

Trabajo Fin de Grado

Desarrollo de una aplicación web de gestión de flota de AGVs para el sector de logística.

AGV fleet management's web application development for the logistic quadrant.

Autor

Fernando Ballester Tolón

Director

Julián Fernández Navajas

Ingeniería informática

Escuela de Ingeniería y Arquitectura

2022

RESUMEN.

DESARROLLO DE UNA APLICACIÓN WEB DE GESTIÓN DE FLOTA DE AGVS PARA EL SECTOR DE LOGÍSTICA.

Un AGV (Automatic Guided Vehicle), como su propio nombre indica, es un vehículo que es guiado de manera automática. Es un robot movable que sigue unas líneas o cables en el suelo, o usa ondas de radio, cámaras de visión, imanes o láseres para la navegación. Son más frecuentemente utilizados en aplicaciones industriales para transportar materiales pesados alrededor de un gran entorno industrial, como una fábrica o un almacén.

En este Trabajo de Fin de Grado se va a diseñar e implementar una aplicación que permita gestionar una flota, es decir, un conjunto de AGVs diseñados por la empresa Ergomaq, la cual es una empresa aragonesa de automatización industrial. Los AGV de Ergomaq son filoguiados, se sincronizan con el control superior leyendo tags RFID. Lo que se busca es realizar un control global de la flota, y permitir a los usuarios del entorno realizar peticiones para desplazar estos vehículos de un sitio a otro.

Para conseguir estos objetivos, se han diseñado distintas aplicaciones mediante la herramienta "Visual Studio", para hacer que en su conjunto, permitan controlar toda la situación perfectamente.

ÍNDICE.

RESUMEN.	2
Capítulo 1	6
Introducción.	6
1.1. Introducción de la situación y motivación.	6
1.2. Objetivos.	8
1.3. Estructura de la memoria.	8
Capítulo 2.	10
Equipos, protocolos y herramientas de desarrollo.	10
2.1. Equipos e infraestructura.	10
2.1.1. Flota de AGVs.	11
2.1.1.1. PLC (Programmable Logic Controller).	14
2.1.1.2. Lector de tags RFID.	14
2.1.1.3. Pantalla HMI.	15
2.1.2. Circuito en el suelo.	15
2.1.2.1. Zona de almacenamiento de carritos vacíos.	16
2.1.2.2. Zona de almacenamiento de carritos llenos.	16
2.1.2.3. Zonas de carga de baterías.	16
2.1.2.4. Zonas de petición.	17
2.1.3. Aplicación de gestión.	17
2.1.4. Aplicación de petición.	17
2.1.5. Base de datos.	17
2.2. Protocolos.	17
2.2.1. Protocolos de comunicación entre la aplicación de gestión y la flota de AGVs.	18
2.2.1.1. Protocolo OPC.	18
2.2.1.2. Protocolo OPC UA.	19
2.2.2. Protocolos de comunicación entre la aplicación de gestión y la base de datos.	19
2.2.3. Protocolos entre la aplicación de gestión y cada aplicación de petición.	20
2.3. Herramientas de desarrollo.	20
2.3.1. Lenguajes de programación.	20
2.3.2. Aplicaciones.	20
2.3.2.1. Aplicaciones utilizadas por el equipo de desarrolladores de Ergomaq.	20
2.3.2.2. Aplicaciones, entornos y herramientas utilizadas en el desarrollo de las aplicaciones.	21
2.3.2.3. Aplicaciones utilizadas en el desarrollo de la base de datos.	21
Capítulo 3.	25
Sistema desarrollado.	25
3.1. Implementación base.	27
3.1.1. Implementación del circuito.	28
3.1.1.1. Primera aproximación de la implementación software del circuito.	29
3.1.1.2. Implementación software final del circuito.	30
3.1.1.2.1. Implementación de la clase Nodo.	32
3.1.1.2.2. Implementación de la clase Distancia.	33
3.1.1.2.3. Implementación de la clase Distancia Única.	34
3.1.1.2.4. Ampliación de la implementación de la clase Nodo.	34
3.1.2. Implementación de los AGVs (PLCs).	36
3.1.3. Implementación de las peticiones de AGVs.	37
3.1.3.1. Implementación de software de la clase Petición.	37

3.1.4. Implementación del movimiento de los AGVs por el circuito.	37
3.2. Implementación de la aplicación de petición.	38
3.2.1. Implementación de la parte de lanzar peticiones.	39
3.2.2. Implementación de la creación y edición de circuitos.	39
3.2.3. Traslado de aplicación Windows a aplicación web.	42
3.3. Aplicación de gestión.	43
3.3.1. Implementación de las tareas periódicas.	43
Capítulo 4.	47
Pruebas realizadas.	47
4.1. Introducción.	47
4.2. Alcance de las pruebas.	47
4.2.1. Pruebas de Usabilidad.	47
4.2.2. Pruebas Funcionales.	47
4.2.2.1. Prueba: lectura y escritura con 1 PLC y 2 PLCs.	47
4.2.2.2. Prueba: creación y edición de circuitos.	48
4.2.2.3. Prueba: simulación de circuito con 1 PLC.	48
4.2.2.4. Prueba: circuito real con 2 AGVs.	50
4.2.3. Pruebas de Compatibilidad.	51
4.3. Estrategia y Desarrollo.	51
4.3.1. Estrategia.	51
4.3.2. Tipo de Pruebas.	52
4.4. Resultados y Conclusiones.	52
Capítulo 5.	54
Conclusiones y líneas futuras.	54
5.1. Conclusiones.	54
5.2. Líneas futuras.	54
Anexos.	57
Anexo 1: Hardware AGV.	57
Anexo 2: Tipos de información en los tags RFID.	64
Anexo 3: Esquemas eléctricos.	66
Anexo 4: Variables en la clase PLC.	74
Anexo 5: Algoritmos relacionados.	78
Anexo 6: Implementación de las dos peticiones predeterminadas.	78
Anexo 7: Ventajas de OPC UA.	82
Bibliografía.	83

Capítulo 1

Introducción.

1.1. Introducción de la situación y motivación.

Ergomaq es una empresa aragonesa de automatización industrial situada en Villanueva de Gállego. Se dedica al desarrollo de líneas automáticas (conjunto de máquinas individuales, que unidas, forman una secuencia de procesos donde el resultado final es que la mercancía en cuestión quede todo lo envuelta y sellada posible, cuyo objetivo final es que se complete el packaging y la mercancía quede totalmente lista para su transporte), células robotizadas (conjunto de máquinas que se organizan dentro de una célula de automatización industrial en la que se instalan uno o varios robots programables), maquinaria específica y robótica móvil. En Ergomaq se hace todo el proceso de fabricación, desde el desarrollo 3D de todo el diseño mecánico, el diseño eléctrico, la mecanización de piezas, cableado, montaje, hasta la programación y comunicación de los elementos de robótica necesarios para su funcionamiento.

Los AGVs (Automatic Guided Vehicles), como su propio nombre indica, son vehículos guiados de manera automática cuya finalidad es transportar materiales. Pueden ser de distintos tipos en función del sistema de navegación: aquellos que utilizan ondas de radio, cámaras de visión, imanes, láseres o líneas o cables en el suelo. Estos últimos que utilizan líneas o cables en el suelo se denominan filoguiados, y son el tipo de AGV que se utilizarán en este trabajo. Los AGVs funcionan mediante baterías, y podrán recargarse a través de unas pastillas para carga de baterías incluidas en su parte inferior. El AGV filoguiado dispone de sensores magnéticos que son los encargados de leer el campo magnético de la cinta, y cuatro motores que mueven las cuatro ruedas, dos en el eje trasero y dos en el eje delantero, pudiendo así conseguir un movimiento bidireccional.



Imagen 1.1: Foto de un AGV.

En el circuito por el que se mueven los AGV, junto a la cinta magnética, existen pequeños dispositivos fijados al suelo denominados tags. Un tag es un objeto circular que contiene información útil sobre el entorno. La comunicación con los tags se realiza mediante un lector RFID (radio-frequency identification) incluido en la parte inferior del vehículo.



Imagen 1.2: Foto de un tag, y a su izquierda cinta magnética.

Existe un conexionado interno dentro del vehículo para conectar sensores, motores, y equipos de transmisión radio, el cual es controlado por un dispositivo interno llamado PLC (Programmable Logic Controller), donde se carga un programa para gestionar las conexiones y órdenes.

Además de AGVs, se usarán unos contenedores con ruedas denominados carritos, cuyo objetivo es transportar los materiales. El AGV agarra el carrito con un cilindro que dispone en la parte superior central denominado pin, que puede subir y bajar.

Por ello, el carrito debe de contar con un orificio en la parte inferior central, de manera que dicho pin pueda elevarse sin problema. Una vez el pin ha subido y se ha introducido en el orificio del carrito, se efectúa el transporte por medio de fricción, con ayuda de varias roldanas disponibles en la parte superior del vehículo, que ayudan a una mejor sujeción del carrito.



Imagen 1.3: Foto de un AGV llevando un carrito azul.

Dentro de la organización de la empresa, es en el departamento de programación donde se desarrolla el software necesario para los diferentes equipos, entre los que se encuentran PLCs, Robots, cámaras para visión artificial, HMIs, SCADAS o aplicaciones sobre PC. En este departamento es en donde se va a desarrollar el presente trabajo de fin de grado.

En un entorno industrial que requiera de transporte de materiales muy pesados o muy grandes, utilizar los carritos y los AGVs para transportar el material será de gran ayuda. Los AGVs se utilizan para llevar los carritos automáticamente de un lugar a otro del entorno industrial. Los AGVs deben gestionarse, de tal manera que, a la hora de llevar un carrito a un destino, se escoja la ruta más óptima. Esto permite ahorrar tiempo. Para facilitar esta labor a los trabajadores, sería de gran utilidad tener una aplicación que les permitiese realizar peticiones a la flota de AGVs, para que éstos les lleven un carrito al lugar donde se encuentran, o para que transporten un carrito hasta otro punto de la fábrica. El planteamiento del desarrollo de este trabajo consiste en la combinación de los AGVs, el circuito y los carritos, para que en sí constituyan un sistema de transporte óptimo y eficaz que ahorre tiempo y esfuerzo.

1.2. Objetivos.

El objetivo principal es abordar el desarrollo, diseño e implementación de la gestión de una flota de AGVs, y así ahorrar tiempo y esfuerzo.

Para ello, se propone el desarrollo de una aplicación que debe gestionar las peticiones de AGVs para recoger y enviar mercancías de unos puntos a otros de una fábrica. Para que esta aplicación sea efectiva, existen una serie de objetivos secundarios, como el control de accesos seguros y personalizables, la monitorización de circuitos y de AGVs, la gestión de la lectura de tags para gestionar cruces y paradas, la gestión de baterías optimizando sus tiempos de carga, la optimización de recorridos, y la solicitud de movimiento de AGVs desde distintos puntos para transportar los carritos.

Este diseño también tiene que ser lo más flexible posible para poder adaptarlo a distintos entornos y que la disposición de los elementos de la aplicación pueda estar en distintos lugares. En resumen, que sea adaptable a diferentes entornos.

1.3. Estructura de la memoria.

La memoria se compone de los siguientes apartados:

- **Capítulo 1:** Introducción y objetivos.
- **Capítulo 2:** Descripción de los equipos y las herramientas utilizadas en el trabajo.
- **Capítulo 3:** Descripción del sistema desarrollado donde se presentan las aplicaciones desarrolladas.
- **Capítulo 4:** Descripción de la puesta en funcionamiento del sistema y de las pruebas realizadas.
- **Capítulo 5:** Conclusiones finales y posibilidades de mejora.

A modo de anexos a la memoria, se incluyen otros apartados con información complementaria sobre el sistema desarrollado, además de un glosario de términos útil para la comprensión de los distintos elementos del sistema.

Capítulo 2.

Equipos, protocolos y herramientas de desarrollo.

2.1. Equipos e infraestructura.

Para que el sistema funcione correctamente, se debe disponer de cinco elementos clave. Por un lado, de una **flota de AGVs**. Por otro lado, de un **circuito en el suelo** formado por tags y cinta magnética adhesiva, sobre el que se desplazarán los AGVs. Además necesitamos la **aplicación de gestión** de dicha flota, que estará corriendo en una máquina y se encargará de establecer sus rutas y movimientos. El conjunto de **aplicaciones de petición** de AGVs por parte de los usuarios, que corren en dispositivos móviles. Por último, una **base de datos** donde se guarda la información, y que sirva de interfaz entre la aplicación de gestión y de petición de AGVs, que corre también en una máquina.

La comunicación red entre los distintos equipos es clave para la comprensión de la unidad del sistema. En la imagen 2.1 se muestra el esquema de la conexión red entre los distintos equipos. Existe una red *LAN* a la que están conectados. Puede haber varios puntos de acceso (*AP*), a los que se conectan tanto los equipos con las aplicaciones de petición como los AGVs. La base de datos y la aplicación de gestión pueden estar corriendo en el mismo equipo, y dado que van a ser fijos, se puede conectar a la red por cable.

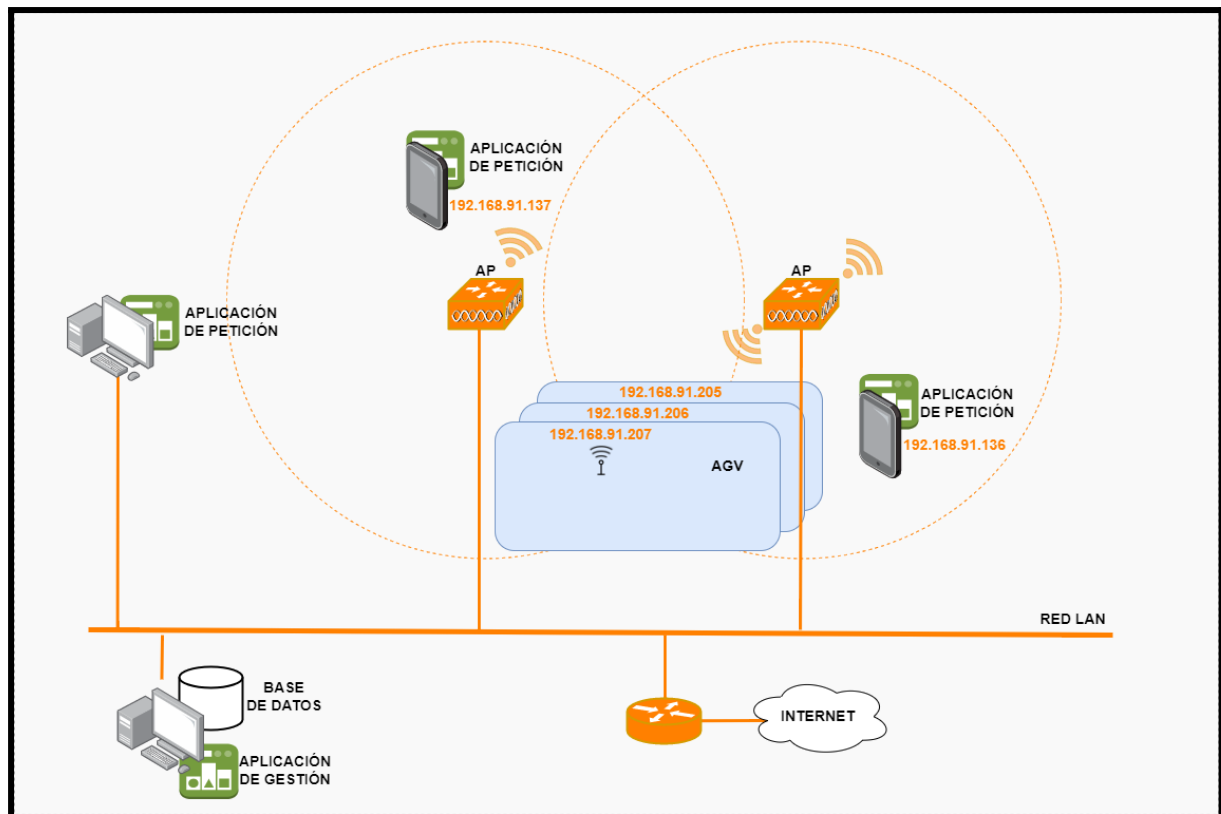


Imagen 2.1: Esquema de la red LAN de todo el sistema.

2.1.1. Flota de AGVs.

El hardware del AGV (*Automatic Guided Vehicle*) está compuesto por varios dispositivos. **[Para ver la información sobre cada uno de los dispositivos, mira en el anexo 1]**, que podemos agrupar en dispositivos de seguridad, dispositivos de control y dispositivos auxiliares. Todos ellos están conectados en el AGV. **[Para ver todos los esquemas eléctricos, mira en el anexo 3]**.

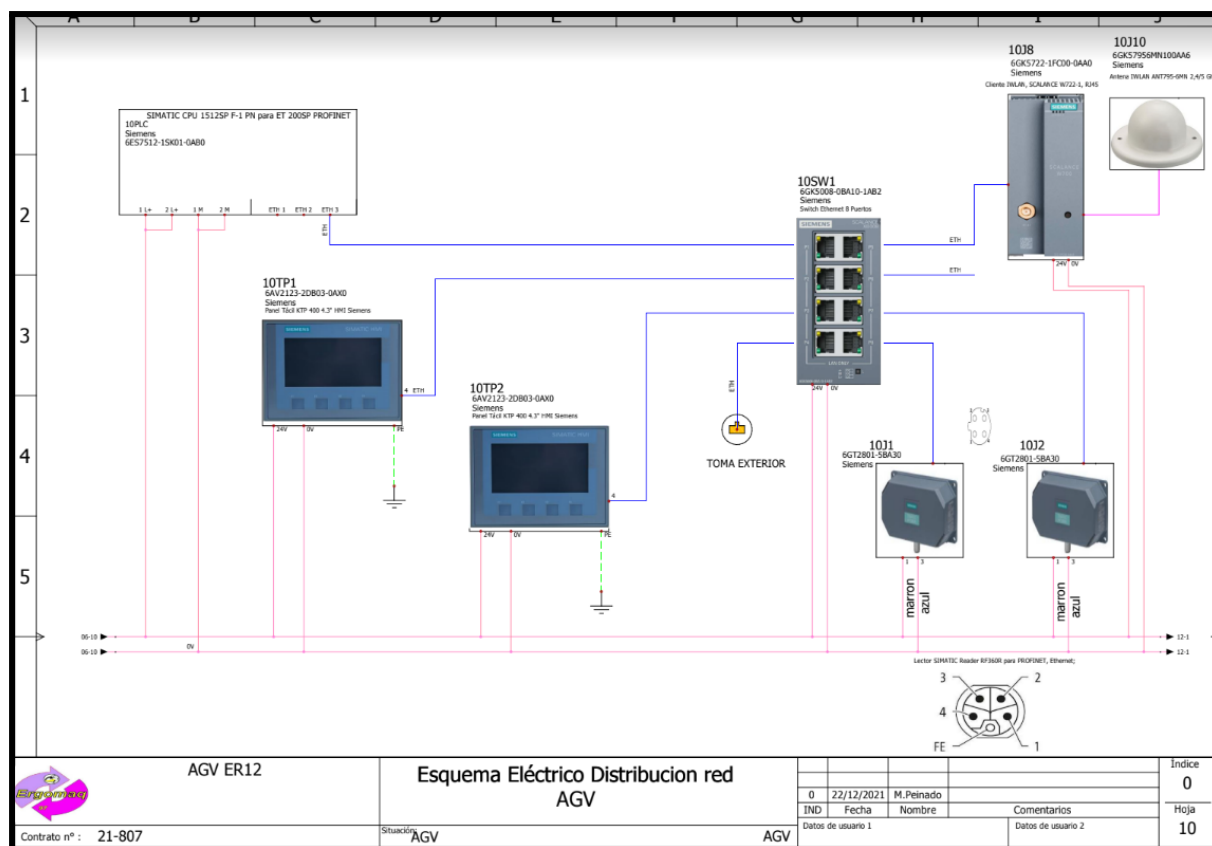


Imagen 2.2: Esquema eléctrico de la distribución de la red interna de un AGV.

Como podemos ver en la última imagen, 2.2, dentro del AGV existe una red interna. Esta red, mediante ethernet, comunicará a los diferentes dispositivos que se encargan del guardado de datos, la recepción de datos y la comunicación con el exterior.

Para el guardado de estos datos, utilizamos el dispositivo más importante del AGV y que está conectado a la red interna, es el PLC (*Programmable Logic Controller*), que lo enmarcamos dentro de los dispositivos de control. Guardamos datos en forma de variables dentro de una base de datos interna en este dispositivo.

Entre los dispositivos de recepción de datos encontramos el lector de tags *RFID* (*radio-frequency identification*), cuya función es ir leyendo los tags que se encuentra por el camino; y las pantallas *HMI* (*human-machine interface*), cuya función es realizar acciones directas en el AGV. Tanto los lectores de tags como las pantallas *HMI* envían información al PLC mediante el protocolo IP por medio de tramas ethernet.

Para la comunicación con el exterior se utilizan una *gateway* llamado “Scalance” de marca Siemens, y una antena WLAN. La gateway se encarga de transformar una trama *Ethernet* en una trama *Wi-Fi*. Utiliza un NAT para realizar este tipo de puentes entre conexiones. Con el NAT, se realiza una transformación de direcciones IP. En este caso, se trata de una NAT estática. Todas las direcciones de la red interna serán iguales para los distintos AGVs, y a la hora de la traducción o transformación, a la antena se le asignará una dirección externa. Las direcciones IP deben tenerlas las dos interfaces del gateway, que internamente tendrá siempre la 192.168.0.1 y externamente la que pongamos manualmente y que será distinta. Para poder modificar esta dirección IP, nos conectamos a la gateway mediante una toma exterior disponible en el AGV, y desde la herramienta TIA PORTAL, establecemos la conexión.

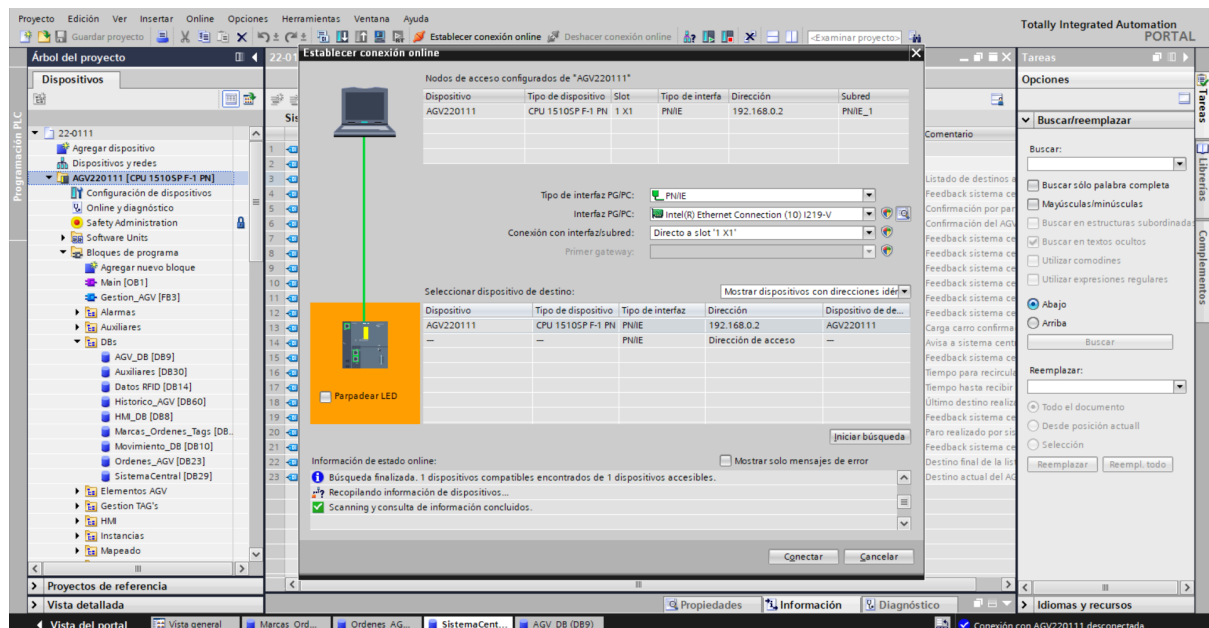


Imagen 2.3: Captura de la herramienta “TIA PORTAL” desde la cual establecemos la conexión.

Después de establecer la conexión, accedemos a la interfaz que nos proporciona SIEMENS accediendo a la dirección interna del gateway, en la cual podemos configurar todas sus opciones.



192.168.91.225/AGV_ANTENA_MRA_8



Imagen 2.4: Interfaz del gateway “SCALANCE” que nos proporciona SIEMENS.

La correspondiente antena *WLAN* sirve para la transmisión radio que permite la conexión con el exterior.

Para la comunicación interna entre los distintos dispositivos, encontramos un switch Ethernet.

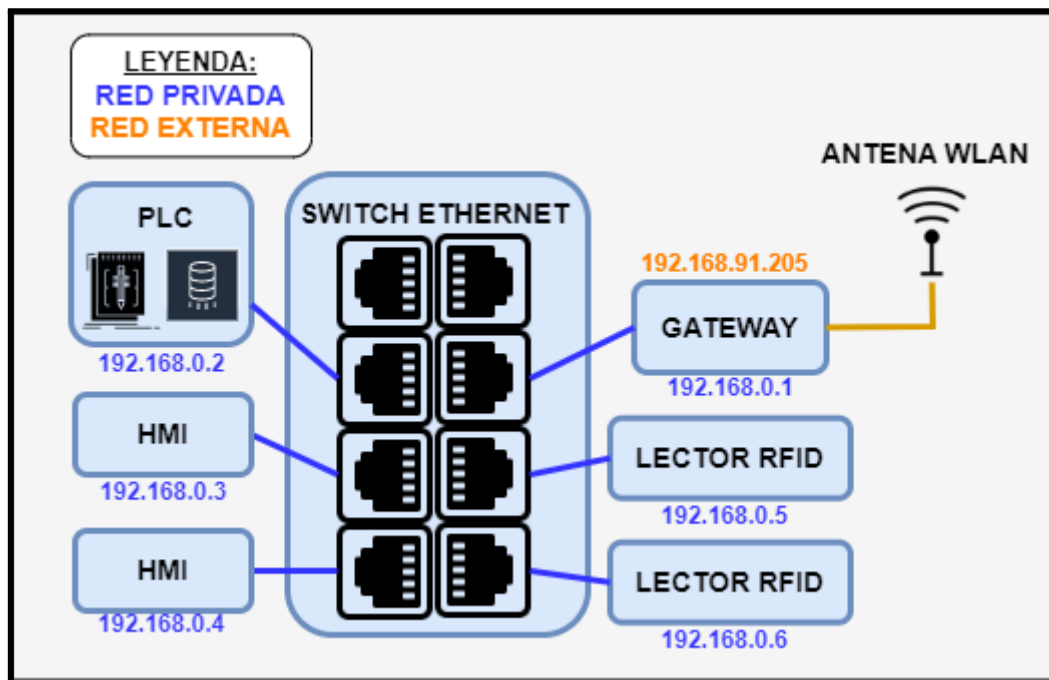


Imagen 2.5: Esquema de la comunicación interna de un AGV.

2.1.1.1. PLC (Programmable Logic Controller).

Existen distintos modelos posibles de *PLCs*. Los instalados en los dispositivos que se han probado son dos, el modelo *6ES7511-1FK02-0AB0* y el modelo *6ES7515-2AM02-0AB0*. Este *PLC* dispone de una tarjeta de memoria donde se carga previamente el programa realizado por el equipo de programadores de la empresa. Al *PLC* también se le provee una base de datos con variables. Las variables son valores que representan información sobre los componentes del AGV. Los principales valores son la velocidad a la que está circulando el AGV, el sentido de movimiento y la posición del cilindro superior.

El programa ofrece una respuesta cuando el lector *RFID* del AGV lee un tag *RFID* y le envía la información o cuando se realiza una acción desde alguna de las pantallas HMI.

2.1.1.2. Lector de tags *RFID*.

El AGV se va a ir encontrando tags por todo el circuito de la fábrica conforme vaya avanzando. Existen dos lectores *RFID* (delantero y trasero) que le permiten leer toda la información interna de los tags al pasar por encima de ellos. Una vez leída la información del tag, el lector de tags envía una trama ethernet al *PLC*, por medio del protocolo IP, con el array de bytes de la información. El programa instalado en el *PLC* interpreta esta información para realizar las acciones correspondientes de los componentes de todo el AGV como pueden ser la activación o desactivación de los motores de las ruedas o el cambio del sentido de la marcha, por ejemplo para realizar giros, cambio de velocidad, cambio de zona de escáner, etc.

2.1.1.3. Pantalla HMI.

Una pantalla *HMI* es una pantalla digital en la que se muestra información proveniente del sistema automatizado y sobre la que se puede maniobrar e indicar acciones a llevar a cabo sobre el sistema automatizado. Permite realizar las acciones manuales en el AGV, como por ejemplo subir o bajar el pin, cambiarle el destino, cambiarle el modo de funcionamiento entre manual y automático, etc.

Tanto el PLC, los lectores de tag RFID y las pantallas HMI están conectados a la misma red IP privada de cada uno de los AGVs, como se ha comentado anteriormente.

2.1.2. Circuito en el suelo.

La flota de AGVs se va a desplazar por medio de un circuito integrado en el suelo. Este circuito está formado tanto por tags, como por cintas magnéticas adhesivas.

Un tag es un objeto circular que dispone de información organizada como un array de bytes. La información de la que dispone está hecha para que sea leída por un lector que existe en los AGV, el lector *RFID*, para su posterior interpretación. La información interna de cada tag *RFID* sólo puede ser escrita por medio de una aplicación para dispositivos móviles ajena a Ergomaq, y proporcionada por el fabricante. Esta aplicación móvil utiliza la tecnología *NFC del dispositivo móvil donde se instala* para leer y escribir información en forma de array de bytes en los tags.

La información se divide principalmente en 3 campos.

- En el primer campo, se decide qué simboliza ese tag en el circuito [**Ver Anexo 2 para los tipos de información que almacenan los tags**]. Puede significar por ejemplo, que hay una bifurcación del circuito a la izquierda, que se va a informar de una solicitud de entrada a una zona comprometida, que se encuentra en un punto para cargar la batería del AGV, etc.
- En el segundo campo, se introducen los valores de las variables de salida, es decir, qué acciones va a realizar el AGV. Estas variables pueden ser por ejemplo la velocidad del AGV, el sentido de su marcha, la posición de su cilindro superior, etc.
- En el tercer campo, se introducen las condiciones que tienen que cumplirse, para que las acciones comentadas en el segundo campo procedan. Por ejemplo, la condición de que nuestra batería es mayor que el 70%, la condición de que el identificador de nuestro próximo destino es mayor que 40, etc.

Para guiar a los AGV por el entorno industrial, además de tags, se instalan unas cintas que representan el circuito por el que se van a desplazar. Los tags se colocan a la izquierda de la cinta magnética, dado que los lectores de tags del AGV se encuentran en el lado izquierdo de ambos sentidos.

Este circuito lo componen distintas zonas importantes que son necesarias para el funcionamiento completo del sistema. Estas zonas son las siguientes: almacenamiento de carritos vacíos, almacenamiento de carritos llenos, carga de baterías, y lugares donde los operarios piden los carritos para añadir el material, es decir, zonas de petición.

2.1.2.1. Zona de almacenamiento de carritos vacíos.

Esta localización es una zona donde se encuentran carritos vacíos, listos para ser solicitados por algún usuario. Esta zona debe rellenarse manualmente con nuevos carritos para no agotar el stock de los mismos. Para esta zona se elige un ramal del circuito que se indica con el correspondiente tag.

2.1.2.2. Zona de almacenamiento de carritos llenos.

En esta localización, se depositan todos los carritos llenos de materiales que traen los AGV. El contenido de estos carritos es movido a mano o con una máquina por los usuarios del entorno industrial desde esta zona hasta probablemente un camión para transportar el contenido a otra parte. Para esta zona se elige un ramal del circuito que se indica con el correspondiente tag.

2.1.2.3. Zonas de carga de baterías.

Los AGV funcionan con baterías, por lo que deben cargarse cada cierto tiempo. En cada circuito debe existir al menos una zona para la carga de baterías. Se trata de una zona dispuesta con uno o más carriles. Al final de cada carril, en el lado derecho existe una pastilla fijada al suelo que el AGV usa para recargar su batería. Esta disposición permite como máximo que un AGV esté cargando por cada carril. En cada carril, a la izquierda de la cinta magnética, existirá un tag.

En esta zona, se aplica el sistema de recirculación. Entendemos con recirculación a cuando un AGV que se encuentra en una posición de carga de batería, se desplaza hacia otro carril de la zona dejando libre la pastilla de carga. Este sistema se pone en marcha, por ejemplo, en el caso de que en un mismo carril existan varios AGV esperando a recargar su batería. De esta manera, se hace equitativa la carga entre los distintos AGVs que necesiten recargar sus baterías.



Imagen 2.6: Representación de una zona de carga de baterías con distintos carriles.

2.1.2.4. Zonas de petición.

En el circuito deben existir zonas desde donde los usuarios lanzan las peticiones a los AGVs. Estas zonas son ramales del circuito y disponen de tags.

2.1.3. Aplicación de gestión.

Esta aplicación es la pieza clave del sistema, dado que es la que gestiona los movimientos de todos los AGVs. Se encarga de interpretar todas las peticiones lanzadas por los usuarios a los AGVs y ofrecer soluciones que permitan la correcta ejecución del sistema.

La aplicación de gestión debe de estar protegida del acceso de usuarios no deseados, para así proteger a los AGVs de una función no deseada.

2.1.4. Aplicación de petición.

Esta aplicación es la que está a la vista de los usuarios, y permite a los mismos realizar peticiones, con las que la aplicación de gestión interactúa. Se han diseñado dos versiones de esta aplicación. Una versión que puede ejecutarse en una máquina *Windows*, y una versión web para navegador. La aplicación debe estar ejecutándose en equipos móviles.

2.1.5. Base de datos.

Para que se establezca la comunicación entre las dos aplicaciones, necesitamos de una base de datos que sea accesible. Se ha establecido una base de datos *SQL*, en la cual se incluyen diferentes tablas referentes al sistema completo. Las tablas de esta base de datos deben incluir, la información sobre cada tag del circuito, la información del recorrido de las cintas magnéticas, la información sobre las peticiones realizadas por los usuarios, y la información sobre los AGVs. Para mejorar la seguridad se podrá instalar un firewall de bases de datos entre la aplicación de petición (que es visible para los usuarios) y el manejador de bases de datos (que interactúa con la base de datos para guardar, modificar u obtener información).

2.2. Protocolos.

Para la comunicación entre los distintos equipos, será necesario el uso de dos protocolos.

- El protocolo *OPC UA*, que comunica a cada AGV con la aplicación de gestión.
- El protocolo *SQL*, sobre el que se realizan los accesos a la base de datos de tanto la aplicación de gestión, como de la aplicación de petición.

Estos protocolos utilizados entre los equipos funcionan sobre el protocolo *TCP/IP*. Se puede observar en la imagen de abajo, 2.5.

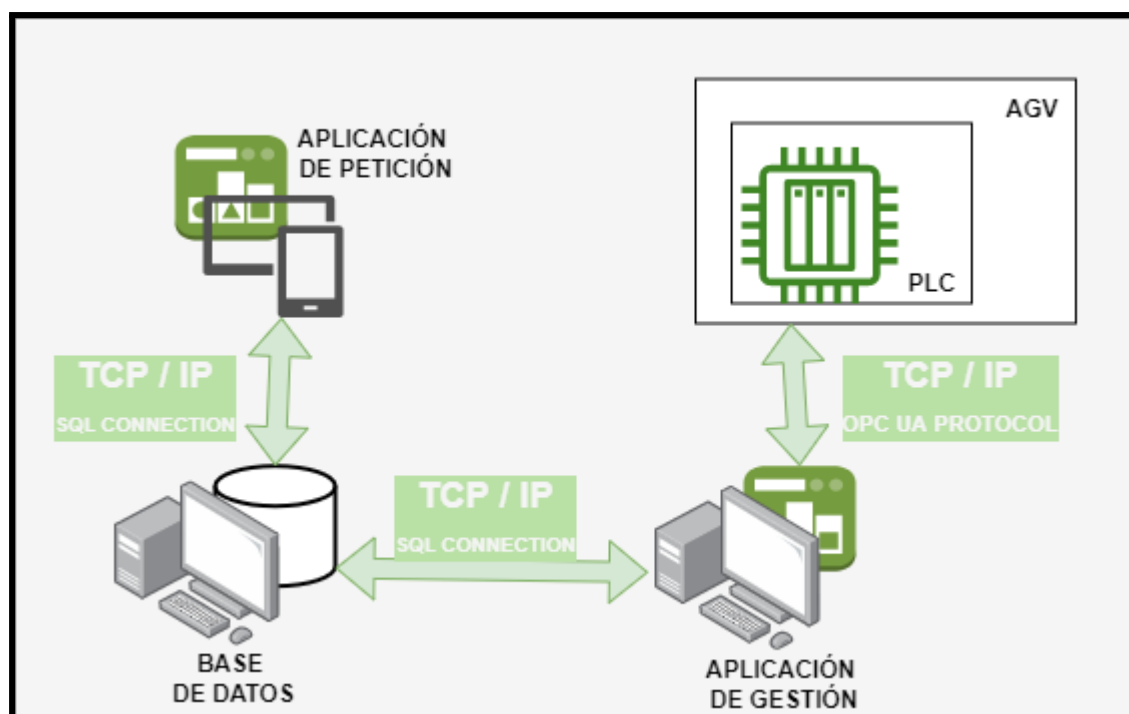


Imagen 2.7: Esquema de los protocolos utilizados en el sistema.

2.2.1. Protocolos de comunicación entre la aplicación de gestión y la flota de AGVs.

Para la comunicación de la aplicación de gestión con cada AGV, se va a utilizar un protocolo de comunicación llamado *OPC UA*, pero para comprenderlo, primero debemos hablar de su predecesor, el protocolo *OPC*. La aplicación se comunica con el PLC del AGV.

2.2.1.1. Protocolo OPC.

El protocolo *OPC* (*Open Protocol Communication*) se fundamenta en la arquitectura cliente-servidor.

OPC [1] son las siglas de *Open Protocol Communication*, y su objetivo como protocolo es entregar un estándar de comunicación abierto e independiente. Su ámbito es la industria, ya que normalmente se encuentra en software industrial tal como *SCADAs*, *Historiadores*, *drivers*, etc. La tecnología *OPC* se fundamenta en básicamente dos tecnologías.

- En primer lugar, se habla de una arquitectura **cliente-servidor**. Esto significa que en cualquier arquitectura debe haber al menos dos software independientes, un servidor que se conecte a la fuente de datos y los exponga, y un cliente que haga las consultas pertinentes para hacer uso de esos datos. Por ejemplo, se pueden encontrar servidores conectados a *PLCs*, que comuniquen los datos a un cliente como un *SCADA*.
- En segundo lugar, se habla de una plataforma de comunicaciones basada en tecnología *Windows*. *OPC* utiliza la tecnología *COM / DCOM* para llevar los paquetes de comunicación entre aplicaciones.

Un servidor *OPC* [2] reduce la carga de los dispositivos de la fuente de datos, ya que permite la comunicación con varias aplicaciones, mientras que sólo tiene una conexión con

la fuente de datos. Esto reduce el acceso múltiple desde la fuente de datos, y por lo tanto reduce la carga en la propia fuente de datos.

2.2.1.2. Protocolo *OPC UA*.

El protocolo *OPC UA (Unified Architecture)* es la evolución de la tecnología *OPC Clásica*. Es una tecnología de comunicación industrial multiplataforma, abierta, orientada a servicios, segura, y con ricos modelos de información. La tecnología es mantenida por la *OPC Foundation*.

Como el *OPC Clásico* [3], se trata de un protocolo de comunicación pensado para comunicar datos de equipos industriales, pero su principal diferencia es que, a diferencia de éste, no se limita sólo a comunicar datos entre aplicaciones *SCADA* y sensores, sino que su objetivo es ir más allá y que pueda comunicarse con todas las aplicaciones de la empresa y a través de todas las capas empresariales.

Todos los mensajes *OPC UA* entre clientes y servidores se entregan a través de una conexión **TCP / IP**. Dentro de *TCP*, hay una “sesión”. Hay dos tipos de estas sesiones que envían mensajes a través de *TCP*, y que se denominan transportes cuando se usa *OPC UA*. Son *OPC UA TCP* y *SOAP / HTTP*.

El transporte *OPC UA TCP* es esencialmente un pequeño protocolo que establece un canal de comunicación de bajo nivel entre un cliente y un servidor. Lo que hace principalmente es negociar los tamaños máximos de búfer. La ventaja del mismo es su tamaño y su impacto insignificante en el rendimiento.

Las especificaciones de *OPC UA* son: la vista general y conceptos, modelo de seguridad, modelo de espacio de direcciones, modelo de información, asignaciones, perfiles, acceso a datos, alarmas y condiciones, programas, acceso histórico, descubrimiento y servicios globales, agregados, publicar y suscribir, seguridad y nombres de alias.

Un nodo es la unidad básica de información en el espacio de direcciones *OPC UA*. Proporciona una forma estándar para que los servidores *OPC UA* representen objetos para los clientes. Los nodos son fragmentos de información y consisten en atributos, el valor real de los datos y una o más referencias a otros nodos, cada uno en su propio espacio de direcciones. Los nodos se referencian a través de un *ID* de nodo único: un URI(identificador de recurso único) de espacio de nombres, un tipo de datos y el identificador en sí.

[Para ver las ventajas de *OPC UA* con respecto a *OPC* clásico, anexo 7]

Para resumir, cada *PLC* tiene incluido un servidor *OPC UA* a una dirección *IP* y un puerto. Se crea una sesión con esa dirección y puerto, y de esta manera se puede conectar, leer y escribir variables dentro del *PLC*. El puerto de esta sesión será el 4840. En cada uno de los AGVs, la dirección *OPC UA* es la misma que la del *PLC*. En este caso será 192.168.0.2. Mediante el NAT, podremos acceder a dicha dirección *IP* interna desde el exterior, por medio de la dirección *IP* externa proporcionada por el bridge, diferente en cada uno de los AGVs del sistema.

2.2.2. Protocolos de comunicación entre la aplicación de gestión y la base de datos.

Por seguridad, para crear conexiones entre la aplicación de gestión y la base de datos, deben de estar en la misma red *IP*. Para establecer dicha conexión, debemos establecer la dirección *IP* del servidor, el nombre de la base de datos, un nombre de usuario

y una contraseña. La aplicación de petición también se conecta con la base de datos de la misma manera.

El protocolo de red utilizado en SQL Server es el protocolo *TCP / IP*. Este protocolo conecta equipos con diferentes especificaciones de hardware y sistemas operativos y les permite comunicarse entre sí. Incluye estándares de enrutamiento de tráfico de red con protección de seguridad avanzada. De manera predeterminada, la instancia de SQL Server escucha en el puerto *TCP* 1433.

2.2.3. Protocolos entre la aplicación de gestión y cada aplicación de petición.

La aplicación de gestión se comunica con la aplicación de petición por medio de la base de datos, de tal manera que cuando se solicita una petición, se queda registrada en la base de datos, y en cierto tiempo, la aplicación de gestión leerá dicha información para realizar acciones oportunas con ella.

2.3. Herramientas de desarrollo.

2.3.1. Lenguajes de programación.

El lenguaje de programación **C#** [5] es un lenguaje de programación orientado a objetos orientado a componentes. Proporciona construcciones de lenguaje para admitir directamente estos conceptos, por lo que se trata de un lenguaje natural en el que crear y usar componentes software. Este lenguaje se ha utilizado para la programación de la aplicación de gestión y para la aplicación de peticiones.

El lenguaje de programación **JavaScript** [6] es un lenguaje de programación interpretado, orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico. Se utiliza principalmente del lado del cliente, implementando como parte de navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas. Este lenguaje se ha utilizado en la programación del *front-end* de la aplicación de peticiones versión web.

2.3.2. Aplicaciones.

2.3.2.1. Aplicaciones utilizadas por el equipo de desarrolladores de Ergomaq.

Los programas del *PLC*, realizados por el equipo de desarrolladores de la empresa y que se encarga de proveer al *PLC* de una base de datos con variables, se cargan utilizando la herramienta **TIA Portal V16**.

TIA Portal (Totally Integrated Automation) [7] es la plataforma de ingeniería de *Siemens* que ofrece soluciones de automatización en multitud de sectores industriales. Un software que integra todos los componentes de las máquinas para controlar procedimientos y operaciones. Al ser una aplicación modular, es posible añadir nuevas funcionalidades que se adapten a las necesidades de la aplicación. *TIA Portal* incluye: lenguaje de programación para *PLCs* y gateways Scalance de *Siemens*, software para control de periféricos, software *WINCC* para la visualización de procesos y dispositivos, y mucho más. La versión 16 incluye la nueva interfaz de control de versiones para sistemas de control de versiones

externas. La importación y exportación de objetos de software a través de la interfaz de *TIA Portal* permite que las versiones se integren sin problemas en el proceso de desarrollo.

Además del programa, se carga con ella la base de datos interna del PLC, y se inicializa el servidor *OPC UA* para establecer la comunicación con el mismo. Con la herramienta también se pueden realizar simulaciones, en las que se pueden modificar directamente los valores de las variables incluidas en el PLC para poder realizar testeos.

2.3.2.2. Aplicaciones, entornos y herramientas utilizadas en el desarrollo de las aplicaciones.

Para realizar las primeras aproximaciones a la aplicación, y la aplicación final de escritorio, se ha utilizado la herramienta **Microsoft Visual Studio 2022** [8] en el entorno de Windows. *Microsoft Visual Studio* es un *IDE* (entorno integrado de desarrollo) que permite desarrollar aplicaciones, sitios web, aplicaciones *WPF* (*Windows Presentation Foundation*), aplicaciones web, servicios web, apps de *Windows Store*, etc. para poder ser ejecutadas en todas las plataformas de *Microsoft Windows*, *Windows Mobile*, *Windows CE*, *.NET Framework*. Permite emplear múltiples lenguajes de programación. La aplicación de escritorio consiste en una aplicación de *Windows Forms*. El lenguaje de programación utilizado es *C#*. *Windows Forms* (o formularios Windows) es el nombre dado a la interfaz de programación de aplicación gráfica (*API*) que se incluye como parte de *Microsoft .NET Framework*, que proporciona acceso a los elementos de la interfaz de *Microsoft Windows* nativas envolviendo la *API* de Windows existente en código administrado.

Para realizar la aplicación de petición versión web, se ha iniciado un proyecto nuevo en **Microsoft Visual Studio**, el proyecto será del tipo Sitio de *ASP.NET Web Forms*.

ASP.NET [9] es un entorno para aplicaciones web desarrollado y comercializado por Microsoft. Se pueden construir sitios web dinámicos, aplicaciones web y servicios web. Las páginas de *ASP.NET*, se conocen como “*Web Forms*” (formularios web), y son el principal medio de construcción para el desarrollo de aplicaciones web. Los formularios web están contenidos en archivos con una extensión *ASPX*. Modelo *Code-behind*: coloca código en un archivo separado. Los nombres de los archivos code-behind están basados en el nombre del archivo *ASPX*, tales como “*MiPagina.aspx.cs*”. *Web Forms* son páginas que los usuarios solicitan mediante su explorador. Estas páginas se pueden escribir mediante una combinación de *HTML*, script de cliente, controles de servidor y código de servidor.

Una herramienta utilizada esencial es el mecanismo a través del cual los desarrolladores pueden crear, compartir y consumir código útil. A menudo, se integra en “paquetes” que contienen código compilado. En *.NET*, el mecanismo compatible para compartir código es **NuGet** [10], que define cómo se crean, hospedan y consumen paquetes en *.NET*. Desde un punto de vista sencillo, un paquete *NuGet* es un archivo *ZIP* con la extensión *.nupkg* que contiene código compilado. Para poder utilizar funciones del protocolo *OPC UA*, se utiliza el paquete *OPC Foundation*. Existen más paquetes para las conexiones *SQL*, e incluso paquetes de diseño que utilizan por defecto *Windows Forms* y *Web Forms*.

2.3.2.3. Aplicaciones utilizadas en el desarrollo de la base de datos.

Para gestionar la base de datos se ha utilizado la herramienta **Microsoft SQL Server Management Studio** [11], que es la interfaz de usuario cliente preferida y oficial con la cual se puede manejar, configurar, desplegar, actualizar y administrar una instancia **SQL Server**. La herramienta es amigable para el usuario y bastante exhaustiva. *Microsoft SQL*

Server es un sistema de gestión de base de datos relacional desarrollado como un producto de software con la función principal de almacenar y recuperar datos según lo solicitado por otras aplicaciones.

Capítulo 3.

Sistema desarrollado.

Para llevar a cabo todo el sistema, se han desarrollado distintas aplicaciones que hacen que el sistema completo funcione correctamente. Existen dos aplicaciones esenciales. Por una parte, la **aplicación de gestión**, y por otra, la **aplicación de petición**.

La *aplicación de gestión* es la que va a estar ejecutándose en el sistema continuamente, va a ir comprobando las variables de la base de datos de los AGVs involucrados en el circuito. Esta aplicación debe ejecutarse en una máquina conectada a la red LAN. Lo ideal es que sólo se ejecute una aplicación de este tipo por circuito al mismo tiempo.

La *aplicación de petición* es con la que van a tener que operar los usuarios de la fábrica en sus dispositivos. Al contrario que la aplicación de gestión, pueden existir varias aplicaciones ejecutándose al mismo tiempo, por un mismo circuito, dado que cada persona va a tener que lanzar peticiones desde una posición distinta del circuito. Esta aplicación tiene dos funcionalidades: la posibilidad de lanzar peticiones, como se ha comentado previamente, y la posibilidad de crear y editar circuitos. La aplicación de petición tiene dos versiones: una versión de Windows Forms (formularios de Windows) y una versión web. Ambas versiones pueden convivir en el mismo escenario, dado que no hay conflictos entre ellas. La funcionalidad de crear y editar circuitos sólo está disponible en la versión de Windows Forms.

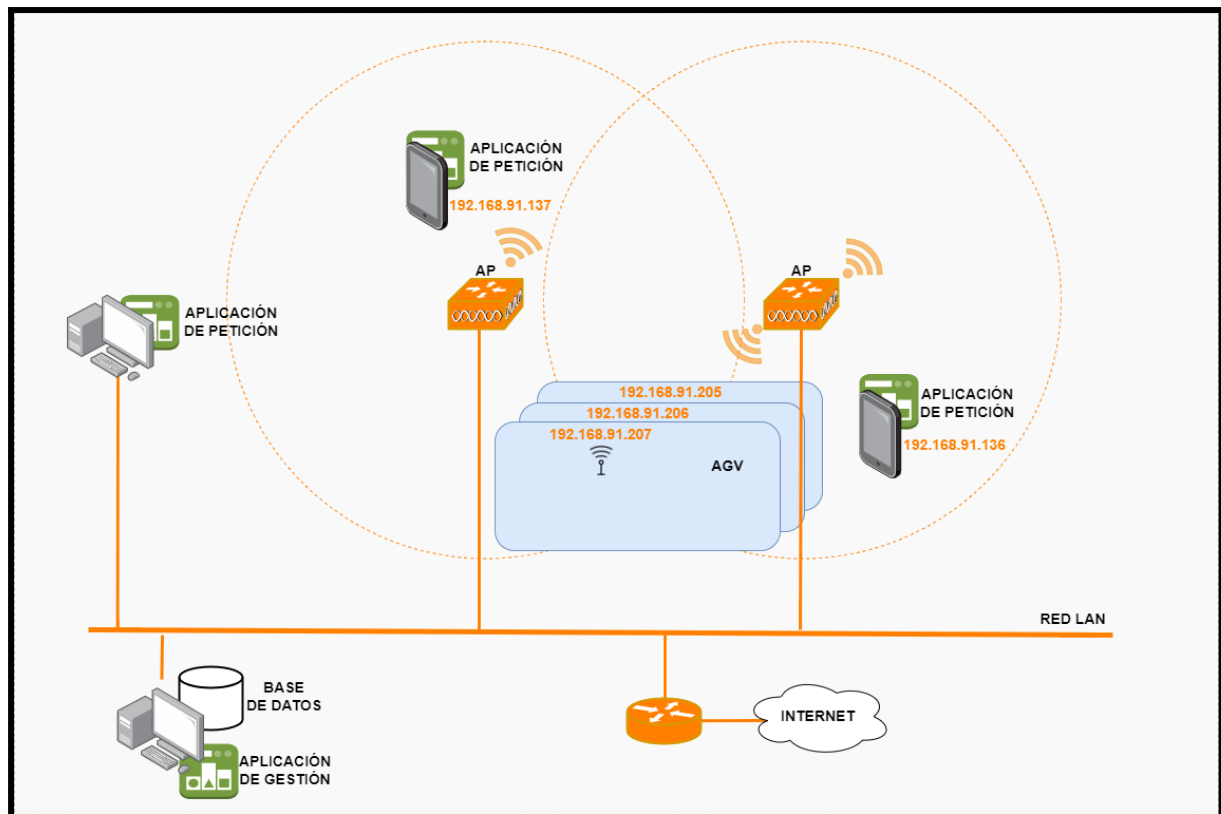


Imagen 3.1: Distribución de las dos aplicaciones.

En la etapa inicial del desarrollo de ambas aplicaciones, se comenzó creando una **aplicación base**, que unía estas dos aplicaciones en una. De esta manera, el sistema, en lugar de considerarse una aplicación multiusuario, se consideraba una aplicación para un único usuario. Esta decisión se tomó teniendo en cuenta que era más ventajoso empezar a desarrollar el sistema con una aplicación base, para facilitar la programación. En algunas partes del capítulo, se hará referencia a esta aplicación base, en forma de primeras aproximaciones de la implementación del software.

Disponemos de un esquema que nos define todos los puntos que se van a tratar en el capítulo 3.

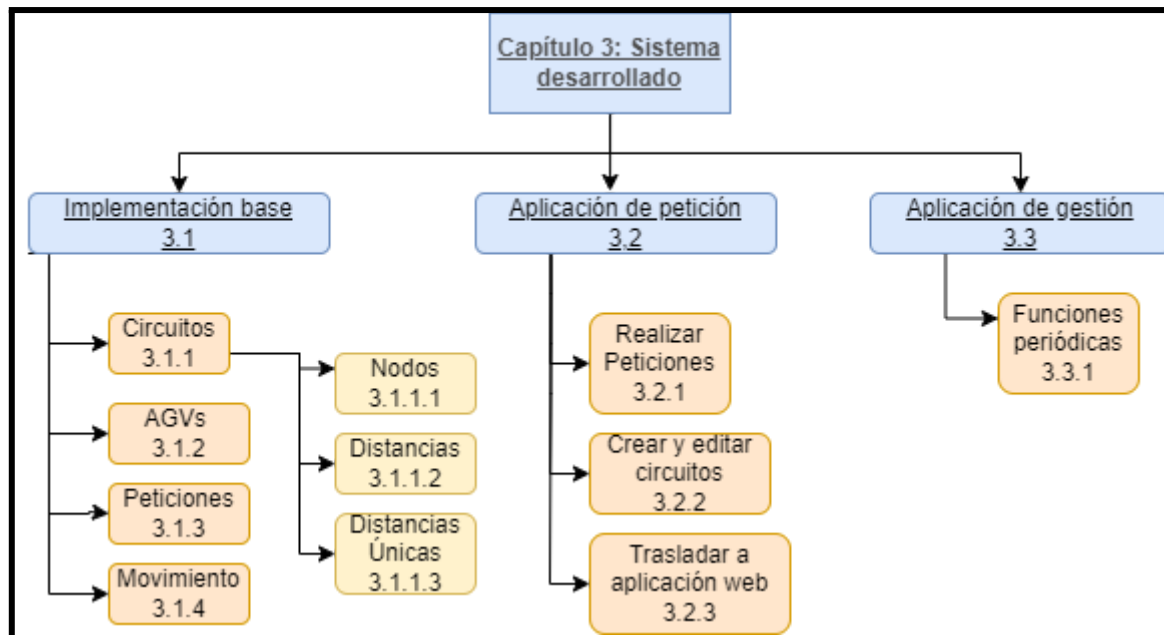


Imagen 3.2: Esquema capítulo 3.

→**3.1.** Se explica la implementación de las clases y funciones que forman los puntos clave en la implementación de las aplicaciones. Estos puntos clave son la base de las dos aplicaciones. En los apartados 3.1.1., 3.1.2. y 3.1.3. se explican las clases que se han implementado en el sistema. En los apartados 3.1.4. se explican las funciones y algoritmos que permiten a los AGVs moverse por el sistema.

→**3.2.** Se explica la implementación de la aplicación de petición, aplicación que utilizan los usuarios para lanzar peticiones. Esta aplicación dispone de dos partes. En el apartado 3.2.1. se habla de la parte de la aplicación de petición que trata sobre el lanzamiento de las peticiones. En el apartado 3.2.2. se habla de la parte de la aplicación de petición en la que se crean y editan los circuitos. En el apartado 3.2.3. se explica cómo se ha trasladado la aplicación de petición desde una aplicación de Windows Forms a una aplicación web.

→**3.3.** Se explica la implementación de la aplicación de gestión, aplicación que corre en paralelo de las aplicaciones de petición, y que se encarga por medio de funciones periódicas de que el funcionamiento del sistema sea el correcto.

3.1. Implementación base.

La base de nuestras aplicaciones consiste en la implementación software de los conceptos: circuitos de AGVs, los propios AGVs, las peticiones de AGVs, y el movimiento de los AGVs por el circuito. Estos conceptos son puntos clave que se presentan en las dos aplicaciones, de tal manera que son la base del funcionamiento del sistema.

Para la implementación software de los tres primeros conceptos se han hecho uso tanto de clases dentro de las aplicaciones como de tablas en una base de datos. En cuanto al movimiento de los AGVs por el circuito, se utilizan funciones que se basan en estas clases y tablas.

Las clases se han diseñado en el lenguaje de programación C#. Es un lenguaje orientado a objetos, y estas clases son la representación simbólica de los objetos. Las

clases describen campos, propiedades, métodos y eventos. Para cada clase de la aplicación en C#, se ha creado una tabla en la base de datos SQL.

La idea principal es que las aplicaciones lean y escriban información en la base de datos, e incluyan estos datos en las clases internas de las aplicaciones para hacer uso de los diferentes métodos de cada clase, y además para proporcionar una programación más visual, correcta y limpia.

Se procede a explicar la implementación software de los puntos clave uno por uno.

3.1.1. Implementación del circuito.

¿Cuál es la mejor manera de implementar en el software un circuito como éste? Para poder llegar a esta conclusión, será necesario repasar cuáles son los puntos clave del circuito, y sacar una conclusión.

El circuito está compuesto por una cinta magnética, tags y pastillas de carga. Cada vez que existe una situación especial en el circuito, existe un tag. Es decir, cuando encontramos un cruce, previamente existirá un tag que el lector de tags RFID del AGV va a leer, y enviará la información al programa interno del PLC, el cual hará que el AGV se desvíe hacia la bifurcación, o continúe recto. Además de cruces, también existen zonas de carga de baterías, con pastillas de carga de batería instaladas en el suelo donde además existirá un tag, para indicar al programa interno del PLC que existe una pastilla de carga de batería. En cada una de las zonas de petición, existirá un tag, desde donde los usuarios realizan peticiones, y donde llegan y se estacionan los carritos, para poder cargar materiales en ellos. Existen varios tags de bloqueo y desbloqueo a lo largo del circuito, los cuales delimitan zonas del circuito en exclusión mutua a la que sólo puede acceder un AGV al mismo tiempo.

Como podemos observar, todas las zonas importantes del circuito, están relacionadas con los tags, por lo que jugarán un papel muy importante a la hora de implementar el circuito en el software.

Se ha decidido, tras la reflexión previa, que para representar el circuito en la aplicación, y almacenar los datos del mismo en la base de datos, se va a utilizar un **plano bidimensional**. Existirán objetos bidimensionales que van a poderse situar perpendicularmente sobre el plano. Estos objetos bidimensionales representan cada tag del circuito.

En un circuito real, lo que físicamente existe entre 2 tags, es una porción de cinta magnética. Por lo tanto, para poder representar esta cinta en dicho plano, se unen los objetos bidimensionales mediante una línea recta. En la imagen de abajo podemos observar una comparación entre un dibujo que simula un circuito real, y la aproximación en el plano de la implementación software que se corresponde con dicho circuito.

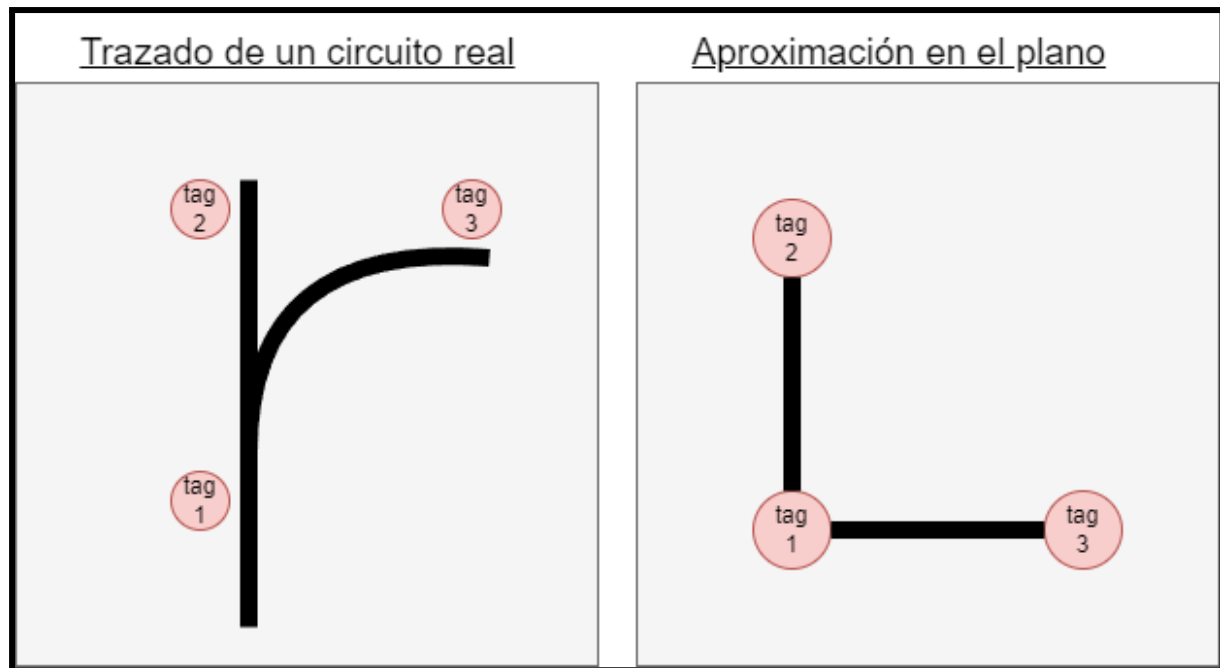


Imagen 3.1.1: Comparación entre la realidad y la implementación del plano bidimensional.

Esta implementación de los objetos bidimensionales y líneas en el plano nos recuerda a un **grafo**. Un grafo es un conjunto de objetos llamados vértices o nodos unidos por enlaces llamados aristas o arcos, que permiten representar relaciones binarias entre elementos de un conjunto.

Un grafo G [12] es un par ordenado $G = (V, E)$, donde:

- V es un conjunto de vértices o nodos.
- E es un conjunto de aristas o arcos, que relacionan estos nodos.

Sin embargo, este grafo es especial, dado que es un grafo que existe en un plano bidimensional, y por tanto, las líneas únicamente pueden dibujarse perpendiculares en el plano. Por lo que un nodo de este grafo en específico como máximo puede estar relacionado con 4 aristas, y como mínimo con 1 arista.

Este tipo de implementación, como se ha comentado, permite a un nodo relacionarse con otros 4 nodos como máximo, lo que en la realidad significa un tag que implica una triple bifurcación en el circuito. En el sistema real, esta posibilidad no existe, dado que por el momento los AGV no están programados para poder tomar decisiones entre 3 bifurcaciones, pero es un incentivo para realizar una mejora, dando así lugar en el futuro a circuitos con rutas más complejas y por lo tanto obtener la posibilidad de optimizar el espacio físico que ocupa cada circuito completo.

3.1.1.1. Primera aproximación de la implementación software del circuito.

En la implementación del circuito en nuestro sistema, a los objetos bidimensionales que representan los tags los llamaremos **tags**. A la recta que une dos tags, y que simula el recorrido de la cinta magnética que existe entre 2 tags que se leen seguidos en el circuito la llamaremos **distancia**.

Debajo, en la imagen, observamos una representación de los tags y distancias en un circuito de AGVs en la aplicación.

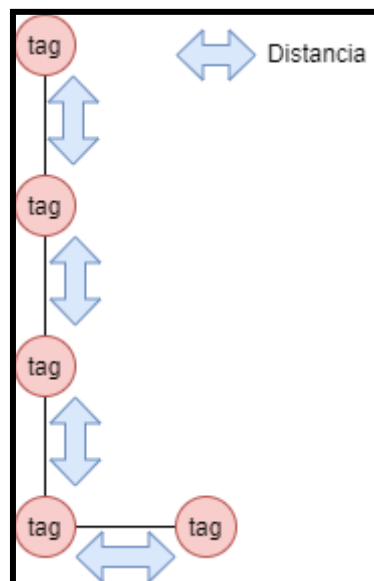


Imagen 3.1.2: Representación de los puntos y distancias en el circuito.

Cada **tag** tendrá asociadas unas coordenadas X e Y del plano, y cada **distancia** tendrá asociadas las dos coordenadas de los dos tags a los que une. Cada distancia también tiene asociado un número entero que representa la distancia real en metros de la cinta magnética que une a los dos puntos.

Dado que vamos a almacenar las distancias como una tupla de dos coordenadas, esto permite que una misma distancia pueda ser representada de dos maneras diferentes. Ejemplo: La distancia que une a los puntos P1 y P2 puede almacenarse como (P1,P2) y (P2,P1). Este aspecto nos recuerda a un grafo no dirigido, el cual ofrece un matiz que nos recuerda más a estas situaciones.

Un **grafo no dirigido** [12] o grafo propiamente dicho es un grafo $G = (V, E)$, donde:

- $V \neq \phi$
- $E \subseteq \{x \in P(V) : |x| = 2\}$ Es un conjunto de pares no ordenados de elementos de V.

Un par no ordenado es un conjunto de la forma {a,b}, de manera que {a,b} = {b,a}. Para los grafos, estos conjuntos pertenecen al conjunto de potencia de V, denotado P(V), y son de cardinalidad 2.

3.1.1.2. Implementación software final del circuito.

Para la segunda y última aproximación de la implementación software, se ha partido de la primera aproximación, y se han corregido varios problemas.

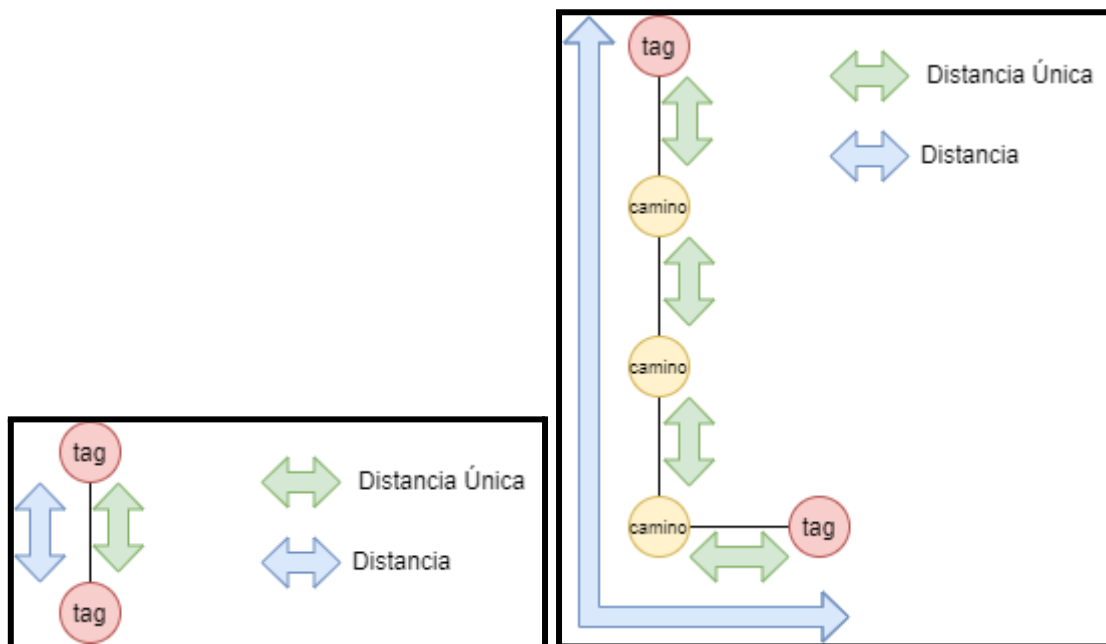
El gran problema de la anterior implementación, es que debido a que los tag sólo pueden relacionarse entre ellos mediante líneas perpendiculares en el plano, existen distintos circuitos que son imposibles de ser implementados, dada la complejidad de ciertos giros en dicho circuito.

Por ello, la solución aportada ante dicho problema, se basa en que ahora no todos los nodos del circuito, es decir, los objetos bidimensionales del plano, representan **tags**. Un nodo del circuito en la nueva implementación software puede representar un **tag** o un **camino**. Un camino representa una prolongación de la cinta magnética que existe entre dos

tags en el circuito. Por lo tanto, las razones de la adición de este tipo de nodos son, por una parte, para que podamos implementar todos los circuitos posibles, y por otra, para que visualmente el grafo que vemos se asemeje más a la realidad.

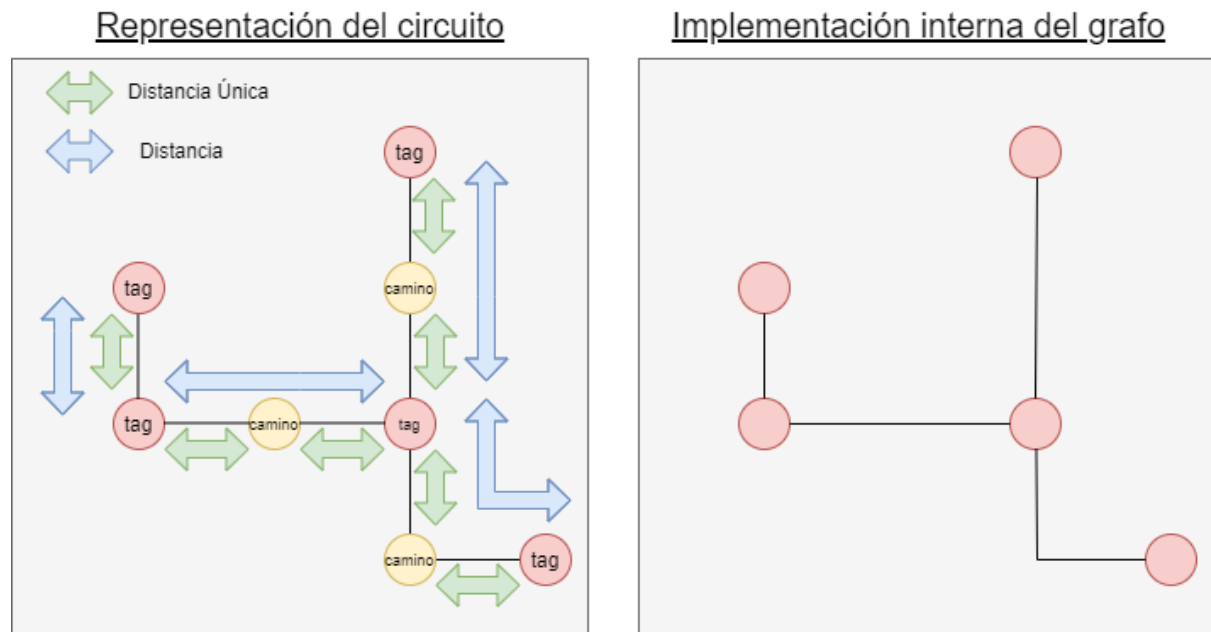
Cuando dos nodos en el plano se unen, ya no lo llamaremos distancia, sino que nos referiremos a ello como **distancia única**. El concepto de **distancia** representa el mismo que antes, el recorrido que hay desde un tag hasta otro en el circuito, pero, con respecto a la implementación software, una distancia es ahora la unión de una o más distancias únicas.

De esta manera, se puede representar mejor un circuito en el plano de dos dimensiones.



Imágenes 3.1.3 y 3.1.4: Representación de los puntos, distancias y distancias únicas en el circuito.

Dado que ahora no necesitamos todos los nodos para poder calcular las distancias entre los tags, necesitamos dos implementaciones del circuito. Una implementación software que nos permita observar todos los nodos, y otra implementación que sea la implementación interna del grafo. En la implementación interna del grafo, no se van a tener en cuenta los nodos camino. En la imagen de abajo tenemos una comparación entre la representación en el circuito, y la información sobre el grafo que almacenamos.



Imágenes 3.1.5 y 3.1.6: Comparación entre los datos guardados para la representación del circuito, y los datos en la implementación interna del grafo.

Para esta implementación, se han creado las clases: **Nodo**, **Distancia** y **Distancia Única**.

Uno de los campos que debe almacenar cada elemento en cada clase, es el **identificador interno artificial del circuito** en el que se encuentran. Este identificador es un número entero generado artificialmente y que es único por cada circuito. De esta manera podremos disponer de diferentes circuitos en una misma red.

3.1.1.2.1. Implementación de la clase **Nodo**.

Esta clase guarda la información referente a cada uno de los nodos que existen en el grafo. Cada uno de los nodos se distingue de los otros por un campo que representa su **identificador interno**. Este identificador es un número entero generado artificialmente y es único en cada elemento nodo de cada circuito. Otros dos campos que se guardan corresponden con las **coordenadas X e Y** en la que se encuentra cada nodo en el plano bidimensional. Son números enteros. Además del identificador interno, se guardan otros dos identificadores: el **"tag ID"**, y el **"número real"**. Para comprender el funcionamiento del sistema, es necesario comprender bien estos dos conceptos. Ambas son variables que almacenan números enteros.

El **"tag ID"** representa el identificador por el que los usuarios del sistema conocen al nodo. A diferencia del identificador interno, este sí es visible para los usuarios, y no se genera artificialmente. Los nodos en los que se va a usar este campo son los tags con información de petición. Es decir, en cada zona de peticiones existirá un tag con dicho campo. De esta manera, los usuarios del entorno podrán diferenciar entre las distintas zonas de peticiones en el circuito.

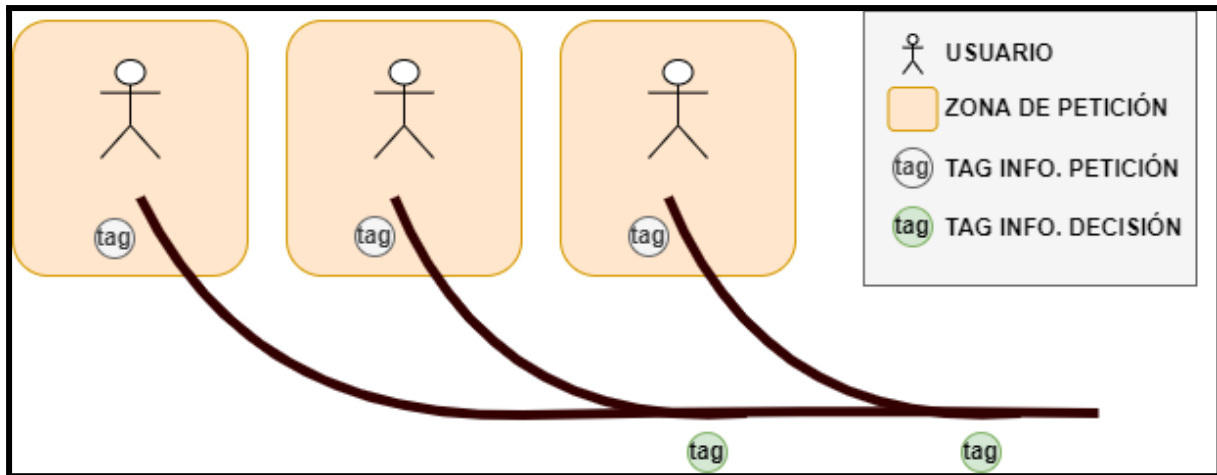


Imagen 3.1.7: Diagrama de un circuito y 3 zonas de petición.

El “**número real**”, se corresponde con el número interno que lee el PLC del AGV cuando pasa por encima de un tag. Por lo tanto, este número no está a la vista del usuario, sino que se corresponde con un número necesario para la lectura de los tags por parte de los lectores de tags RFID instalados en el vehículo. Por tanto, los usuarios del sistema podrán visualizar el “**tag ID**”, pero no el “**número real**” de cada tag.

Para poder realizar la distinción entre los **distintos tipos de información que puede almacenar un tag** [Para ver los tipos de información de los tags, anexo 2], en la clase nodo se guardan campos que corresponden a una serie de booleanos que nos lo indican: si es un tag de decisión, carga de baterías, petición, bloqueo, desbloqueo, almacén de carritos llenos, almacén de carritos vacíos. Además, para distinguir entre un nodo tag o un nodo camino, la clase también dispondrá de un booleano que indicará si se trata de un camino.

Para varios de estos tipos de información, existen varios campos extra que ayudan a la implementación, como por ejemplo cuando es un nodo que representa un lugar de carga de batería, guardamos el número de AGVs que se encuentran esperando en ese mismo momento esperando o cargando su batería en ese carril en la zona de carga de baterías. O por ejemplo, un campo que nos dice si existe un carrito o no en una zona de petición. O como para los tags de bloqueo y desbloqueo, almacenar un identificador que los relacione, además de un campo que avisa si hay un AGV en esa zona de acceso en exclusión mutua. Nos centraremos en estos campos extra más adelante, cuando se comience a especificar su funcionamiento en el sistema completo.

3.1.1.2.2. Implementación de la clase Distancia.

La clase distancia es la unión entre dos nodos tag en el circuito. Esta unión puede ser directa, o pueden existir distintos nodos camino entre los nodos tag, que sirvan como puente entre los dos tags. Para poder distinguir entre cada una de estas distancias, disponen de un **identificador interno único artificial** que aumenta automáticamente. Una distancia alberga información de una **lista de referencias a distancias únicas**, una **lista de referencias a puntos PLC**, dos **referencias** de los nodos que representan los dos tags que conectan la distancia, así como el **sentido** en el que la referencian (izquierda, derecha,

arriba o abajo), y un entero que representa la **medida de distancia real** (en metros) que existe entre los dos tags.

3.1.1.2.3. Implementación de la clase *Distancia Única*.

Una distancia única es la unión entre dos nodos en el circuito implementado. Para poder distinguir entre cada una de estas distancias únicas, disponen de un **identificador interno único artificial** que aumenta automáticamente. Únicamente disponen de 3 campos clave: la **referencia** a los dos nodos a los que une, y el **identificador de la distancia** a la que pertenece.

3.1.1.2.4. Ampliación de la implementación de la clase *Nodo*.

Sería de gran ayuda tener información sobre el grafo completo dentro de la clase nodo, y de esta manera, los cálculos serán mucho más rápidos. Para ello ampliamos la clase nodo y añadimos varios campos adicionales. Se añaden 4 campos que referencian a los nodos relacionados a las aristas existentes. Debajo observamos un ejemplo de la situación anterior y cómo se almacenarán estos 4 campos internamente para formar el grafo:

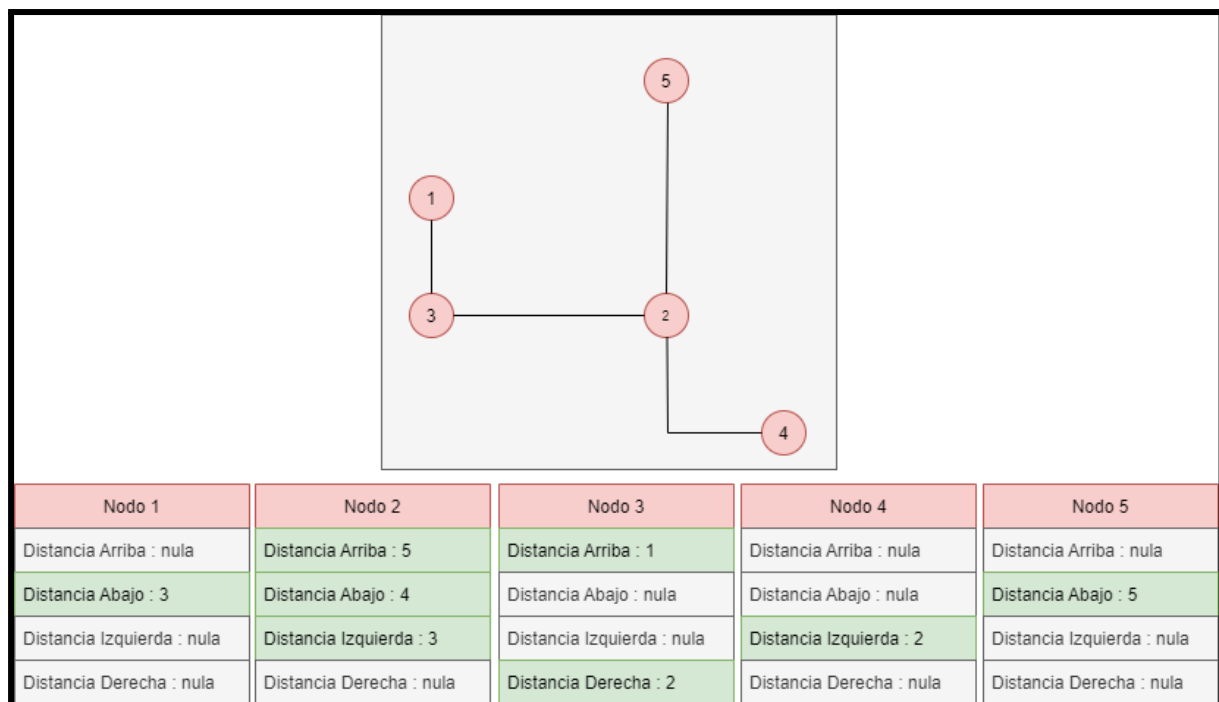


Imagen 3.1.8: Información que se almacena en los campos *Distancia* de la clase *Nodo*.

Se deben establecer 4 campos adicionales que referencian a la distancia (en metros) real que existe entre los dos nodos de tag.

Lo que estamos consiguiendo al implementar estos nuevos campos, es prescindir de las otras clases a la hora de realizar cálculos, y que todo sea mucho más sencillo y rápido, ya que estos 8 últimos campos mencionados corresponden con campos propios de la clase *Distancia*.

Dado que los nodos almacenan información de sus adyacentes, la idea de grafo no dirigido mencionada anteriormente se nos queda obsoleta.

Un **grafo dirigido** [12] o digrafo es un grafo $G = (V, E)$, donde:

- $V \neq \emptyset$
- $E \subseteq \{(a, b) \in V \times V : a \neq b\}$ Es un conjunto de pares ordenados de elementos de V .

Dada una arista (a, b) , a es su nodo inicial, y b su nodo final.

Por definición, los grafos dirigidos no contienen bucles. Sin embargo, la implementación de los circuitos sí que puede contenerlos. Se concluye entonces que esta implementación no corresponde con las dos propuestas clásicas de grafos. Es por ello que hablaremos de un grafo mixto.

Un **grafo mixto** [12] es aquel que se define con la capacidad de poder contener aristas dirigidas y no dirigidas. Tanto los grafos dirigidos como los no dirigidos son casos particulares de este.

En este caso, V o E pueden ser un multiconjunto, pudiendo haber más de una arista entre cada par de vértices. La palabra *grafo* (a secas) puede permitir o no múltiples aristas entre cada par de vértices.

Si se quiere remarcar la inexistencia de múltiples aristas entre cada par de vértices (y en el caso no dirigido, excluir bucles) el grafo puede llamarse **simple**.

Por otra parte, si se quiere asegurar la posibilidad de permitir múltiples aristas, el grafo puede llamarse **multigrafo** (a veces se utiliza el término pseudografo para indicar que se permiten tanto bucles como múltiples aristas entre cada par de vértices).

En nuestra implementación, necesitamos que nuestro grafo corresponda con un multigrafo que tenga bucles, y que sea dirigido. En muchos de los sitios del circuito, los AGVs circularán en los dos sentidos, pero hay en ciertas zonas donde debe ser dirigido, dado que será útil que sólo exista un sentido de circulación, como en la zona de recirculación.

Por lo tanto, podemos asemejar la nueva implementación de los nodos a un **multigrafo con aristas dirigidas**.

Un **multigrafo o grafo multivariado** [13] es una generalización de un grafo que permite aristas múltiples, o equivalentemente, más de un conjunto de aristas. De esta forma, dos nodos pueden estar conectados por más de una arista. Se permite que los multigrafos tengan bucles, es decir, que una arista conecte a un nodo consigo mismo. Un pseudografo se puede definir como un sinónimo de multigrafo, aunque en ocasiones también se utiliza para distinguir a los multigrafos en general, de aquellos que permiten bucles. Si se consideran aristas dirigidas, al multigrafo también se le conoce como **multigrafo dirigido o multidigrafo**. También se puede hablar de grafo complejo en oposición a un grafo simple (esto es, un grafo sin bucles ni aristas múltiples), como un grafo que posee bucles y/o al menos un par de vértices con más de una arista.

En la imagen de abajo podemos observar un ejemplo de un multigrafo dirigido, que contiene aristas dirigidas marcadas en rojo, y bucles marcados en azul.

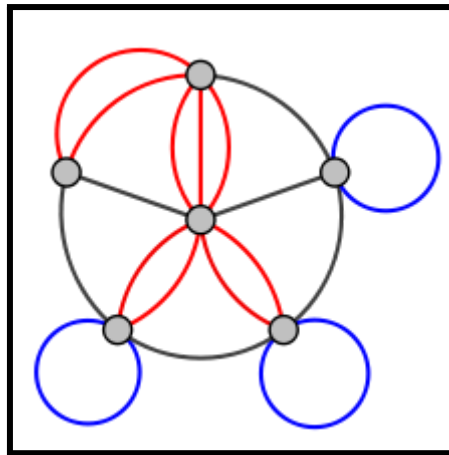


Imagen 3.1.9: Ejemplo de un multigrafo dirigido.

Por tanto, gracias a esta ampliación, tanto la clase Distancia como Distancia Única son prescindibles a la hora de realizar los cálculos de rutas, y el control general del sistema. Se debe a que se han implementado los tags como nodos de un multigrafo con aristas dirigidas. Sin embargo, estas clases no quedan obsoletas, al contrario, son necesarias para poder visualizar los circuitos correctamente y permitir su edición.

3.1.2. Implementación de los AGVs (PLCs).

Dentro de la jerga utilizada en el nombrado de clases y variables en la aplicación, solemos referirnos a lo mismo cuando se habla tanto de AGV como de PLC. Es cierto que son cosas diferentes, pero dado que la aplicación sólo se comunica con el PLC, y este aparato tiene en su base de datos variables que hacen referencia a las demás partes del AGV, el programador, se referirá a ambos conceptos como si fueran solo uno. Es por ello que a veces los nombres de las variables intercalan estos dos nombres.

Se ha añadido un campo en cada PLC con un **número de identificación interno único artificial**, que aumenta automáticamente, para poder distinguir entre los distintos PLC del circuito.

Otro campo extra que se añade indica la **dirección IP** del servidor OPC UA que existe en cada PLC.

Además de estas variables, se ha incluido un campo llamado **“Requested”** que estará activa cuando un AGV esté atendiendo una petición. En caso contrario, si no está atendiendo ninguna petición, estará inactiva. De esta manera, nunca se asignará una nueva petición, a un AGV que tenga este campo activo.

Para llevar a cabo la implementación de la representación en el sistema de los AGV, se han tenido en cuenta las variables que se encuentran en la base de datos interna del PLC, con las que el sistema interactúa. A partir de cada una de estas variables, se ha creado un campo en la nueva clase PLC. **[Para ver la lista de variables, accede al anexo 4].**

3.1.3. Implementación de las peticiones de AGVs.

A través de la implementación de las peticiones obtenemos un registro de los distintos movimientos de los AGVs por el circuito. Una petición representa una orden que lanza un trabajador para solicitar el movimiento de uno de los AGVs y que realice acciones específicas con los carritos existentes en el circuito. No todos los movimientos de los AGVs por el circuito se realizan a través de peticiones, dado que existen ciertos movimientos que se efectúan automáticamente y de éstos, no nos interesa tener registro. Únicamente se van a tener en cuenta como peticiones, las realizadas por medio de los usuarios desde los tags que tengan información de petición. Es decir, los movimientos que requieran de movimiento de un carrito.

Las peticiones se guardan en la base de datos. Para la implementación de las peticiones en el sistema, se ha creado la clase: Petición.

3.1.3.1. Implementación de software de la clase Petición.

Esta clase dispone de un campo que representa el **identificador único** de cada petición, que se crea con la información del momento en el que la petición se genera. Para este identificador se tiene en cuenta el lugar para el que se ha realizado la petición, y la hora exacta. De esta manera, nos aseguramos de que sea única. Se almacena información sobre la fecha y hora del momento de la petición y del destino al que se quiere llegar. El destino siempre será un tag, es decir, un nodo. Sin embargo, a veces, una petición no sólo almacena la información de un destino, sino que podrá guardar hasta dos destinos. Los destinos están relacionados con la variable anteriormente mencionada "**ListaRuta**".

Una petición puede encontrarse en 3 estados. Un primer estado, **pendiente**, cuando está a la espera de que se le asigne un AGV para ser llevada a cabo. Un segundo estado, **en curso**, cuando una petición se está llevando a cabo por un AGV. Por último, un tercer estado final, **completado**, cuando la petición se ha gestionado correctamente y ha terminado de ejecutarse, es decir, el AGV ha llegado a su tag final.

Por lo tanto, los campos que deben formar parte de esta clase son: el estado de la petición, los tags por los que debe pasar, y el AGV involucrado. Una petición va ligada a un AGV, pero sin embargo, no siempre va a estar asignada a uno en concreto, dado que cuando todavía está en estado pendiente, el campo AGV en la petición es nulo.

Está pensado que en el futuro pueda haber distintos tipos de prioridad en las peticiones, de tal manera que existan peticiones más urgentes que deban realizarse antes que otras. Sin embargo, por el momento, no necesitamos de esta distinción.

3.1.4. Implementación del movimiento de los AGVs por el circuito.

Para la implementación del movimiento de los AGVs, será necesario dotarlos de una ruta a seguir. El objetivo principal que se busca a la hora de asignar rutas a los AGVs, es que lleguen al destino de manera óptima, es decir, lo antes posible, y sin quedarse sin batería. Para ello, se han diseñado dos funciones básicas que calculan recorridos óptimos de los AGVs por el circuito implementado.

Una de ellas calcula el recorrido óptimo **desde VARIOS puntos de inicio hasta UN punto final**, y escoge, entre los recorridos anteriores, el mejor de ellos. La otra función calcula el recorrido óptimo **desde UN punto de inicio hasta VARIOS puntos finales**, y escoge, entre los recorridos anteriores, el mejor de ellos.

Las llamamos funciones básicas porque forman parte de la base de las peticiones de los usuarios implementadas finalmente.

La *primera función* se utiliza para escoger el AGV óptimo para llegar a un lugar. Esta función, teniendo en cuenta el grafo del circuito, escoge entre todos, el AGV que tiene la **ruta** óptima para llegar hasta un punto en específico. Para conocer qué ruta es más corta, se calcula la mínima **suma de las distancias** que existe entre cada una de las posiciones de los AGVs y un nodo final.

En teoría de grafos se denomina distancia o distancia geodésica entre dos vértices o nodos de un grafo a la longitud o número de aristas del camino más corto entre ellos. No debemos confundir **este término** con la distancia de la que hablamos en esta función, dado que en nuestro caso, esta distancia, en lugar de ser el número de aristas, es el número de metros que suman el conjunto de aristas.

La *segunda función* se utiliza para escoger la ruta óptima para llegar desde un nodo, hasta el carril de la zona de **carga de baterías** en el que existan menos AGVs esperando a cargar su batería y que se encuentre más cerca del lugar del vehículo. Con esta función se intenta que todos puedan disponer del mayor porcentaje de batería posible al mismo tiempo. Para la implementación de esta función, entra en juego el campo booleano en la clase Nodo que nos indica que un nodo es un tag con información de carga de batería en un carril en una zona de carga de baterías. Se debe tener en cuenta cuántos AGV se encuentran en cada carril de cada zona de carga de batería, para así conocer cuál es el carril con menos saturación. Por ello, el nodo tag del carril guardará un campo llamado **"NumEnCargador"**, que representa el número de AGVs que existen en el carril en ese instante. Este número debe actualizarse cada vez que se quiere utilizar esta función.

La implementación de estas funciones, recuerdan a varios algoritmos cuyo objetivo es calcular caminos mínimos. **[Para ver la información de los tipos de algoritmos, anexo 5].**

Tras varias comparaciones, comprobamos que estos algoritmos tienen cierta similitud con los algoritmos utilizados en las funciones, pero no llegan a usarse exactamente.

3.2. Implementación de la aplicación de petición.

La aplicación de petición se ha implementado en dos versiones: una aplicación de Windows Forms, y una aplicación web. Para la versión de Windows Forms, los usuarios deberán iniciar sesión con un usuario y una contraseña. El sistema diferenciará entre usuarios privilegiados y no privilegiados. Los usuarios privilegiados disponen de más opciones.

Existen dos partes esenciales. La parte del lanzamiento de las peticiones, que es accesible por todos los usuarios, y la parte de la edición y creación de circuitos, disponible únicamente para los usuarios privilegiados. De este modo podemos asegurarnos de que

ningún operario no autorizado puede modificar el circuito, y se limita el error humano, dado que modificar o eliminar un circuito puede ser muy perjudicial para la aplicación.

La interfaz inicial dispone de una lista con los circuitos existentes. Cuando se clickea en uno de los circuitos, aparecen dos botones: un botón para acceder a la parte de lanzar peticiones en el circuito seleccionado, y otro botón para editar el circuito seleccionado. Además de estos dos botones, existe un botón extra que permite crear un circuito desde cero. Los botones de edición y creación de circuitos no serán visibles por los usuarios no privilegiados.

3.2.1. Implementación de la parte de lanzar peticiones.

Para poder acceder a la interfaz que les permite realizar peticiones, los usuarios deben pasar previamente por otra interfaz, en la cual deben seleccionar un lugar del circuito desde donde se van a realizar peticiones. Estos lugares son, las zonas de petición, que se corresponden con los nodos en el grafo que simbolizan un tag de información de petición.

Una vez seleccionado el lugar de petición, y llegando a la interfaz que permite lanzar peticiones, se muestran varios elementos en la pantalla:

- En el centro, se encuentra dibujado el mapa del circuito y los AGV en su posición actual del mismo. La posición de los AGV se va actualizando a tiempo real por medio de una tarea periódica que ejecuta cada aplicación de petición que lee la tabla de los AGV en la base de datos.
- Se observan también abajo a la derecha, 3 listas. Corresponden con la información sobre las peticiones lanzadas por todos los usuarios. Cada una de las listas corresponde con peticiones que se encuentran en un estado determinado. Es decir, hay una lista para las peticiones que se encuentran en estado finalizado, otra para las peticiones que están en curso, y otra para las peticiones que aún siguen pendientes.
- Por último, en la interfaz vemos dos botones que nos permiten realizar dos peticiones predeterminadas: solicitar un carrito vacío del almacén de carritos vacíos, o enviar un carrito lleno al almacén de carritos llenos.

[Para ver la implementación de los dos botones que lanzan peticiones, anexo 6].

No se descarta la inclusión de muchos más botones para flexibilizar las peticiones, sin embargo, para el tipo de oficio que se realiza en la empresa, de momento lo más útil es únicamente disponer de estas dos opciones.

3.2.2. Implementación de la creación y edición de circuitos.

Para poder implementar la creación y la edición de circuitos de AGVs, se ha creado una interfaz que permite a los usuarios realizar estas acciones. Será posible acceder a esta interfaz desde la interfaz inicial, desde la que nos dará opciones de seleccionar un circuito para editarlo, o de crear un circuito completamente nuevo.

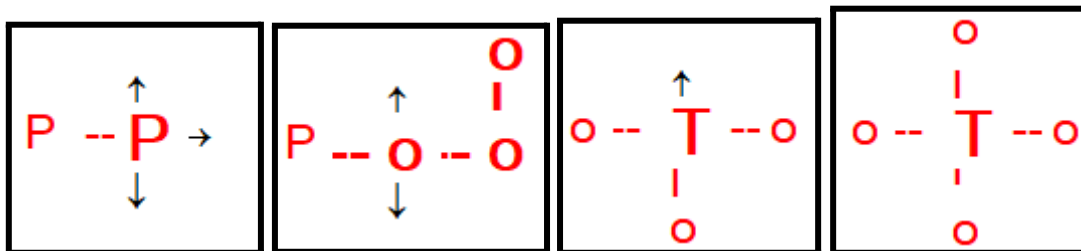
Una vez dentro de la interfaz de creación y edición, observamos que en el centro de la pantalla se encuentra representado el circuito. Arriba a la derecha se encuentran los distintos campos a rellenar de cada nodo o arista, como por ejemplo su “**número real**” o “**tag ID**”; o en el caso de los nodos, permitir diferenciar entre el tipo de información que almacena.

Los nodos son representados en la interfaz con una etiqueta de una letra que nos indica el tipo de información que tiene.

Las aristas son representadas en el circuito con etiquetas que simulan líneas.

Para poder seleccionar un nodo o una arista, y de esta manera modificar sus campos, simplemente haremos clic encima de la etiqueta que los representa. Para mostrar al usuario que un nodo o arista están seleccionados, su etiqueta se hará más grande, y en los campos de arriba a la derecha se mostrará su información. Aparecerán unas etiquetas de flechas alrededor de la etiqueta del nodo. Estas etiquetas de flechas representan las posibles direcciones en las que se pueden añadir nodos en el circuito. Es posible añadir un nodo en una dirección del circuito cuando no existe ningún otro nodo en la siguiente posición. Por ejemplo:

La primera imagen muestra un nodo que tiene disponibles huecos en 3 direcciones, en la segunda imagen, el nodo sólo tiene disponibles 2, en la tercera imagen, tiene disponible 1 hueco. En la cuarta, no tiene disponible ningún hueco.



Imágenes 3.2.8, 3.2.9, 3.2.10, 3.2.11: Huecos disponibles en nodos.

En el caso de la creación de un nuevo circuito, el grafo inicial comienza con 2 nodos, unidos por dos aristas dirigidas, una en cada sentido. El usuario visiona estas dos aristas dirigidas como una sola línea.

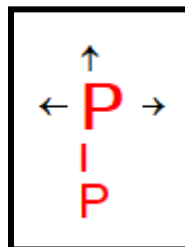


Imagen 3.2.12: Grafo inicial, con el nodo de arriba seleccionado.

En la implementación de la creación de nuevos nodos a partir de dichas flechas, se han tenido en cuenta distintas reglas que hacen que la creación de los mismos sea fácil, eficaz y rápida. Por ejemplo, no puede existir un nodo sin ninguna arista dirigida hacia él. Cuando esto ocurre, este nodo se elimina automáticamente.

Los nodos de los que sólo sale una arista dirigida, se considerarán tags con información de petición, ya que estos puntos límite del circuito, son los lugares desde donde se realizan las peticiones de carritos. Se representan con la letra “P”.

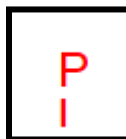


Imagen 3.2.13: Representación de un tag con información de petición, y la arista que lo conecta con el grafo.

Cuando un nodo tiene aristas dirigidas en dos de sus sentidos, se considera camino, y se calcula automáticamente el conjunto de nodos y vértices que simulan una distancia entre dos tags. Se representa con la letra “o”.

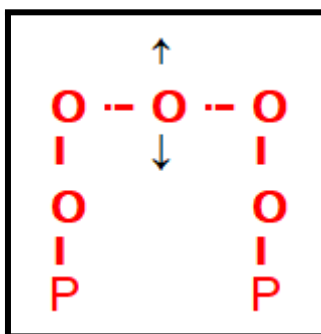


Imagen 3.2.14: Representación de un nodo camino que forma parte de un camino de 5 nodos camino.

Cuando un nodo tiene aristas en 3 o 4 de sus sentidos, se considera un tag con información de decisión. Se representa con la letra “T”.

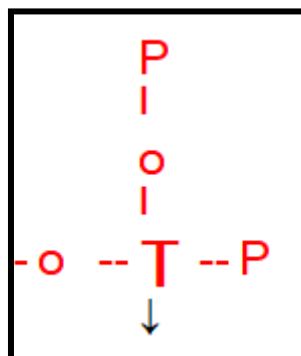
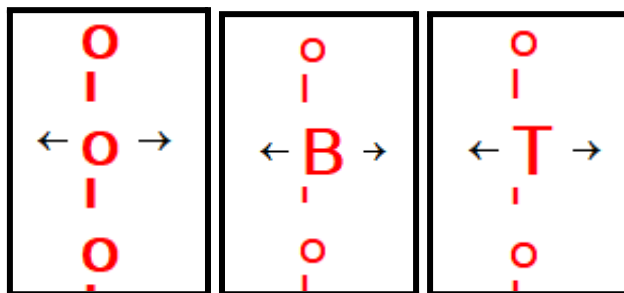


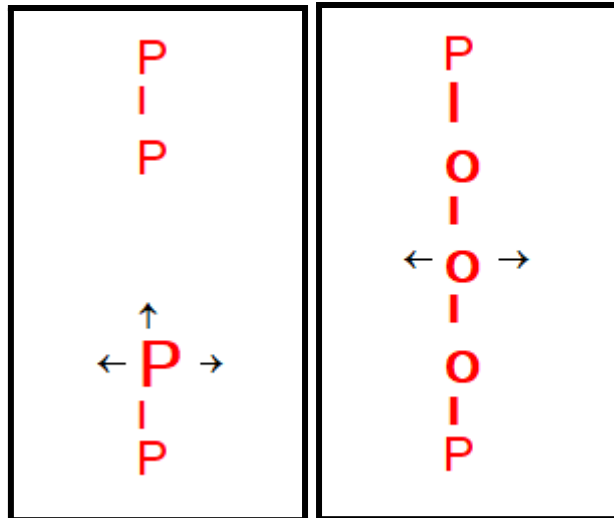
Imagen 3.2.15: Representación de un tag con información de decisión.

Para poder incluir los tags con los otros tipos de información, como el tag con información de carga de batería, el tag con información de bloqueo y desbloqueo, o simplemente un tag con información de posición, utilizaremos los caminos. Un nodo camino se puede sustituir por uno de estos. Al hacerlo, se recalculan las distancias.



Imágenes 3.2.16, 3.2.17, 3.2.18: Representación de un camino, un tag con información de carga de baterías y un tag con información común o de bloqueo/desbloqueo.

Como se ha mencionado, al añadir un nodo en una posición, se calcula qué otros nodos hay a su alrededor, para así realizar un cálculo de qué aristas deben crearse. En este ejemplo, dado que vamos a crear un nodo en medio de estas dos zonas de circuito, se va a crear un camino automáticamente.



Imágenes 3.2.19, 3.2.20: Ejemplo de la creación de un camino.

De la misma manera, cuando un nodo es eliminado de la representación del circuito, se actualizan las nuevas distancias y las nuevas informaciones de los nodos involucrados (si los hubiera). Por ejemplo, al eliminar un nodo tag con información de petición que está unido a un nodo camino, ese nodo camino pasará a convertirse en un nodo tag con información de petición.

3.2.3. Traslado de aplicación Windows a aplicación web.

Para realizar el traslado de una aplicación a otra, se ha creado una aplicación web nueva. Para crear esta aplicación web, desde Visual Studio, se elige la opción “Aplicación web ASP.NET Core”.

ASP.NET [14] es un marco web gratuito para crear excelentes sitios web y aplicaciones web mediante HTML, CSS y JavaScript. También puede crear API web y usar tecnologías en tiempo real como Websockets.

ASP.NET Web Pages y la sintaxis de Razor [15] proporcionan una manera rápida, accesible y ligera de combinar código de servidor con HTML para crear contenido web dinámico.

Se ha ido copiando poco a poco la parte del lanzado de peticiones desde la aplicación de peticiones para realizar un traslado seguro. El backend de esta nueva aplicación es C#, por lo que nos será sencillo trasladar el código de una aplicación a otra. Sin embargo, el frontend de esta aplicación web ya no estará escrito en C#, dado que ya no nos encontramos ante formularios de Windows. En este caso, se rehace el código de la parte del frontend a HTML.

Las variables globales se van a reemplazar por las Session de las aplicaciones web.

Session se utiliza para almacenar datos, es muy potente y muy utilizado ya que puedes guardar todo tipo de datos. No hay que confundir Session con cookies, ya que las

cookies almacenan información en la parte de cliente (memoria del navegador), y las “Session” se almacenan en la parte del servidor.

Dicho esto, podemos definir las Session [16] como variables que pueden almacenar cualquier dato en la parte del servidor, capaz de manejar información en peticiones de ambos sentidos (cliente-servidor).

Al realizar el traslado, se ha hecho uso de páginas maestras.

Una Master Page [17] o Página Principal es una estructura base para un conjunto de páginas pertenecientes a un mismo sitio Web. Este esqueleto base se almacena en un archivo independiente y luego es heredado por otras páginas que requieren esa estructura base.

Un ejemplo donde utilizar una página Master Page es la definición de la cabecera y el pie del sitio, luego todas las otras páginas del sitio heredan de dicha página. La ventaja fundamental en este ejemplo es que si hay cambios en la cabecera del sitio solo debemos hacerlos en la página principal y con esto se propaga en todas las páginas que heredan de la misma. Esto es útil por ejemplo cuando pasamos por las distintas páginas previas a la parte del lanzamiento de peticiones, cuando tenemos que escoger entre los distintos tags con información de petición que pertenecen al circuito escogido.

3.3. Aplicación de gestión.

La aplicación de gestión corre en paralelo a las distintas aplicaciones de petición. Al contrario que ellas, no dispone de interfaz, dado que su función es asegurarse de que el sistema funciona correctamente, con distintas tareas que se ejecutan periódicamente.

3.3.1. Implementación de las tareas periódicas.

Para que todo funcione correctamente, esta aplicación debe ejecutar distintas tareas periódicas:

→ **Tarea que gestiona las peticiones:** Esta tarea se asegura de que las peticiones en estado pendiente, y las peticiones en estado en curso, puedan desempeñarse y ser llevadas a cabo perfectamente por los AGV del circuito. Para las peticiones pendientes, busca AGVs listos para poder llevarlas a cabo, es decir, que dispongan de batería suficiente, y no estén atendiendo ninguna otra petición. Para las peticiones en curso, comprueba si han llegado a su primer destino, y, si es necesario, crea otra petición que lleva un carrito vacío a la posición. Para las peticiones en curso, si ha llegado a su destino final, las establece como completadas.

El funcionamiento es el siguiente. Primero, se seleccionan todas las peticiones pendientes y en curso del circuito, para comprobar si ha habido alguna actualización en su estado.

- En el caso de las pendientes, elige el AGV óptimo de los AGV que no estén realizando ninguna petición para llegar a asignarle dicha petición. Una vez se la ha asignado. Valida el campo “**Requested**” del AGV, que significa que ha sido pedido, y cambia el estado de la petición de pendiente a en curso. También se encargará de escribir en la variable “**ListaRuta**” del AGV los dos

destinos que tiene que seguir, y por ello escribirá dichos puntos en las dos primeras posiciones de la variable "ListaRuta".

- En el caso de las peticiones en curso, se comprueba si ha llegado a su primer destino. Si su primer destino se trataba de una zona de petición, entonces se crea otra petición que se encarga de llevar un carrito vacío a dicha posición, para que en ningún caso, la zona de petición se encuentre sin carrito.
- De las peticiones en curso, si se ha llegado a su segundo y último destino, se cambia el estado de la petición a completada, se invalida el campo "Requested" de dicho AGV y se calcula la vuelta del mismo al carril más óptimo de la zona de carga de baterías en el caso de que lo necesite.

→**Tarea que recircula los AGVs:** Esta tarea comprueba si hay un AGV con batería baja y se encuentra en un carril para cargar, pero no está cargando dado que existen uno o más AGVs delante de él. Si los AGVs de delante suya tienen batería alta, los va recirculando para que den paso al AGV que necesita recargar su batería. Se ejecuta cada 20 segundos.

Para saber qué carril es el óptimo, utiliza la **segunda función básica**. Si existen varios carriles que tienen el mismo número de AGVs, se escoge el que esté más cerca. Por lo tanto, se prioriza la disponibilidad del carril con respecto a su cercanía. Se realiza de esta manera dado que se prevé que los circuitos no sean muy extensos. La heurística de la comprobación de la recirculación es la siguiente: si hay un AGV en un carril cargando con más del 70% de batería y existe un AGV en la cola de ese carril con menos del 15% de batería, se recircula al que está cargando.

→**Tarea que comprueba la carga de los AGVs:** Comprueba si existe un AGV con batería baja, que no está yendo a cargar su batería ni atendiendo ninguna petición, y lo envía a cargar su batería mediante la segunda función base. Se ejecuta cada 8 segundos.

→**Tarea que comprueba las zonas de exclusión mutua:** Comprueba si existen AGVs que quieren entrar en cualquiera de las zonas de exclusión mutua o salir de las mismas. Para asegurarnos de que un AGV se encuentra en una de estas 2 circunstancias, comprobamos distintas variables. Para asegurarnos de que la circulación en estas zonas tanto de entrada como de salida sea la correcta, la aplicación debe de comprobar una serie de variables.

- Por una parte, para asegurarnos de que un AGV puede entrar a una zona de exclusión mutua, comprobará que 3 variables estén activas.
 - La variable "CruceON" de la clase AGV. Esta variable la activa el programa interno del PLC cuando el lector de tags RFID ha leído un tag con información de bloqueo.
 - La variable "EsBloqueo", del nodo en el que se encuentra el AGV, nos indica que el tag sobre el que se encuentra el AGV tiene información de bloqueo.
 - Por último, dicho nodo, dispondrá de otra variable llamada "LibreBloqueo" que estará activa cuando la zona que delimitan el tag de bloqueo y desbloqueo no tiene a ningún AGV en su interior.

Si estas 3 variables están activas, entonces la aplicación desactiva la variable “LibreBloqueo” de dicho nodo, y activará la variable “CruceOK” del AGV, para informarle al programa del PLC que puede continuar.

- Por otra parte, para asegurarnos de que un AGV está saliendo de una zona de exclusión mutua, comprobará que 2 variables están activas.
 - La variable “CruceOK” de un AGV, es decir, la ha activado la aplicación previamente e indica que se el AGV se encuentra en una zona de exclusión mutua .
 - La variable “EsDesbloqueo”, del nodo en el que se encuentra el AGV, Nos indica que el tag sobre el que se encuentra el AGV tiene información de desbloqueo.

En este caso, activamos la variable de la zona de bloqueo correspondiente, “LibreBloqueo” otra vez, para establecer esta zona como libre. Además, establecemos la variable “CruceOK” a inactiva del AGV que pasa por el tag de desbloqueo.

La variable “LibreBloqueo” actúa como el “mutex” del sistema.

Los algoritmos de exclusión mutua ayudan a garantizar que un proceso pueda escribir en una sola dirección y que en ciertas secciones de código otro proceso pueda acceder a los recursos necesarios. El utilizado nos recuerda a cierto tipo de algoritmos basados en tokens: Un único token se comparte en varios nodos, el proceso que posea el token se entrega a la región crítica. En nuestro caso la variable “CruceOK” y la variable “LibreBloqueo” actúan como tokens. Varios ejemplos de estos algoritmos pueden ser tanto el algoritmo de servidor centralizado, algoritmo en anillo o algoritmo de Ricart y Agrawala.

Capítulo 4.

Pruebas realizadas.

4.1. Introducción.

En el capítulo 4 se va a explicar toda la información correspondiente a las pruebas que se han ido realizando a lo largo del trabajo. Además, se dará una pequeña explicación sobre el motivo por el cual se han llevado a cabo.

Las pruebas no sólo han sido realizadas al final del trabajo, sino que también se han probado a lo largo del desarrollo del mismo, para las comprobaciones correspondientes de que el trabajo siguiera su curso eficazmente.

4.2. Alcance de las pruebas.

4.2.1. Pruebas de Usabilidad.

En lo correspondiente a las pruebas de usabilidad, se ha comprobado que los distintos usuarios en las interfaces de las aplicaciones de petición de AGVs, tanto en la aplicación web, como en la aplicación Windows, sean capaces de navegar por todas las pantallas de las mismas sin problema, además de que la navegación sea intuitiva, simple y se ejecute con normalidad.

En cuanto a la usabilidad de los usuarios, se espera que los usuarios dispongan de la aplicación de Windows en su dispositivo Windows y de la aplicación web, accediendo a ellas desde los distintos dispositivos de la empresa Ergomaq. Las aplicaciones se subirán al servidor compartido de la empresa, y serán accesibles desde ahí.

Para comprobar que se cumplen estos requisitos, a lo largo del proyecto, se han ido haciendo pruebas constantes, conforme se ha ido desarrollando la aplicación.

Los resultados de estas pruebas han sido satisfactorios.

4.2.2. Pruebas Funcionales.

4.2.2.1. Prueba: lectura y escritura con 1 PLC y 2 PLCs.

En el proyecto se va a trabajar con varios PLCs, y para comenzar a comprobar las librerías de acceso al PLC mediante OPC UA, se realizó una pequeña prueba inicial.

La comprobación de esta prueba se ha realizado mediante una aplicación auxiliar cuya función principal era escribir y leer variables de tipos diferentes que se incluyeron en

una base de datos del PLC. Los tipos de las variables que se comprobaron fueron: “string” (cadena de caracteres), “integer” (número entero) y “date” (fecha). De esta manera, se comprobó la eficacia del paquete que contiene la librería de OPC UA.

Los resultados de estas pruebas son satisfactorios.

4.2.2.2. Prueba: creación y edición de circuitos.

Para comprobar que la generación y edición de circuitos funciona correctamente, a lo largo del proyecto, se han ido haciendo pruebas constantes, conforme se ha ido desarrollando la aplicación. En las etapas de desarrollo, se idearon diferentes circuitos, y se plasmaron en este editor correctamente.

Sin comprobar la creación de circuitos, ninguna de las pruebas siguientes podrían ser posibles, por lo que es importante cerciorarse de la eficacia de la funcionalidad. Para comprobarlo, además de la visibilidad del propio circuito en las aplicaciones de petición, se ha ido comprobando la información en la base de datos SQL conforme se han ido haciendo cambios en el circuito.

Las pruebas se llevaron a cabo satisfactoriamente.

4.2.2.3. Prueba: simulación de circuito con 1 PLC.

Para realizar las pruebas de simulación, se ha simulado el comportamiento de un AGV real en un PLC. Para realizarlo, se ha hecho uso de la herramienta “TIA Portal V16”. Esta herramienta es la encargada de instalar en el PLC del AGV el programa que se va a utilizar, junto con la inclusión de las variables en su base de datos correspondiente. Una vez cargado el programa en el PLC, desde el propio “TIA Portal V16”, se es capaz de modificar las variables, simulando así el comportamiento que tendría en el caso de estar en un circuito real. Para poder simular una situación real, en esta simulación se ha “” un circuito que contenga las alternativas más comunes, como puede ser una zona de carga de batería, una zona de peticiones y una zona de exclusión mutua, y se ha utilizado un PLC real en el que se ha instalado un programa que simula el comportamiento que tendría en el caso de estar en una situación real.

Al realizar esta simulación, se ha creado un programa que es una pequeña recreación del programa que hay instalado en un PLC de un AGV real. El programa se compone de varios bloques de programa. Cada bloque de programa se distribuye en varios segmentos, en los que se define el comportamiento del mismo. Uno de los bloques más importantes es el mapeado de las variables del PLC. Gracias a mapear las variables, no se accede directamente a ellas, por lo que aporta seguridad al sistema completo.

Las variables del PLC pueden ser de entrada o de salida. Únicamente las de entrada pueden ser modificadas mediante el protocolo OPC UA, es decir, desde la aplicación de gestión. Las variables de salida, son meramente informativas y no pueden ser modificadas, pero nos ayudan a saber la situación del AGV en cada momento.

Para realizar la simulación, desde el propio programa de TIA Portal, se pueden modificar dichas variables de tal forma que reproduzca una situación completamente real. Son datos como el último tag leído, que aporta información sobre en qué ubicación se encuentra el AGV, su porcentaje de batería, la velocidad a la que circula, etc.

Para poder manejar mejor estas variables, se ha creado una base de datos "AUXILIAR" dentro del programa interno del PLC, que no existiría en un entorno real, pero que ayuda en este caso a realizar cambios sencillos, para depurar más eficazmente. Tiene variables que establecen la batería al 100%, que resetea las variables que referencian a los tags de bloqueo y desbloqueo, y que permite escribir en la variable que referencia el último tag leído.

El entorno establecido en esta situación ha sido el siguiente:

- La red externa que se estableció fue la red que existe en las oficinas de Ergomaq.
- La dirección IP externa del AGV es 192.168.1.1.
- El circuito está completamente simulado.
- El equipo accede a la base de datos SQL que existe en la red de las oficinas de Ergomaq. En el mismo equipo se han ejecutado la aplicación de gestión, la aplicación de petición Windows y la aplicación de petición web.

En este entorno, se han comprobado los siguientes objetivos:

- 1.-Acceso a la pantalla de petición.
 - 2.-Solicitud de un carrito y visualización del entorno.
 - 3.-Gestión de puntos del circuito de acceso exclusivo.
 - 4.-Vuelta a cargar la batería.
-
- **1.-Acceso a la pantalla de petición:** al abrir ambas aplicaciones de petición, tanto la aplicación web, como la aplicación Windows, debe seleccionarse el lugar desde el cual se quieren hacer peticiones de carritos. Previo a este paso, debemos haber comprobado que el circuito utilizado se ha guardado en la base de datos correctamente. Una vez se escoge el circuito, se deben observar los puntos establecidos para las peticiones dentro del circuito, y elegir uno de ellos.
 - **2.-Solicitud de un carrito y visualización del entorno:** tras elegir el lugar de petición, al llegar a la pantalla principal de peticiones, observamos el mapa del circuito, la lista de las peticiones, y distintos botones que permiten solicitar carritos vacíos, o llevar carritos llenos a los lugares establecidos. Se pulsó el botón de solicitud de un carrito vacío y se creó una petición, la cual gestionó la aplicación de gestión. Esta aplicación decidió cuál es el AGV óptimo para realizar este recorrido. Dado que en este ejemplo sólo hay un PLC, escribirá en su variable de rutas: "Lista_Ruta", los destinos que tiene que seguir. Para simular que se mueve, desde el programa de TIA Portal, se va modificando la variable del último tag leído. La aplicación de gestión va observando estos cambios, y plasmándolos en la base de datos. La aplicación de petición lee de esta base de datos, y cambia la posición del AGV en el mapa de la pantalla. Además, ocurre lo mismo con los cambios en la petición realizada; una vez se gestiona totalmente, en la aplicación de petición se plasmará el estado de dicha petición.

- **3.-Gestión de puntos del circuito de acceso exclusivo:** Se estableció un punto en el circuito donde el PLC lee un tag de bloqueo. En ese caso, desde la aplicación de TIA Portal, se escribirá un 1 en la variable “Cruce_ON”, para que la aplicación de gestión pueda detectar que hay un PLC interesado en entrar en dicha sección exclusiva. La aplicación de gestión comprueba que no hay nadie en esa sección crítica, y escribe en la variable del AGV “Cruce_OK” un 1 para avisar de que puede continuar con su recorrido con normalidad. Una vez sale de ese acceso en exclusión mutua, la aplicación de gestión lo gestiona correctamente.
- **4.-Vuelta a cargar la batería:** una vez el PLC termina el recorrido simulado, es decir, ha llegado al último destino de la lista de rutas, se para. La aplicación de gestión debe darse cuenta, y llevar a este AGV a cargar la batería. Para ello escribe en su lista de rutas, la ruta a la zona de carga de baterías más óptima. Para saber qué lugar es el más óptimo, se calcula en qué zona de baterías hay menos AGV esperando, para así garantizar una carga de baterías repartida entre los distintos AGV.

Las pruebas se llevaron a cabo satisfactoriamente.

Problemas encontrados en la prueba: escritura en variables de salida mediante el protocolo OPC UA. A la hora de escribir en variables mediante el protocolo, es vital que éstas se organicen como variables de entrada. En el caso de que se trate de modificar esta variable, mientras está como variable de salida, no se va a reflejar el cambio. El problema es solucionado siempre estableciendo como se ha comentado, dichas variables, como variables de entrada y no de salida.

4.2.2.4. Prueba: circuito real con 2 AGVs.

Para poder realizar esta prueba, se ha creado un circuito real en una nave de la empresa. En ella, se han utilizado 2 AGVs.

El entorno establecido en esta situación ha sido el siguiente:

- La red externa que se estableció fue la 192.168.1.0.
- Las dos direcciones IP externas de ambos AGVs son las siguientes: 192.168.1.1 y 192.168.1.2.
- El circuito dispone de tags normales, con información de decisión, y de petición.
- El equipo que dispone de la base de datos SQL es un ordenador portátil conectado a la red externa. En el mismo equipo se han ejecutado la aplicación de gestión, la aplicación Windows de petición y la aplicación web de petición.

En este entorno, se han comprobado los siguientes objetivos:

- Puesta en marcha de los AGVs.
- Solicitudes de carritos y visualización del entorno.
- **Puesta en marcha de los AGVs:** al comenzar a funcionar, los AGVs deben estar parados, esperando a que se les asigne una petición. Esta inicialización se ha

llevado a cabo desde la aplicación de gestión, la cual, al ponerse en marcha, los para completamente y los deja listos para poder atender peticiones.

- **Solicitud de un carrito y visualización del entorno:** al seleccionar un punto desde el que se van a realizar peticiones desde la aplicación de petición, nos encontraremos en la interfaz al circuito con los AGVs. Esta posición se va actualizando cada cierto tiempo. Al solicitar el carrito, pulsando el botón, se obtienen los datos de la aplicación de gestión, la cual asigna la petición al AGV más cercano al tag que representa el almacén de carritos vacíos. Una vez asignado, la variable *Lista_Ruta* del PLC de dicho AGV cambia, y comienza a ir hacia el primer destino. Una vez se ha llegado al primer destino, continúa yendo hacia el segundo destino. La petición se ha llevado a cabo desde la aplicación de petición web y desde la aplicación de petición Windows. Se han realizado diferentes peticiones, y se ha comprobado que siempre se asignaba al AGV más cercano a la posición de la primera parada en la ruta. Además, se ha gestionado satisfactoriamente sin ningún problema.

Debido a las limitaciones de material en el momento en el que se realizó la prueba real, no se han podido comprobar los objetivos de la prueba de simulación en referencia al acceso en exclusión mutua, y la vuelta a la carga de baterías. Sin embargo, se es consciente de que en una situación real, la comprobación de estos objetivos es esencial.

Las pruebas se llevaron a cabo satisfactoriamente.

4.2.3. Pruebas de Compatibilidad.

Dado que existen dos versiones de aplicación de petición, es necesario realizar distintas pruebas referentes a la compatibilidad entre los dos tipos de aplicaciones. Para ello, se han puesto en marcha simultáneamente tanto una aplicación web de petición, como una aplicación Windows de petición.

Se realizaron peticiones simultáneas desde el mismo punto. Las peticiones fueron atendidas por la aplicación de gestión correctamente. Esto se debe a que aunque existan dos aplicaciones de petición, sólo hay una de gestión, por lo que no hay ningún problema con la cantidad de aplicaciones de petición que existan simultáneamente.

Por lo tanto, las pruebas se llevaron a cabo satisfactoriamente.

4.3. Estrategia y Desarrollo.

4.3.1. Estrategia.

La estrategia escogida a la hora de realizar las pruebas, depende directamente de los objetivos que se establecieron.

Los objetivos establecidos al comienzo fueron la monitorización de circuitos y de AGVs, la gestión de la lectura de tags para gestionar cruces y paradas, la gestión de baterías optimizando sus tiempos de carga, la optimización de recorridos, y la solicitud de

movimiento de AGVs desde distintos puntos para transportar los carritos. Se han tenido en cuenta los mismos para realizar las pruebas.

4.3.2. Tipo de Pruebas.

Tipos de pruebas: se pueden dividir en 2 tipos específicos: pruebas de caja blanca y pruebas de caja negra.

Con respecto a las pruebas de caja blanca, están ligadas al código de las 3 aplicaciones principales. Para estas pruebas, se han realizado monitorizaciones de los datos que van cambiando a lo largo de la ejecución de las mismas. Tanto los datos leídos por el protocolo OPC UA, como los datos leídos de la base de datos compartida mediante el visor de bases de datos SQL Manager. Para leer estos datos se ha hecho uso de la propia interfaz de las aplicaciones de Windows, y del cuadro de salida que ofrece Visual Studio, por el cual se pueden mostrar datos. Además, para comprobar que todo funciona correctamente, y que las variables dentro del PLC estaban siendo cambiadas, con ayuda del programa TIA PORTAL, desde el cual podemos leer el contenido de dichas variables, podemos asegurarnos de que todo funciona correctamente. En las pruebas realizadas en un entorno real, incluso se puede visualizar las pantallas HMI de cada AGV, que ofrecen información sobre la base de datos interna de su PLC.

Con respecto a las pruebas de caja negra, se han establecido unos patrones, y se ha ido comprobando poco a poco que los datos daban como resultado lo esperado. En estos casos, para ciertos datos de entrada, la salida debía ser igual que la esperada. En cuanto a las pruebas de caja negra relacionadas con la interfaz de usuario, se comprueba que el diseño fuera el correcto, y que los datos que se visualizan en la propia interfaz, sean correspondientes a los guardados en la base de datos, por ejemplo en el caso del muestreo del circuito.

4.4. Resultados y Conclusiones.

Para sumarizar, todas las pruebas realizadas tienen como resultado una salida favorable. En ellas, se comprueba que los objetivos se cumplen de manera satisfactoria, además de que la representación de los datos en la interfaz de usuarios sea la correcta, incluyendo además que cumple con los requisitos de utilidad, intuitividad y diseño.

Capítulo 5.

Conclusiones y líneas futuras.

5.1. Conclusiones.

A lo largo de este Trabajo de Fin de Grado, se ha diseñado e implementado un sistema que permite gestionar una flota de AGVs

En resumen, se ha conseguido por medio de un entorno de red apropiado para guardar información en una base de datos, una aplicación que se encarga de recibir peticiones, y enviar órdenes a los vehículos, y otra aplicación que sirve para poder enviar dichas peticiones.

Por un lado, la primera aplicación mencionada, corre en background, en un sólo equipo conectado a la red, y es la encargada de gestionar todas las peticiones, controla todo lo relacionado con el circuito.

Por otro lado, la segunda aplicación, es la aplicación que utilizarán los usuarios que quieran enviar peticiones y conectarse al sistema. Por medio de un front-end; tanto de una aplicación de Windows, como de una página web, serán capaces de escoger un punto en un circuito de AGVs para el cual pueden realizar peticiones, que solicitan la ayuda de uno de estos vehículos.

Asimismo, la base de datos utilizada está alojada en un servidor de la empresa, al cual se accede por medio de una red interna. De esta manera se es capaz de acceder a todos los datos, y poder coordinar todas las aplicaciones; y por lo tanto las peticiones.

En el desarrollo de este Trabajo de Fin de Curso, no sólo se han adquirido conocimientos sobre la programación en lenguaje C #, sino que también se ha investigado sobre todas las librerías que rodean el entorno. Además, se ha comprendido cómo funciona el protocolo OPC, así como el funcionamiento de un vehículo AGV, y todo el entorno que lo rodea.

Finalmente, se ha comprobado que las aplicaciones consiguen solventar todos los problemas que se plantearon inicialmente, gracias a las pruebas realizadas en la simulación del circuito. Sin embargo, hay algunos puntos que no se han probado en un circuito real. Este proyecto ha sido un primer contacto con el mundo de los AGV, tecnologías nuevas, ámbitos quizá desconocidos anteriormente, pero se ha conseguido aprender sobre todo ello, y salir adelante. Añadir que la parte probada sobre un entorno real de este proyecto ya podría ser implementada en producción en la empresa, para una aplicación inmediata en el entorno.

5.2. Líneas futuras.

Una vez acabado, se proponen mejoras, que están previstas de realizar próximamente, con las cuales, se podrán mejorar y expandir las posibilidades del sistema.

Como primera mejora, se prevé y se espera añadir la tecnología 5G, para que de esta manera, nuestra conexión con el AGV no sólo esté ligada a una red Wi-Fi interna, y se puedan realizar peticiones desde lugares remotos.

Además, es conveniente cambiar la visualización del mapa gráficamente, dado que nuestra representación del circuito, como de los AGVs en el mapa puede mejorar.

Por último, sería también interesante trasladar las funcionalidades de la aplicación Windows que no están plasmadas en la aplicación web, como la creación de circuitos.

Anexos.

Anexo 1: Hardware AGV.

Dispositivos de seguridad.

·**Setas de seguridad:** Setas de seguridad color rojo que se utilizan para parar la marcha del AGV. Al pulsar la seta, el AGV parará inmediatamente, y no reanudará la marcha hasta que se vuelva a dejar la seta en la posición inicial.



Imagen A1.1: Seta de seguridad.

·**Escáneres láseres:** Principal dispositivo de seguridad. Estos escáneres disponen de 3 modos. En el modo normal, o también llamado “GO”, el AGV avanza con normalidad, y las luces estarán de color verde. En el modo “Alert”, el AGV reduce la velocidad a la mitad dado que está detectando un objeto cercano, y las luces se ponen de color amarillo. En el modo “Stop”, el AGV para completamente hasta que se retire el objeto que lo está obstaculizando, y pone las luces de color rojo.



Imagen A1.2: Escáner láser.

Dispositivos de control.

·**PLC (Programmable Logic Controller):** El PLC es el dispositivo que lleva todas las señales de entrada y salida del AGV. Es el elemento más importante y crítico. La antena está conectada al mismo para que la aplicación de gestión se pueda conectar al AGV y le comunique las órdenes.

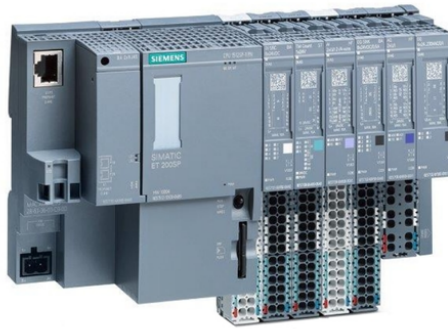


Imagen A1.3: PLC.

·**HMI (Human-Machine Interface) :** pantalla que permite realizar las acciones manuales en el AGV, como por ejemplo subir o bajar el pin, cambiarle el destino, cambiarle el modo de funcionamiento entre manual y automático.



Imagen A1.4: HMI.

·**Lector RFID (Radio-frequency Identification):** Para leer los tags que se va a ir encontrando el AGV por todo el circuito de la fábrica, existen dos lectores RFID (delantero y trasero) que le permiten pasar por encima de un tag y leer toda la información que lleva interna. A continuación, el AGV traducirá esta información a su control interno para saber lo que tiene que hacer en todo momento (giros, cambio de velocidad, cambio de zona de escáner, etc...)



Imagen A1.5: Lector RFID.

•**Sensores magnéticos:** Los sensores magnéticos se encargan de leer el campo magnético de la cinta y guiar al AGV por el circuito. El sensor detectará y administrará bifurcaciones de hasta 2 vías y se le puede indicar que siga la pista izquierda o derecha utilizando los comandos emitidos a través de los puertos serial/USB o el estado de dos entradas digitales.

*Imagen A1.6: Sensor magnético.*

•**Motores:** El AGV dispone de 4 motores, dos en el eje trasero y dos en el eje delantero. Así se puede conseguir un movimiento bidireccional. Estos ejes pueden moverse de forma manual con una llave Allen.

*Imagen A1.7: Motor.*

Dispositivos auxiliares.

•**Antena:** Actúa como cliente y se conecta a la red Wi-Fi de la fábrica. A través de ella, la aplicación de gestión es capaz de conectarse al PLC del AGV y asignarle órdenes, así como leer datos.



Imagen A1.8: Antena.

·**Bocina:** Es la encargada de generar un sonido mientras el AGV se mantiene en movimiento. Además también genera un sonido cuando existe un error activo, o algún aviso.



Imagen A1.9: Bocina.

·**Luces LED:** Las luces LED indican diferentes estados en los que se encuentra el AGV. Si están en color blanco, el AGV se encuentra en modo carga.

→ Si están en color verde, el AGV se encuentra en modo automático, con el escáner sin detectar objetos cercanos.

→ Si está en color amarillo, el AGV se encuentra en modo automático, pero el escáner está en modo alerta. Como ya se había anticipado antes, el AGV reduce su velocidad a la mitad, al estar detectando un objeto cercano.

→ Si están en color rojo, está en estado de error. El AGV se encontrará parado, y tendrá algún error activo.

→ Si están en color azul, el AGV se encuentra en modo manual con el escáner sin detectar objetos.

→ Si están en color morado, el AGV se encuentra en modo manual, con el escáner detectando un objeto cercano.



Imagen A1.10: Luz LED.

·**Luces intermitentes:** Estas luces se mantienen parpadeando mientras el AGV se encuentra en movimiento. Si tiene un giro hacia un lado, sólo parpadea el lado correspondiente actuando como un intermitente.

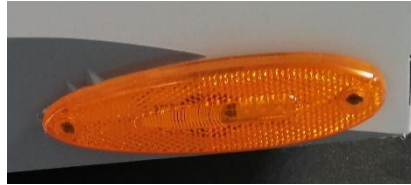


Imagen A1.11: Luz intermitente.

·**Joystick:** Se coloca en el conector situado debajo de la bocina, y permite mover el AGV en modo manual a distancia.



Imagen A1.12: Joystick.

·**Roldanas:** Las roldanas se encuentran en la parte superior del AGV y permiten fijar el carrito mientras el AGV lo transporta.



Imagen A1.13: Roldana.

•**Pin:** Los AGVs disponen de un cilindro en el centro llamado pin, el cual es usado para arrastrar los carritos de un lugar a otro de la fábrica. El sistema consiste en un motor con un muelle que siempre realiza el mismo movimiento entre dos puntos (pin arriba y pin abajo). El agv dispone de dos sensores de posición para detectar cuando el pin ha llegado arriba o abajo. Cuando sube el pin, se introduce en una abertura en el centro del carrito que le permite arrastrarlo siguiendo el movimiento del AGV con ayuda de las roldanas de la parte superior.

•**Sensor de carro (carrito):** El agv dispone de un sensor de carrito situado al lado del pin que le permite detectar cuando tiene un carrito encima. Es útil para añadir cierto control extra al AGV como por ejemplo, para detectar si ha cogido correctamente un carrito puede comparar si tiene el pin arriba pero el sensor de carrito no detecta nada encima. En este caso, el AGV lanzará una alarma indicando que algo ha ocurrido y no ha cogido el carrito en la posición.

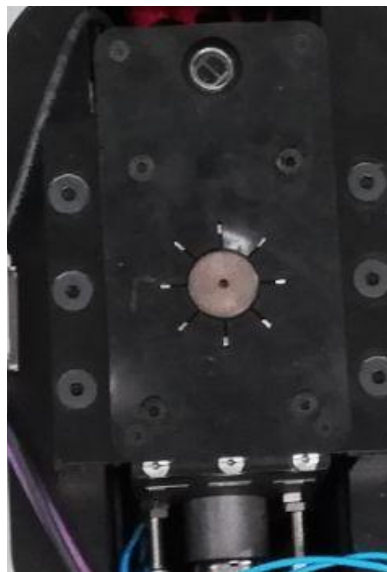


Imagen A1.14: Sensor de carrito y pin.

•**Pastillas para carga de baterías:** A la hora de cargar la batería, el AGV dispone de una forma manual y otra automática. La manual se encuentra abriendo la tapa delantera y se compone de un conector donde colocar el cable del cargador directamente. Para la forma automática se utilizan unas pastillas de carga situadas en la parte baja del AGV. Estas pastillas solo dejan pasar corriente hacia la batería cuando el sistema de control lo permite y esto solo ocurre cuando el AGV se encuentra en un punto de carga en el suelo. Una vez las pastillas entran en contacto con el punto de carga del suelo y el tag correspondiente da la orden de carga, el sistema de control activa el paso de corriente desde el punto de carga hasta la batería del AGV.

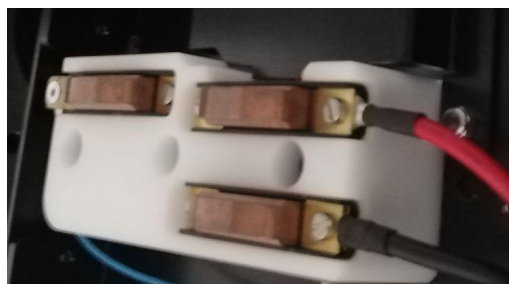


Imagen A1.15: Pastilla para carga de baterías.

- Botón Pin:** se utiliza para subir y bajar el pin de forma manual.
- Botón Reset:** se emplea para rearmar el AGV después de haber apretado una seta de seguridad.
- Selector ON/OFF:** se utiliza para apagar y encender el AGV
- Pulsador marcha:** se utiliza para funciones especiales según las necesidades del cliente o para permitir al AGV continuar avanzando por un cruce en el caso de que el servidor del tráfico haya perdido conexión.



Imagen A1.16: Botón pin, reset, ON/OFF, pulsador marcha.

- Botón extra:** el otro lado de los AGV tiene un botón extra que actualmente no tiene utilidad. Solo si fuera necesario para una función específica que necesite el cliente.

Anexo 2: Tipos de información en los tags RFID.

Existen distintos tipos de información que albergan los tags RFID si los clasificamos en la función que tienen en el circuito.

→**Decisión:** Cuando existe una bifurcación en el circuito, es necesaria la inclusión de un tag con información de decisión que indique al AGV por cuál de los dos caminos debe continuar. Normalmente esta decisión consiste en una bifurcación del circuito, es decir, existirá un camino que sigue recto, y otro que se bifurca hacia la izquierda o la derecha. El programa interno del PLC distingue entre un tag que indica una bifurcación hacia la izquierda u otro que indica una bifurcación hacia la derecha. Debe indicar al lector de la

cinta magnética si tiene que girar, por ello es importante que estos tag estén programados con su respectiva bifurcación. Sin embargo, en nuestra implementación, no nos interesa diferenciar entre estos dos tipos de bifurcación, dado que es información que no nos aporta nada y podría complicar el problema. Nos referiremos a estos tags entonces como un sólo tipo de información: de decisión.

→**Petición:** En una zona de peticiones de AGVs del entorno industrial, donde los trabajadores van a solicitar carritos, y van a solicitar su traslado a otro lugar de la fábrica, debe existir un tag con información específica para ello, dado que se quiere que cada vez que venga un AGV con un carrito, deje el carrito en este sitio, y se vaya. Y cuando detecte que tiene que llevarse, se lo lleve.

→**Bloqueo:** en el circuito existen varias zonas en las que sólo puede acceder un AGV por vez. Por lo tanto se debe gestionar éste acceso en exclusión mutua por medio de unos tags especiales. Cuando un AGV lea este tipo de tags, parará y se quedará esperando a que le den permiso para pasar.

→**Desbloqueo:** cuando un AGV lee este tag, significa que se encontraba en exclusión mutua, y que ha salido de la misma. Por lo tanto se deberá gestionar esta exclusión, y si existía otro AGV que estaba esperando para acceder, se le notifique.

→**Carga de baterías:** en un carril de una zona de carga de batería, al lado de las pastillas de carga de batería, se debe instalar un tag.

→**Posición:** estos tags no comprueban nada, en el circuito pueden existir tags que simplemente sirvan como indicación que un AGV ha pasado por ahí. Pueden ser útiles para indicar al sistema que el AGV se encuentra en un lugar específico, o para modificar su velocidad en una recta, u otras funcionalidades.

Anexo 3: Esquemas eléctricos.

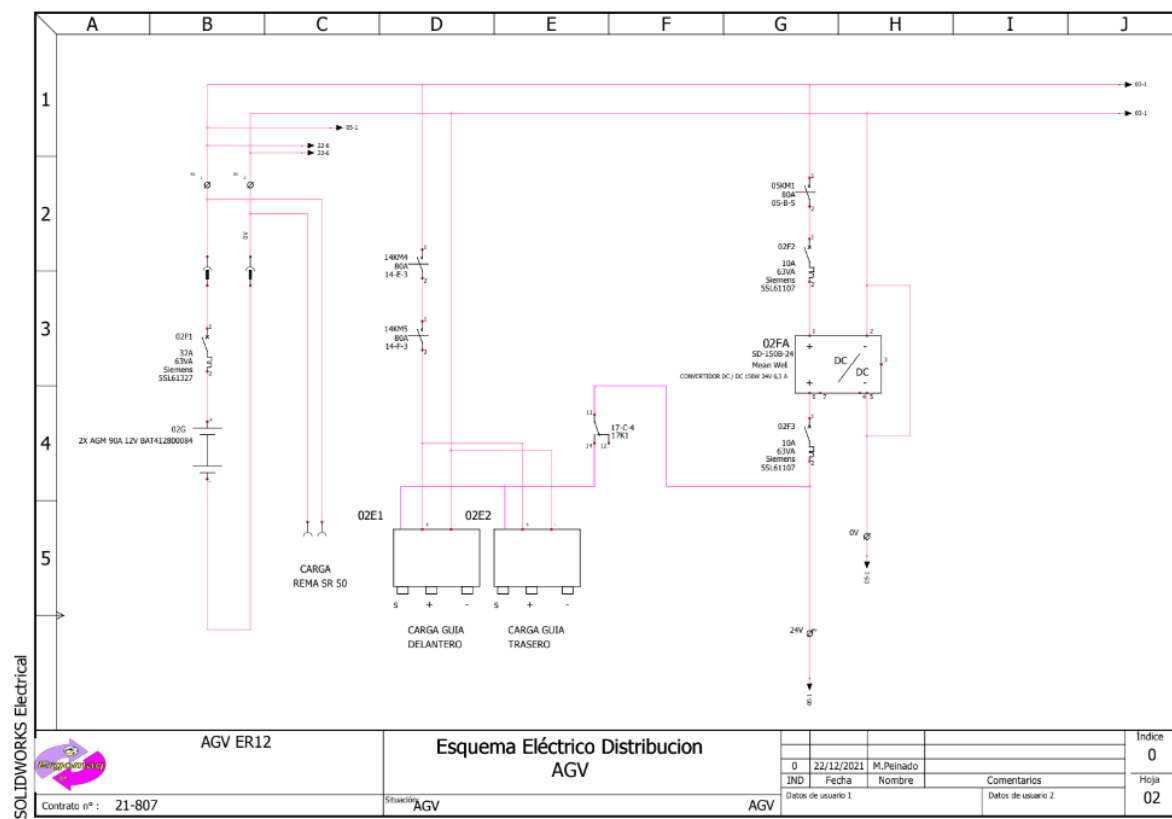


Imagen A3.1: Esquema eléctrico distribución carga.

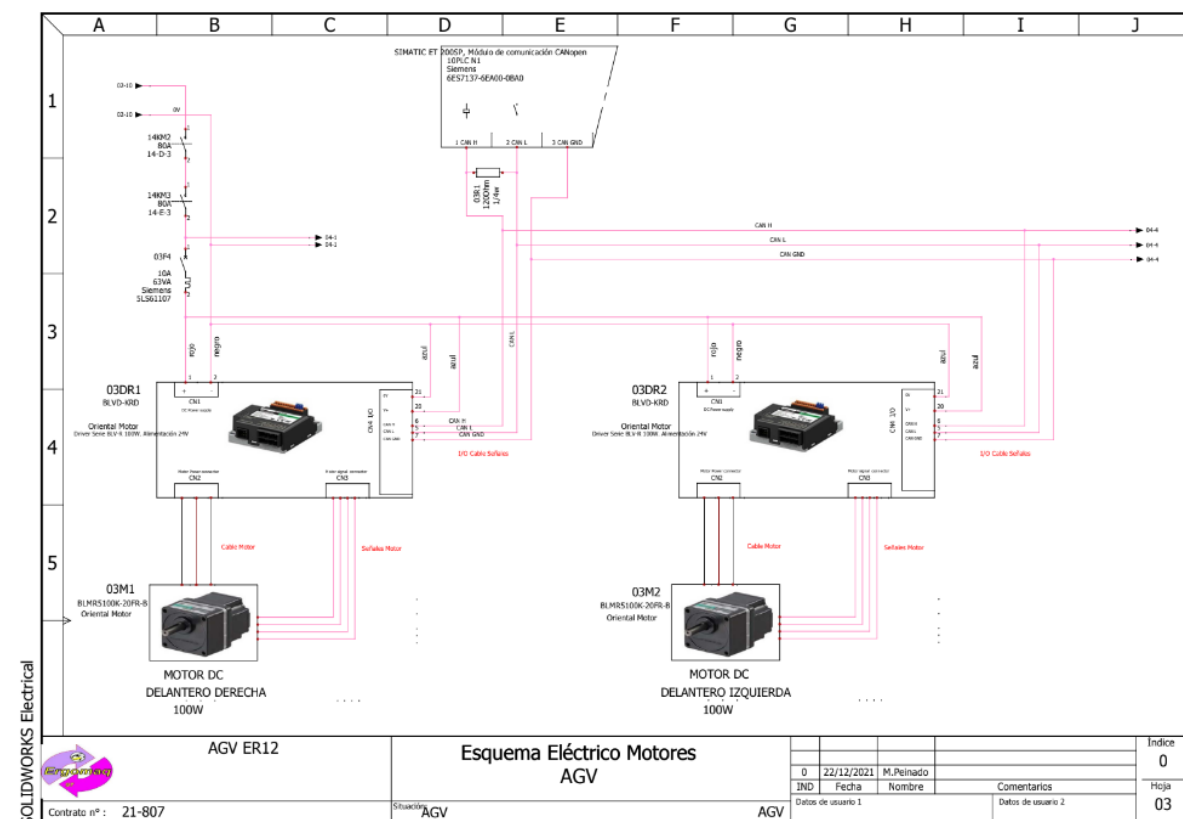


Imagen A3.2: Esquema eléctrico motores.

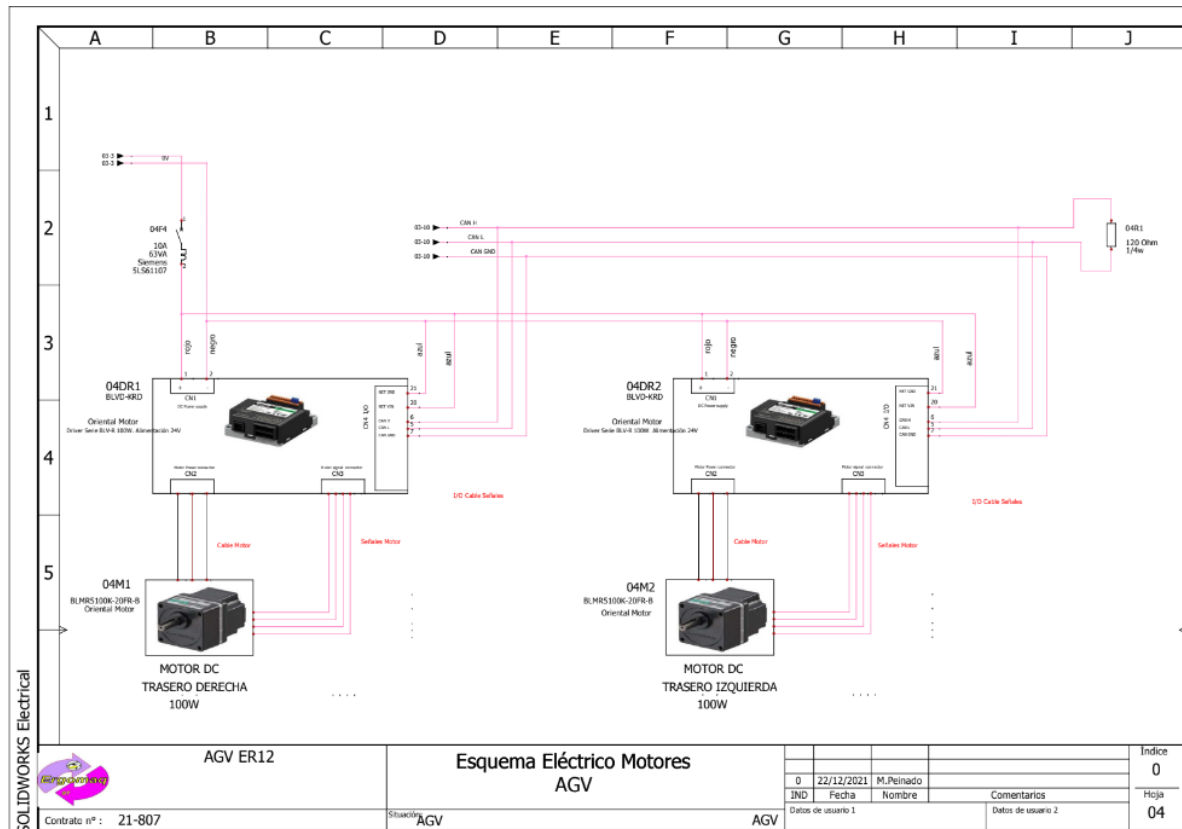


Imagen A3.3: Esquema eléctrico motores II.

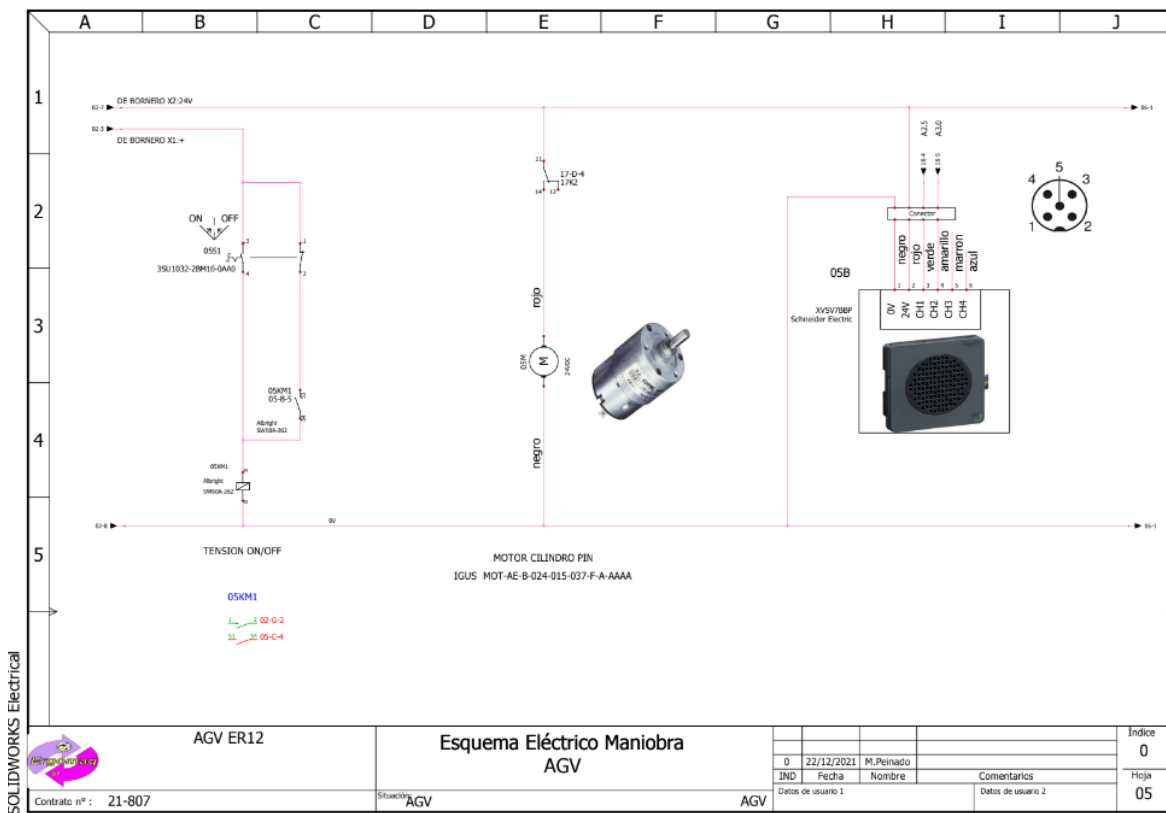


Imagen A3.4: Esquema eléctrico maniobra.

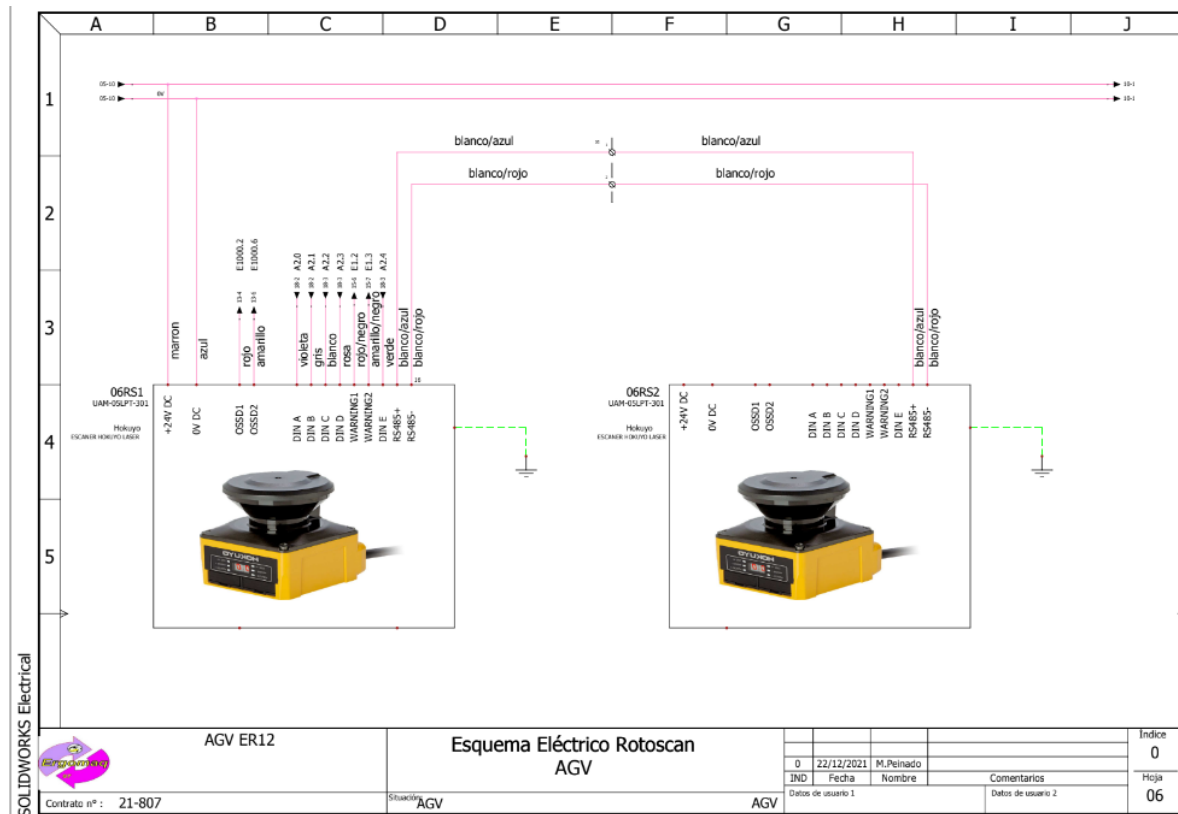


Imagen A3.5: Esquema eléctrico Rotoscan.

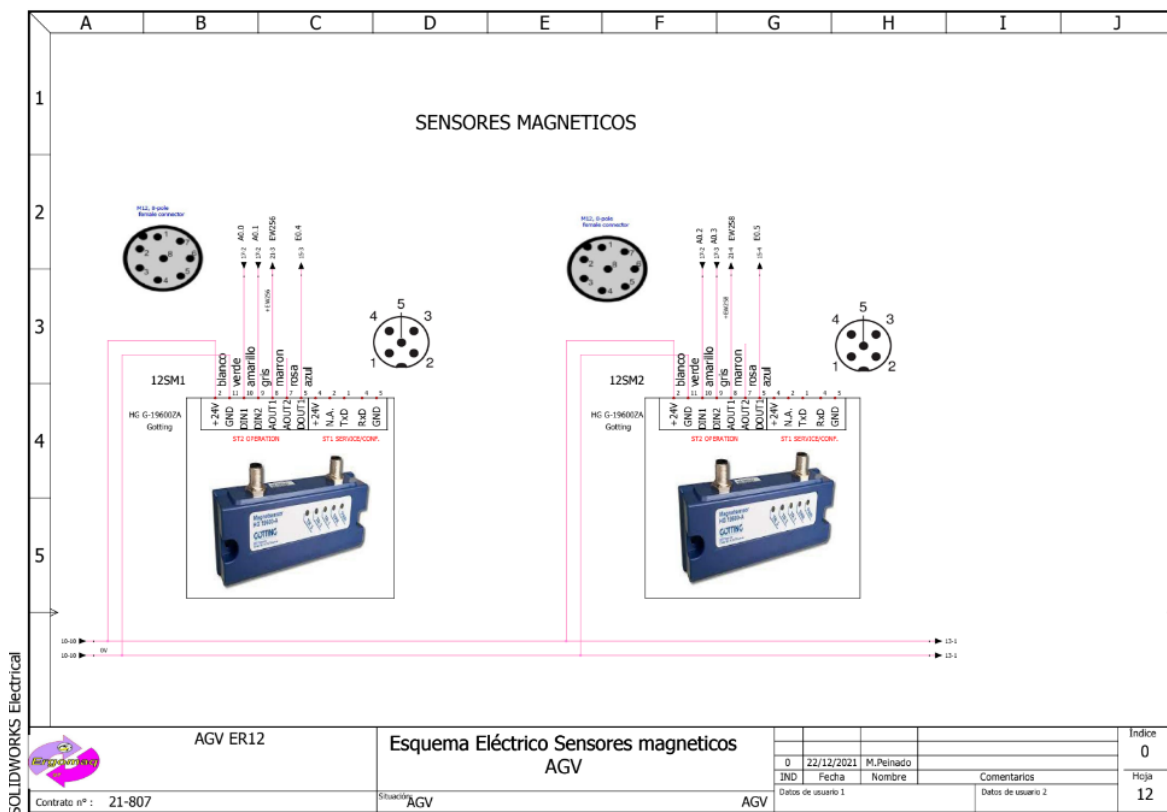


Imagen A3.6: Esquema eléctrico sensores magnéticos.

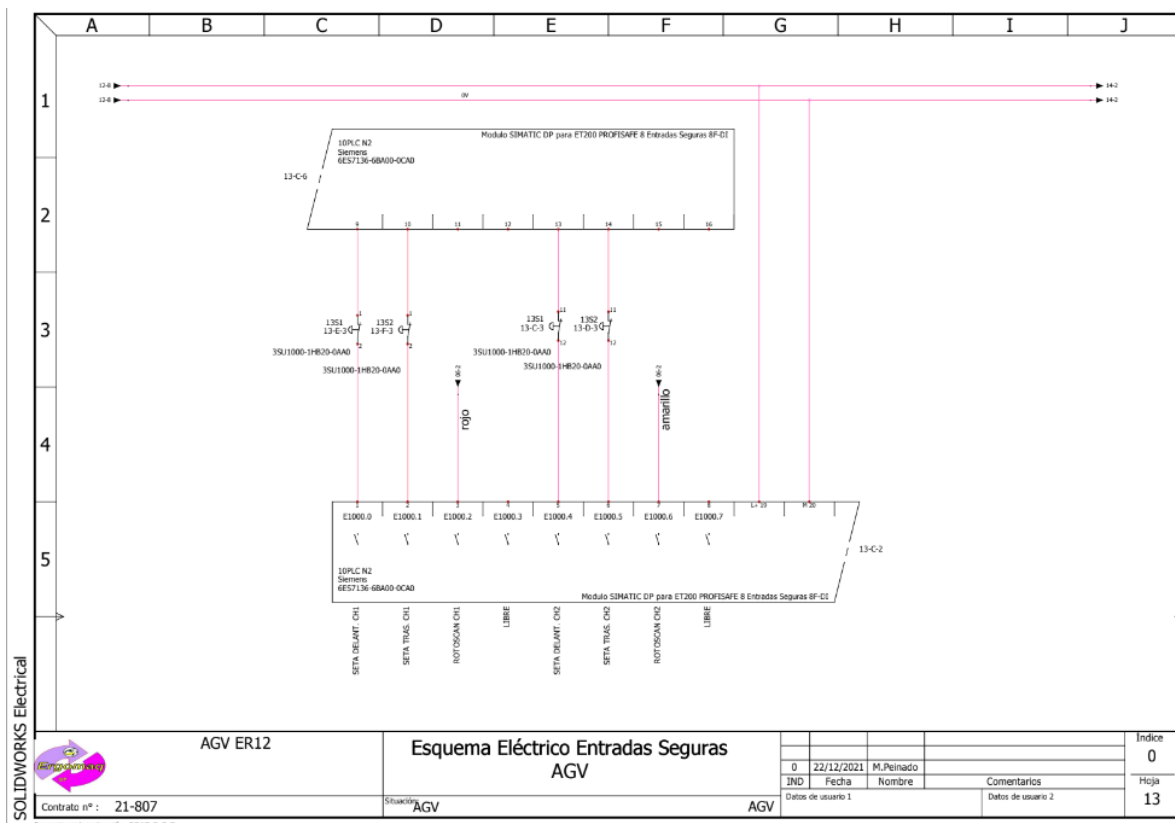


Imagen A3.7: Esquema eléctrico de entradas seguras..

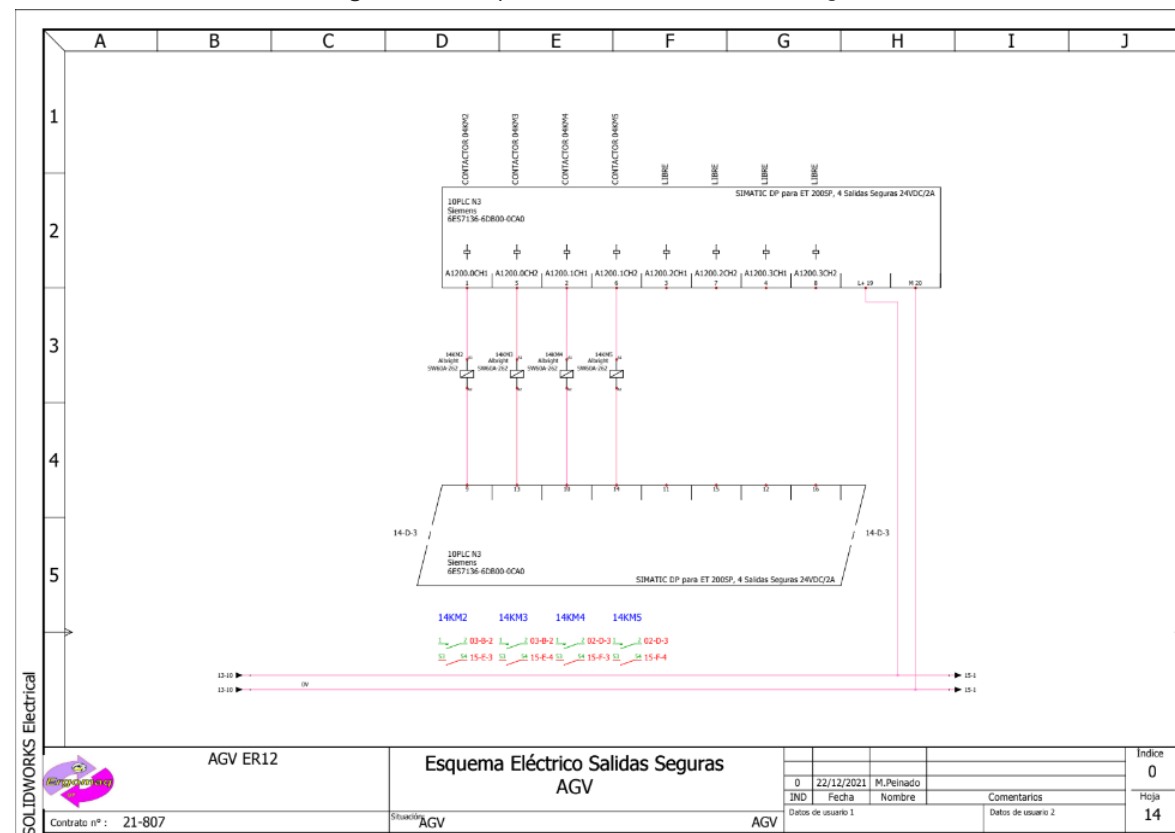


Imagen A3.8: Esquema eléctrico de salidas seguras..

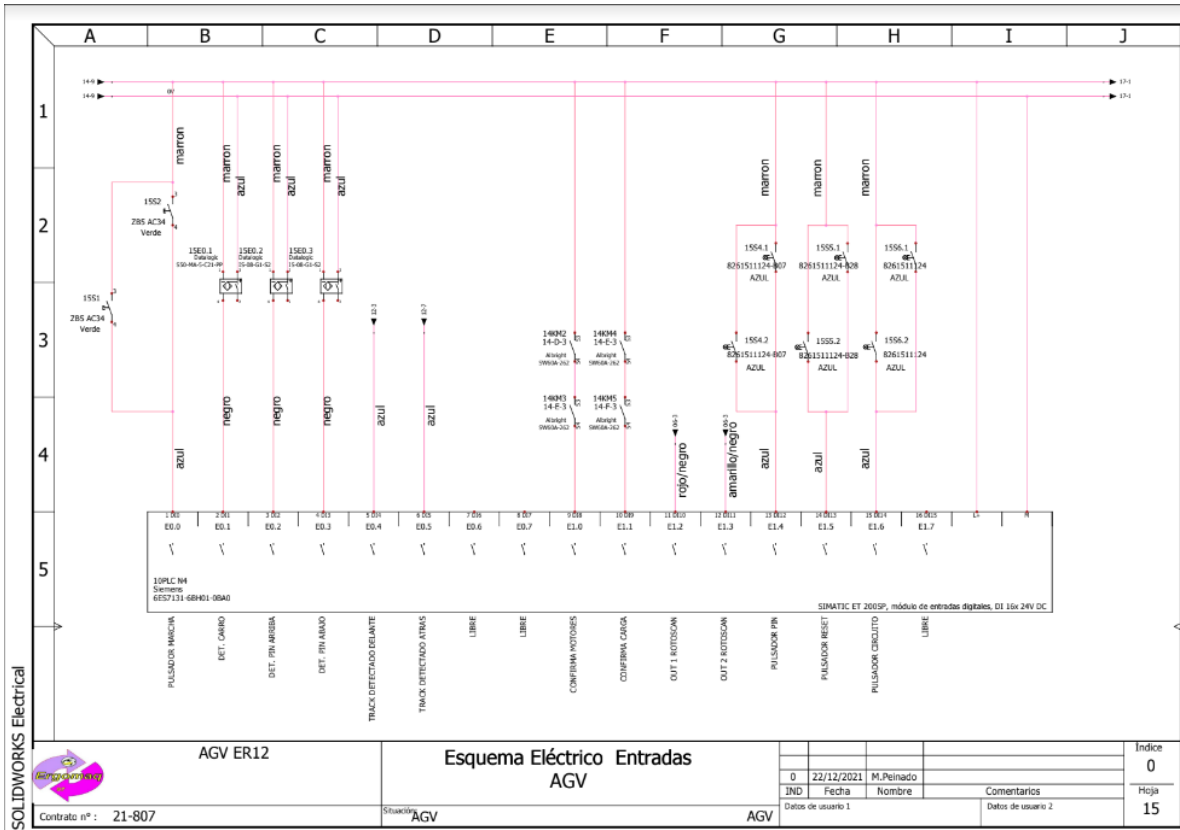


Imagen A3.9: Esquema eléctrico de entradas.

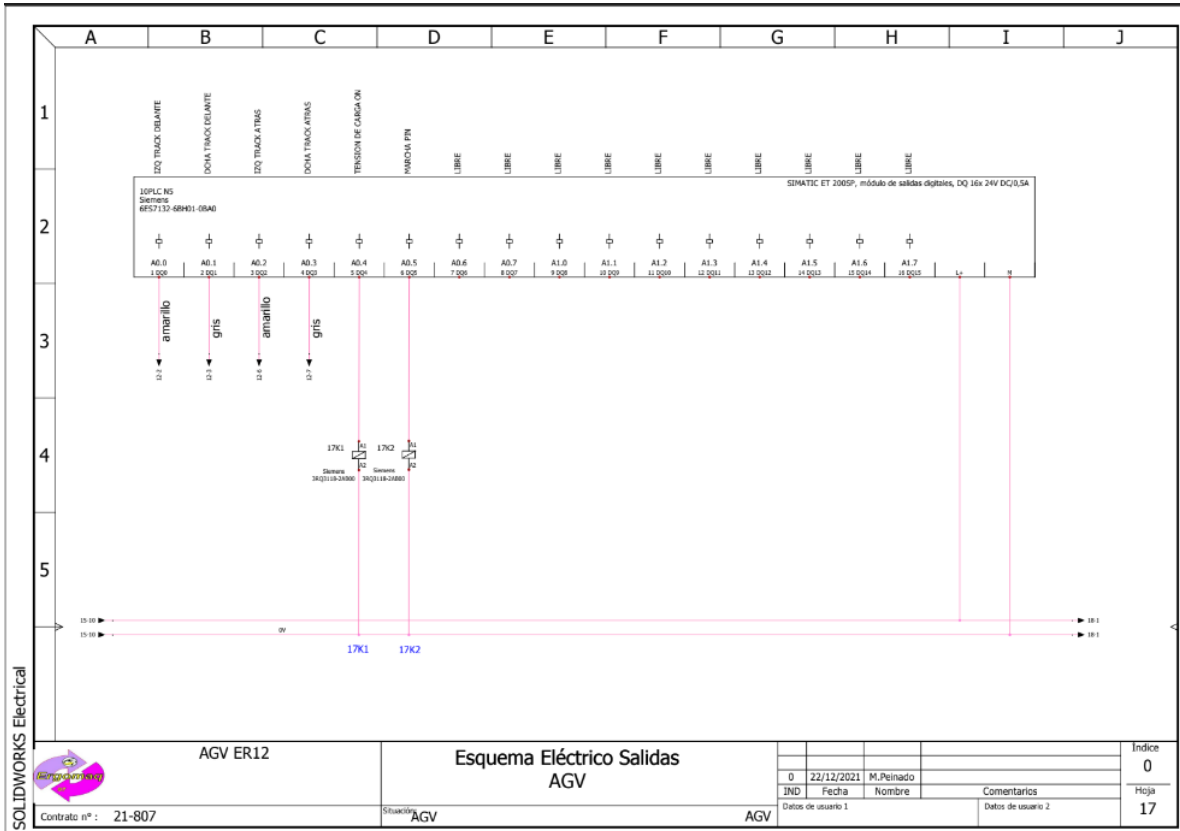


Imagen A3.10: Esquema eléctrico de salidas.

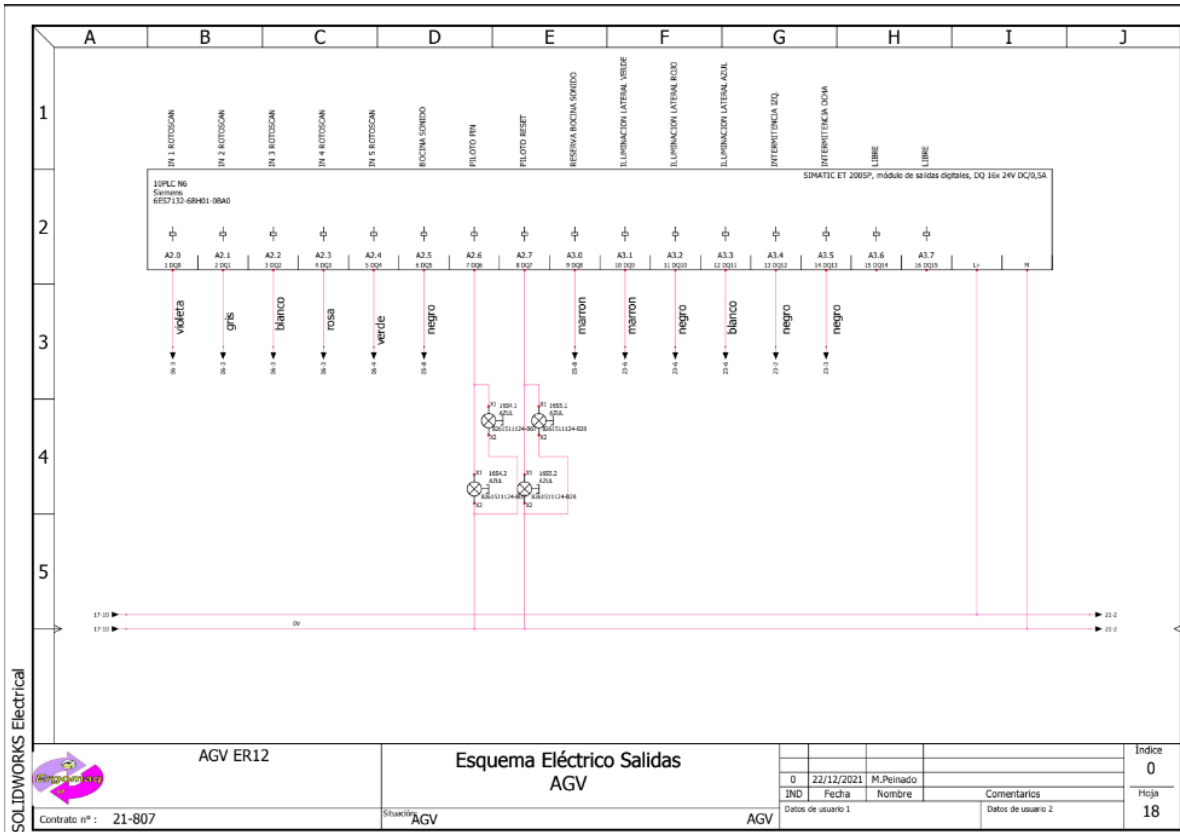


Imagen A3.11: Esquema eléctrico de salidas II.

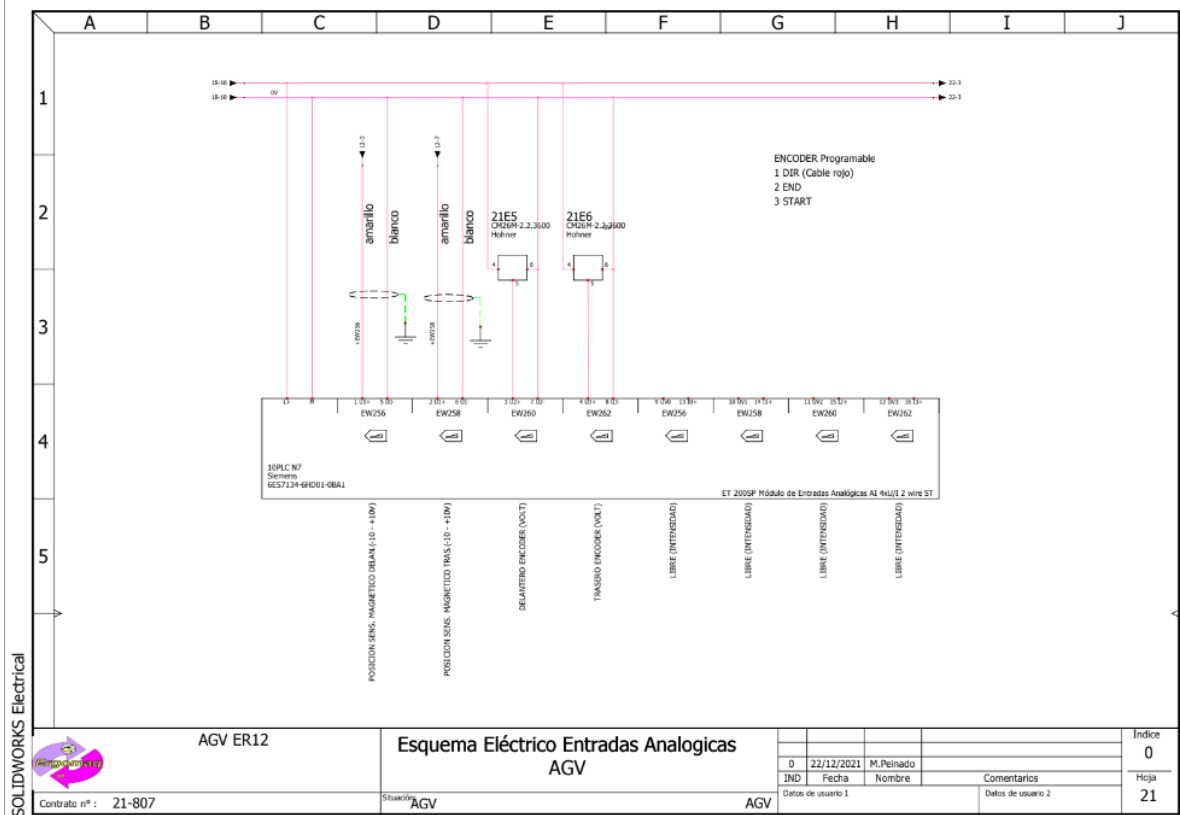


Imagen A3.12: Esquema eléctrico entradas analógicas.

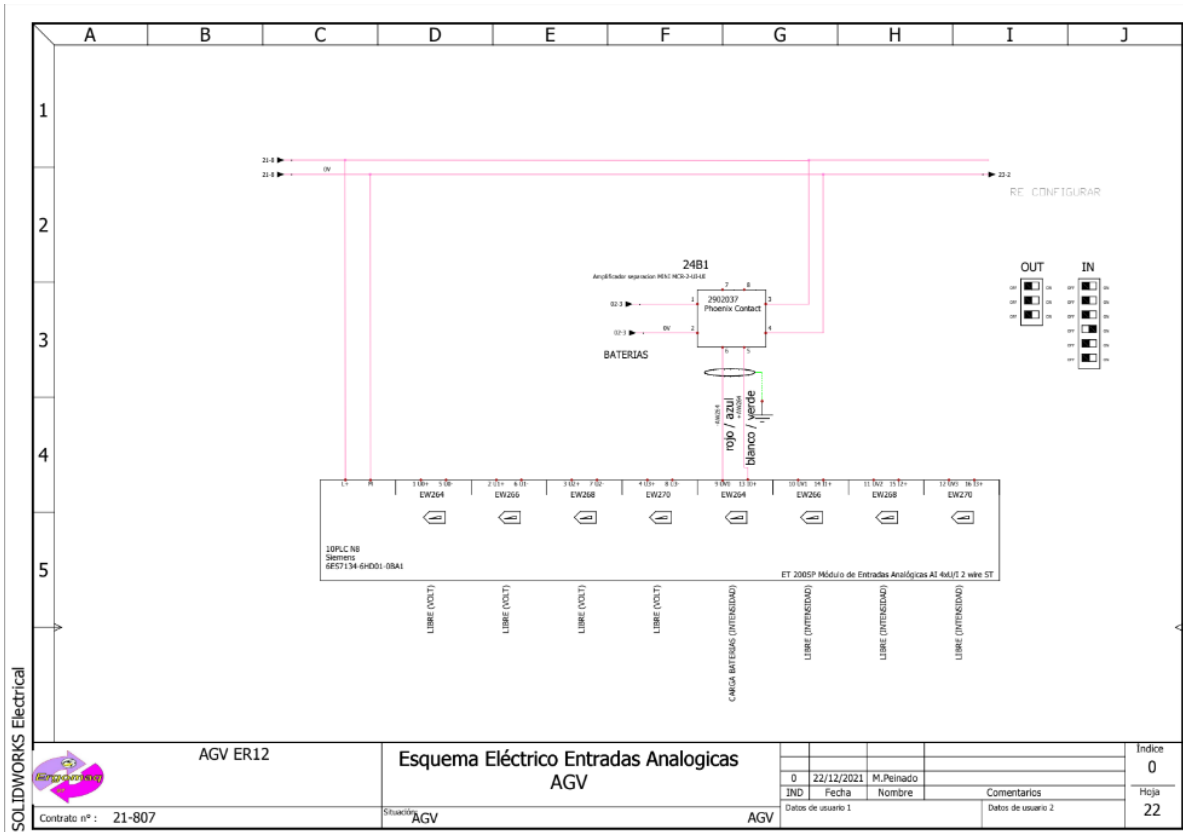


Imagen A3.13: Esquema eléctrico entradas analógicas II.

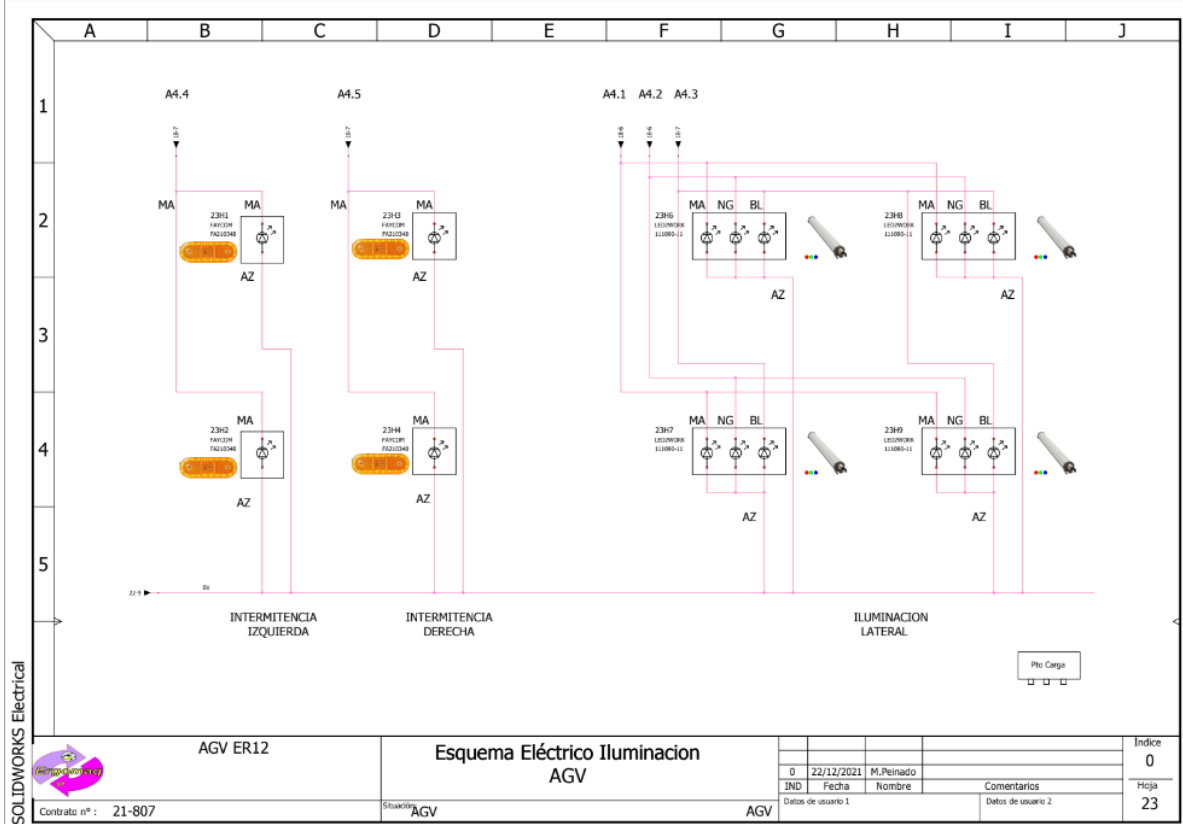


Imagen A3.14: Esquema eléctrico iluminación.

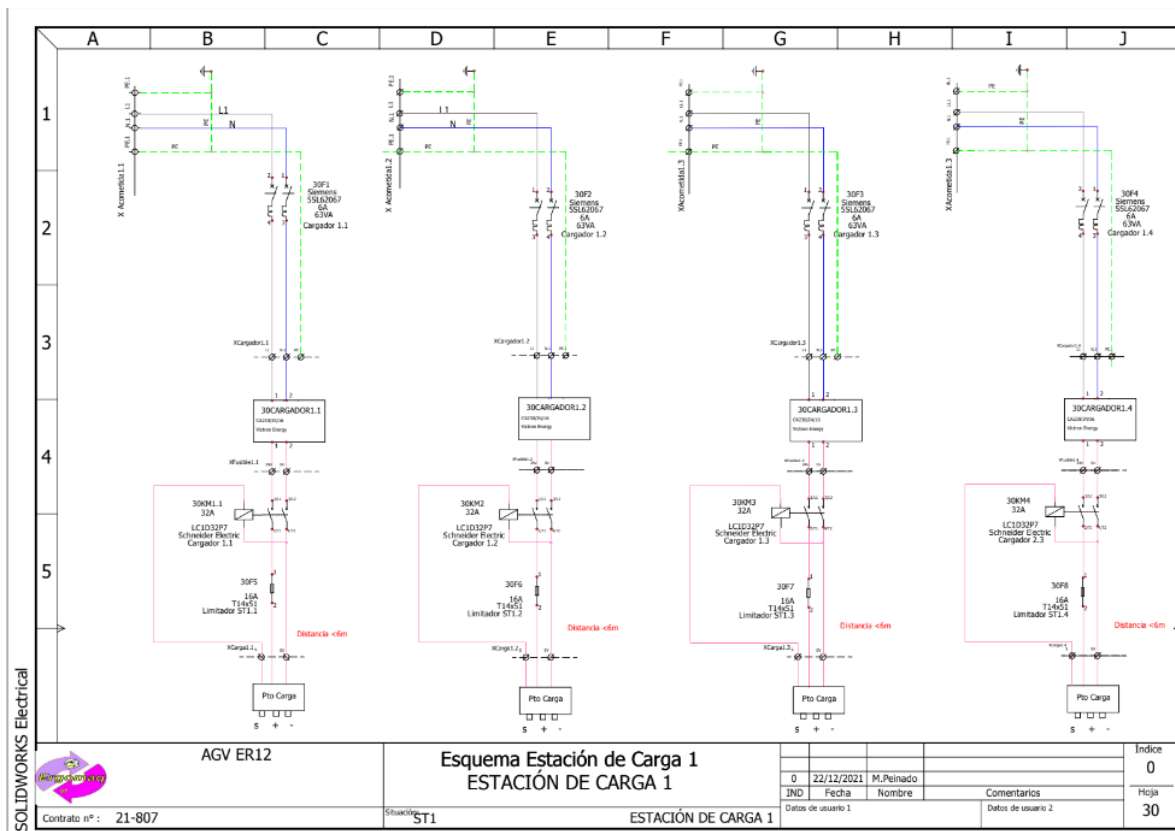


Imagen A3.15: Esquema eléctrico estación de carga.

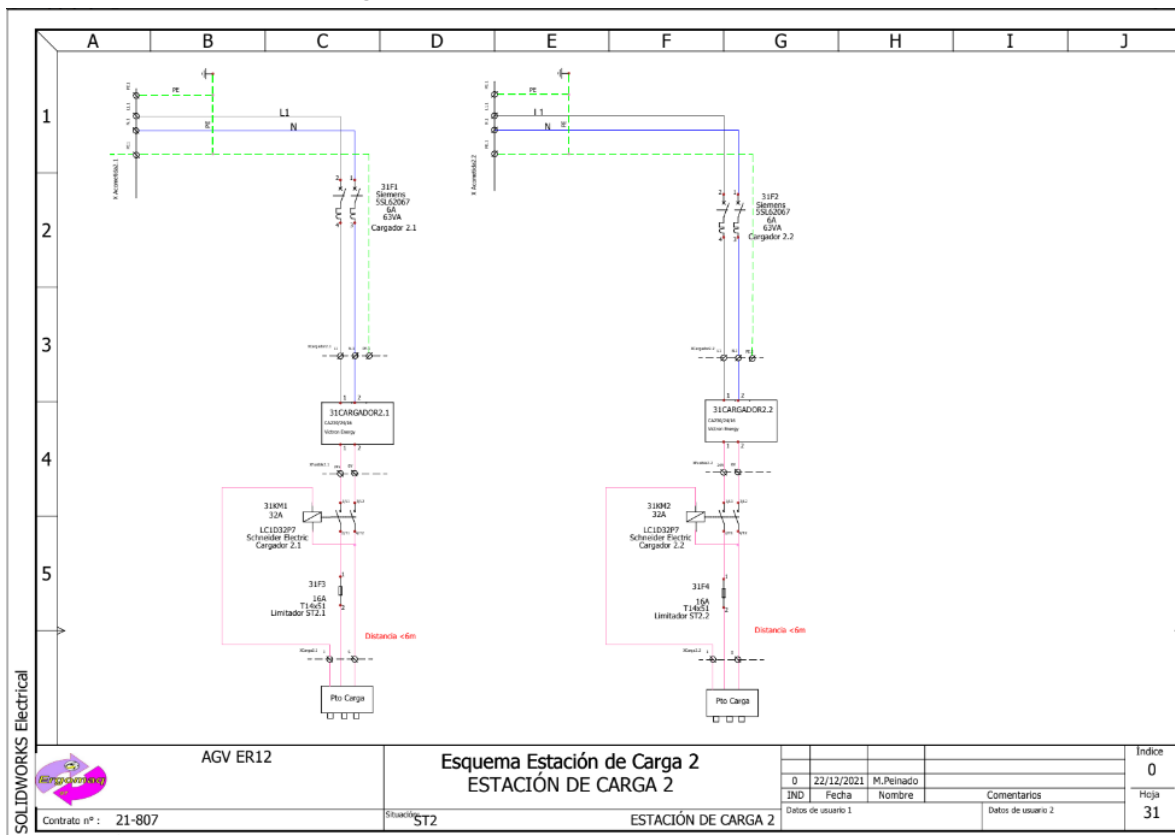


Imagen A3.16: Esquema eléctrico estación de carga II.

Anexo 4: Variables en la clase PLC.

•**Lista_Ruta (Array[0...5] de short):** Lista de identificadores “número real”. Representan los destinos activos a los que tiene que llegar el AGV. Cada posición en el array es el siguiente punto del circuito donde tiene que dirigirse el AGV, cronológicamente, con respecto a la posición en el array. Es decir, si la primera posición del array es un 5, y la siguiente es un 12; deberá dirigirse primero al tag en el circuito que corresponda con el “número real” 5, y seguidamente se dirigirá al que le corresponda el 12.

Una vez detecta que ha llegado al primer destino, es decir, la primera posición del array, automáticamente, el programa interno que tiene instalado el PLC desplazará hacia la izquierda el array, y se incluirá un 0 en la última posición del mismo. Cuando la primera posición del array es 0, el AGV no tiene más destinos, por lo que se para.

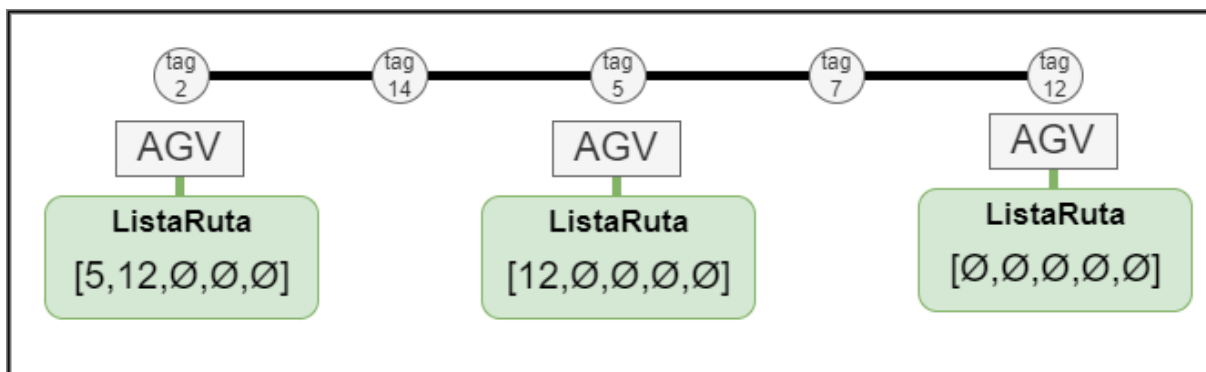


Imagen A4.1: Ejemplo de la variable ListaRuta cuando un AGV pasa por sus destinos.

El array tiene 5 posiciones, pero sólo se van a utilizar una o dos de ellas. No se va a dar una ruta con más de 2 paradas, este array más grande nos indica que en un futuro no se descarta que pueda haber más destinos. Dependiendo del circuito que se crea, o de las peticiones que deben de producirse, el sistema está pensado para poder escalar este array a más si fuese necesario, esta implementación no costaría mucho trabajo.

Además, dado que se trata de “shorts”, no ocupan casi memoria, por lo que podemos determinar esta decisión como válida y eficiente.

La variable ListaRuta puede leerse y escribirse por la aplicación. Es la razón por la que un AGV se mueve, y será la aplicación de gestión la que calcule e inserte la ruta a seguir en el AGV que corresponda.

•**porcentaje_bateria (entero):** Indica el porcentaje actual de la batería del AGV. El porcentaje de batería de un AGV se tiene en cuenta a la hora de aceptar o no ciertas peticiones, dado que no sería efectivo que un AGV se quedara sin batería en mitad de una petición, por lo que será una variable de lectura por parte de la aplicación que ayudará a tomar esta serie de decisiones.

•**Cargando_bateria (booleano):** Indica si el AGV se encuentra sobre un tag con información de carga de batería. Es decir, está cargando su batería. Gracias a esta variable de lectura por parte de la aplicación, se podrá saber qué AGVs están cargando, y tomar decisiones.

•**CruceON (booleano):** Variable únicamente de lectura por parte de la aplicación. El programa interno del PLC activará automáticamente la variable en el PLC cuando el AGV se encuentre sobre un tag con información de bloqueo, es decir, hay una petición de acceso a una zona de acceso en exclusión mutua. Esta variable está relacionada con la siguiente, “CruceOK”.

•**CruceOK (booleano):** Variable únicamente de escritura por parte de la aplicación. Cuando se activa, el AGV continúa por la zona de acceso en exclusión mutua a la que quería acceder. La aplicación debe gestionar el acceso en exclusión mutua que se ha comentado anteriormente, por lo que deberá conocer si existe algún AGV accediendo a esa sección crítica. Por lo tanto, cuando la aplicación lea que “CruceON” está activa en un PLC, comprobará si puede activar esta variable para dejarle pasar, o de lo contrario, tendrá que esperar a que el AGV que se encuentra en la sección crítica, salga de ella. Nos enteramos de que ha salido cuando pase por el tag de desbloqueo de esa zona en exclusión mutua. En ese momento activará esta variable para dejar al AGV pasar.

La siguiente sucesión de imágenes simula una situación real. Se muestran las variables de la clase PLC de las que estamos hablando, y las variables que influyen de la clase Nodo en una situación de bloqueo y desbloqueo. Como vemos, los tag de bloqueo y desbloqueo tienen un identificador específico para ello. Además de este identificador, el tag de bloqueo dispone de un campo que indica si la zona está libre, es decir, puede entrar un AGV, o no.



Imagen A4.2: Momento previo a la llegada al tag de bloqueo.

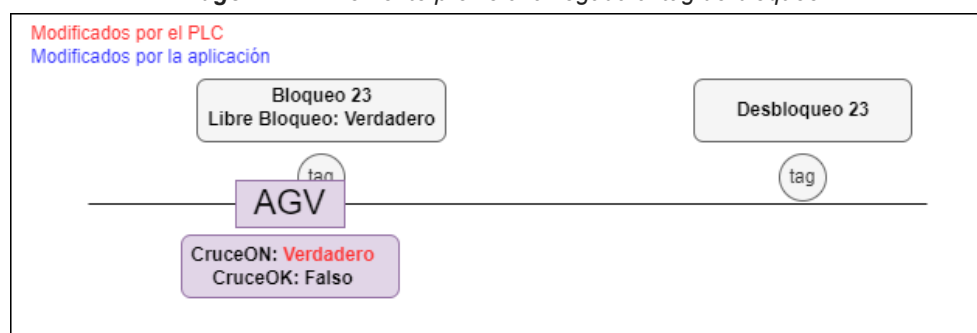


Imagen A4.3: Llegada al tag de bloqueo.

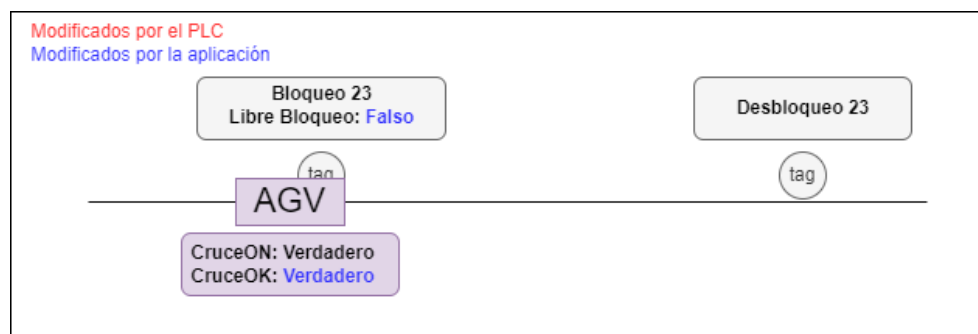


Imagen A4.4: Aceptación de la entrada a la zona de exclusión mutua.

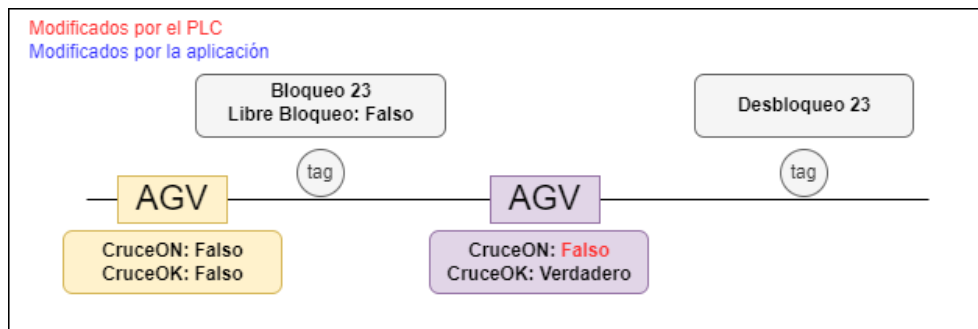


Imagen A4.5: Momento previo a la llegada del AGV amarillo al tag de bloqueo.

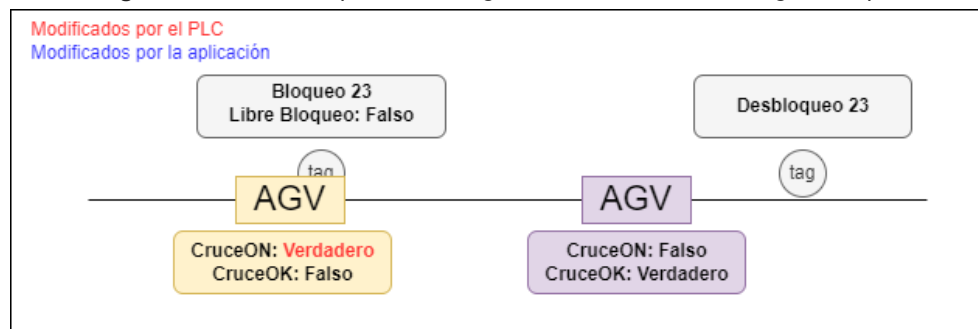


Imagen A4.6: AGV amarillo espera a que la zona quede libre.

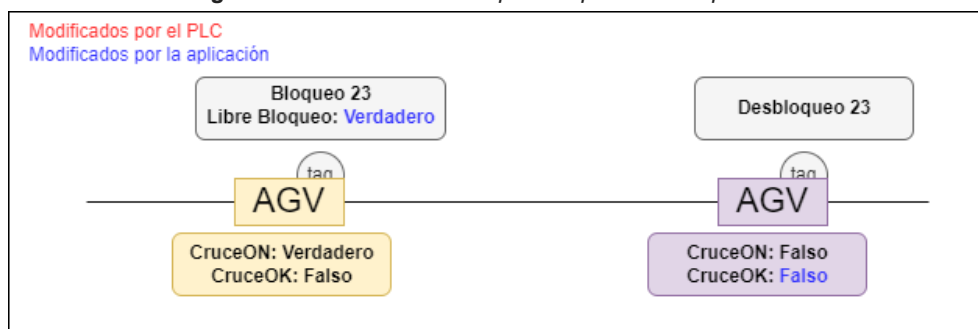


Imagen A4.7: AGV morado sale de la zona de exclusión mutua.



Imagen A4.8: Aceptación de la entrada a la zona de exclusión mutua para el AGV amarillo.

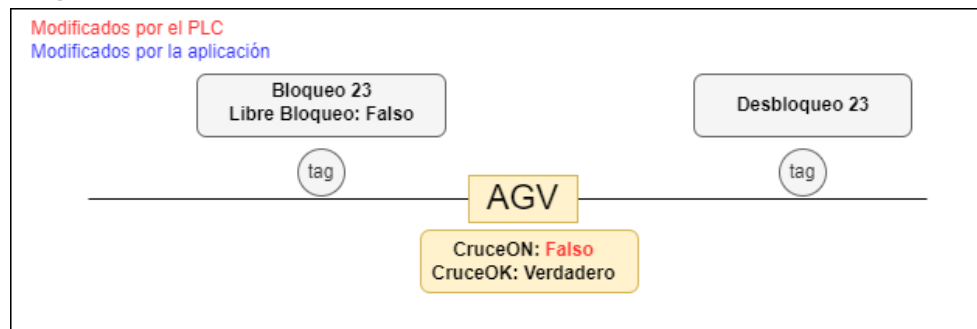


Imagen A4.9: AGV amarillo circula por la zona de exclusión mutua.

·**Pin_Abajo (booleano):** Indica si el Pin se encuentra abajo. Nos referimos como “Pin” al cilindro situado en la zona de arriba del AGV que ayuda a mover los carritos mediante fricción. Es una variable de lectura por parte de la aplicación.

·**Pin_Arriba (booleano):** Indica si el Pin se encuentra arriba. Es una variable de lectura por parte de la aplicación.

·**Direccion_Adelante (booleano):** Indica si el AGV va hacia delante. Si esta variable está activa, se va a estar leyendo por medio del lector de tags RFID que se encuentra en la parte delantera del AGV y va a estar cargando su batería por medio de la pastilla de carga incluida en su parte delantera.

·**Dirección_Atras (booleano):** Indica si el AGV va hacia atrás actualmente. Si esta variable está activa, se va a estar leyendo por medio del lector de tags RFID que se encuentra en la parte trasera del AGV y va a estar cargando su batería por medio de la pastilla de carga incluida en su parte trasera.

·**EnMovimiento (booleano):** Indica si el AGV se está moviendo. Esta variable puede ayudar a detectar la situación de un AGV y es de lectura por parte de la aplicación.

·**Velocidad (número real):** Indica la velocidad en revoluciones por minuto a la que se mueve el AGV. Es meramente informativa y únicamente de lectura por parte de la aplicación.

·**recirculacion (booleano):** Indica que un AGV debe recircular. La aplicación debe estar leyendo esta variable para que cuando se active, enviar a dicho AGV a recircular.

•**tiempo_recirculacion (time)**: Tiempo máximo que puede pasar un AGV cargando sin recircular. Cuando un AGV comienza a cargar, el programa interno del PLC establece un tiempo estático, que por defecto es de 2 minutos, y lo asigna a esta variable. Este tiempo va decreciendo, y si llega a 0, se activa la variable de recirculación del AGV, y comienza a recircular.

•**ID (double)**: Identificador “tag ID” del último TAG leído por el AGV. De esta manera se puede situar a cada AGV en una posición del circuito. Este identificador leído es el identificador “tag ID”, no el “número real”.

•**ModoAutomatico (booleano)**: El AGV puede estar en 2 modos: automático y manual. Cuando está en modo automático, se mueve automáticamente, como se espera, pero al estar en modo manual, únicamente será capaz de desplazarse manualmente por un trabajador en el entorno industrial, utilizando las pantallas HMI. La variable muestra el modo actual del AGV (modo manual cuando no está activo, modo automático cuando está activo).

Anexo 5: Algoritmos relacionados.

El algoritmo de Dijkstra [18], también llamado algoritmo de caminos mínimos, es un algoritmo para la determinación del camino más corto, dado un vértice origen, hacia el resto de los vértices en un grafo que tiene pesos en cada arista.

También podemos mencionar el algoritmo de *Bellman-Ford*. El algoritmo de Bellman-Ford [19] genera el camino más corto en un grafo dirigido ponderado (en el que el peso de alguna de las aristas puede ser negativo). El algoritmo de Dijkstra resuelve este mismo problema en un tiempo menor, pero requiere que los pesos de las aristas no sean negativos, salvo que el grafo sea dirigido y sin ciclos. Por lo que el Algoritmo Bellman-Ford normalmente se utiliza cuando hay aristas con peso negativo.

Un último algoritmo a mencionar, sería el algoritmo de *Floyd-Warshall*. En informática, el algoritmo de Floyd-Warshall [20], es un algoritmo de análisis sobre grafos para encontrar el camino mínimo en grafos dirigidos ponderados. El algoritmo encuentra el camino entre todos los pares de vértices en una única ejecución. El algoritmo de Floyd-Warshall es un ejemplo de programación dinámica.

Anexo 6: Implementación de las dos peticiones predeterminadas.

La primera petición consiste en traer un carrito vacío. Cuando un usuario se encuentra en el entorno de trabajo, en una zona de petición, y no dispone de ningún carrito para poder trasladar los objetos que desea, clicla en este botón. Esta petición debe llamar a la **primera función básica** para encontrar al AGV más cercano al almacén de carritos vacíos, coger uno de ellos, y llevarlo a la posición del usuario. Si ya existe un carrito en la zona de petición del mismo, cuando el usuario pulse el botón, se notificará al usuario de que no puede pedir un carrito porque ya dispone de uno. La siguiente secuencia de imágenes muestra qué ruta tomará un AGV que acepta esta petición.

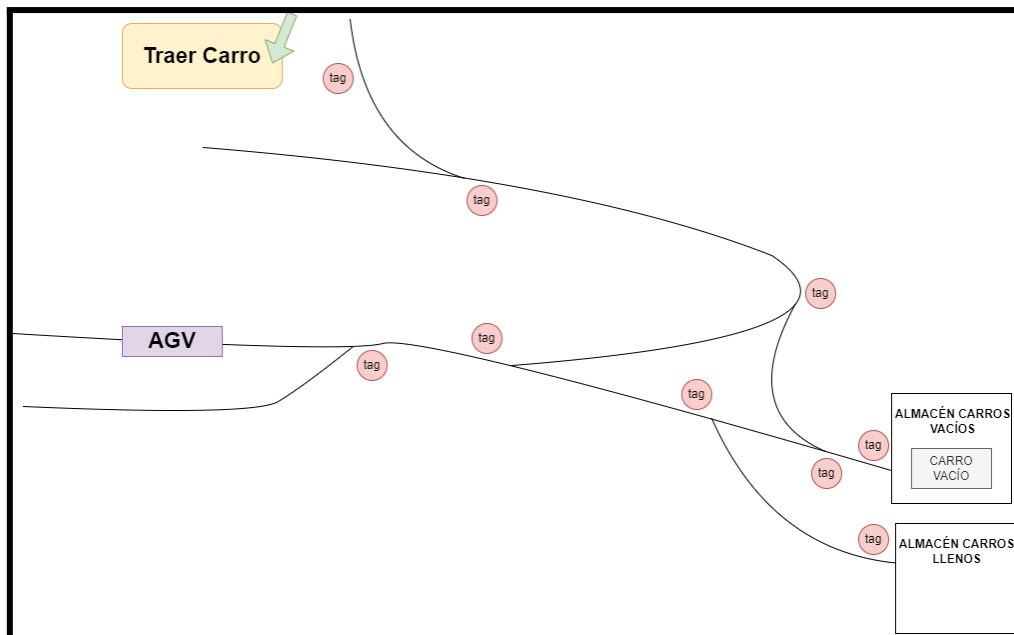


Imagen A6.1: Traer carrito. Primera fase. El usuario pulsa el botón.

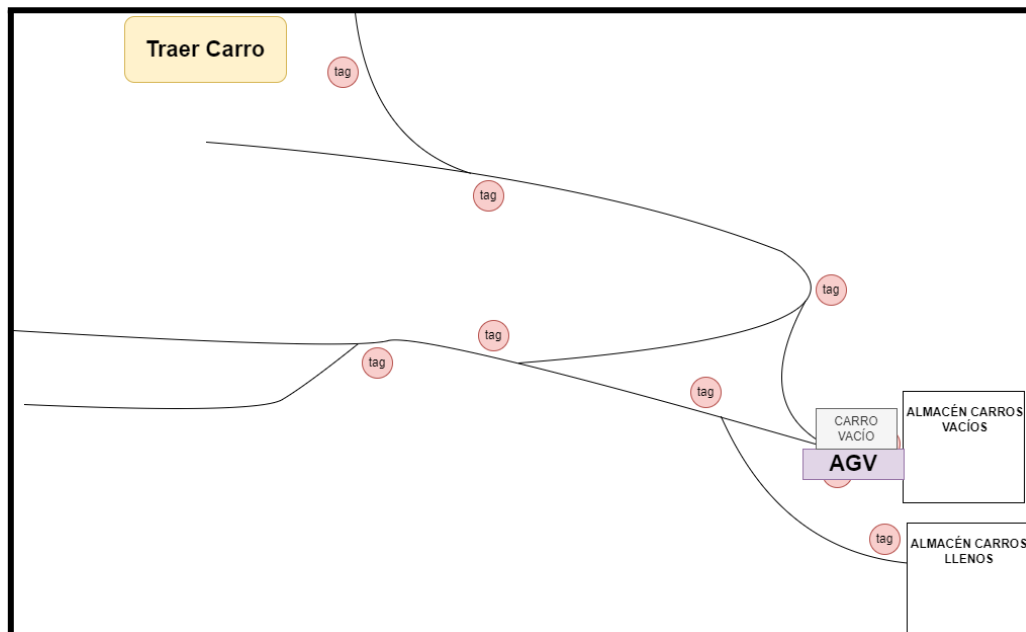


Imagen A6.2: Traer carrito. Segunda fase. Recogida del carrito vacío.

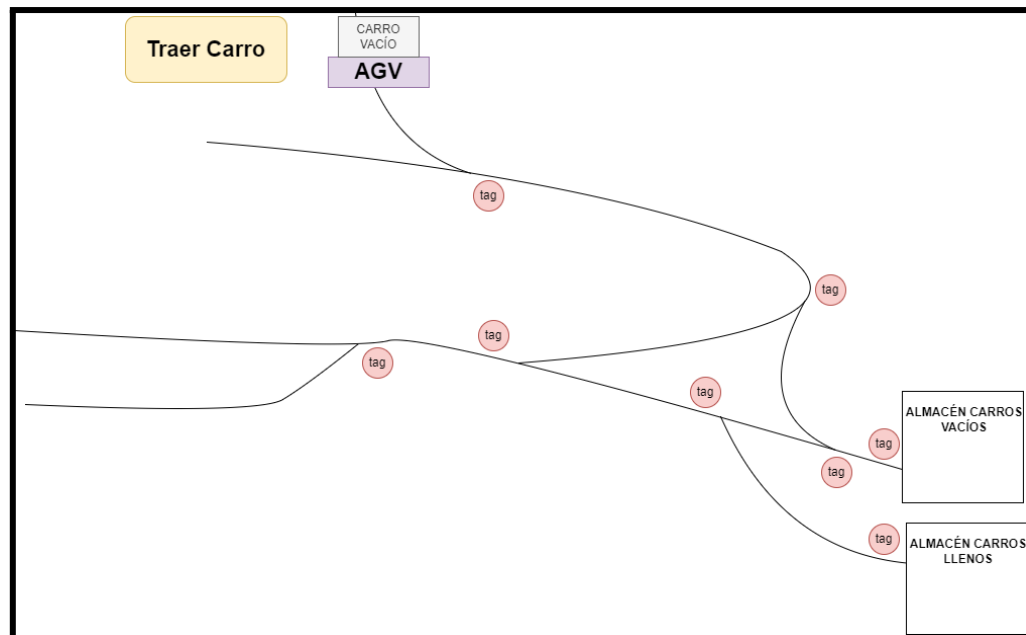


Imagen A6.3: Traer carrito. Tercera fase. Llegada del vehículo con el carrito a la zona de petición.

La segunda petición consiste en llevar un carrito lleno. Cuando el usuario se encuentra en una zona de petición, donde se encuentra un carrito el cual está listo para transportarse a un camión que se encuentra en la zona de almacenes llenos, el usuario clicca este botón. La aplicación utilizará la **primera función base** que buscará al AGV que se encuentre más cerca de la zona de petición, y lo trasladará ahí, para que recoja al carrito y termine llevándolo al almacén de carritos llenos. Cuando el AGV llega al primer destino, y coge el carrito, la aplicación se encargará de reponer esa zona que ha quedado vacía, con un nuevo carrito vacío. Se crea entonces una nueva petición de traer un carrito vacío. Si existieran más AGVs preparados para ello, en ese momento, un AGV aceptaría la petición y comenzaría a llevar un carrito vacío a la zona de petición. Si no existiera ningún AGV disponible, esta petición se quedaría en estado pendiente.

Si esta petición es lanzada y no hay ningún carrito en el lugar desde donde se lanza, se notificará al usuario de que es imposible realizarlo, dado que no existe ningún carrito. La siguiente secuencia de imágenes muestra qué ruta tomará un AGV que acepta esta petición. En esta secuencia, existe también otro AGV que aceptará la petición de traer un carrito vacío.

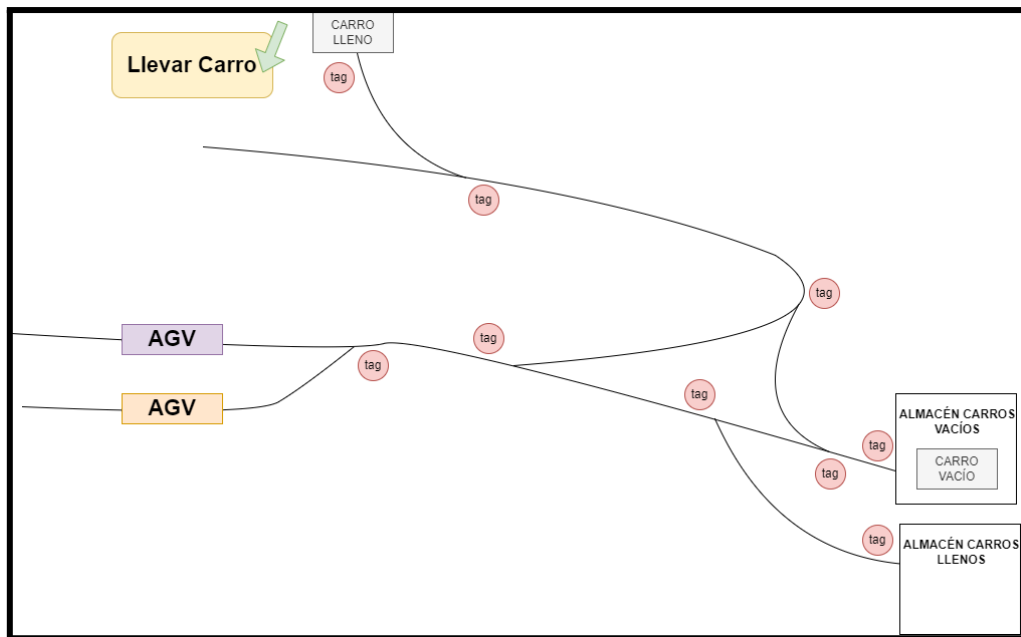


Imagen A6.4: Llevar carrito. Primera fase. El usuario pulsa el botón.

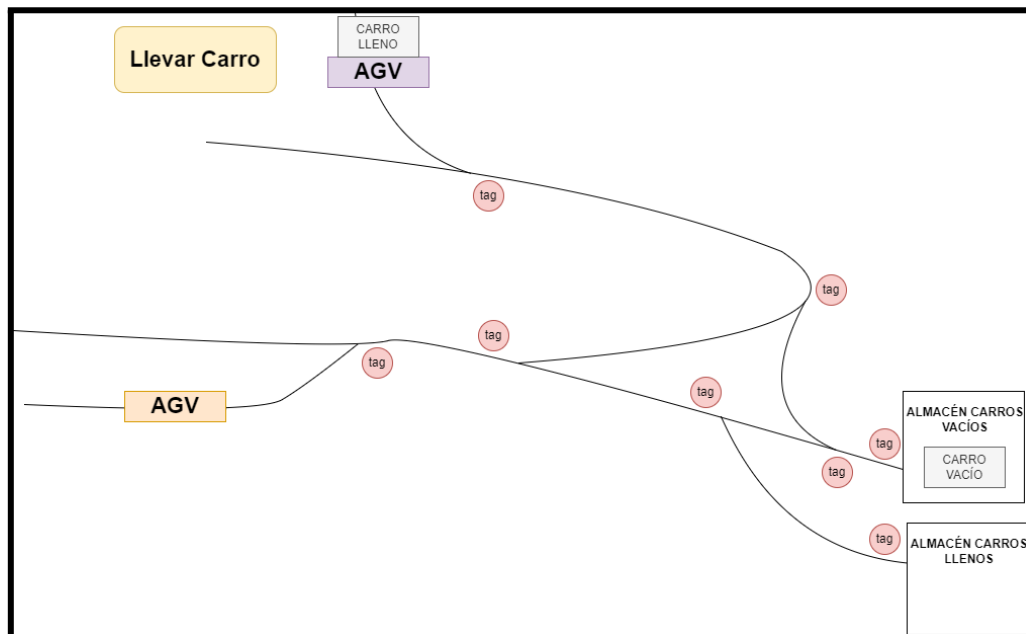


Imagen A6.5: Llevar carrito. Segunda fase. Recogida del carrito lleno, y por lo tanto se genera nueva petición de traer carrito vacío.

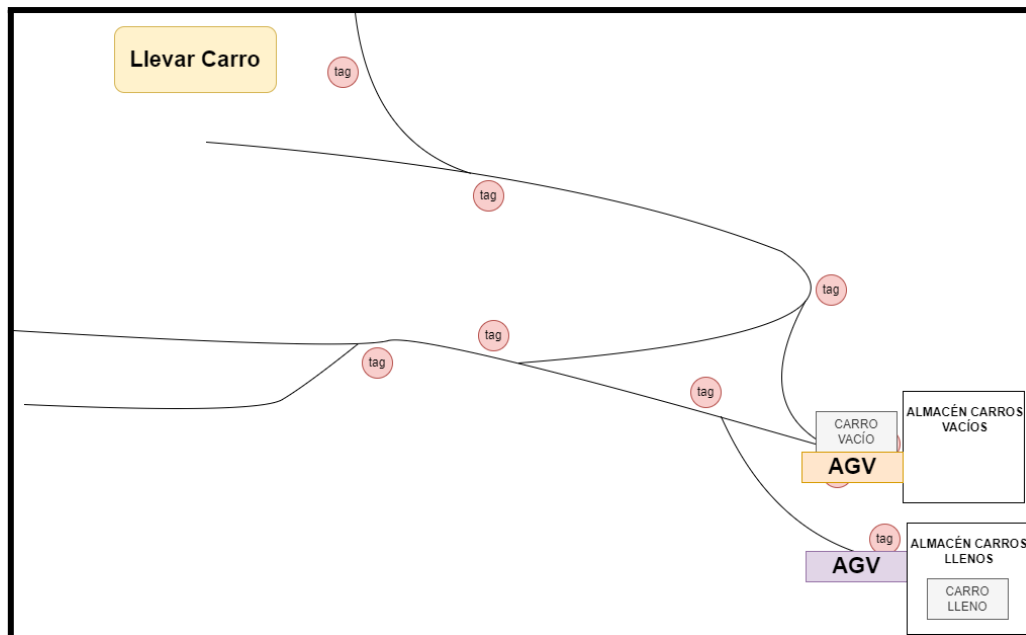


Imagen A6.6: Llevar carrito. Tercera fase. Recepción del carrito lleno por parte del AGV morado y recolección del carrito vacío por parte del AGV amarillo.

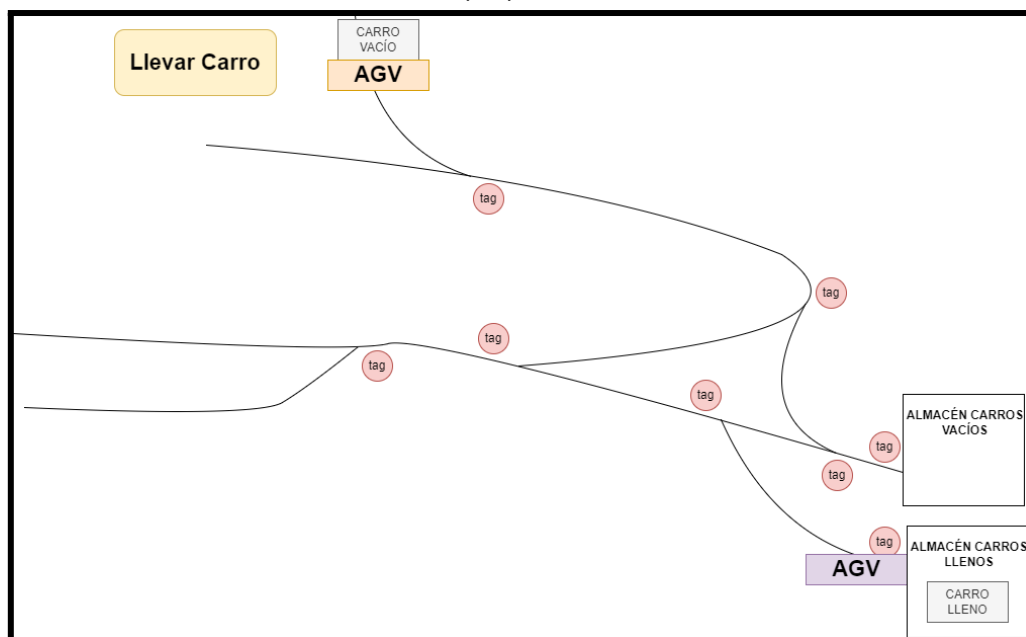


Imagen A6.7: Llevar carrito. Cuarta fase. Recepción del carrito vacío por parte del AGV amarillo.

Anexo 7: Ventajas de OPC UA.

Las ventajas de *OPC UA* con respecto de *OPC Clásico* [4], son los siguientes:

→ Descentralización: Históricamente, la pirámide de automatización en los sistemas industriales es una estructura jerárquica que describe el flujo de información desde los dispositivos de bajo nivel, como controladores, sensores o mediadores, hasta aplicaciones *ERP* de alto nivel. *OPC UA* elimina esta estructura piramidal al descentralizar los componentes del sistema y facilitar el uso de estructuras de modelo de datos más flexibles en una red de malla.

- Independencia de la plataforma: Gracias a *OPC UA* los sistemas industriales pueden integrar software de cualquier proveedor.
- Escalabilidad: Es un sistema preparado para el futuro que permite a las organizaciones desarrollar sistemas *SCADA* escalables.
- Descubrimiento: Tiene capacidad *plug-and-play*; cuando se agregan nuevas plantas remotas a una organización, *OPC UA* puede descubrir automáticamente sus redes, configurarlas e integrarlas en la red de la empresa.

Bibliografía.

- [1] Opiron. ¿Qué es OPC? (Visitado 19/11/2022)
<https://www.opiron.com/que-es-opc/>
- [2] Aula 21. Qué es OPC y cómo funciona. (Visitado 19/11/2022)
<https://www.cursosaula21.com/que-es-opc-y-como-funciona/>
- [3] Contaval. Conectividad IoT: De la automatización al Cloud. (Visitado 19/11/2022)
<https://www.contaval.es/conectividad-iot-de-la-automatizacion-al-cloud/>
- [4] Paessler. IT Explained: OPC UA. (Visitado 19/11/2022)
<https://www.paessler.com/es/it-explained/opc-ua>
- [5] C# Spot .NET. Qué es C#. (Visitado 19/11/2022)
<https://csharpspot.net/que-es-c-sharp/>
- [6] Velneo. ¿Qué es JavaScript? (Visitado 19/11/2022)
<https://www.velneo.com/blog/que-es-javascript>
- [7] Festo. TIA Portal. (Visitado 19/11/2022)
https://www.festo.com/es/es/e/tendencias/tia-portal-id_828990/
- [8] Espacio Honduras. Concepto de Microsoft Visual Studio. (Visitado 19/11/2022)
<https://www.espaciohonduras.net/microsoft-visual-studio-concepto-y-que-es-y-para-que-sirve-microsoft-visual-studio>
- [9] Wikipedia. ASP.NET. (Visitado 19/11/2022)
<https://es.wikipedia.org/wiki/ASP.NET>
- [10] Microsoft. Una introducción a NuGet. (Visitado 19/11/2022)
<https://learn.microsoft.com/es-es/nuget/what-is-nuget>
- [11] SQLShack. SQL Server Management Studio - Una guía de instalación paso a paso. (Visitado 19/11/2022)
<https://www.sqlshack.com/es/sql-server-management-studio-una-guia-de-instalacion-paso-a-paso>

- [12] Wikipedia. Grafo. (Visitado 19/11/2022)
<https://es.wikipedia.org/wiki/Grafo>
- [13] Wikipedia. Multigrafo. (Visitado 19/11/2022)
<https://es.wikipedia.org/wiki/Multigrafo>
- [14] Microsoft. Información general de ASP.NET. (Visitado 19/11/2022)
<https://learn.microsoft.com/es-es/aspnet/overview>
- [15] Microsoft. Introducción a la programación web de ASP.NET. (Visitado 19/11/2022)
<https://learn.microsoft.com/es-es/aspnet/web-pages/overview/getting-started/introducing-razor-syntax-vb>
- [16] Tech Riders. Variable Session en ASP.NET. (Visitado 19/11/2022)
<https://techriders.tajamar.es/variable-session-en-asp-net/>
- [17] Tutoriales programación ya. Master Page. (Visitado 19/11/2022)
<https://www.tutorialesprogramacionya.com/aspnetya/detalleconcepto.php?codigo=72>
- [18] Wikipedia. Algoritmo de Dijkstra. (Visitado 19/11/2022)
https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra
- [19] Wikipedia. Algoritmo de Bellman-Ford. (Visitado 19/11/2022)
https://es.wikipedia.org/wiki/Algoritmo_de_Bellman-Ford
- [20] Wikipedia. Algoritmo de Floyd-Warshall. (Visitado 19/11/2022)
https://es.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall