ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA

Trabajo de Fin de Grado:

"Diseño e Implementación de un Sistema Autónomo de Detección y Clasificación de Residuos"

Titulación: *Grado en Ingeniería Electrónica Industrial y Automática*

Autores:
Alberto Méndez García
Stefano Fernando Panzeri Reyes

Tutores:
Santiago Torres Álvarez
Silvia Alayón Miranda

11/06/2021

AGRADECIMIENTOS

Lo primero, a nuestros tutores, Santiago Torres y Silvia Alayón, por habernos apoyado desde que les comentamos por primera vez la idea. Por estar disponibles siempre para poder ayudarnos y poder seguir avanzando con nuestro proyecto, incluso en fines de semana.

En general, a todo el profesorado del grado por sus enseñanzas. En especial, a todos los que nos han ayudado directamente en este proyecto. A Alejandro Ayala, por habernos orientado para resolver el acondicionamiento de señales electrónicas. A José Sigut, por buscar información para el entrenamiento de la red. A Iván Rodríguez, por habernos ayudado con todo lo relativo a la impresión 3D. A Fernando Rosa, por habernos aconsejado sobre el uso de las redes neuronales. A Alberto Hamilton, por habernos ayudado en la implementación del Bot de la Raspberry.

Gracias a los técnicos de laboratorio, como Manuel, por habernos proporcionado todo el material que nos ha hecho falta para la realización de este proyecto; y a Ramón, por habernos facilitado el material de impresión 3D que necesitábamos en su momento.

Gracias también a la Escuela Superior de Ingeniería y Tecnología de la ULL, en especial, al Departamento de Ingeniería Informática y de Sistemas, por habernos siempre puesto todas las facilidades a la hora de solicitar cualquier tipo de material.

A todos nuestros amigos, que nos ayudaron en la recolección de residuos para obtener las imágenes necesarias para el entrenamiento de la red (María Jesús, Marina, Paula y Reda). Y a los que además nos ayudaron a pintar el prototipo (Nashira, Patricia y Sergio). Sin olvidarnos de los que nos aportaron otra visión de ingeniería, para implementar ciertos elementos mecánicos y eléctricos (Adrián y Tomás).

A nuestro vecino Alfonso, por habernos facilitado material y herramientas para la programación del prototipo cuando aún no disponíamos de ellas.

Y, por último, pero no menos importante, a nuestras familias por habernos apoyado durante todo este proceso, aportándonos ánimos e ideas cuando más lo necesitábamos. Así como, ayudándonos directamente en la implementación del prototipo.

A todos ellos, mil gracias, porque seguramente, un resultado tan bueno como el conseguido, no hubiera sido posible sin el apoyo y contribución de todos ellos.

RESUMEN

En el siguiente Proyecto denominado "Sistema Autónomo de Detección y Clasificación de Residuos" se ha construido un prototipo de madera de una papelera inteligente. El prototipo en cuestión es capaz, mediante un autómata y una Raspberry, de clasificar de manera totalmente autónoma objetos de los tres materiales de reciclaje principales: cartón, plástico y vidrio.

El funcionamiento del prototipo implementado es muy intuitivo. El usuario debe introducir el objecto a reciclar en una cinta transportadora superior. Una vez se detecte el objeto, esta comenzará su movimiento hasta que el objeto caiga a través de una rampa a una cinta transportadora inferior. En este momento, una cámara conectada a la placa Raspberry Pi, capturará una imagen del objeto y la procesará a través de una red neuronal. Una vez determinado el material del citado objeto, el sistema lo posicionará sobre su correspondiente cubo para posteriormente empujarlo y permitir su caída dentro del mismo.

ABSTRACT

In this project, "Autonomous System for Detection and Sorting of Waste" a wood prototype of a smart bin is built. The mentioned prototype is able, with an automaton and a Raspberry Pi, to classify objects belonging to one of the three main recycling materials: carboard, plastic and glass.

The functioning of the implemented prototype is very intuitive. The user must introduce the object to recycle on a superior conveyor belt. After that, this conveyor belt will start its movement till the object falls down a ramp on an inferior conveyor belt. At this moment, a camera connected to the Raspberry Pi will take a picture of the object and process it through a Neuronal Network of Deep Learning, to determine its material. Once the material is clear, the system will move the object to its position (according to the material) and will push it into the corresponding bucket. The main control of the system is done by an automaton.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. OBJETIVOS DEL PROYECTO

El principal objetivo de este proyecto es el de implementar físicamente un sistema autónomo de identificación y clasificación de residuos reciclables. Para ello, se ha construido un prototipo de papelera inteligente que es capaz de identificar el material del objeto depositado, mediante un sistema de visión en combinación con un sistema de Inteligencia Artificial, para posteriormente situarlo adecuadamente en su correspondiente cubo de manera automatizada. De esta manera, se consigue un reciclaje rápido y efectivo, que no requiere una clasificación previa por parte del usuario, eliminando así el posible error humano e incentivando a reciclar debido a las facilidades que este prototipo proporciona.

Adicionalmente, este proyecto está concebido para superar la asignatura de "Trabajo de Fin de Grado" de 12 ECTS, perteneciente al Grado en Ingeniería Electrónica Industrial y Automática de la Universidad de La Laguna (ULL). El trabajo ha sido tutorizado por Santiago Torres Álvarez y Silvia Alayón Miranda, ambos profesores pertenecientes al Departamento de Ingeniería Informática y de Sistemas de la ULL.

Para la realización del proyecto, se han invertido unas 270 horas de trabajo de trabajo autónomo por parte de cada alumno, así como la asistencia a reuniones con los tutores al menos dos veces al mes.

2. INTRODUCCIÓN AL PROBLEMA

Este proyecto nace de la necesidad de aumentar la cultura del reciclaje en España. En un viaje a Italia nos llamó la atención que la mayoría de la población reciclaba, quizás motivados por el funcionamiento de su sistema de recogida de residuos. En España, en general, la población carece de dicha motivación. Las estadísticas de Ecoembes en el año 2018 [1] indican que Italia alcanza un 45% de reciclaje de residuos municipales, cifra cercana al 55% exigido por la UE para el año 2025. España, por su parte, está lejos de esa cifra, con un 33% de reciclaje. En Canarias, en ese mismo año, la cifra era aún peor: sólo el 18%. Aunque un año más tarde se empezó a revertir la situación y el reciclaje en las Islas Canarias superó a la media nacional, creciendo un 9,1%. Además, muchas veces no se recicla de forma correcta, pues los residuos tienen restos orgánicos, o su propio diseño dificulta su reciclaje ya que confunde al usuario, como sucede con los tetrabriks.

Para solventar estos problemas nace este prototipo de papelera inteligente. Con él se pretende mostrar que, haciendo uso de la tecnología, se puede simplificar la tarea del reciclaje y, en consecuencia, mejorar la cultura del reciclaje en el país.

Con esta papelera el usuario no tendría que preocuparse por el error humano, ni por tener contenedores separados, solo tendría que introducir los residuos en el prototipo y, cuando viera un cubo lleno fuera, ir a tirarlo al correspondiente contenedor para el posterior reciclaje municipal. Además, debido a las facilidades proporcionadas, el prototipo podría mejorar el hábito de reciclaje de todos los miembros de la familia, desde los más pequeños hasta los mayores de la casa. De esta forma, aumentaría el alcance de la cultura del reciclaje, persiguiendo el objetivo de reciclar más y mejor.

3. ESTADO DEL ARTE

El proyecto planteado es innovador y único en cuanto a diseño se refiere, pero existen ya en el mercado unos productos comerciales que tienen una filosofía similar. Estos proyectos se detallan a continuación:

BIN-E [2] es un proyecto desarrollado en Polonia. En esta papelera se introducen uno a uno los objetos a reciclar, y el tipo de material se reconoce con técnicas basadas en Inteligencia Artificial (albergadas en la nube). Posteriormente, con un raíl interno, se desplaza el objeto hasta su correspondiente cubo. Este producto cuenta con 4 cubos para los diferentes materiales reciclables: plástico, cartón, papel y metal. Además, cuenta con una pantalla que muestra información acerca del nivel de llenado de los cubos, así como de las estadísticas de los objetos depositados.

Figura 1. Papelera BIN-E. [2]

La principal ventaja del prototipo desarrollado en este TFG en comparación con BIN-E es la posibilidad que tiene el usuario de poder tirar varios objetos simultáneamente, y que sea la propia papelera quien los gestione para procesarlos de uno en uno.

IRBin [3] es un robot social desarrollado en la Pontificia Universidad Católica de Perú. Se trata de una papelera que interactúa con el usuario (con la idea de crear un vínculo y un hábito de reciclaje) y que es capaz de reciclar diferentes objetos, utilizando Inteligencia Artificial para el reconocimiento de estos. Esta papelera recibe los objetos de uno en uno y es capaz de separar botellas de plástico, de vidrio y basura orgánica, mediante una cinta transportadora y un brazo mecánico.

Figura 2. Promo del IRBin. [3]

Nuestro prototipo ofrece la posibilidad de tirar varios objetos a la vez, además de poder separar cartón, aunque no acepta basura orgánica, como IRBin.

El **"Automatic Waste Segregation System"** [4] se trata de un sistema de identificación y clasificación de residuos que es capaz de separar entre basura metálica, seca y mojada. Este sistema está controlado por una PCB que evalúa si se ha tirado un objeto, mediante un sensor infrarrojo, y mediante un segundo sensor, evalúa de qué tipo de material se trata. Una vez identificado, mueve los cubos situados en una plataforma circular para que el objeto caiga en el correspondiente cubo.

Figura 3. Prototipo de Nevon Projects. [4]

El aspecto en el que más se diferencia nuestro prototipo de este producto es que realiza la clasificación entre los tres materiales de reciclaje más habituales en la vida cotidiana (plástico, cartón y vidrio), además de prescindir de sensores para identificar el tipo de material, utilizando para ello técnicas de Inteligencia Artificial.

TrashBot [5] es un sistema desarrollado por la empresa CleanRobotics que se encarga de separar materiales reciclables de aquellos que no lo son. Para ello, se introduce el objeto en una determinada zona, donde es fotografiado y evaluado con Inteligencia Artificial para determinar si es reciclable o no. Acorde a la decisión tomada, un sistema robótico hace que el objeto caiga en un cubo u otro.

Figura 4. TrashBot. [5]

La principal ventaja de nuestro prototipo, en comparación con este, es que es capaz de determinar no solo si el material es reciclable, sino también de qué tipo de material se trata, y separarlo adecuadamente en consecuencia.

Otro proyecto similar al nuestro es **"MY041-Automatic Sorting Recycle Bin with Level Indicator"** [6] . Este sistema presenta un contenedor, en el que se deposita el objeto, con unos sensores capaces de distinguir aluminio, papel y plástico. Una vez determinado el tipo de objeto, una plataforma rotatoria inferior mueve las diferentes bolsas hasta dejar bajo el contenedor la bolsa correspondiente. Una vez posicionada, el contenedor se abre, cayendo así el objeto en su bolsa correspondiente.

Una vez más, la ventaja diferencial de nuestro prototipo es que permite al usuario la introducción de varios objetos simultáneamente y que el reconocimiento se realiza mediante técnicas de Inteligencia Artificial.

4. FUNCIONAMIENTO DEL PROTOTIPO

El prototipo está compuesto principalmente por los siguientes elementos: un autómata, una fuente de alimentación de 24V DC externa y una placa Raspberry Pi con su respectiva cámara. Todo esto, protegido por un cuadro eléctrico. Además, internamente cuenta con dos cintas transportadoras accionadas cada una por un motor, 15 sensores industriales, un desviador accionado por 4 motores y varios cubos de reciclaje, aparte de la propia estructura de madera. Estos elementos se pueden observar en la figura 5

Figura 5. Vista completa del prototipo

En el prototipo hay 6 cubos de reciclaje, 2 para cada material (plástico, cartón y vidrio). De esta manera, se asegura un proceso de reciclaje lo más continuo posible, evitando que el sistema se atasque si se llena un cubo, ya que existe otro de reserva, mientras el usuario vacía el primero.

El funcionamiento de la papelera es bastante sencillo. El usuario tiene acceso a una cinta transportadora superior en la cual pude ir colocando los diferentes objetos para su clasificación. Es importante que los objetos se coloquen con el lado más largo en paralelo al lado más corto de la cinta.

Figura 6. Objeto colocado correctamente

Figura 7. Objeto colocado incorrectamente

Desde que se coloca el primer objeto, se activa un sensor y el autómata ordena poner en marcha la cinta superior. Cuando el objeto llega al final de la cinta y está a punto de caer por la rampa, otro sensor detecta el objeto en esta posición y el autómata da la orden de parada a la cinta superior. Después de unos segundos (tiempo necesario para que el objeto caiga y se estabilice) el autómata se comunica con la Raspberry, que se encarga de la identificación del objeto.

Recibida la orden correspondiente, la Raspberry comienza obteniendo una imagen mediante una cámara, para posteriormente pasársela a una Red Neuronal de Aprendizaje Profundo (Deep Learning) previamente cargada al arrancar la Raspberry. Es importante indicar que esta Red Neuronal ha sido entrenada y ajustada con imágenes de residuos reciclables antes de ser introducida en la Raspberry. Esta técnica de Inteligencia Artificial procesa la imagen del objeto depositado y devuelve la probabilidad de que el objeto sea de plástico, vidrio o cartón. Si esta probabilidad es mayor que un 50% para alguno de los objetos, entonces se envía una señal al autómata identificando el tipo de material. Pero si estos porcentajes son todos menores que 50%, se le envía al autómata una señal indicando que no se ha podido identificar con certeza el objeto.

En caso de que el resultado lanzado por la Raspberry no sea concluyente (es decir, no sepa determinar de qué tipo de material se trata), el autómata encenderá automáticamente un piloto rojo de emergencia, para indicar que es requerida una acción por el usuario. En cambio, en caso de que el resultado lanzado por la Raspberry sea concluyente (es decir, que haya determinado correctamente de qué tipo de material se trata), entonces comenzará el proceso de clasificación.

La primera parte de este proceso será asegurarse de que el cubo en el cual se debe tirar el objeto esté disponible, es decir, que no esté lleno. Si esta condición se cumple, el sistema posicionará el objeto encima de su cubo correspondiente. Si el material es plástico o vidrio, se activará la cinta inferior y se parará cuando el sensor correspondiente a la posición deseada detecte que el objeto ha llegado. En cambio, en el caso del cartón, este posicionamiento no es

necesario, ya que, al caer por la rampa, el objeto se encuentra de manera predeterminada encima del cubo del cartón.

Una vez posicionado el objeto, se activará el avance del desviador hasta llegar al final de carrera correspondiente. Una vez allí, se parará su avance y se esperará un tiempo para retroceder nuevamente. Si durante el tiempo de avance o el tiempo de espera el sensor correspondiente detecta la caída de un objeto, entonces se entenderá que el objeto ha caído correctamente en su cubo. En caso contrario, si el desviador ya se encuentra retrocediendo y aún no ha caído el objeto, se entiende que el objeto no ha caído, por lo que se encenderá un piloto luminoso de emergencia, indicando al usuario que se requiere su intervención en el sistema.

En caso de que el objeto haya caído correctamente, se evaluará si el correspondiente cubo está lleno. En caso afirmativo, se comenzará con el intercambio de cubos. Para ello, se levantará un dispositivo motorizado (fabricado con 2 DVDs) que bloquea el paso del primer cubo, permitiendo que este salga de la zona de llenado y se ubique en el exterior para que el usuario pueda acceder a él y vaciarlo.

Una vez sacado, se vuelve a bajar el dispositivo motorizado de sujeción a la vez que se eleva el trasero que retiene al cubo de reserva. Una vez que ambos han llegado a su tope, se evaluará si el cubo ha cambiado de posición. En caso afirmativo, se procederá a bajar el dispositivo de sujeción trasero, dando por concluido el proceso de cambio. En caso negativo, se encenderá un piloto rojo de emergencia, indicando que se requiere intervención por parte del usuario.

Debido a que no se disponen de sensores en la parte exterior de la papelera, que puedan detectar la presencia de los cubos fuera, la única manera que tiene el autómata de saber que un cubo se ha vaciado y quitado de la zona exterior es cuando se vuelve a introducir en la parte trasera del prototipo.

Cabe destacar que existen una serie de condiciones límites que se contemplan de la siguiente manera:

I. En caso de que haya un cubo fuera (es decir, que se haya llenado, haya salido y el usuario aún no lo haya vuelto a introducir en la parte trasera) y se llene el segundo cubo del que se dispone, el sistema sabrá que hay un cubo fuera y que el que se tiene actualmente está lleno. Por tanto, no intentará sacar el lleno (para no colisionar con el de fuera) pero tampoco activará la emergencia. Esto es así, ya que, si el siguiente objeto

a reciclar fuera de otro material (cuyo cubo no estuviera lleno), el sistema podría continuar con normalidad.

II. En caso de que haya un cubo fuera, que el segundo cubo también se haya llenado previamente (y no haya podido salir) y se vaya a reciclar un objeto de ese mismo material, el sistema entrará en un estado de emergencia, encendiendo un piloto rojo que indicará al usuario que se requiere una acción (en este caso, vaciar el cubo de fuera y volverlo a introducir).

III. Una vez activo el piloto de emergencia, la única manera para volver al ciclo con normalidad será presionando el botón de rearme.

IV. Al iniciar cada ciclo de identificación y clasificación, se comprueba que el interruptor de marcha está activo. En caso de apagar el interruptor, el sistema acabará la clasificación del correspondiente ciclo, pero no continuará con el siguiente.

Para poder ver con más detalle el funcionamiento anteriormente descrito, se facilitan unas grabaciones de vídeos que se podrán encontrar en la siguiente referencia: [7]

5. DISEÑO E IMPLEMENTACIÓN DEL PROTOTIPO

5.1. EVOLUCIÓN DEL DISEÑO

Lo primero para implementar la papelera fue realizar su diseño. La base de las ideas partía de una serie de componentes clave como las cintas transportadoras, las bolsas/cubos para los residuos y un sistema de salida de estos. Durante el desarrollo de este proyecto el diseño del prototipo fue evolucionando. Para realizar los distintos diseños y entenderlos mejor se simuló su implementación 3D utilizando el programa 3D Builder [8].

5.1.1. PRIMER DISEÑO

Partiendo de las ideas base, el primer diseño consistía en tres cintas, dos en paralelo y una transversal para pasar los objetos de una a otra. Con unos desviadores activados por motores paso a paso, se tirarían los objetos a su correspondiente bolsa y cuando esta se llenara, un pistón eléctrico móvil la presionaría para cerrarla. Luego la bolsa rodaría sobre una rampa de salida para que el usuario la depositara en el contenedor correspondiente. Para cambiar de bolsa se utilizaría un raíl móvil de movimiento circular que al llegar a su extremo soltaría la bolsa debido a la física de su enganche.

Figura 8. Primer diseño propuesto.

Este diseño se descartó debido a la complejidad de sus elementos, sobre todo el pistón móvil, y sus limitaciones físicas, ya que por las dimensiones de las cintas no se podían tirar objetos de "grandes dimensiones". Y a los problemas que podía causar el hecho de controlar algunos elementos mediante Raspberry y otros con el autómata en el prototipo.

5.1.2. SEGUNDO DISEÑO

Para solventar los errores del anterior diseño, se optó por cambiar la disposición de las cintas transportadoras, aumentando su ancho y posicionando una sobre la otra, con una rampa que desplazaba los objetos de la superior a la inferior. En cambio, se mantenía el raíl móvil para el cambio de bolsas.

Figura 9. Segundo diseño propuesto.

Algunos de los problemas se arrastraban del diseño anterior como el conflicto en el control de elementos entre Raspberry y autómata. Y nuevos problemas como que dependiendo del tipo de material, el cierre de la bolsa por su propio peso no era del todo eficiente. Además, la optimización de espacio del prototipo no era la mejor, puesto que la bolsa que se llenaba ocupaba el espacio de dos, debido a que tenía una zona de llenado y otro de "caída libre".

5.1.3. TERCER DISEÑO

Pese a los problemas que presentaba, el segundo diseño estableció el camino a seguir para el diseño final del prototipo. El sistema de una cinta sobre otra parecía ideal en cuanto optimización de espacio. Para mejorar ese diseño, se decidió cambiar las bolsas por unos cubos a medida (evitando así la utilización del pistón para el cierre de bolsas), retenidos por un sistema de barrera que permitiría o bloquearía su desplazamiento sobre una rampa, que daría el movimiento necesario a los cubos para salir del prototipo cuando se encontraran llenos.

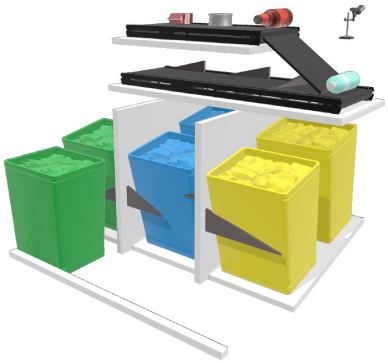


Figura 10. Tercer diseño propuesto.

Este diseño fue el que finalmente se decidió implementar, aunque sufrió algunas modificaciones durante su implementación. Dichas modificaciones se describen en detalle en los siguientes apartados.

5.2. DISEÑO MECÁNICO

Finalizado el diseño, había que pasar a su implementación y la de cada uno de sus componentes. En este apartado, se describen los elementos mecánicos más importantes del prototipo como lo son: su estructura, las piezas 3D, las cintas transportadoras, la rampa, el desviador y los cubos.

5.2.1. ESTRUCTURA DEL PROTOTIPO

Para implementar el diseño se seleccionó la madera como material de construcción, debido a su fácil manipulación (cortes, agujeros, montaje, etc.) La implementación del diseño comenzó con el montaje de la estructura exterior, compuesta de cuatro caras (fondo, paredes laterales y pared trasera). Más adelante se montaron las dos caras restantes (techo y pared

frontal) para no dificultar el montaje de la estructura interna. Las dimensiones de estas paredes (Largo x Ancho) son:

- Fondo: 107 x 100 cm.
- Paredes laterales: 65 x 100 cm
- Pared trasera: 107 x 100 cm
- Techo: 109 x 65 cm.
- Pared frontal: 109 x 45 cm

Las medidas se pueden apreciar mejor en los planos de dimensiones del prototipo adjuntos en el Anexo I realizados mediante Autocad [9].

Estas paredes se montaron con tornillos de ensamblaje y tornillos corrientes. Se utilizó como base el fondo del prototipo, sobre esta superficie se atornillaron nueve listones de unión (tres por cada pared), de tal forma que sirva de sujeción para todas las paredes con las que esté en contacto. Además, para mejorar la estabilidad, desde la cara exterior se atornilla una pared con otra utilizando los tornillos normales. De esta forma se montaron las paredes laterales y la pared trasera, definiendo el contorno del prototipo, como se puede observar en la figura 11.

Figura 11. Contorno del prototipo.

En la parte inferior del interior del prototipo se construyeron 3 rampas y dos paredes interiores para definir las zonas de los cubos, estas estructuras se pueden apreciar en la figura 12. Las rampas se construyeron con unas dimensiones de 60 x 32’5 cm. Y se montaron sobre dos listones anclados al fondo del prototipo, dichos listones le dan una altura de 7 cm a la rampa, generando una pendiente de 6º respecto a la horizontal. Las paredes interiores, con sus dimensiones de 63 x 50 cm, se montaron atornillando desde las caras externas del fondo y la pared trasera, teniendo una sujeción en forma de L. Más adelante, cuando se implementaron los cubos, se hicieron unos agujeros a medida para permitir su entrada en la papelera, con su correspondiente sistema de abertura consiste en una polea que tira de una lona de caucho y permite el acceso a los cubos. Este sistema se puede observar en detalle en la figura 13.

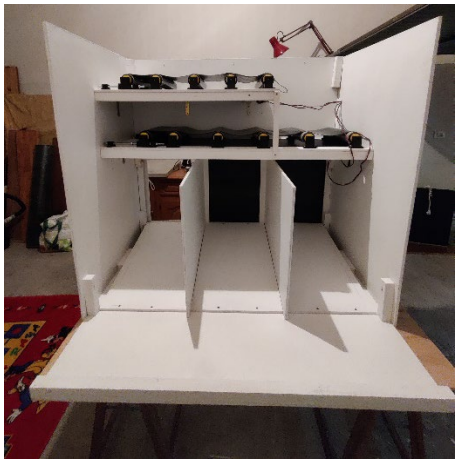


Figura 12. Estructura parte inferior del prototipo.

Figura 13. Cara posterior del prototipo.

En cuanto al montaje de las cintas, para la cinta inferior se utilizó una pieza de 40 x 107 cm. sujetada por escuadras a las paredes y con un listón de lateral a lateral de la papelera para mejorar su estabilidad y resistencia. De igual forma, se montó la cinta superior, pero en esta ocasión con dos listones formando una L, debido a que por sus dimensiones de 40 x 70 cm no llega de lateral a lateral.

Una vez se montaron todos los elementos del interior de la papelera, se montaron las dos caras restantes, techo y pared frontal. El techo tiene dos partes, una parte fija y otra móvil. Para montarlas, previamente se puso un listón de lateral a lateral en el borde delantero y otro desde este hasta la pared trasera, formando una T, para dar estabilidad y sujeción. La parte fija en forma de L cubre la parte superior del prototipo dejando acceso solo a la parte de la cinta, esta parte se atornilla sobre el listón y las paredes lateral y trasera. La parte móvil permite el acceso a la zona de clasificación en la cinta inferior, por si el objeto no fuera reconocido. Esta parte debido a sus características se sujeta con unas bisagras en el listón que permiten su abertura hacia dentro. Además, esta parte cuenta con un agujero para la cámara de la Raspberry.

Figura 14. Parte superior del prototipo

5.2.2. PIEZAS 3D

Para determinadas funciones de soporte y sujeción en los distintos elementos mecánicos del prototipo, se imprimieron piezas en 3D utilizando filamento de PLA [10] debido a su fácil manipulación (todas las piezas 3D utilizadas en este proyecto comparten dicho material). Se optó por este tipo de piezas para estas aplicaciones puesto que su diseño podía ser único, adaptándose a las necesidades del prototipo. El diseño de las piezas se realizó utilizando el programa Fusion 360 [11] y se exportaron los archivos a imprimir utilizando, Ultimaker Cura [12] y Prusa Slicer [13].

Las piezas implementadas son:

- Soportes para los rodillos.
- Conjunto tensor de rodillos.
    - Soporte pasante.
    - Soporte del rodillo tensor.
- Soporte del motor de la cinta.
- Polea del motor de la cinta.
- Soporte del motor del desviador.
- Pieza de unión para la transmisión.

5.2.3. CINTAS

Las cintas son el elemento fundamental para el desplazamiento de los objetos por el interior de la papelera. Para su construcción se utilizaron unos rodillos transportadores de PVC con unas dimensiones de 30 cm de largo y un diámetro de 5 cm, la propia cinta y un motor DC.

Figura 15. Cinta transportadora implementada

El primer problema fue encontrar un material adecuado para la cinta, en primera instancia se pensó en el caucho como el material ideal, pero tras su adquisición y puesta en práctica, este no resultó ser capaz de soportar la tensión requerida, debido a la falta de dureza y firmeza del material. Por ello, se optó por utilizar otro caucho más resistente, encontrando otro problema, el peso: este nuevo material era demasiado pesado para ser convenientemente accionado por el motor DC. Finalmente, se decidió probar con tela asfáltica, teniendo con este material un resultado satisfactorio debido a su ligereza y resistencia cumpliendo así los requisitos de la cinta transportadora. Una vez elegido el material, faltaba realizar la unión para formar la cinta. La exigencia de disponer de una unión resistente provocó que se decidiera, utilizar pegamento junto a una costura en forma de cruz.

Para montar la cinta y los rodillos, se imprimen en 3D unos soportes con un agujero donde encaja el eje del rodillo, permitiendo su giro, y otros agujeros para ser atornillados sobre la madera. Las dimensiones de dicho soporte son 50 x 50 x 20 mm, con un agujero de 8’5 mm de diámetro para el eje del rodillo y dos agujeros de 4 mm de diámetro para los tornillos. Estas dimensiones se pueden observar mejor en las vistas de la pieza acotadas.

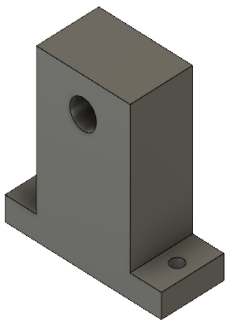


Figura 16. Soporte para los rodillos.

Además de estos soportes, dos por cada rodillo, se imprimieron dos conjuntos tensores por cada rodillo situado en uno de los extremos de las cintas. Estos conjuntos se componen de

tres piezas: dos soportes pasantes, donde se introduce la varilla que atravesará la pieza restante; y el soporte de rodillo tensor, que será el que con la ayuda de unas tuercas genere la tensión necesaria en la cinta. Esta configuración se puede ver en detalle en la figura 17. Las dimensiones del soporte pasante son 22 x 40 x 20 mm, mientras que las del soporte de rodillo tensor son iguales a las de los soportes de rodillos normales. Ambas piezas tienen un hueco de 8’5 mm de diámetro por el que pasa la varilla roscada y, en el caso de los soportes pasantes, dos agujeros de 4 mm de diámetro para los tornillos.

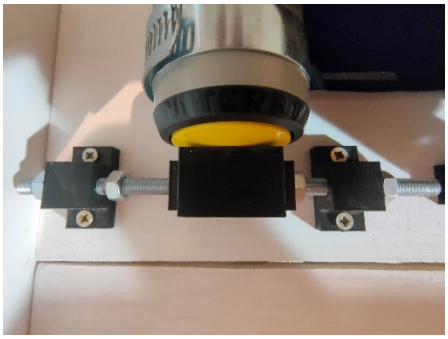


Figura 17. Conjunto tensor implementado.

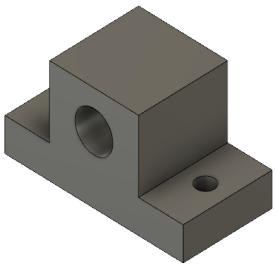


Figura 18. Soporte pasante.

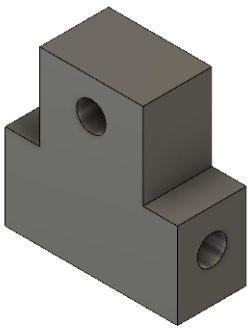


Figura 19. Soporte de rodillo tensor.

Los motores DC utilizados para la cinta son los mencionados en el apartado 5.3.2.2 para dicha función. Dichos motores se montan junto a las cintas transportadoras utilizando unos soportes impresos en 3D al igual que los rodillos. Estos soportes tienen las siguientes dimensiones: 55 x 73 x 32 mm, un agujero de 37 mm de diámetro para el motor y cuatro agujeros de 4 mm de diámetro para los tornillos.

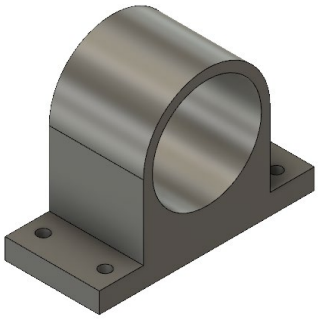


Figura 20. Soporte del motor de la cinta.

Una vez construida la cinta y elegidos los motores, se realiza la transmisión del movimiento del motor DC al rodillo. Esto supuso un nuevo problema, debido a que el eje del rodillo no gira solidario con el propio rodillo. La primera idea consistía en utilizar un sistema de polea-correa. Este sistema se descartó pese a ser funcional puesto que, para que la cinta transportadora girará correctamente, sobre todo en la zona de unión de la cinta, la tensión de la correa tenía que ser muy grande. Esto generaba problemas de sujeción del motor encajonado en un soporte impreso en 3D atornillado a la madera. La polea estaba impresa en 3D, para adaptarse a los motores y correa que ya teníamos, sus dimensiones son: 17 x 17 x 18 mm, con un espacio de 12 mm para la correa y un agujero con parte plana para el eje de 6’25 mm de diámetro.

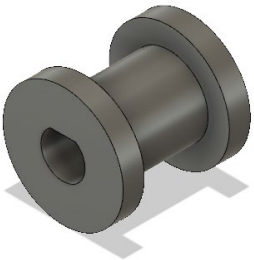


Figura 21. Polea del motor de la cinta.

Para solventar este problema, se implementó un nuevo sistema de transmisión directo del eje del motor al rodillo. Esto fue posible gracias a una nueva pieza en 3D. Dicha pieza consiste en un cilindro con una hendidura donde encaja el rodillo, la pieza además cuenta con un agujero para atravesar con una varilla roscada el rodillo y un hueco para el eje del motor,

generando así el giro solidario del eje del rodillo respecto al eje del motor. Esta pieza es un cilindro de 30 mm con un diámetro de 56’5 mm, la hendidura tiene un diámetro de 44’4 mm con un espesor de 3’05 mm, además del agujero de 8’5 mm para la varilla roscada y el hueco de 6 mm de diámetro con parte plana para el eje del motor.

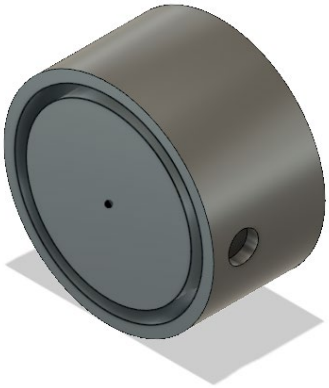


Figura 22. Cara delantera de la pieza de unión para la transmisión.

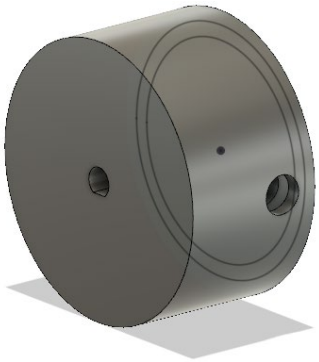


Figura 23. Cara trasera de la pieza de unión para la transmisión.

5.2.4. RAMPA

Los objetos caen de la cinta superior a la cinta inferior por medio de una rampa. Para su diseño se valoraron distintas opciones, con sus ventajas e inconvenientes, teniendo en cuenta el peso a transportar, la velocidad de caída de los objetos y el espacio físico disponible.

La idea principal consistía en una rampa plegable, que constaba de dos partes, una parte fija y otra móvil. La parte móvil tenía un eje conectado a un motor que con su giro hacía rotar la superficie móvil, de tal forma que al girar en sentido antihorario formaba una rampa completa y al girar en sentido horario se recogía sobre la parte fija, quedando una rampa con la mitad de longitud que permitiría el paso de los objetos por debajo. Ambas partes estaban unidas por una unión metálica a través de dos ejes metálicos que permitían el movimiento. En la implementación, el motor no tenía la fuerza suficiente para vencer la gravedad. Además, no

era posible sustituir el motor por uno más potente debido a las limitaciones de espacio físico, por lo que esta opción quedó descartada.

Puesto a que la rampa era imprescindible, se volvieron a revisar los diseños descartados y se optó por implementar una rampa fija, que pese a su gran inconveniente que es la caída libre de los objetos, no tenía limitaciones a nivel de construcción. Para intentar paliar dicha caída libre, preocupante especialmente en los objetos de vidrio, ya que son los más pesados, se posicionó en la pared una superficie de celulosa moldeada para realizar la función de parachoques. Además, se le añadieron unos rodillos de 10 cm integrados en la propia rampa para ayudar a los objetos más ligeros a caer. Una vez implementada, pese a los golpes de los objetos pesados al caer en la cinta inferior, y ayudados por la tensión de la propia cinta y la presencia del parachoques, el funcionamiento era como se esperaba.

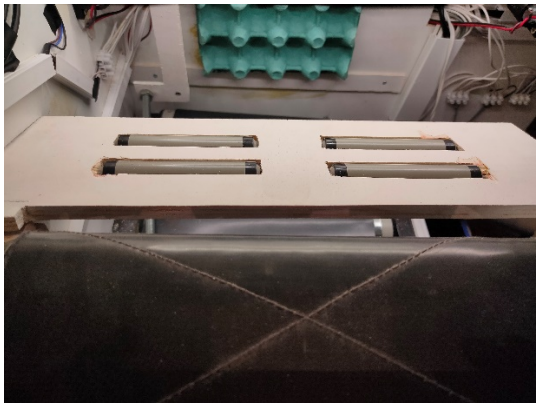


Figura 24. Rampa implementada.

5.2.5. DESVIADOR

El desviador es otro elemento fundamental, pues es el encargado de, una vez clasificados los objetos, tirarlos a su correspondiente cubo.

Al principio, la intención era poner un desviador por cada tipo de material clasificado, dejando el material de la última posición fijo. La primera idea era una pestaña accionada por un motor paso a paso, configurado con dos posiciones: la de permitir el paso y la desviar el objeto. Estos motores necesitan ser configurados con una serie de pulsos eléctricos, por lo que esta opción fue descartada al considerar que todos los elementos del prototipo serían controlados por el autómata, que no permitía esta posibilidad. Para controlarlo desde el autómata, se pensó en otro sistema que consistía en una biela - manivela que transforme el movimiento circular del motor en un movimiento rectilíneo del desviador. Su principal problema era la implementación, pues era un sistema tosco y las limitaciones de espacio físico eran muy grandes. Implementar este sistema suponía que la biela - manivela tenía que pasar

por el medio de la cinta transportadora, pudiendo entrar en conflicto con su movimiento. Además, en la primera posición de clasificación este sistema podía ser golpeado de lleno por la caída de los objetos desde la rampa.

Estos problemas obligaban a buscar otra solución, por lo que se diseñó un único desviador, que cubre todo el largo de la cinta y se acciona con unos motores DC controlables desde el autómata.

Para la construcción de este desviador único, se utilizó una varilla roscada que conecta una pieza de madera posicionada en cada lateral de la papelera con una rueda para reducir la fricción. Esta pieza de madera engancha en su parte superior con un raíl también de madera sobre el que se desplaza. El desplazamiento viene dado por dos motores conectados que transmiten el movimiento a través de un sistema de polea y correa dentadas. Cuando el desviador avanza, se activa el motor delantero que recoge la correa y hace que la pieza de madera avance; de igual forma se hace con el motor trasero para que el desviador retroceda. Este sistema también se probó con un único motor de doble sentido de giro, pero daba problemas. Al estar una polea dentada sin motor en la posición opuesta a este, cuando el movimiento del desviador dependía del giro del propio motor no había problema; sin embargo, cuando el movimiento dependía del giro de la polea dentada, el empuje del motor y la tensión de la correa no eran suficientes para que girara correctamente. Este sistema es simétrico, por lo que para su implementación se utilizaron cuatro motores y dos sistemas de polea-correa dentada. Los motores se implementaron utilizando unos soportes impresos en 3D, como se hizo con los motores de las cintas. Las dimensiones en este caso son de 39 x 54 x 31 mm, con un hueco de 25 mm de diámetro para el motor y los cuatro agujeros para los tornillos de 4 mm de diámetro.

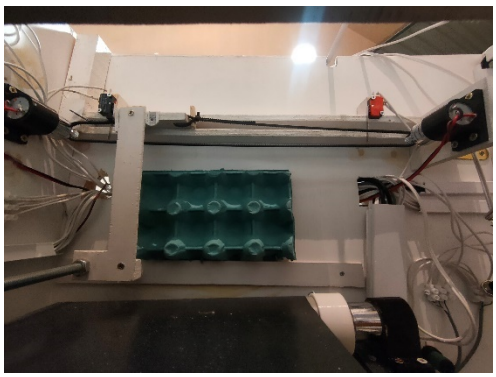


Figura 25. Desviador implementado.

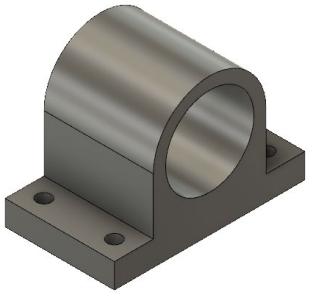


Figura 26. Soporte del motor del desviador.

Por su parte, las poleas y correas utilizadas están fabricadas en aluminio y son del tipo GT-2. Este tipo es muy utilizado en las impresoras 3D para el desplazamiento del extrusor sobre el plano de impresión.

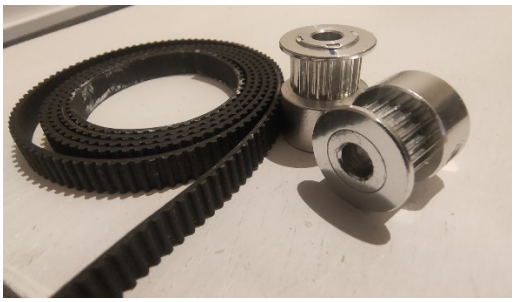


Figura 27. Correas y poleas GT-2.

5.2.6. CUBOS

Para finalizar el recorrido del objeto por toda la papelera, es necesaria una correcta salida de los cubos, cuyo fundamento es la inclinación del fondo sobre el que apoyan los cubos. Primero hay que diseñar los cubos, utilizando madera como material por las mismas razones que se utilizó en la construcción de toda la estructura. Las dimensiones de los cubos son de 35 (alto) x 27 (largo) x 25 (ancho) cm, con una capacidad de 22 L. Para que los cubos caigan por su propia gravedad con la inclinación dada, se le colocaron unas ruedas de 35 mm, todo esto se puede apreciar en la figura 28. Se escogieron de este tamaño porque, tras realizar varias pruebas, se comprobó que con ruedas de menor altura la propia inercia del cubo era incapaz de vencer la fuerza de fricción con la madera. No se montaron unas ruedas de mayor altura debido a las limitaciones de física del espacio.

Figura 28. Cubo implementado.

Con los cubos construidos, faltaba implementar el sistema de parada de estos. En primera instancia iban a ser retenidos por una pestaña, pero sucedería el mismo problema que con el desviador con los motores paso-paso, por lo que esta opción se descartó y se pensó en unos pistones eléctricos. El problema de estos es que eran muy caros y difíciles de conseguir. Por lo que, finalmente, al estar realizando pruebas de electrónica con un DVD se tuvo una nueva idea: construir una barrera móvil accionada por el motor del DVD.

Para construir dicha barrera móvil se colocó el DVD sobre la pared con una pieza de madera sobre la propia disquetera, que ejercía de freno para el cubo y que al subir por la acción del motor permitía el paso del cubo. Pero tras realizar varias pruebas se encontraron fallos: el motor no tenía la suficiente potencia, la punta de la pieza de madera colgaba un poco por la gravedad y rozaba con las paredes, etc. Para solucionar este problema y dar estabilidad a la barrera se decidió enfrentar dos DVDs, uno en cada pared lateral de cada rampa, de tal forma que la barrera de madera apoyará sobre los dos DVDs. Al probarlo se hizo notar un nuevo problema: el rozamiento entre madera y madera. Sin cubo este sistema de barrera móvil funcionaba perfectamente, pero al colocar el cubo los motores de los DVDs no eran capaces de vencer la fuerza de fricción producida por el rozamiento entre las dos maderas. Para solventar esto se introdujo una rueda en medio de la barrera. De aquí en adelante, estos elementos pasaran a denominarse dispositivo motorizado de sujeción de cubos, o de forma resumida, DVDs.

Con los cubos siendo capaces de rodar y un sistema de retención funcional, se hicieron pruebas de funcionamiento real, consistentes en introducir el cubo y activar las barreras móviles para permitir su paso, observando si estos salían de la papelera. Se detectaron dos problemas. El primer problema era que la inercia de los cubos hacía que al avanzar por la parte plana del fondo se fueran frenando y no llegaran a salir del todo. Para solventarlo se le dio una pendiente de 12º respecto a la horizontal, a la parte saliente del fondo, ampliando de esta

forma la inclinación de la estructura interna para la salida de los cubos. El segundo problema observado fue que los cubos no seguían una trayectoria recta hacia la salida de la papelera. Se propusieron varias soluciones. La primera en ponerse a prueba fue direccionar los cubos con unos raíles de caucho, pero la fricción de estos con las ruedas los frenaba. Entonces, se tenía que diseñar un sistema que los direccionara, pero sin frenarlo. La idea más sencilla era poner unos topes con ruedas, de forma que si el cubo chocaba con este tope rodará por él hasta salir de la papelera. Este diseño se descartó debido a que poner tantas ruedas en un tope tan largo era muy costoso. Para ello se pensó en direccionarlos con la repulsión de unos imanes, evitando así la fricción y necesitando solo cuatro imanes, dos por cada lado del cubo.

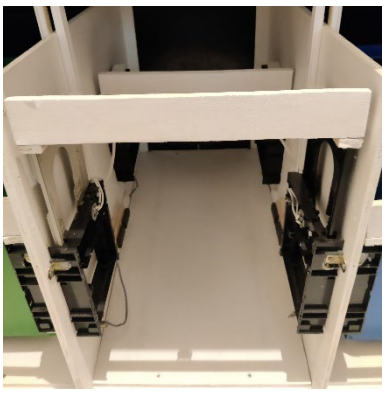


Figura 29. Sistema de salida de cubos implementado

5.3. DISEÑO ELÉCTRICO

Una vez montada toda la estructura y los elementos mecánicos correspondientes, se ha comenzado con el diseño e implementación de la parte eléctrica del prototipo.

La parte eléctrica se compone principalmente de tres grandes elementos: un autómata con su correspondiente módulo de ampliación; una Raspberry, la cual tiene asociada a su funcionamiento una tira LED; y una fuente de alimentación externa.

Para que el prototipo funcione es necesario conectarlo a una toma de corriente alterna de 230V. Voltaje que, como se explicará con detalle a continuación, se utilizará para alimentar los tres elementos principales del prototipo.

Por otro lado, todos los elementos de funcionamiento internos trabajan a diferentes voltajes de continua. Estos niveles de tensión se obtienen de la fuente de corriente continua que proporciona el autómata y de la fuente externa de alimentación AC/DC (a través de conversión mediante PCB).

5.3.1. CUADRO ELÉCTRICO

Para la correcta protección de todos los elementos del circuito, se ha implementado un cuadro eléctrico de protección, para el cual se han utilizado los siguientes elementos:

- 1 automático 1P+N 25A 6kA C, AEG
- 1 diferencial 2P 40A 30mA, AEG
- 3 automáticos 1P+N 10A 6kA C, General Electric

Estos tres elementos, dispuesto de la siguiente manera y con las siguientes utilidades:

- IGA (25 A): Alimentado a través de cable SCHUKO de 3x1,5mm de 2m, que se conectará a una toma de corriente. Este será el que proporcionará la alimentación general del prototipo.
- Diferencial: Actúa como elemento de protección ante posibles derivaciones a tierra/sobreintensidades.
- Magnetotérmicos (10A): Habilitan el funcionamiento de los tres elementos principales del prototipo: autómata, fuente de alimentación externa, enchufe de Raspberry y alimentación de la tira LED.

En la figura 30 se puede apreciar el montaje del cuadro eléctrico realizado.

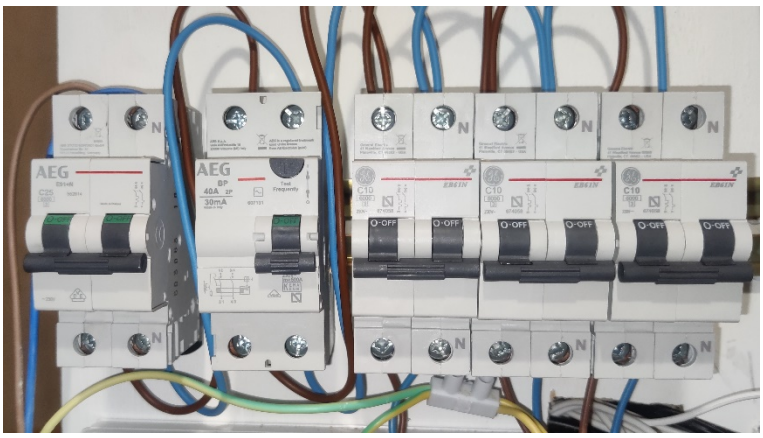


Figura 30. Conexionado cuadro eléctrico

5.3.2. AUTÓMATA

Para el prototipo, se ha utilizado un autómata Siemens S7-200 CPU 224 AC/DC/RLY. [14]

Las especificaciones eléctricas más relevantes de este modelo son las que se detallan a continuación:

- Alimentación del autómata: 230V AC
- Tensión de salida del autómata: 24V DC
- Tensión necesaria en las entradas: 24V DC
- Tipo de salida: Relé. Normalmente abierto a 24V DC y normalmente cerrado en circuito abierto.

A continuación, en la figura 31, se muestra una imagen del conexionado del autómata realizado

Figura 31. Conexionado autómata

5.3.2.1. ENTRADAS

La CPU 224 cuenta con 12 entradas digitales. Debido al número de sensores necesarios para la implementación del prototipo, se ha tenido que hacer uso de un módulo de expansión de E/S digitales de Siemens, concretamente el modelo EM223 [14]. Este se compone de 8 Entradas digitales

Las entradas del conjunto se han ocupado en su totalidad, con la siguiente disposición:

- 6 sensores industriales de 24V DC con tecnología de luz infrarroja, con alcance regulable máximo de 100 cm. DAWEI E3F-DS100C4 [15]

Figura 32. Sensor industrial IR 100 cm

- 3 sensores industriales de 24V DC con tecnología de luz infrarroja, con alcance regulable máximo de 300 cm. DAWEI E3F-DS300C4 [15]

Figura 33. Sensor industrial IR 300 cm

- 6 sensores industriales ópticos de 24V DC de luz visible LED, con alcance regulable máximo de 20 cm. VISOLUX GLV12-8-200/36/40b/115 [16]

Figura 34. Sensor industrial óptico 20 cm

- 4 entradas digitales provenientes de Raspberry. Indicando el resultado de la identificación del objeto a clasificar.
- 1 pulsador de Rearme

Figura 35. Pulsador de rearme

- 1 interruptor de enclavamiento para la Marcha

Figura 36. Interruptor de marcha

- 2 finales de carrera.

Figura 37. Final de carrera

Los 14 sensores industriales requieren alimentación externa; para ello se utiliza la propia fuente de 24V DC del autómata. La citada fuente también se utiliza para alimentar las entradas del pulsador de Rearme, del interruptor de Marcha y de los finales de carrera. Los elementos se han colocado en serie con la fuente de alimentación y la entrada del autómata (cada uno en su circuito respectivamente), permitiendo o no el paso de la corriente hacia el mismo.

Algunas zonas del prototipo, como la pared lateral de la zona del vidrio o el interior de todos los cubos, se han tenido que pintar de color negro. Esto es debido a que, por el rango de detección de los sensores IR y el color del prototipo (blanco, que permite el total rebote de la luz), los sensores estaban detectando constantemente. Al pintar estas zonas de negro, un color que debilita el rebote de la luz en el material se evita que esté el sensor detectando constantemente y solo lo haga cuando realmente debe.

Las señales de entrada provenientes de la Raspberry son de 3’3V. Por ello, se ha tenido que hacer una conversión de voltaje de a 3’3V DC a 24V DC a través de un circuito con un relé. Esta conversión se explica con mayor detalle en el apartado 5.4.1.2

Previamente a la obtención de los sensores industriales IR, para detectar la caída del objeto por la rampa y para detectar que el objeto (plástico o vidrio) se había posicionado correctamente, se habían utilizado unos sensores de Arduino de infrarrojos. El problema era que debido a su bajo rango de detección se debían utilizar dos sensores (uno a cada lado de la cinta) para poder detectar correctamente los objetos una vez posicionados. Además, estos sensores requerían una alimentación externa de 5VDC y la señal que proporcionaban también era de esta magnitud. Estos sensores no se han utilizado finalmente, pero sí se encuentran montados en el prototipo, como alternativa a los industriales, y se tuvieron en cuenta en el diseño de la PCB (para realizar la correspondiente conversión de voltaje).

A continuación, se muestra una tabla comparativa con el número de entradas utilizadas, así como la tensión de la señal proporcionada y su alimentación (en caso necesario).

Tabla 1. Entradas autómata con alimentaciones

NÚMERO DE ELEMENTOS	ENTRADAS UTILIZADAS	DESCRIPCIÓN	TENSIÓN PROPORCIONADA	ALIMENTACIÓN
9	8	Sensores Industriales IR	24V DC	24V DC
6	6	Sensores Industriales Ópticos	24V DC	24V DC
-	4	Comunicación con Raspberry	3’3 V DC	-
2	2	Finales de Carrera	24V DC	-

1	1	Interruptor de marcha	24V DC	-
1	1	Pulsador Rearme	24V DC	-
5	3	Sensores Arduino IR	5V DC	5V DC

5.3.2.2. SALIDAS

La CPU 224 cuenta con 10 salidas digitales. Debido al número de actuadores necesarios en el prototipo, se ha hecho uso de un módulo de ampliación de E/S digitales de Siemens, concretamente el modelo EM223. Este módulo proporciona 8 salidas digitales adicionales.

Las salidas en su conjunto se han ocupado en su totalidad con la siguiente disposición:

- 2 salidas para controlar el doble sentido de giro del desviador, compuesto por 4 motores de 24VDC y 200 rpm. ZHENGKE ZGA28RP. [17]

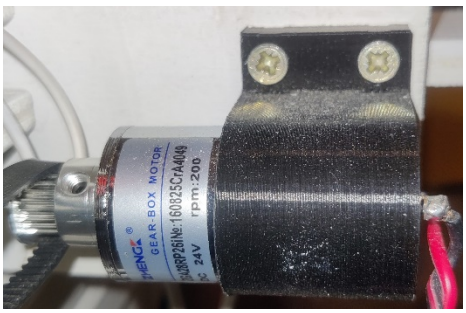


Figura 38. Motor desviador

- 1 salida para controlar el movimiento de la cinta inferior, compuesta por 1 motor de 24V DC y 50 rpm. ZHENGKE ZGA37RG [18]

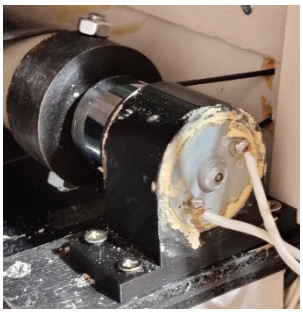


Figura 39. Motor cinta inferior

- 1 salida para controlar el movimiento de la cinta superior, compuesta por 1 motor de 24V DC y 60 rpm. ZHENGKE ZGB37RG [19]

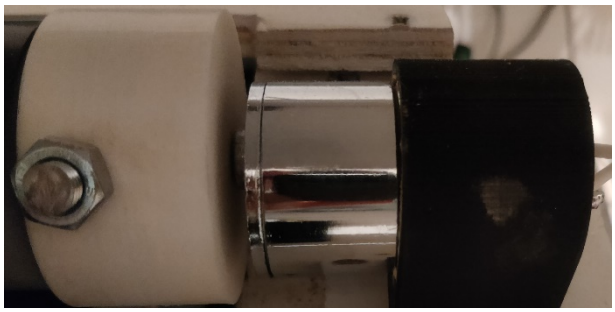


Figura 40. Motor cinta superior

- 12 salidas para controlar el doble sentido de giro de 6 dispositivos motorizados para la sujeción de cubos. Estos están compuestos por 12 DVDs (extraídos de PC' s de sobremesa) de diferentes fabricantes, todos ellos con motores de 5V DC.

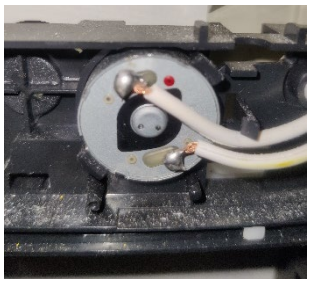


Figura 41. Motor DVD

- 1 salida que enciende un piloto luminoso rojo de emergencia.

Figura 42. Piloto luminoso de emergencia

Las dos cintas transportadoras y el piloto luminoso de emergencia se alimentan directamente desde la salida del autómata. En cambio, para la alimentación de los motores de 5V, para controlar el desviador con doble sentido de giro y para comunicarse el autómata con la Raspberry, se ha tenido que diseñar un circuito acondicionador de señales, que se explicará con más detalle en el apartado 5.4.1.1.

A continuación, se muestra una tabla resumen de las salidas utilizadas, su tensión de funcionamiento y se indica qué motores son de doble sentido de giro.

Tabla 2. Salidas autómata con alimentaciones

NÚMERO DE ELEMENTOS	**SALIDAS UTILIZADAS**	**DESCRIPCIÓN**	**TENSIÓN NECESARIA**	**DOBLE SENTIDO DE GIRO**
12	12	Motores DVDs	5V DC	Sí
2	2	Motores Cintas Transportadoras	24V DC	No
4	2	Motores del Desviador	24V DC	Sí

-	1	Comunicación con Raspberry	3'3V DC	-
1	1	Piloto luminoso de emergencia	24V DC	-

5.3.3. RASPBERRY

La Raspberry utilizada es una Raspberry Pi3 Model B+ [20] a la cual se le ha acoplado una cámara Raspberry Pi Camera V2.1 [21] para el sistema de visión artificial.

Las características eléctricas más relevantes de la citada placa con las siguientes:

- Alimentación de funcionamiento: 5V.
- Corriente de funcionamiento: 2'5A.
- Voltaje de los GPIO: 3'3V.
- Tensiones de alimentación que puede proporcionar: 3'3V y 5V.

La Raspberry se alimenta a través de un transformador AC/DC que es capaz de proporcionar 5V/3A. Este transformador se ha conectado a una base SCHUKO sobre pared blanca, conectada a su vez a uno de los magnetotérmicos del cuadro eléctrico. Además, para mejorar la toma de imágenes del sistema de visión artificial, se ha iluminado la zona de captación de imagen con una tira LED de 12V, alimentada a través de un enchufe SCHUKO de sobre pared blanco. El citado conexionado, se puede observar en la figura 43.

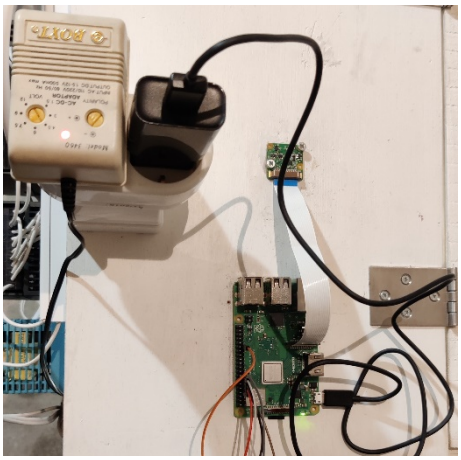


Figura 43. Alimentación Raspberry y Tira LED

Para comunicarse con el autómata, la Raspberry hará uso de cinco pines. Cuatro de ellos serán de comunicación Raspberry-Autómata, se usarán para indicar el tipo de material del objeto analizado. El último pin será de comunicación Autómata-Raspberry, para indicarle cuándo debe comenzar a ejecutar el código de identificación de objetos.

A continuación, se muestra una tabla resumen de los pines utilizados en la Raspberry

Tabla 3. GPIOs utilizados en la Raspberry

NÚMERO PIN	NOMBRE	TIPO	DESCRIPCIÓN
7	GPIO 4	SALIDA	El objeto identificado es de cartón
9	GND	GND	Conexión de la tierra de la Raspberry con el resto de las tierras
11	GPIO 17	SALIDA	El objeto identificado es de plástico
13	GPIO 27	SALIDA	El objeto identificado es de vidrio
15	GPIO 22	SALIDA	No se reconoce el tipo de material del objeto
29	GPIO 5	ENTRADA	Inicio del proceso de identificación del objeto

5.3.4. CABLEADO

Para la interconexión de todos los elementos entre sí se han hecho uso de diversos cableados.

Para conectar las diferentes salidas del autómata a la PCB y a los actuadores se ha hecho uso de cable paralelo de 0'75 mm^2 de sección. Para conectar los diferentes sensores a las entradas del autómata, se ha utilizado el propio cable del sensor. Cuando este no ha tenido la longitud suficiente, se ha ampliado mediante regletas eléctricas con cable paralelo de 0'75 mm^2 de sección. Para conectar el pulsador y el interruptor, también se ha hecho uso de este tipo de cable.

Figura 44. Cable paralelo de 0.75 mm2

Para conectar los GPIO de la Raspberry, se ha utilizado cable Dupont hembra-macho, para posteriormente alargarlo mediante regletas eléctricas con Cable Paralelo de 0'75 mm^2 hasta llegar al autómata.

Figura 45. Cable Dupont

Para conectar los elementos de protección del cuadro eléctrico entre sí y con los elementos a los que alimentan (autómata, fuente y enchufe Raspberry) se ha utilizado cable de instalación de 1’5 mm^2.

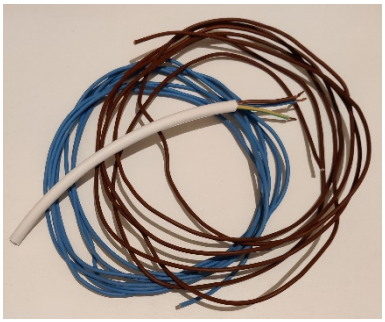


Figura 46. Cable de instalación 1'5 mm^2

Para alimentar la Raspberry, así como la tira LED de 12VDC, se ha utilizado el propio cable de los transformadores AC/DC con los que se alimentan.

A continuación, se proporciona una tabla resumen de todo lo descrito anteriormente:

Tabla 4. Cableado utilizado para cada elemento

ELEMENTOS CONECTADOS	**TIPO DE CABLE**
Cuadro eléctrico	Instalación 1’5 mm^2
Autómata, Fuente de Alimentación y Enchufe Raspberry	Manguera 3x1’5 mm^2
Sensores Industriales IR	Cable manguera sensor + Cable paralelo 0’75 mm^2
Sensores Industriales Ópticos	Cable manguera sensor + Cable paralelo 0’75 mm^2

Finales de carrera	Cable paralelo 0’75 mm^2
Interruptor de marcha	Cable paralelo 0’75 mm^2
Pulsador de rearme	Cable paralelo 0’75 mm^2
Comunicación Raspberry	Cable Dupont Hembra-Macho + Cable paralelo 0’75 mm^2
Motores DVDs	Cable paralelo 0’75 mm^2
Motores Cintas transportadoras	Cable paralelo 0’75 mm^2
Motores desviadores	Cable paralelo 0’75 mm^2
Piloto de emergencia	Cable paralelo 0’75 mm^2
Tira LED 12V DC	Cable Transformador AC/DC 12V
Raspberry 5V DC	Cable Transformador AC/DC 5V

5.4. DISEÑO ELECTRÓNICO

Debido a que en el prototipo en cuestión existen diferentes voltajes, tanto para establecer comunicación entre Autómata y Raspberry, como para alimentar los diferentes actuadores desde el autómata, se ha decidido implementar una Placa de Circuito Impreso (PCB).

5.4.1. COMPONENTES

La placa está compuesta por los siguientes elementos:

- 15 relés de 5 pines de 24V DC [22]

Figura 47. Relé de 24V.

- 4 relés de 5 pines de 3V DC [22]

Figura 48. Relé de 5V.

- 1 circuito integrado LM324 [23]

Figura 49. Circuito integrado LM324.

- 1 regulador de tensión LM7805 de 5V [24]

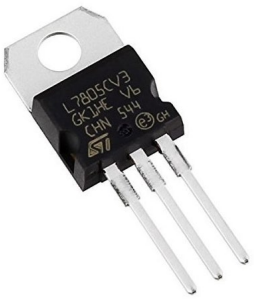


Figura 50. Regulador de tensión LM7805. [25]

- 1 regulador de tensión LM3940 de 3V [26]

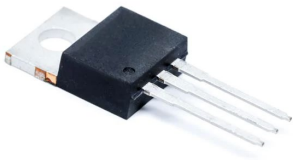


Figura 51. Regulador de tensión LM3940. [26]

- 1 regulador de tensión LM7812 de 12V [24]

Figura 52. Regulador de tensión LM7812 [27]

- 2 condensadores electrolíticos de 0’47 uF y 33 uF

Figura 53. Condensador electrolítico de 0’47uF. [28]

- 2 condensadores de 330nF y 100nF

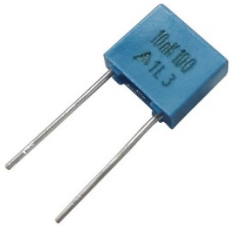


Figura 54. Condensador de 330 nF. [29]

- 26 clemas de 2 bornes

Figura 55. Clema de 2 bornes.

- 19 diodos 1N4007 [30]

Figura 56. Diodo 1N4007.

- 4 transistores BC550 [31]

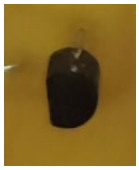


Figura 57. Transistor BC550.

La alimentación de entrada de la PCB viene proporcionada a través de 1 Fuente de alimentación, concretamente la MDR-100-24 de la marca Mean Well.

Figura 58. Fuente de alimentación

A continuación, se explica el funcionamiento de cada elemento.

5.4.1.1. RELÉS DE 24V DC

Doce de estos relés se utilizan para controlar los motores de 5V DC de los dispositivos motorizados de sujeción de los cubos (DVDs). Se deben utilizar, ya que, las salidas del autómata proporcionan 24V DC. Para ello, se hace uso de estos Relés de 24V DC de esta manera:

- La salida del autómata se conecta a uno de los bornes de la bobina, mientras que el otro se pone a tierra.
- El contacto Normalmente Cerrado del Relé (NC) se conecta a Tierra
- El contacto Normalmente Abierto del Relé (NO) se conecta a 5V DC.
- El común del Relé se conecta a uno de los bornes del motor. Mientras que el otro borne del motor va al común del otro Relé de 24V con la misma configuración.

De esta manera, es como si se trabajara con "pares de relés". Cuando se activa una salida para darle un sentido de giro al motor, uno de los relés conmutará, poniendo un borne del motor a 5V. Por otro lado, nos aseguraremos de que la otra salida esté desactivada, poniendo así el otro borne del motor a tierra. De esta manera, se consigue el giro del motor en uno de los sentidos. Si intercambiamos el valor de las entradas, resultará en el mismo funcionamiento anteriormente descrito, pero intercambiando los bornes del motor que están a 5V y a tierra. De esta manera, se consigue que el motor gire en el otro sentido.

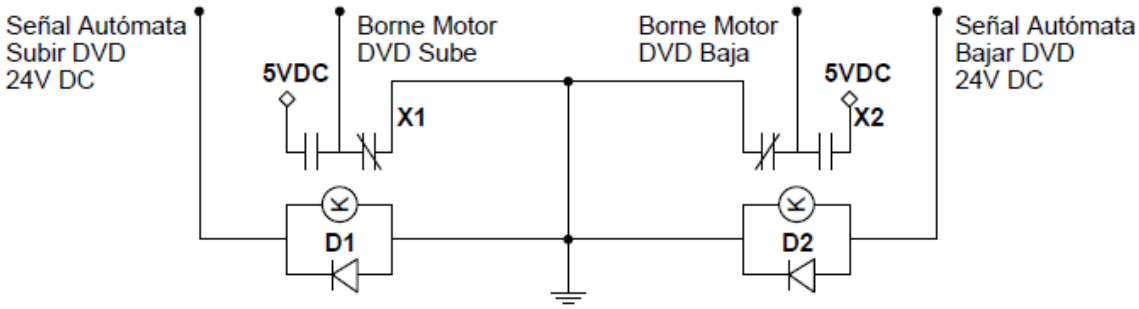


Figura 59. Conexionado de motores de DVDs a Relés de 24V

Dos relés más se utilizan para controlar los motores de 24V DC, que permiten el avance y retroceso del desviador. Debido a que el relé interno del autómata conmuta entre 24V y circuito abierto, no es posible controlar un motor de doble sentido de giro directamente desde el propio autómata. Para ello, se hace uso de estos Relés de 24V DC de la misma manera que en el apartado anterior (los motores de 5V). La única diferencia, en cuanto a conexionado se

refiere, es que el contacto normalmente abierto del relé (NO) se conectará a 24 VDC, debido a la alimentación que solicitan los motores.

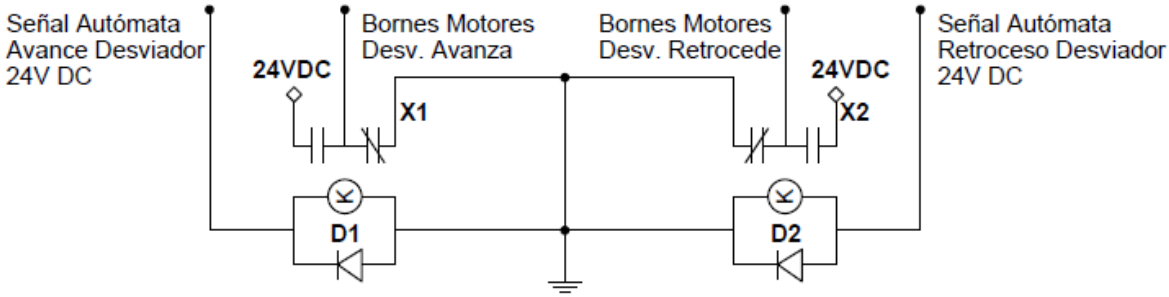


Figura 60. Conexionado desviador a Relés 24V

El otro relé se utiliza para comunicar el autómata con la Raspberry. Esto es necesario debido a que los GPIO de la Raspberry solo permiten una tensión de entrada de 3'3V DC, mientras que el autómata proporciona 24V DC en sus salidas. El conexionado del relé es idéntico a los dos apartados anteriores, con la excepción de que, en este caso, el normalmente abierto del relé se conectará a 3'3V DC.

De esta manera, cuando el autómata active la señal para que se comience la clasificación, la bobina conmutará y en el común del relé pasará a haber 3'3V DC, que es el voltaje de entrada al GPIO correspondiente de la Raspberry.

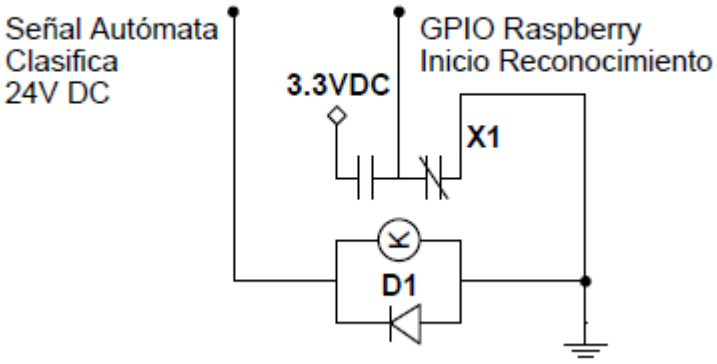


Figura 61. Conexionado señal "clasifica" a Relé 24V

5.4.1.2. RELÉS DE 3V DC

Estos relés se han utilizado para comunicar la Raspberry con el autómata. Esto es necesario debido a que el autómata admite entradas de 24VDC y la Raspberry únicamente es capaz de proporcionar 3'3V DC a través de sus GPIO.

Para ello, se ha hecho uso de este relé de 3VDC de la siguiente manera:

- La salida de la Raspberry se conecta a la base de un transistor NPN, el cual tiene el emisor a tierra y el colector a uno de los bornes de la bobina.
- El otro borne de la bobina está conectado a 3'3V.
- El contacto Normalmente Cerrado del Relé (NC) se conecta a tierra.
- El contacto Normalmente Abierto del Relé (NO) se conecta a 24 V.
- El común del Relé se conecta a una entrada del autómata.

De esta manera, cuando la Raspberry active uno de los GPIO, este tendrá un voltaje de 3'3V, que será el que le llegue a la base del transistor. De esta manera, el transistor se cerrará, permitiendo el paso de corriente desde su colector hasta su emisor (alimentando así la bobina a 3'3V). Una vez que la bobina haya conmutado, el común pasará a tener 24V, tensión que le llegará a la entrada del autómata.

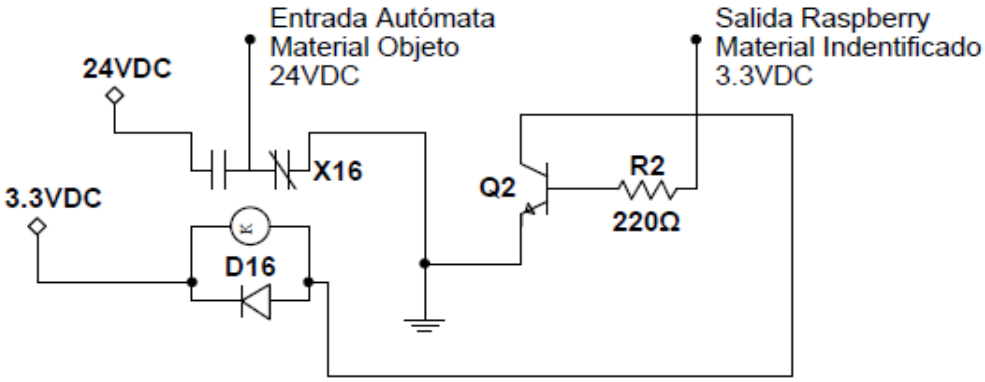


Figura 62. Conexionado relés 3V

5.4.1.3. CIRCUITO INTEGRADO LM324

Este integrado se incluyó inicialmente para poder utilizar unos módulos de sensores infrarrojos de Arduino, cuando no se disponía de suficientes sensores industriales infrarrojos para cubrir completamente todas las necesidades. Aunque finalmente llegaron los sensores industriales, y este circuito no se utiliza actualmente, se dejó en el diseño de la PCB y es totalmente utilizable en caso de fallo de algunos de los sensores industriales.

Los sensores infrarrojos de Arduino se alimentan a 5V DC y son de lógica negada, con señal de salida de 5V. Como se ha indicado anteriormente, el autómata requiere de una tensión de 24VDC en sus entradas para realizar una correcta lectura del estado de los sensores. Por esta razón se tenía que realizar una conversión de voltaje, de 5V DC a 24V DC.

La primera opción que se planteó para realizar esta conversión fue la de hacerlo mediante relés, como todas las conversiones de voltaje realizadas hasta ahora. Los sensores en cuestión necesitan de una alimentación de 5VDC. El problema que surgió fue que, al conectar la salida del sensor a un extremo de la bobina y el otro extremo de la misma a tierra (siendo esta tierra la misma que la de alimentación, necesario para una referencia correcta), debido a la baja resistencia de la bobina, la salida del sensor se referenciaba a tierra. Al ser de lógica negada, el sensor era como si estuviera detectando constantemente (0V).

Por este motivo, se tuvo que incluir el integrado LM324, que contiene varios amplificadores operacionales que se configurarán como comparadores, para realizar la conversión lógica de voltaje. El circuito integrado en cuestión se conecta de la siguiente forma:

- Vcc: 24V DC
- -Vcc: 0V
- V+: Salida del sensor Arduino
- V-: Referencia de comparación, fijada en 3.3V
- OUT: Entrada del autómata

De esta manera, si el sensor no está detectando, ofrece una salida de 5V, y por tanto, la salida del comparador se saturaría a +Vcc, que es 24V. Cuando el sensor detecta ofrece a su salida 0 V, siendo este voltaje menor que el de referencia, por lo que la salida del comparador se saturaría a -Vcc, que es 0V.

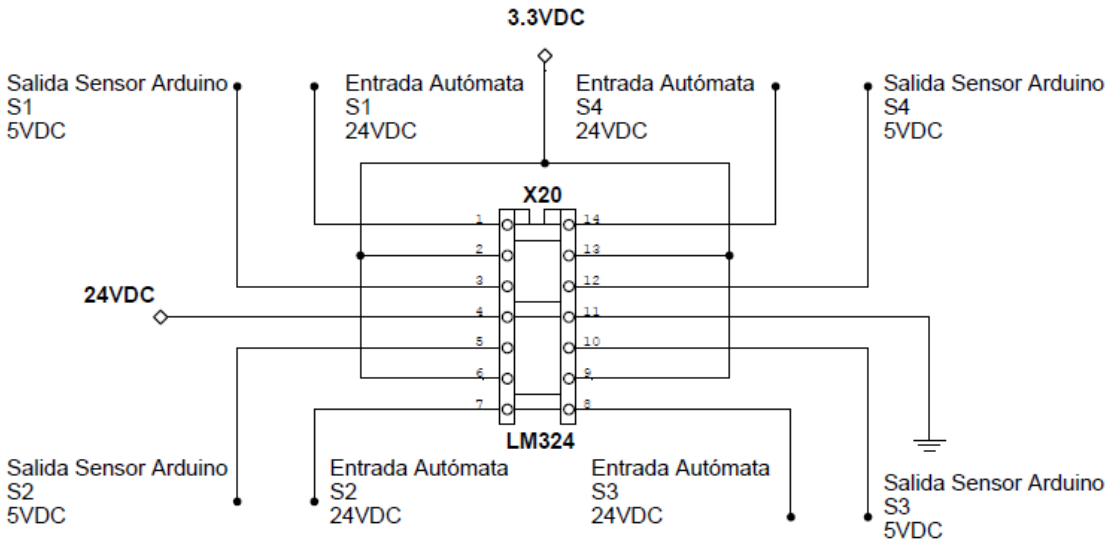


Figura 63. Conexionado LM324

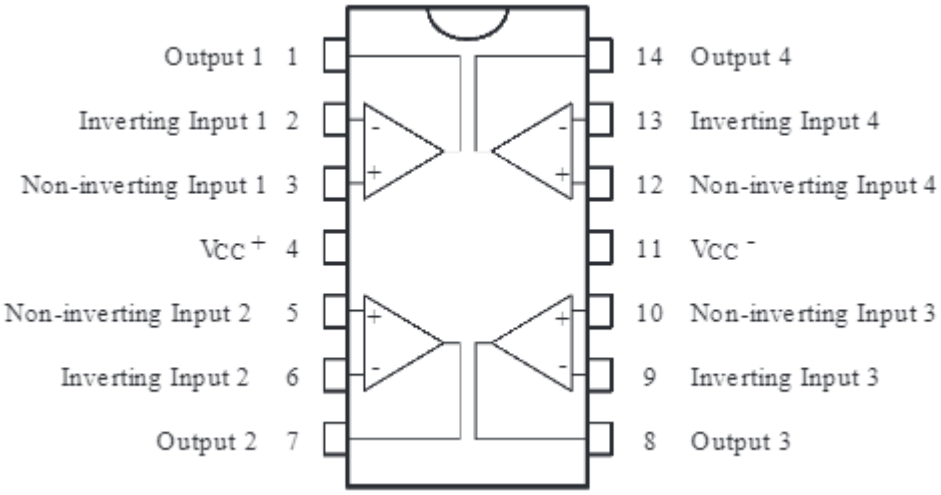


Figura 64. Esquema interno LM324 [23]

5.4.1.4. REGULADORES DE TENSIÓN

Para el correcto funcionamiento de la placa y del prototipo en general, se han utilizado tres reguladores de tensión:

- Regulador de 12V: regulador de tensión que convierte los 24V de entrada a la placa (proporcionados por la fuente de alimentación externa) a 12V. Este voltaje, es el voltaje de entrada para el regulador de 5V y se pensaba utilizar para alimentar la tira LED que dispone el prototipo. Debido al alto consumo de los motores de 5V (corriente en parte proporcionada por este regulador) en el arranque y los 300 mA de consumo constante de la tira LED, se ha decidido no alimentar la tira LED mediante este regulador. Al realizar pruebas, se observó que, si los motores de 5V solicitaban mucha corriente (bien por estar en su tope o bien por un pico en el arranque) la tira LED se apagaba, ya que el regulador no era capaz de dar tanta corriente a la vez. Para conseguir un voltaje estable a la salida del regulador, la entrada del mismo se ha conectado en paralelo con un condensador de 330nF y la salida en paralelo con un condensador de 100nF.

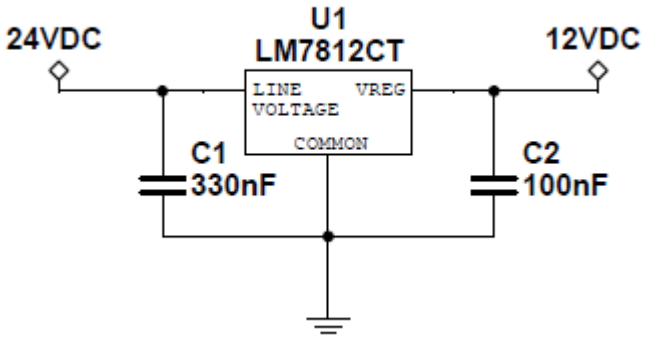


Figura 65. Conexionado LM7812

- Regulador de 5V: Se utilizó para dar la tensión necesaria para los motores de los dispositivos motorizados de sujeción de cubos (DVDs) y como tensión de entrada para el regulador de 3'3V.

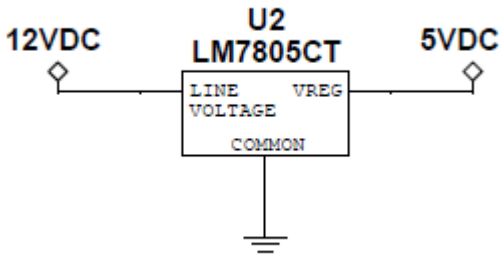


Figura 66. Conexionado LM7805

- Regulador de 3'3V: Se utilizó para para alimentar los transistores, el relé de comunicación entre Autómata y Raspberry y como tensión de referencia en el LM324. Para conseguir un voltaje estable a la salida del regulador, la entrada del mismo se ha conectado en paralelo con un condensador electrolítico de 0'47uF y la salida en paralelo con un condensador electrolítico de 33uF.

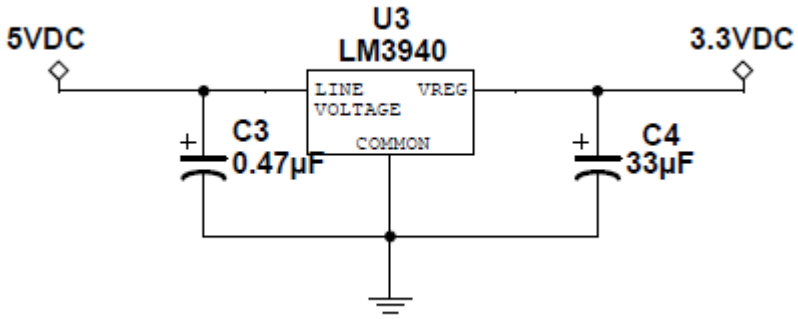


Figura 67. Conexionado LM3940

Para evitar un sobrecalentamiento de los reguladores de tensión, se ha intentado que el voltaje de entrada a cada uno sea lo más bajo y próximo posible al de salida. Por ello se ha hecho, como indicado, de manera escalonada: de 24V a 12V, de 12V a 5V y de 5V a 3'3V. Adicionalmente, se han equipado los tres reguladores de tensión de unos disipadores de calor metálicos.

5.4.1.5. DIODOS DE PROTECCIÓN, TRANSISTORES Y CLEMAS

Además de todos los elementos citados anteriormente, en paralelo a cada bobina de relé se han incluido unos diodos de protección, para evitar que posibles corrientes inversas afecten a los dispositivos conectados.

Por otra parte, como se ha explicado anteriormente, los transistores se han utilizado como amplificadores de corriente para las salidas de la Raspberry. De esta manera, se asegura que la corriente que recibe la bobina del relé de 3V es la suficiente para que el relé conmute.

Finalmente, para poder conectar las salidas y entradas de los autómatas, las salidas y entradas de la Raspberry, así como las salidas y entradas de las diferentes alimentaciones, se han utilizado conectores de entrada dobles.

5.4.2. CARACTERÍSTICAS E IMPLEMENTACIÓN DE LA PLACA

Todo el conexionado eléctrico, así como la utilidad de cada componente se ha explicado en el apartado anterior. A continuación, se muestra una imagen del conexionado realizado en Multisim [32] y Ultiboard [32].

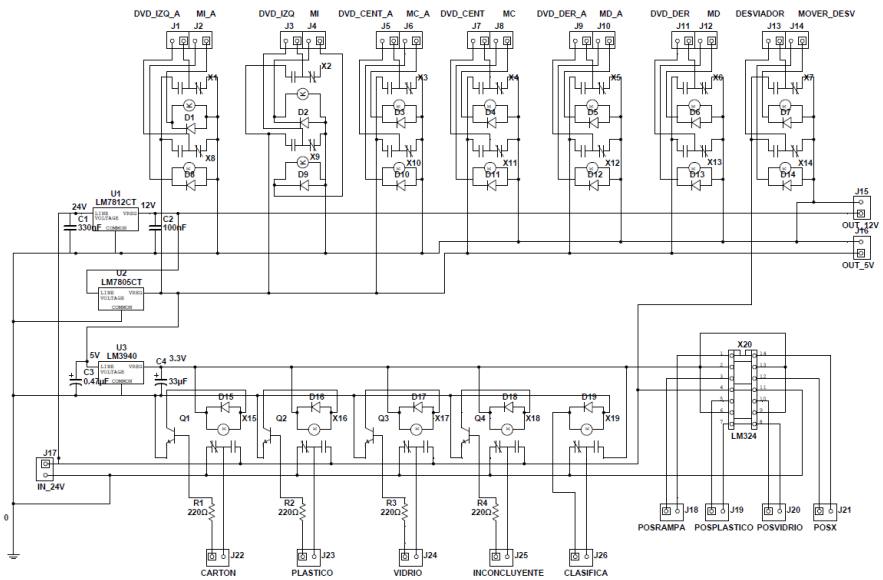


Figura 68. Conexionado de la PCB en Multisim

Una vez terminado el diseño en Multisim, este se envió al Servicio de Electrónica de la ULL. Ellos se encargaron de transferir el diseño a Ultiboard, para posicionar los diferentes componentes (acordes a unas especificaciones dadas) y realizar el correspondiente ruteo.

La placa cuenta con un total de 76 componentes, que dan lugar a 228 pines y 168 conexiones diferentes, todas ellas con un ancho de pista de 0'5 mm. Debido al alto número de pistas generadas, se han tenido que implementar 4 vías para el conexionado de dos componentes mediante cable externo a la placa. Además del conexionado eléctrico, se han dispuesto 4 agujeros de montaje en las esquinas de la placa, para su anclaje al prototipo mediante tornillos a la madera. El tamaño final de la placa es de 20x12'5 cm.

Como se explicó anteriormente, para el conexionado de los elementos externos a la PCB se han utilizado clemas dobles de conexión, siguiendo siempre las siguientes reglas para facilitar el conexionado:

- Todas las clemas en las que se introduce o saca alimentación, están dispuestas de tal manera que el borne izquierdo será el de la alimentación y el derecho el de tierra.

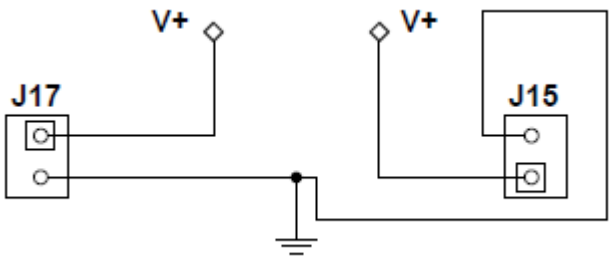


Figura 69. Conexionado clemas alimentación

- Las clemas que establecen la comunicación Raspberry-Autómata y Autómata-Raspberry están dispuestas de tal manera que el borne izquierdo es al que se conecta la señal de entrada (la que activa el relé correspondiente) mientras que el borne derecho es al que se conecta la señal de salida (la que está conectada al común de cada relé).

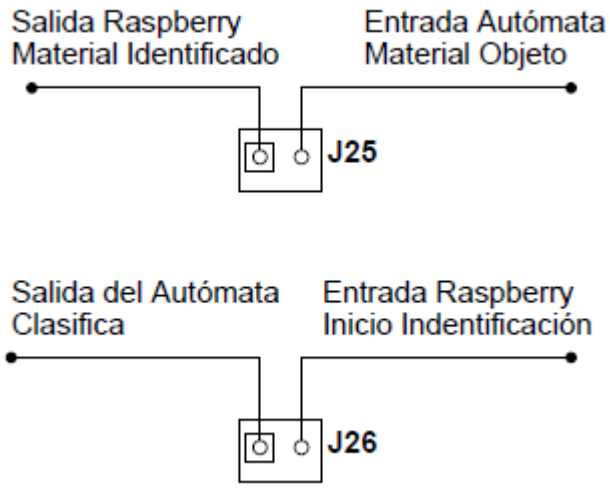


Figura 70. Conexionado clemas de comunicación

- Para controlar los motores de doble sentido de giro se dispone de 2 clemas para cada motor, que se conectan de la siguiente manera.

    - Clema de la izquierda: El borne izquierdo es el que proviene del autómata con la señal de subida/avance del actuador. El borne derecho es el que proviene del autómata con la señal de bajada/retroceso del actuador.

    - Clema de la derecha: El borne izquierdo es el que se conecta al motor para permitir la subida/avance del actuador. El borne derecho es el que se conecta al otro borne del motor para permitir la bajada/retroceso del actuador.

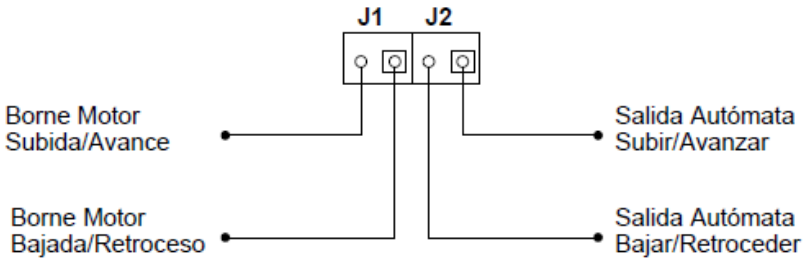


Figura 71. Conexionado clemas motores doble sentido de giro

- Las clemas que conectan los sensores de Arduino con el LM324 están dispuestas de tal manera que el borne izquierdo es al que se conecta la señal proveniente de la salida del sensor (V+) y el derecho es al que se conecta la señal que se dirige hacia la correspondiente entrada del autómata (OUT).

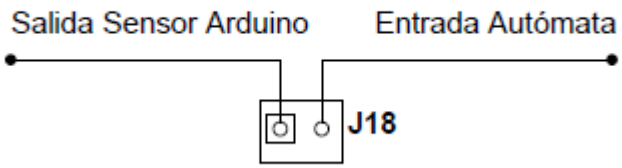


Figura 72. Conexionado clemas entrada/salida LM324

Cabe destacar que todas las referencias de izquierda/derecha se toman mirando la placa por el lado en el que se encuentra cada conexión.

La implementación de la placa se realizó en el laboratorio del Servicio de Electrónica de la ULL mediante una máquina de control numérico, concretamente la CNC Protomat S62, LPKF. Una vez impresa la placa, se procedió a darle un lacado para evitar la oxidación del cobre y finalmente a soldar todos los componentes.

Figura 73. CNC Protomat S62, LPKF [33]

En la imagen a continuación, se puede observar una imagen del resultado de la PCB impresa.

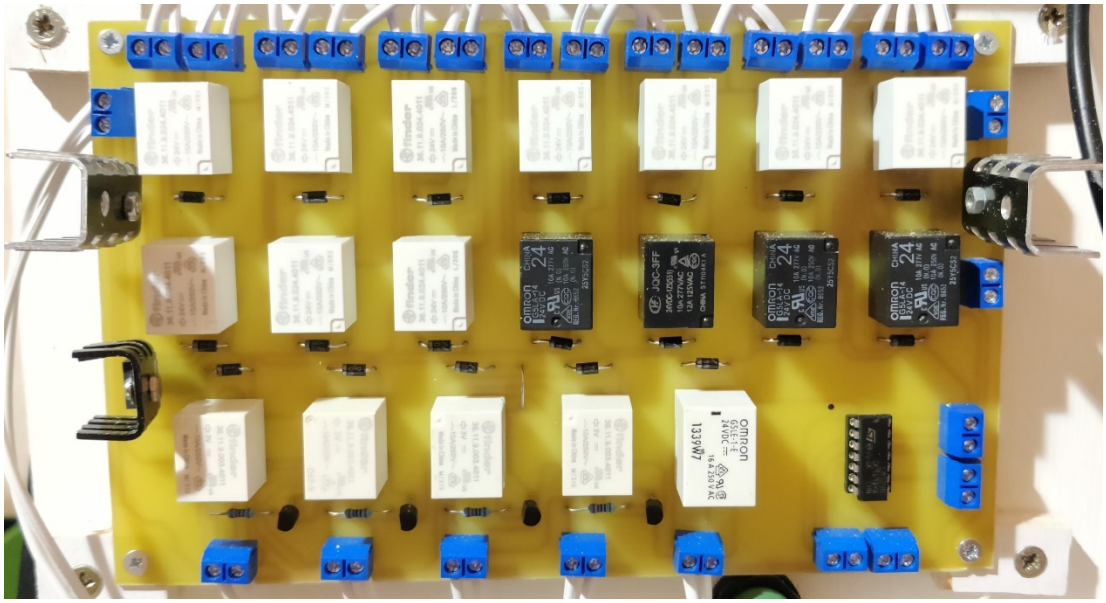


Figura 74. PCB terminada

Información más detallada de la placa realizada se puede encontrar en el Anexo IV.

6. PROGRAMACIÓN

Otro aspecto importante del prototipo presentado es su parte software, dividida en las siguientes partes: primero, la programación del autómata en KOP, necesaria para automatizar el proceso, segundo, el entrenamiento y ajuste de una Red Neuronal capaz de clasificar imágenes de diferentes tipos de objetos, y, por último, el Bot realizado en Python para automatizar las comunicaciones entre Raspberry y Autómata.

6.1. PROGRAMACIÓN DEL AUTÓMATA

Debido a que el proceso a programar es un proceso muy secuencial (descrito en el apartado de Funcionamiento), pero el autómata ejecuta todo el código de manera simultánea, se ha tenido que hacer uso de marcas para la secuenciación, además de organizar el código en diversas subrutinas para su facilitar su lectura y entendimiento. Para realizar todo el código KOP, se ha utilizado el programa Step 7 MicroWin SP4 [34].

En total se utilizaron 27 marcas (M0.0-M4.2), 10 subrutinas (sin contar el OB1), 5 temporizadores TONR (T0-T4), 2 temporizadores TON (T32, T33), las 22 entradas disponibles de la CPU y módulo de expansión (I0.0-I1.5, I2.0-I2.7) y las 18 salidas disponibles de la CPU y módulo de expansión (Q0.0-Q1.1, Q2.0-Q2.7).

El programa comienza en el bloque Principal (OB1), en donde se inicializan todos los temporizadores y marcas y se da paso a la primera subrutina del programa (SBR0-Primer Ciclo). En esta subrutina, se realizan todas las comprobaciones iniciales necesarias, tales como la posición del desviador y de los cubos; y se actúa sobre las salidas en caso de que estos elementos no estén en las posiciones iniciales deseadas. Una de las tareas que se realiza en esta subrutina es comprobar si existe cubo en la posición delantera, y en caso de que no haya, pero sí haya cubo en la posición trasera, este se traerá hacia delante.

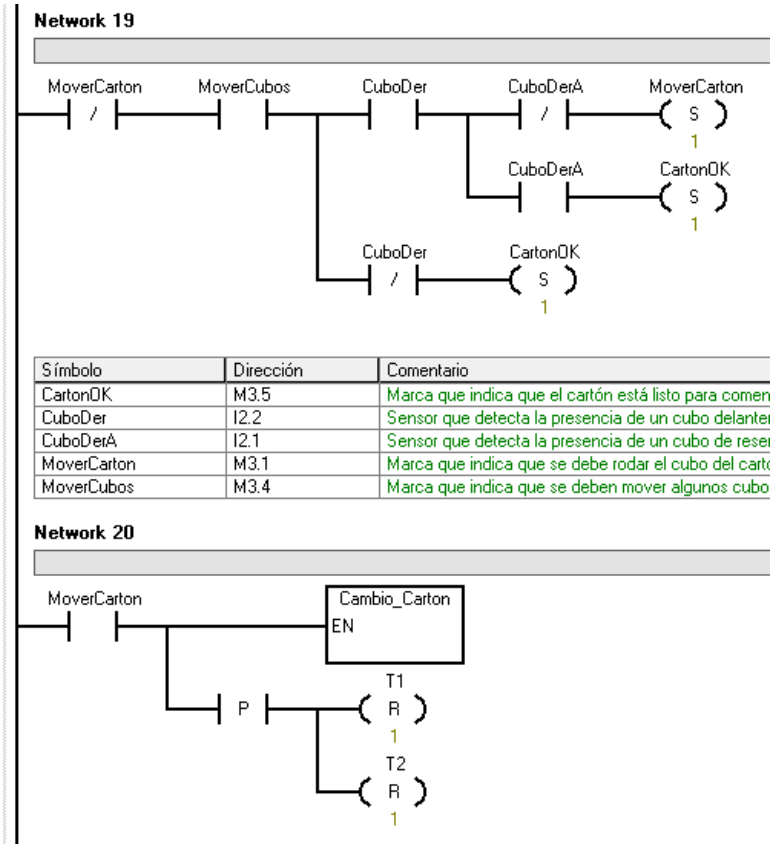


Figura 75. Segmentos que indican el desplazamiento hacia delante del cubo de cartón

Además, como no se tiene información de la posición de los dispositivos motorizados de sujeción de cubos, para asegurar un correcto funcionamiento del programa, se activa su bajada durante unos segundos. Cabe destacar que esta bajada se realiza solo si se accede al Ciclo Inicial desde el primer ciclo del autómata o bien desde una emergencia ocasionada por una incorrecta salida de cubos. De esta manera, no se están forzando los motores de los dispositivos de sujeción de cubos innecesariamente.

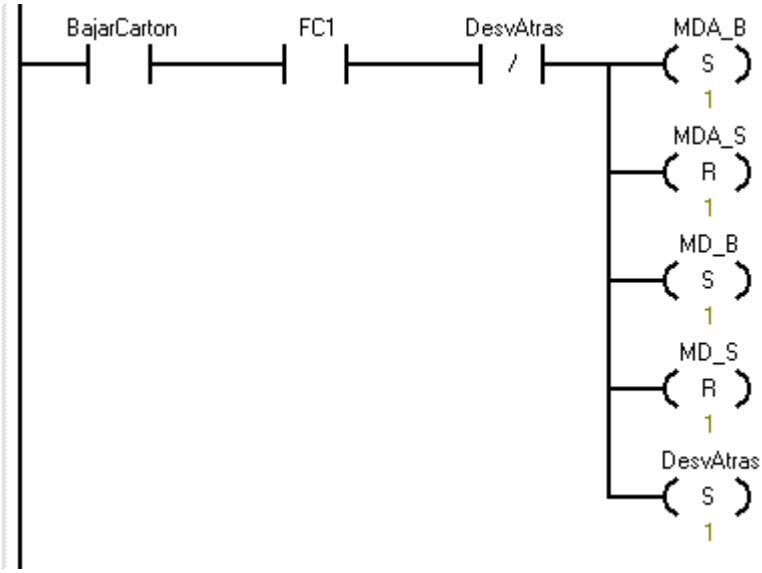


Figura 76. Segmento KOP en el que se inicia la bajada de los DVDs del cartón

Una vez establecidas las condiciones iniciales, se vuelve al OB1 en donde se espera que se introduzca un objeto y se pulse marcha. Acto seguido, se realiza la secuencia de operaciones descrita anteriormente en el apartado 4 de este documento: se activa la cinta, se espera a que el objeto caiga por la rampa y se manda la orden a la Raspberry para que capture la imagen y comience la identificación.

En este punto, el programa se queda esperando la respuesta de la Raspberry, una vez que la Red Neuronal DL determine qué tipo de material es, se entra en la subrutina correspondiente (una por cada material), siempre que el cubo no esté lleno, o en caso de estarlo, que no haya cubo fuera. En caso contrario, se activa la condición de emergencia.

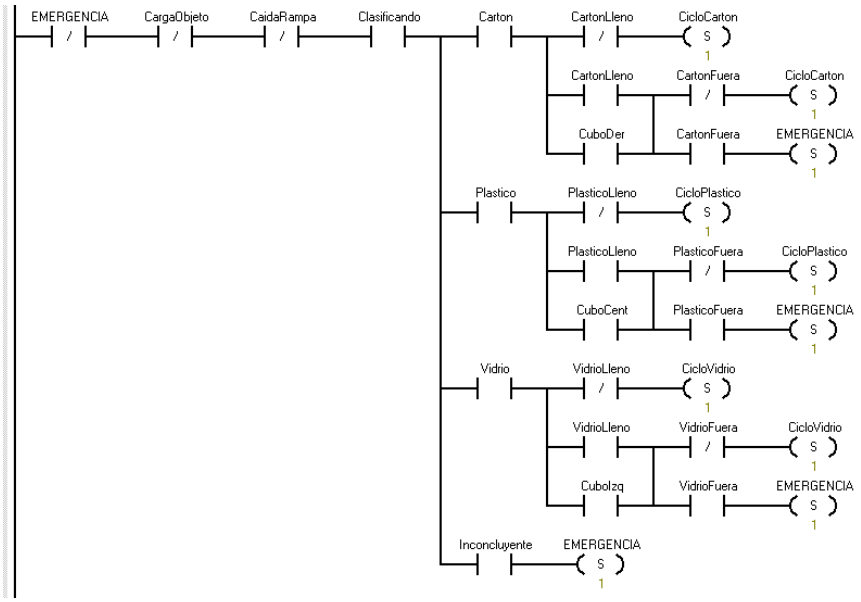


Figura 77. Segmento KOP de identificación de material

Cuando se entra en una de las subrutinas de clasificación de material, se evalúa si se entró habiendo un cubo lleno y un cubo libre detrás. En este caso, lo primero que se hace es realizar un cambio de cubos, es decir, sacar el delantero y traer adelante el trasero, pudiendo posteriormente comenzar la clasificación.

El proceso de clasificación es muy simple. En caso de ser cartón, simplemente se activará el desviador para tirar el objeto al primer cubo, evaluando mediante sensores la caída del mismo. Mientras que, en caso de ser plástico o vidrio, se activará primero la cinta transportadora y, una vez se detecte mediante sensores que se ha posicionado el objeto encima del cubo correspondiente, se activará el desviador. Si, durante este proceso, no se detecta la caída del objeto, se activará la condición de emergencia. En caso de que, una vez tirado el objeto, el cubo se haya llenado, se procederá a realizar el cambio de cubos.

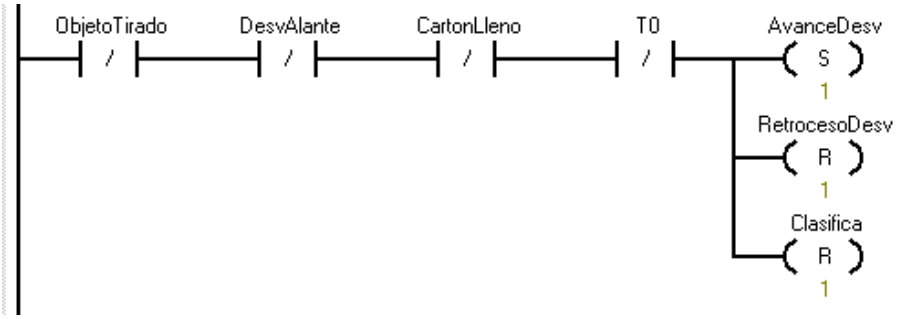


Figura 78. Segmento inicial de clasificación de cartón

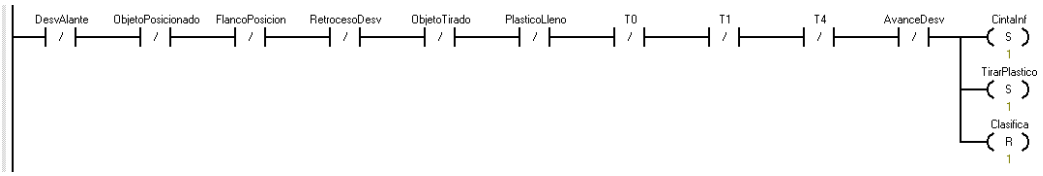


Figura 79. Segmento inicial de clasificación de plástico

Una vez terminado el proceso de clasificación, se volverá al programa principal, en donde se comenzará un nuevo ciclo de clasificación.

Adicionalmente a todo lo descrito anteriormente, en el OB1 se disponen segmentos para evaluar la presencia o no de cubos en la parte trasera de la papelera (para saber si se puede o no realizar un cambio de cubos), además de evaluarse la condición de emergencia. Esta última, resetea todas las marcas del proceso, todos los temporizadores y todas las salidas. Una vez que el usuario pulse el botón de rearme, se ejecutará el ciclo inicial, para establecer nuevamente las condiciones iniciales y comenzar un nuevo ciclo.

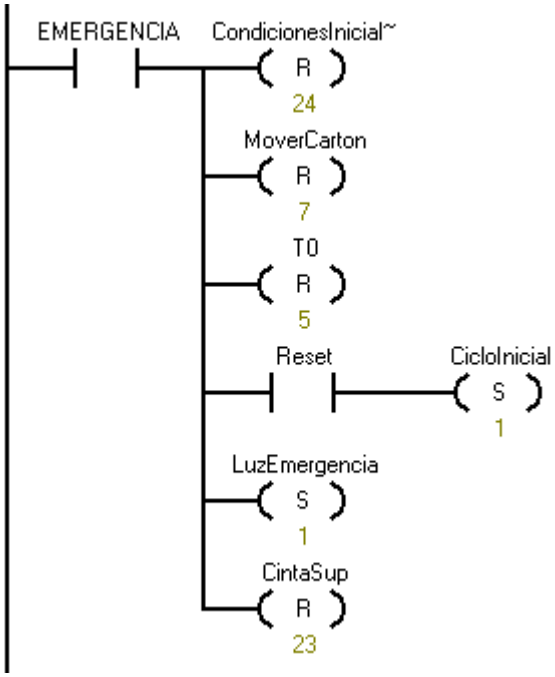


Figura 80. Segmento KOP en donde se evalúa la marca de Emergencia

La totalidad del código KOP, así como su tabla de símbolos, se encuentra disponible en el Anexo V.

6.2. RECONOCIMIENTO INTELIGENTE DE IMÁGENES

Para la identificación de los diferentes materiales de los objetos a reciclar (cartón, plástico y vidrio) se estudiaron en primer lugar diversos tipos de sensores y varios modos de uso. A continuación, se explican estas posibilidades, y los motivos por las que fueron descartadas:

- Galgas extensiométricas: Estos sensores se pueden usar para medir el peso del objeto. Mediante el peso se puede asegurar la distinción del vidrio de los otros dos materiales, pero es complicado diferenciar entre plástico y cartón.
- Sensores capacitivos: Cada tipo de material tiene una capacidad eléctrica diferente, debido a sus propiedades. Investigamos cuán grande podría ser esta diferencia y descubrimos que la diferencia podría ser de 10^{-4}, por lo que se requería de unos sensores y un sistema de lectura analógico muy preciso para distinguir los materiales. Además, el sensor debía encontrarse extremadamente cerca del objeto y no podíamos asegurar esa cercanía al caer el objeto libremente por una rampa.
- Sensores ópticos: Mediante el cálculo de la refracción de una onda de luz, se pretendía determinar de qué tipo de material se trataba. Este método se podría

haber implementado si todos los plásticos fueran transparentes, todas las botellas de vidrio también, siendo el cartón el único material opaco. Debido a que no es así, ya que existen muchos objetos plásticos opacos, no se pudo implementar.

Finalmente, al no encontrar un grupo de sensores capaz de distinguir adecuadamente los tres materiales, optamos por entrenar una Red Neuronal de Aprendizaje Profundo (Deep Learning - DL) para que se ocupara del reconocimiento de objetos.

Las Redes Neuronales DL forman parte de una de las áreas de inteligencia artificial conocida como Aprendizaje Automático (o en inglés, Machine Learning - ML) (véase figura 81)

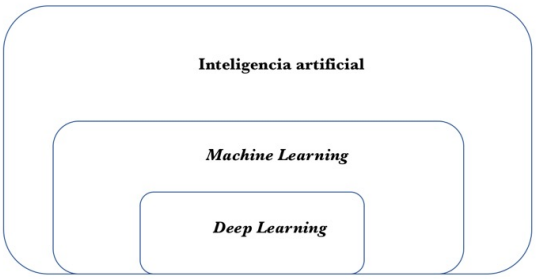


Figura 81. Subáreas de la Inteligencia Artificial. [35]

El ML proporciona a los ordenadores la capacidad de aprender sin ser explícitamente programados. Para ello, se debe desarrollar un "algoritmo" de predicción para cada caso en particular. Este algoritmo será el que aprenda de manera autónoma en base a datos de entrada, con el fin de encontrar patrones o tendencias.

Un caso particular de algoritmos de ML son las redes neuronales artificiales, en concreto, las de Deep Learning (DL). Estas redes son estructuras algorítmicas compuestas por múltiples capas de procesamiento (capas de "neuronas") para aprender representaciones complejas de datos (véase figura 82) A lo largo de estas capas se realizan una serie de transformaciones lineales y no lineales que hacen que la red sea capaz de generar una salida próxima a la esperada en base a datos de entrada previamente etiquetados (aprendizaje supervisado). Normalmente, la entrada a una neurona es la suma pesada de las salidas de todas las neuronas de la capa anterior, con un sesgo añadido (véase figura 83) Todos estos pesos y sesgos son parámetros que hay que ajustar en una etapa conocida como "etapa de entrenamiento".

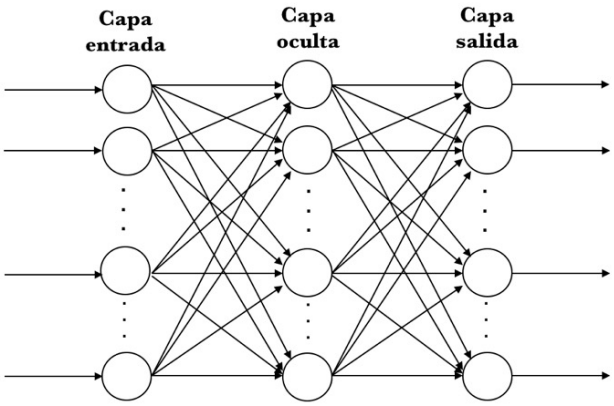


Figura 82. Capas de "neuronas" de una Red Neuronal DL [35]

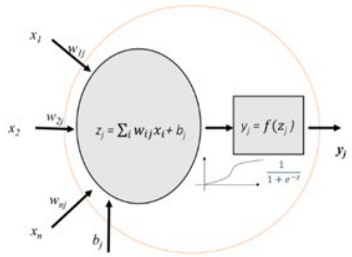


Figura 83. Algoritmo de entrada de cada "neurona" de una Red Neuronal DL [35]

En el área de visión son especialmente importantes las Redes Neuronales Convolucionales (Convolutional Neural Networks - CNN), que son unas redes específicamente diseñadas para el procesamiento eficiente de imágenes. Como particularidad de las mismas, podemos destacar que las neuronas de las primeras capas de una CNN son capaces de aprender a reconocer patrones muy básicos (bordes, texturas), las siguientes capas son capaces de combinar estos patrones básicos para formar patrones más complejos, y las últimas aprenden a combinar estos patrones más complejos para reconocer y clasificar objetos (véase figura 84).

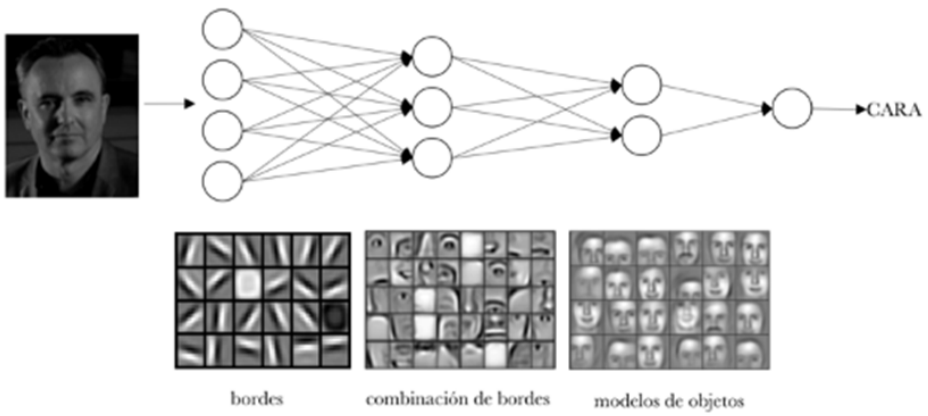


Figura 84. Aprendizaje de cada una de las capas de una Red Neuronal DL [35]

Un aspecto importante en el uso de este tipo de redes es la gran cantidad de datos de entrada requeridos durante el entrenamiento, y el alto coste computacional del mismo. Gracias a la propiedad de "Transfer Learning" de estos sistemas, podemos utilizar redes pre-entrenadas con cualquier conjunto de imágenes, porque las capas iniciales e intermedias reconocen patrones comunes a todas las imágenes, y hacer un entrenamiento sólo de las últimas capas de la red con las imágenes de entrada que pretendemos realmente clasificar. De esta manera, los tiempos de cómputo y la cantidad de imágenes necesarias disminuyen notablemente. [35]

En este TFG se ha utilizado una Red Neuronal DL pre-entrenada, concretamente la Red VGG16, que se compone de 13 capas convolucionales y 3 densas, de ahí que en su nombre incluya el "16". Esta es una red neuronal convolucional que fue entrenada para clasificar 1.4 millones de imágenes naturales en 1000 clases diferentes, usando la base de imágenes ImageNet [36]. Esta red consiguió alcanzar un 96% de precisión para esta tarea. Por este motivo, la consideramos adecuada para el problema abordado, que es mucho más simple, la clasificación de materiales reciclables en 3 clases diferentes: cartón, plástico y vidrio. Existen disponibles otras redes neuronales más potentes, como es el caso de VGG19, Densenet, etc. Que contienen más capas de neuronas, pero debido a las limitaciones de cómputo de la Raspberry y a la simpleza de la clasificación, consideramos que la VGG16 podría ser suficiente. [37]

En primer lugar, fue necesario adquirir fotografías para el entrenamiento de la red, para lo que se recolectaron muchos objetos reciclables, de diversos materiales. En total se tomaron 2014 fotos (en diferentes perspectivas), de las cuales 626 son de cartón, 1002 de plástico y 386 de vidrio.

A continuación, se realizó el entrenamiento de la red VGG16 en Google Colaboratory [38]. Los pasos fueron los siguientes:

- Preprocesamiento de las imágenes de entrada a la red, cambiando su tamaño y pasando de RGB a BGR (porque la red VGG16 pre-entrenada de Keras trabaja en este espacio de color).
- Construcción de los conjuntos de entrenamiento, validación y test. Se ha implementado una validación cruzada (5-fold cross-validation).
- En el entrenamiento se partió de la red VGG16 con los pesos originales, exceptuando los pesos de las capas superiores. Se implementó "Data Augmentation" para aumentar el número de imágenes disponibles (32 imágenes extras por cada imagen real de entrada). El parámetro "Learning Rate" se fijó a 1e-5 y el optimizador usado fue el Root

Mean Square Propagation (RMSProp).

- Se realizó un ajuste fino (de las capas superiores de la red) durante 30 épocas.
- Se realizó un ajuste profundo (de todas las capas de la red) durante 50 épocas.
- Se seleccionaron las épocas para las que se consiguieron mejores resultados en cada uno de estos entrenamientos previos, y se repitió el proceso de ajuste fino y profundo utilizando este número de épocas.

La precisión alcanzada en cada partición (fold) fue la siguiente: 96.60% (fold 1), 94.73% (fold 2), 96.27% (fold 3), 96.27% (fold 4) y 89.09% (fold 5). Las curvas de entrenamiento se pueden observar en el anexo VI.

Finalmente, se selecciona la red más óptima. Al procesar las 2014 imágenes con esta red, se obtienen los siguientes resultados de clasificación:

- 626 imágenes de cartón: 624 bien clasificadas, 2 mal clasificadas.
- 1002 imágenes de plástico: 991 bien clasificadas, 11 mal clasificadas.
- 383 imágenes de vidrio: todas bien clasificadas.

En términos de precisión, la red presenta una precisión global ("accuracy") de 99.35%. La precisión alcanzada en la clasificación de cartón es del 99.68%, de plástico es 98.90% y de vidrio es el 100%.

Una vez que se dispone de la red entrenada específicamente para el problema abordado, se carga en la plataforma (en nuestro caso la Raspberry). Para poder usarla, será necesario pasarle como entrada una imagen de un residuo (adquirida con la cámara situada en la papelera) y comunicar su salida (tipo de material) al autómata programable.

6.3. BOT RASPBERRY

Con el objetivo de automatizar completamente el proceso, se generó un código para gestionar las comunicaciones entre la Raspberry y el autómata, y así saber cuándo procesar una imagen con la red neuronal DL.

Para que la placa pudiera realizar adecuadamente la citada tarea, se le instaló el sistema operativo Raspberry Pi OS, una distribución del sistema operativo GNU/Linux basado en Debian. Dentro de este sistema, se creó un entorno virtual (que dispone de las librerías y paquetes necesarios para ejecutar la Red) y un código Python, utilizando el programa Thonny [39], que se encarga de realizar lo siguiente:

- Cuando se arranca el programa, este debe cargar la red neuronal con sus correspondientes pesos por primera vez. Además de cargar la configuración de pines de entrada y salida de Raspberry, y algunas funciones de preprocesamiento de imágenes, necesarias para pasarle las capturas a la Red Neuronal.
- Posteriormente entra en un bucle infinito que se encarga de realizar las siguientes acciones:
    - Cuando recibe la señal por parte del autómata de que debe comenzar la identificación, saca una foto del objeto con la cámara.
    - Posteriormente, la imagen capturada es redimensionada, para que pueda tener un menor peso.

Figura 85. Imagen capturada y redimensionada en RGB

    - Una vez redimensionada, se pasa de RGB a BGR. Esto es necesario porque la red ha sido entrenada con imágenes BGR, y, por tanto, las nuevas imágenes a procesar deben estar en las mismas condiciones.

Figura 86. Imagen preprocesada y convertida a BGR

    - Esa imagen preprocesada es la entrada de la red neuronal DL. Una vez que la red ha procesado la imagen, ofrece un vector de probabilidades de que el objeto de la imagen sea de plástico, de cartón o de vidrio.

- Si alguno de esos porcentajes es mayor que 85 % se activa la correspondiente salida de la Raspberry, que le envía la información del tipo de material al autómata.
- Si ninguno de los porcentajes supera el umbral marcado, se activa la salida que le indica al autómata que el resultado no es concluyente, para que el autómata avise al usuario de que es necesaria su clasificación manual.
- Una vez terminado el proceso de identificación, la salida correspondiente se apagará, esperando nuevamente a la señal del autómata para una nueva identificación del objeto.

Para conseguir que todo lo anterior funcionara correctamente, se decidió crear un entorno virtual basado en Python 3.7 dentro de la propia Raspberry, en donde se instalaron todas las librerías y paquetes necesarios para la ejecución de la red. Debido a que la red había sido entrenada en el entorno Google Colaboratory [38] con la versión 1.2 de Tensorflow, fue necesario instalar esta misma librería con su versión correspondiente de Keras asociada (en este caso, la 2.3). Adicionalmente a estos dos paquetes, que son específicos para la ejecución de la red neuronal, se tuvieron que instalar varias librerías básicas para la programación, representación gráfica y activación de pines. Algunas de ellas son: numpy, scipy, pandas, matplotlib, picamera, RPi.GPIO, etc. Una descripción detallada del proceso de instalación del entorno, así como de la mayoría de sus paquetes, se puede encontrar en la siguiente referencia: [40]

Como deseamos que el proceso de carga de la red y posterior uso de la misma se realice de manera automática al encender la Raspberry, sin intervención por parte del usuario, se tuvo que crear un nuevo servicio (.service) de systemd.

Acorde a la página ArchLinux: *"systemd es un conjunto de bloques básicos de compilación para un sistema Linux. Proporciona un gestor de sistemas y servicios que se ejecuta como PID 1 e inicia el resto del sistema. systemd proporciona una notable capacidad de paralelización, utiliza la activación de socket y D-Bus para iniciar los servicios, permite el inicio de demonios bajo demanda, realiza un seguimiento de los procesos con el uso de los grupos de control de Linux, mantiene los puntos montaje y servicios de montaje automático e implementa un elaborado sistema de gestión de dependencias basado en un control lógico de los servicios."* [41]

El citado servicio se pone en marcha siempre que se enciende la Raspberry e incluso si esta se reinicia en algún momento de la ejecución del programa. Debido a que el código Python debe de ser ejecutado dentro de un entorno virtual, se creó un archivo bash (.sh) que simula

comandos en la consola. Este archivo es el que se ejecuta al arrancar la placa, por orden del servicio creado. Posteriormente, el archivo bash es el que activa el entorno virtual y ejecuta, dentro del mismo, el código Python.

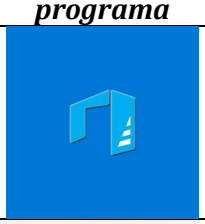
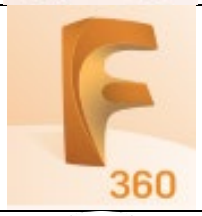
Adicionalmente, toda la información sacada por el código python es redirigida a un archivo .log ubicado en la carpeta temporal del sistema. En este archivo, se irán escribiendo los tipos de materiales clasificados y el vector de probabilidades sacado por la Red Neuronal DL en cada caso.

Todos los citados códigos, el Python, el bash y el servicio, se pueden ver en el anexo VII.

7. SOFTWARE UTILIZADO

Para realizar este proyecto, se ha hecho uso de software específico de ingeniería y otras especialidades. Todos los programas utilizados han sido, por una parte, bajo licencias educativas y licencias proporcionadas por la propia universidad, por otra parte, programas de uso gratuito. Dichos programas se recogen la siguiente tabla resumen.

Tabla 5. Software utilizado

Logotipo del programa	***Nombre y versión del programa***	***Función***	***Referencia***
	3D Builder. Versión 18.0.1931.0	Realización de los diseños del prototipo en 3D.	[8]
	Autodesk Autocad 2021. Versión R.47.0.0	Realización de los planos y esquemas necesarios para la implementación del prototipo.	[9]
	Autodesk Fusion 360. Versión 2.0.10253.	Realización del diseño de las piezas 3D utilizadas en la implementación del prototipo.	[11]
	Prusa Slicer. Versión 2.2.0.	Generar los archivos para la impresora 3D a partir de los diseños realizados.	[13]

	Ultimaker Cura. Versión 4.9.0.		[12]
	National Instruments Multisim Versión 11.0.	Diseñar la PCB implementada en el prototipo.	[32]
	National Instruments Ultiboard Versión 11.0		
	Step 7 MicroWin SP4 Versión 3.2.	Programación en KOP del proceso a automatizar en el prototipo.	[34]
	Thonny. Versión 3.3.9	Programación de la ejecución de la red en la Raspberry.	[39]
	Google Colaboratory	Programación y entrenamiento de la red neuronal implementada.	[38]
	Google Docs	Redacción de la memoria provisional, facilitando la revisión periódica de los tutores.	[42]
	Microsoft Word. Versión Office 365 para empresas.	Redacción y diseño final de la memoria.	[43]
	Microsoft Excel. Versión Office 365 para empresas.	Calcular los metros de cableado.	[44]

8. PRESUPUESTO

La construcción del prototipo ha sido, en gran parte, subvencionada por el Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna. A continuación, se muestra una tabla resumen con los gastos realizados más relevantes.

MATERIAL	UNIDADES	PRECIO TOTAL
Planchas de madera	5,5 m^2	155,34 €
Listones de madera	9,6 m	11,96 €
Tornillería	500+	25,93 €
Varios ferretería (varillas, tuercas, escuadras, etc.)	-	50 €
Ruedas	28	52, 95€
Caucho	4,9 m^2	65,79€
Tela asfáltica	0,75 m^2	3,38 €
Rodillos	15	100,35 €
PLA	1 bobina	20 €
Pintura	6 botes	71,44 €
Varios pintura (pasta madera, cola de contacto, etc.)	-	30,88 €
Sensores IR	9	59,11 €
Sensores Ópticos	6	748,14 €
Sensores Arduino	5	7,50 €
Poleas GT2	4	6,80 €
Correa dentada GT2	5 m	7,50 €
Motores	6	43,56 €
DVD's	12	221,88 €
Cuadro de mando (piloto, interruptor y pulsador)	3	6,10 €
Relés	19	72,20 €
Clemas de conexión	40	18 €
Finales de carrera	4	10,20 €
Electrónica variada (reguladores, diodos, transistores, circuitos integrados, etc.)	-	15 €
Canaletas	10 m	22,30 €
Cableado	80 m	37,92 €
Autómata y módulo de ampliación	1	500 €
Raspberry	1	38 €

Cámara Raspberry	1	28 €
Caja Raspberry	1	15 €
Fuente de alimentación	1	50 €
Cuadro eléctrico	1	31,36 €
Tira Led y conectores	5m	24,20 €
Mano de obra (30€/h)	480 h. (240h. c/u)	14.400 €
TOTAL	**-**	**16.897,84 €**

9. CONCLUSIONES Y LÍNEAS ABIERTAS

9.1. CONCLUSIONES

Para concluir, en este documento se encuentran recogidas todas las experiencias y conocimientos adquiridos, aparte del esfuerzo y dedicación que se ha empleado para hacer este prototipo realidad. Es gratificante ver como hemos sido capaz de llevar a la realidad una idea concreta.

Aunque parezca que todo fue bastante sencillo, no todo lo fue. A lo largo de la implementación de este prototipo, tuvimos que enfrentarnos a numerosas dificultades y ser capaces de adaptarnos a los nuevos problemas y situaciones que iban surgiendo. Uno de los aspectos más complicados para nosotros fue el de programar y entrenar una red neuronal, algo totalmente nuevo y que tuvimos que aprender en poco tiempo. Aunque los resultados obtenidos son bastante buenos, en el apartado 9.2.1. se exponen posibilidades para la mejora de estos resultados.

El diseño en sí también fue todo un reto, debido a que no tenemos apenas experiencia ni conocimiento profundo en estructuras y componentes mecánicos. Aún así, fuimos capaces de diseñar y llevar a cabo todo el diseño mecánico por nuestra cuenta, intentando siempre buscar la solución óptima para cada problema.

Incluso en las áreas que nos competen fueron apareciendo problemas, debido a que se implementó todo desde cero. Tuvimos que realizar todo el conexionado eléctrico y acondicionar las diferentes señales para comunicar el autómata con la Raspberry. Además, fue bastante complicado programar todo el Código de automatización de la manera más clara posible, debido a la envergadura del mismo.

Finalmente, nos gustaría remarcar que en este proyecto se han aplicado la mayoría de las áreas que se estudian en nuestro grado. Se ha tenido que realizar el diseño de una placa de circuito impreso, lidiar con electrónica, crear un conexionado eléctrico, conocer y diseñar componentes mecánicos y estructuras, trabajar con automatización y programar en diferentes lenguajes de programación.

Además, consideramos que la experiencia de haber podido trabajar en grupo en la construcción de un prototipo real, conforma un perfecto final de carrera.

9.2. LÍNEAS ABIERTAS

9.2.1. MEJORA DEL RECONOCIMIENTO DE IMÁGENES

Para mejorar el acierto del sistema de identificación inteligente de imágenes, se propone mejorar la Red Neuronal DL implementada.

Una de las opciones que se plantea es volver a entrenar la misma red que se utilizó en el prototipo (la vgg16) con más imágenes y durante más tiempo. Debido a las limitaciones de tiempo que presenta el Google Colaboratory [38], solo se pudo entrenar la Red Neuronal durante unas 5 horas en total, con un total de 30 ciclos (como se indicó anteriormente). Además, esta fue entrenada con 2014 imágenes, pero solo 383 de ellas eran de vidrio, debido a que, por probabilidad, es lo que menos se consume. Si se pudieran conseguir más imágenes para su entrenamiento, especialmente del material citado, se mejorarían los resultados de clasificación de esta.

Otra de las opciones es utilizar otra red diferente, la vgg19, que es más potente que la vgg16 utilizada. Para ello, se recomienda su ejecución en otra plataforma diferente a la Raspberry, debido a su gran peso y necesidad de capacidad de cómputo.

9.2.2. MEJORA DE LA EJECUCIÓN DE LA RED

Para no tener que ejecutar la Red Neuronal DL en local en la Raspberry, se plantean varias posibilidades.

Una de ellas sería ejecutar la Red Neuronal directamente en la plataforma Google Colaboratory [38], en donde fue entrenada. Para ello, es necesario crear una interfaz que sea capaz de simular el comportamiento humano para poder interactuar con la plataforma citada, ya que esta no está pensada para comunicarse directamente con un programa. El problema

que tiene esta mejora es que el Google Colaboratory tiene un tiempo determinado de ejecución continua, después de la cual se termina la sesión y no permite la ejecución del archivo hasta pasado un tiempo.

Por tanto, como idea análoga a la anteriormente planteada, se pensó en montar un servidor propio en donde se ejecute la Red Neuronal. Se tendría que establecer comunicación con el citado servidor desde la Raspberry para así poder enviarle la imagen capturada con la cámara y leer el resultado devuelto por la Red Neuronal.

Cualquiera de estas dos opciones evita la ejecución de la red en local, evitando así un sobrecaliento excesivo de la placa y mejorando la capacidad de cómputo disponible para la red.

9.2.3. EXTRACCIÓN DE OBJETOS NO IDENTIFICADOS

A la hora de clasificar el objeto por parte de la Red Neuronal, se plantea una opción de salida que es la de "Inconcluyente". Esta opción, se programó para que se diera cuando todas las probabilidades (de cada una de las clases) fueran inferiores a un 50 %. En caso de que esta condición ocurra, el autómata entra en un estado de emergencia, para indicarle al usuario que recicle el objeto de forma manual.

Para evitar esta condición de emergencia, se plantea permitir la salida de aquellos objetos que cumplen esta condición. Para ello, habría que hacer un agujero en el lateral del prototipo que está más pegado al vidrio, poner una bolsa colgando en esa pared e implementar un sensor para determinar la caída del objeto. De esta forma, si la Raspberry devuelve el resultado "Inconcluyente", el autómata activaría la cinta inferior hasta detectar que el objeto haya salido y caído en la bolsa. De esta manera, no se interrumpiría el ciclo de clasificación de objetos y el usuario podría clasificar manualmente los objetos a posteriori.

9.2.4. FINALES DE CARRERA PARA DISPOSITIVOS MOTORIZADOS

Para evitar el forzado de los motores de los dispositivos motorizados en los ciclos iniciales y al volver de una condición de emergencia, se plantea la opción de introducir unos finales de carrera para saber la posición de estos.

Una posibilidad sería utilizar un autómata que disponga de más entradas o bien introducir un módulo de ampliación de E/S digitales adicional sistema automatizado del que ya se dispone. De esta manera, se tendría información de la posición de todos los dispositivos motorizados de sujeción de cubos y se podría actuar sobre ellos solo cuando fuera necesario.

El problema es que se necesitarían 12 entradas más, para poder saber la posición superior e inferior de los 6 dispositivos de sujeción.

La otra opción, sería implementar los finales de carrera en un circuito eléctrico con los motores mencionados para que, en caso de que el final de carrera estuviera pulsado, no se permitiera el paso de la corriente hacia el motor. De esta manera, aunque el autómata tuviera dicha salida activada, si el dispositivo motorizado ya estuviera en la posición deseada, no circularía corriente, al estar el circuito abierto.

9.2.5. INTERACCIÓN CON EL SISTEMA

Para poder monitorizar e interactuar con el autómata, se plantea conectar una pantalla HMI y diseñar un sistema SCADA. En esta pantalla se podrían mostrar datos tales como: número de objetos clasificados según material, número de veces que ha salido la condición de inconcluyente, número de veces que se ha dado una condición de emergencia, información acerca del por qué se ha ejecutado la condición de emergencia, etc.

Otra posible interacción con el sistema sería la de interactuar con la Raspberry y a su vez con la Red Neuronal. La idea sería implementar un sistema capaz de guardar las imágenes que el código ha determinado como “Inconcluyentes” para que el usuario pueda revisarlas a posteriori y determinar, mediante una pantalla que interacción con el sistema, de qué material son. De esta manera, se podría hacer un entrenamiento continuo de la red, para que fuera aprendiendo de los errores que pueda ir teniendo.

9. CONCLUSIONS AND FUTURE LINES

9.1. CONCLUSIONS

To conclude, in this document all the experiences and knowledge acquired are shown. Apart from the effort and dedication to make this prototype reality. It is very grateful to see how our idea to help the environment became truth.

Although this seems very nice, not all went well. During the implementation of the prototype, we had to face many difficulties and readapt ourselves to new situations. For us, it was the first time working with a neuronal network, so we had to learn how to programme it and train it. Despite, the results obtained were very good according to the time of training and

the image database, we think that they can be betted. To improve them, we left a future line to implement.

The design also was a challenge due that we are not specialized on mechanical structures and components, necessaries to do the implementation. Still, we did all the mechanical design by ourselves and looking for the best solution to each problem.

Even in our competencies, the issues appeared due to added difficulty of starting from scratch. We had to design all the electronics adaptations to communicate the Raspberry Pi with the automaton. Also, it was difficult to code all the process making it the cleaner possible because of the dimensions of the automation process.

Finally, it is important to emphasise the application of the knowledge acquired during the degree studies in all the areas. In this project, we had to deal with printed circuits design, electronic, electric machines, mechanical components and structures, automation, and programming. Also, the experience of working in team creating a real prototype, makes this project the perfect ending for the degree.

9.2. FUTURE LINES

9.2.1. IMPROVEMNT OF THE DETECTION OF OBJECTS

In order to improve the accuracy given by the identification system, solutions to improve the neuronal network are described.

One of the possible options is to train again the same neuronal network implemented in this prototype (vgg16) with more images during more time. Due to the time limitations that Google Colaboratory [38] provides, the Neuronal Network could only be trained during 5 hours with a total of 30 cycles. Apart from that, the neuronal network was trained with a total of 2014 images, but only 383 of them where of glass objects. Collecting more images of the mentioned material, the results could be improved.

Another option could be using a different neuronal network, for example the vgg19. This one is much more powerful than the vgg16. To do that, another platform rather than Raspberry Pi is recommended, because of the hardware needs.

9.2.2. IMPROVEMENT OF THE EXECUTION OF THE NEURONAL NETWORK

In order to not execute the neuronal network of deep learning on the Raspberry Pi, different possibilities are suggested.

One of the possibilities could be executing the neuronal network directly on the platform where it was trained, Google Colaboratory [38]. To do that it is necessary to program n interface able to simulate the human behavior in order to interact with the platform, because this is not able to communicate directly with another program. The problem of this platform is that the execution time is limited.

Therefore, another option is to build the neuronal network on a local server. The raspberry Pi should be able to communicate with it, send the image and read the reply given. With this solution, there Will not be any timing problems.

Whichever option avoids the execution of the neuronal network on the Raspberry Pi directly, avoiding an overheating of the Raspberry and giving the neuronal network a better computing capacity.

9.2.3. REMOVAL OF NON INDENTIFIED OBJECTS

When the neuronal network must decide which material is the object of, in case the probability that gives is lower than a 50% in the three classes, the result given by the system will be: “Not Identified”. In this case, according to the actual design, an emergency light will be turned on and the user will classify the object manually.

To avoid this emergency condition, one of the options is to let the non-identified object exit by themselves from the prototype. To implement this, a hole on the left side wall should be done, apart from putting a bag with a sensor located outside the hole. If the object is not classified, the conveyor belt will bring the object to the hole and the sensor will detect its falling into the bag.

With this improvement, the recycling process will not be interrupted anymore by this emergency condition and the user could recycle the objects later manually.

9.2.4. LIMIT SWITCHS FOR THE MOTORIZED BUCKETS SUPPORT

To avoid the electric forcing of the DVD motors during the initial cycle and when returning from an emergency condition, one option could be introducing limit switched to know its positions.

One possibility could be using a larger automaton with more inputs or introducing more I/O ampliation modules. In this way, limit switches could be implemented don the DVDs and the system could be able to know the position of each one of them and act only if necessary. The problem is that 12 inputs are needed, to know the top and low position of the six DVDs

The other option could be implementing this limit switches in an electronic circuit with the DVD motors. So, with this option, in case the limit switch was pressed, even if the automaton gives the order to move the DVD, this won't do anything, because the circuit would be open.

9.2.5. INTERACTION WITH THE SYSTEM

To control and interact with the automaton, an HMI connected into a SCADA system is required. This scree, could show different statistics and relevant information such as: number of objects thrown of each material, number of times an emergency has occurred, information about where the emergency came from, etc.

Another possibility of interaction with the system could be interact with the Raspberry Pi and in consequence with the neuronal network. The idea could be to implement a system able to save the images that the system has not identified so the user could tell the system the material of each one of them. With this improvement, the neuronal network could be in constant training.

BIBLIOGRAFÍA

[1] Estadística Ecoembes, «ecoembes.com,» [En línea]. Available: https://www.ecoembes.com/sites/default/files/cifras-reciclaje-2018.pdf. [Último acceso: Junio 2021].

[2] Proyecto BIN-E, «bine.world,» [En línea]. Available: https://www.bine.world/newbine/. [Último acceso: Junio 2021].

[3] Proyecto IRBin, «departamento.pucp.edu.pe,» [En línea]. Available: https://departamento.pucp.edu.pe/ingenieria/2020/02/17/irbin-el-robot-que-educa-sobre-reciclaje/. [Último acceso: Junio 2021].

[4] Proyecto Automation Waste Segregation System, «nevonprojects.com,» [En línea]. Available: https://nevonprojects.com/automatic-waste-segregation-system/. [Último acceso: Junio 2021].

[5] Proyecto TrashBot, «cleanrobotics.com,» [En línea]. Available: https://cleanrobotics.com/trashbot/. [Último acceso: Junio 2021].

[6] Proyecto MY041, «youtube.com,» [En línea]. Available: https://www.youtube.com/watch?app=desktop&v=P2XsRP6cBHM&ab_channel=DreamCatcherAsia. [Último acceso: Junio 2021].

[7] Videos Funcionamiento Prototipo, A. Méndez y S. Panzeri, «youtube.com,» [En línea]. Available: https://youtube.com/playlist?list=PLnKOLtJ8q_dOhKpIGwrkMz3gkjdMt6r7C. [Último acceso: Junio 2021].

[8] Programa 3D Builder, «microsoft.com,» [En línea]. Available: https://www.microsoft.com/es-es/p/3d-builder/9wzdncrfj3t6?activetab=pivot:overviewtab. [Último acceso: Junio 2021].

[9] Programa AutoCad, «autodesk.es,» [En línea]. Available: https://www.autodesk.es/products/autocad/overview?panel=buy&mktvar002=4417848|SEM|8054028189823645911 99kwd-297275808151&plc=ACDIST&term=1-YEAR&support=ADVANCED&quantity=1&ef_id=Cj0KCQjw8IaGBhCHARIsAGIRRYpXRGHTVfa23eFPb21DOn2M9H9ebSnvNBoqV4fFGQ9bl8yGWFDiIG. [Último acceso: Junio 2021].

[10] Filamento PLA, «3dnatives.com,» [En línea]. Available: https://www.3dnatives.com/es/guia-filamento-pla-en-la-impresion-3d-190820192/. [Último acceso: Junio 2021].

[11] Programa Fusion 3D, «autodesk.es,» [En línea]. Available: https://www.autodesk.es/products/fusion-360/overview?term=1-YEAR. [Último acceso: Junio 2021].

[12] Programa Ultimaker Cura, «ultimaker.com,» [En línea]. Available: https://ultimaker.com/es/software/ultimaker-cura. [Último acceso: Junio 2021].

[13] Programa Prusa Slicer, «prusa3d.es,» [En línea]. Available: https://www.prusa3d.es/prusaslicer/. [Último acceso: Junio 2021].

[14] Manual S7-200, «support.industry.siemens.com,» [En línea]. Available: https://support.industry.siemens.com/cs/document/1109582/s7-200-manual-del-sistema?dti=0&lc=es-ES. [Último acceso: Junio 2021].

[15] Datasheet Sensores IR, «aliExpress.com,» [En línea]. Available: https://cutt.ly/hnOehxC. [Último acceso: Junio 2021].

[16] Datasheet Sensores Ópticos, «octopart.com,» [En línea]. Available: https://datasheet.octopart.com/GLV12-8-200/36/40B/115-Pepperl-datasheet-75729.pdf. [Último acceso: Junio 2021].

[17] Datasheet Motores 200rpm, «zhengkemotor.com,» [En línea]. Available: http://www.zhengkemotor.com/product/zhengkemotor_product_Dc_Brushed_Gear_Motor_28mm_RF370_ZGA28RP-en.html. [Último acceso: Junio 2021].

[18] Datasheet Motor 50 rpm, «zhengkemotor.com,» [En línea]. Available: http://www.zhengkemotor.com/product/Zhengk_DCGearMotor_Brushed_CentricShaft_37mm_RoundFlange_ZYTD520_ZGA37RG-en.html. [Último acceso: Junio 2021].

[19] Datasheet Motor 60 rpm, «zhengkemotor.com,» [En línea]. Available: http://www.zhengkemotor.com/product/zhengkemotor_product_DCGearMotor_Brushed_EccentricShaft_37mmDiameter_RoundFlange_520Motor-en.html. [Último acceso: Junio 2021].

[20] Raspberry Pi3 B+, «raspberrypi.org,» [En línea]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/. [Último acceso: Junio 2021].

[21] Raspberry Pi Camera V2.1, «raspberrypi.org,» [En línea]. Available: https://www.raspberrypi.org/products/camera-module-v2/. [Último acceso: Junio 2021].

[22] Datasheet réles utilizados, «cdn.findernet.com,» [En línea]. Available: https://cdn.findernet.com/app/uploads/2020/09/09101228/S36ES.pdf. [Último acceso: Junio 2021].

[23] Datasheet LM324, «alldatasheet.com,» [En línea]. Available: https://html.alldatasheet.com/html-pdf/22753/STMICROELECTRONICS/LM324/1619/1/LM324.html. [Último acceso: Junio 2021].

[24] Datasheet LM78, «mouser.com,» [En línea]. Available: https://www.mouser.com/datasheet/2/149/LM7812-461970.pdf. [Último acceso: Junio 2021].

[25] Imagen del LM7805, «tienda.bricogeek.com,» [En línea]. Available: https://tienda.bricogeek.com/reguladores/91-regulador-de-tension-5v-lm7805.html. [Último acceso: Junio 2021].

[26] Datasheet del LM3940, «mouser.es,» [En línea]. Available: https://www.mouser.es/ProductDetail/Texas-Instruments/LM3940IT-33?qs=sGAEpiMZZMsGz1a6aV8DcCERHZHPu4lvA3HG9OH8Mo4=. [Último acceso: Junio 2021].

[27] Imagen del LM7812, «sumador.com,» [En línea]. Available: https://sumador.com/products/regulador-lm7812-12-vdc. [Último acceso: Junio 2021].

[28] Imagen de condensador electrolítico, «amazon.es,» [En línea]. Available: https://www.amazon.es/Condensador-electrol%C3%ADtico-50V-0-47uF-pack/dp/B00WF793EA. [Último acceso: Junio 2021].

[29] Imagen de condensador , «diotronic.com,» [En línea]. Available: https://diotronic.com/poliester-metalizado-mkt-mkte/10821-cond-b32529-330nf-100v-r-5. [Último acceso: Junio 2021].

[30] Datasheet del 1N4007, «vishay.com,» [En línea]. Available: https://www.vishay.com/docs/88503/1n4001.pdf. [Último acceso: Junio 2021].

[31] Datasheet del BC550, «mouser.com,» [En línea]. Available: https://www.mouser.com/datasheet/2/149/BC550-888526.pdf. [Último acceso: Junio 2021].

[32] Programa Multisim, «ni.com,» [En línea]. Available: https://www.ni.com/es-es/support/downloads/software-products/download.multisim.html#306443. [Último acceso: Junio 2021].

[33] Imagen de la CNC utilizada, «davidcominotoca1992.wordpress.com,» [En línea]. Available: https://davidcominotoca1992.wordpress.com/2012/02/27/proceso-de-trabajo-cnc-protomat-s62/. [Último acceso: Junio 2021].

[34] Programa Step7 MicroWin, «support.industry.siemens.com,» [En línea]. Available: https://support.industry.siemens.com/cs/document/58523240/step7-microwin-v4-0-sp8-y-sp9?dti=0&lc=es-WW. [Último acceso: Junio 2021].

[35] Deep Learning, Introducción práctica con Keras y J. Torres, «torres.ai,» [En línea]. Available: https://torres.ai/deep-learning-inteligencia-artificial-keras/#Inteligencia_artificial_Machine_Learning_y_Deep_Learning. [Último acceso: Junio 2021].

[36] Base de imágenes ImageNet, «image-net.org,» [En línea]. Available: https://www.image-net.org/. [Último acceso: Junio 2021].

[37] Descripción red neuronal vg16, «personal.cimat.mx,» [En línea]. Available: http://personal.cimat.mx:8181/~mrivera/cursos/aprendizaje_profundo/preentrenadas/preentrenadas.html. [Último acceso: Junio 2021].

[38] Programa Google Colaboratory, «colab.research.google.com,» [En línea]. Available: https://colab.research.google.com/notebooks/intro.ipynb?utm_source=scs-index. [Último acceso: Junio 2021].

[39] Programa Thonny, «thonny.org,» [En línea]. Available: https://thonny.org/. [Último acceso: Junio 2021].

[40] Instrucciones para montar entorno virtual , «codementor.io,» [En línea]. Available: https://www.codementor.io/@ferrorodolfo/setup-your-raspberry-pi-model-b-as-google-colab-feb-19-to-work-with-tensorflow-keras-and-opencv-tb2nwpbyb. [Último acceso: Junio 2021].

[41] Descripción systemd, «wiki.archlinux.org,» [En línea]. Available: https://wiki.archlinux.org/title/Systemd_(Espa%C3%B1ol). [Último acceso: Junio 2021].

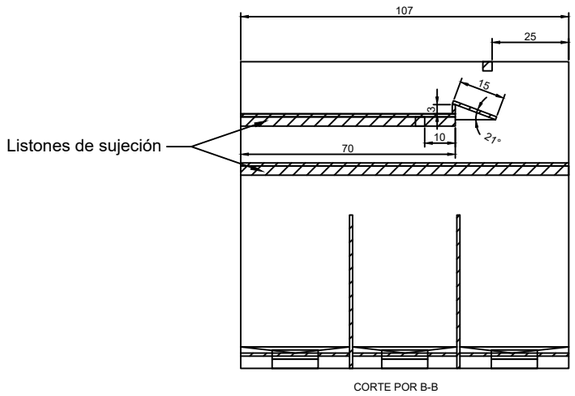
[42] Programa Google Docs, «docs.google.com,» [En línea]. Available: https://docs.google.com/document/u/0/?hl=es&pli=1. [Último acceso: Junio 2021].

[43] Programa Microsoft Word, «microsoft.com,» [En línea]. Available: https://www.microsoft.com/es-es/microsoft-365/word. [Último acceso: Junio 2021].

[44] Microsoft Excel, «microsoft.com,» [En línea]. Available: https://www.microsoft.com/es-es/microsoft-365/excel. [Último acceso: Junio 2021].

ANEXOS

ANEXO I.
PLANOS DEL PROTOTIPO

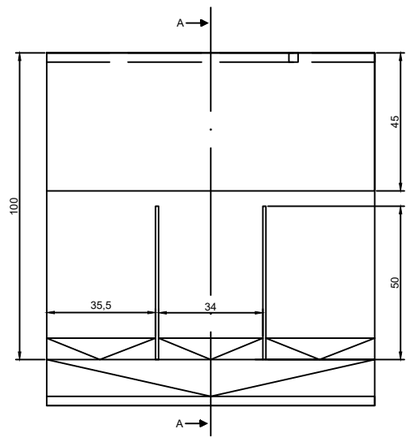


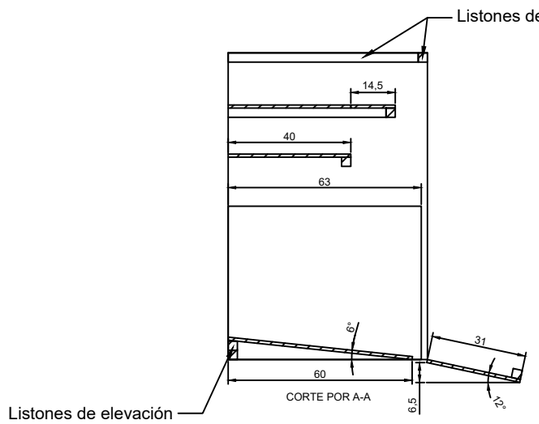
CORTE POR B-B

NOTA: TODAS LAS COTAS ESTÁN EN CM.

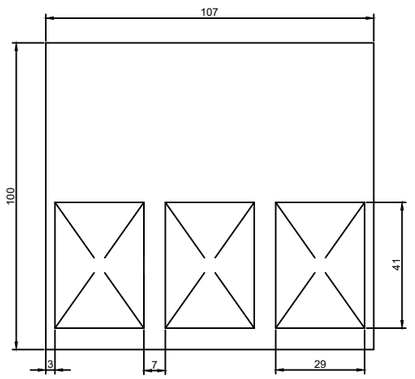
Nota$_1$: todos los **listones** utilizados son 3 x 3 cm.

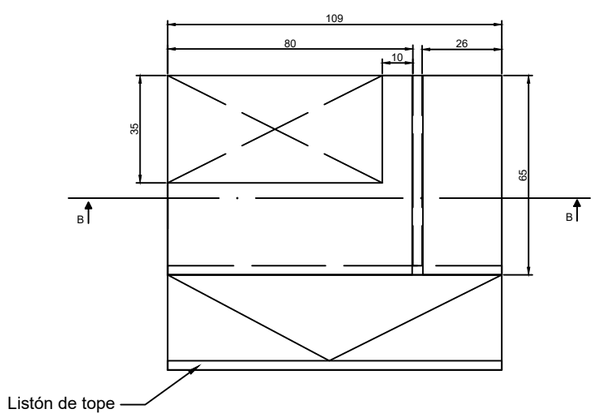
Nota$_2$: el prototipo no está en contacto con el suelo, por la elevación de unas ruedas de 7.5 cm





CORTE POR A-A

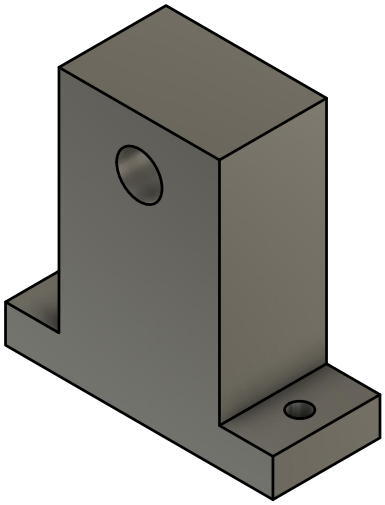




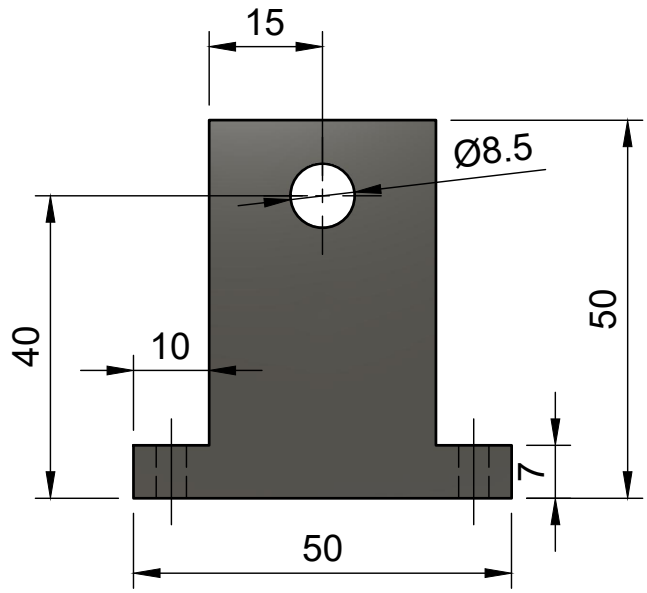
TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO			
Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESCALA: *E 1:20*	DIMENSIONES DEL PROTOTIPO		Nº P. : 1 Nom.Arch: Planos_TFG.dwg

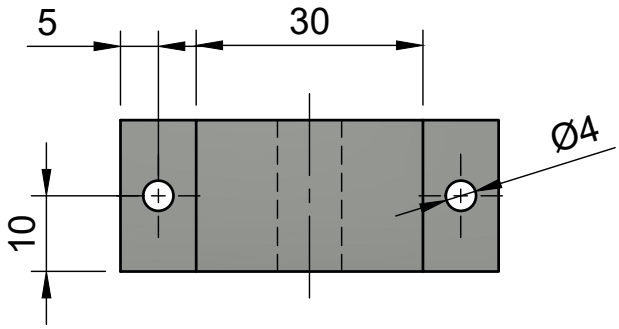
ANEXO II.

PLANOS PIEZAS 3D



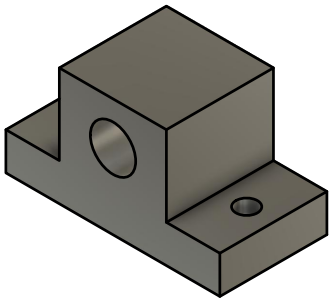
"Vista isométrica"



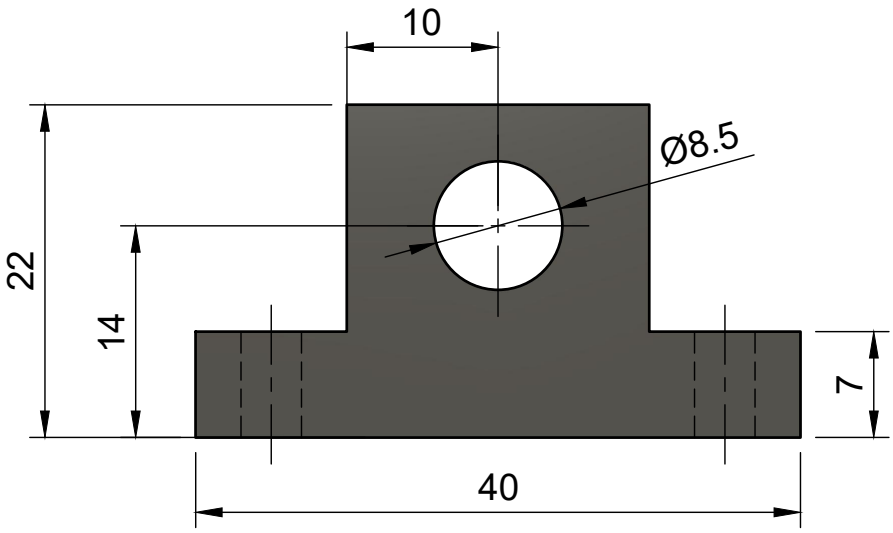


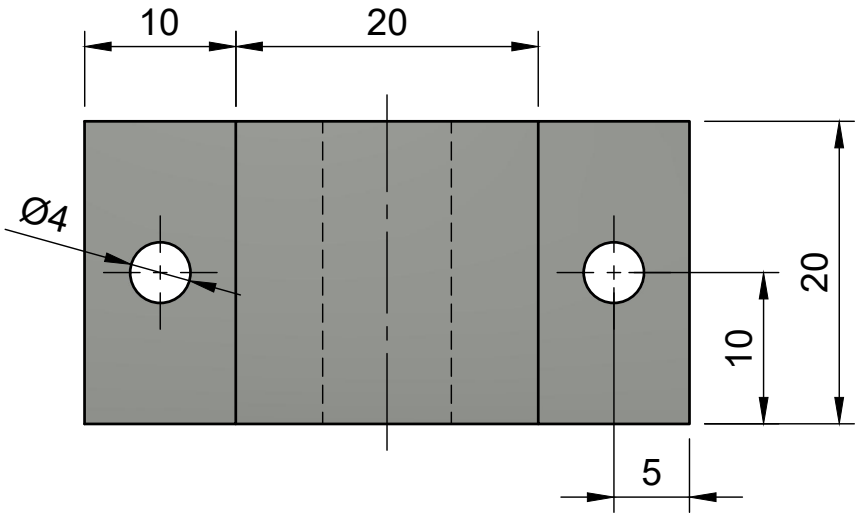
TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO

Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESCALA: *E 1:1*	SOPORTE RODILLO ACOTADO		Nº P. : 2.1 Nom.Arch: Pieza Soporte Rodillo v8



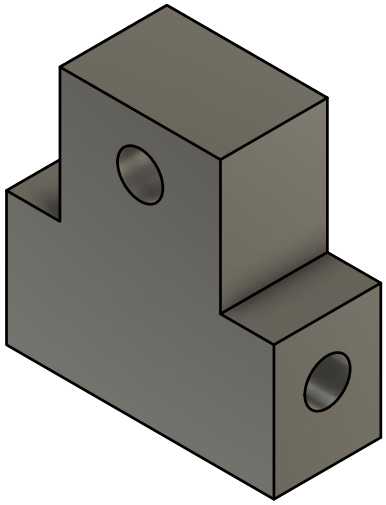
"Vista isométrica E 1:1"



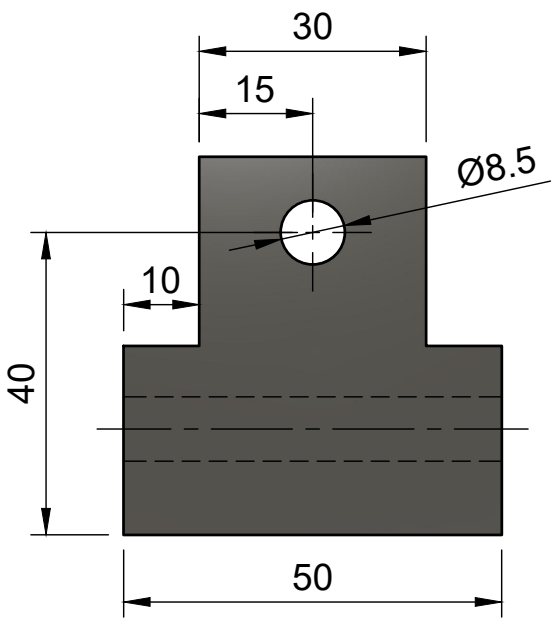


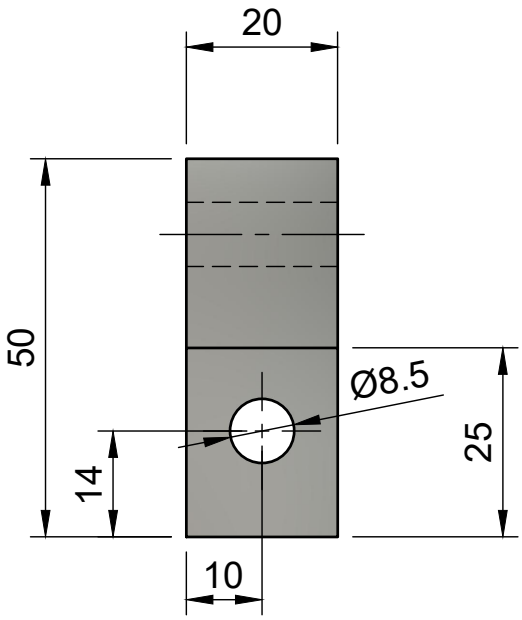
TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO

Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESCALA: *E 2:1*	SOPORTE PASANTE ACOTADO		Nº P. : 2.2 Nom.Arch: Pieza Soporte Pasante_Tensor v7



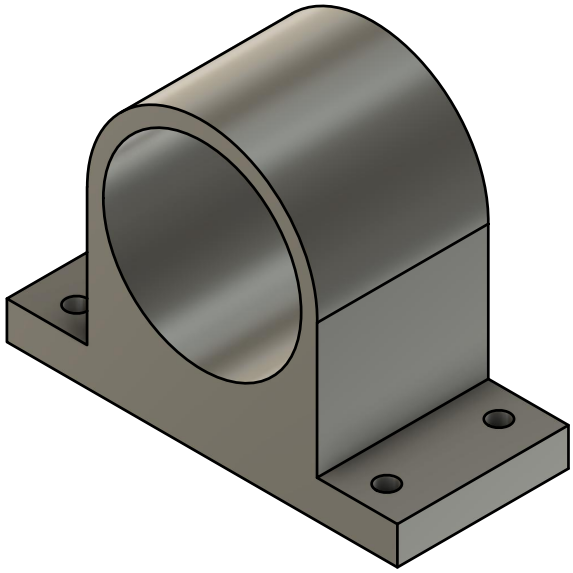
"Vista isométrica"



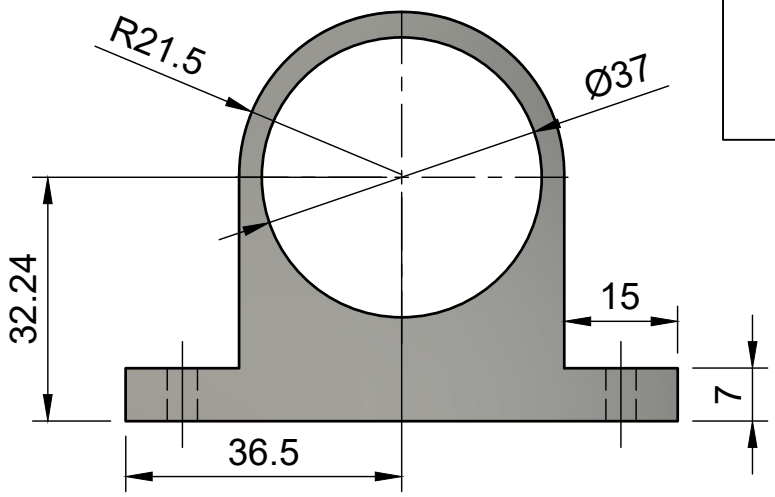


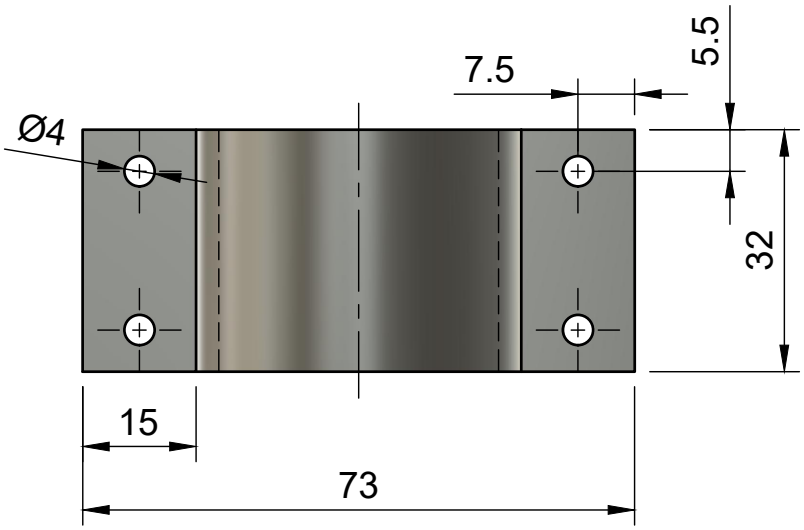
TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO

Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESCALA: *E 1:1*	SOPORTE RODILLO TENSOR ACOTADO		Nº P. : 2.3 Nom.Arch: Pieza Soporte Rodillo_Tensor v6



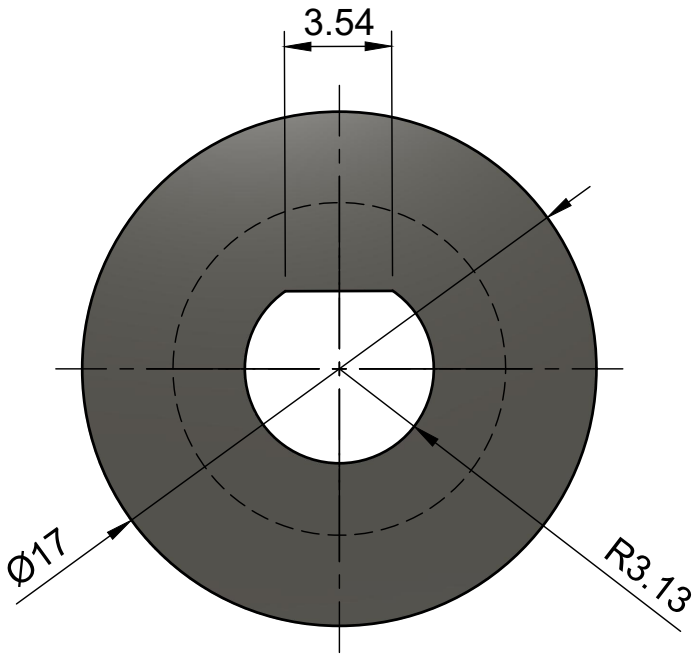
"Vista isométrica"

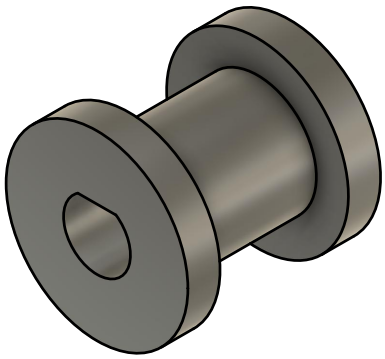




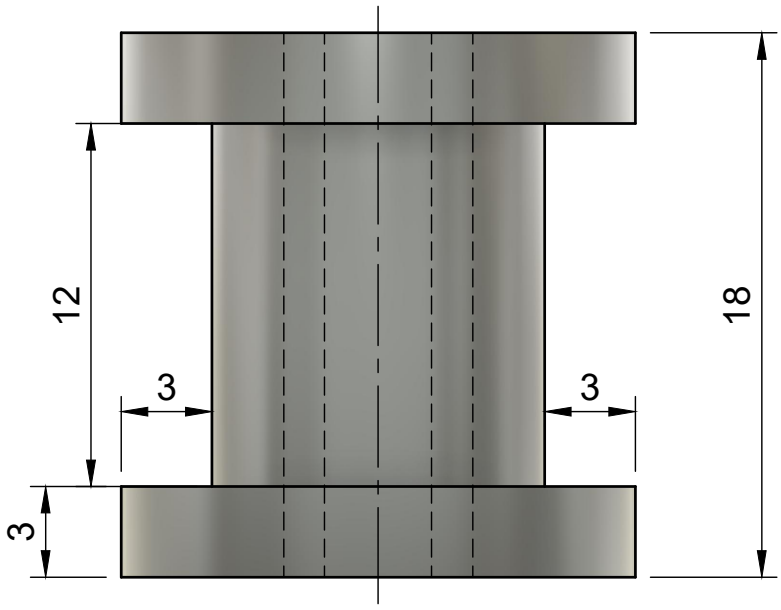
TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO

Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESCALA: *E 1:1*	SOPORTE MOTOR GRANDE ACOTADO		Nº P. : 2.4 Nom.Arch: Pieza Soporte Motor Grande v7

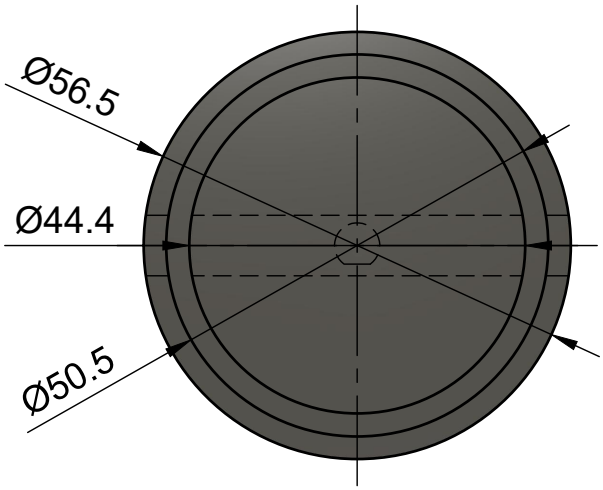




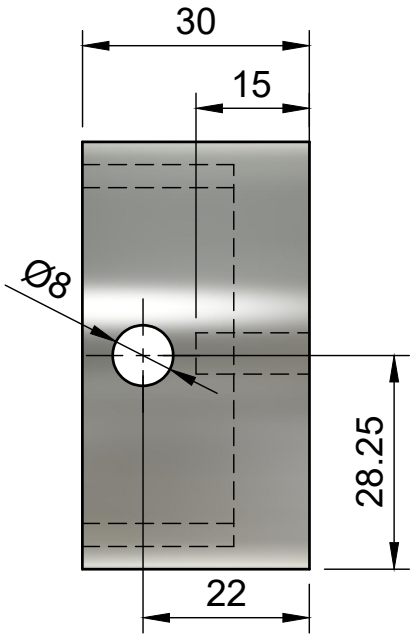
Vista isométrica (E 2:1)

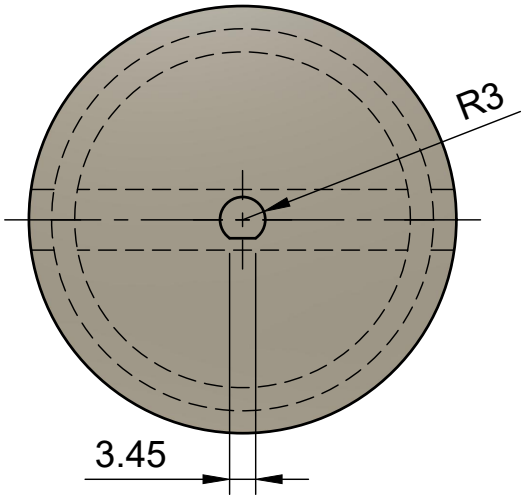


TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO			
Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESCALA: *E 4:1*	POLEA MOTOR GRANDE ACOTADA		Nº P. : 2.5 Nom.Arch: Soporte Correa Motor Grande v9

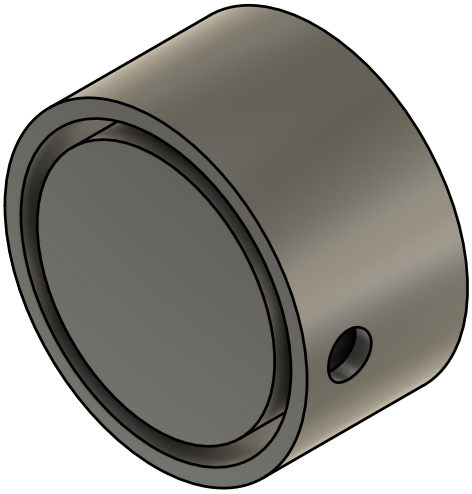


"Alzado"



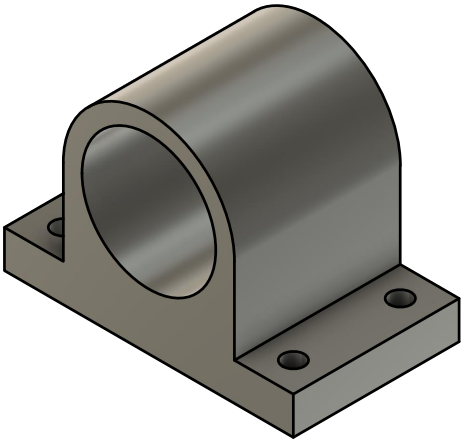


"Alzado Posterior"

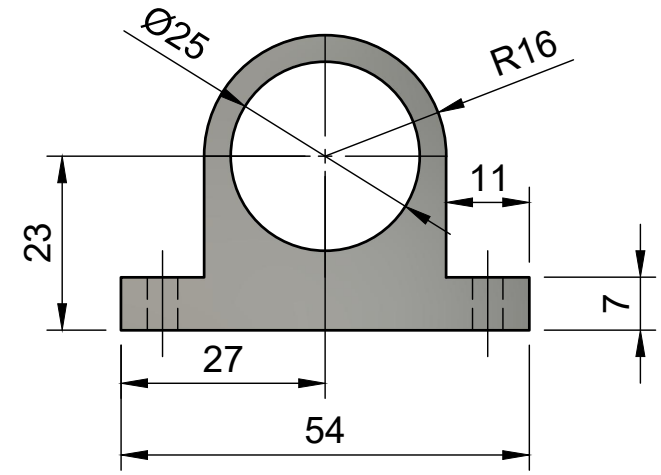


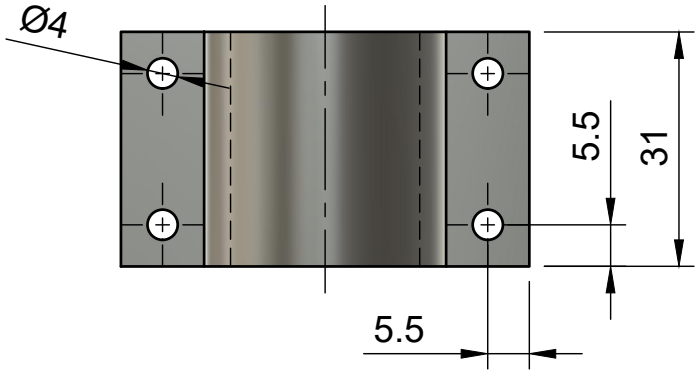
"Vista isométrica"

TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO			
Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESCALA: *E 1:1*	ENCAJE RODILLO ACOTADO		Nº P. : 2.6 Nom.Arch: Encaje Rodillo v8



"Vista isométrica"



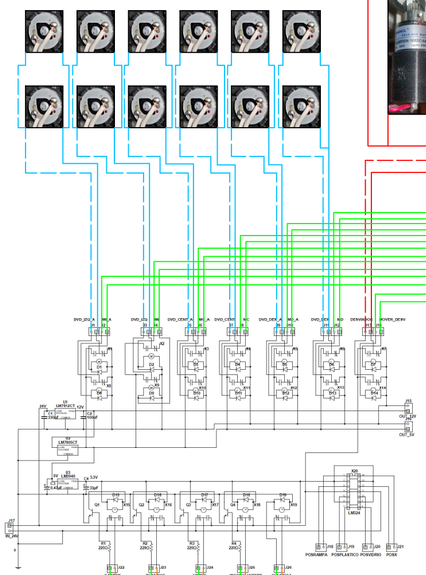


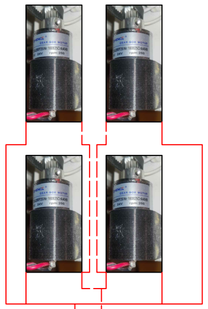
TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO

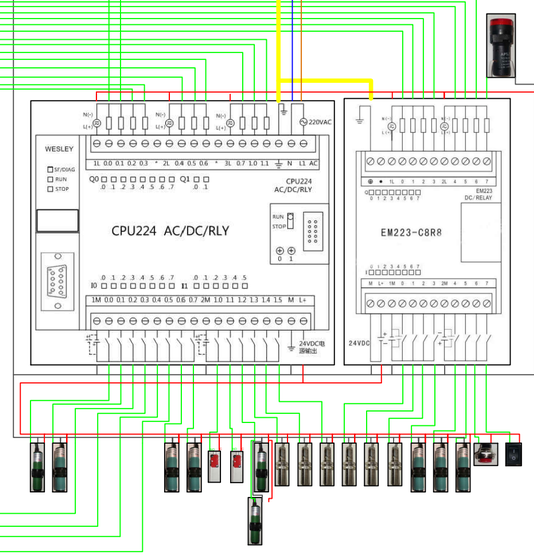
Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESCALA: *E 1:1*	SOPORTE MOTOR ACOTADO		Nº P. : 2.7 Nom.Arch: Pieza Soporte Motor v4

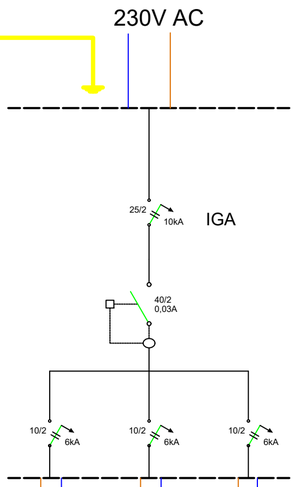
ANEXO III.
ESQUEMA DE CONEXIONADO ELÉCTRICO DEL PROTOTIPO

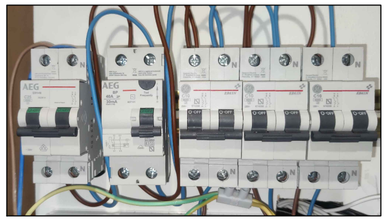
LEYENDA	
Línea	Descripición
	5V DC
	12V DC
	24V DC
	230V AC L
	230V AC N
	GND AC
	GND DC
	SEÑALES AUTÓMATA
	SEÑALES RASPBERRY
	5V DC (SUBE)
	5V DC (BAJA)
	24V DC (AVANZA)
	24V DC (RETROCEDE)

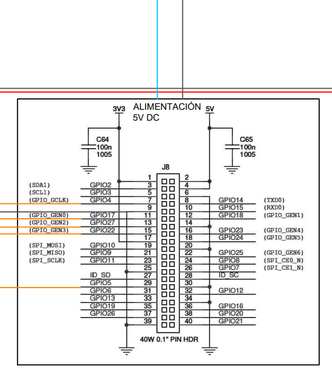


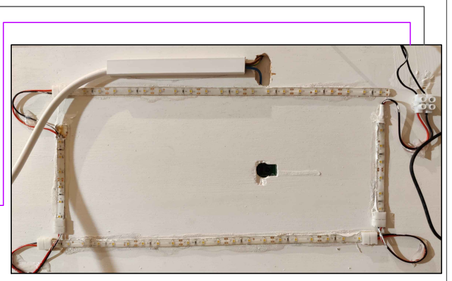


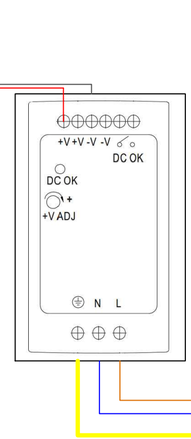


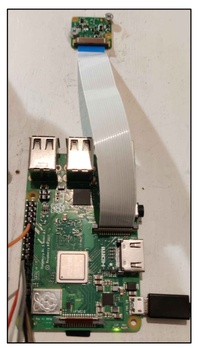


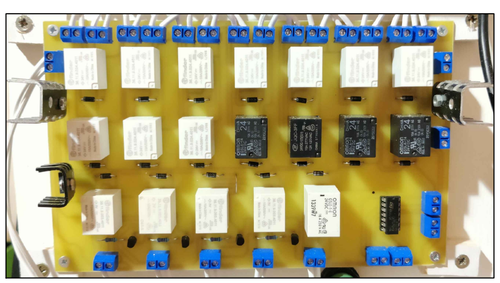








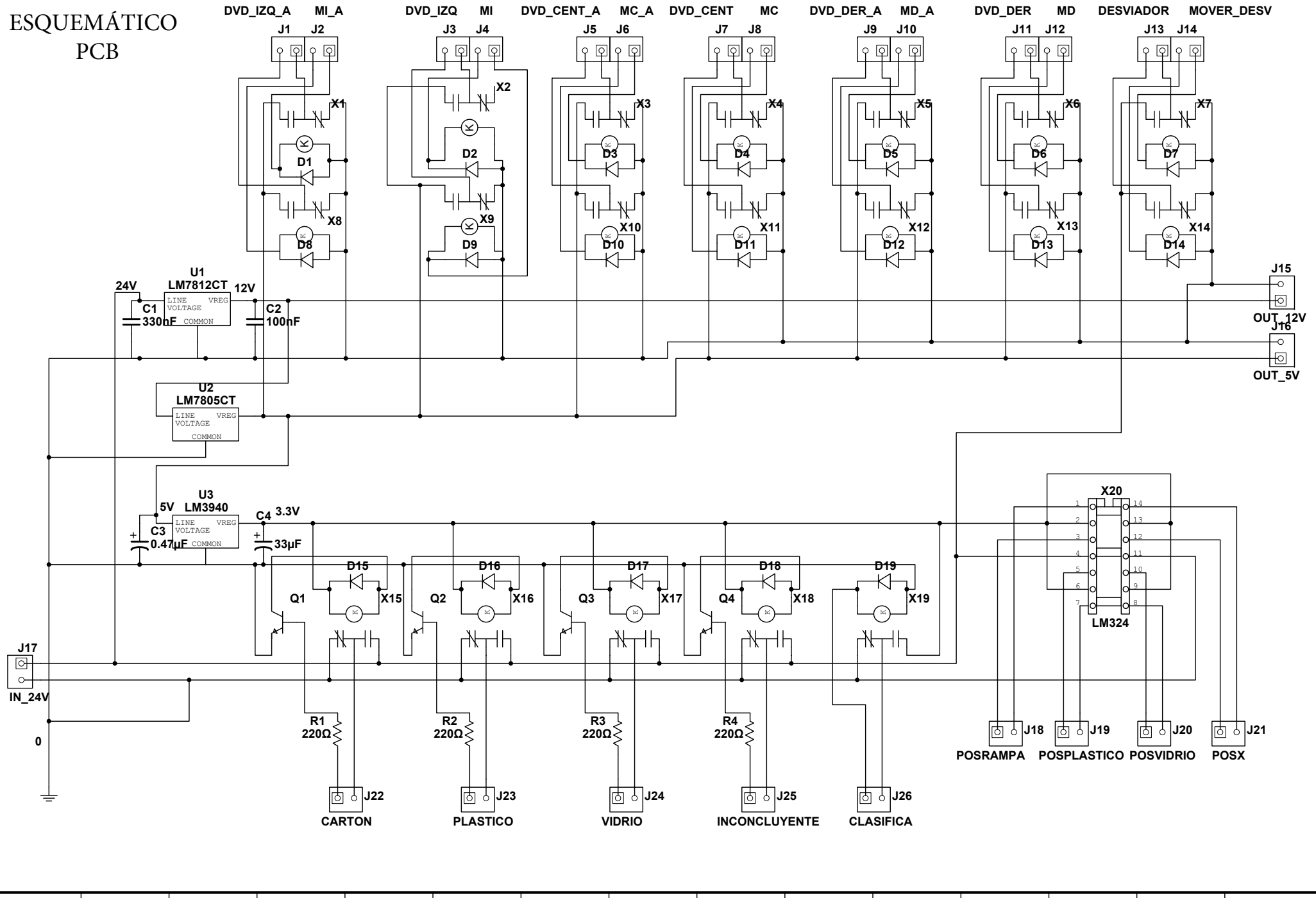




TFG - SISTEMA DE IDENTIFICACIÓN Y CLASIFICACIÓN DE RESIDUOS AUTÓNOMO			
Autores	*Alberto Méndez García* *Stefano Fernando Panzeri Reyes*	Universidad de La Laguna	ESCUELA SUPERIOR DE INGENIERÍA Y TECNOLOGÍA *Grado en Ingeniería Eléctronica Industrial y Automática*
Fecha	*JUNIO 2021*		
ESQUEMA DE CONEXIONADO ELÉCTRICO DEL PROTOTIPO			Nº P. : 3 Nom.Arch: Esquema_Electrico.dwg

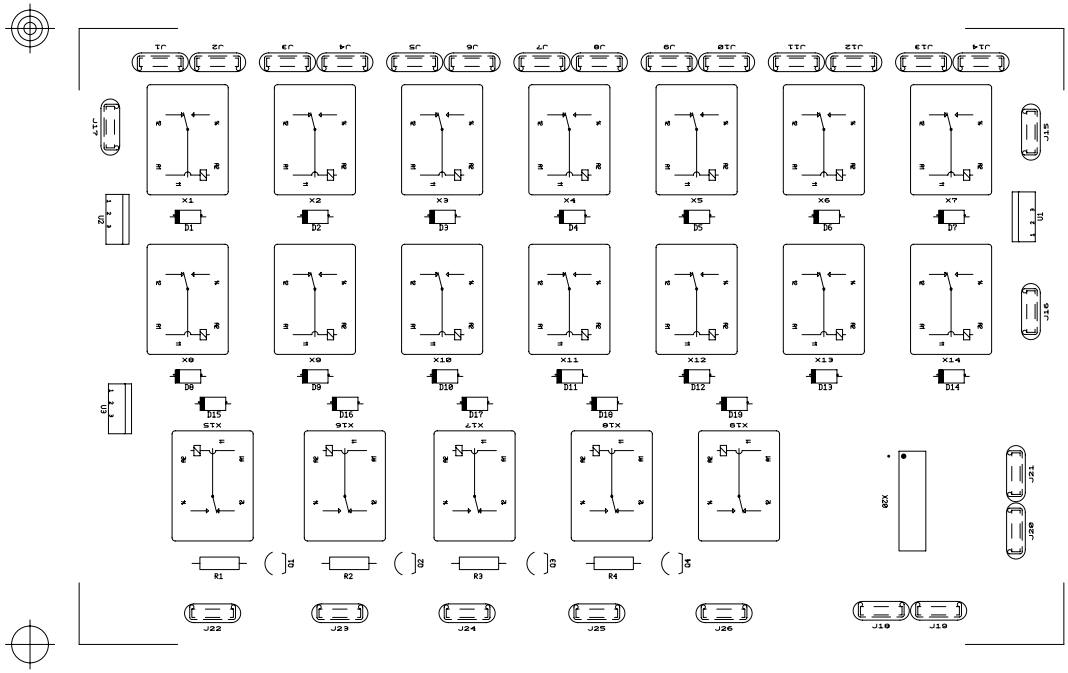
ANEXO IV.
DOCUMENTACIÓN DE LA PCB

ESQUEMÁTICO PCB

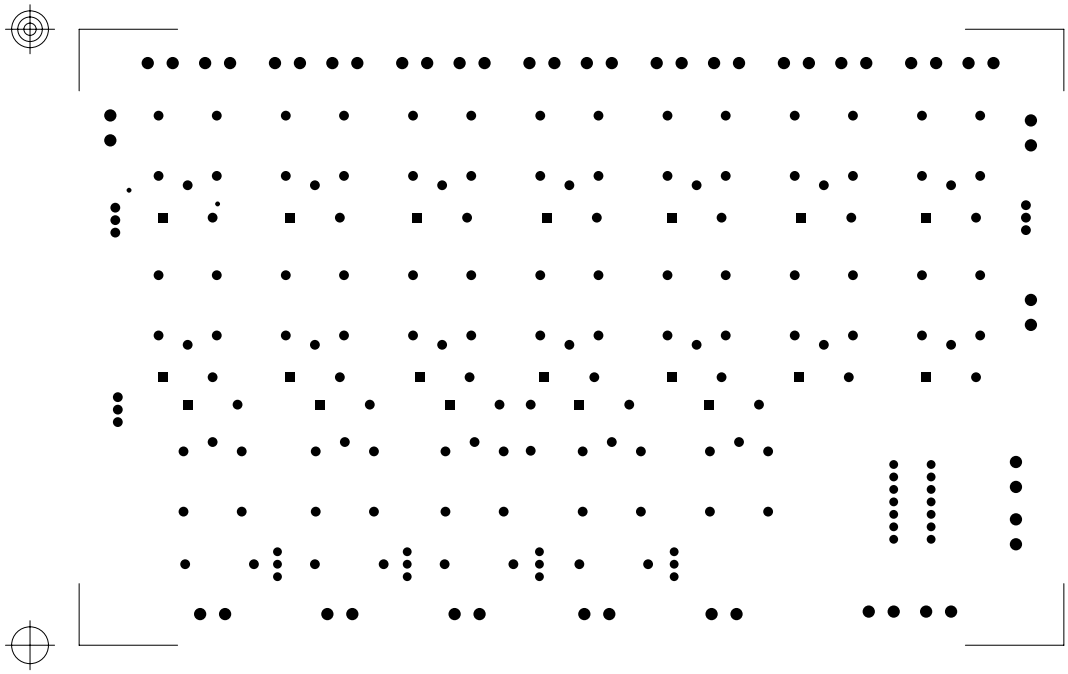


FOTOLITOS EXTRAÍDOS DE LOS GERBERS - PCB

SILKSCREEN TOP

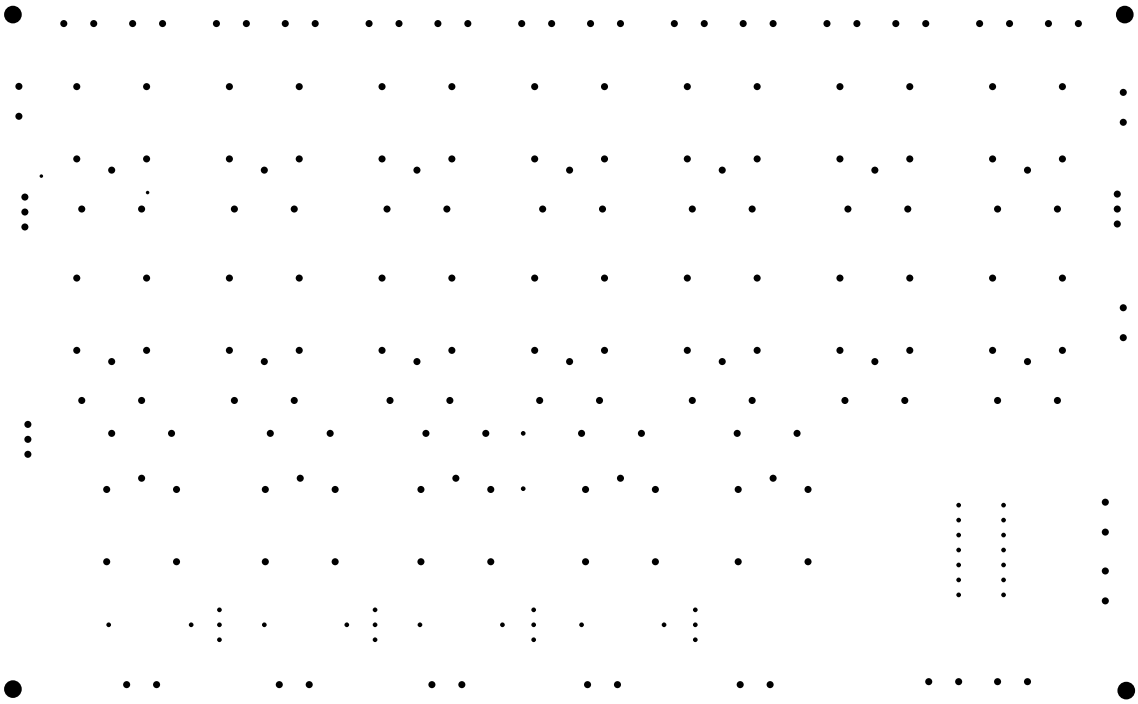


SOLDERMASK BOTTOM

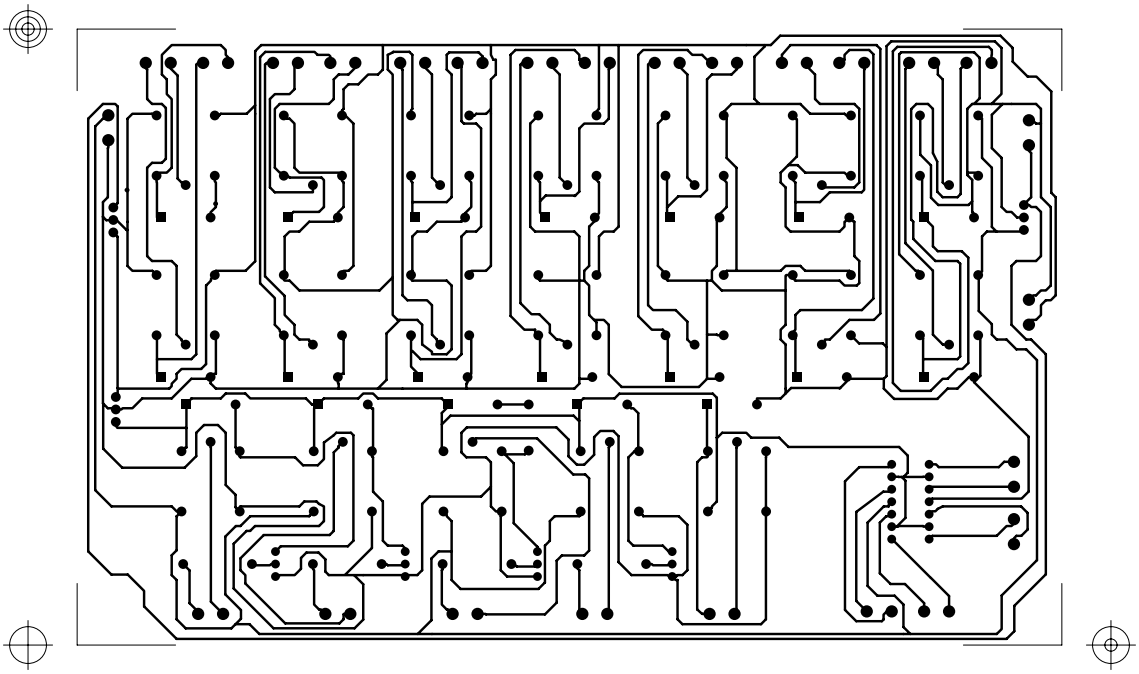


FOTOLITOS EXTRAÍDOS DE LOS GERBERS - PCB

DRILL



COPPER BOTTOM



Ultiboard Information Export File

Design Name : TFG_V11_ReRouteo(TFG_V11_ReRouteo) - Bill Of Materials
Report Date : 11 June 2021
Report Time : 16:21

VALUE	SHAPE	#	REFDES
1N4007GP	DO-41	19	D1,D2,D3,D4,D5,D6,D7,D8,D9,D10,D11,D12,D13,D14,D15,D16,D17,D18,D19
220Ohm	RES1400-800X250	4	R1,R2,R3,R4
BC337AP	TO-92(Bent)	4	Q1,Q2,Q3,Q4
DIP14	DIP14300	1	X20
HDR1X2	FLST2R5	26	J1,J2,J3,J4,J5,J6,J7,J8,J9,J10,J11,J12,J13,J14,J15,J16,J17,J18,J19,J20,J21,J22,J23,J24,J25,J26
LM7805CT	to220	2	U2,U3
LM7812CT	to220	1	U1
RELAY_NONC	RELSIE_T70	19	X1,X2,X3,X4,X5,X6,X7,X8,X9,X10,X11,X12,X13,X14,X15,X16,X17,X18,X19

ULTIBOARD DRILL REPORT

Design: TFG_V11_ReRouteo(TFG_V11_ReRouteo)

DRILL FILE SETTINGS
Excellon 2, Absolute, Inch, 2.5

DEFINITION COPPER LAYERS
Copper Top
Copper Bottom

DRILLFILE	LAMINATION
TFG_V11_ReRouteo(TFG_V11_ReRouteo) - Copper Top-Copper Bottom.drl	Copper Top .. Copper Bottom

TOOL	SIZE	TYPE	COUNT
1	0.11811	NPTH	4
2	0.02362	PTH	2
3	0.03150	PTH	36
4	0.04724	PTH	194

DRILL POINTS

```
M48
INCH,TZ
VER,1
FMAT,2
DETECT,ON
ATC,ON
T1C0.11811
T2C0.02362
T3C0.03150
T4C0.04724
%
T1
X0.20000Y0.21000
Y4.72000
X7.64000Y0.20000
X7.63000Y4.72000
T2
X0.39000Y3.64000
X1.10000Y3.53000
T3
X3.61000Y1.92000
Y1.55000
X2.62000Y0.64000
Y0.54000
Y0.74000
X0.84000Y0.64000
X1.39118
X1.88000
X2.43118
X1.58000
Y0.54000
Y0.74000
X3.68000Y0.64000
Y0.54000
Y0.74000
X2.92000Y0.64000
X3.47118
X4.00000
X4.55118
X4.76000
Y0.54000
Y0.74000
X6.52000Y1.24000
X6.82000
X6.52000Y0.94000
Y0.84000
Y1.14000
Y1.04000
X6.82000Y0.94000
Y0.84000
Y1.14000
Y1.04000
X6.52000Y1.44000
Y1.34000
X6.82000Y1.44000
Y1.34000
T4
```

X4.70667Y3.75500
X4.94000Y3.68000
X4.70667Y4.23833
X5.17333Y3.75500
Y4.23833
X4.70667Y2.47500
X4.94000Y2.40000
X4.70667Y2.95833
X5.17333Y2.47500
Y2.95833
X5.04667Y1.06167
Y1.54500
X5.51333Y1.06167
X5.28000Y1.62000
X5.51333Y1.54500
X5.08000Y4.66000
X5.28000
X5.06000Y0.24000
X5.26000
X4.74000Y2.14000
X5.14000
X5.04000Y1.92000
X5.44000
X4.74000Y3.42000
X5.14000
X2.78000Y4.66000
X2.58000
X1.64667Y2.47500
X2.11333
X1.88000Y2.40000
X1.64667Y2.95833
X2.11333
X0.62667Y2.47500
X1.09333
X0.86000Y2.40000
X0.62667Y2.95833
X1.09333
X0.96000Y0.24000
X1.16000
X1.26000Y1.92000
X0.86000
X1.29333Y1.06167
Y1.54500
X0.82667Y1.06167
Y1.54500
X1.06000Y1.62000
X0.66000Y2.14000
X1.06000
X0.30000Y1.88000
Y1.78000
Y1.98000
X1.88667Y1.06167
X2.35333
X1.88667Y1.54500
X2.12000Y1.62000
X2.35333Y1.54500
X2.18000Y0.24000

X1.98000
X1.68000Y2.14000
X2.08000
X1.92000Y1.92000
X2.32000
X1.00000Y4.66000
X1.20000
X0.62667Y3.75500
X1.09333
X0.86000Y3.68000
X0.62667Y4.23833
X1.09333
X0.66000Y3.42000
X1.06000
X0.28000Y3.50000
Y3.30000
Y3.40000
X0.54000Y4.66000
X0.74000
X0.24000Y4.04000
Y4.24000
X1.64667Y3.75500
X2.11333
X1.88000Y3.68000
X1.64667Y4.23833
X2.11333
X1.68000Y3.42000
X2.08000
X1.56000Y4.66000
X1.76000
X2.22000
X2.02000
X3.68667Y2.47500
X4.15333
X3.92000Y2.40000
X3.68667Y2.95833
X4.15333
X2.66667Y2.47500
X3.13333
X2.90000Y2.40000
X3.13333Y2.95833
X2.66667
X3.72000Y2.14000
X4.12000
X2.92667Y1.06167
X3.39333
X3.16000Y1.62000
X2.92667Y1.54500
X3.39333
X3.20000Y0.24000
X3.00000
X2.96000Y1.92000
X3.36000
X3.12000Y2.14000
X2.72000
X4.02667Y1.06167
X4.49333

X4.02667Y1.54500
X4.26000Y1.62000
X4.49333Y1.54500
X4.04000Y0.24000
X4.24000
X4.40000Y1.92000
X4.00000
X3.68667Y3.75500
Y4.23833
X4.15333Y3.75500
X3.92000Y3.68000
X4.15333Y4.23833
X3.80000Y4.66000
X3.60000
X3.74000Y3.42000
X4.14000
X3.13333Y3.75500
X2.66667
X2.90000Y3.68000
X3.13333Y4.23833
X2.66667
X2.70000Y3.42000
X3.10000
X3.04000Y4.66000
X3.24000
X4.06000
X4.26000
X4.62000
X4.82000
X7.58000Y3.52000
Y3.32000
Y3.42000
X7.62000Y2.56000
Y2.76000
Y4.00000
Y4.20000
X7.50000Y1.00000
Y0.80000
Y1.26000
Y1.46000
X5.72667Y2.47500
X6.19333
X5.96000Y2.40000
X5.72667Y2.95833
X6.19333
X6.74667Y2.47500
X7.21333
X6.98000Y2.40000
X6.74667Y2.95833
X7.21333
X6.32000Y0.26000
X6.52000
X5.76000Y2.14000
X6.16000
X6.98000Y0.26000
X6.78000
Y2.14000

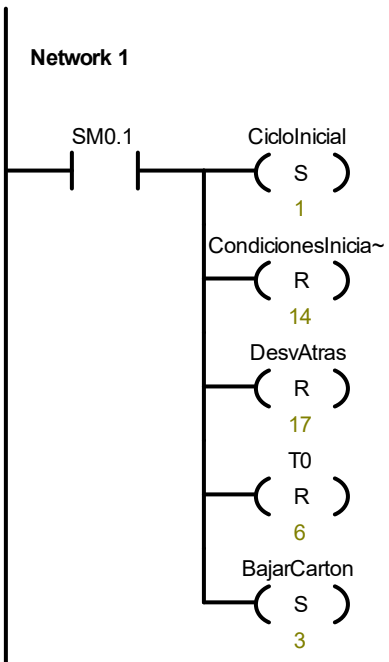
X7.18000
X6.30000Y4.66000
X6.10000
X5.72667Y3.75500
X6.19333
X5.96000Y3.68000
X5.72667Y4.23833
X6.19333
X5.78000Y3.42000
X6.18000
X5.64000Y4.66000
X5.84000
X6.74667Y3.75500
X7.21333
X6.98000Y3.68000
X6.74667Y4.23833
X7.21333
X6.78000Y3.42000
X7.18000
X6.66000Y4.66000
X6.86000
X7.12000
X7.32000
M30

ANEXO V.
CÓDIGO KOP Y TABLA DE SÍMBOLOS

Bloque: PRINCIPAL
Autor:
Fecha de creación: 02.05.2021 13:03:48
Última modificación: 11.06.2021 3:59:01

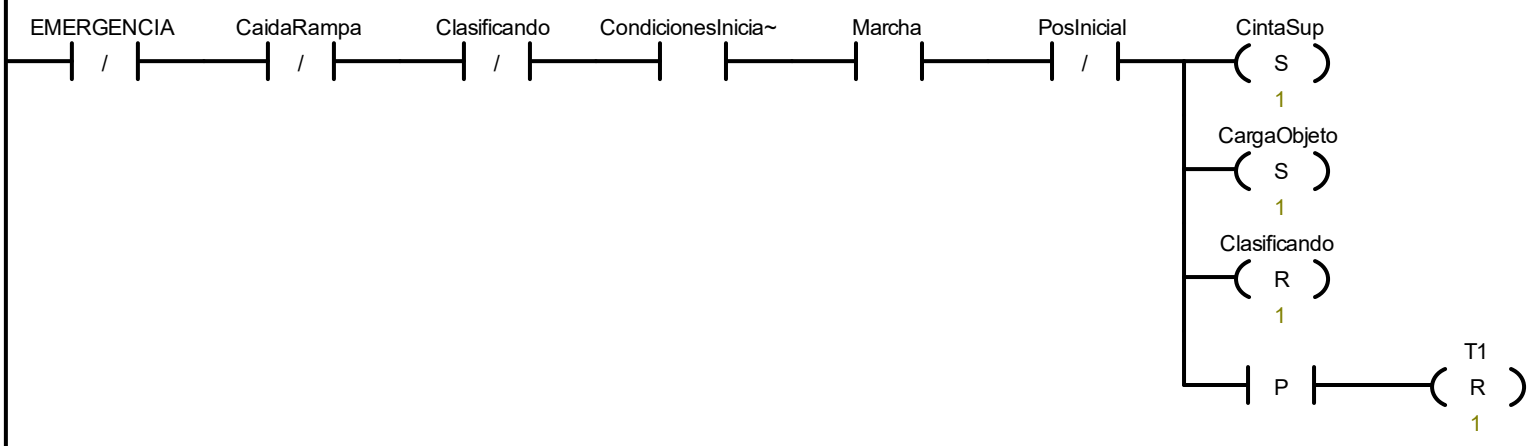
Símbolo	Tipo var.	Tipo de datos	Comentario
	TEMP		
	TEMP		
	TEMP		
	TEMP		

Network 1



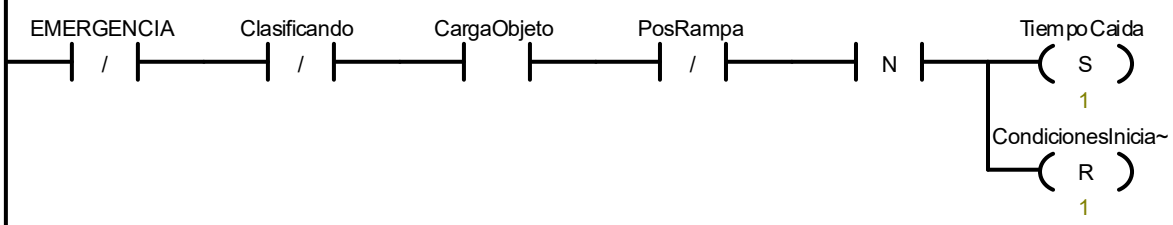
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
CicloInicial	M1.6	Marca que indica que debe ejecutarse el ciclo inicial
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
DesvAtras	M1.7	Marca que indica que el desviador se encuentra en su posición trasera

Network 2



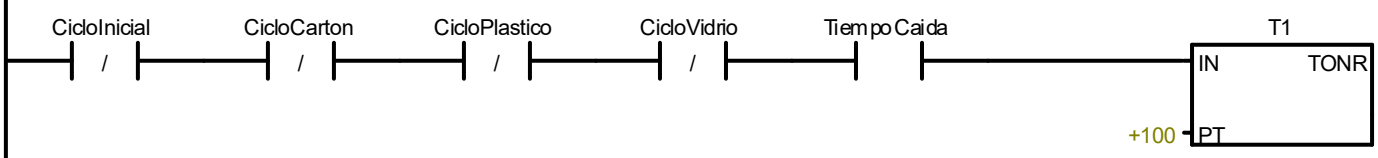
Símbolo	Dirección	Comentario
CaidaRampa	M0.2	Marca que indica que un objeto ha caído por la rampa
CargaObjeto	M0.1	Marca que indica que un objeto ha sido cargado
CintaSup	Q0.0	Movimiento de la cinta superior
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
Marcha	I2.7	Interruptor de marcha
PosInicial	I0.0	Sensor que detecta la presencia de objetos en la cinta superior

Network 3



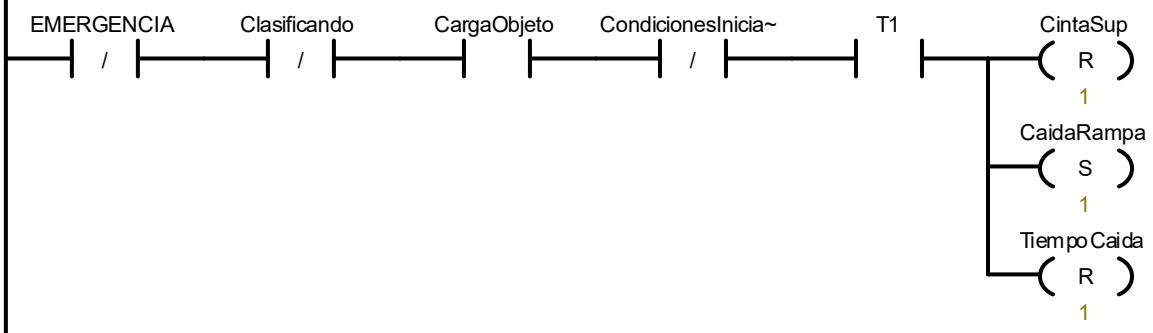
Símbolo	Dirección	Comentario
CargaObjeto	M0.1	Marca que indica que un objeto ha sido cargado
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
PosRampa	I0.1	Sensor que detecta la caída de un objeto por la rampa
TiempoCaida	M2.7	Marca que espera para que se estabilice el objeto antes de capturar la imagen

Network 4



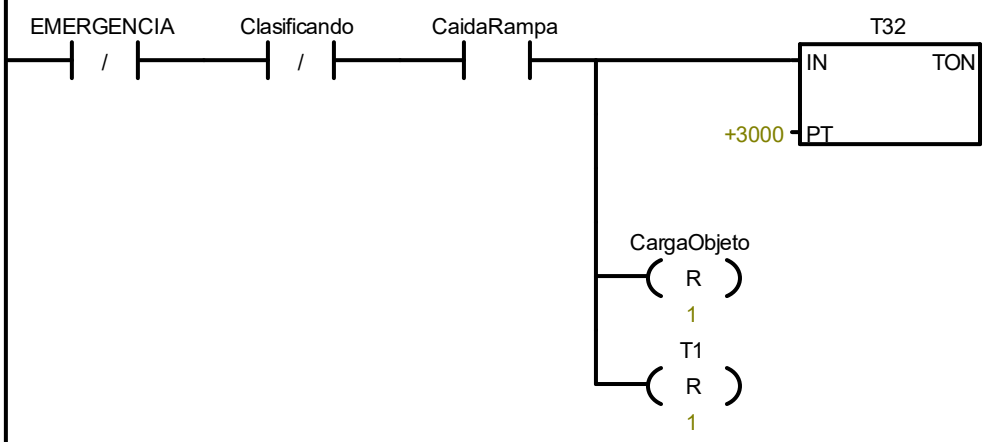
Símbolo	Dirección	Comentario
CicloCarton	M2.3	Marca que indica que se debe de clasificar el objeto como cartón
CicloInicial	M1.6	Marca que indica que debe ejecutarse el ciclo inicial
CicloPlastico	M2.4	Marca que indica que se debe de clasificar el objeto como plástico
CicloVidrio	M2.5	Marca que indica que se debe de clasificar el objeto como vidrio
TiempoCaida	M2.7	Marca que espera para que se estabilice el objeto antes de capturar la imagen

Network 5



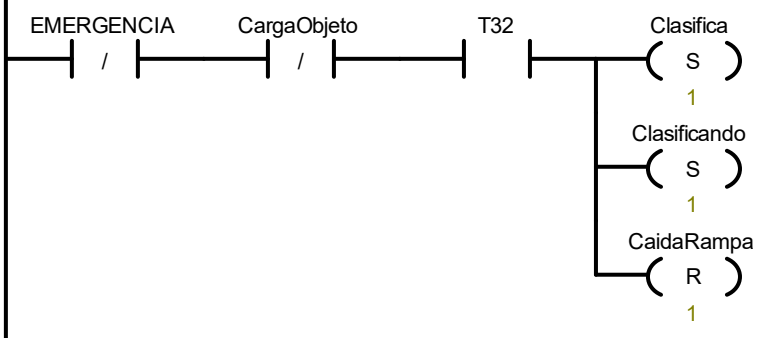
Símbolo	Dirección	Comentario
CaidaRampa	M0.2	Marca que indica que un objeto ha caído por la rampa
CargaObjeto	M0.1	Marca que indica que un objeto ha sido cargado
CintaSup	Q0.0	Movimiento de la cinta superior
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
TiempoCaida	M2.7	Marca que espera para que se estabilice el objeto antes de capturar la imagen

Network 6



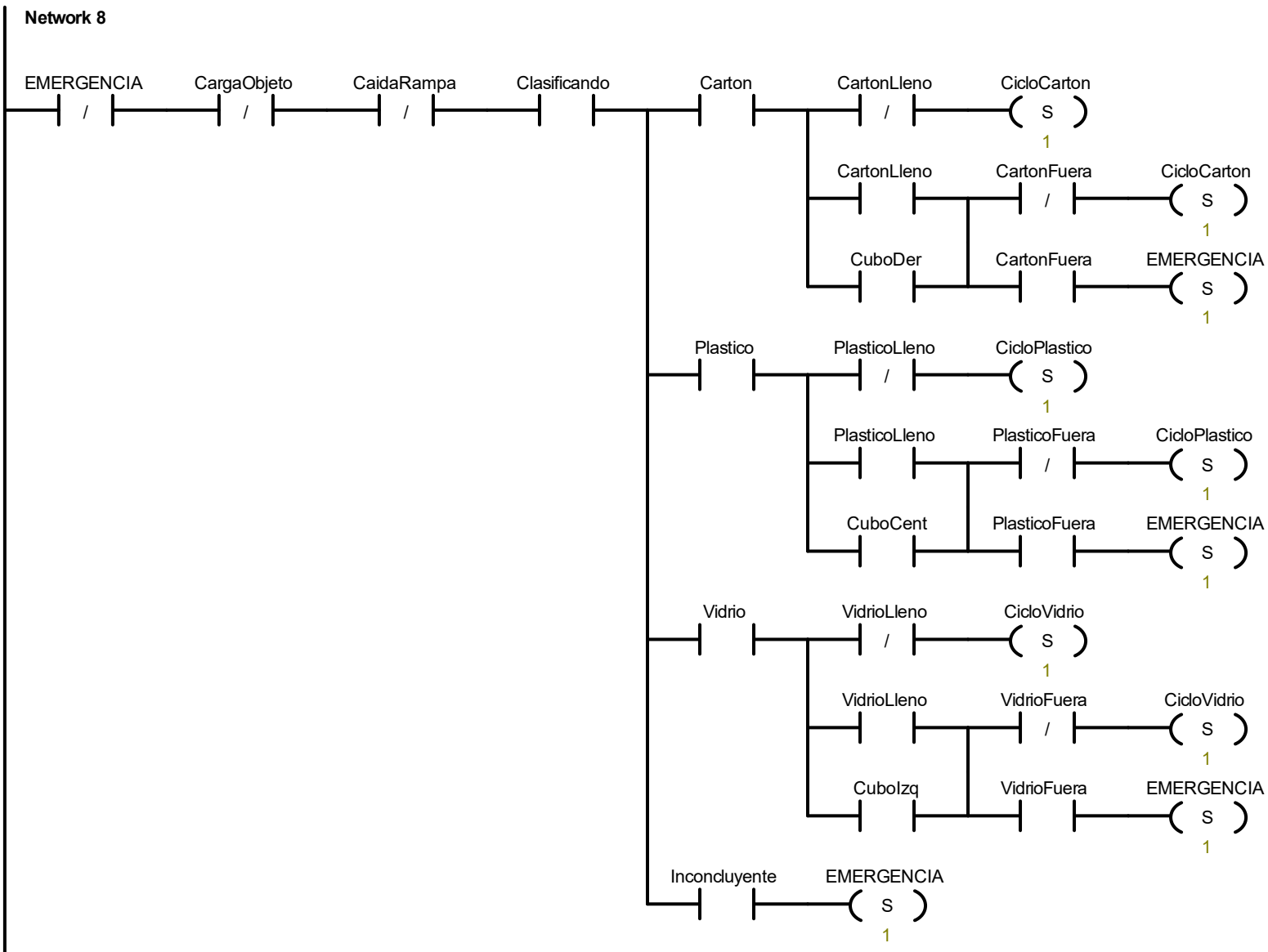
Símbolo	Dirección	Comentario
CaidaRampa	M0.2	Marca que indica que un objeto ha caído por la rampa
CargaObjeto	M0.1	Marca que indica que un objeto ha sido cargado
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
EMERGENCIA	M3.0	Marca que indica un estado de emergencia

Network 7



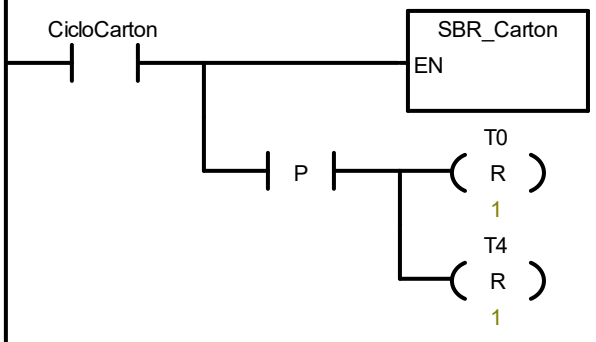
Símbolo	Dirección	Comentario
CaidaRampa	M0.2	Marca que indica que un objeto ha caído por la rampa
CargaObjeto	M0.1	Marca que indica que un objeto ha sido cargado
Clasifica	Q0.2	Señal que le indica a la Raspberry de comenzar la clasificación
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
EMERGENCIA	M3.0	Marca que indica un estado de emergencia

Network 8



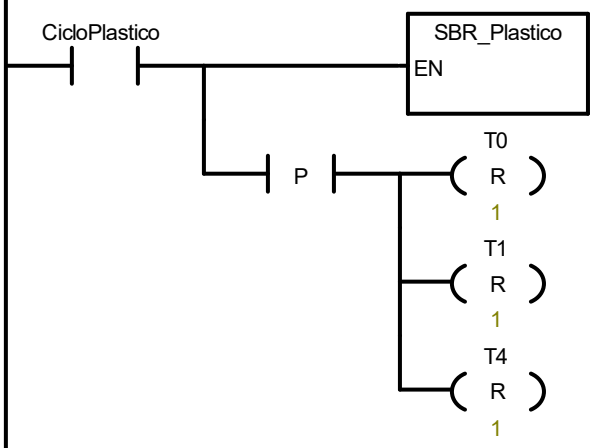
Símbolo	Dirección	Comentario
CaidaRampa	M0.2	Marca que indica que un objeto ha caído por la rampa
CargaObjeto	M0.1	Marca que indica que un objeto ha sido cargado
Carton	I0.2	Entrada que indica que el objeto a clasificar es de cartón
CartonFuera	M2.0	Marca que indica que hay un cubo de cartón fuera
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
CicloCarton	M2.3	Marca que indica que se debe de clasificar el objeto como cartón
CicloPlastico	M2.4	Marca que indica que se debe de clasificar el objeto como plástico
CicloVidrio	M2.5	Marca que indica que se debe de clasificar el objeto como vidrio
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CuboCent	I2.0	Sensor que detecta la presencia de un cubo deelantero de plástico
CuboDer	I2.2	Sensor que detecta la presencia de un cubo delantero de cartón
CuboIzq	I1.4	Sensor que detecta la presencia de un cubo delantero de vidrio
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
Inconcluyente	I0.5	Entrada que indica que se desconoce el material del objeto a clasificar
Plastico	I0.3	Entrada que indica que el objeto a clasificar es de plástico
PlasticoFuera	M2.1	Marca que indica que hay un cubo de plástico fuera
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno
Vidrio	I0.4	Entrada que indica que el objeto a clasificar es de vidrio
VidrioFuera	M2.2	Marca que indica que hay un cubo de cartón fuera
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 9



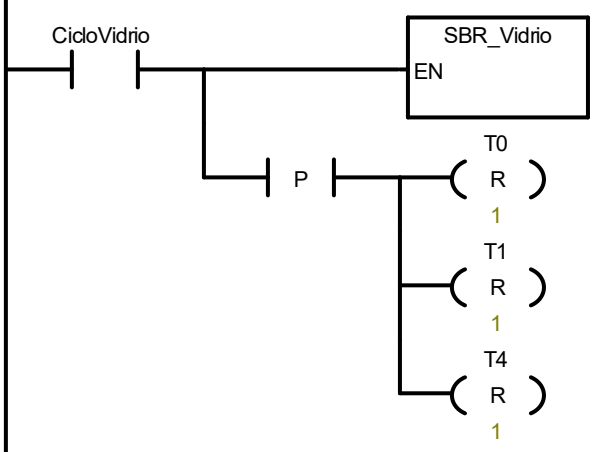
Símbolo	Dirección	Comentario
CicloCarton	M2.3	Marca que indica que se debe de clasificar el objeto como cartón

Network 10



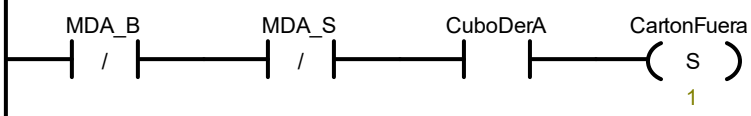
Símbolo	Dirección	Comentario
CicloPlastico	M2.4	Marca que indica que se debe de clasificar el objeto como plástico

Network 11



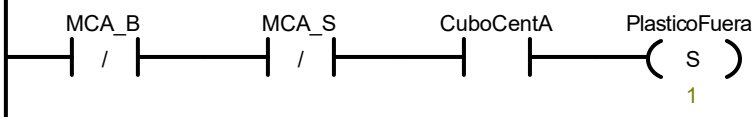
Símbolo	Dirección	Comentario
CicloVidrio	M2.5	Marca que indica que se debe de clasificar el objeto como vidrio

Network 12



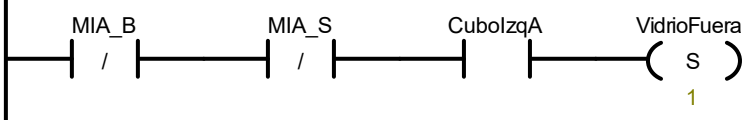
Símbolo	Dirección	Comentario
CartonFuera	M2.0	Marca que indica que hay un cubo de cartón fuera
CuboDerA	I2.1	Sensor que detecta la presencia de un cubo de reserva de cartón
MDA_B	Q2.4	Bajada DVD trasero cartón
MDA_S	Q2.3	Subida DVD trasero cartón

Network 13



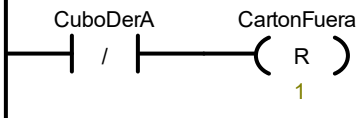
Símbolo	Dirección	Comentario
CuboCentA	I1.5	Sensor que detecta la presencia de un cubo de reserva de plástico
MCA_B	Q2.0	Bajada DVD trasero plástico
MCA_S	Q1.1	Subida DVD trasero plástico
PlasticoFuera	M2.1	Marca que indica que hay un cubo de plástico fuera

Network 14



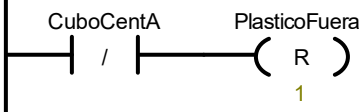
Símbolo	Dirección	Comentario
CuboIzqA	I1.3	Sensor que detecta la presencia de un cubo de reserva de vidrio
MIA_B	Q0.6	Bajada DVD trasero vidrio
MIA_S	Q0.5	Subida DVD trasero vidrio
VidrioFuera	M2.2	Marca que indica que hay un cubo de cartón fuera

Network 15



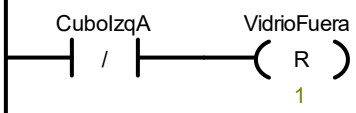
Símbolo	Dirección	Comentario
CartonFuera	M2.0	Marca que indica que hay un cubo de cartón fuera
CuboDerA	I2.1	Sensor que detecta la presencia de un cubo de reserva de cartón

Network 16



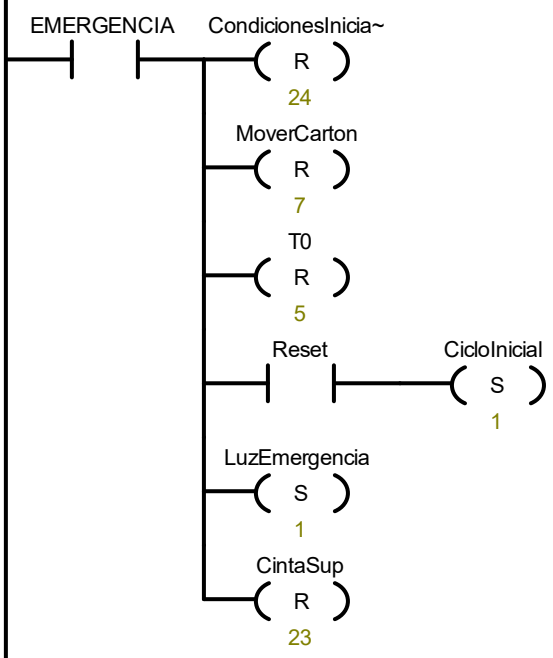
Símbolo	Dirección	Comentario
CuboCentA	I1.5	Sensor que detecta la presencia de un cubo de reserva de plástico
PlasticoFuera	M2.1	Marca que indica que hay un cubo de plástico fuera

Network 17



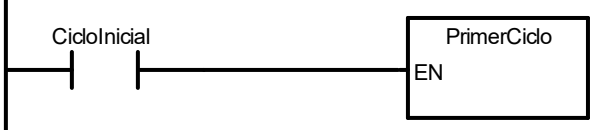
Símbolo	Dirección	Comentario
CuboIzqA	I1.3	Sensor que detecta la presencia de un cubo de reserva de vidrio
VidrioFuera	M2.2	Marca que indica que hay un cubo de cartón fuera

Network 18



Símbolo	Dirección	Comentario
CicloInicial	M1.6	Marca que indica que debe ejecutarse el ciclo inicial
CintaSup	Q0.0	Movimiento de la cinta superior
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
LuzEmergencia	Q2.7	Piloso luminoso de emergencia
MoverCarton	M3.1	Marca que indica que se debe rodar el cubo del cartón en el ciclo inicial
Reset	I2.6	Pulsador de Rearme

Network 19



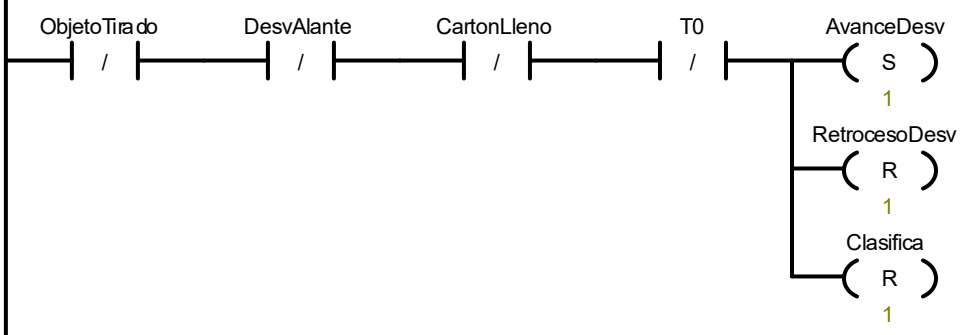
Símbolo	Dirección	Comentario
CicloInicial	M1.6	Marca que indica que debe ejecutarse el ciclo inicial

Símbolo	Dirección	Comentario
CicloInicial	M1.6	Marca que indica que debe ejecutarse el ciclo inicial

Bloque: SBR_Carton
Autor:
Fecha de creación: 02.05.2021 13:03:48
Última modificación: 11.06.2021 2:10:59

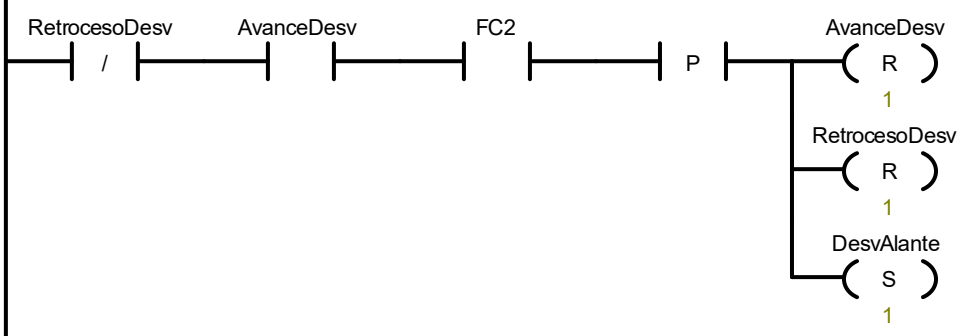
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



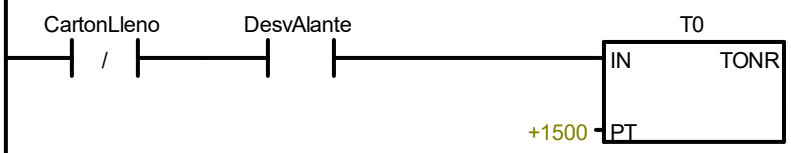
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
Clasifica	Q0.2	Señal que le indica a la Raspberry de comenzar la clasificación
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 2



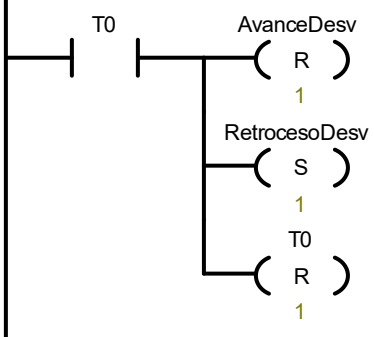
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
FC2	I1.1	Final de carrera del retroceso del desviador
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 3



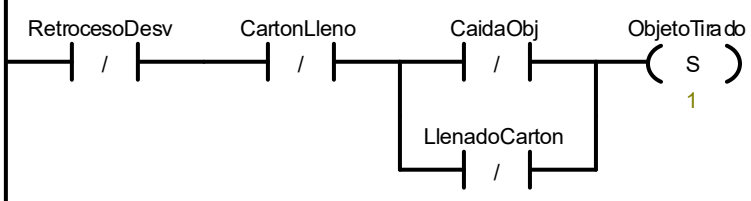
Símbolo	Dirección	Comentario
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera

Network 4



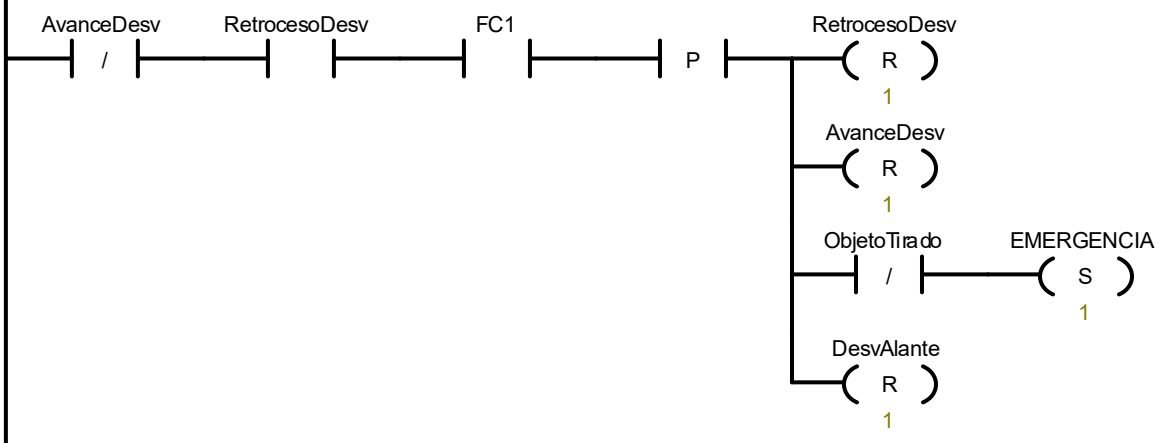
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 5



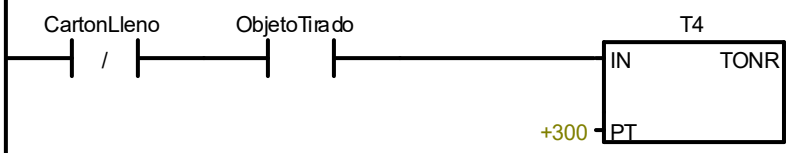
Símbolo	Dirección	Comentario
CaidaObj	I1.2	Sensor que detecta la caída de los objetos en los cubos
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
LlenadoCarton	I2.5	Sensor que detecta el llenado del cubo de cartón
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 6



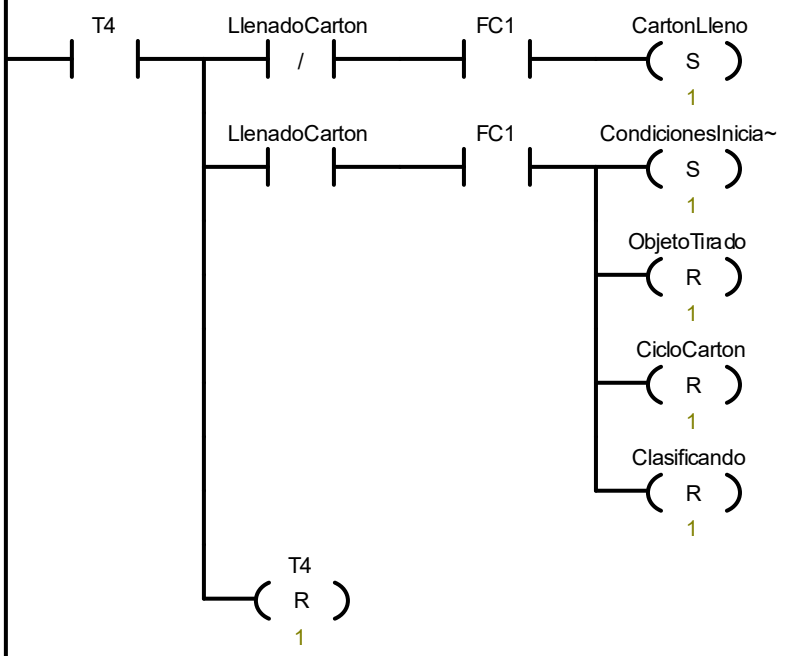
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
FC1	I1.0	Final de carrera del avance del desviador
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 7



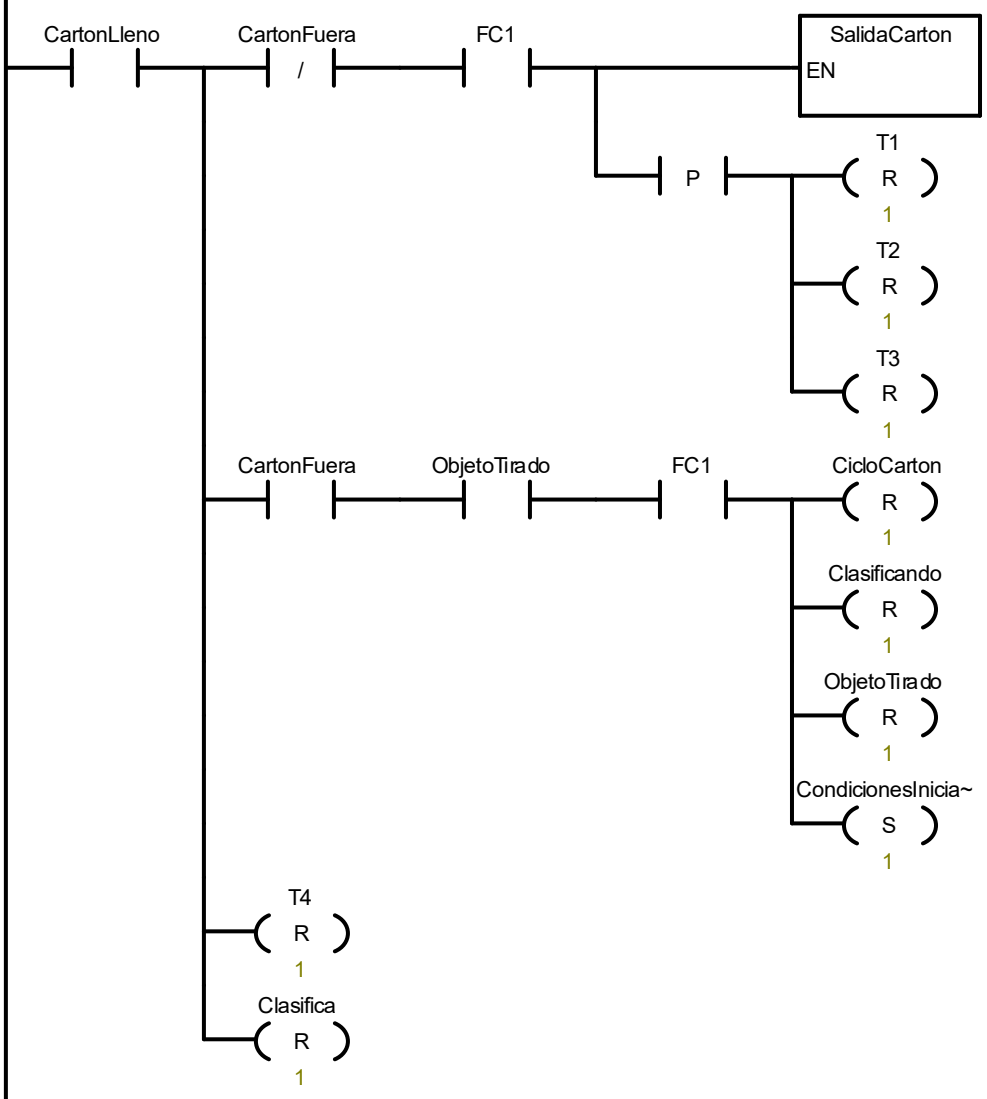
Símbolo	Dirección	Comentario
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente

Network 8



Símbolo	Dirección	Comentario
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
CicloCarton	M2.3	Marca que indica que se debe de clasificar el objeto como cartón
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
FC1	I1.0	Final de carrera del avance del desviador
LlenadoCarton	I2.5	Sensor que detecta el llenado del cubo de cartón
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente

Network 9

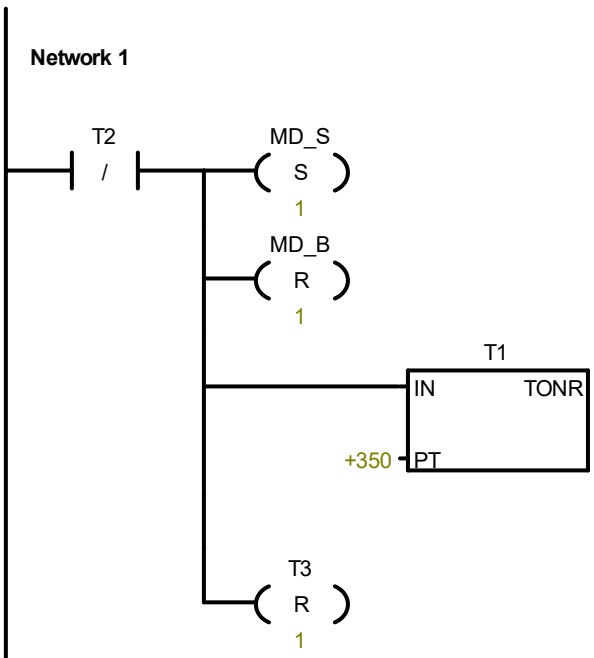


Símbolo	Dirección	Comentario
CartonFuera	M2.0	Marca que indica que hay un cubo de cartón fuera
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
CicloCarton	M2.3	Marca que indica que se debe de clasificar el objeto como cartón
Clasifica	Q0.2	Señal que le indica a la Raspberry de comenzar la clasificación
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
FC1	I1.0	Final de carrera del avance del desviador
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente

Bloque: SalidaCarton
Autor:
Fecha de creación: 06.05.2021 15:03:26
Última modificación: 11.06.2021 2:11:08

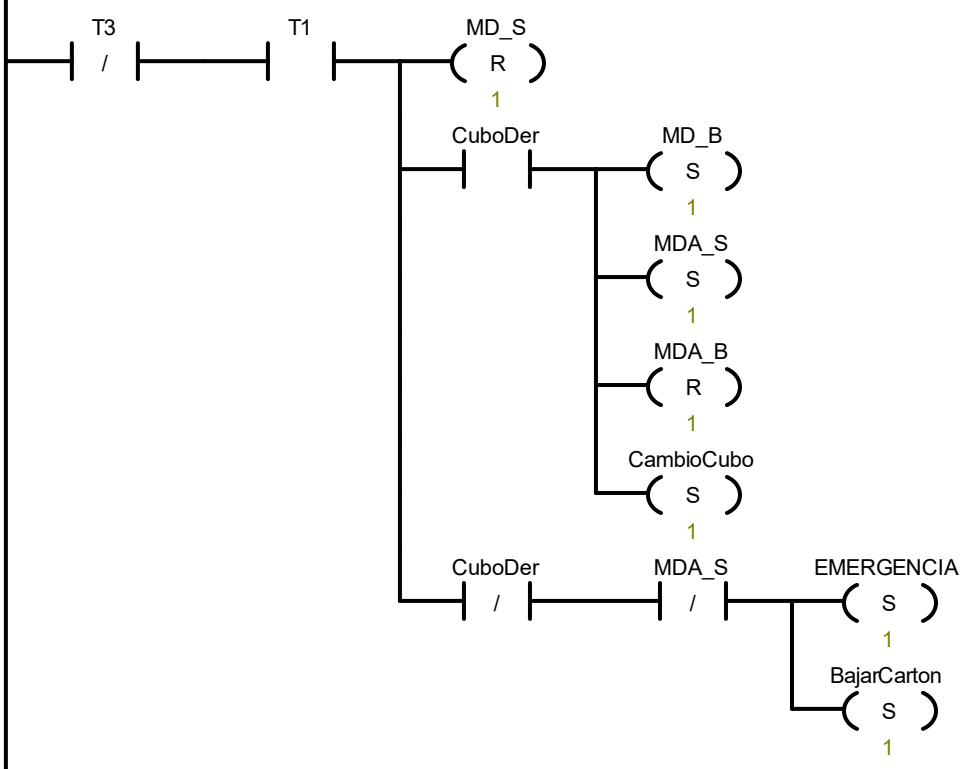
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



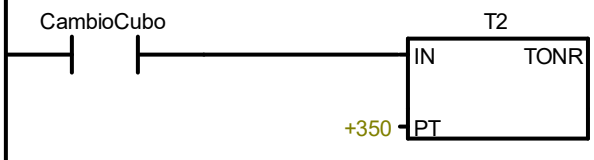
Símbolo	Dirección	Comentario
MD_B	Q2.6	Bajada DVD delantero cartón
MD_S	Q2.5	Subida DVD delantero cartón

Network 2



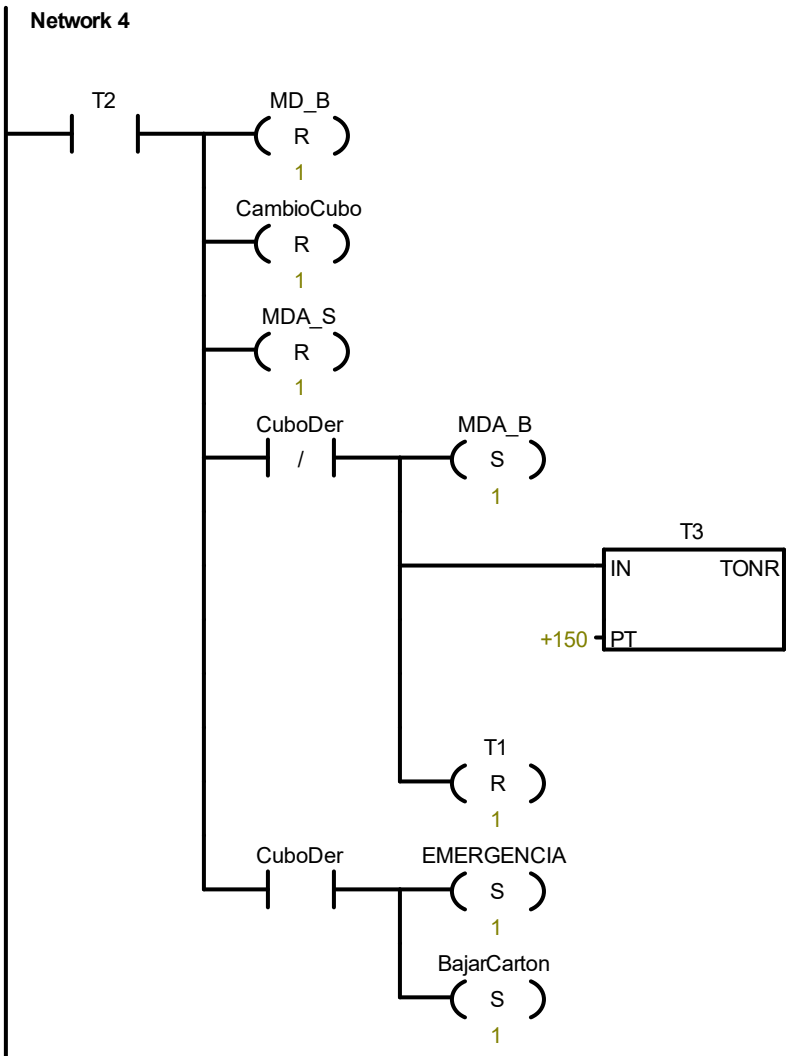
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboDer	I2.2	Sensor que detecta la presencia de un cubo delantero de cartón
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MD_B	Q2.6	Bajada DVD delantero cartón
MD_S	Q2.5	Subida DVD delantero cartón
MDA_B	Q2.4	Bajada DVD trasero cartón
MDA_S	Q2.3	Subida DVD trasero cartón

Network 3



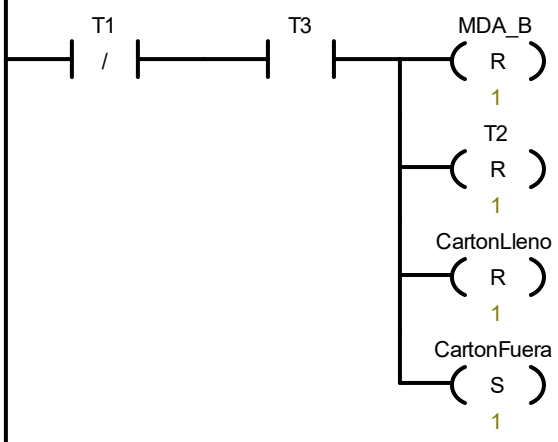
Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo

Network 4



Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboDer	I2.2	Sensor que detecta la presencia de un cubo delantero de cartón
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MD_B	Q2.6	Bajada DVD delantero cartón
MDA_B	Q2.4	Bajada DVD trasero cartón
MDA_S	Q2.3	Subida DVD trasero cartón

Network 5

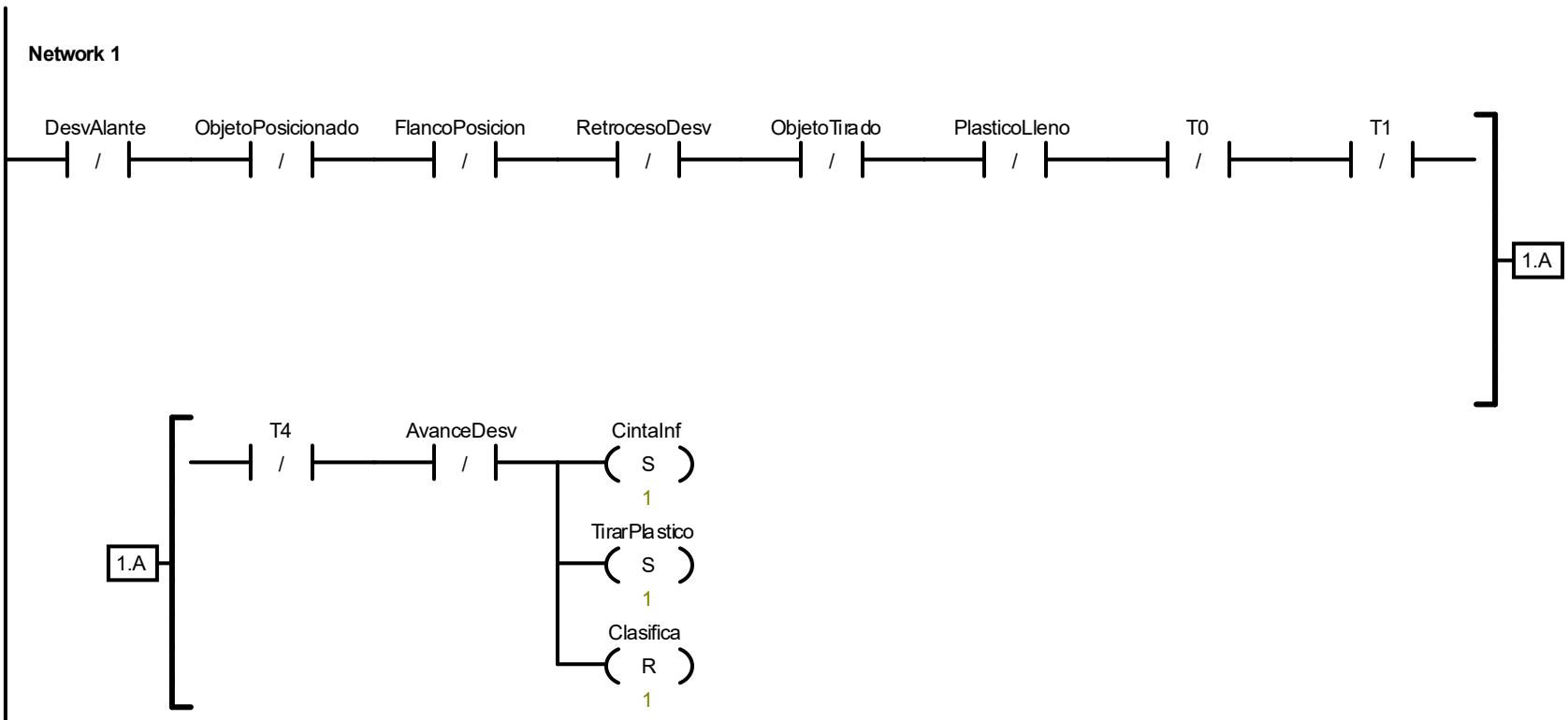


Símbolo	Dirección	Comentario
CartonFuera	M2.0	Marca que indica que hay un cubo de cartón fuera
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
MDA_B	Q2.4	Bajada DVD trasero cartón

Bloque: SBR_Plastico
Autor:
Fecha de creación: 02.05.2021 13:03:48
Última modificación: 08.06.2021 10:32:14

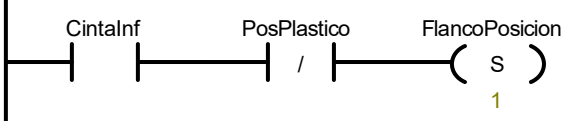
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



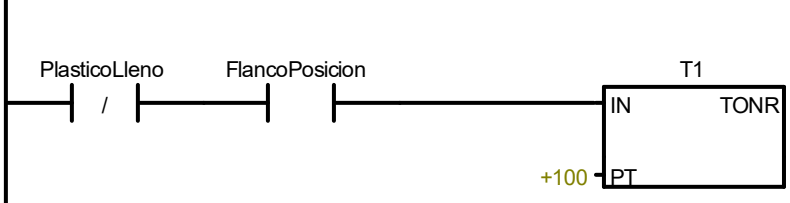
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
CintaInf	Q0.1	Movimiento de la cinta inferior
Clasifica	Q0.2	Señal que le indica a la Raspberry de comenzar la clasificación
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
ObjetoPosicionado	M0.6	Marca que indica que el objeto ha sido posicionado
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador
TirarPlastico	M0.4	Marca que indica que se debe posicionar plástico

Network 2



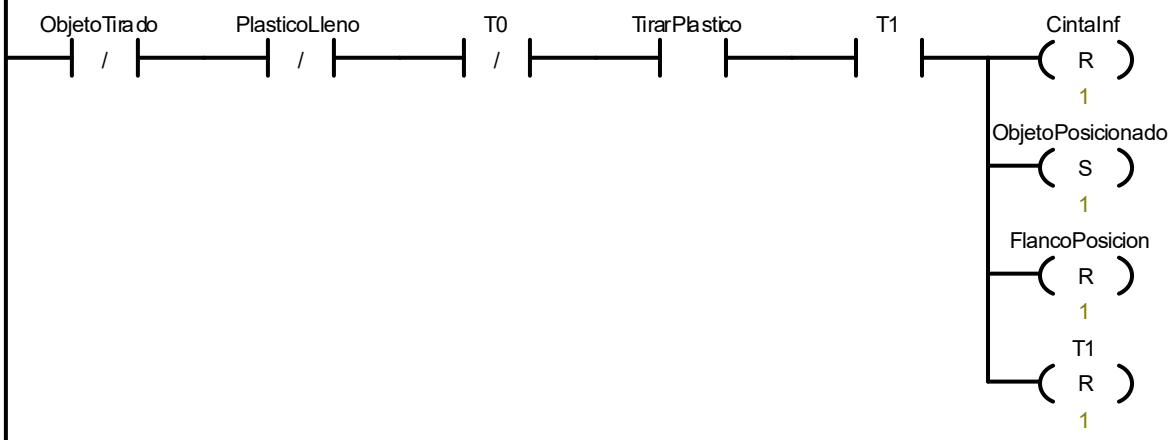
Símbolo	Dirección	Comentario
CintaInf	Q0.1	Movimiento de la cinta inferior
FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
PosPlastico	I0.6	Sensor que indica que el objeto plástico está posicionado

Network 3



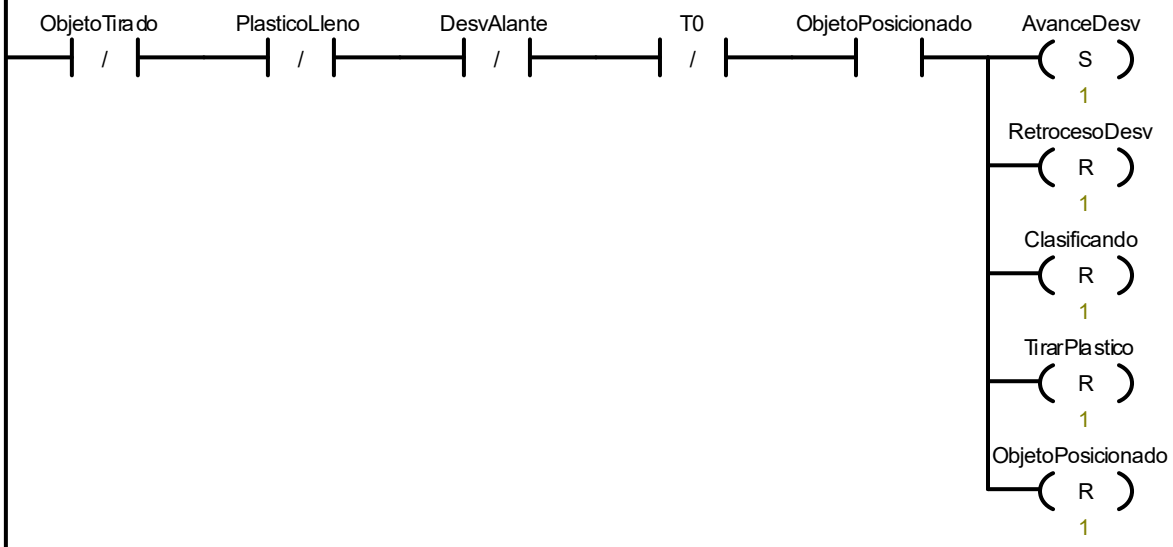
Símbolo	Dirección	Comentario
FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno

Network 4



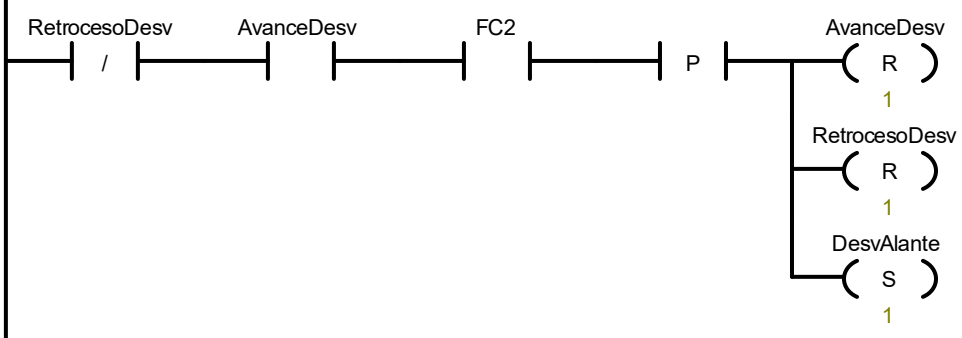
Símbolo	Dirección	Comentario
CintaInf	Q0.1	Movimiento de la cinta inferior
FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
ObjetoPosicionado	M0.6	Marca que indica que el objeto ha sido posicionado
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno
TirarPlastico	M0.4	Marca que indica que se debe posicionar plástico

Network 5



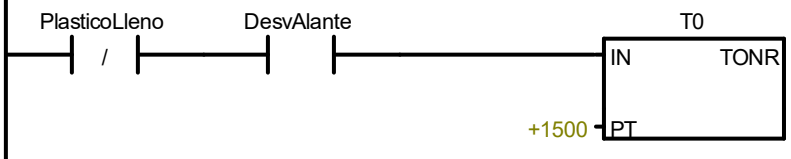
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
ObjetoPosicionado	M0.6	Marca que indica que el objeto ha sido posicionado
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador
TirarPlastico	M0.4	Marca que indica que se debe posicionar plástico

Network 6



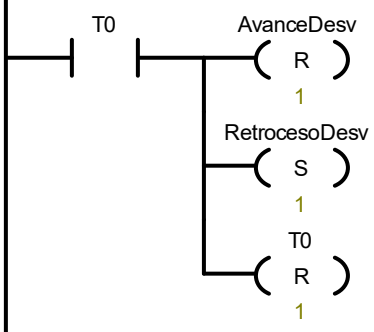
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
FC2	I1.1	Final de carrera del retroceso del desviador
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 7



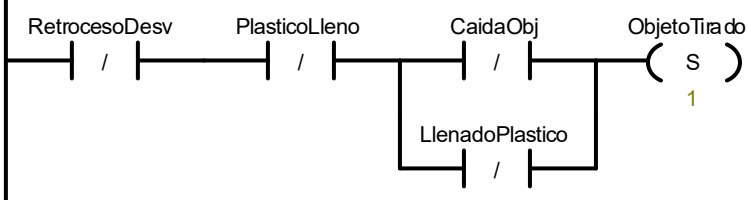
Símbolo	Dirección	Comentario
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno

Network 8



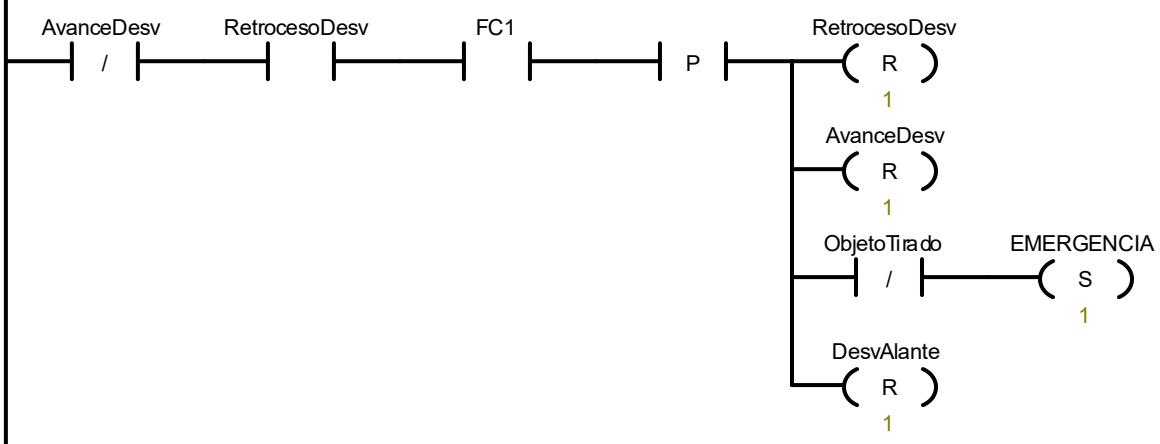
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 9



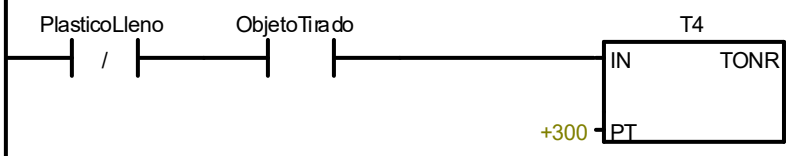
Símbolo	Dirección	Comentario
CaidaObj	I1.2	Sensor que detecta la caída de los objetos en los cubos
LlenadoPlastico	I2.4	Sensor que detecta el llenado del cubo de plástico
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 10



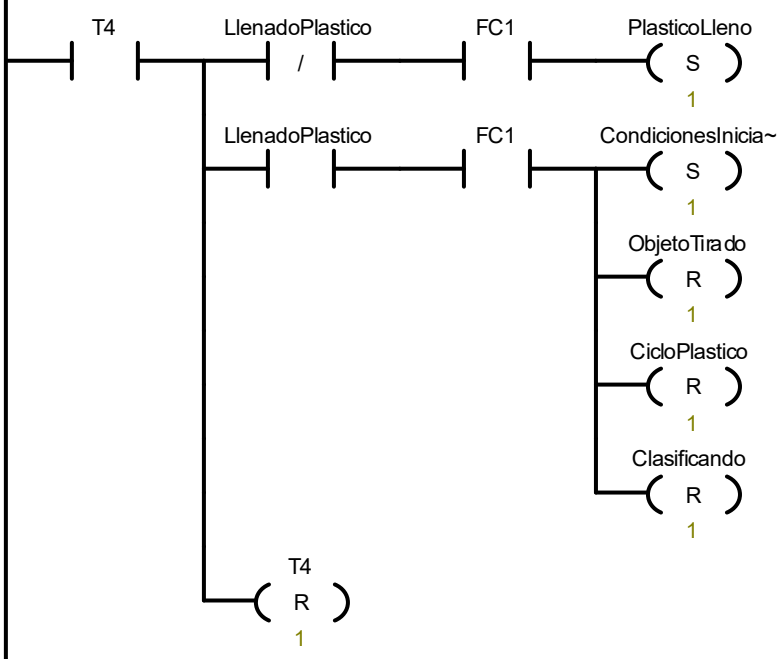
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
FC1	I1.0	Final de carrera del avance del desviador
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 11



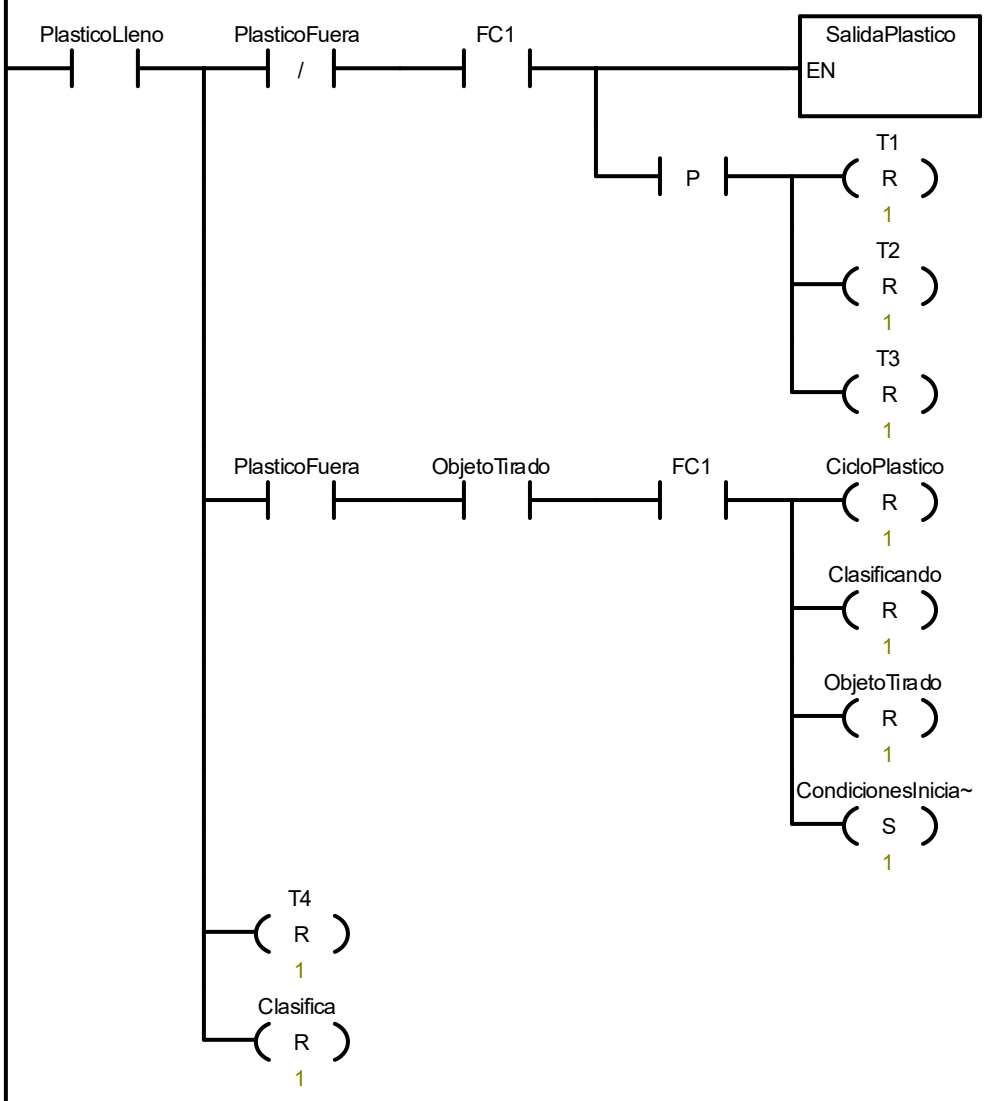
Símbolo	Dirección	Comentario
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno

Network 12



Símbolo	Dirección	Comentario
CicloPlastico	M2.4	Marca que indica que se debe de clasificar el objeto como plástico
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
FC1	I1.0	Final de carrera del avance del desviador
LlenadoPlastico	I2.4	Sensor que detecta el llenado del cubo de plástico
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno

Network 13

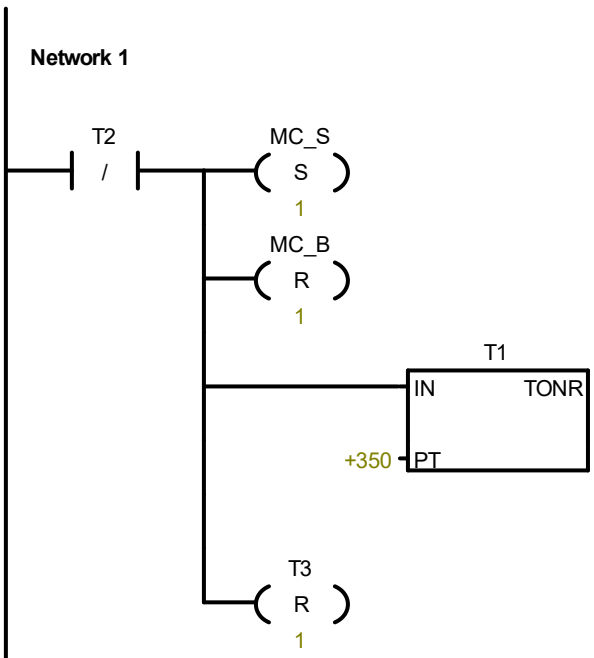


Símbolo	Dirección	Comentario
CicloPlastico	M2.4	Marca que indica que se debe de clasificar el objeto como plástico
Clasifica	Q0.2	Señal que le indica a la Raspberry de comenzar la clasificación
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
FC1	I1.0	Final de carrera del avance del desviador
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
PlasticoFuera	M2.1	Marca que indica que hay un cubo de plástico fuera
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno

Bloque: SalidaPlastico
Autor:
Fecha de creación: 06.05.2021 15:03:26
Última modificación: 11.06.2021 2:11:26

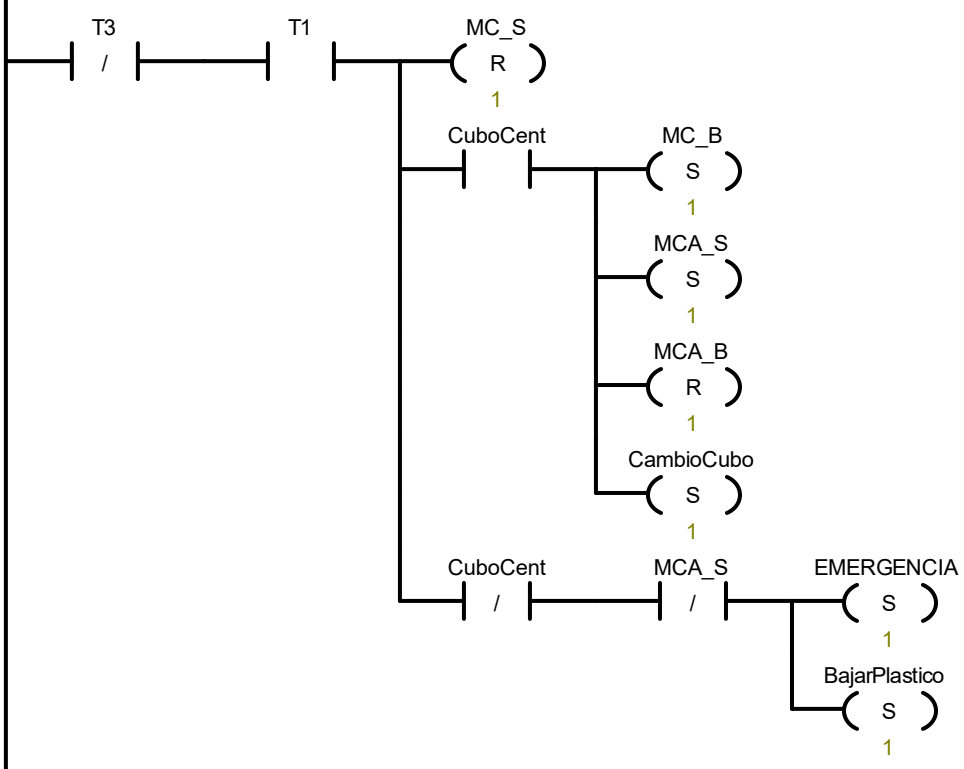
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



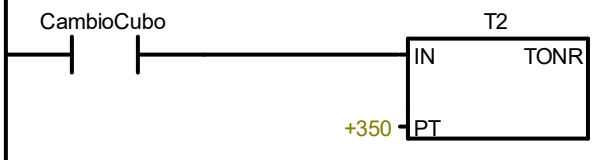
Símbolo	Dirección	Comentario
MC_B	Q2.2	Bajada DVD delantero plástico
MC_S	Q2.1	Subida DVD delantero plástico

Network 2



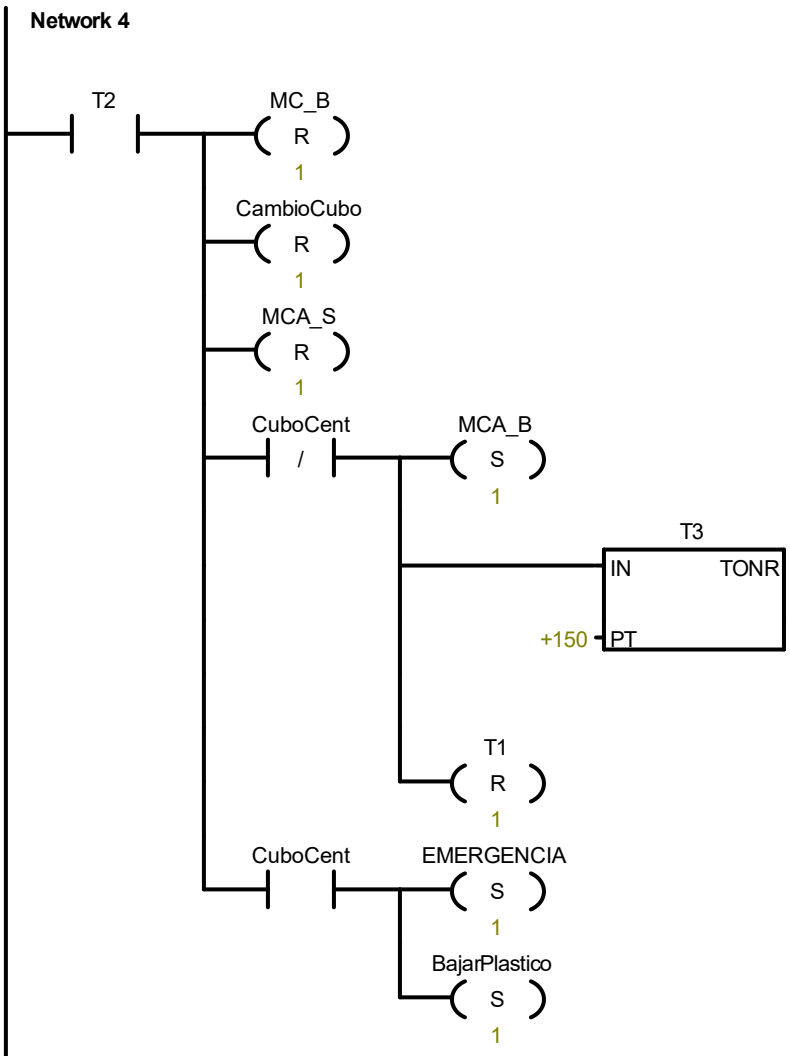
Símbolo	Dirección	Comentario
BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboCent	I2.0	Sensor que detecta la presencia de un cubo deelantero de plástico
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MC_B	Q2.2	Bajada DVD delantero plástico
MC_S	Q2.1	Subida DVD delantero plástico
MCA_B	Q2.0	Bajada DVD trasero plástico
MCA_S	Q1.1	Subida DVD trasero plástico

Network 3



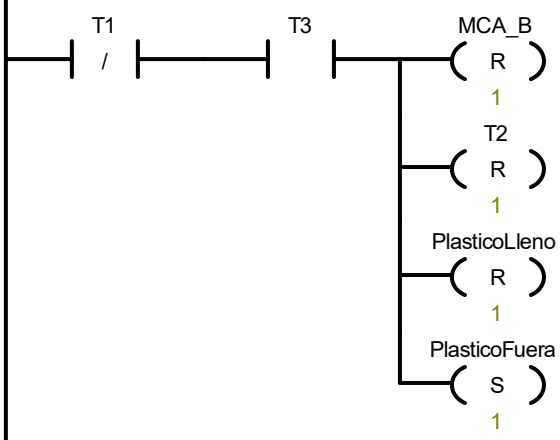
Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo

Network 4



Símbolo	Dirección	Comentario
BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboCent	I2.0	Sensor que detecta la presencia de un cubo deelantero de plástico
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MC_B	Q2.2	Bajada DVD delantero plástico
MCA_B	Q2.0	Bajada DVD trasero plástico
MCA_S	Q1.1	Subida DVD trasero plástico

Network 5

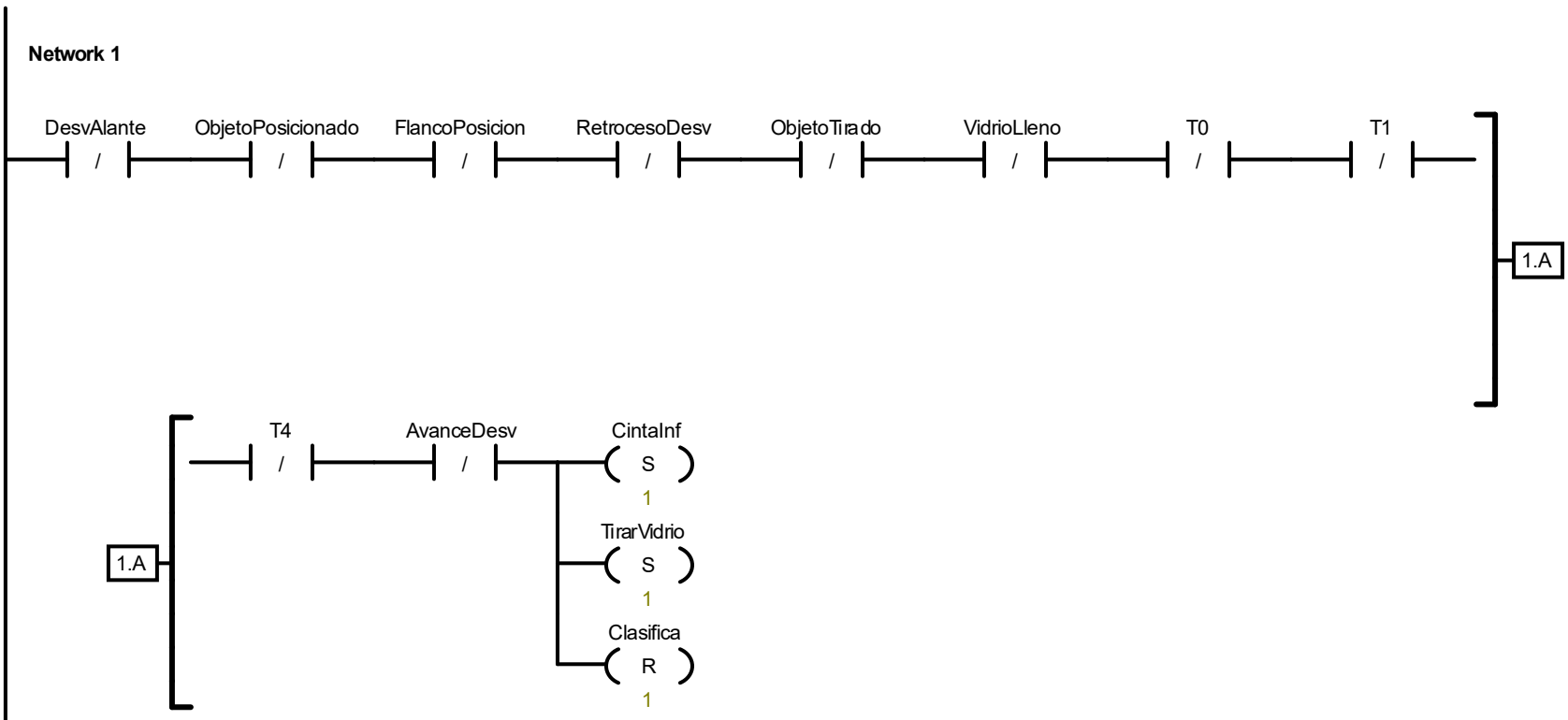


Símbolo	Dirección	Comentario
MCA_B	Q2.0	Bajada DVD trasero plástico
PlasticoFuera	M2.1	Marca que indica que hay un cubo de plástico fuera
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno

Bloque: SBR_Vidrio
Autor:
Fecha de creación: 02.05.2021 13:03:48
Última modificación: 11.06.2021 2:11:36

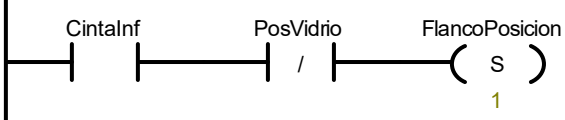
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



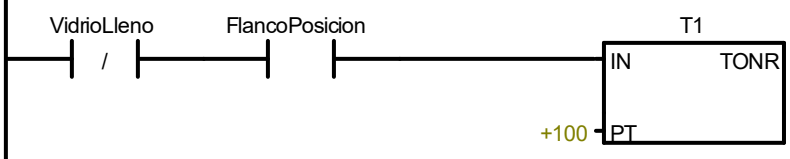
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
CintaInf	Q0.1	Movimiento de la cinta inferior
Clasifica	Q0.2	Señal que le indica a la Raspberry de comenzar la clasificación
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
ObjetoPosicionado	M0.6	Marca que indica que el objeto ha sido posicionado
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador
TirarVidrio	M0.5	Marca que indica que se debe posicionar vidrio
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 2



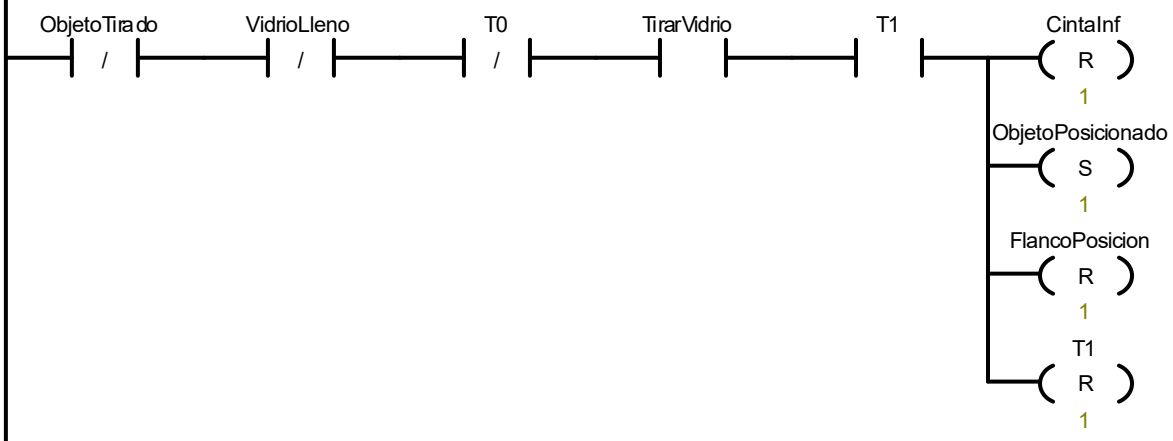
Símbolo	Dirección	Comentario
CintaInf	Q0.1	Movimiento de la cinta inferior
FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
PosVidrio	I0.7	Sensor que indica que el objeto de vidrio está posicionado

Network 3



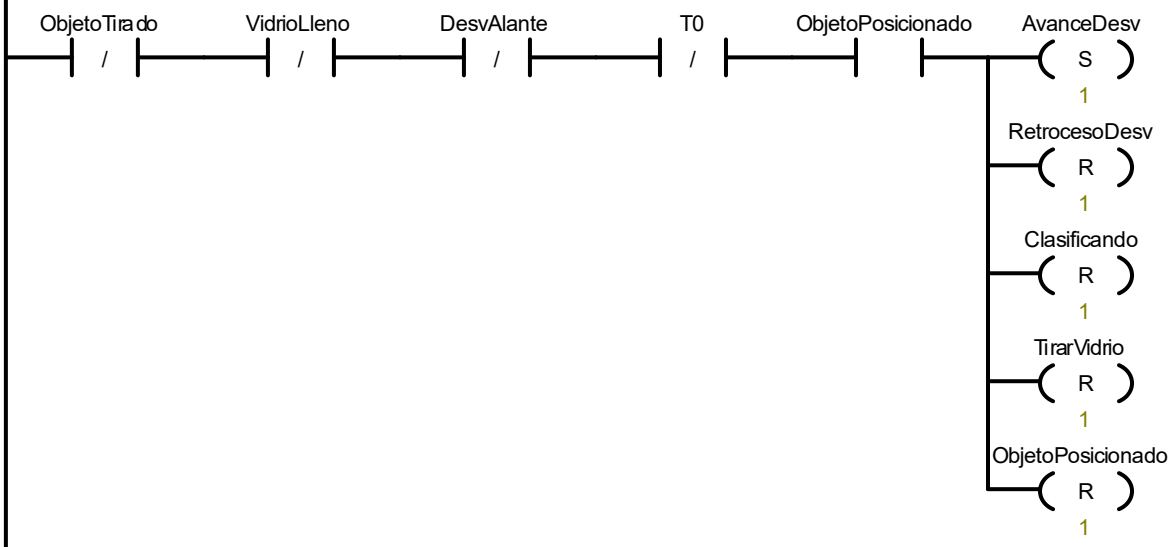
Símbolo	Dirección	Comentario
FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 4



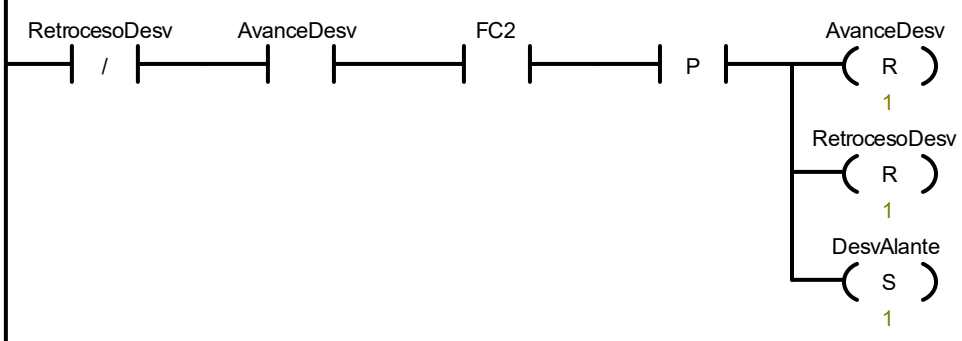
Símbolo	Dirección	Comentario
CintaInf	Q0.1	Movimiento de la cinta inferior
FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
ObjetoPosicionado	M0.6	Marca que indica que el objeto ha sido posicionado
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
TirarVidrio	M0.5	Marca que indica que se debe posicionar vidrio
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 5



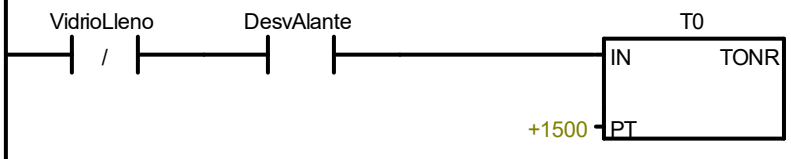
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
ObjetoPosicionado	M0.6	Marca que indica que el objeto ha sido posicionado
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador
TirarVidrio	M0.5	Marca que indica que se debe posicionar vidrio
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 6



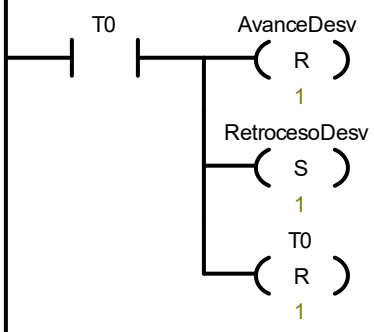
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
FC2	I1.1	Final de carrera del retroceso del desviador
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 7



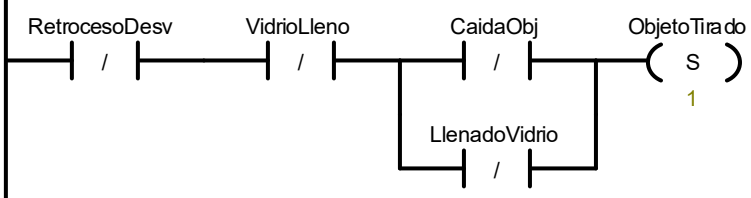
Símbolo	Dirección	Comentario
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 8



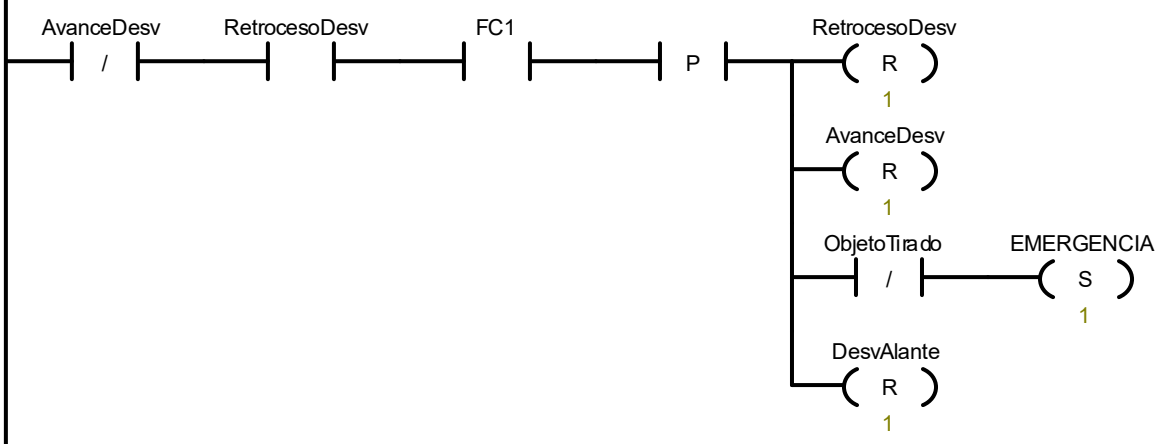
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 9



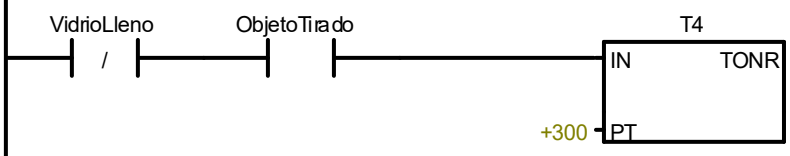
Símbolo	Dirección	Comentario
CaidaObj	I1.2	Sensor que detecta la caída de los objetos en los cubos
LlenadoVidrio	I2.3	Sensor que detecta el llenado del cubo de vidrio
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 10



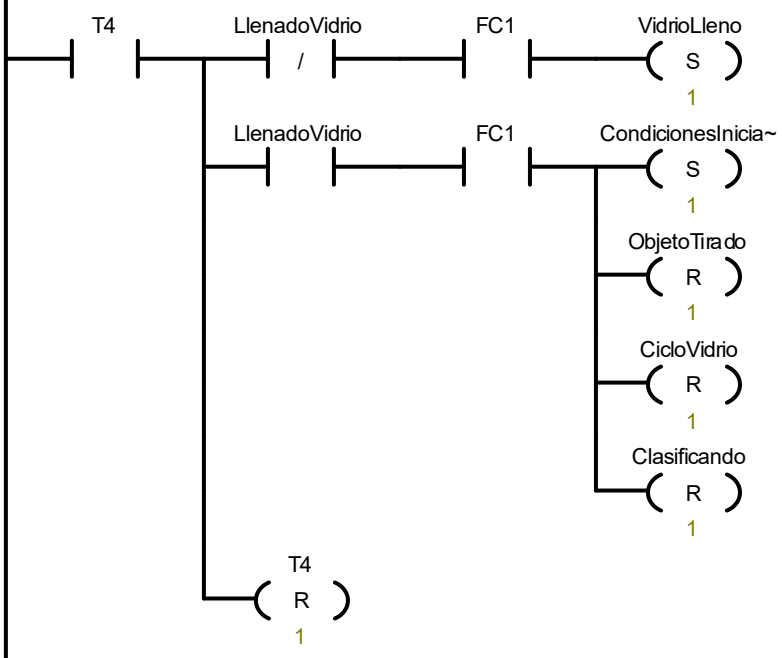
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
FC1	I1.0	Final de carrera del avance del desviador
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 11



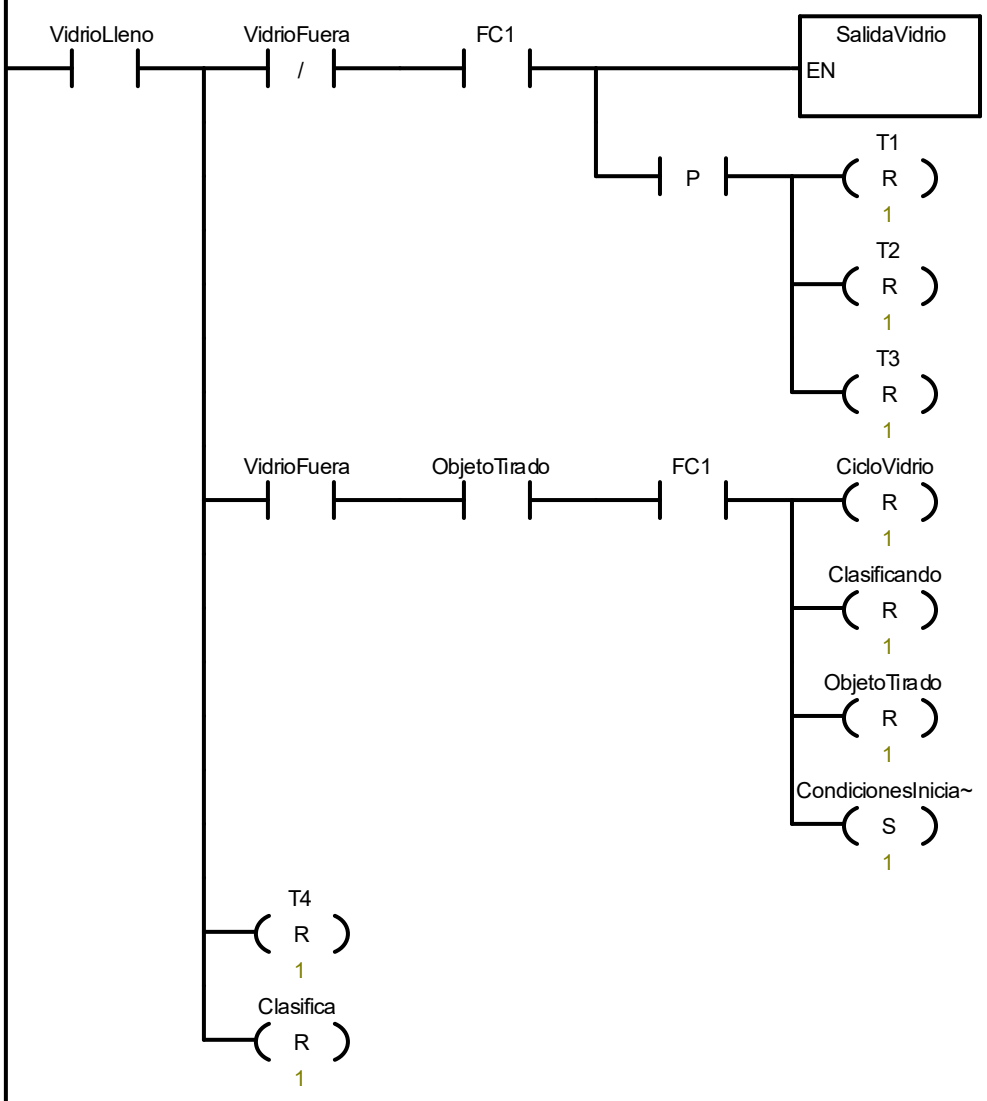
Símbolo	Dirección	Comentario
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 12



Símbolo	Dirección	Comentario
CicloVidrio	M2.5	Marca que indica que se debe de clasificar el objeto como vidrio
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
FC1	I1.0	Final de carrera del avance del desviador
LlenadoVidrio	I2.3	Sensor que detecta el llenado del cubo de vidrio
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 13

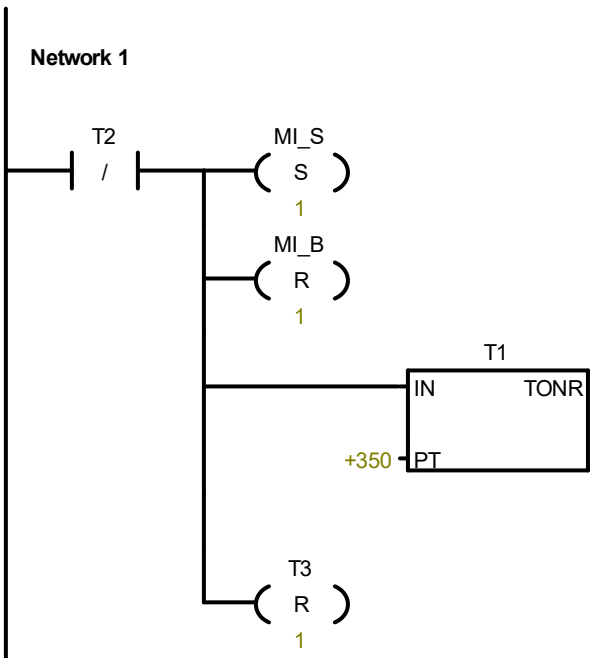


Símbolo	Dirección	Comentario
CicloVidrio	M2.5	Marca que indica que se debe de clasificar el objeto como vidrio
Clasifica	Q0.2	Señal que le indica a la Raspberry de comenzar la clasificación
Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
FC1	I1.0	Final de carrera del avance del desviador
ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
VidrioFuera	M2.2	Marca que indica que hay un cubo de cartón fuera
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Bloque: SalidaVidrio
Autor:
Fecha de creación: 06.05.2021 15:03:26
Última modificación: 11.06.2021 2:11:49

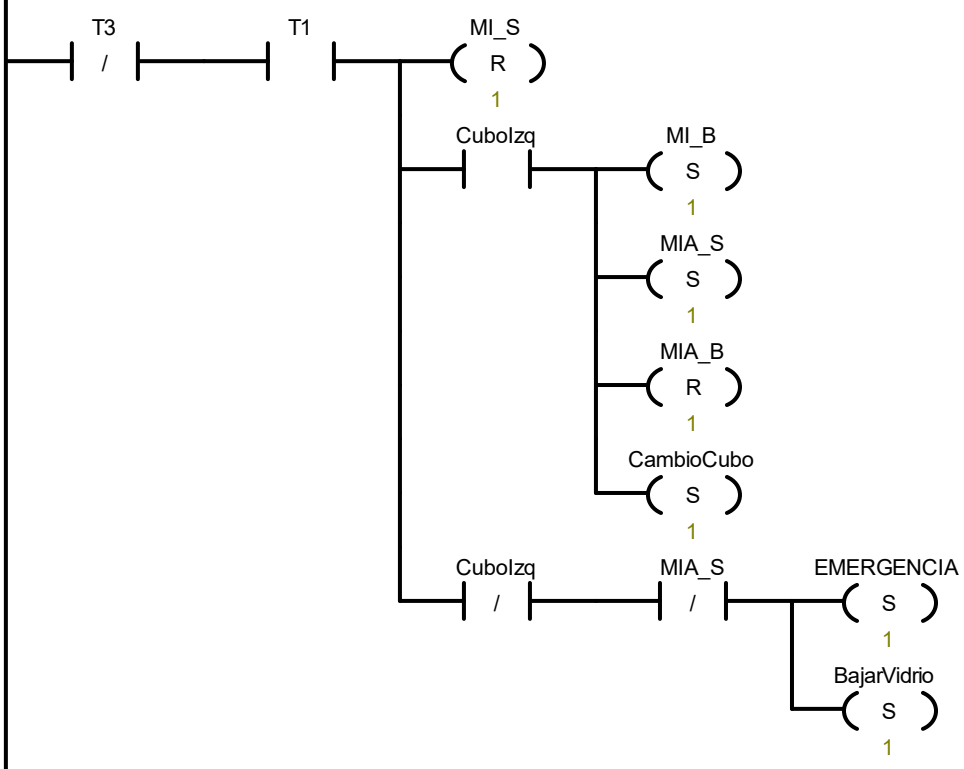
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



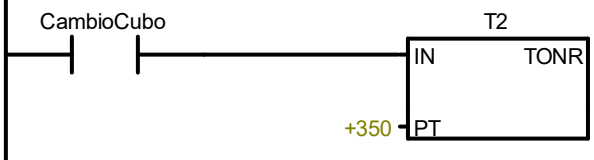
Símbolo	Dirección	Comentario
MI_B	Q1.0	Bajada DVD delantero vidrio
MI_S	Q0.7	Subida DVD delantero vidrio

Network 2



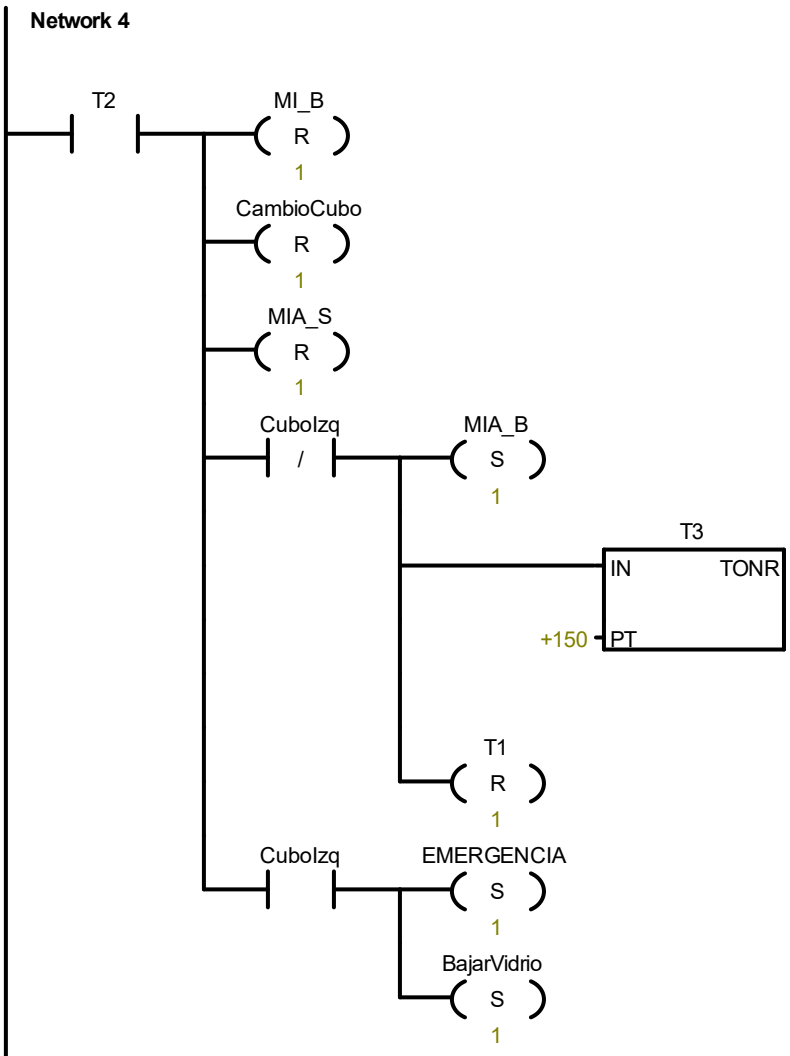
Símbolo	Dirección	Comentario
BajarVidrio	M4.2	Marca que indica que es necesario bajar los DVDs del vidrio
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboIzq	I1.4	Sensor que detecta la presencia de un cubo delantero de vidrio
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MI_B	Q1.0	Bajada DVD delantero vidrio
MI_S	Q0.7	Subida DVD delantero vidrio
MIA_B	Q0.6	Bajada DVD trasero vidrio
MIA_S	Q0.5	Subida DVD trasero vidrio

Network 3



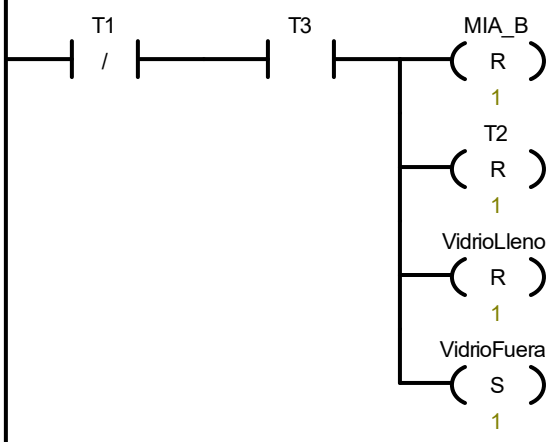
Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo

Network 4



Símbolo	Dirección	Comentario
BajarVidrio	M4.2	Marca que indica que es necesario bajar los DVDs del vidrio
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboIzq	I1.4	Sensor que detecta la presencia de un cubo delantero de vidrio
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MI_B	Q1.0	Bajada DVD delantero vidrio
MIA_B	Q0.6	Bajada DVD trasero vidrio
MIA_S	Q0.5	Subida DVD trasero vidrio

Network 5

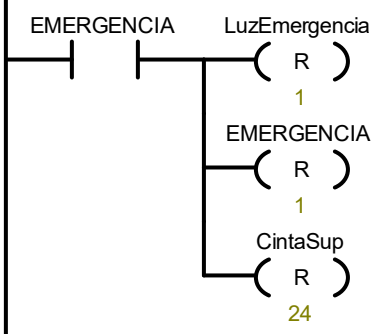


Símbolo	Dirección	Comentario
MIA_B	Q0.6	Bajada DVD trasero vidrio
VidrioFuera	M2.2	Marca que indica que hay un cubo de cartón fuera
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Bloque: PrimerCiclo
Autor:
Fecha de creación: 06.05.2021 19:57:06
Última modificación: 10.06.2021 21:09:36

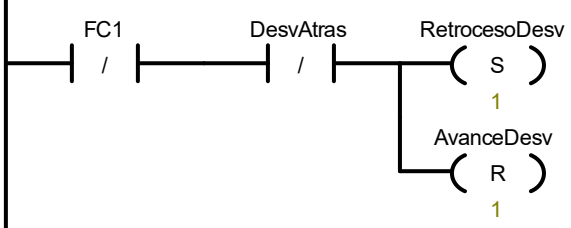
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



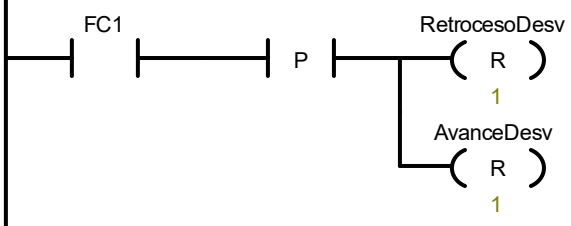
Símbolo	Dirección	Comentario
CintaSup	Q0.0	Movimiento de la cinta superior
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
LuzEmergencia	Q2.7	Piloso luminoso de emergencia

Network 2



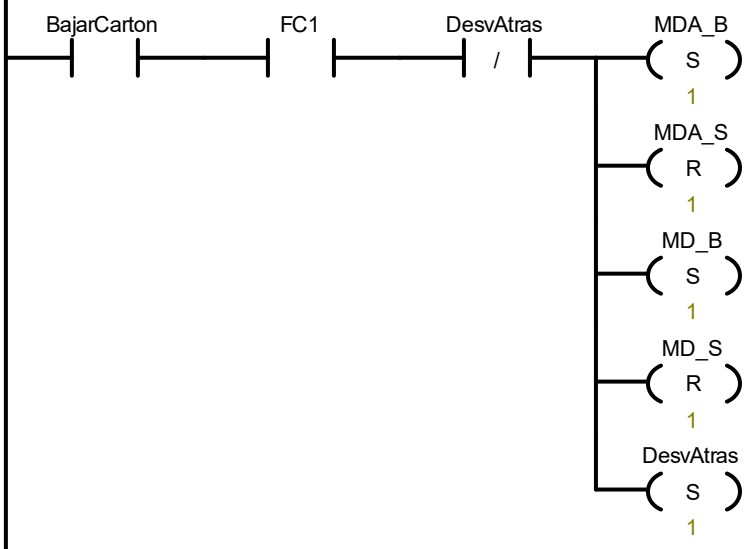
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
DesvAtras	M1.7	Marca que indica que el desviador se encuentra en su posición trasera
FC1	I1.0	Final de carrera del avance del desviador
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 3



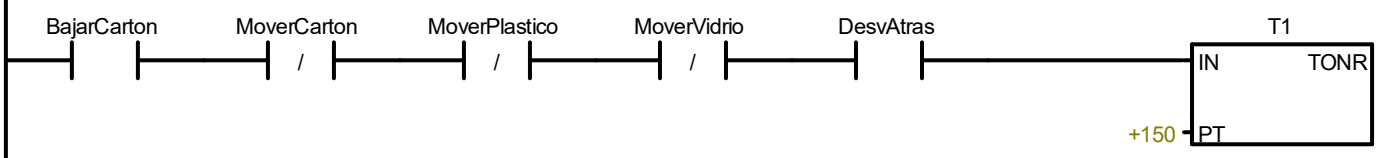
Símbolo	Dirección	Comentario
AvanceDesv	Q0.3	Movimiento de avance del desviador
FC1	I1.0	Final de carrera del avance del desviador
RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador

Network 4



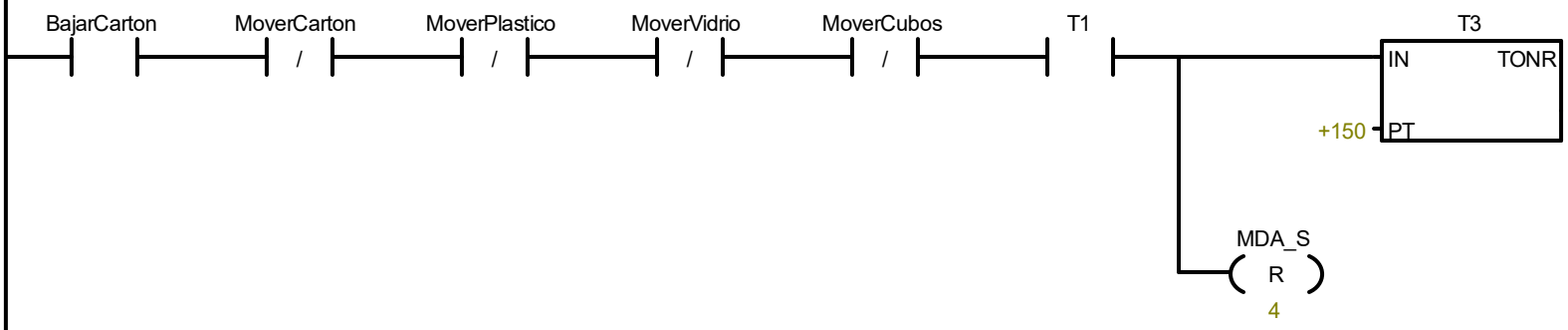
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
DesvAtras	M1.7	Marca que indica que el desviador se encuentra en su posición trasera
FC1	I1.0	Final de carrera del avance del desviador
MD_B	Q2.6	Bajada DVD delantero cartón
MD_S	Q2.5	Subida DVD delantero cartón
MDA_B	Q2.4	Bajada DVD trasero cartón
MDA_S	Q2.3	Subida DVD trasero cartón

Network 5



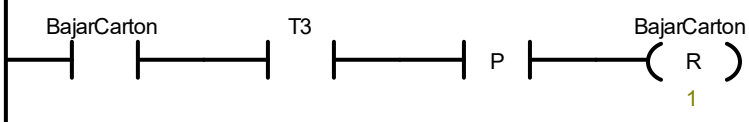
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
DesvAtras	M1.7	Marca que indica que el desviador se encuentra en su posición trasera
MoverCarton	M3.1	Marca que indica que se debe rodar el cubo del cartón en el ciclo inicial
MoverPlastico	M3.2	Marca que indica que se debe rodar el cubo del plástico en el ciclo inicial
MoverVidrio	M3.3	Marca que indica que se debe rodar el cubo del vidrio en el ciclo inicial

Network 6



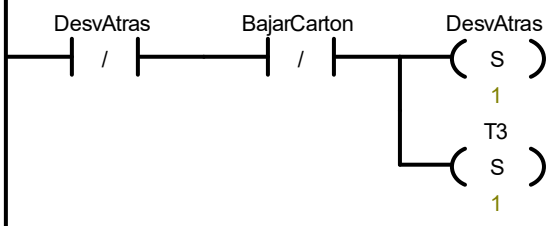
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
MDA_S	Q2.3	Subida DVD trasero cartón
MoverCarton	M3.1	Marca que indica que se debe rodar el cubo del cartón en el ciclo inicial
MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial
MoverPlastico	M3.2	Marca que indica que se debe rodar el cubo del plástico en el ciclo inicial
MoverVidrio	M3.3	Marca que indica que se debe rodar el cubo del vidrio en el ciclo inicial

Network 7



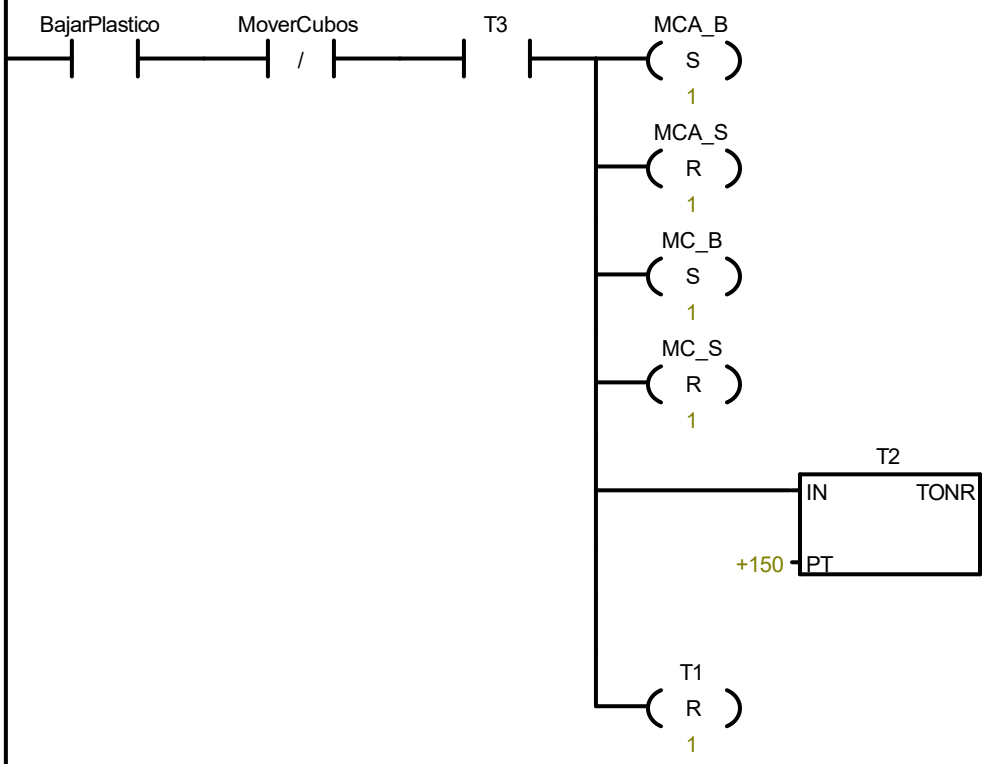
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón

Network 8



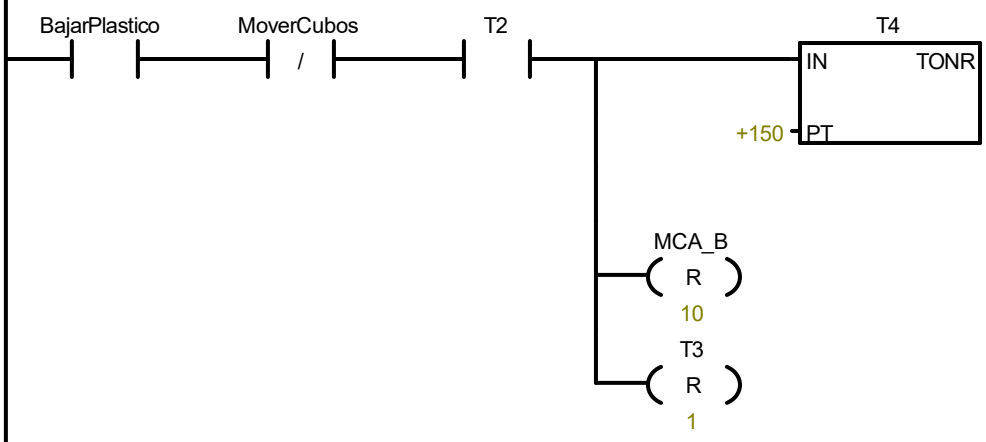
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
DesvAtras	M1.7	Marca que indica que el desviador se encuentra en su posición trasera

Network 9



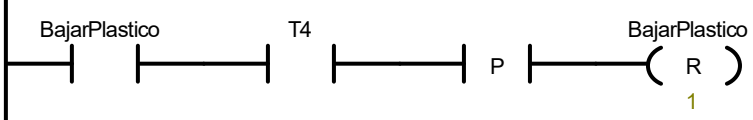
Símbolo	Dirección	Comentario
BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico
MC_B	Q2.2	Bajada DVD delantero plástico
MC_S	Q2.1	Subida DVD delantero plástico
MCA_B	Q2.0	Bajada DVD trasero plástico
MCA_S	Q1.1	Subida DVD trasero plástico
MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial

Network 10



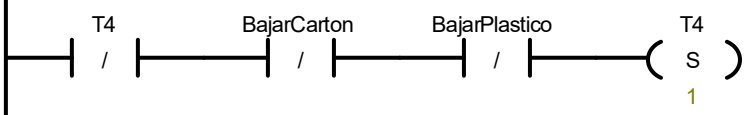
Símbolo	Dirección	Comentario
BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico
MCA_B	Q2.0	Bajada DVD trasero plástico
MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial

Network 11



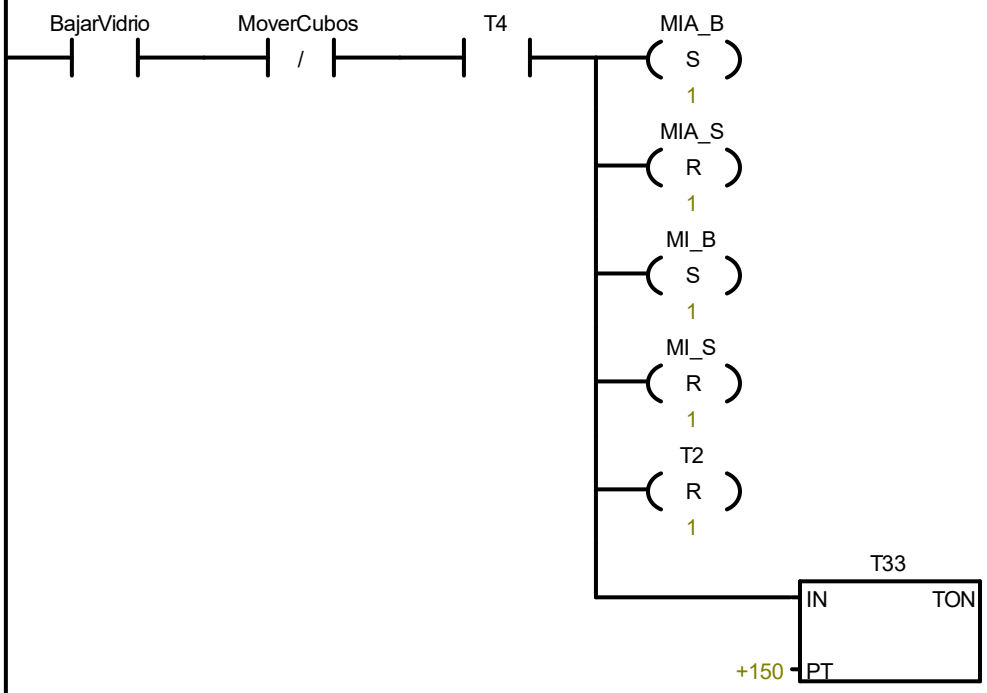
Símbolo	Dirección	Comentario
BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico

Network 12



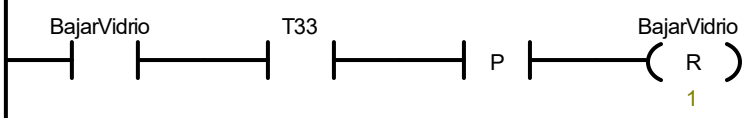
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico

Network 13



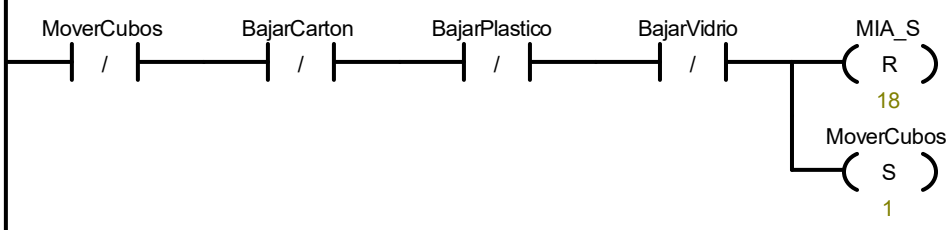
Símbolo	Dirección	Comentario
BajarVidrio	M4.2	Marca que indica que es necesario bajar los DVDs del vidrio
MI_B	Q1.0	Bajada DVD delantero vidrio
MI_S	Q0.7	Subida DVD delantero vidrio
MIA_B	Q0.6	Bajada DVD trasero vidrio
MIA_S	Q0.5	Subida DVD trasero vidrio
MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial

Network 14



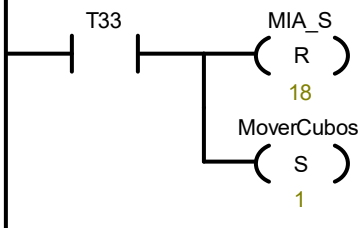
Símbolo	Dirección	Comentario
BajarVidrio	M4.2	Marca que indica que es necesario bajar los DVDs del vidrio

Network 15



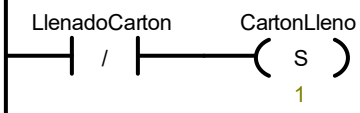
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico
BajarVidrio	M4.2	Marca que indica que es necesario bajar los DVDs del vidrio
MIA_S	Q0.5	Subida DVD trasero vidrio
MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial

Network 16



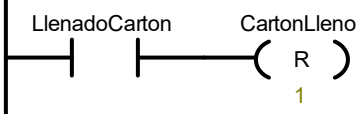
Símbolo	Dirección	Comentario
MIA_S	Q0.5	Subida DVD trasero vidrio
MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial

Network 17



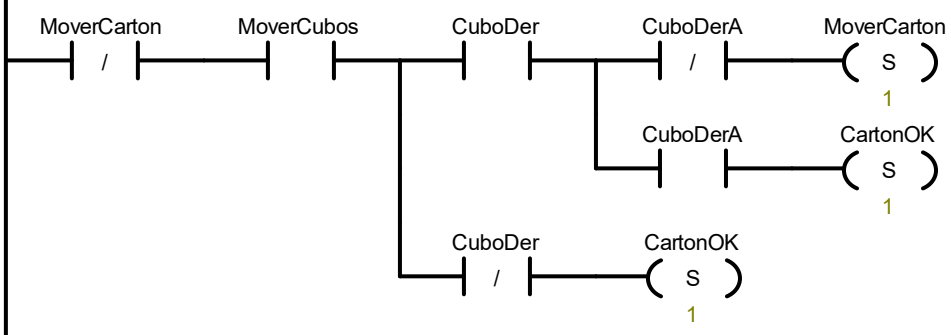
Símbolo	Dirección	Comentario
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
LlenadoCarton	I2.5	Sensor que detecta el llenado del cubo de cartón

Network 18



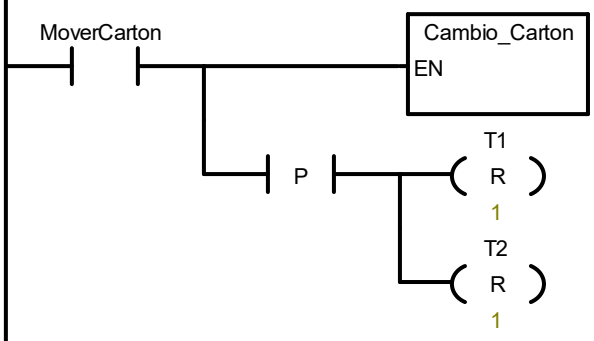
Símbolo	Dirección	Comentario
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
LlenadoCarton	I2.5	Sensor que detecta el llenado del cubo de cartón

Network 19



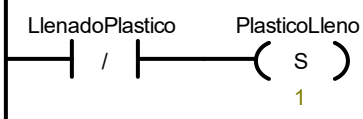
Símbolo	Dirección	Comentario
CartonOK	M3.5	Marca que indica que el cartón está listo para comenzar el ciclo
CuboDer	I2.2	Sensor que detecta la presencia de un cubo delantero de cartón
CuboDerA	I2.1	Sensor que detecta la presencia de un cubo de reserva de cartón
MoverCarton	M3.1	Marca que indica que se debe rodar el cubo del cartón en el ciclo inicial
MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial

Network 20



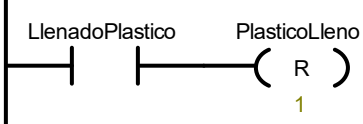
Símbolo	Dirección	Comentario
MoverCarton	M3.1	Marca que indica que se debe rodar el cubo del cartón en el ciclo inicial

Network 21



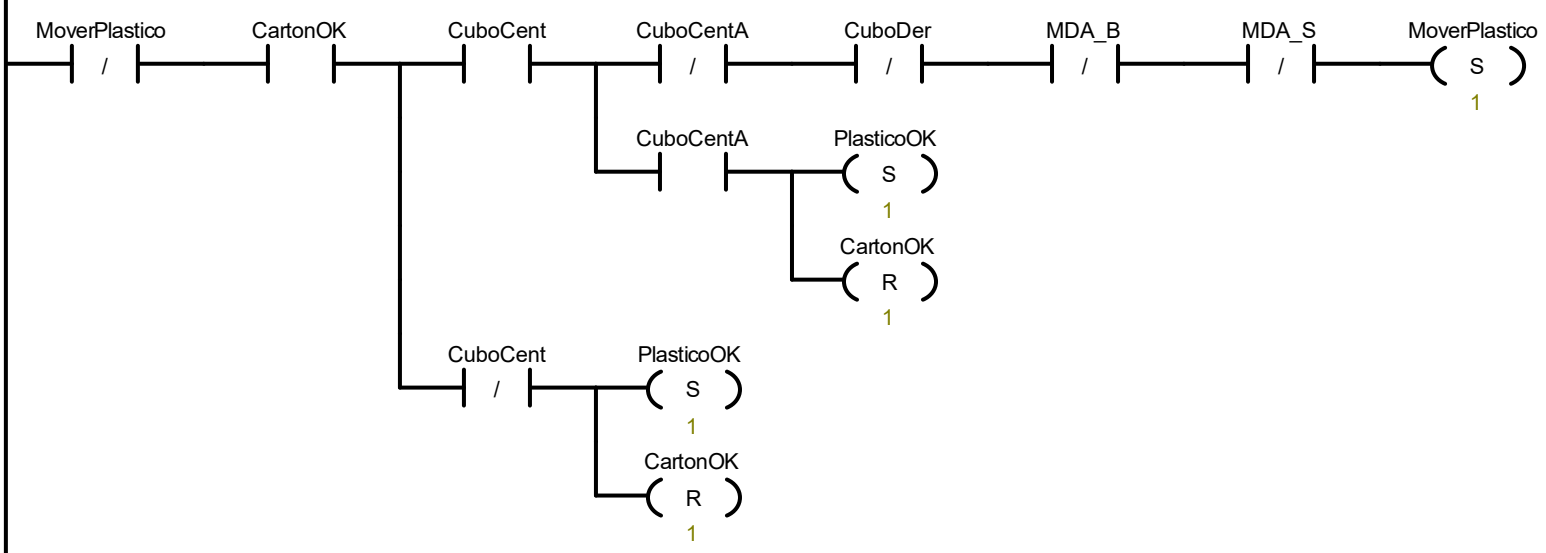
Símbolo	Dirección	Comentario
LlenadoPlastico	I2.4	Sensor que detecta el llenado del cubo de plástico
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno

Network 22



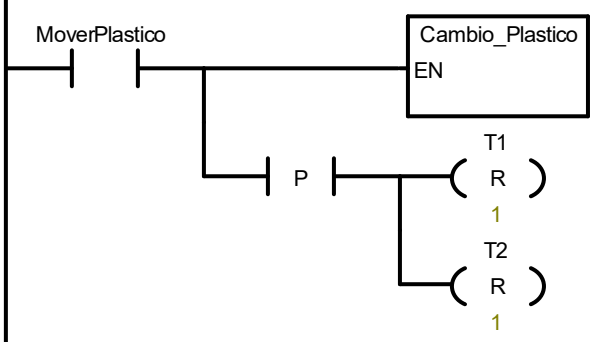
Símbolo	Dirección	Comentario
LlenadoPlastico	I2.4	Sensor que detecta el llenado del cubo de plástico
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno

Network 23



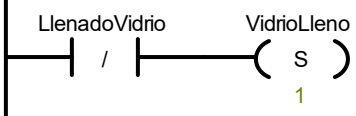
Símbolo	Dirección	Comentario
CartonOK	M3.5	Marca que indica que el cartón está listo para comenzar el ciclo
CuboCent	I2.0	Sensor que detecta la presencia de un cubo deelantero de plástico
CuboCentA	I1.5	Sensor que detecta la presencia de un cubo de reserva de plástico
CuboDer	I2.2	Sensor que detecta la presencia de un cubo delantero de cartón
MDA_B	Q2.4	Bajada DVD trasero cartón
MDA_S	Q2.3	Subida DVD trasero cartón
MoverPlastico	M3.2	Marca que indica que se debe rodar el cubo del plástico en el ciclo inicial
PlasticoOK	M3.6	Marca que indica que el plástico está listo para comenzar el ciclo

Network 24



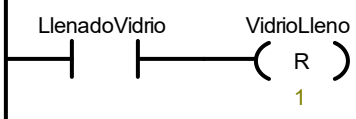
Símbolo	Dirección	Comentario
MoverPlastico	M3.2	Marca que indica que se debe rodar el cubo del plástico en el ciclo inicial

Network 25



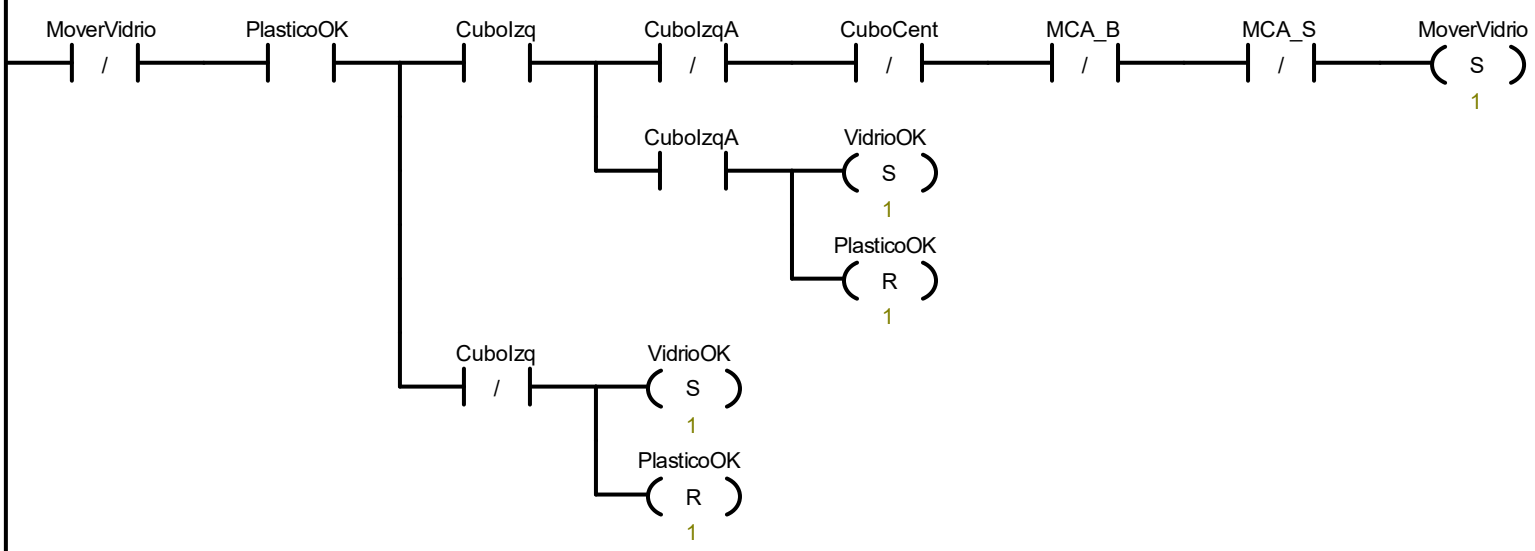
Símbolo	Dirección	Comentario
LlenadoVidrio	I2.3	Sensor que detecta el llenado del cubo de vidrio
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 26



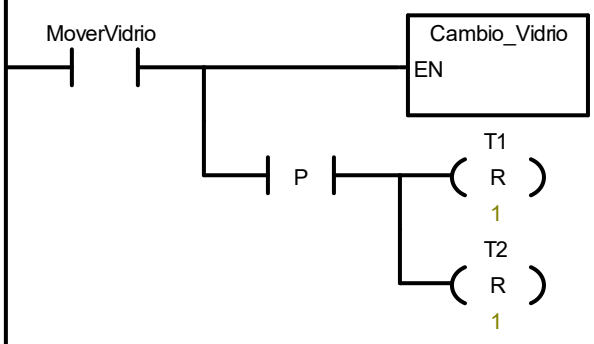
Símbolo	Dirección	Comentario
LlenadoVidrio	I2.3	Sensor que detecta el llenado del cubo de vidrio
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno

Network 27



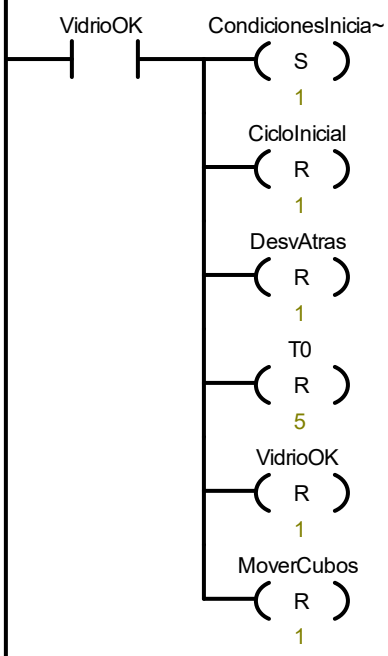
Símbolo	Dirección	Comentario
CuboCent	I2.0	Sensor que detecta la presencia de un cubo deelantero de plástico
CuboIzq	I1.4	Sensor que detecta la presencia de un cubo delantero de vidrio
CuboIzqA	I1.3	Sensor que detecta la presencia de un cubo de reserva de vidrio
MCA_B	Q2.0	Bajada DVD trasero plástico
MCA_S	Q1.1	Subida DVD trasero plástico
MoverVidrio	M3.3	Marca que indica que se debe rodar el cubo del vidrio en el ciclo inicial
PlasticoOK	M3.6	Marca que indica que el plástico está listo para comenzar el ciclo
VidrioOK	M3.7	Marca que indica que el vidrio está listo para comenzar el ciclo

Network 28



Símbolo	Dirección	Comentario
MoverVidrio	M3.3	Marca que indica que se debe rodar el cubo del vidrio en el ciclo inicial

Network 29

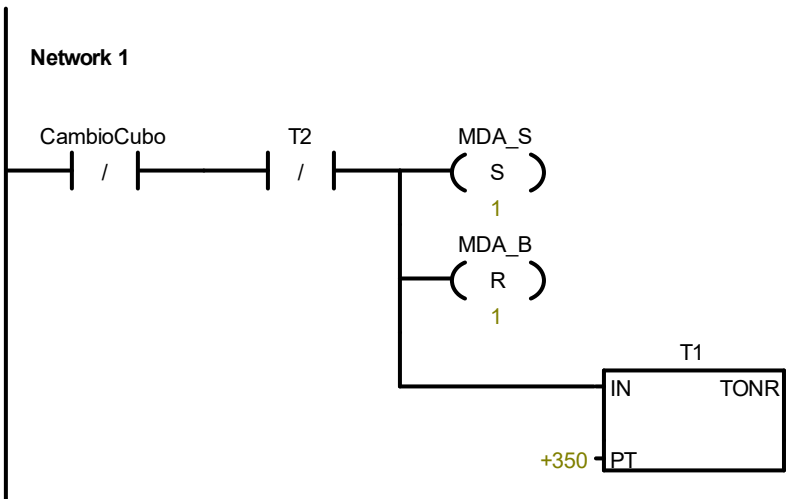


Símbolo	Dirección	Comentario
CicloInicial	M1.6	Marca que indica que debe ejecutarse el ciclo inicial
CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
DesvAtras	M1.7	Marca que indica que el desviador se encuentra en su posición trasera
MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial
VidrioOK	M3.7	Marca que indica que el vidrio está listo para comenzar el ciclo

Bloque: Cambio_Carton
Autor:
Fecha de creación: 06.05.2021 15:03:26
Última modificación: 11.06.2021 2:12:06

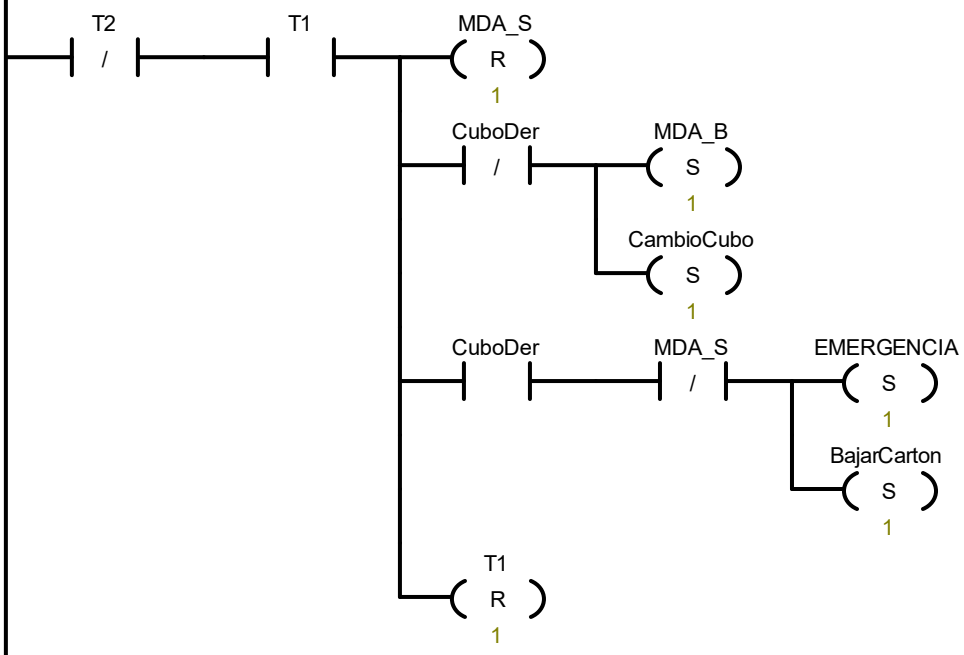
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



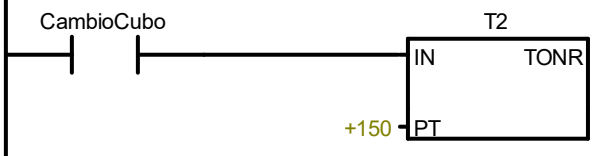
Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
MDA_B	Q2.4	Bajada DVD trasero cartón
MDA_S	Q2.3	Subida DVD trasero cartón

Network 2



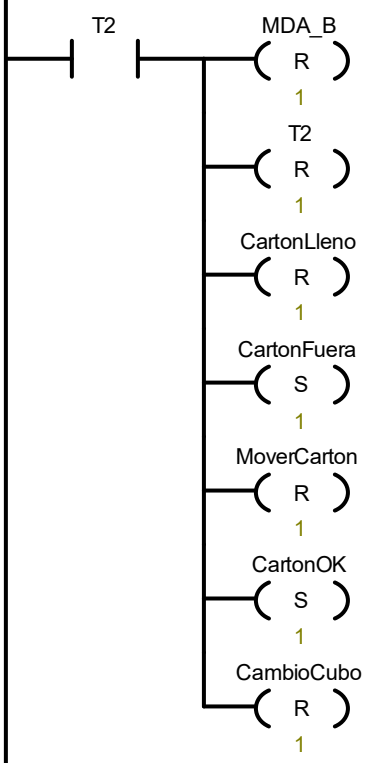
Símbolo	Dirección	Comentario
BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboDer	I2.2	Sensor que detecta la presencia de un cubo delantero de cartón
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MDA_B	Q2.4	Bajada DVD trasero cartón
MDA_S	Q2.3	Subida DVD trasero cartón

Network 3



Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo

Network 4

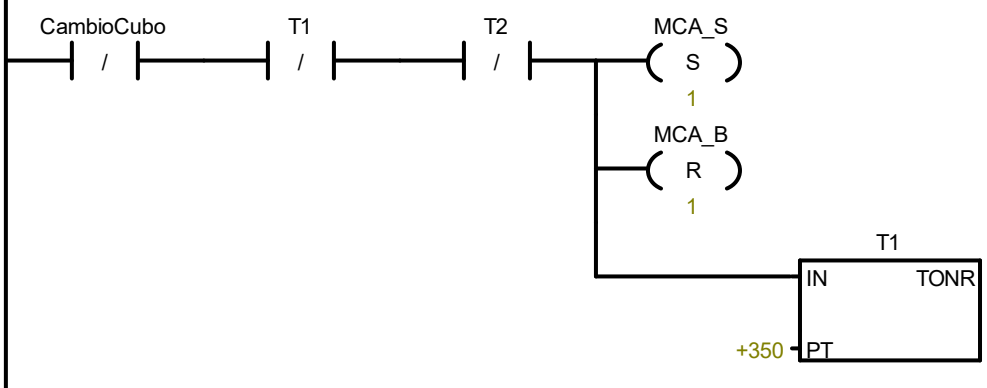


Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CartonFuera	M2.0	Marca que indica que hay un cubo de cartón fuera
CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
CartonOK	M3.5	Marca que indica que el cartón está listo para comenzar el ciclo
MDA_B	Q2.4	Bajada DVD trasero cartón
MoverCarton	M3.1	Marca que indica que se debe rodar el cubo del cartón en el ciclo inicial

Bloque: Cambio_Plastico
Autor:
Fecha de creación: 06.05.2021 15:03:26
Última modificación: 11.06.2021 2:12:18

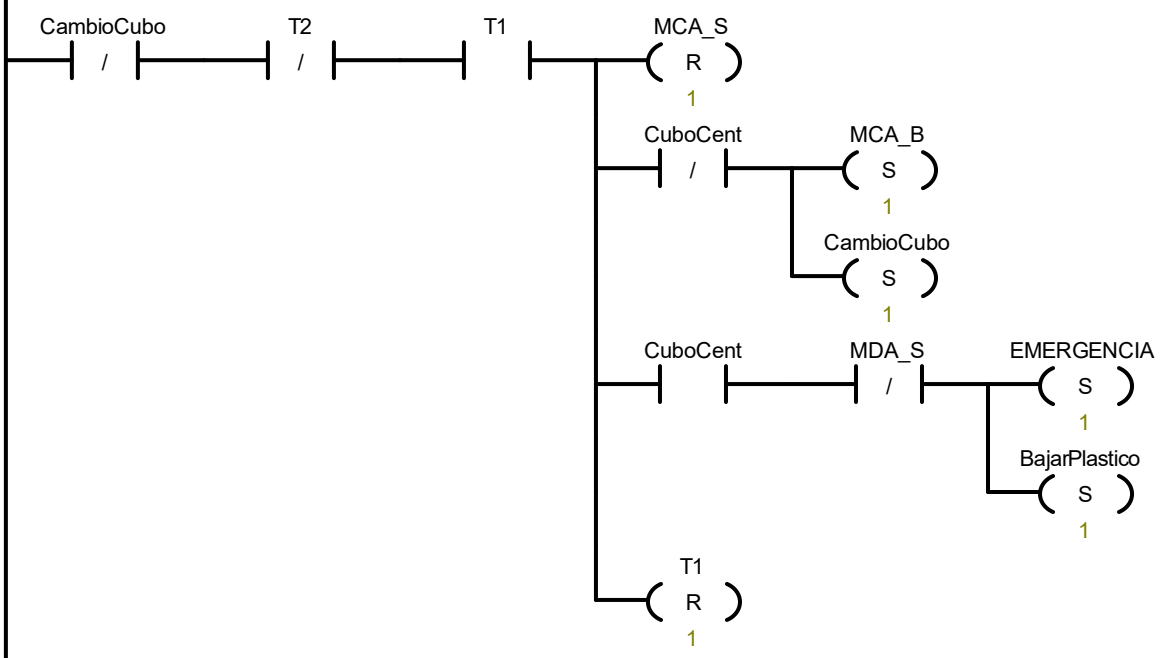
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
MCA_B	Q2.0	Bajada DVD trasero plástico
MCA_S	Q1.1	Subida DVD trasero plástico

Network 2

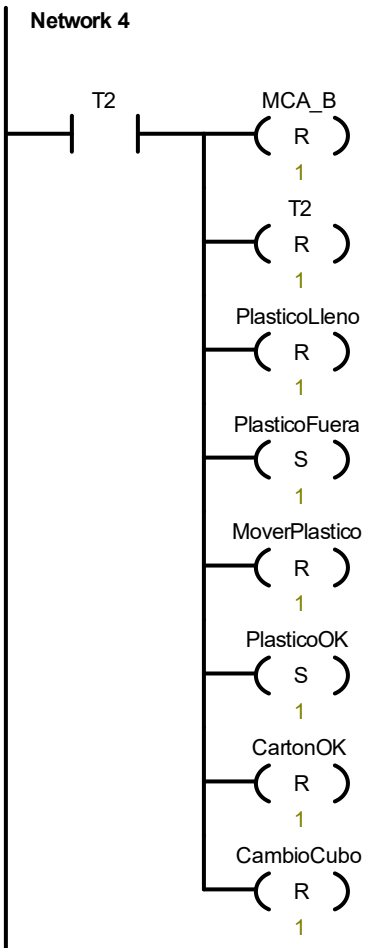


Símbolo	Dirección	Comentario
BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboCent	I2.0	Sensor que detecta la presencia de un cubo deelantero de plástico
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MCA_B	Q2.0	Bajada DVD trasero plástico
MCA_S	Q1.1	Subida DVD trasero plástico
MDA_S	Q2.3	Subida DVD trasero cartón

Network 3

Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo

Network 4

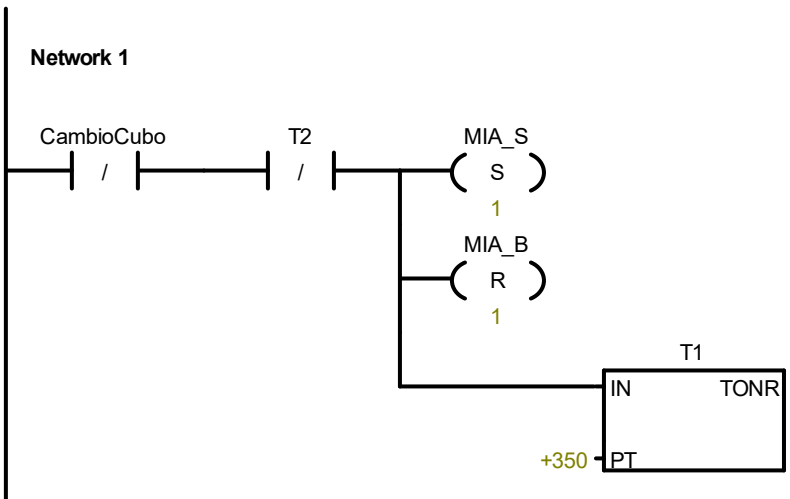


Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CartonOK	M3.5	Marca que indica que el cartón está listo para comenzar el ciclo
MCA_B	Q2.0	Bajada DVD trasero plástico
MoverPlastico	M3.2	Marca que indica que se debe rodar el cubo del plástico en el ciclo inicial
PlasticoFuera	M2.1	Marca que indica que hay un cubo de plástico fuera
PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno
PlasticoOK	M3.6	Marca que indica que el plástico está listo para comenzar el ciclo

Bloque: Cambio_Vidrio
Autor:
Fecha de creación: 06.05.2021 15:03:26
Última modificación: 11.06.2021 2:12:32

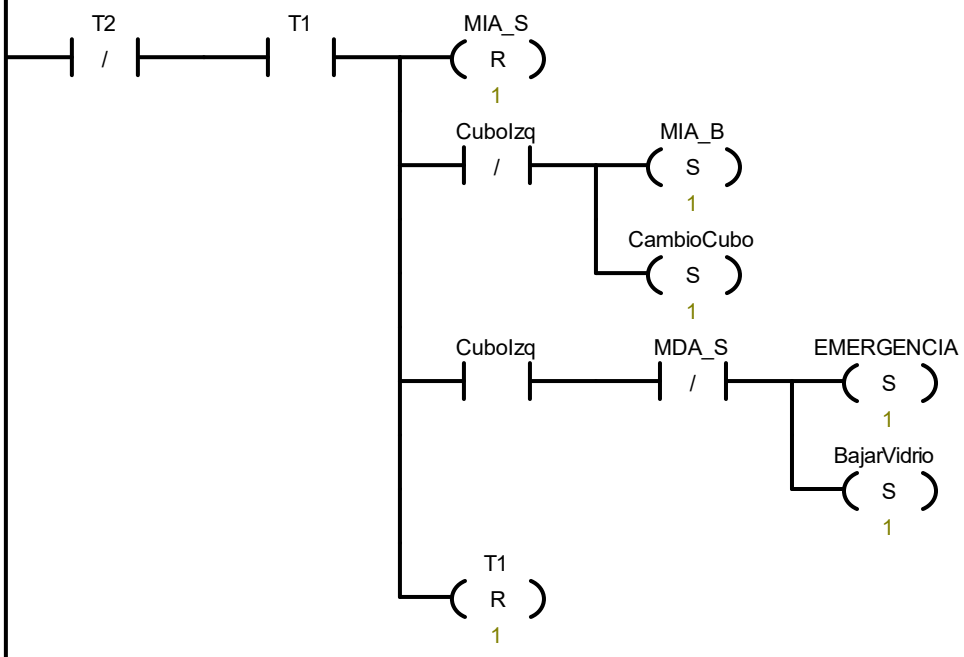
Símbolo	Tipo var.	Tipo de datos	Comentario
EN	IN	BOOL	
	IN		
	IN_OUT		
	OUT		
	TEMP		

Network 1



Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
MIA_B	Q0.6	Bajada DVD trasero vidrio
MIA_S	Q0.5	Subida DVD trasero vidrio

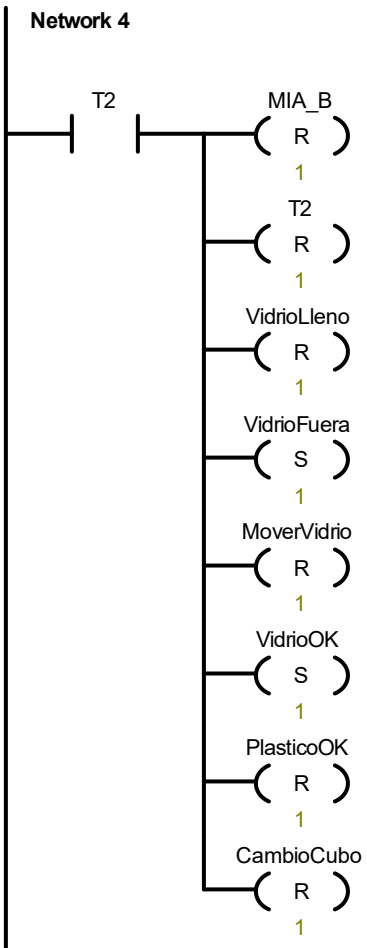
Network 2



Símbolo	Dirección	Comentario
BajarVidrio	M4.2	Marca que indica que es necesario bajar los DVDs del vidrio
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
CuboIzq	I1.4	Sensor que detecta la presencia de un cubo delantero de vidrio
EMERGENCIA	M3.0	Marca que indica un estado de emergencia
MDA_S	Q2.3	Subida DVD trasero cartón
MIA_B	Q0.6	Bajada DVD trasero vidrio
MIA_S	Q0.5	Subida DVD trasero vidrio

Network 3

Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo

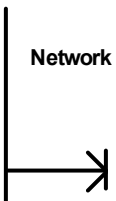


Símbolo	Dirección	Comentario
CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
MIA_B	Q0.6	Bajada DVD trasero vidrio
MoverVidrio	M3.3	Marca que indica que se debe rodar el cubo del vidrio en el ciclo inicial
PlasticoOK	M3.6	Marca que indica que el plástico está listo para comenzar el ciclo
VidrioFuera	M2.2	Marca que indica que hay un cubo de cartón fuera
VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno
VidrioOK	M3.7	Marca que indica que el vidrio está listo para comenzar el ciclo

Bloque: INT_0
Autor:
Fecha de creación: 02.05.2021 13:03:48
Última modificación: 11.06.2021 2:12:40

Símbolo	Tipo var.	Tipo de datos	Comentario
	TEMP		
	TEMP		
	TEMP		
	TEMP		

Network 1



	Símbolo	Dirección	Comentario
1	PosInicial	I0.0	Sensor que detecta la presencia de objetos en la cinta superior
2	PosRampa	I0.1	Sensor que detecta la caída de un objeto por la rampa
3	Carton	I0.2	Entrada que indica que el objeto a clasificar es de cartón
4	Plastico	I0.3	Entrada que indica que el objeto a clasificar es de plástico
5	Vidrio	I0.4	Entrada que indica que el objeto a clasificar es de vidrio
6	Inconcluyente	I0.5	Entrada que indica que se desconoce el material del objeto a clasificar
7	PosPlastico	I0.6	Sensor que indica que el objeto plástico está posicionado
8	PosVidrio	I0.7	Sensor que indica que el objeto de vidrio está posicionado
9	FC1	I1.0	Final de carrera del avance del desviador
10	FC2	I1.1	Final de carrera del retroceso del desviador
11	CaidaObj	I1.2	Sensor que detecta la caída de los objetos en los cubos
12	CuboIzqA	I1.3	Sensor que detecta la presencia de un cubo de reserva de vidrio
13	CuboIzq	I1.4	Sensor que detecta la presencia de un cubo delantero de vidrio
14	CuboCentA	I1.5	Sensor que detecta la presencia de un cubo de reserva de plástico
15	CuboCent	I2.0	Sensor que detecta la presencia de un cubo deelantero de plástico
16	CuboDerA	I2.1	Sensor que detecta la presencia de un cubo de reserva de cartón
17	CuboDer	I2.2	Sensor que detecta la presencia de un cubo delantero de cartón
18	LlenadoVidrio	I2.3	Sensor que detecta el llenado del cubo de vidrio
19	LlenadoPlastico	I2.4	Sensor que detecta el llenado del cubo de plástico
20	LlenadoCarton	I2.5	Sensor que detecta el llenado del cubo de cartón
21	Reset	I2.6	Pulsador de Rearme
22	Marcha	I2.7	Interruptor de marcha
23	CintaSup	Q0.0	Movimiento de la cinta superior
24	CintaInf	Q0.1	Movimiento de la cinta inferior
25	Clasifica	Q0.2	Señal que le indica a la Raspberry de comenzar la clasificación
26	AvanceDesv	Q0.3	Movimiento de avance del desviador
27	RetrocesoDesv	Q0.4	Movimiento de retroceso del desviador
28	MIA_S	Q0.5	Subida DVD trasero vidrio
29	MIA_B	Q0.6	Bajada DVD trasero vidrio
30	MI_S	Q0.7	Subida DVD delantero vidrio
31	MI_B	Q1.0	Bajada DVD delantero vidrio
32	MCA_S	Q1.1	Subida DVD trasero plástico
33	MCA_B	Q2.0	Bajada DVD trasero plástico
34	MC_S	Q2.1	Subida DVD delantero plástico
35	MC_B	Q2.2	Bajada DVD delantero plástico
36	MDA_S	Q2.3	Subida DVD trasero cartón
37	MDA_B	Q2.4	Bajada DVD trasero cartón
38	MD_S	Q2.5	Subida DVD delantero cartón
39	MD_B	Q2.6	Bajada DVD delantero cartón
40	LuzEmergencia	Q2.7	Piloso luminoso de emergencia
41	CondicionesIniciales	M0.0	Marca que indica que el sistema se encuentra en condiciones iniciales
42	CargaObjeto	M0.1	Marca que indica que un objeto ha sido cargado
43	CaidaRampa	M0.2	Marca que indica que un objeto ha caído por la rampa
44	Clasificando	M0.3	Marca que indica que el sistema se encuentra clasificando
45	TirarPlastico	M0.4	Marca que indica que se debe posicionar plástico
46	TirarVidrio	M0.5	Marca que indica que se debe posicionar vidrio
47	ObjetoPosicionado	M0.6	Marca que indica que el objeto ha sido posicionado
48	FlancoPosicion	M0.7	Marca que indica que ha habido un flanco del sensor de posicionamiento
49	DesvAlante	M1.0	Marca que indica que el desviador está en su posición delantera
50	ObjetoTirado	M1.1	Marca que indica que el objeto ha caído correctamente
51	CartonLleno	M1.2	Marca que indica que el cubo de cartón está lleno
52	PlasticoLleno	M1.3	Marca que indica que el cubo de plástico está lleno
53	VidrioLleno	M1.4	Marca que indica que el cubo de vidrio está lleno
54	CuboSaliendo	M1.5	FINALMENTE NO IMPLEMENTADA
55	CicloInicial	M1.6	Marca que indica que debe ejecutarse el ciclo inicial
56	DesvAtras	M1.7	Marca que indica que el desviador se encuentra en su posición trasera
57	CartonFuera	M2.0	Marca que indica que hay un cubo de cartón fuera
58	PlasticoFuera	M2.1	Marca que indica que hay un cubo de plástico fuera
59	VidrioFuera	M2.2	Marca que indica que hay un cubo de cartón fuera
60	CicloCarton	M2.3	Marca que indica que se debe de clasificar el objeto como cartón
61	CicloPlastico	M2.4	Marca que indica que se debe de clasificar el objeto como plástico
62	CicloVidrio	M2.5	Marca que indica que se debe de clasificar el objeto como vidrio
63	CambioCubo	M2.6	Marca que indica que se está realizando un cambio de cubo
64	TiempoCaida	M2.7	Marca que espera para que se estabilice el objeto antes de capturar la imagen
65	EMERGENCIA	M3.0	Marca que indica un estado de emergencia
66	MoverCarton	M3.1	Marca que indica que se debe rodar el cubo del cartón en el ciclo inicial
67	MoverPlastico	M3.2	Marca que indica que se debe rodar el cubo del plástico en el ciclo inicial
68	MoverVidrio	M3.3	Marca que indica que se debe rodar el cubo del vidrio en el ciclo inicial
69	MoverCubos	M3.4	Marca que indica que se deben mover algunos cubos en el ciclo inicial
70	CartonOK	M3.5	Marca que indica que el cartón está listo para comenzar el ciclo
71	PlasticoOK	M3.6	Marca que indica que el plástico está listo para comenzar el ciclo
72	VidrioOK	M3.7	Marca que indica que el vidrio está listo para comenzar el ciclo
73	BajarCarton	M4.0	Marca que indica que es necesario bajar los DVDs del cartón
74	BajarPlastico	M4.1	Marca que indica que es necesario bajar los DVDs del plástico
75	BajarVidrio	M4.2	Marca que indica que es necesario bajar los DVDs del vidrio

	Símbolo	Dirección	Comentario
1	SBR_Carton	SBR0	
2	SalidaCarton	SBR1	
3	SBR_Plastico	SBR2	
4	SalidaPlastico	SBR3	
5	SBR_Vidrio	SBR4	
6	SalidaVidrio	SBR5	
7	PrimerCiclo	SBR6	
8	Cambio_Carton	SBR7	
9	Cambio_Plastico	SBR8	
10	Cambio_Vidrio	SBR9	
11	INT_0	INT0	
12	PRINCIPAL	OB1	

ANEXO VI.

ENTRENAMIENTO DE LA RED NEURONAL

```
# -*- coding: utf-8 -*-
"""Untitled2.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/12Y_3vponl-D8ucysO9jVYJwhyEeGxUY0
"""

# Commented out IPython magic to ensure Python compatibility.
# %tensorflow_version 1.x

from tensorflow.keras.applications.xception import Xception
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.vgg19 import VGG19
from tensorflow.keras.applications.resnet50 import ResNet50
from keras.applications.resnet_v2 import ResNet50V2
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.applications.inception_resnet_v2 import InceptionResNetV2
from tensorflow.keras.applications.mobilenet import MobileNet
from tensorflow.keras.applications.densenet import DenseNet121
from tensorflow.keras.applications.nasnet import NASNetMobile
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2

from tensorflow.keras import backend as K  # para el preprocesamiento

from tensorflow.keras.layers import Dense, Input, GlobalAveragePooling2D
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import RMSprop

from google.colab import drive

from math import ceil

import matplotlib.pyplot as plt

import datetime

import numpy as np

import os


drive.mount('/content/gdrive')

# Copiamos y descomprimimos el conjunto de datos en la máquina virtual
#### CAMBIAR AQUÍ: LA RUTA DEL ARCHIVO DE IMÁGENES
!cp "/content/gdrive/My Drive/TFGBasura/scripts/Imagenes/Datos.tar.gz" /tmp
!tar xzvf /tmp/Datos.tar.gz
!ls "/content/Datos"

# Ubicación de los splits, del conjunto de test y de los resultados.
BASE = '/content/Datos/folds'
ALL_TRAIN_SET_PATH = '/content/Datos/training_set'
TEST_SET_PATH = '/content/Datos/test_set'

#### CAMBIAR AQUÍ: LA RUTA DEL DIRECTORIO DE RESULTADOS
BASE_RESULTS = '/content/gdrive/My Drive/TFGBasura/resultado'
```

```
#### CAMBIAR AQUÍ: DESCOMENTAR LA RED QUE QUERAMOS USAR Y COMENTAR EL RESTO
# Nombre de la red que vamos a usar para el entrenamiento. En función del nombre
# se creará una subcarpeta dentro de la carpeta de resultados anterior.
#RED = 'xception'
RED = 'vgg16'
#RED = 'vgg19'
#RED = 'resnet50'
#RED = 'resnet50v2' #este da fallo siempre
#RED = 'inceptionv3'
#RED = 'inceptionresnetv2'  #este dura más de 10 horas y se desconecta
#RED = 'mobilenet'
#RED = 'densenet'
#RED = 'nasnetmobile'
#RED = 'mobilenetv2'

# funciones para el preprocesamiento
def preprocess_input1(x): # For VGG16, VGG19 and ResNet50
  data_format = K.image_data_format()
  assert data_format in {'channels_last', 'channels_first'}
  if data_format == 'channels_first':
      # 'RGB'->'BGR'
      x = x[::-1, :, :]
      # Zero-center by mean pixel

      x[0, :, :] -= 103.939
      x[1, :, :] -= 116.779
      x[2, :, :] -= 123.68
  else:
      # 'RGB'->'BGR'
      x = x[:, :, ::-1]
      # Zero-center by mean pixel
      x[:, :, 0] -= 103.939
      x[:, :, 1] -= 116.779
      x[:, :, 2] -= 123.68
  return x

def preprocess_input2(x): # For InceptionV3, Xception
  x /= 255.
  x -= 0.5
  x *= 2.
  return x


if RED == 'vgg16' or RED == 'vgg19' or RED == 'resnet50':
    print("Configurando función de preprocesamiento para VGG16, VGG19 o ResNet50")
    preprocess_func = preprocess_input1
elif RED == 'xception' or RED == 'inceptionv3':
    print("Configurando función de preprocesamiento para Xception o InceptionV3")
    preprocess_func = preprocess_input2
else:
    raise Exception("Red no compatible con función de preprocesamiento")




def plot_history(histories, base_results, network, fold, additional_suffix_str, key='acc'):
    plt.figure(figsize=(16,10))

    val_key = 'val_' + key
    for name, history in histories:
        if val_key in history.history:
```

```
            val = plt.plot(history.epoch, history.history['val_'+key],
                '--', label=name.title()+' Val')
            plt.plot(history.epoch, history.history[key], color=val[0].get_color(),
                label=name.title()+' Train')
        else:
            plt.plot(history.epoch, history.history[key], color='b',
                label=name.title()+' Train')

    plt.xlabel('Epochs')
    plt.ylabel(key.replace('_',' ').title())
    plt.legend()

    plt.xlim([0,max(history.epoch)])
    plt.savefig(base_results + '/' + network + "/" + 'fold_' + str(fold) + '_' + additional_suffix_str +
'_acc_history.png', dpi=None, facecolor='w', edgecolor='w',
        orientation='portrait', papertype=None, format=None,
        transparent=False, bbox_inches='tight', pad_inches=0.5,
        frameon=None)


def get_base_model(network, input_size):
    input_tensor = Input(shape=(input_size, input_size, 3))

    if network == 'xception':
        base_model = Xception(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'vgg16':
        base_model = VGG16(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'vgg19':
        base_model = VGG19(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'resnet50':
        base_model = ResNet50(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'resnet50v2':
        base_model = ResNet50V2(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'inceptionv3':
        base_model = InceptionV3(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'inceptionresnetv2':
        base_model = InceptionResNetV2(input_tensor=input_tensor, weights='imagenet',
include_top=False)
    elif network == 'mobilenet':
        base_model = MobileNet(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'densenet':
        base_model = DenseNet121(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'nasnetmobile':
        base_model = NASNetMobile(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'mobilenetv2':
        base_model = MobileNetV2(input_tensor=input_tensor, weights='imagenet', include_top=False)
    else:
        print('Network unknown')
        return ''

    return base_model


def model_fine_tuning(network, base_model, model, train_batches, val_batches, EPOCHS_FINE_TUNING,
STEPS_PER_EPOCH, VALIDATION_STEPS, base_results, fold):
    #model.summary()
    print('This is the number of trainable weights before freezing the conv base:',
len(model.trainable_weights))
    base_model.trainable = False
    print('This is the number of trainable weights after freezing the conv base:',
len(model.trainable_weights))
```

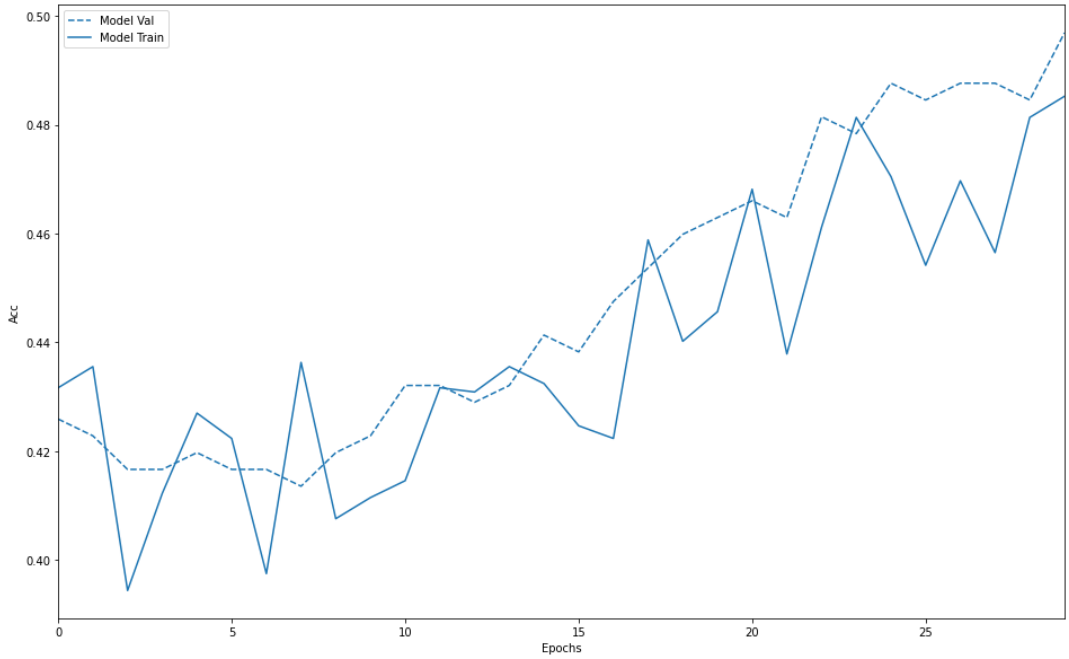
```
    # primero hacemos un ajuste fino de los pesos de la última capa
    model.compile(RMSprop(lr=2e-5), loss='categorical_crossentropy', metrics=['accuracy'])


    # Fijar steps_per_epoch y validation_steps

    print('Epochs: ', EPOCHS_FINE_TUNING)
    print('Steps per epoch: ', STEPS_PER_EPOCH)
    print('Validation steps: ', VALIDATION_STEPS)
    print('Fold ' + str(fold) + ' - Starting fine-tuning...')
    a = datetime.datetime.now()
    if val_batches is None:
        model_history = model.fit_generator(train_batches, workers=4, use_multiprocessing=True,
steps_per_epoch=STEPS_PER_EPOCH, epochs=EPOCHS_FINE_TUNING, verbose=2)
    else:
        model_history = model.fit_generator(train_batches, workers=4, use_multiprocessing=True,
steps_per_epoch=STEPS_PER_EPOCH, epochs=EPOCHS_FINE_TUNING, validation_data=val_batches,
validation_steps=VALIDATION_STEPS, verbose=2)
        #model_history = model.fit_generator(train_batches, epochs=EPOCHS_FINE_TUNING,
validation_data=val_batches, verbose=2)
    b = datetime.datetime.now()
    delta = b - a
    print('Elapsed time for fine-tuning:')
    print(delta)
    plot_history([('model', model_history)], base_results, network, fold, 'fine_tuning')
#    plot_history([('model', model_history)], base_results, network, fold, 'fine_tuning', key='loss')
    return model_history, base_model, model


def build_model(network, input_size):
    # Cambiar el base_model en función de la red que vayamos a utilizar
    base_model = get_base_model(network, input_size)
    if base_model == '':
        print('Network unknown')
        return

    # Añadimos sólo GlobalAveragePooling2D,
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    predictions = Dense(3, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    return base_model, model


def model_deep_tuning(network, input_size, base_model, model, train_batches, val_batches, test_path,
EPOCHS_DEEP_TUNING, STEPS_PER_EPOCH, VALIDATION_STEPS, base_results, fold, results_file):
    # luego hacemos un ajuste de todas las capas: deep-tuning
    print('This is the number of trainable weights before unfreezing the conv base:',
len(model.trainable_weights))
    base_model.trainable = True
    print('This is the number of trainable weights after unfreezing the conv base:',
len(model.trainable_weights))
    model.compile(RMSprop(lr=1e-5), loss='categorical_crossentropy', metrics=['accuracy'])
    # Fijar steps_per_epoch y validation_steps

    print('Fold ' + str(fold) + ' - Starting deep-tuning...')
    a = datetime.datetime.now()
    if val_batches is None:
        model_history = model.fit_generator(train_batches, workers=4, use_multiprocessing=True,
```

```
steps_per_epoch=STEPS_PER_EPOCH, epochs=EPOCHS_DEEP_TUNING, verbose=2)
        model.save_weights(base_results + '/' + network + "/" + 'fold_' + str(fold) + "_w_net.h5")
        test_batches =
ImageDataGenerator(preprocessing_function=preprocess_func).flow_from_directory(test_path,
target_size=(input_size, input_size), batch_size=BATCH_SIZE, classes=['Carton', 'Plastico','Vidrio'])
        loss, acc = model.evaluate_generator(test_batches)
        results_file.write('Fold ' + str(fold) + " - Test Acc: " + str(acc) + "\n")
    else:
        model_history = model.fit_generator(train_batches, workers=4, use_multiprocessing=True,
steps_per_epoch=STEPS_PER_EPOCH, epochs=EPOCHS_DEEP_TUNING, validation_data=val_batches,
validation_steps=VALIDATION_STEPS, verbose=2)
        loss, acc = model.evaluate_generator(val_batches)
        results_file.write('Fold ' + str(fold) + " - Val Acc:  " + str(acc) + "\n")
    #model_history = model.fit_generator(train_batches, epochs=EPOCHS_DEEP_TUNING,
validation_data=val_batches, verbose=2)
    b = datetime.datetime.now()
    delta = b - a
    print('Elapsed time for deep-tuning:')
    print(delta)
    plot_history([('model', model_history)], base_results, network, fold, 'deep_tuning')
    # TODO: guardamos los resultados de cada época (accuracy, AUC...?)

    results_file.close()

    return model_history




# Beginning of the trainings

results_path = BASE_RESULTS + '/' + RED
os.makedirs(results_path, exist_ok=True)

INPUT_SIZE_BY_NETWORK = {
    'xception': 224,
    'vgg16': 224,
    'vgg19': 224,
    'resnet50': 224,
    'resnet50v2': 224,
    'inceptionv3': 224,
    'inceptionresnetv2': 224,
    'mobilenet': 224,
    'densenet': 224,
    'nasnetmobile': 224,
    'mobilenetv2': 224
}


#### CAMBIAR AQUÍ: muestras totales de train y val sumando todas las clases - para cada fold
# Number of training and validation samples per fold
train_samples_num = [1288, 1289, 1290, 1290, 1291]
val_samples_num = [324, 323, 322, 322, 321]
NUM_FOLDS = 5
# Network parameters
input_size = INPUT_SIZE_BY_NETWORK[RED]
BATCH_SIZE = 32
EPOCHS_FINE_TUNING = 30
all_acc_histories = []
# For loop repeating the training for each fold: fine-tuning freezing the base model
for fold in range(1, NUM_FOLDS+1):
```

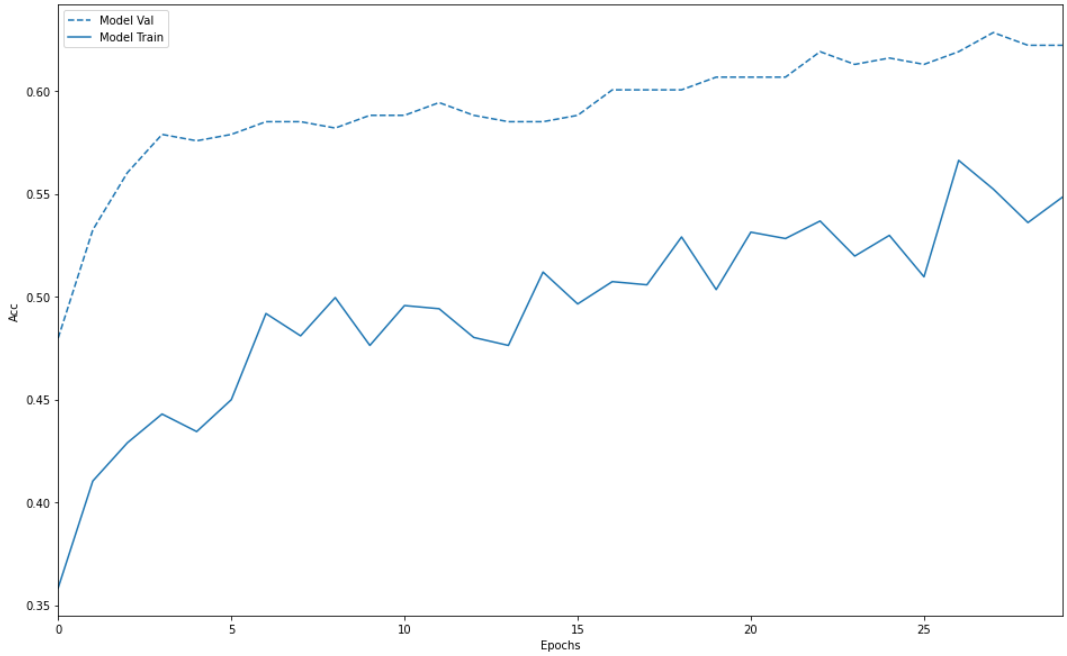
```
    # Carpetas que contiene el conjunto de entrenamiento y el conjunto de validación
    fold_path = BASE + '/fold_' + str(fold)
    train_path = fold_path + '/training_set'
    val_path = fold_path + '/validation_set'

    if fold == 1:
      results_file = open(results_path + "/info.txt", "w")
    else:
      results_file = open(results_path + "/info.txt", "a")

    STEPS_PER_EPOCH = ceil(train_samples_num[fold-1] / BATCH_SIZE)
    VALIDATION_STEPS = ceil(val_samples_num[fold-1] / BATCH_SIZE)
    #train_test_model(RED, train_path, val_path, TEST_SET_PATH, train_samples_num[fold-1],
val_samples_num[fold-1], BASE_RESULTS, fold, results_file)
    base_model, model = build_model(RED, input_size)
    train_batches = ImageDataGenerator(preprocessing_function=preprocess_func, rotation_range=30,
vertical_flip=True, horizontal_flip=True, zoom_range=[0.8,1.2]).flow_from_directory(train_path,
target_size=(input_size, input_size), batch_size=BATCH_SIZE, classes=['Carton', 'Plastico','Vidrio'])
    val_batches =
ImageDataGenerator(preprocessing_function=preprocess_func).flow_from_directory(val_path,
target_size=(input_size, input_size), batch_size=BATCH_SIZE, classes=['Carton', 'Plastico','Vidrio'])
    model_history, base_model, model = model_fine_tuning(RED, base_model, model, train_batches,
val_batches, EPOCHS_FINE_TUNING, STEPS_PER_EPOCH, VALIDATION_STEPS, BASE_RESULTS,
str(fold) + '_pre')
    all_acc_histories.append(model_history.history['val_acc'])


print(all_acc_histories)
average_acc_history = [np.mean([x[i] for x in all_acc_histories]) for i in range(EPOCHS_FINE_TUNING)]
print(average_acc_history)
max_acc_epoch = average_acc_history.index(max(average_acc_history)) + 1
print('Best epoch for fine-tuning: ' + str(max_acc_epoch))

# plt.plot(range(0, len(average_acc_history)), average_acc_history)
# plt.xlabel('Epochs')
# plt.ylabel('Validation Acc')
# plt.show()

EPOCHS_DEEP_TUNING = 50  # número máximo de épocas usadas en ACRIMA (ellos sugieren 200 como
mejor...)
all_acc_histories_deep = []
# For loop repeating the training for each fold: deep-tuning (train all layers)
for fold in range(1, NUM_FOLDS+1):
    # Carpetas que contiene el conjunto de entrenamiento y el conjunto de validación de un pliegue.
    fold_path = BASE + '/fold_' + str(fold)
    train_path = fold_path + '/training_set'
    val_path = fold_path + '/validation_set'

    if fold == 1:
      results_file = open(results_path + "/info.txt", "w")
    else:
      results_file = open(results_path + "/info.txt", "a")

    STEPS_PER_EPOCH = ceil(train_samples_num[fold-1] / BATCH_SIZE)
    VALIDATION_STEPS = ceil(val_samples_num[fold-1] / BATCH_SIZE)
    #train_test_model(RED, train_path, val_path, TEST_SET_PATH, train_samples_num[fold-1],
val_samples_num[fold-1], BASE_RESULTS, fold, results_file)
    base_model, model = build_model(RED, input_size)
    train_batches = ImageDataGenerator(preprocessing_function=preprocess_func, rotation_range=30,
vertical_flip=True, horizontal_flip=True, zoom_range=[0.8,1.2]).flow_from_directory(train_path,
target_size=(input_size, input_size), batch_size=BATCH_SIZE, classes=['Carton', 'Plastico','Vidrio'])
```

```
    val_batches =
ImageDataGenerator(preprocessing_function=preprocess_func).flow_from_directory(val_path,
target_size=(input_size, input_size), batch_size=BATCH_SIZE, classes=['Carton', 'Plastico','Vidrio'])
    # Do a fine-tuning first for the added top layers, using the best epoch of the fine-tuning performed
previously
    model_history, base_model, model = model_fine_tuning(RED, base_model, model, train_batches,
val_batches, max_acc_epoch, STEPS_PER_EPOCH, VALIDATION_STEPS, BASE_RESULTS, fold)
    # Deep-tuning: train all layers
    model_history = model_deep_tuning(RED, input_size, base_model, model, train_batches, val_batches,
TEST_SET_PATH, EPOCHS_DEEP_TUNING, STEPS_PER_EPOCH, VALIDATION_STEPS, BASE_RESULTS,
fold, results_file)
    all_acc_histories_deep.append(model_history.history['val_acc'])


print(all_acc_histories_deep)
average_acc_history_deep = [np.mean([x[i] for x in all_acc_histories_deep]) for i in
range(EPOCHS_DEEP_TUNING)]
print(average_acc_history_deep)
max_acc_epoch_deep = average_acc_history_deep.index(max(average_acc_history_deep)) + 1
print('Best epoch for deep-tuning: ' + str(max_acc_epoch_deep))


# Final training with all images using the best epoch found for fine and deep tuning
results_file = open(results_path + "/final_train_test.txt", "w")
results_file.write('Fine-tuning epochs: ' + str(max_acc_epoch) + "\n")
results_file.write('Deep-tuning epochs: ' + str(max_acc_epoch_deep) + "\n")
all_train_samples_num = 1612
STEPS_PER_EPOCH = ceil(all_train_samples_num / BATCH_SIZE)
base_model, model = build_model(RED, input_size)
train_batches = ImageDataGenerator(preprocessing_function=preprocess_func, rotation_range=30,
vertical_flip=True, horizontal_flip=True, zoom_range=
[0.8,1.2]).flow_from_directory(ALL_TRAIN_SET_PATH, target_size=(input_size, input_size),
batch_size=BATCH_SIZE, classes=['Carton', 'Plastico','Vidrio'])
model_history, base_model, model = model_fine_tuning(RED, base_model, model, train_batches, None,
max_acc_epoch, STEPS_PER_EPOCH, None, BASE_RESULTS, 'final')
model_history = model_deep_tuning(RED, input_size, base_model, model, train_batches, None,
TEST_SET_PATH, max_acc_epoch_deep, STEPS_PER_EPOCH, None, BASE_RESULTS, 'final', results_file)
```

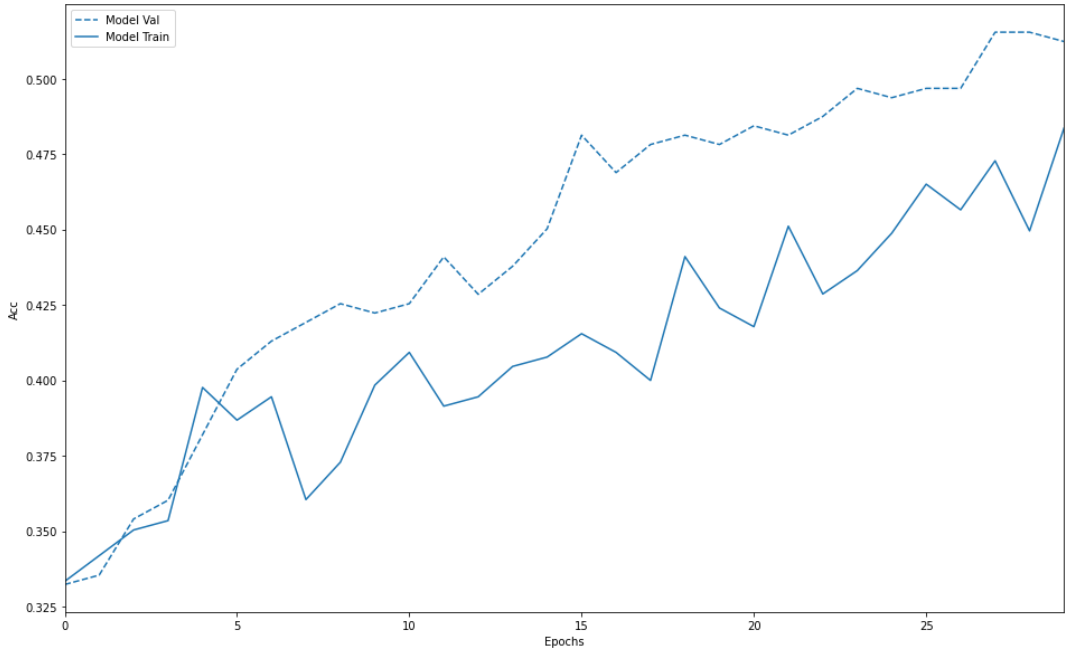
Fold 1. Pre-fine Tuning



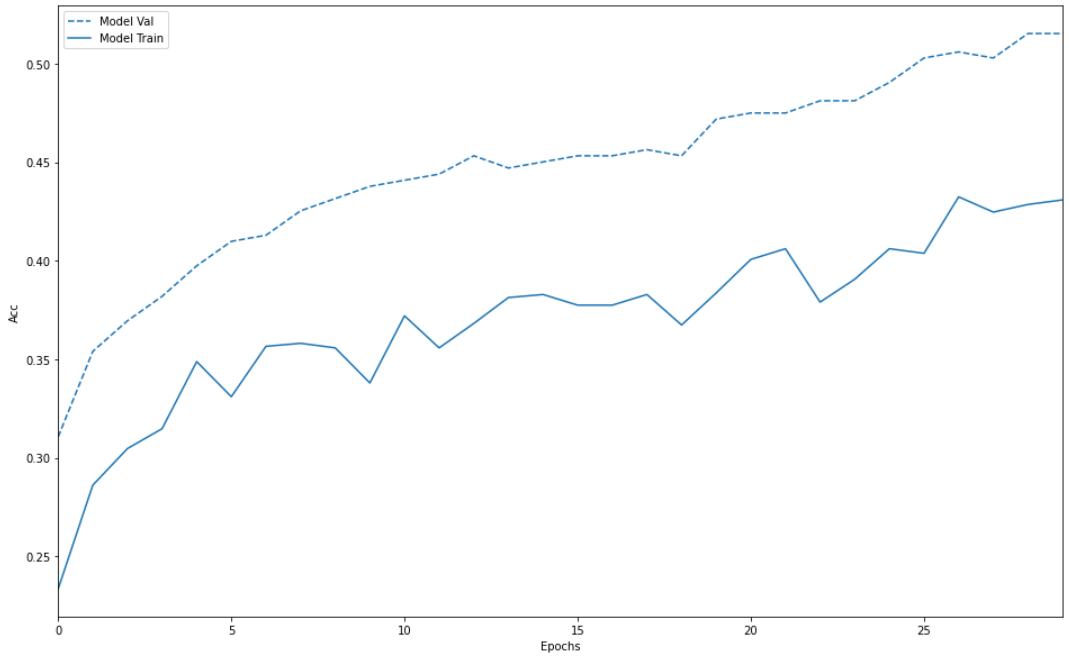
Fold 2. Pre-fine Tuning



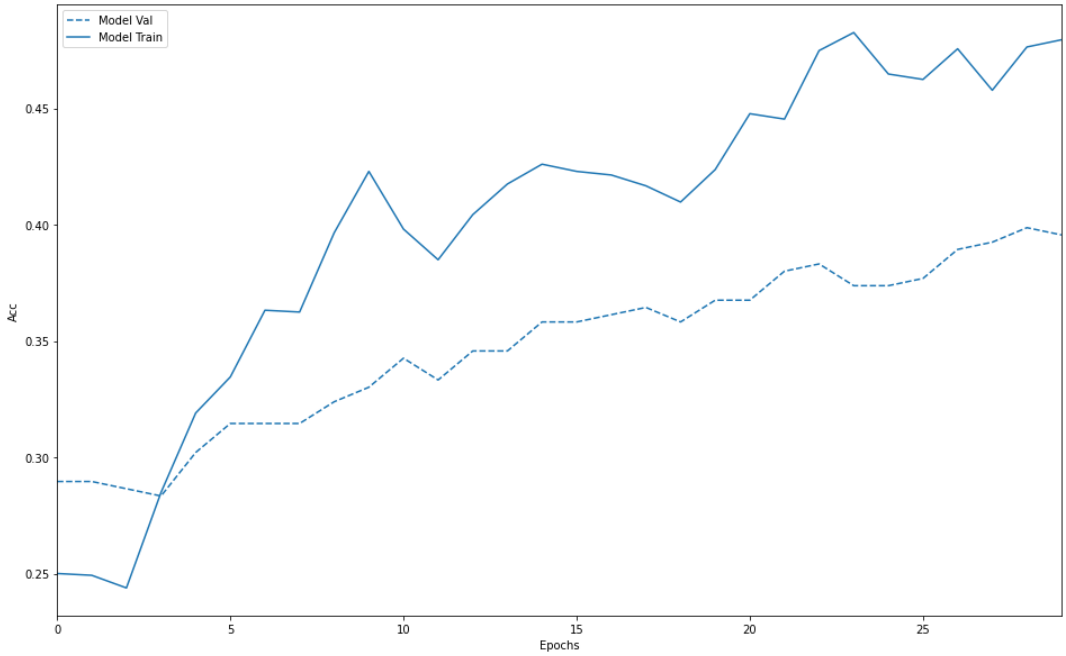
Fold 3.Pre-fine Tuning



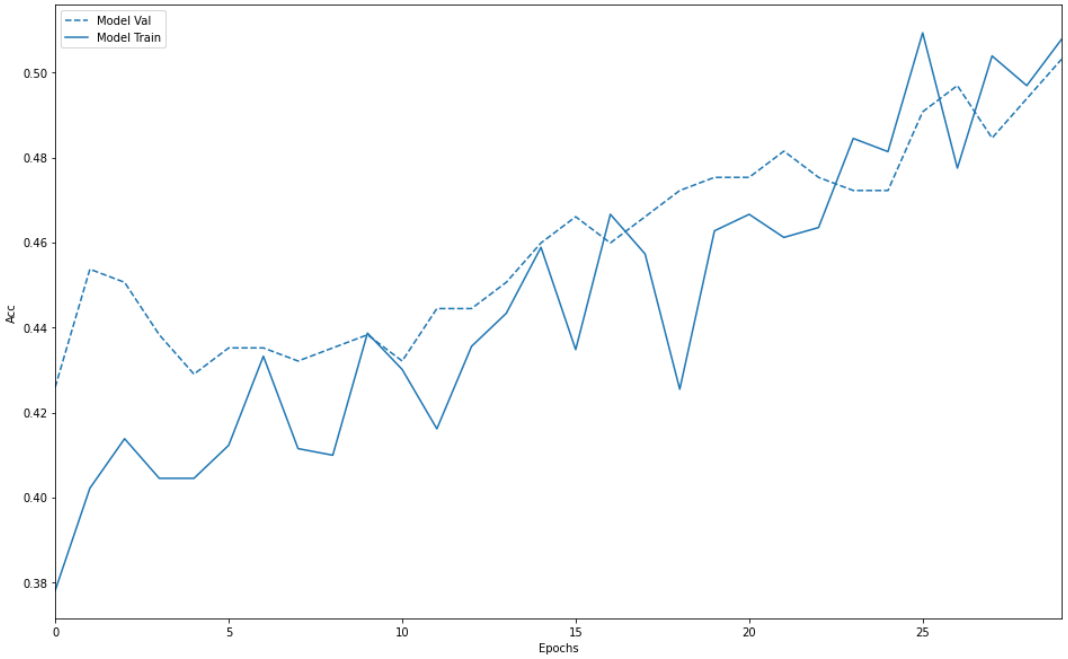
Fold 4. Pre-fine Tuning



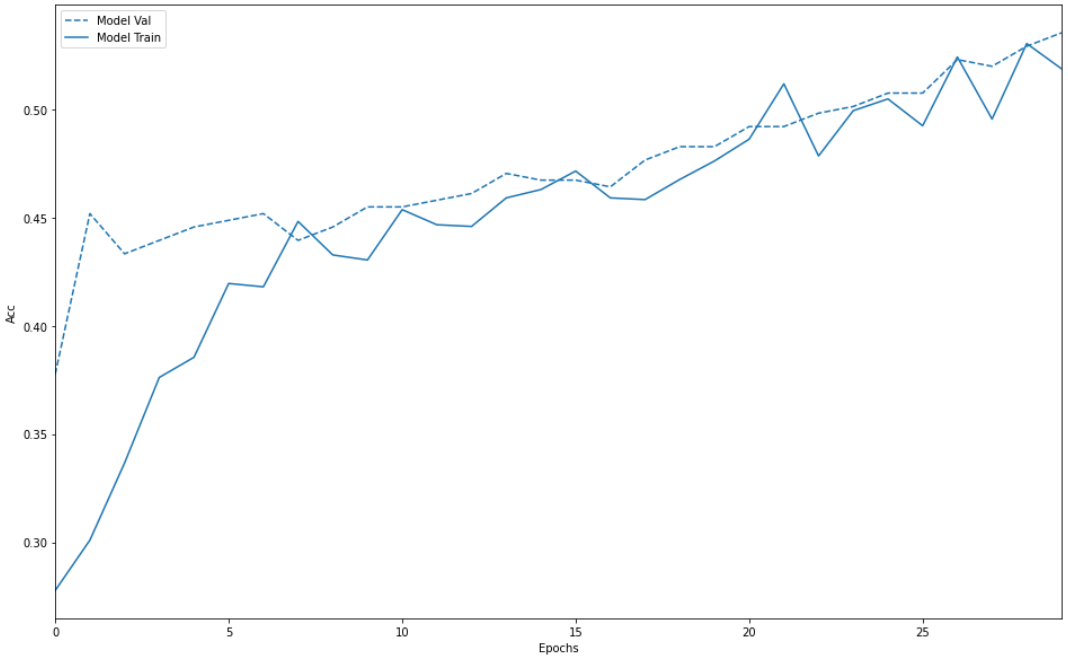
Fold 5. Pre-fine Tuning



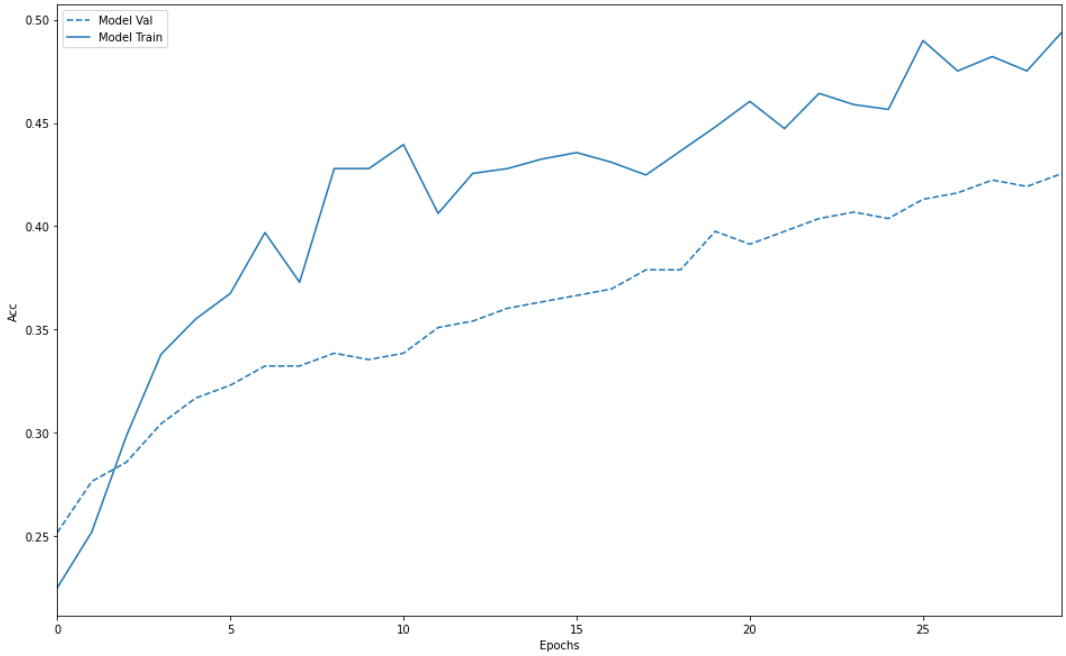
Fold 1. Fine Tuning



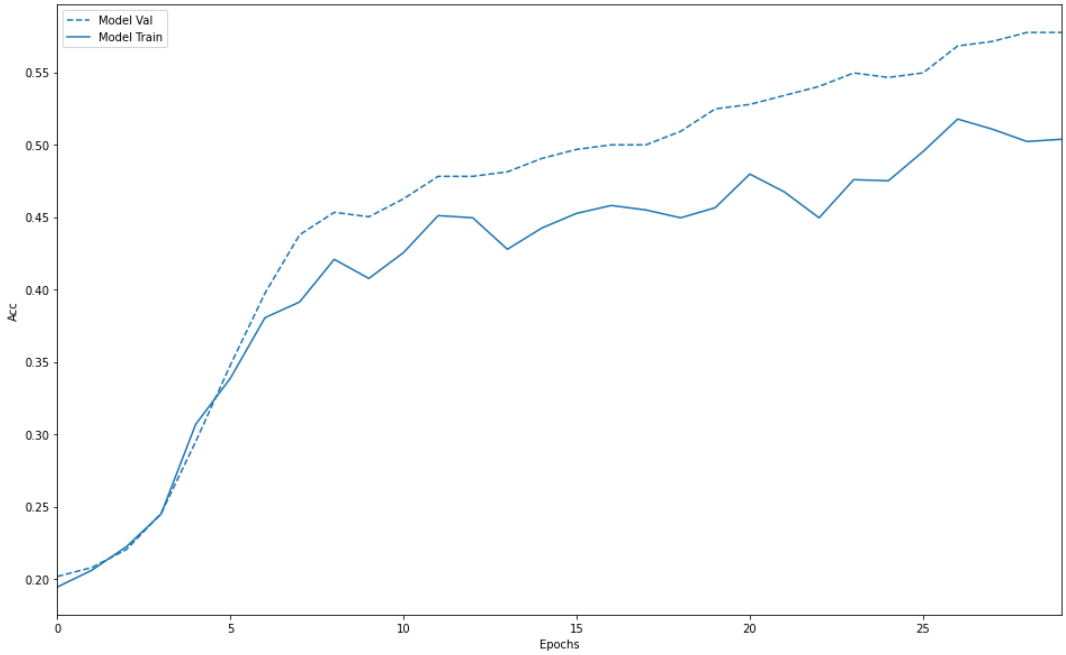
Fold 2. Fine Tuning



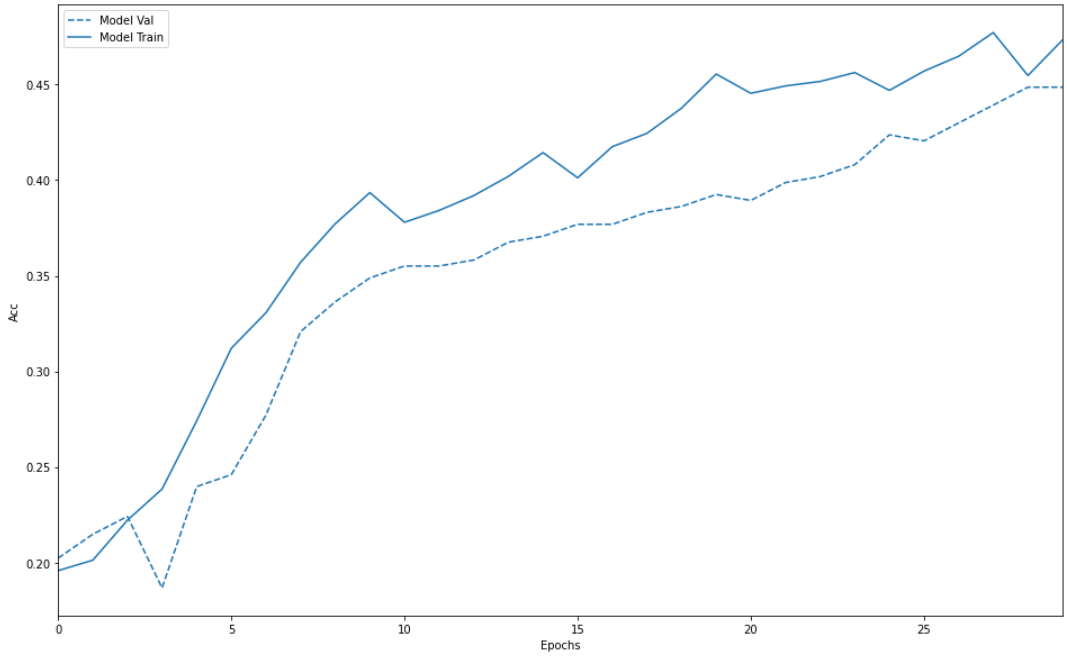
Fold 3. Fine Tuning



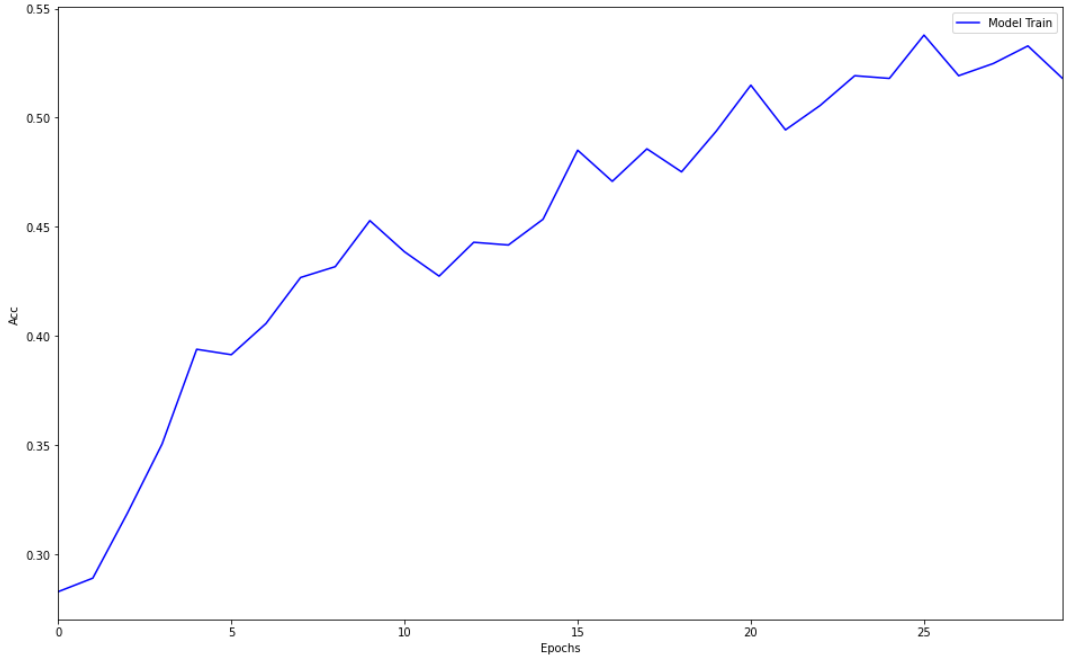
Fold 4. Fine Tuning



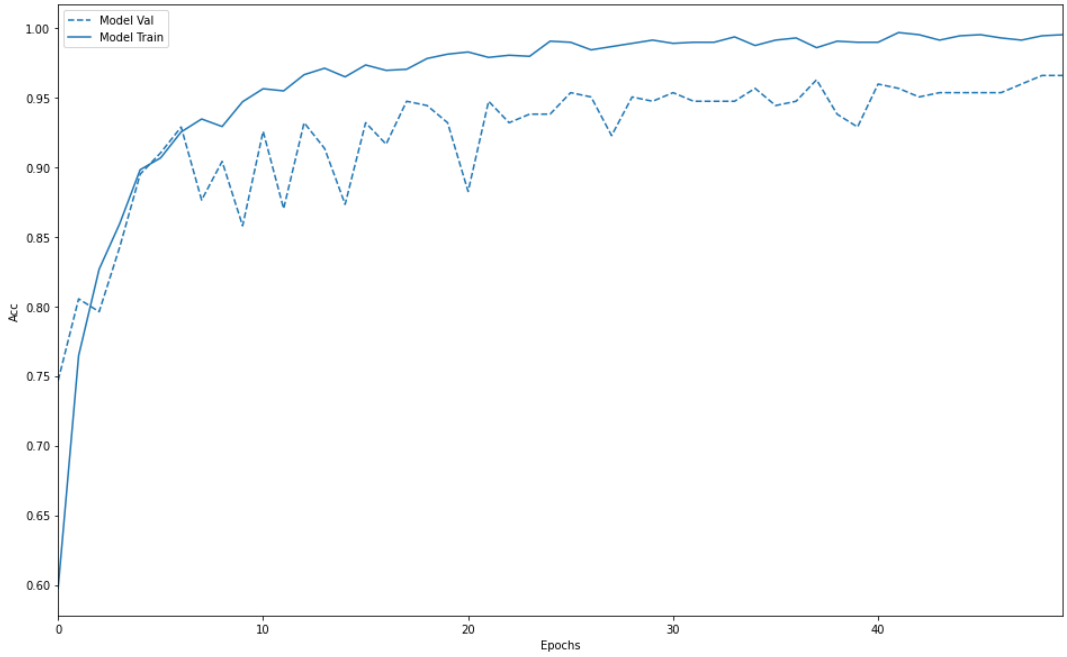
Fold 5. Fine Tuning



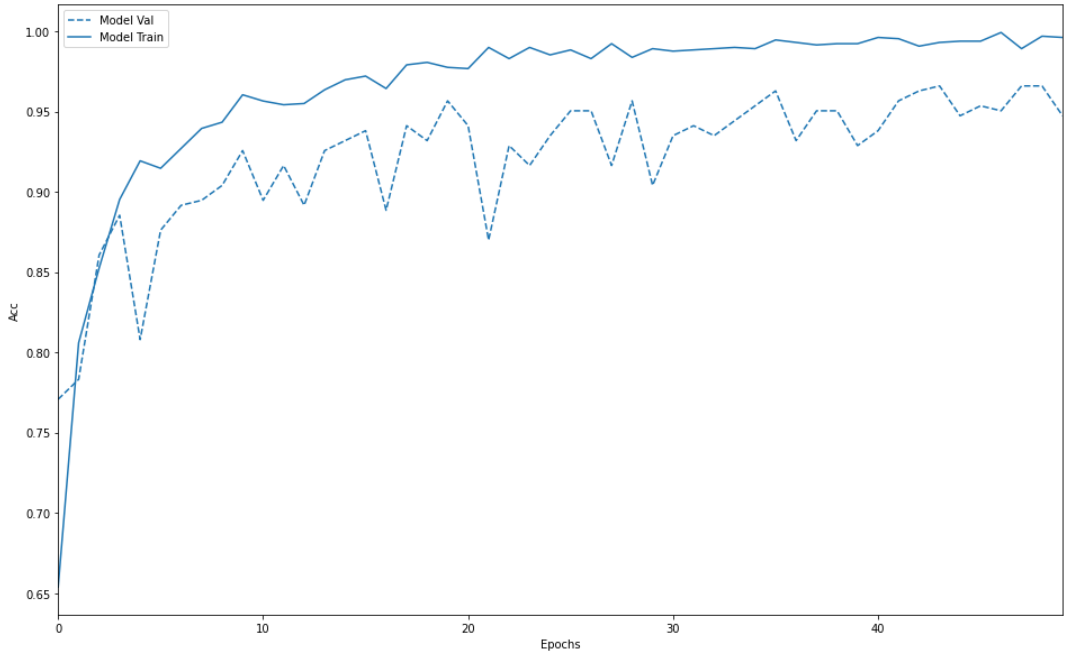
Final. Fine Tuning



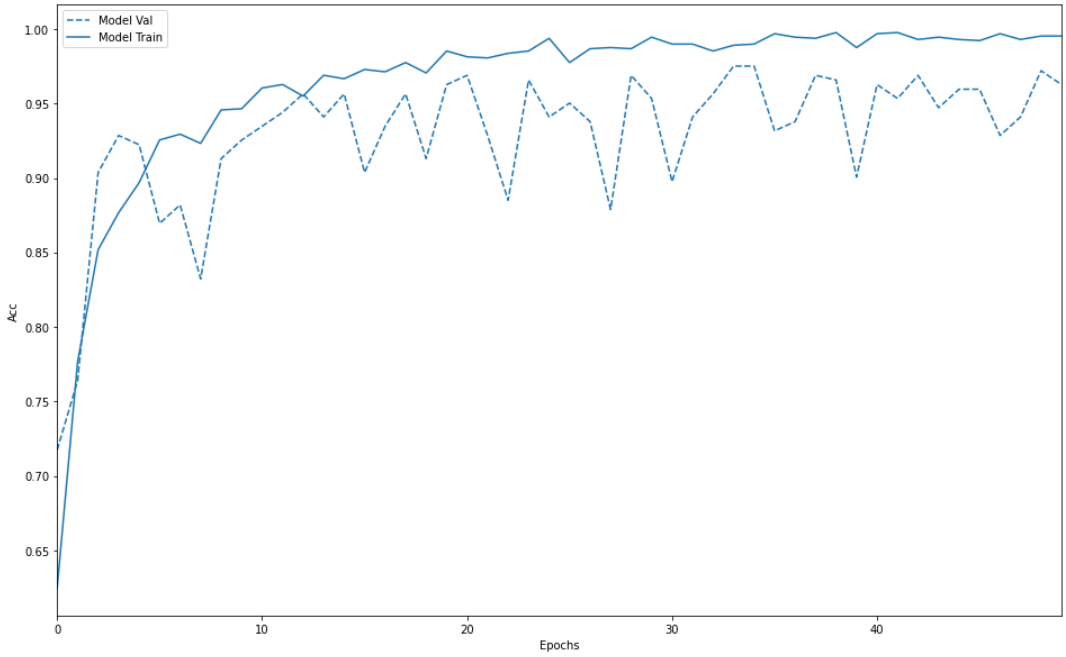
Fold 1. Deep Tuning



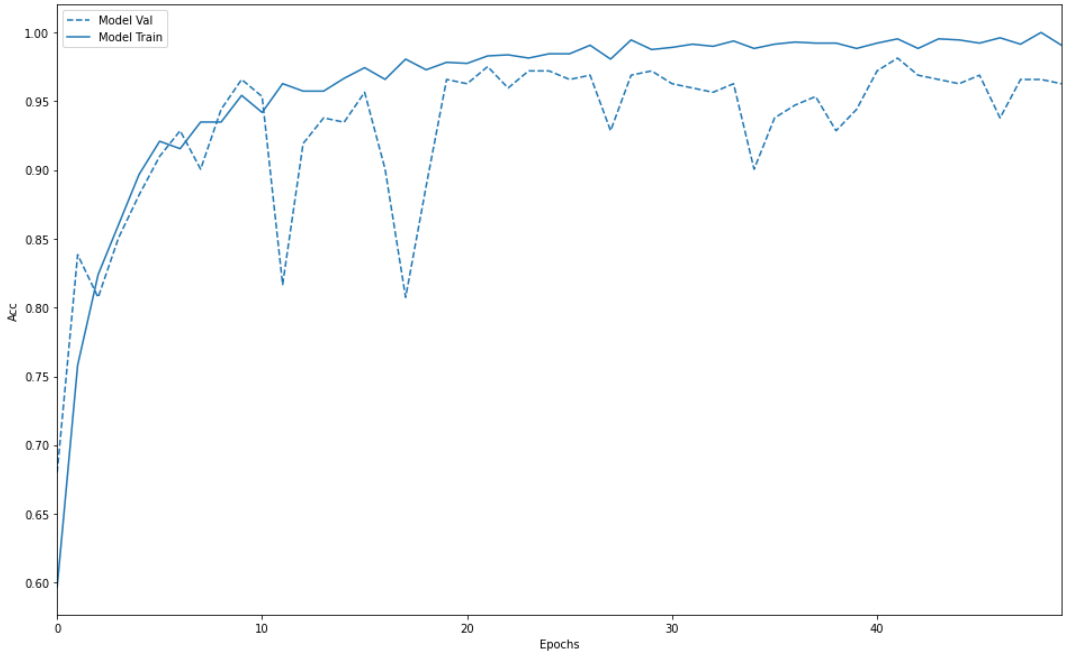
Fold 2. Deep Tuning



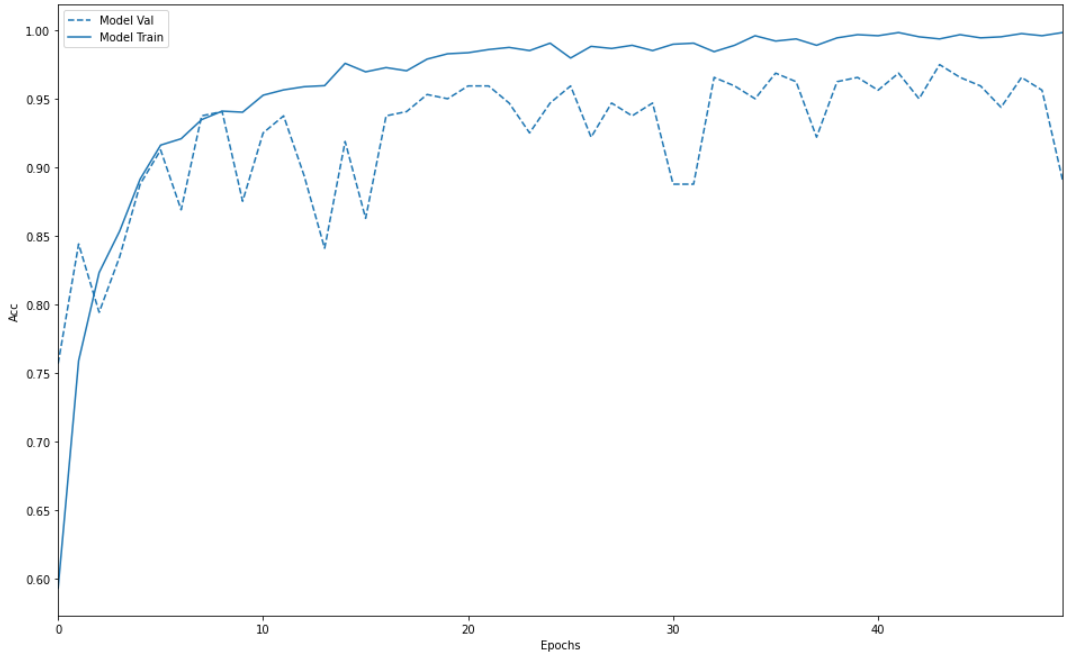
Fold 3. Deep Tuning



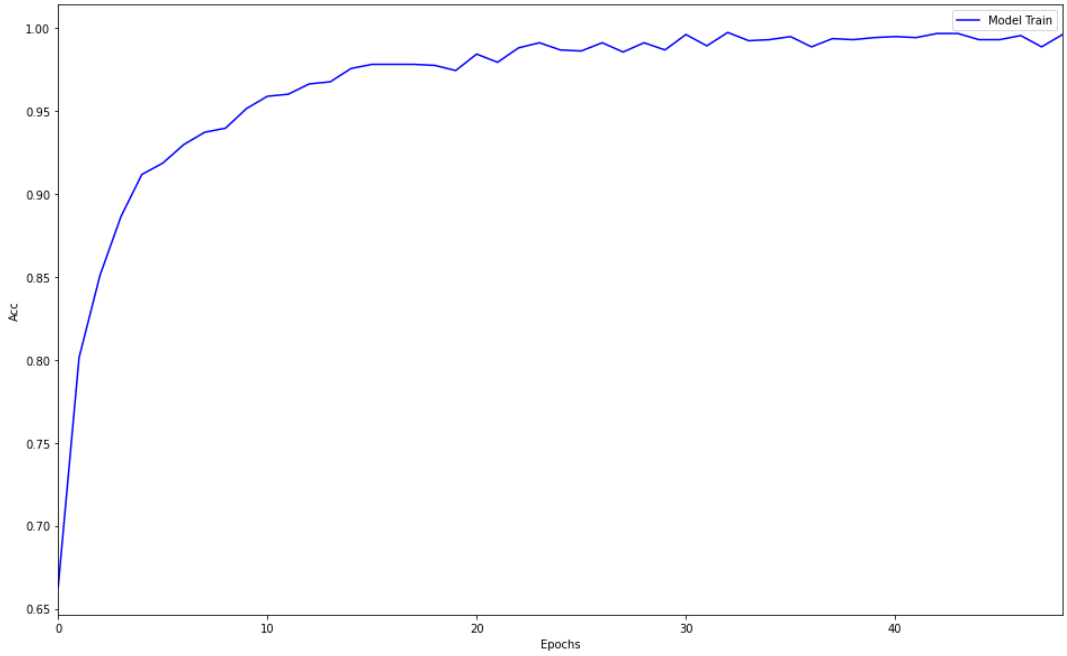
Fold 4. Deep Tuning



Fold 5. Deep Tuning



Final. Deep Tuning



ANEXO VII.
CÓDIGO DEL BOT DE LA RASPBERRY

CÓDIGO SERVICIO (.service)

```
[Unit]
Description = Ejecucion_RedNeuronal

[Service]
User=pi
ExecStart=/bin/bash /home/pi/Documents/Red.sh
Restart=always
RestartSec=2

[Install]
WantedBy=multi-user.target
```

CÓDIGO BASH (.sh)

```
#!/bin/bash

cd /home/pi
source ~/.profile
workon gcolab
cd /home/pi/Documents
python3 -u EJECUTABLE.py &> /tmp/Red_log.log
```

```
# -*- coding: utf-8 -*-
"""Carga_red.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1RsdjbnjVkJInCKH_FDzOwcrkW1_MqYtE
"""

# Commented out IPython magic to ensure Python compatibility.
# cargamos la red que queremos visualizar con los pesos ya ajustados después del entrenamiento

# %tensorflow_version 1.x  # algunas funciones de keras-vis solo funcionan con esta version de tensorflow

#CARGA RED
import numpy as np

import tensorflow as tf


from keras.preprocessing import image

#from google.colab import drive

from keras.applications import VGG16
from keras.layers import Dense, Input, GlobalAveragePooling2D
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import RMSprop


#drive.mount('/content/gdrive')

# construimos el modelo

def get_base_model(network, input_size):
    input_tensor = Input(shape=(input_size, input_size, 3))

    if network == 'xception':
        base_model = Xception(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'vgg16':
        base_model = VGG16(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'vgg19':
        base_model = VGG19(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'resnet50':
        base_model = ResNet50(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'resnet50v2':
        base_model = ResNet50V2(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'inceptionv3':
        base_model = InceptionV3(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'inceptionresnetv2':
        base_model = InceptionResNetV2(input_tensor=input_tensor, weights='imagenet',
include_top=False)
    elif network == 'mobilenet':
        base_model = MobileNet(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'densenet':
        base_model = DenseNet121(input_tensor=input_tensor, weights='imagenet', include_top=False)
    elif network == 'nasnetmobile':
        base_model = NASNetMobile(input_tensor=input_tensor, weights='imagenet', include_top=False)
```

```
    elif network == 'mobilenetv2':
        base_model = MobileNetV2(input_tensor=input_tensor, weights='imagenet', include_top=False)
    else:
        print('Network unknown')
        return ''

    return base_model


def build_model(network, input_size):
    # Cambiar el base_model en función de la red que vayamos a utilizar
    base_model = get_base_model(network, input_size)
    if base_model == '':
        print('Network unknown')
        return

    # Añadimos sólo GlobalAveragePooling2D,
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    # x = Dense(1024, activation='relu')(x)
    # Añadimos dropout?
    predictions = Dense(2, activation='softmax')(x)
    model = Model(inputs=base_model.input, outputs=predictions)
    return base_model, model

RED = 'vgg16'
INPUT_SIZE_BY_NETWORK = {
    'xception': 224,
    'vgg16': 224,
    'vgg19': 224,
    'resnet50': 224,
    'resnet50v2': 224,
    'inceptionv3': 224,
    'inceptionresnetv2': 224,
    'mobilenet': 224,
    'densenet': 224,
    'nasnetmobile': 224,
    'mobilenetv2': 224
}
input_size = INPUT_SIZE_BY_NETWORK[RED]
base_model, model1 = build_model(RED, input_size)


# cargamos los pesos de la red entrenada
from keras.models import load_model

#model1.load_weights('/content/gdrive/My Drive/TFGBasura/resultado/vgg16/fold_final_w_net.h5')

model1.load_weights('/home/pi/Documents/fold_final_w_net.h5')


##################################################################################
#################################
import sys
import picamera
import time
import RPi.GPIO as GPIO
import cv2
#Para utilizar numeracion fisica de la placa, no el numero del GPIO
# Esta numeracion empieza en la esquina superior izquierda y es correlativa de Izquierda a Derecha y
luego Abajo.
```

```
GPIO.setmode(GPIO.BOARD)
GPIO.setup(29, GPIO.IN) #GPIO 5 "Clasifica"
GPIO.setup(7, GPIO.OUT) #GPIO 4 "Carton"
GPIO.setup(11, GPIO.OUT) #GPIO 17 "Plastico"
GPIO.setup(13, GPIO.OUT) #GPIO 27 "Vidrio"
GPIO.setup(15, GPIO.OUT) #GPIO 22 "Inconcluyente"
#Habilitar camara


# Funciones para el preprocesamiento
from tensorflow.keras import backend as K
def preprocess_input1(x): # For VGG16, VGG19 and ResNet50
  data_format = K.image_data_format()
  assert data_format in {'channels_last', 'channels_first'}
  if data_format == 'channels_first':
      # 'RGB'->'BGR'
      x = x[::-1, :, :]
      # Zero-center by mean pixel

      x[0, :, :] -= 103.939
      x[1, :, :] -= 116.779
      x[2, :, :] -= 123.68
  else:
      # 'RGB'->'BGR'
      x = x[:, :, ::-1]
      # Zero-center by mean pixel
      x[:, :, 0] -= 103.939
      x[:, :, 1] -= 116.779
      x[:, :, 2] -= 123.68
  return x

def preprocess_input2(x): # For InceptionV3, Xception
  x /= 255.
  x -= 0.5
  x *= 2.
  return x

while True:


    if GPIO.input(29):



        #CAPTURA IMAGEN
        with picamera.PiCamera() as picam:
            picam.resolution =(2560,1920)
            picam.capture ('/home/pi/Documents/Objeto.jpg')
            picam.close()


        #CAMBIO TAMAÑO IMAGEN
        imagen = cv2.imread('/home/pi/Documents/Objeto.jpg')
        imagen_escalada = cv2.resize(imagen, (224,224))
        cv2.imwrite('/home/pi/Documents/Objeto.jpg', imagen_escalada)


        #CARGA Y PREPROCESAMIENTO DE IMAGEN
        imgPath = ('/home/pi/Documents')
        RED == 'vgg16'
          # para el preprocesamiento
```

```
    if RED == 'vgg16' or RED == 'vgg19' or RED == 'resnet50':
        print("Configurando función de preprocesamiento para VGG16, VGG19 o ResNet50")
        preprocess_func = preprocess_input1
    elif RED == 'xception' or RED == 'inceptionv3':
        print("Configurando función de preprocesamiento para Xception o InceptionV3")
        preprocess_func = preprocess_input2
    else:
        raise Exception("Red no compatible con función de preprocesamiento")

#     Se aplcia la funcion de preprocesamiento correspondiente (en VGG16, pasar a de RGB  BGR)
    test_batches =
ImageDataGenerator(preprocessing_function=preprocess_func).flow_from_directory(imgPath,
target_size=(224, 224), batch_size=1, classes=[''],shuffle=False)

    filenames = test_batches.filenames
    claass = test_batches.classes
    img_tensor = test_batches


#       img=tf.keras.preprocessing.image.load_img(imgPath,target_size=(224,224))
#       img_tensor = image.img_to_array(img)
#       img_tensor = np.expand_dims(img_tensor, axis=0)
#       img_tensor /= 255.
#       print(img_tensor.shape)

#      import matplotlib.pyplot as plt
#       plt.style.use('ggplot')
#
#       plt.imshow(img_tensor[0])
#       plt.axis('off')
#       plt.show()


    #CLASIFICACION

    # clasificamos la imagen
    #img2 = np.expand_dims(img, axis=0)
    x_batch, y_batch = next(img_tensor)
#       import matplotlib.pyplot as plt
#       plt.imshow(x_batch[0])
#       plt.axis('off')
#       plt.show()
    clasif=model1.predict(x_batch)

#     Determinamos el tipo de material en base al porcentaje sacado por la red de cada material
    print(clasif[0])
    if (clasif[0,0] >= 0.5):
        print('Es carton')
        GPIO.output(7, GPIO.HIGH)
        time.sleep(5) #Tiempo para que se comunique con el autómata
        GPIO.output(7, GPIO.LOW)

    elif (clasif[0,1] >= 0.5):
        print('Es plastico')
        GPIO.output(11, GPIO.HIGH)
        time.sleep(5) #Tiempo para que se comunique con el autómata
        GPIO.output(11, GPIO.LOW)

    elif (clasif[0,2] >= 0.5):
```

```
            print('Es vidrio')
            GPIO.output(13, GPIO.HIGH)
            time.sleep(5) #Tiempo para que se comunique con el autómata
            GPIO.output(13, GPIO.LOW)

        else:
            print ('Inconcluyente')
            GPIO.output(15, GPIO.HIGH)
            time.sleep(5) #Tiempo para que se comunique con el autómata
            GPIO.output(15, GPIO.LOW)



#       print("Enter para continuar o 'p' para parar")
#       tecla=sys.stdin.read(1)
#       if tecla == 'p':
#           print('Detenido')
#           break
        time.sleep (5)

GPIO.cleanup()
```