLES.

%Fondo interfaz
axes(handles.axes174);
background=imread('fondoreal1.jpg');
imshow(background);
axis off;

%Fondo robot.
axes(handles.axes298);
background=imread('fondo23.jpg');
imshow(background);
axis off;

%Paneles.
set(handles.togglebutton10,'visible','off');
set(handles.togglebutton11,'visible','off');
set(handles.togglebutton12,'visible','off');
set(handles.uibuttongroup5,'visible','off');
set(handles.uipanel11,'visible','off');
set(handles.uipanel16,'visible','off');
set(handles.uipanel17,'visible','off');
set(handles.uipanel18,'visible','off');
set(handles.uipanel19,'visible','off');
set(handles.uipanel20,'visible','off');
[handles]=LedsInicio(handles);

%Visibilidad textos plataformas.
[handles]=textos_grande(handles);
[handles]=textos_pequeno(handles);

%Comprobar si se puede escribir palabra.
handles.nosepuede=0; %La palabra se puede escribir con un 0.
set(handles.axes299,'visible','off');
set(handles.text620,'visible','off');

%Cerrar ficheros.
fclose('all');

%Botón start/stop.
handles.Ts=0.5;
handles.estado=0; %Estado 0 = stop.

%Palabra.
handles.palabra=[];

%Leds.
handles.ocupado=0;
handles.p1=0;
handles.p2=0;
handles.p3=0;
handles.p4=0;
handles.p5=0;
handles.p6=0;
handles.p7=0;
```

```
handles.p8=0;
handles.p9=0;
handles.p10=0;
handles.p11=0;
handles.p12=0;
handles.p13=0;
handles.p14=0;
handles.p15=0;
handles.p16=0;

%Estado conexion TCP.
set(handles.axes300,'visible','on');
set(handles.text637,'visible','on');
set(handles.axes302,'visible','on');
axes(handles.axes302);
background=imread('Verde.png');
imshow(background);
axis off;
set(handles.text638,'visible','on');
set(handles.text636,'visible','on');
set(handles.text637,'string','Sin conexión');
set(handles.text638,'string','Robot quieto');
axes(handles.axes300);
background=imread('Rojo.png');
imshow(background);
axis off;

%Caras plataformas.
set(handles.axes303,'visible','off');
set(handles.axes304,'visible','off');
set(handles.axes305,'visible','off');
set(handles.axes306,'visible','off');
set(handles.axes307,'visible','off');
handles.contento=0;
handles.triste=0;

%Texto velocidad.
set(handles.text894,'string',0);

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes Planta wait for user response (see UIRESUME)
% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.
function varargout = control_planta_OutputFcn(~, ~, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;

%--------------------------------------------------------------------------
% --- Executes when selected object is changed in uibuttongroup1.

% --- Executes on button press in pushbutton5.
```

```
function pushbutton5_Callback(~, ~, ~)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%BOTON CERRAR PROGRAMA.
close(gcf);

% --- Executes on button press in togglebutton2.
function togglebutton2_Callback(hObject, ~, handles)
% hObject    handle to togglebutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% Hint: get(hObject,'Value') returns toggle state of togglebutton2

%BOTON START/STOP.
    guidata(hObject,handles);
    %START.
if get(hObject,'Value')==1
    handles.pushbutton5.Enable = 'off';
    handles.encendido=1;
    hObject.String = 'Stop';
    set(handles.togglebutton2,'Backgroundcolor','r');
    set(handles.uipanel11,'visible','on');
    set(handles.uibuttongroup5,'visible','on');
    set(handles.togglebutton10,'visible','on');
    set(handles.togglebutton11,'visible','on');
    set(handles.togglebutton12,'visible','on');
    set(handles.uipanel19,'visible','on');

else
    %STOP.
    handles.pushbutton5.Enable = 'on';
    handles.encendido=0;
    hObject.String = 'Start';
    set(handles.togglebutton2,'Backgroundcolor','g');
    set(handles.togglebutton10,'visible','off');
    set(handles.togglebutton11,'visible','off');
    set(handles.togglebutton12,'visible','off');
    set(handles.uibuttongroup5,'visible','off');
    set(handles.uipanel11,'visible','off');
    set(handles.uipanel16,'visible','off');
    set(handles.uipanel17,'visible','off');
    set(handles.uipanel18,'visible','off');
    set(handles.uipanel19,'visible','off');

end
while get(hObject,'Value')==1
    %General para ver los dibujos de las palabras.
    [handles]=VerPalabras(handles);
    %IDIOMA.
    if handles.radiobutton6.Value == 1 %Castellano
            set(handles.pushbutton9,'string','TELEVISOR');
            set(handles.pushbutton44,'string','CABLE');
            set(handles.pushbutton47,'string','PLANOS');
            set(handles.pushbutton50,'string','INGENIERO');
            set(handles.pushbutton53,'string','TECLADO');
            set(handles.pushbutton39,'string','CÁMARA');
```

```
            set(handles.pushbutton42,'string','LÁMPARA');

            set(handles.pushbutton40,'string','COCHE');
            set(handles.pushbutton43,'string','ROBOT');
            set(handles.pushbutton46,'string','MÁSTER');
            set(handles.pushbutton49,'string','ESCUELA');
            set(handles.pushbutton45,'string','ESTUFA');
            set(handles.pushbutton48,'string','PIEZAS');
            set(handles.pushbutton51,'string','MOTOR');
            set(handles.pushbutton54,'string','MÓVIL');

    else % Euskera
            set(handles.pushbutton9,'string','TELEBISTA');
            set(handles.pushbutton44,'string','KABLEA');
            set(handles.pushbutton47,'string','PLANOAK');
            set(handles.pushbutton50,'string','INGENIARI');
            set(handles.pushbutton53,'string','TEKLATUA');
            set(handles.pushbutton39,'string','KAMERA');
            set(handles.pushbutton42,'string','LANPARA');

            set(handles.pushbutton40,'string','AUTOMOBIL');
            set(handles.pushbutton43,'string','ROBOTA');
            set(handles.pushbutton46,'string','MASTERRA');
            set(handles.pushbutton49,'string','ESKOLA');
            set(handles.pushbutton45,'string','BEROGAILU');
            set(handles.pushbutton48,'string','PIEZAK');
            set(handles.pushbutton51,'string','MOTOREA');
            set(handles.pushbutton54,'string','MUGIKORRA');
    end
    drawnow;
    guidata(hObject,handles);
    pause(handles.Ts);
end

guidata(hObject,handles);

% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, ~, handles)
% hObject    handle to pushbutton9 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p1=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='televisor';
    else % Euskera
        handles.palabra='telebista';
    end
    [handles]=General_letras(handles);
end
guidata(hObject,handles);

% --- Executes on button press in pushbutton39.
function pushbutton39_Callback(~, ~, handles)
% hObject    handle to pushbutton39 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
```

```
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p7=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='camara';
    else % Euskera
        handles.palabra='kamera';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton40.
function pushbutton40_Callback(~, ~, handles)
% hObject    handle to pushbutton40 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p13=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='coche';
    else % Euskera
        handles.palabra='automobil';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton41.
function pushbutton41_Callback(~, ~, handles)
% hObject    handle to pushbutton41 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p2=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='upvehu';
    else % Euskera
        handles.palabra='upvehu';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton42.
function pushbutton42_Callback(~, ~, handles)
% hObject    handle to pushbutton42 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p8=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='lampara';
```

```
    else % Euskera
        handles.palabra='lanpara';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton43.
function pushbutton43_Callback(~, ~, handles)
% hObject    handle to pushbutton43 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p14=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='robot';
    else % Euskera
        handles.palabra='robota';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton44.
function pushbutton44_Callback(~, ~, handles)
% hObject    handle to pushbutton44 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p3=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='cable';
    else % Euskera
        handles.palabra='kablea';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton45.
function pushbutton45_Callback(~, ~, handles)
% hObject    handle to pushbutton45 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p9=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='estufa';
    else % Euskera
        handles.palabra='berogailu';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton46.
```

```
function pushbutton46_Callback(~, ~, handles)
% hObject    handle to pushbutton46 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p15=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='master';
    else % Euskera
        handles.palabra='masterra';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton47.
function pushbutton47_Callback(~, ~, handles)
% hObject    handle to pushbutton47 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p4=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='plano';
    else % Euskera
        handles.palabra='planoak';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton48.
function pushbutton48_Callback(~, ~, handles)
% hObject    handle to pushbutton48 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p10=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='piezas';
    else % Euskera
        handles.palabra='piezak';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton49.
function pushbutton49_Callback(~, ~, handles)
% hObject    handle to pushbutton49 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
```

```
    handles.p16=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='escuela';
    else % Euskera
        handles.palabra='eskola';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton50.
function pushbutton50_Callback(~, ~, handles)
% hObject    handle to pushbutton50 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p5=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='ingeniero';
    else % Euskera
        handles.palabra='ingeniari';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton51.
function pushbutton51_Callback(~, ~, handles)
% hObject    handle to pushbutton51 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p11=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='motor';
    else % Euskera
        handles.palabra='motorea';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in pushbutton53.
function pushbutton53_Callback(~, ~, handles)
% hObject    handle to pushbutton53 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p6=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='teclado';
    else % Euskera
        handles.palabra='teklatua';
    end
    [handles]=General_letras(handles);
```

```
end

% --- Executes on button press in pushbutton54.
function pushbutton54_Callback(~, ~, handles)
% hObject    handle to pushbutton54 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

if handles.ocupado==0
    handles.ocupado=1;
    handles.p12=1;
    if handles.radiobutton6.Value == 1 %Castellano
        handles.palabra='movil';
    else % Euskera
        handles.palabra='mugikorra';
    end
    [handles]=General_letras(handles);
end

% --- Executes on button press in togglebutton10.
function togglebutton10_Callback(hObject, ~, handles)
% hObject    handle to togglebutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%PANEL ESCRIBIR PALABRA
if get(hObject,'Value') == 1
    set(handles.uipanel17,'visible','on');
else
    set(handles.uipanel17,'visible','off');
end
guidata(hObject,handles);


% --- Executes on button press in togglebutton11.
function togglebutton11_Callback(hObject, ~, handles)
% hObject    handle to togglebutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%PANEL PLATAFORMA GRANDE
if get(hObject,'Value') == 1
    set(handles.uipanel16,'visible','on');
else
    set(handles.uipanel16,'visible','off');
end
guidata(hObject,handles);

% --- Executes on button press in togglebutton12.
function togglebutton12_Callback(hObject, ~, handles)
% hObject    handle to togglebutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%PANEL PLATAFORMA GRANDE
if get(hObject,'Value') == 1
    set(handles.uipanel18,'visible','on');
else
```

```
    set(handles.uipanel18,'visible','off');
end
guidata(hObject,handles);

function edit8_Callback(hObject, ~, handles)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of edit8 as text
%        str2double(get(hObject,'String')) returns contents of edit8 as a double

%BOTON DE COMPROBAR PALABRA.
handles.comprobar=get(hObject,'String');
handles.comprobar=lower(handles.comprobar);
[handles]=General_letras_comprobar(handles);

guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function edit8_CreateFcn(hObject, ~, ~)
% hObject    handle to edit8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in pushbutton118.
function pushbutton118_Callback(~, ~, ~)
% hObject    handle to pushbutton118 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --- Executes on button press in pushbutton119.
function pushbutton119_Callback(~, ~, ~)
% hObject    handle to pushbutton119 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% --------------------------------------------------------------------
function Fichero_escritura_Callback(~, ~, ~)
% hObject    handle to Fichero_escritura (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('Historial_Palabras.txt');

% --------------------------------------------------------------------
function guia_usuario_Callback(~, ~, ~)
% hObject    handle to guia_usuario (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
winopen('Manual_Usuario.txt');
```

```
% --- Executes on slider movement.
function slider14_Callback(hObject, eventdata, handles)
% hObject    handle to slider14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'Value') returns position of slider
%        get(hObject,'Min') and get(hObject,'Max') to determine range of slider

%Slider Velocidad.
handles.velocidad=get(hObject,'value');
handles.velocidad=round(handles.velocidad);
set(handles.text894,'string',handles.velocidad);
guidata(hObject,handles);

% --- Executes during object creation, after setting all properties.
function slider14_CreateFcn(hObject, eventdata, handles)
% hObject    handle to slider14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: slider controls usually have a light gray background.
if isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor',[.9 .9 .9]);
end
```

III.II. Función comprobar palabras.

```
function [handles]=Comprobar_formar_palabra(handles)

%Cuando no hay suficientes letras para escribir la palabra no se escribe.
for i=1:length(handles.caracteres)
    if handles.caracteres(i)=='ñ'
        handles.nosepuede=1;
    end
end

%Cuando no hay suficientes letras para escribir la palabra no se escribe.
for i=1:length(handles.Posicion_x)
    if handles.Posicion_x(i)==0
        handles.nosepuede=1;
    elseif handles.Posicion_y(i)==0
        handles.nosepuede=1;
    end
end

%Si se superan las nueve letras.
if length(handles.Posicion_x)>9
    handles.nosepuede=1;
end

%Si se puede escribir.
if handles.nosepuede==0
    set(handles.text620,'string','Palabra Disponible');
```

```
    set(handles.text620,'visible','on');

    %LED VERDE.
    set(handles.axes299,'visible','on');
    axes(handles.axes299);
    background=imread('Verde.png');
    imshow(background);
    axis off;

else
    set(handles.text620,'string','Palabra No Disponible');
    set(handles.text620,'visible','on');

    %LED ROJO.
    set(handles.axes299,'visible','on');
    axes(handles.axes299);
    background=imread('Rojo.png');
    imshow(background);
    axis off;

end
```

III.II. Comunicación TCP.

```
function [handles]=comunicacion_TCP(handles)

%Color led comunicacion robot.
axes(handles.axes300);
background=imread('Verde.png');
imshow(background);
axis off;

%Color led robot en movimiento.
axes(handles.axes302);
background=imread('Warning.png');
imshow(background);
axis off;

%Se definen los textos de la comunicación.
set(handles.text637,'string','Comunicación establecida');
set(handles.text638,'string','Robot en movimiento');

pause(0.1)

%Defino el path del programa de python.
pathtosend=fileparts(which('miarchivo.py'));
if count(py.sys.path,pathtosend)==0
    insert(py.sys.path,int32(0),pathtosend);
end

% Convierto la matriz en bytes de 0 a 255.
T=uint32(handles.vector_TCP);

%Llamo a la funcion de python.
```

```
py.miarchivo.add_numbers_2(T(1), T(2), T(3), T(4), T(5), T(6), T(7), T(8), T(9), T(10),
T(11), T(12), T(13), T(14), T(15), T(16), T(17), T(18), T(19), T(20), T(21), T(22), T(23),
T(24), T(25), T(26), T(27), T(28));

%Color led comunicacion robot.
axes(handles.axes300);
background=imread('Rojo.png');
imshow(background);
axis off;

%Color led robot en movimiento.
set(handles.axes302,'visible','on');
axes(handles.axes302);
background=imread('Verde.png');
imshow(background);
axis off;

%Se definen los textos de la comunicación.
set(handles.text637,'string','Sin conexión');
set(handles.text638,'string','Robot quieto');
```

III.III. Escoger cara.

```
function [handles]=contento_triste(handles)

%Easter egg Visens.
if handles.caracteres(1)=='v' && length(handles.palabra)==6
    if handles.caracteres(2)=='i'
        if handles.caracteres(3)=='s'
            if handles.caracteres(4)=='e'
                if handles.caracteres(5)=='n'
                    if handles.caracteres(6)=='s'

                        set(handles.uipanel20,'visible','on');
                        set(handles.axes301,'visible','on');
                        axes(handles.axes301);
                        background=imread('foto_visens.png');
                        imshow(background);
                        axis off;

                        %Se pone cubo contento.
                        handles.matriz_letras(9,7)=0;
                        i=length(handles.Posicion_x);
                        handles.Posicion_x(i+1)=9;
                        handles.Posicion_y(i+1)=7;
                        handles.contento=1;

                    end
                end
            end
        end
    end
end
```

```
%Easter egg Itziar.
if handles.caracteres(1)=='i' && length(handles.palabra)==6
    if handles.caracteres(2)=='t'
        if handles.caracteres(3)=='z'
            if handles.caracteres(4)=='i'
                if handles.caracteres(5)=='a'
                    if handles.caracteres(6)=='r'

                        %Se pone cubo contento.
                        handles.matriz_letras(9,7)=0;
                        i=length(handles.Posicion_x);
                        handles.Posicion_x(i+1)=9;
                        handles.Posicion_y(i+1)=7;
                        handles.contento=1;

                    end
                end
            end
        end
    end
end

%Easter egg Andrea.
if handles.caracteres(1)=='a' && length(handles.palabra)==6
    if handles.caracteres(2)=='n'
        if handles.caracteres(3)=='d'
            if handles.caracteres(4)=='r'
                if handles.caracteres(5)=='e'
                    if handles.caracteres(6)=='a'

                        %Se pone cubo contento.
                        handles.matriz_letras(9,7)=0;
                        i=length(handles.Posicion_x);
                        handles.Posicion_x(i+1)=9;
                        handles.Posicion_y(i+1)=7;
                        handles.contento=1;

                    end
                end
            end
        end
    end
end

%Easter egg Cesar.
if handles.caracteres(1)=='c' && length(handles.palabra)==5
    if handles.caracteres(2)=='e'
        if handles.caracteres(3)=='s'
            if handles.caracteres(4)=='a'
                if handles.caracteres(5)=='r'

                    %Se pone cubo contento.
                    handles.matriz_letras(9,7)=0;
                    i=length(handles.Posicion_x);
                    handles.Posicion_x(i+1)=9;
                    handles.Posicion_y(i+1)=7;
                    handles.contento=1;
```

```
                end
            end
        end
    end
end

%Easter egg Aitziber.
if handles.caracteres(1)=='a' && length(handles.palabra)==8
    if handles.caracteres(2)=='i'
        if handles.caracteres(3)=='t'
            if handles.caracteres(4)=='z'
                if handles.caracteres(5)=='i'
                    if handles.caracteres(6)=='b'
                        if handles.caracteres(7)=='e'
                            if handles.caracteres(8)=='r'

                                %Se pone cubo contento.
                                handles.matriz_letras(9,7)=0;
                                i=length(handles.Posicion_x);
                                handles.Posicion_x(i+1)=9;
                                handles.Posicion_y(i+1)=7;
                                handles.contento=1;

                            end
                        end
                    end
                end
            end
        end
    end
end

%Easter egg patricia.
if handles.caracteres(1)=='p' && length(handles.palabra)==8
    if handles.caracteres(2)=='a'
        if handles.caracteres(3)=='t'
            if handles.caracteres(4)=='r'
                if handles.caracteres(5)=='i'
                    if handles.caracteres(6)=='c'
                        if handles.caracteres(7)=='i'
                            if handles.caracteres(8)=='a'

                                %Se pone cubo contento.
                                handles.matriz_letras(9,7)=0;
                                i=length(handles.Posicion_x);
                                handles.Posicion_x(i+1)=9;
                                handles.Posicion_y(i+1)=7;
                                handles.contento=1;

                            end
                        end
                    end
                end
            end
        end
    end
end
```

```
%Easter egg roberto.
if handles.caracteres(1)=='r' && length(handles.palabra)==7
    if handles.caracteres(2)=='o'
        if handles.caracteres(3)=='b'
            if handles.caracteres(4)=='e'
                if handles.caracteres(5)=='r'
                    if handles.caracteres(6)=='t'
                        if handles.caracteres(7)=='o'

                            %Se pone cubo contento.
                            handles.matriz_letras(9,7)=0;
                            i=length(handles.Posicion_x);
                            handles.Posicion_x(i+1)=9;
                            handles.Posicion_y(i+1)=7;
                            handles.contento=1;

                        end
                    end
                end
            end
        end
    end
end

%Easter egg patrick.
if handles.caracteres(1)=='p' && length(handles.palabra)==7
    if handles.caracteres(2)=='a'
        if handles.caracteres(3)=='t'
            if handles.caracteres(4)=='r'
                if handles.caracteres(5)=='i'
                    if handles.caracteres(6)=='c'
                        if handles.caracteres(7)=='k'

                            %Se pone cubo contento.
                            handles.matriz_letras(9,7)=0;
                            i=length(handles.Posicion_x);
                            handles.Posicion_x(i+1)=9;
                            handles.Posicion_y(i+1)=7;
                            handles.contento=1;

                        end
                    end
                end
            end
        end
    end
end

%Easter egg alazne.
if handles.caracteres(1)=='a' && length(handles.palabra)==6
    if handles.caracteres(2)=='l'
        if handles.caracteres(3)=='a'
            if handles.caracteres(4)=='z'
                if handles.caracteres(5)=='n'
                    if handles.caracteres(6)=='e'

                        %Se pone cubo contento.
                        handles.matriz_letras(9,7)=0;
```

```
                        i=length(handles.Posicion_x);
                        handles.Posicion_x(i+1)=9;
                        handles.Posicion_y(i+1)=7;
                        handles.contento=1;

                    end
                end
            end
        end
    end
end

%Easter egg caca.
if handles.caracteres(1)=='c' && length(handles.palabra)==4
    if handles.caracteres(2)=='a'
        if handles.caracteres(3)=='c'
            if handles.caracteres(4)=='a'

                %Se pone cubo contento.
                handles.matriz_letras(1,7)=0;
                i=length(handles.Posicion_x);
                handles.Posicion_x(i+1)=1;
                handles.Posicion_y(i+1)=7;
                handles.triste=1;

            end
        end
    end
end

% Si se va a coger alguna cara, se muestra en la casilla correspondiente
% en la plataforma grande.

if handles.contento==1
    set(handles.axes303,'visible','on');
    axes(handles.axes303);
    background=imread('sonrie.jpg');
    imshow(background);
    axis off;
end

if handles.triste==1
    set(handles.axes304,'visible','on');
    axes(handles.axes304);
    background=imread('triste.jpg');
    imshow(background);
    axis off;
end
```

III.IV. Control color pilotos.

```
function [handles]=controlluces(handles)

% Se ponen luces rojas.
```

```
%F1
axes(handles.axes124);
background=imread('Rojo.png');
imshow(background);
axis off;

%F2
axes(handles.axes127);
background=imread('Rojo.png');
imshow(background);
axis off;

%F3
axes(handles.axes130);
background=imread('Rojo.png');
imshow(background);
axis off;

%F4
axes(handles.axes133);
background=imread('Rojo.png');
imshow(background);
axis off;

%F5
axes(handles.axes136);
background=imread('Rojo.png');
imshow(background);
axis off;

%F6
axes(handles.axes139);
background=imread('Rojo.png');
imshow(background);
axis off;

%F7
axes(handles.axes142);
background=imread('Rojo.png');
imshow(background);
axis off;

%F8
axes(handles.axes145);
background=imread('Rojo.png');
imshow(background);
axis off;

%F13
axes(handles.axes160);
background=imread('Rojo.png');
imshow(background);
axis off;

%F14
axes(handles.axes163);
background=imread('Rojo.png');
```

```
imshow(background);
axis off;

%F15
axes(handles.axes166);
background=imread('Rojo.png');
imshow(background);
axis off;

%F16
axes(handles.axes169);
background=imread('Rojo.png');
imshow(background);
axis off;

%F9
axes(handles.axes148);
background=imread('Rojo.png');
imshow(background);
axis off;

%F10
axes(handles.axes151);
background=imread('Rojo.png');
imshow(background);
axis off;

%F11
axes(handles.axes154);
background=imread('Rojo.png');
imshow(background);
axis off;

%F12
axes(handles.axes157);
background=imread('Rojo.png');
imshow(background);
axis off;

% Luces de trabajo.

if handles.p1==1
    axes(handles.axes124);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p2==1
    axes(handles.axes127);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p3==1
    axes(handles.axes130);
    background=imread('Amarillo.png');
```

```
    imshow(background);
    axis off;
end

if handles.p4==1
    axes(handles.axes133);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p5==1
    axes(handles.axes136);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p6==1
    axes(handles.axes139);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p7==1
    axes(handles.axes142);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p8==1
    axes(handles.axes145);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p9==1
    axes(handles.axes148);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p10==1
    axes(handles.axes151);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p11==1
    axes(handles.axes154);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
```

```
end

if handles.p12==1
    axes(handles.axes157);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p13==1
    axes(handles.axes160);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p14==1
    axes(handles.axes163);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p15==1
    axes(handles.axes166);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end

if handles.p16==1
    axes(handles.axes169);
    background=imread('Amarillo.png');
    imshow(background);
    axis off;
end
```

III.V. Actualizar color pilotos.

```
function [handles]=controlluces_terminado(handles)

% Volver a poner los leds en disponible.

%F1
axes(handles.axes124);
background=imread('Verde.png');
imshow(background);
axis off;

%F2
axes(handles.axes127);
background=imread('Verde.png');
imshow(background);
axis off;
```

```
%F3
axes(handles.axes130);
background=imread('Verde.png');
imshow(background);
axis off;

%F4
axes(handles.axes133);
background=imread('Verde.png');
imshow(background);
axis off;

%F5
axes(handles.axes136);
background=imread('Verde.png');
imshow(background);
axis off;

%F6
axes(handles.axes139);
background=imread('Verde.png');
imshow(background);
axis off;

%F7
axes(handles.axes142);
background=imread('Verde.png');
imshow(background);
axis off;

%F8
axes(handles.axes145);
background=imread('Verde.png');
imshow(background);
axis off;

%F13
axes(handles.axes160);
background=imread('Verde.png');
imshow(background);
axis off;

%F14
axes(handles.axes163);
background=imread('Verde.png');
imshow(background);
axis off;

%F15
axes(handles.axes166);
background=imread('Verde.png');
imshow(background);
axis off;

%F16
axes(handles.axes169);
background=imread('Verde.png');
```

```
imshow(background);
axis off;

%F9
axes(handles.axes148);
background=imread('Verde.png');
imshow(background);
axis off;

%F10
axes(handles.axes151);
background=imread('Verde.png');
imshow(background);
axis off;

%F11
axes(handles.axes154);
background=imread('Verde.png');
imshow(background);
axis off;

%F12
axes(handles.axes157);
background=imread('Verde.png');
imshow(background);
axis off;
```

III.VI. Extraer coordenadas de cada cubo.

```
Function [posy,posx]=deteccion_coordenadas(i,alfa,matriz_coordenadasx,matriz_coordenadasy,
                                           valor, centroids)

coordenadax=round(centroids(i,1)*alfa);
coordenaday=round(centroids(i,2)*alfa);

for v1=1:9%length(vector_coordenadax)
    valornuevox=abs(coordenadax-matriz_coordenadasx(1,v1));
    if valornuevox<valor
        posx=v1;
    end
    valor=valornuevox;
end
valor=100;
for v2=1:7%length(vector_coordenaday)
    valornuevoy=abs(coordenaday-matriz_coordenadasy(v2,1));
    if valornuevoy<valor
        posy=v2;
    end
    valor=valornuevoy;
end
```

III.VII. Crear vector TCP.

```
function [handles]=enviar_coordenadas_TCP(handles)

% Se obtienen las coordenadas del robot.
for i=1:length(handles.Posicion_x)

    valor_X=handles.coordenadas_robot_x(handles.Posicion_y(i),handles.Posicion_x(i));
    valor_Y=handles.coordenadas_robot_y(handles.Posicion_y(i),handles.Posicion_x(i));

    % Se define el segundo interprete.
    if valor_X>=0 && valor_Y>=0
        interprete2=0;
    elseif valor_X>=0 && valor_Y<=0
        interprete2=1;
    elseif valor_X<=0 && valor_Y>=0
        interprete2=2;
    else
        interprete2=3;
    end

    %Se define cada cubo de datos (coordx, coordy, d).
    coordenada_TCP_x=valor_X;
    coordenada_TCP_y=valor_Y;
    coordenada_TCP_d=interprete2;

    %Se guarda en el vector segun su valor.
    if i==1
        vector_TCP(1,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end
    if i==2
        vector_TCP(2,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end
    if i==3
        vector_TCP(3,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end
    if i==4
        vector_TCP(4,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end
    if i==5
        vector_TCP(5,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end
    if i==6
        vector_TCP(6,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end
    if i==7
        vector_TCP(7,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end
    if i==8
        vector_TCP(8,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end
    if i==9
        vector_TCP(9,:)=[coordenada_TCP_x,coordenada_TCP_y,coordenada_TCP_d];
    end

end
```

```
%Se rellena el resto de filas con 255.
[size1,size2]=size(vector_TCP);
while size1<9
    vector_TCP=[vector_TCP;255 0 0];
    [size1,size2]=size(vector_TCP);
end
%Se pasan las conlumnas a filas y el vector se completa.
vector_TCP=[vector_TCP(1,:),vector_TCP(2,:),vector_TCP(3,:),vector_TCP(4,:),vector_TCP(5,:),v
ector_TCP(6,:),vector_TCP(7,:),vector_TCP(8,:),vector_TCP(9,:),handles.velocidad1];

%Si las coordenadas son negativas, se convierten en positivas
% porque los bytes van de 0 a 255.
for i=1:length(vector_TCP)
    if vector_TCP(i)<0
        vector_TCP(i)=-vector_TCP(i);
    end
end

%Se pasa de variable interna a global.
handles.vector_TCP=vector_TCP;
```

III.VIII. Exportar fichero palabras.

```
function [handles]=exportarfichero(handles)

%Si la palabra se ha podido escribir.
if handles.nosepuede==0
    %Abrir fichero de escritura.
    fid=fopen('Historial_Palabras.txt','a');

    %Pintar datos.
    fprintf(fid,'%s','Fecha: ');
    fprintf(fid,'%s ',datetime);
    fprintf(fid,'%s',' --->  Palabra escrita: ');
    fprintf(fid,'%s \n',handles.palabra);

    %Cerrar fichero.
    fclose(fid);
end
```

III.IX. Secuencia panel palabras.

```
function [handles]=General_letras(handles)

%--------------------------------------------------------------------------
%Ocultar pulsadores
%--------------------------------------------------------------------------
[handles]=ocultar_pulsadores(handles);

%--------------------------------------------------------------------------
```

```
%Control de las luces
%--------------------------------------------------------------------------
[handles]=controlluces(handles);

%--------------------------------------------------------------------------
%Se borran los textos de la plataforma grande.
%--------------------------------------------------------------------------
[handles]=textos_grande(handles);
[handles]=textos_pequeno(handles);

%--------------------------------------------------------------------------
%Función de velocidad.
%--------------------------------------------------------------------------
[handles]=Velocidad_Robot(handles);

%--------------------------------------------------------------------------
%Extracción de las coordenadas de los cubos de la palabra.
%--------------------------------------------------------------------------
[handles]=Inicio_Vision(handles);

%--------------------------------------------------------------------------
%Pintar las letras en la plataforma grande.
%--------------------------------------------------------------------------
[handles]=plataforma_grande(handles);

%--------------------------------------------------------------------------
%Pintar las letras en la plataforma pequeña.
%--------------------------------------------------------------------------
[handles]=plataforma_pequena(handles);

%--------------------------------------------------------------------------
%Se exporta la palabra al fichero.
%--------------------------------------------------------------------------
[handles]=exportarfichero(handles);

%--------------------------------------------------------------------------
%Se calculan las coordenadas del robot para mandarlas por TCP.
%--------------------------------------------------------------------------
[handles]=enviar_coordenadas_TCP(handles);

%--------------------------------------------------------------------------
%Se envian los datos por TCP al controlador del robot.
%--------------------------------------------------------------------------
[handles]=comunicacion_TCP(handles);

%--------------------------------------------------------------------------
%Volver a poner los leds en disponible.
%--------------------------------------------------------------------------
[handles]=controlluces_terminado(handles);

%--------------------------------------------------------------------------
%Mostrar pulsadores
%--------------------------------------------------------------------------
[handles]=mostrar_pulsadores(handles);
```

III.X. Secuencia panel comprobar palabras.

```
function [handles]=General_letras_comprobar(handles)

%--------------------------------------------------------------------------
%Ocultar pulsadores
%--------------------------------------------------------------------------
[handles]=ocultar_pulsadores(handles);

%--------------------------------------------------------------------------
%Control de las luces
%--------------------------------------------------------------------------
[handles]=controlluces(handles);
handles.palabra=handles.comprobar;

%--------------------------------------------------------------------------
%Se borran los textos de la plataforma grande.
%--------------------------------------------------------------------------
[handles]=textos_grande(handles);
[handles]=textos_pequeno(handles);

%--------------------------------------------------------------------------
%Función de velocidad.
%--------------------------------------------------------------------------
[handles]=Velocidad_Robot(handles);

%--------------------------------------------------------------------------
%Extracción de las coordenadas de los cubos de la palabra.
%--------------------------------------------------------------------------
[handles]=Inicio_Vision(handles);

%--------------------------------------------------------------------------
%Comprobar si se puede formar la palabra
%--------------------------------------------------------------------------
[handles]=Comprobar_formar_palabra(handles);

if handles.nosepuede==0

    %--------------------------------------------------------------------------
    %Se calcula si se van a mostrar las caritas de contento o triste.
    %--------------------------------------------------------------------------
    [handles]=contento_triste(handles);

    %--------------------------------------------------------------------------
    %Pintar las letras en la plataforma grande y pequeña.
    %--------------------------------------------------------------------------
    [handles]=plataforma_grande(handles);
    [handles]=plataforma_pequena(handles);

    %--------------------------------------------------------------------------
    %Se calculan las coordenadas del robot para mandarlas por TCP.
    %--------------------------------------------------------------------------
    [handles]=enviar_coordenadas_TCP(handles);
    %handles.vector_TCP

    %--------------------------------------------------------------------------
```

```
    %Se envian los datos por TCP al controlador del robot.
    %--------------------------------------------------------------------------
    [handles]=comunicacion_TCP(handles);

    %--------------------------------------------------------------------------
    %Pintar las letras en la plataforma pequeña.
    %--------------------------------------------------------------------------
    [handles]=exportarfichero(handles);

else

    %--------------------------------------------------------------------------
    %Se escribe no disponile en la plataforma grande.
    %--------------------------------------------------------------------------
    [handles]=plataforma_grande_mala(handles);
    pause(2);

end

%--------------------------------------------------------------------------
%Volver a poner los leds en disponible.
%--------------------------------------------------------------------------
[handles]=controlluces_terminado(handles);

%--------------------------------------------------------------------------
%Mostrar pulsadores
%--------------------------------------------------------------------------
[handles]=mostrar_pulsadores(handles);


set(handles.uipanel20,'visible','off');
handles.nosepuede=0;
handles.contento=0;
handles.triste=0;
```

III.XI. Visión artificial.

```
function [handles]=Inicio_Vision(handles)

%Sacar foto y guardar para panel.
cam=webcam;
fotopanel=snapshot(cam);
fotopanel1=imcrop(fotopanel,[184.51,66.51,398.98,349.98]);
imwrite(fotopanel1,'result.jpg');
%Fondo robot.
axes(handles.axes298);
background=imread('result.jpg');
imshow(background);

% Charge image.
I=imread('result.jpg');
% figure(),imshow(I);
I=imcrop(I,[316.51,236.51,403.98,247.98]);
```

```
axis off;

% %% 0. Calibrate camera.
patern=imread('Pattern.bmp');
% figure(),imshow(patern)
% Select the start and finish pixels to know the pixels of a grid.
h=imdistline(gca,[500 500],[96 172]);
% Amount of pixels.
dist=76;
% 1square= 30mmx30mm
% Alfa parameter to conver pixels in mm.
alfa=30/dist; % mm/pixel

% %% 1. Image pre-processing.
% Change color space to gray.
Igray=rgb2gray(I);
% figure(),imshow(Igray);

bw=imbinarize(Igray,0.79);
% figure(),imshow(bw);

%Use morphological transofrmation to get final image.
bw=bwmorph(bw,'open',1);
bw=bwmorph(bw,'open',1);
bw=bwmorph(bw,'erosion',1);
bw=bwmorph(bw,'dilate',1);
% figure(),imshow(bw);

% %% 1. Detect objects.
%Centroid.
s=regionprops(bw,'centroid');
stats = regionprops('table',bw,'Area','Perimeter');
centroids=cat(1,s.Centroid);

% figure(), imshow(I);
% hold on
% plot(centroids(:,1),centroids(:,2),'b*');
% hold off

%Area.
area=regionprops(bw,'Area');
%Perimeter.
perimeter=regionprops(bw,'Perimeter');

for i=1:length(perimeter)
    perimetro(i)=round(perimeter(i).Perimeter);
end

% %% 2. Se crean las matrices de coordenadas.
valor=inf;

% Para cuando no haya letras suficientes
a=0; b=0; c=0; d=0; e=0; f=0; g=0; h=0; i=0; j=0; k=0; l=0; m=0;
n=0; o=0; p=0; q=0; r=0; s=0; t=0; u=0; v=0; w=0; x=0; y=0; z=0;
handles.nosepuede=0;

% %% 3. Se extraen las posiciones de las letras de la plataforma.
% Generate a vector of zeros of the object that we have. % 27 characters.
```

```
vector_coordenadax=1:5;
vector_coordenaday=1:5;
detector=zeros(length(perimeter));
for i=1:length(perimeter)
    %El detector es el perimetro ya redondeado.
    detector(i)=perimetro(i);

    if (area(i).Area==85 && perimetro(i)==46) || (area(i).Area==56 && perimetro(i)==36) ||
(area(i).Area==107 && perimetro(i)==51)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %A
        handles.matriz_letras(posy,posx)=1;
        valor=inf;
        a=a+1;

    elseif (area(i).Area==80 && perimetro(i)==39) ||(area(i).Area==92 && perimetro(i)==42) ||
(area(i).Area==154 && perimetro(i)==54)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %B
        handles.matriz_letras(posy,posx)=2;
        valor=inf;
        b=b+1;

      elseif (area(i).Area==62 && perimetro(i)==55) || (area(i).Area==53 && perimetro(i)==51)
```

```
|| (area(i).Area==100 && perimetro(i)==70)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %C
        handles.matriz_letras(posy,posx)=3;
        valor=inf;
        c=c+1;

      elseif (area(i).Area==63 && perimetro(i)==37) || (area(i).Area==105 &&
perimetro(i)==47) ||(area(i).Area==84 && perimetro(i)==40)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %D
        handles.matriz_letras(posy,posx)=4;
        valor=inf;
        d=d+1;

      elseif  (area(i).Area==86 && perimetro(i)==71) || (area(i).Area==87 &&
perimetro(i)==72) || (area(i).Area==103 && perimetro(i)==77)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
```

```
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %E
        handles.matriz_letras(posy,posx)=5;
        valor=inf;
        e=e+1;

    elseif (area(i).Area==78 && perimetro(i)==58) || (area(i).Area==41 && perimetro(i)==44)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %F
        handles.matriz_letras(posy,posx)=6;
        valor=inf;
        f=f+1;

    elseif (area(i).Area==73 && perimetro(i)==72) || (area(i).Area==86 && perimetro(i)==75)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
```

```
        end
        %G
        handles.matriz_letras(posy,posx)=7;
        valor=inf;
        g=g+1;

      elseif (area(i).Area==100 && perimetro(i)==72) || (area(i).Area==59 &&
perimetro(i)==53)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %H
        handles.matriz_letras(posy,posx)=8;
        valor=inf;
        h=h+1;

      elseif (area(i).Area==31 && perimetro(i)==22) || (area(i).Area==41 && perimetro(i)==28)
|| (area(i).Area==30 && perimetro(i)==24)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %I
        handles.matriz_letras(posy,posx)=9;
        valor=inf;
        i=i+1;

      elseif (area(i).Area==36 && perimetro(i)==34) ||(area(i).Area==49 && perimetro(i)==41)
        coordenadax=round(centroids(i,1)*alfa);
```

```
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %J
        handles.matriz_letras(posy,posx)=10;
        valor=inf;
        j=j+1;

      elseif (area(i).Area==124 && perimetro(i)==80) || (area(i).Area==125 &&
perimetro(i)==76)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %K
        handles.matriz_letras(posy,posx)=11;
        valor=inf;
        k=k+1;

      elseif (area(i).Area==33 && perimetro(i)==37) ||(area(i).Area==56 && perimetro(i)==44)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
```

```
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %L
        handles.matriz_letras(posy,posx)=12;
        valor=inf;
        l=l+1;

      elseif (area(i).Area==198 && perimetro(i)==118) ||(area(i).Area==210 &&
perimetro(i)==117)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %M
        handles.matriz_letras(posy,posx)=13;
        valor=inf;
        m=m+1;

      elseif (area(i).Area==115 && perimetro(i)==75) || (area(i).Area==91 &&
perimetro(i)==64) || (area(i).Area==85 && perimetro(i)==63)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
```

```
        %N
        handles.matriz_letras(posy,posx)=14;
        valor=inf;
        n=n+1;

      elseif (area(i).Area==57 && perimetro(i)==34) || (area(i).Area==138 &&
perimetro(i)==52) || (area(i).Area==65 && perimetro(i)==37)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %O
        handles.matriz_letras(posy,posx)=15;
        valor=inf;
        o=o+1;

      elseif (area(i).Area==50 && perimetro(i)==35) || (area(i).Area==50 && perimetro(i)==33)
|| (area(i).Area==78 && perimetro(i)==41)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %P
        handles.matriz_letras(posy,posx)=16;
        valor=inf;
        p=p+1;

      elseif (area(i).Area==107 && perimetro(i)==49) || (area(i).Area==155 &&
perimetro(i)==58)
        coordenadax=round(centroids(i,1)*alfa);
```

```
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %Q
        handles.matriz_letras(posy,posx)=17;
        valor=inf;
        q=q+1;

      elseif (area(i).Area==128 && perimetro(i)==63) || (area(i).Area==65 &&
perimetro(i)==43) || (area(i).Area==148 && perimetro(i)==74)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %R
        handles.matriz_letras(posy,posx)=18;
        valor=inf;
        r=r+1;

      elseif (area(i).Area==79 && perimetro(i)==70) || (area(i).Area==79 && perimetro(i)==75)
|| (area(i).Area==67 && perimetro(i)==66)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
```

```
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %S
        handles.matriz_letras(posy,posx)=19;
        valor=inf;
        s=s+1;

      elseif (area(i).Area==57 && perimetro(i)==46) || (area(i).Area==38 && perimetro(i)==38)
|| (area(i).Area==85 && perimetro(i)==52)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %T
        handles.matriz_letras(posy,posx)=20;
        valor=inf;
        t=t+1;

      elseif (area(i).Area==62 && perimetro(i)==54) || (area(i).Area==59 && perimetro(i)==51)
||(area(i).Area==100 && perimetro(i)==67)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
```

```
        end
        %U
        handles.matriz_letras(posy,posx)=21;
        valor=inf;
        u=u+1;

      elseif (area(i).Area==61 && perimetro(i)==47) || (area(i).Area==70 && perimetro(i)==51)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %V
        handles.matriz_letras(posy,posx)=22;
        valor=inf;
        v=v+1;

      elseif (area(i).Area==84 && perimetro(i)==69)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %W
        handles.matriz_letras(posy,posx)=23;
        valor=inf;
        w=w+1;

      elseif (area(i).Area==102 && perimetro(i)==68) || (area(i).Area==128 &&
perimetro(i)==77)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);
```

```
        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %X
        handles.matriz_letras(posy,posx)=24;
        valor=inf;
        x=x+1;

    elseif (area(i).Area==48 && perimetro(i)==39)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %Y
        handles.matriz_letras(posy,posx)=25;
        valor=inf;
        y=y+1;

    elseif (area(i).Area==115 && perimetro(i)==76) ||(area(i).Area==61 && perimetro(i)==57)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
```

```
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %Z
        handles.matriz_letras(posy,posx)=26;
        valor=inf;
        z=z+1;

        elseif (area(i).Area==52 && perimetro(i)==43) || (area(i).Area==45 &&
perimetro(i)==40)
        coordenadax=round(centroids(i,1)*alfa);
        coordenaday=round(centroids(i,2)*alfa);

        for v1=1:length(vector_coordenadax)
            valornuevox=abs(coordenadax-handles.matriz_coordenadasx(1,v1));
            if valornuevox<valor
                posx=v1;
            end
            valor=valornuevox;
        end
        valor=100;
        for v2=1:length(vector_coordenaday)
            valornuevoy=abs(coordenaday-handles.matriz_coordenadasy(v2,1));
            if valornuevoy<valor
                posy=v2;
            end
            valor=valornuevoy;
        end
        %Caras
        handles.matriz_letras(posy,posx)=27;
        valor=inf;

    end
end

% %% 4. Extracción de las coordenadas de los cubos de la handles.palabra.
handles.caracteresconvertido=length(handles.palabra);
handles.caracteres=convertStringsToChars(handles.palabra);

% Para cuando no haya letras suficientes
a1=0; b1=0; c1=0; d1=0; e1=0; f1=0; g1=0; h1=0; i1=0; j1=0; k1=0; l1=0; m1=0;
n1=0; o1=0; p1=0; q1=0; r1=0; s1=0; t1=0; u1=0; v1=0; w1=0; x1=0; y1=0; z1=0;

%Extraemos las letras de la handles.palabra.
for i=1:length(handles.palabra)
    if handles.caracteres(i)=='a'
        handles.caracteresconvertido(i)=1;
        a1=a1+1;
        if a1>a
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='b'
        handles.caracteresconvertido(i)=2;
        b1=b1+1;
        if b1>b
```

```
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='c'
        handles.caracteresconvertido(i)=3;
        c1=c1+1;
        if c1>c
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='d'
        handles.caracteresconvertido(i)=4;
        d1=d1+1;
        if d1>d
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='e'
        handles.caracteresconvertido(i)=5;
        e1=e1+1;
        if e1>e
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='f'
        handles.caracteresconvertido(i)=6;
        f1=f1+1;
        if f1>f
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='g'
        handles.caracteresconvertido(i)=7;
        g1=g1+1;
        if g1>g
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='h'
        handles.caracteresconvertido(i)=8;
        h1=h1+1;
        if h1>a
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='i'
        handles.caracteresconvertido(i)=9;
        i1=i1+1;
        if i1>i
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='j'
        handles.caracteresconvertido(i)=10;
        j1=j1+1;
        if j1>j
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='k'
        handles.caracteresconvertido(i)=11;
        k1=k1+1;
        if k1>k
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='l'
        handles.caracteresconvertido(i)=12;
```

```
        l1=l1+1;
        if l1>l
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='m'
        handles.caracteresconvertido(i)=13;
        m1=m1+1;
        if m1>m
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='n'
        handles.caracteresconvertido(i)=14;
        n1=n1+1;
        if n1>n
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='o'
        handles.caracteresconvertido(i)=15;
        o1=o1+1;
        if o1>o
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='p'
        handles.caracteresconvertido(i)=16;
        p1=p1+1;
        if p1>p
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='q'
        handles.caracteresconvertido(i)=17;
        q1=q1+1;
        if q1>q
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='r'
        handles.caracteresconvertido(i)=18;
        r1=r1+1;
        if r1>r
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='s'
        handles.caracteresconvertido(i)=19;
        s1=s1+1;
        if s1>s
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='t'
        handles.caracteresconvertido(i)=20;
        t1=t1+1;
        if t1>t
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='u'
        handles.caracteresconvertido(i)=21;
        u1=u1+1;
        if u1>u
            handles.nosepuede=1;
        end
```

```
    elseif handles.caracteres(i)=='v'
        handles.caracteresconvertido(i)=22;
        v1=v1+1;
        if v1>v
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='w'
        handles.caracteresconvertido(i)=23;
        w1=w1+1;
        if w1>w
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='x'
        handles.caracteresconvertido(i)=24;
        x1=x1+1;
        if x1>x
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='y'
        handles.caracteresconvertido(i)=25;
        y1=y1+1;
        if y1>y
            handles.nosepuede=1;
        end
    elseif handles.caracteres(i)=='z'
        handles.caracteresconvertido(i)=26;
        z1=z1+1;
        if z1>z
            handles.nosepuede=1;
        end
    end
    i=i+1;
end

%Matrices de coordenadas adecuadas para detectar
%Para que funcione bien, hay que coger la fila o columna deseada.
handles.matriz_coordenadasx1=handles.matriz_coordenadasx(1,:);
%Para que funcione bien, hay que coger la fila o columna deseada.
handles.matriz_coordenadasy1=handles.matriz_coordenadasy(:,1);
[tx,ty]=size(handles.matriz_letras);
%Buscar letras.
%Parametro para borrar las letras bien.
detectada=0;
for i=1:length(handles.palabra)
    detectada=0;
    for x11=1:tx
        if detectada==0
            for x2=1:ty

                if handles.caracteresconvertido(i)==handles.matriz_letras(x11,x2)
                    handles.Posicion_x(i)=x2;
                    handles.Posicion_y(i)=x11;
                    handles.coordenadas_a_coger_x(i)=handles.matriz_coordenadasx1(x2);
                    handles.coordenadas_a_coger_y(i)=handles.matriz_coordenadasy1(x11);

                    %Se pone un cero en la matriz de letras, si el cubo se selecciona.
                    handles.matriz_letras(x11,x2)=0;
                    x2=ty;
```

```
                    x11=tx;
                    detectada=1;
                else
                    x2=x2+1;
                end

            end
            x11=x11+1;
        end
    end
    i=i+1;
end
```

III.XII. Color pilotos inicialización.

```
function [handles]=LedsInicio(handles)

%--------------------------------------------------------------------------
%Leds
%--------------------------------------------------------------------------

%F1
axes(handles.axes124);
background=imread('Verde.png');
imshow(background);
axis off;

%F2
axes(handles.axes127);
background=imread('Verde.png');
imshow(background);
axis off;

%F3
axes(handles.axes130);
background=imread('Verde.png');
imshow(background);
axis off;

%F4
axes(handles.axes133);
background=imread('Verde.png');
imshow(background);
axis off;

%F5
axes(handles.axes136);
background=imread('Verde.png');
imshow(background);
axis off;

%F6
axes(handles.axes139);
background=imread('Verde.png');
```

```
imshow(background);
axis off;

%F7
axes(handles.axes142);
background=imread('Verde.png');
imshow(background);
axis off;

%F8
axes(handles.axes145);
background=imread('Verde.png');
imshow(background);
axis off;

%F9
axes(handles.axes148);
background=imread('Verde.png');
imshow(background);
axis off;

%F10
axes(handles.axes151);
background=imread('Verde.png');
imshow(background);
axis off;

%F11
axes(handles.axes154);
background=imread('Verde.png');
imshow(background);
axis off;

%F12
axes(handles.axes157);
background=imread('Verde.png');
imshow(background);
axis off;

%F13
axes(handles.axes160);
background=imread('Verde.png');
imshow(background);
axis off;

%F14
axes(handles.axes163);
background=imread('Verde.png');
imshow(background);
axis off;

%F15
axes(handles.axes166);
background=imread('Verde.png');
imshow(background);
axis off;

%F16
```

```
axes(handles.axes169);
background=imread('Verde.png');
imshow(background);
axis off;
```

III.XIII. Mostrar pulsadores.

```
function [handles]=mostrar_pulsadores(handles)

%Mostrar las palabras de nuevo.
set(handles.pushbutton9,'visible','on');
set(handles.pushbutton41,'visible','on');
set(handles.pushbutton44,'visible','on');
set(handles.pushbutton47,'visible','on');
set(handles.pushbutton50,'visible','on');
set(handles.pushbutton53,'visible','on');
set(handles.pushbutton39,'visible','on');
set(handles.pushbutton42,'visible','on');
set(handles.pushbutton40,'visible','on');
set(handles.pushbutton43,'visible','on');
set(handles.pushbutton46,'visible','on');
set(handles.pushbutton49,'visible','on');
set(handles.pushbutton45,'visible','on');
set(handles.pushbutton48,'visible','on');
set(handles.pushbutton51,'visible','on');
set(handles.pushbutton54,'visible','on');
set(handles.pushbutton118,'visible','on');
```

III.XIV. Ocultar pulsadores.

```
function [handles]=ocultar_pulsadores(handles)

%Ocultar palabras.
set(handles.pushbutton9,'visible','off');
set(handles.pushbutton41,'visible','off');
set(handles.pushbutton44,'visible','off');
set(handles.pushbutton47,'visible','off');
set(handles.pushbutton50,'visible','off');
set(handles.pushbutton53,'visible','off');
set(handles.pushbutton39,'visible','off');
set(handles.pushbutton42,'visible','off');
set(handles.pushbutton40,'visible','off');
set(handles.pushbutton43,'visible','off');
set(handles.pushbutton46,'visible','off');
set(handles.pushbutton49,'visible','off');
set(handles.pushbutton45,'visible','off');
set(handles.pushbutton48,'visible','off');
set(handles.pushbutton51,'visible','off');
set(handles.pushbutton54,'visible','off');
set(handles.pushbutton118,'visible','off');
```

III.XV. Acondicionar palabra escogida.

```
function [handles]=Palabra_Escogida(handles)

%Convertimos la palabra en caracteres.
handles.caracteres=convertStringsToChars(handles.palabra);
```

III.XVI. Palabras del panel palabras.

```
function [handles]=Palabras(handles)

handles.p1p='televisor';
handles.p2p='camara';
handles.p3p='coche';
handles.p4p='upv';
handles.p5p='lampara';
handles.p6p='robot';
handles.p7p='cable';
handles.p8p='estufa';
handles.p9p='master';
handles.p10p='plano';
handles.p11p='piezas';
handles.p12p='escuela';
handles.p13p='ingeniero';
handles.p14p='motor';
handles.p15p='teclado';
handles.p16p='movil';
```

III.XVII. Panel plataforma grande.

```
function [handles]=plataforma_grande(handles)

%Si el ultimo caracter es una cara, no es un caracter y se muestra en la
% funcion contento_triste, aquí solo se muestran caracteres.

if handles.contento==1 || handles.triste==1
    longitud1=length(handles.Posicion_x)-1;
else
    longitud1=length(handles.Posicion_x);
end

%Se juntan las coordenadas y se crea el texto.
for i=1:longitud1
    handles.Posicion_x_letras=int2str(handles.Posicion_y(i));
    handles.Posicion_y_letras=int2str(handles.Posicion_x(i));
    Posiciones=strcat(handles.Posicion_x_letras,handles.Posicion_y_letras);
    handles.caracteres1=upper(handles.caracteres);

    if Posiciones=='11'
```

```
        set(handles.text511,'string',handles.caracteres1(i));
        set(handles.text511,'visible','on');
    elseif Posiciones=='12'
        set(handles.text512,'string',handles.caracteres1(i));
        set(handles.text512,'visible','on');
    elseif Posiciones=='13'
        set(handles.text513,'string',handles.caracteres1(i));
        set(handles.text513,'visible','on');
    elseif Posiciones=='14'
        set(handles.text514,'string',handles.caracteres1(i));
        set(handles.text514,'visible','on');
    elseif Posiciones=='15'
        set(handles.text515,'string',handles.caracteres1(i));
        set(handles.text515,'visible','on');
    elseif Posiciones=='16'
        set(handles.text516,'string',handles.caracteres1(i));
        set(handles.text516,'visible','on');
    elseif Posiciones=='17'
        set(handles.text517,'string',handles.caracteres1(i));
        set(handles.text517,'visible','on');
    elseif Posiciones=='18'
        set(handles.text518,'string',handles.caracteres1(i));
        set(handles.text518,'visible','on');
    elseif Posiciones=='19'
        set(handles.text519,'string',handles.caracteres1(i));
        set(handles.text519,'visible','on');


    elseif Posiciones=='21'
        set(handles.text521,'string',handles.caracteres1(i));
        set(handles.text521,'visible','on');
    elseif Posiciones=='22'
        set(handles.text522,'string',handles.caracteres1(i));
        set(handles.text522,'visible','on');
    elseif Posiciones=='23'
        set(handles.text523,'string',handles.caracteres1(i));
        set(handles.text523,'visible','on');
    elseif Posiciones=='24'
        set(handles.text524,'string',handles.caracteres1(i));
        set(handles.text524,'visible','on');
    elseif Posiciones=='25'
        set(handles.text525,'string',handles.caracteres1(i));
        set(handles.text525,'visible','on');
    elseif Posiciones=='26'
        set(handles.text526,'string',handles.caracteres1(i));
        set(handles.text526,'visible','on');
    elseif Posiciones=='27'
        set(handles.text527,'string',handles.caracteres1(i));
        set(handles.text527,'visible','on');
    elseif Posiciones=='28'
        set(handles.text528,'string',handles.caracteres1(i));
        set(handles.text528,'visible','on');
    elseif Posiciones=='29'
        set(handles.text529,'string',handles.caracteres1(i));
        set(handles.text529,'visible','on');


    elseif Posiciones=='31'
```

```
        set(handles.text531,'string',handles.caracteres1(i));
        set(handles.text531,'visible','on');
    elseif Posiciones=='32'
        set(handles.text532,'string',handles.caracteres1(i));
        set(handles.text532,'visible','on');
    elseif Posiciones=='33'
        set(handles.text533,'string',handles.caracteres1(i));
        set(handles.text533,'visible','on');
    elseif Posiciones=='34'
        set(handles.text534,'string',handles.caracteres1(i));
        set(handles.text534,'visible','on');
    elseif Posiciones=='35'
        set(handles.text535,'string',handles.caracteres1(i));
        set(handles.text535,'visible','on');
    elseif Posiciones=='36'
        set(handles.text536,'string',handles.caracteres1(i));
        set(handles.text536,'visible','on');
    elseif Posiciones=='37'
        set(handles.text537,'string',handles.caracteres1(i));
        set(handles.text537,'visible','on');
    elseif Posiciones=='38'
        set(handles.text538,'string',handles.caracteres1(i));
        set(handles.text538,'visible','on');
    elseif Posiciones=='39'
        set(handles.text539,'string',handles.caracteres1(i));
        set(handles.text539,'visible','on');


    elseif Posiciones=='41'
        set(handles.text541,'string',handles.caracteres1(i));
        set(handles.text541,'visible','on');
    elseif Posiciones=='42'
        set(handles.text542,'string',handles.caracteres1(i));
        set(handles.text542,'visible','on');
    elseif Posiciones=='43'
        set(handles.text543,'string',handles.caracteres1(i));
        set(handles.text543,'visible','on');
    elseif Posiciones=='44'
        set(handles.text544,'string',handles.caracteres1(i));
        set(handles.text544,'visible','on');
    elseif Posiciones=='45'
        set(handles.text545,'string',handles.caracteres1(i));
        set(handles.text545,'visible','on');
    elseif Posiciones=='46'
        set(handles.text546,'string',handles.caracteres1(i));
        set(handles.text546,'visible','on');
    elseif Posiciones=='47'
        set(handles.text547,'string',handles.caracteres1(i));
        set(handles.text547,'visible','on');
    elseif Posiciones=='48'
        set(handles.text548,'string',handles.caracteres1(i));
        set(handles.text548,'visible','on');
    elseif Posiciones=='49'
        set(handles.text549,'string',handles.caracteres1(i));
        set(handles.text549,'visible','on');
```

```
    elseif Posiciones=='51'
        set(handles.text551,'string',handles.caracteres1(i));
        set(handles.text551,'visible','on');
    elseif Posiciones=='52'
        set(handles.text552,'string',handles.caracteres1(i));
        set(handles.text552,'visible','on');
    elseif Posiciones=='53'
        set(handles.text553,'string',handles.caracteres1(i));
        set(handles.text553,'visible','on');
    elseif Posiciones=='54'
        set(handles.text554,'string',handles.caracteres1(i));
        set(handles.text554,'visible','on');
    elseif Posiciones=='55'
        set(handles.text555,'string',handles.caracteres1(i));
        set(handles.text555,'visible','on');
    elseif Posiciones=='56'
        set(handles.text556,'string',handles.caracteres1(i));
        set(handles.text556,'visible','on');
    elseif Posiciones=='57'
        set(handles.text557,'string',handles.caracteres1(i));
        set(handles.text557,'visible','on');
    elseif Posiciones=='58'
        set(handles.text558,'string',handles.caracteres1(i));
        set(handles.text558,'visible','on');
    elseif Posiciones=='59'
        set(handles.text559,'string',handles.caracteres1(i));
        set(handles.text559,'visible','on');


    elseif Posiciones=='61'
        set(handles.text561,'string',handles.caracteres1(i));
        set(handles.text561,'visible','on');
    elseif Posiciones=='62'
        set(handles.text562,'string',handles.caracteres1(i));
        set(handles.text562,'visible','on');
    elseif Posiciones=='63'
        set(handles.text563,'string',handles.caracteres1(i));
        set(handles.text563,'visible','on');
    elseif Posiciones=='64'
        set(handles.text564,'string',handles.caracteres1(i));
        set(handles.text564,'visible','on');
    elseif Posiciones=='65'
        set(handles.text565,'string',handles.caracteres1(i));
        set(handles.text565,'visible','on');
    elseif Posiciones=='66'
        set(handles.text566,'string',handles.caracteres1(i));
        set(handles.text566,'visible','on');
    elseif Posiciones=='67'
        set(handles.text567,'string',handles.caracteres1(i));
        set(handles.text567,'visible','on');
    elseif Posiciones=='68'
        set(handles.text568,'string',handles.caracteres1(i));
        set(handles.text568,'visible','on');
    elseif Posiciones=='69'
        set(handles.text569,'string',handles.caracteres1(i));
        set(handles.text569,'visible','on');
```

```
    elseif Posiciones=='71'
        set(handles.text571,'string',handles.caracteres1(i));
        set(handles.text571,'visible','on');
    elseif Posiciones=='72'
        set(handles.text572,'string',handles.caracteres1(i));
        set(handles.text572,'visible','on');
    elseif Posiciones=='73'
        set(handles.text573,'string',handles.caracteres1(i));
        set(handles.text573,'visible','on');
    elseif Posiciones=='74'
        set(handles.text574,'string',handles.caracteres1(i));
        set(handles.text574,'visible','on');
    elseif Posiciones=='75'
        set(handles.text575,'string',handles.caracteres1(i));
        set(handles.text575,'visible','on');
    elseif Posiciones=='76'
        set(handles.text576,'string',handles.caracteres1(i));
        set(handles.text576,'visible','on');
    elseif Posiciones=='77'
        set(handles.text577,'string',handles.caracteres1(i));
        set(handles.text577,'visible','on');
    elseif Posiciones=='78'
        set(handles.text578,'string',handles.caracteres1(i));
        set(handles.text578,'visible','on');
    elseif Posiciones=='79'
        set(handles.text579,'string',handles.caracteres1(i));
        set(handles.text579,'visible','on');
    end

    i=i+1;
end
```

III.XVIII. Palabra no disponible.

```
function [handles]=plataforma_grande_mala(handles)

%TEXTO NO.
set(handles.text534,'string','N');
set(handles.text534,'visible','on');
set(handles.text535,'string','O');
set(handles.text535,'visible','on');

%TEXTO POSIBLE.
set(handles.text542,'string','P');
set(handles.text542,'visible','on');

set(handles.text543,'string','O');
set(handles.text543,'visible','on');

set(handles.text544,'string','S');
set(handles.text544,'visible','on');

set(handles.text545,'string','I');
set(handles.text545,'visible','on');
```

```
set(handles.text546,'string','B');
set(handles.text546,'visible','on');

set(handles.text547,'string','L');
set(handles.text547,'visible','on');

set(handles.text548,'string','E');
set(handles.text548,'visible','on');
```

III.XIX. Panel plataforma pequeña.

```
function [handles]=plataforma_pequena(handles)

%Se centra y se forma la palabra en la plataforma pequeña.
if length(handles.Posicion_x)==6 && handles.p2==1

    set(handles.text612,'string',handles.caracteres1(1));
    set(handles.text612,'visible','on');

    set(handles.text613,'string',handles.caracteres1(2));
    set(handles.text613,'visible','on');

    set(handles.text614,'string',handles.caracteres1(3));
    set(handles.text614,'visible','on');

    set(handles.text615,'string','/');
    set(handles.text615,'visible','on');

    set(handles.text616,'string',handles.caracteres1(4));
    set(handles.text616,'visible','on');

    set(handles.text617,'string',handles.caracteres1(5));
    set(handles.text617,'visible','on');

    set(handles.text618,'string',handles.caracteres1(6));
    set(handles.text618,'visible','on');

elseif length(handles.Posicion_x)==1

    set(handles.text615,'string',handles.caracteres1(1));
    set(handles.text615,'visible','on');


elseif length(handles.Posicion_x)==2

    set(handles.text614,'string',handles.caracteres1(1));
    set(handles.text614,'visible','on');

    set(handles.text615,'string',handles.caracteres1(2));
    set(handles.text615,'visible','on');

elseif length(handles.Posicion_x)==3
```

```
    set(handles.text614,'string',handles.caracteres1(1));
    set(handles.text614,'visible','on');

    set(handles.text615,'string',handles.caracteres1(2));
    set(handles.text615,'visible','on');

    set(handles.text616,'string',handles.caracteres1(3));
    set(handles.text616,'visible','on');

elseif length(handles.Posicion_x)==4

    set(handles.text613,'string',handles.caracteres1(1));
    set(handles.text613,'visible','on');

    set(handles.text614,'string',handles.caracteres1(2));
    set(handles.text614,'visible','on');

    set(handles.text615,'string',handles.caracteres1(3));
    set(handles.text615,'visible','on');

    set(handles.text616,'string',handles.caracteres1(4));
    set(handles.text616,'visible','on');

elseif length(handles.Posicion_x)==5
    set(handles.text613,'string',handles.caracteres1(1));
    set(handles.text613,'visible','on');

    set(handles.text614,'string',handles.caracteres1(2));
    set(handles.text614,'visible','on');

    set(handles.text615,'string',handles.caracteres1(3));
    set(handles.text615,'visible','on');

    set(handles.text616,'string',handles.caracteres1(4));
    set(handles.text616,'visible','on');

    %Si el ultimo caracter es una cara, se carga la imagen.
    if handles.contento==0 && handles.triste==0

        set(handles.text617,'string',handles.caracteres1(5));
        set(handles.text617,'visible','on');

    else

        if handles.contento==1
            set(handles.axes305,'visible','on');
            axes(handles.axes305);
            background=imread('sonrie.jpg');
            imshow(background);
            axis off;
        end

        if handles.triste==1
            set(handles.axes305,'visible','on');
            axes(handles.axes305);
            background=imread('triste.jpg');
            imshow(background);
            axis off;
```

```
        end

    end

elseif length(handles.Posicion_x)==6

    set(handles.text612,'string',handles.caracteres1(1));
    set(handles.text612,'visible','on');

    set(handles.text613,'string',handles.caracteres1(2));
    set(handles.text613,'visible','on');

    set(handles.text614,'string',handles.caracteres1(3));
    set(handles.text614,'visible','on');

    set(handles.text615,'string',handles.caracteres1(4));
    set(handles.text615,'visible','on');

    set(handles.text616,'string',handles.caracteres1(5));
    set(handles.text616,'visible','on');

    %Si el ultimo caracter es una cara, se carga la imagen.
    if handles.contento==0 && handles.triste==0

        set(handles.text617,'string',handles.caracteres1(6));
        set(handles.text617,'visible','on');

    else

        if handles.contento==1
            set(handles.axes305,'visible','on');
            axes(handles.axes305);
            background=imread('sonrie.jpg');
            imshow(background);
            axis off;
        end

        if handles.triste==1
            set(handles.axes305,'visible','on');
            axes(handles.axes305);
            background=imread('triste.jpg');
            imshow(background);
            axis off;
        end

    end

elseif length(handles.Posicion_x)==7

    set(handles.text612,'string',handles.caracteres1(1));
    set(handles.text612,'visible','on');

    set(handles.text613,'string',handles.caracteres1(2));
    set(handles.text613,'visible','on');

    set(handles.text614,'string',handles.caracteres1(3));
    set(handles.text614,'visible','on');
```

```
    set(handles.text615,'string',handles.caracteres1(4));
    set(handles.text615,'visible','on');

    set(handles.text616,'string',handles.caracteres1(5));
    set(handles.text616,'visible','on');

    set(handles.text617,'string',handles.caracteres1(6));
    set(handles.text617,'visible','on');

    %Si el ultimo caracter es una cara, se carga la imagen.
    if handles.contento==0 && handles.triste==0

        set(handles.text618,'string',handles.caracteres1(7));
        set(handles.text618,'visible','on');

    else

        if handles.contento==1
            set(handles.axes306,'visible','on');
            axes(handles.axes306);
            background=imread('sonrie.jpg');
            imshow(background);
            axis off;
        end

        if handles.triste==1
            set(handles.axes306,'visible','on');
            axes(handles.axes306);
            background=imread('triste.jpg');
            imshow(background);
            axis off;
        end

    end

elseif length(handles.Posicion_x)==8

    set(handles.text611,'string',handles.caracteres1(1));
    set(handles.text611,'visible','on');

    set(handles.text612,'string',handles.caracteres1(2));
    set(handles.text612,'visible','on');

    set(handles.text613,'string',handles.caracteres1(3));
    set(handles.text613,'visible','on');

    set(handles.text614,'string',handles.caracteres1(4));
    set(handles.text614,'visible','on');

    set(handles.text615,'string',handles.caracteres1(5));
    set(handles.text615,'visible','on');

    set(handles.text616,'string',handles.caracteres1(6));
    set(handles.text616,'visible','on');

    set(handles.text617,'string',handles.caracteres1(7));
    set(handles.text617,'visible','on');
```

```
    %Si el ultimo caracter es una cara, se carga la imagen.
    if handles.contento==0 && handles.triste==0

        set(handles.text618,'string',handles.caracteres1(8));
        set(handles.text618,'visible','on');

    else

        if handles.contento==1
            set(handles.axes306,'visible','on');
            axes(handles.axes306);
            background=imread('sonrie.jpg');
            imshow(background);
            axis off;
        end

        if handles.triste==1
            set(handles.axes306,'visible','on');
            axes(handles.axes306);
            background=imread('triste.jpg');
            imshow(background);
            axis off;
        end

    end

elseif length(handles.Posicion_x)==9

    set(handles.text611,'string',handles.caracteres1(1));
    set(handles.text611,'visible','on');

    set(handles.text612,'string',handles.caracteres1(2));
    set(handles.text612,'visible','on');

    set(handles.text613,'string',handles.caracteres1(3));
    set(handles.text613,'visible','on');

    set(handles.text614,'string',handles.caracteres1(4));
    set(handles.text614,'visible','on');

    set(handles.text615,'string',handles.caracteres1(5));
    set(handles.text615,'visible','on');

    set(handles.text616,'string',handles.caracteres1(6));
    set(handles.text616,'visible','on');

    set(handles.text617,'string',handles.caracteres1(7));
    set(handles.text617,'visible','on');

    set(handles.text618,'string',handles.caracteres1(8));
    set(handles.text618,'visible','on');

    %Si el ultimo caracter es una cara, se carga la imagen.
    if handles.contento==0 && handles.triste==0

        set(handles.text619,'string',handles.caracteres1(9));
        set(handles.text619,'visible','on');
```

```
    else

        if handles.contento==1
            set(handles.axes307,'visible','on');
            axes(handles.axes307);
            background=imread('sonrie.jpg');
            imshow(background);
            axis off;
        end

        if handles.triste==1
            set(handles.axes307,'visible','on');
            axes(handles.axes307);
            background=imread('triste.jpg');
            imshow(background);
            axis off;
        end

    end

end

handles.p2=0;
```

III.XX. Ocultar textos panel plataforma grande.

```
function [handles]=textos_grande(handles)

set(handles.text511,'visible','off');
set(handles.text512,'visible','off');
set(handles.text513,'visible','off');
set(handles.text514,'visible','off');
set(handles.text515,'visible','off');
set(handles.text516,'visible','off');
set(handles.text517,'visible','off');
set(handles.text518,'visible','off');
set(handles.text519,'visible','off');
set(handles.text521,'visible','off');
set(handles.text522,'visible','off');
set(handles.text523,'visible','off');
set(handles.text524,'visible','off');
set(handles.text525,'visible','off');
set(handles.text526,'visible','off');
set(handles.text527,'visible','off');
set(handles.text528,'visible','off');
set(handles.text529,'visible','off');
set(handles.text531,'visible','off');
set(handles.text532,'visible','off');
set(handles.text533,'visible','off');
set(handles.text534,'visible','off');
set(handles.text535,'visible','off');
set(handles.text536,'visible','off');
set(handles.text537,'visible','off');
set(handles.text538,'visible','off');
```

```
set(handles.text539,'visible','off');
set(handles.text541,'visible','off');
set(handles.text542,'visible','off');
set(handles.text543,'visible','off');
set(handles.text544,'visible','off');
set(handles.text545,'visible','off');
set(handles.text546,'visible','off');
set(handles.text547,'visible','off');
set(handles.text548,'visible','off');
set(handles.text549,'visible','off');
set(handles.text551,'visible','off');
set(handles.text552,'visible','off');
set(handles.text553,'visible','off');
set(handles.text554,'visible','off');
set(handles.text555,'visible','off');
set(handles.text556,'visible','off');
set(handles.text557,'visible','off');
set(handles.text558,'visible','off');
set(handles.text559,'visible','off');
set(handles.text561,'visible','off');
set(handles.text562,'visible','off');
set(handles.text563,'visible','off');
set(handles.text564,'visible','off');
set(handles.text565,'visible','off');
set(handles.text566,'visible','off');
set(handles.text567,'visible','off');
set(handles.text568,'visible','off');
set(handles.text569,'visible','off');
set(handles.text571,'visible','off');
set(handles.text572,'visible','off');
set(handles.text573,'visible','off');
set(handles.text574,'visible','off');
set(handles.text575,'visible','off');
set(handles.text576,'visible','off');
set(handles.text577,'visible','off');
set(handles.text578,'visible','off');
set(handles.text579,'visible','off');

%Caras plataforma.
set(handles.axes303,'visible','off');
set(handles.axes304,'visible','off');
```

III.XXI. Ocultar textos panel plataforma pequeña.

```
function [handles]=textos_pequeno(handles)

set(handles.text611,'visible','off');
set(handles.text612,'visible','off');
set(handles.text613,'visible','off');
set(handles.text614,'visible','off');
set(handles.text615,'visible','off');
set(handles.text616,'visible','off');
set(handles.text617,'visible','off');
set(handles.text618,'visible','off');
```

```
set(handles.text619,'visible','off');

%Caras plataforma.
set(handles.axes305,'visible','off');
set(handles.axes306,'visible','off');
set(handles.axes307,'visible','off');
```

III.XXII. Velocidad robot.

```
function [handles]=Velocidad_Robot(handles)

%Obtengo la velocidad del slider.
handles.velocidad=get(handles.text894,'string');
handles.velocidad=str2double(handles.velocidad);
handles.velocidad=round(handles.velocidad);

%Convierto la velocidad del slider en velocidad del robot.
if handles.velocidad==0
    handles.velocidad1=1500;
elseif handles.velocidad==100
    handles.velocidad1=1000;
else
    handles.velocidad1=round(-5*handles.velocidad+1500);
end

%Adapto el valor de velocidad a byte 0-255 (167-250).
handles.velocidad1=round(handles.velocidad1/6);
```

III.XXIII. Imágenes panel palabras.

```
function [handles]=VerPalabras(handles)

%--------------------------------------------------------------------------
%Dibujos
%--------------------------------------------------------------------------
axes(handles.axes39);
background=imread('Televisor.jpg');
imshow(background);
axis off;

axes(handles.axes71);
background=imread('upv_ehu.jpg');
imshow(background);
axis off;

axes(handles.axes74);
background=imread('Cable.jpg');
imshow(background);
axis off;

axes(handles.axes77);
```

```
background=imread('Plano.jpg');
imshow(background);
axis off;

axes(handles.axes80);
background=imread('Ingeniero.jpg');
imshow(background);
axis off;

axes(handles.axes83);
background=imread('Teclado.jpg');
imshow(background);
axis off;

axes(handles.axes69);
background=imread('Camara.jpg');
imshow(background);
axis off;

axes(handles.axes72);
background=imread('Lampara.jpg');
imshow(background);
axis off;

axes(handles.axes75);
background=imread('Estufa.jpg');
imshow(background);
axis off;

axes(handles.axes78);
background=imread('Piezas.jpg');
imshow(background);
axis off;

axes(handles.axes81);
background=imread('Motor.jpg');
imshow(background);
axis off;

axes(handles.axes84);
background=imread('Movil.jpg');
imshow(background);
axis off;

axes(handles.axes70);
background=imread('Coche.jpg');
imshow(background);
axis off;

axes(handles.axes73);
background=imread('Robot.jpg');
imshow(background);
axis off;

axes(handles.axes76);
background=imread('Master.jpg');
imshow(background);
axis off;
```

```
axes(handles.axes79);
background=imread('Escuela.jpg');
imshow(background);
axis off;

%--------------------------------------------------------------------------
%Plataformas
%--------------------------------------------------------------------------
axes(handles.axes176);
background=imread('plataforma_grande.png');
imshow(background);
axis off;

axes(handles.axes296);
background=imread('plataforma_peque.png');
imshow(background);
axis off;
```

Anexo IV:
CÓDIGO PYTHON

IV. Código Python.

A continuación, se muestra el Código de Python que permite la comunicación entre el PLC y PC.

```
import socket

def add_numbers_2(n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16, n17, n18, n19, n20, n21, n22, n23, n24, n25, n26, n27, n28):

    # Función que manda al PLC el vector de información y devuelve la confirmación de finalización.

    import socket

    # Create a TCP/IP socket
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    # Connect the local socket to the server socket.
    server_address = ('192.168.137.100', 8080)
    sock.connect(server_address)

    # Se envian los datos.
    numerito = bytes([n1,n2,n3, n4,n5,n6, n7,n8,n9, n10,n11,n12, n13,n14,n15, n16,n17,n18, n19,n20,n21, n22,n23,n24, n25,n26,n27, n28])
    sock.send(numerito)

    buffer_size = 64
    # Se recibe la confirmación del PLC.
    data = sock.recv(buffer_size)

    # Se cierra la comunicación.
    sock.close()
```

Anexo V:

MANUAL DE USUARIO

V. Manual de usuario.

En este capítulo se van a exponer, por un lado, los pasos necesarios que hay que realizar para la correcta puesta en marcha y detención del robot y, por otro lado, el manual de usuario de la interfaz gráfica desarrollada en este proyecto.

V.I. Arranque del robot.

Para realizar el arranque del robot, hay que realizar los pasos que se explican a continuación:

1. Subir los interruptores diferencial y magnetotérmico situados en la parte superior izquierda del cuadro eléctrico.
2. Esperar 5 segundos y pulsar el botón verde de la botonera lateral.
3. Esperar unos segundos a que el robot se inicialice. Se oirá un “clack” que indica que los frenos del motor se han soltado y tras colocar el selector de la botonera en la posición superior el robot irá a su posición inicial. Mientras el robot se está inicializando la luz verde parpadeará. Una vez inicializado la luz verde quedará fija.
4. Encender el compresor y esperar a que alcance la presión necesaria.
5. Con la luz verde iluminada fija, colocar el selector en la posición inferior y el robot estará listo para funcionar.

V.II. Apagado del robot.

Para apagar el robot se recomienda esperar a que se haya terminado de formar una palabra completa, pero se puede apagar sin necesidad de ello. Tan solo hay que pulsar el botón rojo de la botonera lateral de activación y colocar el selector en la posición intermedia (pausa).

V.III. Seguridad del robot y operario.

Cuando se quiera acceder al habitáculo del robot abriendo alguna de las puertas el robot se detendrá inmediatamente, mostrando en los displays de los servodrivers el error “St” y encendiéndose el piloto rojo. Una vez cerradas las puertas para arrancar el robot se siguen los pasos del arranque del robot desde el paso 3.

El selector de tres posiciones cuenta con una posición central que es la posición de pausa y para lanzar la aplicación es necesario colocar el selector a la posición inferior. Si durante el movimiento del robot el selector se coloca en la posición intermedia o superior, el robot

completará el movimiento que está realizando, se desplazará hasta la posición de reposo y se parará.

Si se pulsa el botón rojo de paro o la seta de emergencia se cortará la alimentación a los servomotores, el robot se detendrá inmediatamente, se mostrará en los displays de los servodrivers el error “13” y se encenderá el piloto rojo.

Después de una parada de emergencia pulsando la seta de emergencia, para volver a arrancar el robot, hay que soltar la seta de emergencia y pulsar el botón verde para dar otra vez alimentación. Si solo se ha pulsado el botón rojo de paro simplemente habrá que pulsar el pulsador verde de arranque y seguir los pasos del arranque del robot.

Siempre que ocurra cualquiera de los casos de este apartado, una vez restablecidas las condiciones de funcionamiento habrá que pasar el selector a la posición de inicialización (posición superior) y proceder con los pasos del apartado “Arranque del robot”.

V.IV. Instalación de la aplicación.

Para instalar la aplicación, se proporciona un instalador dentro de la carpeta del programa como muestra la figura V.1.

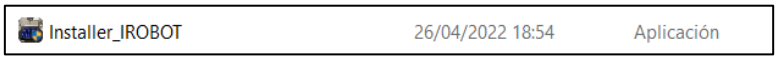


Figura V.1. Instalador de la aplicación.

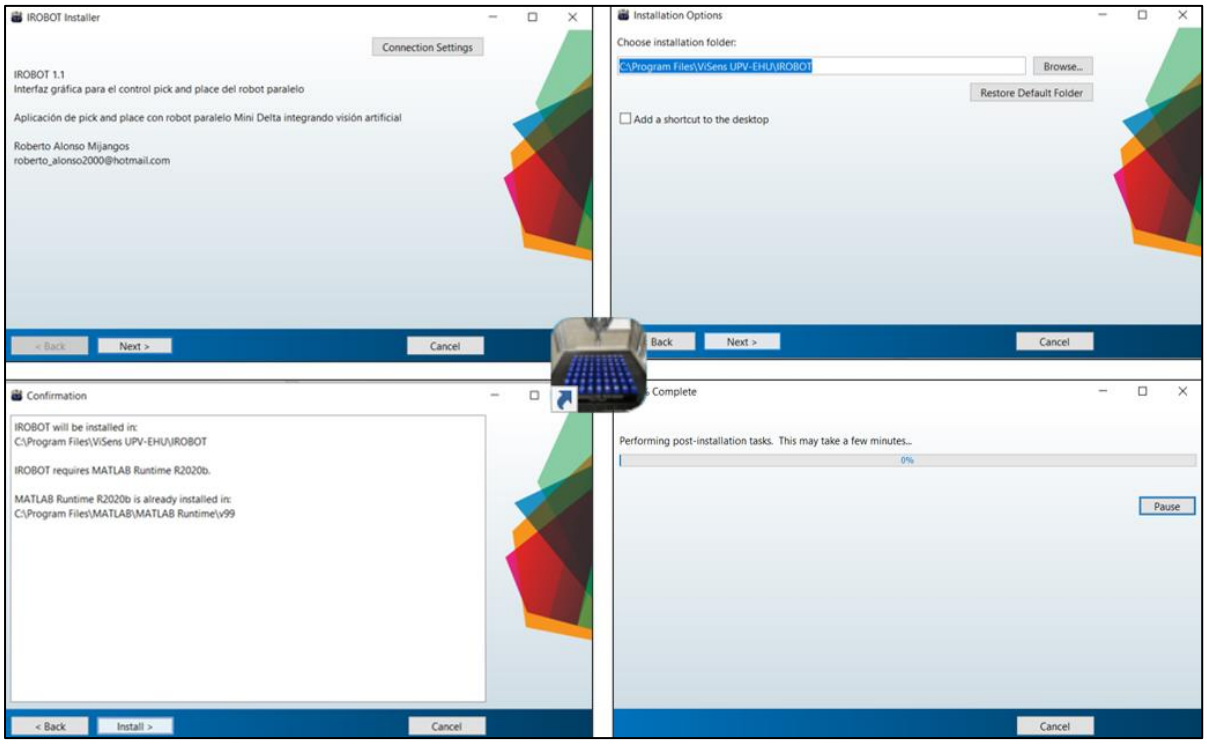


Figura V.2. Pasos de la instalación de la aplicación.

Para instalarlo aparecerá una ventana con la información y versión de la aplicación (ver Figura V.2a). Después se elige el directo de instalación (ver Figura V.2b) y comenzará la instalación (ver Figura V.2c). Al finalizar, el programa ya se encuentra instalado y se genera un acceso directo en el escritorio que permite abrir la aplicación (ver Figura V.2d).

La carpeta del programa cuenta con tres elementos:

- Un instalador. Un ejecutable que instala el programa en el ordenador del cliente.
- Un ejecutable para realizar pruebas y modificaciones en el código que no afectan a la versión final.
- La aplicación propiamente dicha.

Para que la aplicación funcione correctamente, además de tener la aplicación instalada es necesario que el cableado este correctamente conexionado. Por un lado, es necesario conectar la webcam por USB al equipo donde se va a correr la interfaz y, por otro lado, es imprescindible que se conecte un cable de red de Ethernet entre el PC y el PLC para que se pueda producir la comunicación TCP.

Con la aplicación instalada y el robot arrancado, tan solo es necesario abrir la aplicación desde el acceso directo del escritorio para abrir la interfaz gráfica.

V.V. Manual de usuario de la interfaz gráfica.

Para arrancar la interfaz una vez abierta la aplicación hay que pulsar en el botón verde ‘Start’ de la figura V.3.

Figura V.3. Botón de arranque de la interfaz.

En la parte superior central se encuentran cinco interruptores. El primero permite parar la interfaz, cuando la interfaz esté arrancada, aparecerá en color rojo con la palabra ‘Stop’ para parar la interfaz y se pondrá en color verde cuando la interfaz está parada y se quiera arrancar.

El segundo pulsador, que aparece en color negro permite cerrar la interfaz, pero este botón se encuentra deshabilitado si la interfaz está en marcha. Si se quiere cerrar la interfaz, primero hay que pararla, pulsando en ‘stop’ y posteriormente pulsa en ‘cerrar programa’.

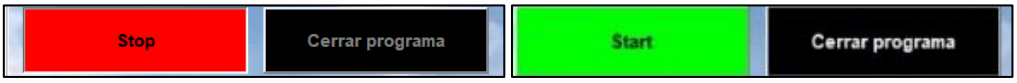


Figura V.4. Pulsadores de arranque y paro de la interfaz.

La interfaz desarrollada cuenta con seis paneles:

- Panel de selección de idioma.
- Panel de palabras predefinidas.
- Panel de comprobar palabra.
- Panel de la plataforma grande.
- Panel de la plataforma pequeña.
- Panel del robot.

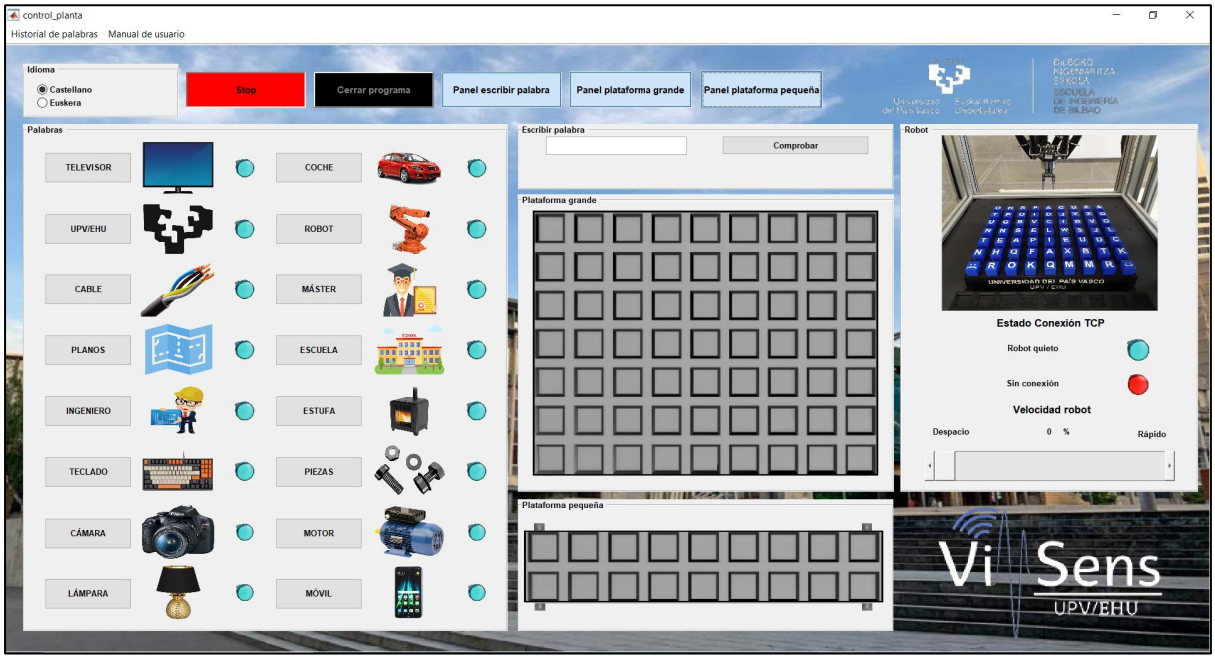


Figura V.5. Distribución de los paneles de la interfaz.

V.V.I. Panel de selección de idioma.

El panel de selección de idioma se encuentra en la parte superior izquierda y permite cambiar el idioma de la interfaz entre castellano y euskera.

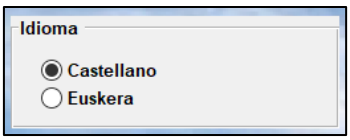


Figura V.6. Panel de selección de idioma.

V.V.II. Panel de palabras.

El panel de ‘Palabras’ contiene 16 pulsadores con 16 palabras que el robot puede escribir. Cada una de estas palabras tiene a su lado derecho una imagen de la palabra y unos pilotos verdes que indican que la palabra está disponible para escribirla.

Apretando cualquiera de estos pulsadores, el robot comenzará a escribir la palabra con el texto del pulsador apretado. Los pilotos tienen tres posibles estados como muestra la figura V.7:

- Piloto verde. Indica que la palabra asociada a ese piloto está disponible para escribirse.
- Piloto amarillo. Indica que la palabra se encuentra en proceso de formación por el robot. Este piloto aparece al pulsarse una palabra para escribir.
- Piloto rojo. Indica que la palabra que no está disponible para escribir. Cuando se pulsa una palabra para escribir, todas las demás se bloquean hasta que se termine de formar y recoger la seleccionada.

Figura V.7. Pilotos de la interfaz.

En este panel se encuentran las palabras predefinidas que el robot siempre podrá escribir con la distribución de letras de la plataforma. En la figura V.8, se muestra el panel de palabra, donde en la parte izquierda se muestra el panel por defecto y en la parte izquierda se muestra el panel cuando se pulsa una palabra para escribir, en este caso 'Televisor'.

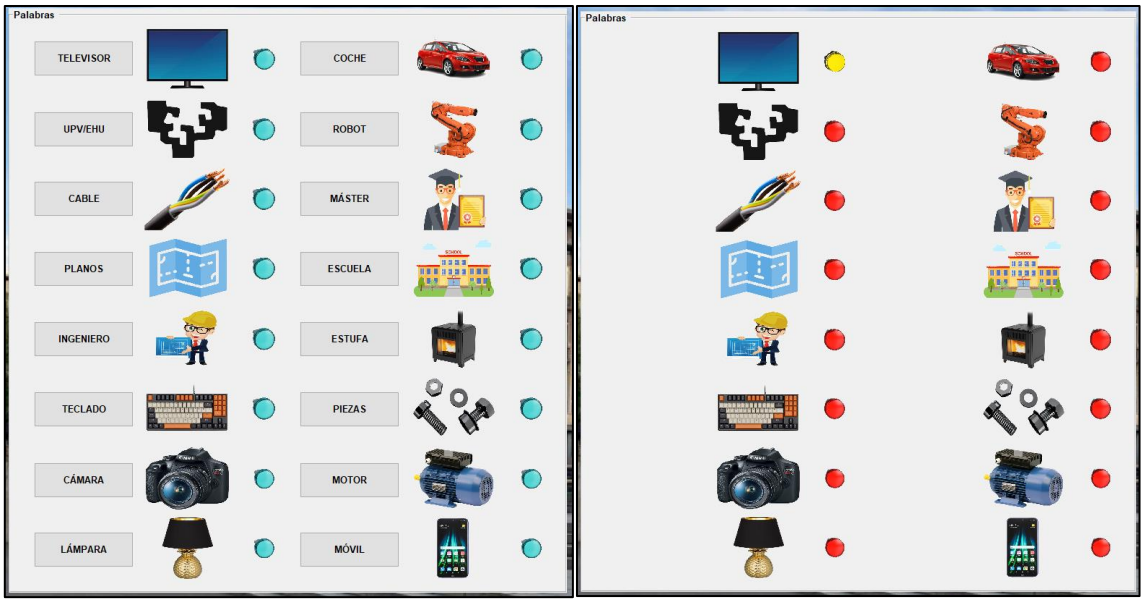


Figura V.8. Panel de palabras.

V.V.III. Panel de comprobar palabra.

Si se quiere que el robot escriba una palabra fuera de las que están predefinidas, hay que utilizar este panel que permite comprobar si es posible escribir la palabra deseada. Dentro de

este panel se habilita un recuadro de texto, donde el usuario puede escribir cualquier palabra que quiera que el robot escriba. Posteriormente es necesario pulsar el botón de comprobar para verificar si esa palabra se puede escribir y si es posible, se comenzará a ello.

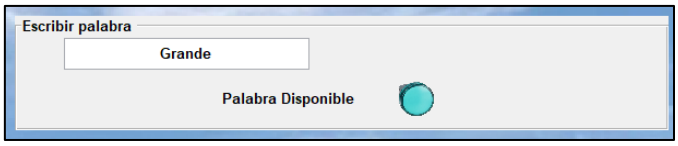


Figura V.9. Palabra disponible para escribir.

En ocasiones no se pueden escribir todas las palabras, porque depende de la cantidad de letras de cada tipo que se hayan colocado, si la palabra tiene más de 9 letras o si contiene la letra ‘ñ’.

En estos casos, al apretar el botón de comprobar, aparecerá un piloto rojo y un mensaje de que la palabra no se puede escribir y pasados 3 segundos, permitirá escribir otra.

Figura V.10. Palabra no disponible para escribir.

Este panel cuenta con un pulsador (recuadrado en color amarillo) que permite desplegar u ocultar el panel. En la figura V.11, se muestra el panel en la parte izquierda y la parte derecha el pulsador asociado.

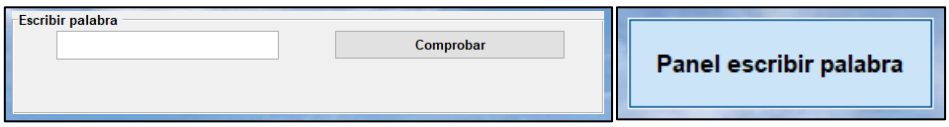


Figura V.11. Panel de comprobar palabra.

V.V.IV. Panel de la plataforma grande.

El panel de la plataforma grande muestra la plataforma grande y el cuadrante de donde el robot va a recoger cada letra para formar la palabra. Este panel cuenta con un pulsador que permite desplegar u ocultar el panel.

En la figura V.12, se muestra el panel vacío en la parte izquierda, el panel con los cuadrantes donde se va a recoger cada cubo para la palabra ‘televisor’ en la parte derecha y la parte inferior el pulsador asociado.

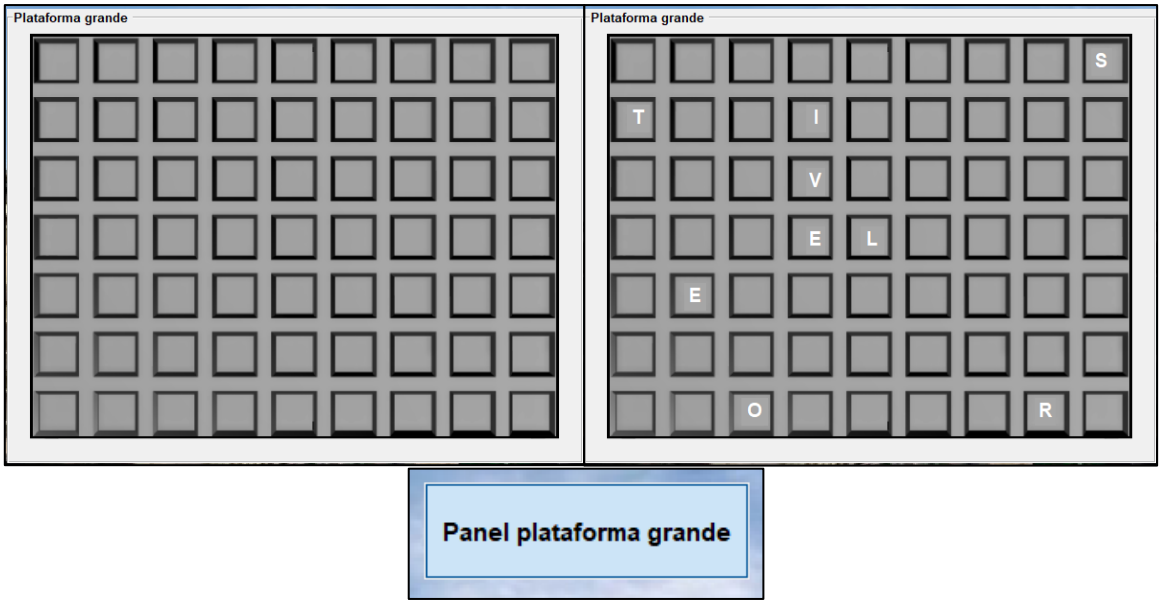


Figura V.12. Panel de la plataforma grande.

V.V.V. Panel de la plataforma pequeña.

El panel de la plataforma pequeña permite ver la plataforma pequeña y el lugar donde el robot va a depositar cada cubo para formar la palabra. Este panel cuenta con un pulsador que permite desplegar u ocultar este panel.

En la figura V.13, se muestra el panel vacío en la parte izquierda, el panel con el lugar donde se va a depositar cada cubo para la palabra 'televisor' en la parte derecha y el pulsador asociado en la parte inferior.

Figura V.13. Panel de la plataforma pequeña.

V.V.VI. Panel del robot.

El último panel se encuentra en la parte derecha, que es el panel que muestra lo que ve el robot a través de la webcam. Al mismo tiempo se muestra el estado de la conexión TCP.

Figura V.14. Visión de la webcam.

Donde la interfaz informa al usuario de si el robot se encuentra quieto o en movimiento, a través de un mensaje y un piloto verde en caso de que este parado y con una señal de advertencia en caso de que esté en movimiento. En la figura V.15, se puede observar en la parte izquierda los avisos con el robot quieto y en la parte derecha con el robot en movimiento.

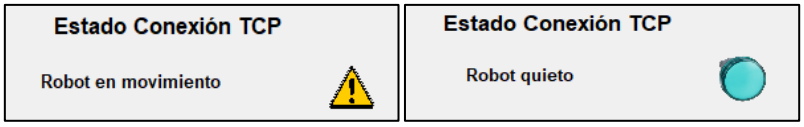


Figura V.15. Avisos del estado del robot.

Al mismo tiempo se informa del estado de la conexión TCP mediante un mensaje y un piloto de color. En la figura V.16, se puede ver en la parte izquierda los avisos sin establecer la conexión TCP y en la parte derecha con la conexión TCP establecida.

Figura V.16. Estado de la conexión del robot.

Además, en este panel se permite regular la velocidad a la que se quiere que se mueva el robot mediante una barra deslizante que se puede llevar desde el 0% al 100%. Este rango de velocidades del robot no se corresponde con la velocidad mínima y máxima del robot sino con un rango de velocidad segura tanto para el robot como para el usuario. En la figura V.17, se muestra diferentes velocidades del robot.

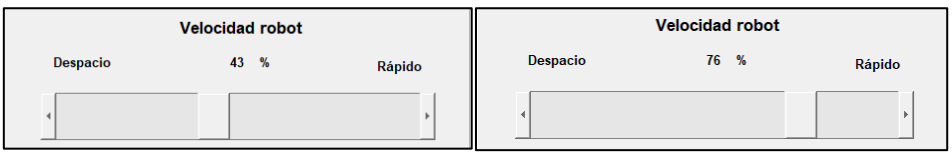


Figura V.17. Barra deslizante para regular la velocidad del robot.

Es importante saber que antes de seleccionar la palabra que se quiere formar, se seleccione a la velocidad a la cual se quiere que se mueva el robot. Finalmente, en la figura V.18, se puede observar el panel completo del robot.

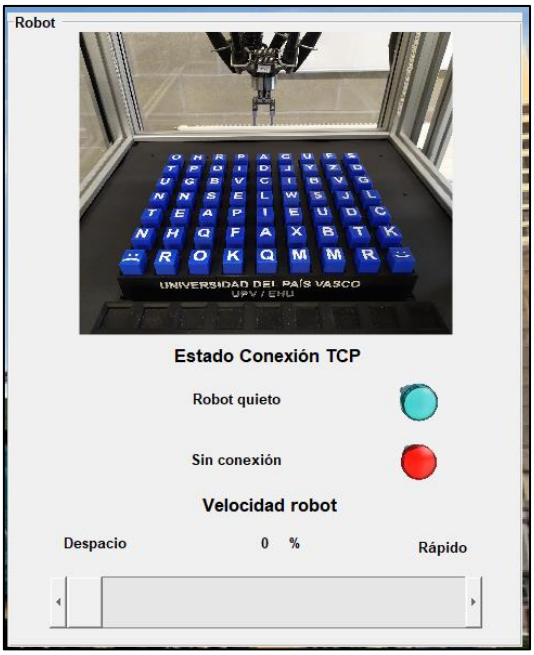


Figura V.18. Panel completo del robot.

V.V.VII. Pestañas habilitadas.

Por otro lado, se han habilitados dos pestañas en la parte superior izquierda de la interfaz. La primera pestaña muestra un documento con el histórico de las palabras que se han escrito con la fecha y hora de cada una de ellas, como se puede ver en la figura V.19.

```
Fecha: 28-Mar-2022 10:07:39  --->  Palabra escrita: camara
Fecha: 28-Mar-2022 10:09:10  --->  Palabra escrita: lampara
Fecha: 28-Mar-2022 10:10:59  --->  Palabra escrita: robot
Fecha: 28-Mar-2022 10:12:18  --->  Palabra escrita: master
Fecha: 28-Mar-2022 10:14:07  --->  Palabra escrita: escuela
Fecha: 28-Mar-2022 10:15:50  --->  Palabra escrita: estufa
Fecha: 28-Mar-2022 10:17:32  --->  Palabra escrita: estufa
Fecha: 28-Mar-2022 10:17:56  --->  Palabra escrita: estufa
Fecha: 28-Mar-2022 10:18:20  --->  Palabra escrita: estufa
Fecha: 28-Mar-2022 10:19:53  --->  Palabra escrita: piezas
Fecha: 28-Mar-2022 10:21:23  --->  Palabra escrita: motor
Fecha: 28-Mar-2022 10:21:24  --->  Palabra escrita: motor
Fecha: 28-Mar-2022 10:23:17  --->  Palabra escrita: movil
```

Figura V.19. Parte del documento de historial de palabras.

También se ha implementado una pestaña pensada para ayudar al usuario a utilizar correctamente la interfaz. Esta pestaña se llama ‘Manual de usuario’ y al pinchar en ella despliega por pantalla un documento con una interfaz, como se muestra en la figura V.20.

```
---------------------------------- MANUAL DE USUARIO PARA EL ROBOT PARALELO -------------------------

----------------------------------- ALIMENTAR ROBOT Y ENCENDER INTERFAZ ----------------------------

PASO 1. Apretar el pulsador verde para alimentar el robot.

PASO 2. Esperar 5 segundos y colocar el selector en la posición superior (3 INIT), cuando el piloto
ver se encienda, bajar el selector a la posición inferior (4 RUN).

PASO 3. Ejecutar la aplicación del escritorio de Windows IROBOT.

PASO 4. Una vez se cargue la aplicación, aparecerá la ventana de la interfaz donde habrá que pulsar
en 'START' para comenzar a utilizar la interfaz gráfica.

PASO 5. Después de pulsar 'START', pulsar en tres los paneles para desplegarlos ('panel escribir
palabra', 'panel plataforma grande', 'panel plataforma pequeña').

Los paneles de las plataformas muestran los huecos de donde el cubo va a recoger cada de las letras
para formar la palabra.

En la pestaña Idioma, se puede seleccionar el idioma de las palabras de la interfaz para escribir
las palabras en euskera.

------------------- REPETIR LOS PASOS 6, 7 Y 8 PARA ESCRIBIR LA SIGUIENTE PALABRA -----------------

PASO 6. Primero habrá que ajustar la velocidad del robot (0 – 100%) deseada de la barra de la
derecha del panel robot deslizándola.

PASO 7. Con la velocidad seleccionada hay que seleccionar la palabra que se desea escribir, para
ello hay dos paneles; el panel 'palabras' donde pulsando sobre los botones de cualquier palabra el
robot comenzará a escribir esa palabra.
La segunda forma es desde el panel 'escribir palabra' donde se escribirá la palabra deseada en el
cuadro en blanco y se pulsará en comprobar. Si la palabra se puede escribir con las letras de la
plataforma del robot, este comenzará a moverse, en cambio, si no hay suficientes letras para
escribir dicha palabra, mostrará un mensaje de 'palabra no disponible' y permitirá escribir otra.

PASO 8. Cuando el robot termine de recoger los cubos después de escribir una palabra, esperar 3
segundos y colocar el selector en la posición superior (3 INIT), esperar un segundo y bajar el
selector a la posición inferior (4 RUN).

Paso 9. Para cerrar la interfaz, es necesario pulsar en 'Stop', esperar 3 segundos y posteriormente
se puede pulsar el boton de 'Cerrar programa' para apagar la aplicación.
```

Figura V.20. Manual de usuario de la interfaz.

En la figura V.21, se puede observar las pestañas de la interfaz.

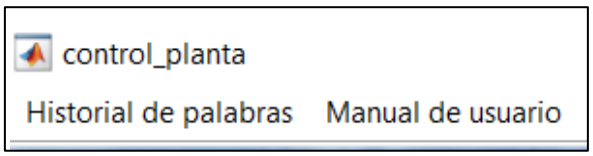


Figura V.21. Pestañas de la interfaz gráfica.

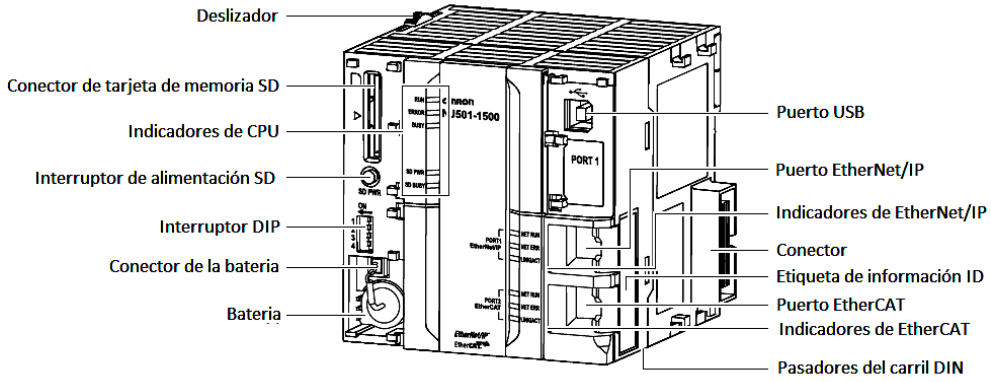
Anexo VI:

INFORMACIÓN DISPOSITIVOS DEL ROBOT

VI.I. Controlador.

Partes y funciones que tiene el controlador:

- Deslizador. Para mantener las unidades juntas.
- Conector de tarjeta de memoria SD. Conecta la tarjeta de memoria SD a la CPU.
- Interruptor de alimentación SD. Se desactiva la fuente de alimentación de modo que se puede quitar la memoria SD.
- Indicadores de CPU. Muestra el estado de funcionamiento de la CPU.
- Interruptor DIP. Se utiliza en modo seguro o al realizar copias de seguridad de los datos. Normalmente están en OFF.
- Conector de la batería. Conector para montar la batería de reserva.
- Batería. Batería para copia de seguridad.
- Puerto USB. Permite conectar el Sysmac Studio a través de un cable USB.
- Puerto Ethernet/IP. Conecta Ethernet/IP con un cable Ethernet.
- Indicadores de Ethernet/IP. Muestra el estado de funcionamiento del puerto Ethernet/IP.
- Conector. Se conecta lateralmente a otras unidades.
- Etiqueta de información ID. Muestra la información de identificación de la CPU.
- Puerto EtherCAT. Conecta EtherCAT con un cable Ethernet.
- Indicadores de EtherCAT. Muestra el estado de funcionamiento del puerto EtherCAT.
- Pasadores del carril DIN. Asegura el montaje de la CPU en el carril DIN.



En cuanto a las características del hardware, la CPU contiene:

- Un puerto de comunicaciones Ethernet/IP, que utiliza para las redes entre controladores o como una red de campo (puede tener hasta 32 conexiones).
- Una red de control EtherCAT, que le permite conectar todos los dispositivos necesarios para el control de la maquina en la misma red.

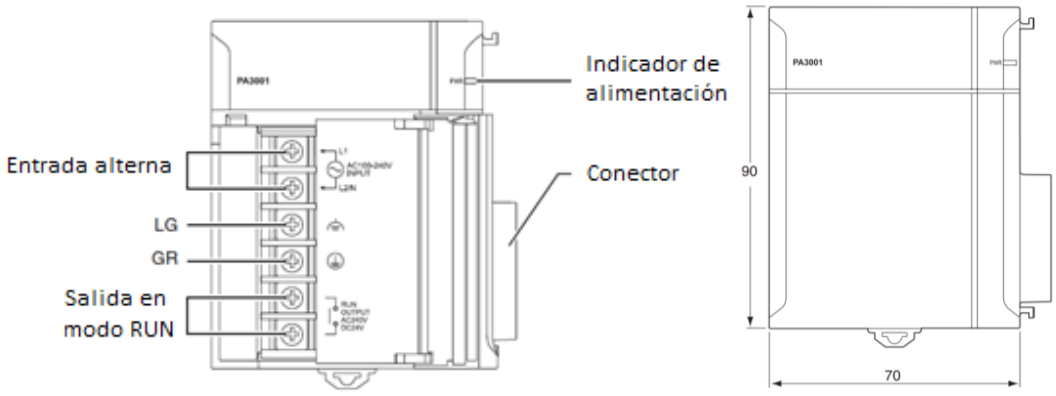
- Un puerto USB para conectar el equipo que ejecuta el software de soporte directamente a la CPU.
- Una tarjeta de memoria SD que se monta en la CPU desde el programa de usuario.

Para más información sobre el controlador NJ501-4300 [34] consultar el documento técnico de la bibliografía.

VI.II. Fuente de alimentación del controlador.

La fuente de alimentación está compuesta de las siguientes partes y funciones:

- Entrada alterna. La entrada por la que recibe el suministro de alimentación.
- LG. Protección a tierra, para aumentar la resistencia al ruido y evitar descargas eléctricas.
- GR. Protección a tierra, para evitar descargas eléctricas.
- Salida en modo RUN. Cuando el controlador está en modo RUN, proporciona una tensión de salida de 24Vdc.
- Indicador de alimentación. Muestra el estado de funcionamiento de la fuente de alimentación.
- Conector. Se conecta lateralmente a otras unidades.



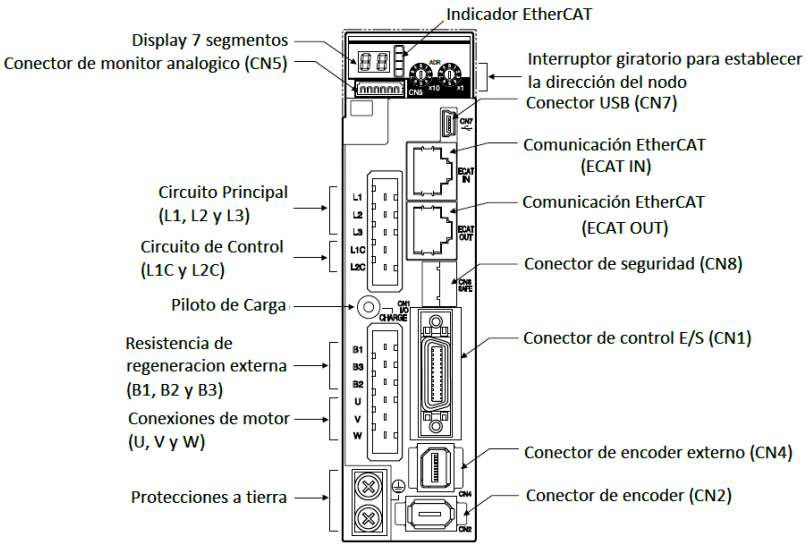
Para más información sobre la fuente de alimentación del PLC NJ-PA3001 [35] consultar el documento técnico de la bibliografía.

VI.III. Servodrivers.

El servodriver está compuesto de las siguientes partes y funciones:

- Display. Un display de 7 segmentos de 2 dígitos, que muestra la dirección del nodo, códigos de errores, y otros estados del servomotor.

- Piloto de carga. Se ilumina cuando la alimentación del circuito de alimentación principal está encendida.
- Indicadores EtherCAT. Estos indicadores muestran el estado de las comunicaciones EtherCAT.
- Conector de control E/S (CN1). Se utiliza para las señales de entrada de comando y señales de entrada/salida.
- Conector de encoder (CN2). Conector para el encoder instalado en el servomotor.
- Conector de encoder externo (CN4). Conector para una señal de codificador utilizado durante el control de cierre total.
- Comunicación EtherCAT (ECAT IN y ECAT OUT). Estos conectores son para comunicaciones EtherCAT.
- Conector de monitor analógico (CN5). Puede utilizar un cable especial para supervisar los valores, tales como la rotación del motor velocidad, el par de valor nominal, etc.
- Conector USB (CN7). Conector de comunicaciones para el equipo.
- Conector de seguridad (CN8). Conector para dispositivos de seguridad. Si no se utilizan los dispositivos de seguridad, mantenga el conector de bypass de seguridad ajustados de fábrica instalado.
- Circuito principal (L1, L2, L3). Conector de donde el servomotor recibe la alimentación.
- Circuito de control (L1C, L2C). Conector de donde el control recibe la alimentación.
- Resistencia de regeneración externa (B1, B2, B3). Conector para la resistencia de regeneración externa.
- Conexión del motor (U, V, W). Conector para alimentar el motor.
- Protección a tierra. Conector a tierra para proteger el dispositivo.

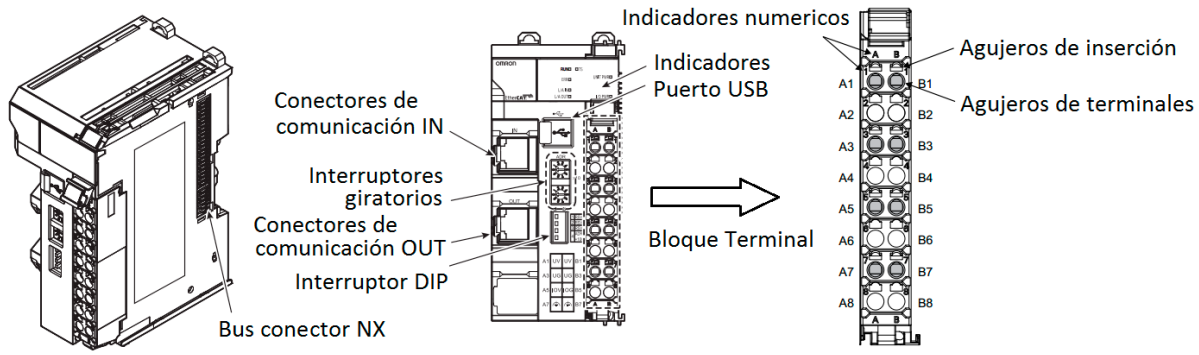


Para más información sobre los servodriver R88D-KN04H-ECT [36] consultar el documento técnico de la bibliografía.

VI.IV. Acoplador EtherCAT.

El acoplador está compuesto por las partes y funcionalidades (Figura 56):

- Bus conector NX: Este conector se utiliza para conectar la unidad de acoplador EtherCAT a la Unidad NX en el lado derecho.
- Indicadores: Los indicadores muestran el estado de funcionamiento actual del acoplador y el estado de la fuente de alimentación.
- Conector de comunicación: Estos conectores están conectados a los cables de comunicaciones de la red EtherCAT. Hay dos conectores: uno para el puerto de entrada y uno para el puerto de salida.
- Puerto USB: Este puerto se utiliza para conectar con el soporte de software de Studio Sysmac.
- Interruptores giratorios: Los interruptores giratorios se utilizan para ajustar el dígito 1s y 10s dígitos de la dirección del nodo del acoplador EtherCAT como un esclavo EtherCAT.
- Interruptor DIP: El interruptor DIP se utiliza para ajustar el dígito 100s de la dirección del nodo del acoplador EtherCAT como un esclavo EtherCAT.
- Bloque terminal: El bloque de terminales se utiliza para conectarse a los cables de alimentación y cable de tierra. Tiene unos indicadores numéricos para asignar las posiciones los agujeros de terminales. Los terminales se introducen en los agujeros de terminales con la ayuda de los agujeros de inserción.



Para más información sobre el acoplador EtherCAT NX-ECC201 [38] consultar el documento técnico de la bibliografía.

Anexo VII:

CALIBRACIÓN DEL ROBOT

VII.I. Método de calibración del robot Mini Delta.

La siguiente tarea a realizar es la calibración de los encoders del robot, que se pueden descalibrar debido a que las baterías de los servodrivers se agotan. Con objeto de recalibrar los encoders absolutos para que el robot vuelva a funcionar correctamente existen dos métodos:

- Kit de calibración.
- Método alternativo.

VII.I.I. Calibración mediante kit de calibración.

La calibración mediante kit de calibración consiste en emplear una herramienta especial que suministra Omron. Este juego de calibración tiene la referencia CR_ART.1058.

Figura V.1. Kit de calibración CR_ART.1058.

Para realizar la calibración con este método, se deben realizar los siguientes pasos para poner los tres brazos superiores del robot en la posición cero:

1. Soltar el freno del motor y asegurarse de que todos los brazos primarios estén girados hacia abajo, lo suficiente para que la herramienta de calibración pueda ser montada.
2. Deslizar la herramienta de calibración en la placa base como se muestra en la figura 6.40.
3. Apretar la tuerca de estrella hasta que la herramienta esté fija.

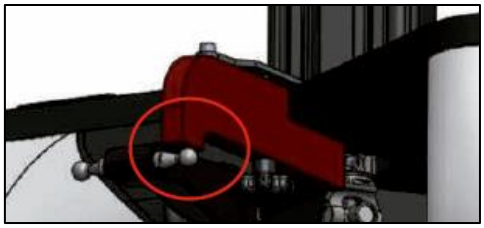


Figura V.2. Tuerca de estrella de la herramienta.

4. Soltar el freno del motor seleccionado y empujar el brazo superior con su rótula contra la herramienta de calibración.

5. Reponer el freno del motor seleccionado.
6. Repetir los pasos de calibración para los otros dos brazos primarios.

Una vez hecho esto, todos los brazos primarios están en posición cero y desde el modelo cinemático y siguiendo los siguientes dos pasos, el robot queda totalmente calibrado:

1. Poner los valores del codificador de los servomotores en 0º.
2. Comprobar que el ángulo indicado para los tres motores sea 0° (± 0,1°).

Para conseguir definir el origen de la máquina, hay que emplear el siguiente procedimiento:

1. Soltar los ejes de la parte móvil para permitir que cada eje se mueva libremente.
2. Montar un tope mecánico en el origen de máquina de cada brazo del robot.

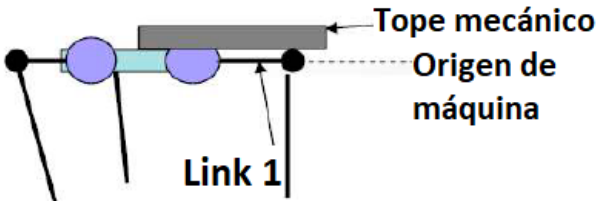


Figura V.22.Esquema del tope mecánico y origen de la máquina.

3. Liberar el freno del motor de Link 1.
4. Empujar el eje manualmente hasta el tope mecánico.
5. Poner el freno mientras se presiona el eje.
6. Resetear el valor multivuelta del encoder absoluto a 0 utilizando la función de encoder absoluto de Sysmac Studio.
7. Establecer la posición actual a 0 mediante la función de Sysmac Studio MC Test Run. De esta forma, el valor de compensación del encoder absoluto para la posición 0 se establece en el área de memoria no volátil de la CPU.
8. Repetir la operación para los ejes 2 y 3 para definir el origen de todos los ejes.
9. Cuando se haya establecido el origen de todos los ejes, quitar la alimentación a los ejes y montar la parte móvil.

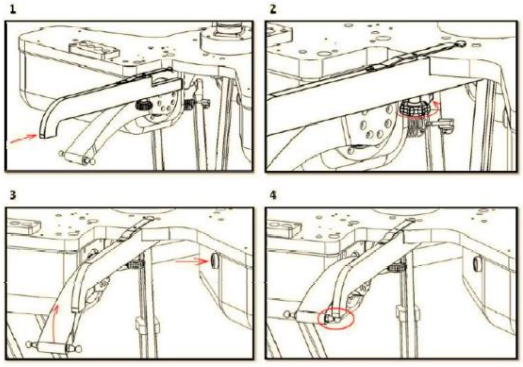


Figura V.23. Procedimiento de Homing.

VII.I.II. Calibración mediante método alternativo.

El segundo método está pensado para cuando no se dispone de dicha herramienta, como es este caso. Es el método empleado para este proyecto y consiste en calibrar el robot sin utilizar

la herramienta que proporciona Omron. Para ello, se ha planteado un sistema parecido al de la herramienta mediante tres perfiles de aluminio y tres sargentos que se pueden ver en la figura 6.43.

Figura V.24. Perfiles y sargentos utilizados.

Los tres perfiles tienen que tener la misma sección, en este caso se ha utilizado una sección de 40mm. Una vez se dispone de las herramientas, hay que colocar cada perfil en cada eje de giro y fijarlo mediante los sargentos. Los tres perfiles tendrán que estar situados a la misma altura y paralelos al techo para que todos los ejes estén calibrados respecto a la misma referencia, como se ilustra en la figura 6.44.

Figura V.25. Colocación de los perfiles y los sargentos.

VII.II. Calibración del hardware del PLC.

Con los sargentos y perfiles colocados, se muestran los pasos realizados para mover cada eje individualmente y así poder calibrar el robot desde el PLC.

Para ello, primero hay que realizar una conexión directa a través del cable ethernet del PLC al ordenador, se ejecuta Sysmac Studio y se selecciona la opción conectar con dispositivo

permitiendo realizar una transferencia de datos desde el dispositivo, como muestran las indicaciones de la figura 6.45.

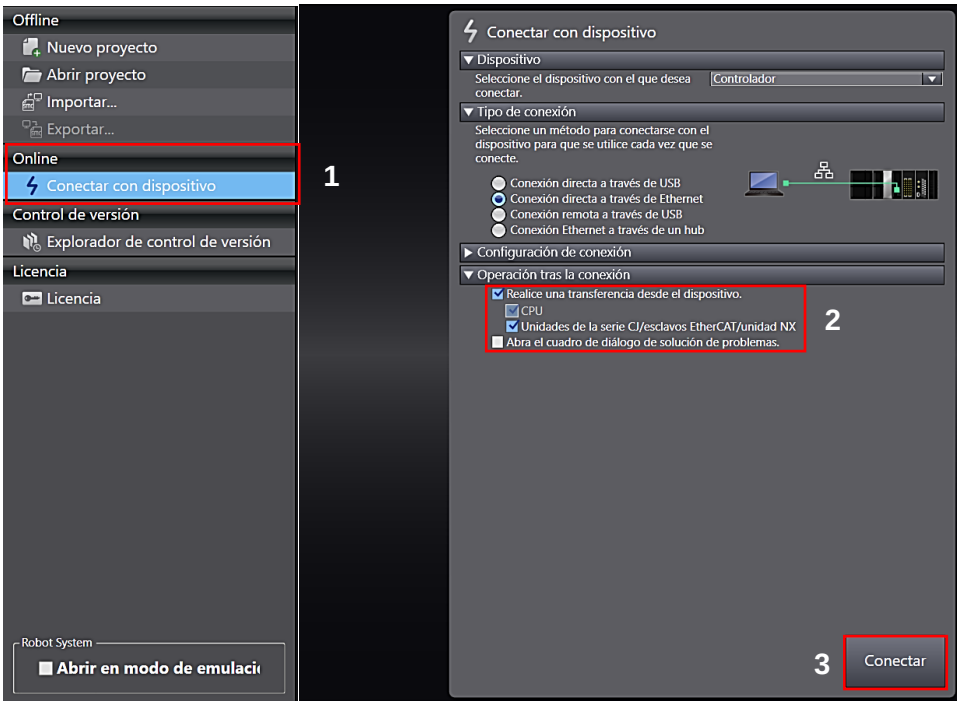


Figura V.26. Conexión con el dispositivo.

A continuación, se muestra el desplegable de configuraciones y ajustes. Se selecciona la opción configuración de control de movimiento, posteriormente configuración de eje y aparecen los tres ejes del robot. Se pulsa en el eje X, se selecciona “Iniciar prueba de Funcionamiento MC” y aparecerá el aviso de precaución de la figura 6.46 y se pulsa en aceptar.

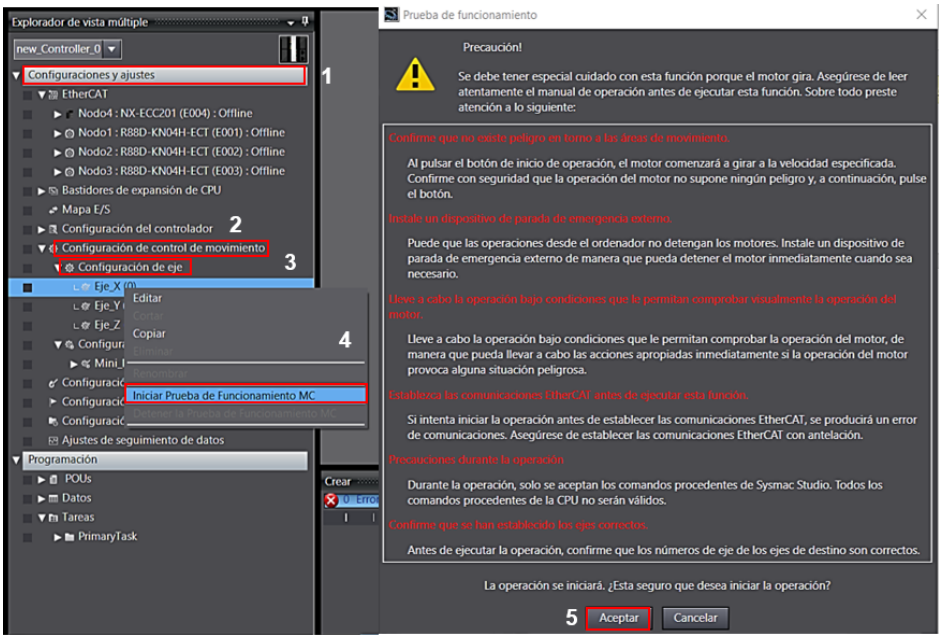


Figura V.27. Prueba de funcionamiento MC.

Se abrirá una nueva pestaña donde se podrá elegir el eje con el que se quiere trabajar, pero para ello, es necesario pulsar el botón verde de la botonera del robot y restablecer los errores desde Sysmac. Estos pueden ser debidos a que los servomotores no están alimentados o a que se han superado los umbrales límite de giro de los mismos.

Tras encender el servo, se permiten cuatro tipos de movimiento para el eje seleccionado:

- Jogging.
- Posicionamiento relativo.
- Posicionamiento absoluto.
- Homing.

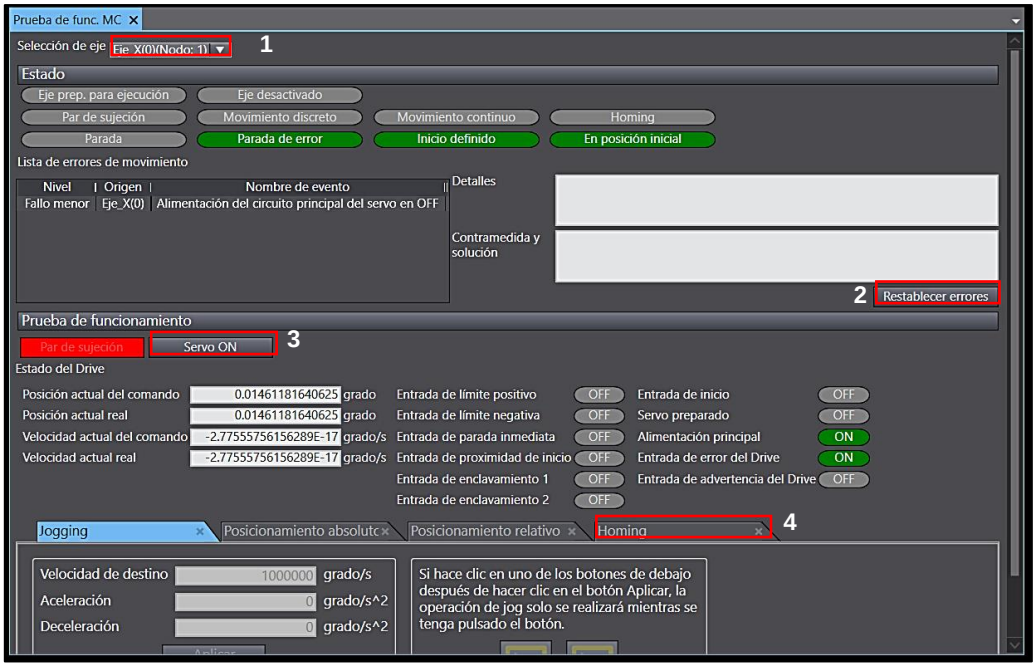


Figura V.28. Arranque del servomotor.

Para calibrar el encoder, la pestaña que se utiliza es la de Homing. En ella ha sido necesario configurar previamente el desplazamiento a la posición inicial, como se puede ver en la figura 6.48.

Figura V.29. Prueba de funcionamiento homing.

Como resultado aparece la ventana de la figura 6.49, donde hay que seleccionar el método de homing, la dirección de inicio de homing, la dirección de detección de entrada de inicio y la velocidad.

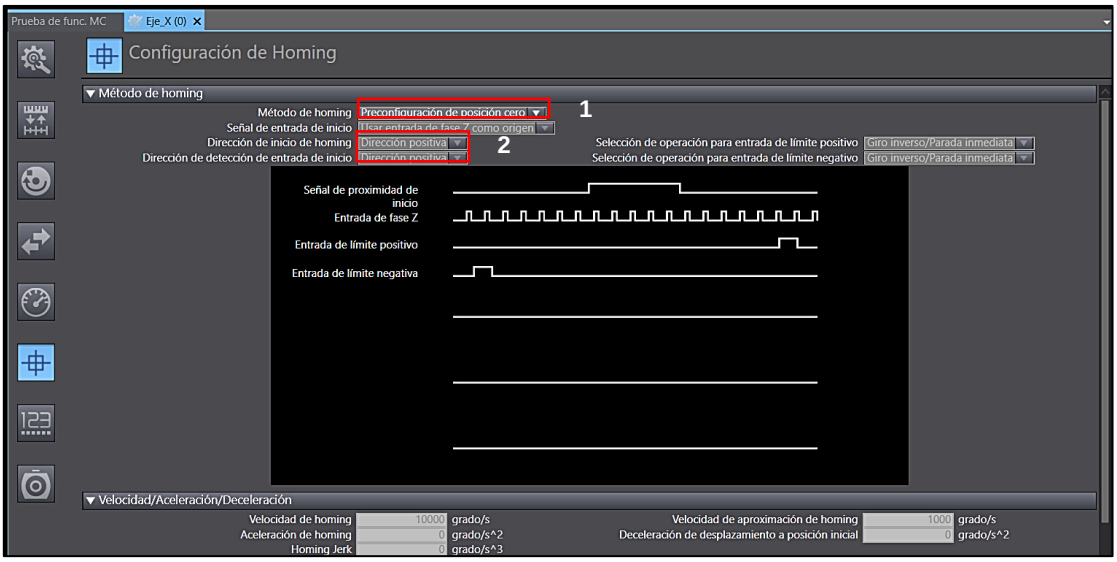


Figura V.30. Configuración del homing del eje.

Con la configuración del homing del eje definida, se aplica la configuración de desplazamiento que se acaba de editar y se pulsa en “play”, de esta manera se está definiendo que la posición en la que se encuentra actualmente el eje es su nueva posición cero (posición de reposo).

Ahora, se regresa a la pestaña de movimiento “Jogging”, se define la velocidad de destino a 10 º/s (velocidad del movimiento del robot segura), se aplica y se pulsa en los botones de la derecha para avanzar en sentido horario o antihorario y el robot comenzará a desplazarse. Para detenerlo basta con soltar el pulsador.

Una vez se alcanza la posición deseada, hay que guardar esa posición como home. Para ello, se vuelve a la pestaña de movimiento “Jogging”, se pulsa de nuevo en aplicar y después en “play”, como se muestra en la figura 6.50. Por último, hay que apagar el servomotor.

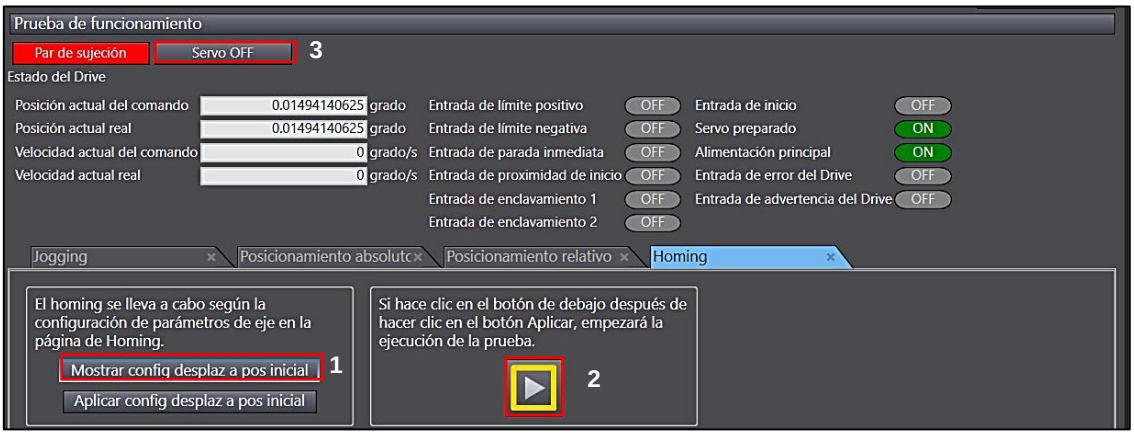


Figura V.31. Configuración para el guardado de la posición.

Para calibrar correctamente los tres ejes, se emplean los tres perfiles comentados anteriormente. Para ello, cuando se pulse el “play”, se permitirá al robot moverse hasta que el extremo de uno de los dos bulones del eje toque el perfil (ver Figura 6.51), en ese momento habrá que parar el servomotor.

Figura V.32. Posición de homing del eje.

Se repiten estos pasos para cada uno de los ejes del robot y de esta manera queda definida la nueva posición de reposo. Para la aplicación que se ha desarrollado, la nueva posición de reposo, se encuentra a una distancia de 15cm desde el final de la pinza hasta la base de la célula robotizada, como se puede observar en la figura 6.52.

Figura V.33. Distancia de la nueva posición de reposo.

Es importante conocer que el espacio de trabajo del robot es móvil, es decir, la posición de reposo del TCP, define la posición de la punta de la flecha de la figura 6.53. De manera que si con una configuración de reposo, no es posible alcanzar todos los puntos para una determinada aplicación, puede que si sea posible alcanzarlos definiendo el TCP a una distancia menor de la plataforma.

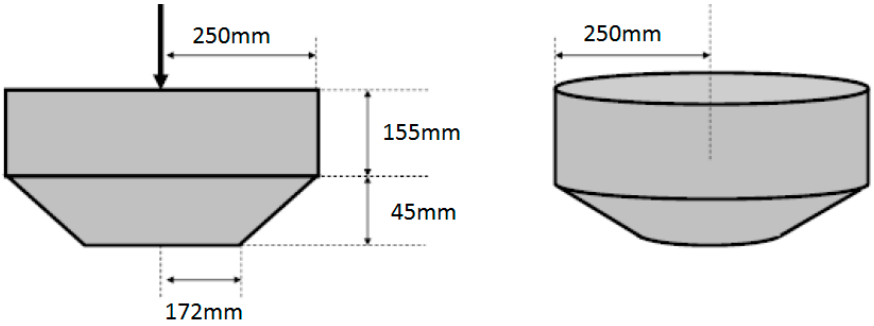


Figura V.34. Espacio de trabajo del robot.

Esto es debido a que el rango de giro de los servomotores va desde -40º a 110º y se define como 0º el punto en el que se realiza la preconfiguración como posición cero.

VII.III. Selección de coordenadas.

El robot Mini Delta tiene tres tipos de sistemas de coordenadas, como se muestra en la figura 6.54.

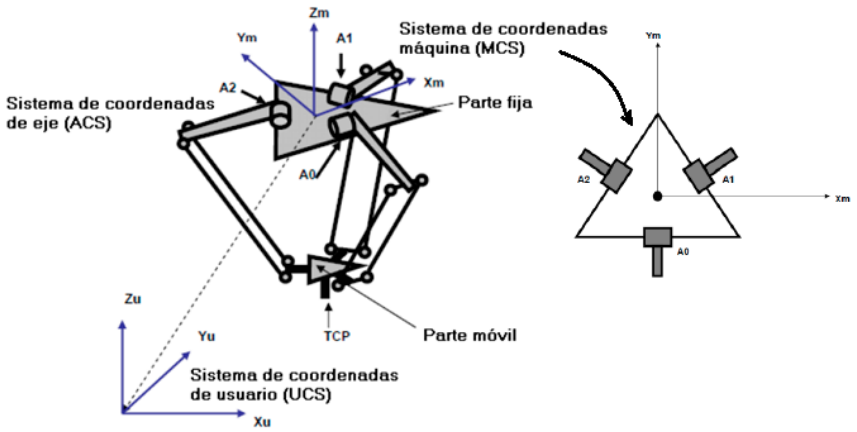


Figura V.35. Sistemas de coordenadas del robot Mini Delta.

- ACS (Axis Coordinate System). Es un sistema de coordenadas de eje, corresponden al sistema de coordenadas específico de cada eje donde se crea un sistema de coordenadas por cada eje.
- MCS (Machine Coordinate System). Hace referencia al sistema de coordenadas de máquina, es el sistema de coordenadas específico de cada robot y existe un sistema de coordenadas por cada robot.
- UCS (User Coordinate System). Es el sistema de coordenadas de usuario, se crean a partir de datos de posición y rotación (Tx, Ty, Tz, Rx, Ry, Rz) creados en base al MCS.

Para cambiar el sistema de coordenadas de la máquina, y establecer el sistema de coordenadas del usuario se utiliza la función MC_DefineCoordSystem, que permite colocar el sistema de coordenadas en la posición y orientación más adecuada para el espacio de trabajo en el que se trabaje.

El robot Delta siempre trabaja en coordenadas de eje (ACS), sin embargo, se ha definido un programa en el que se pueden introducir las posiciones en MCS y mediante la función MC_InverseKin se realiza la cinemática inversa, convirtiendo la referencia de posición MCS a coordenada ACS.