Universidad del País Vasco Euskal Herriko Unibertsitatea

BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

MÁSTER UNIVERSITARIO EN INGENIERÍA DE CONTROL, AUTOMATIZACIÓN Y ROBÓTICA

TRABAJO FIN DE MÁSTER

CONTROL DE ESTACION ROBOTIZADA CON LAS LIBRERÍAS MXAUTOMATION. REGISTRO DE DATOS EN LA NUBE CON PROTOCOLO MQTT

Estudiante *Manzanares Dominguez, Ignacio*
Director *Orive Revillas, Darío*
Departamento *Ingeniería de Sistemas y Automática*
Curso académico *2021/2022*

Bilbao, 20, 12, 2021

RESUMEN TRILINGÜE

Resumen: En el nuevo paradigma introducido por la industria 4.0, es cada vez mas importante disponer de procesos de fabricación flexibles, que sean capaces de adaptarse a la demanda en un corto espacio de tiempo, con menos tiempos de paro, y mas predecibles.

Para ello es importante disponer de herramientas de programación más estandarizadas, que permitan una abstracción respecto a las ofrecidas por cada fabricante, y que ahorren tiempo y complicaciones a los programadores encargados del desarrollo y mantenimiento del código.

También con este objetivo se otorga una creciente importancia a la recolección de datos, que permiten conocer mejor el estado de los sistemas, adelantarse a fallos, y conocer con precisión que elemento es el que da problemas.

En este proyecto se dispone de un controlador SIMATIC ET 200SP Open Controller cuyas capacidades permiten cumplir ambas tareas. Mediante la utilización de las librerías mxAutomation se podrá realizar la programación de un robot serie de seis ejes directamente en el PLC, sin necesidad de recurrir a programar su controlador propio. También, empleando la funcionalidad PC de dicho controlador, se pueden recoger datos del robot y subirlos a una base de datos en la nube.

Abstract: As Industry 4.0 introduces a new paradigm, greater importance is given to the implementation of flexible manufacturing processes capable of swiftly adapting to demand, more predictable and with less stop times.

In order to achieve that, it is important to have standardized programming tools, allowing a greater level of abstraction from the tools from each manufacturer. This would help programmers, reducing complexity and saving time.

Data acquisition and analysis becomes also a great way to achieve this objective. By having a deeper knowledge of the insides of our systems, preventive maintenance can be achieved, and failures can be pinpointed.

The SIMATIC ET 200SP Open Controller used in this project allows the completion of both tasks. Using mxAutomation libraries, an will be able to program a six axis robot in the PLC directly, not needing to program in the robot's controller. Moreover, the PC functionality of this controller can be used to acquire data and deploy it into a cloud-based database.

Laburpena: 4.0 industriak sortutako paradigma berrian, fabrikazio-prozesu malguak izatea gero eta garrantzitsuagoa da, aldi laburretan eskakizunera moldatzeko gai diren prozesuak, geldialdi gutxiago eta aurreikusteko moduko errazagoak.

Horretarako programaziorako tresna estandarizatuenak izatea da gakoa, fabrikatzaile bakoitzak eskainitakoen aldean abstrakzioa onartzen dutenak, eta kode garapen eta mantentze-lanetarako arduradunei denbora eta zailtasunak aurrezten dutenak.

Helburu horrekin, gainera, gero eta garrantzi handiagoa ematen zaio datu-bilketari, sistemen egoera hobeto ezagutzeko, akatsei aurrea hartzeko eta arazoak zer elementuk ematen dituen zehatz-mehatz jakiteko aukera ematen baitute.

Proiektu honetan, SIMATIC ET 200SP Open Controller kontrolatzaile bat dago, eta haren gaitasunek bi atazak betetzea ahalbidetzen dute. MxAutomation liburutegiak erabiliz, sei ardatzeko robot seriea PLCan zuzenean programatu ahal izango da, kontroladore propioa programatu beharrik gabe. Halaber, kontroladore horren PC funtzionaltasuna erabiliz, robotaren datuak jaso eta hodeiko datu-base batera igo daitezke.

ÍNDICE

LISTADO DE ILUSTRACIONES Y TABLAS

1. MEMORIA

1.1.Introducción

En este documento se va a trabajar sobre una estación robotizada consistente en un robot serie de seis ejes. La función de esta estación es realizar un montaje, cuyas características variarán en base a las solicitudes que va realizando un agente. Los montajes constan de cuatro piezas que se deben colocar en secuencia sobre una base, aunque no siempre es necesario colocarlas todas, ya que la base puede venir con piezas ya colocadas, o puede no ser necesario completar todo el montaje. También se debe tener en cuenta la ubicación de las bases en su pallet, que dispone de espacio para seis de ellas.

En base a estos requisitos, se han identificado diez servicios posibles que se deben programar, que además se deben adaptar para cada una de las seis posiciones posibles en las que se puede encontrar la base.

Para realizar esta programación, se han dividido varias de las tareas en distintas operaciones, que se combinarán en base al servicio solicitado. La programación se ha realizado en TIA Portal, empleando las librerías mxAutomation, que permiten programar el robot directamente en el PLC, sin tener que acceder a su propia controladora.

Al terminar cada servicio, el robot envía una serie de datos, indicando que servicio ha realizado, a cuantas bases, y sus timestamps de inicio y fin de operación. Estos datos se envían a una base de datos en la nube mediante el protocolo MQTT, con el objetivo de, posteriormente, analizarlos y realizar alguna aplicación con ellos.

Este trabajo se divide en cuatro grandes apartados.

El primero se denomina Memoria. En él se analizan el contexto y los antecedentes que permitirán entender mejor los desarrollos posteriores, se definen los objetivos y el alcance del proyecto, se describen los beneficios que aporta, en los ámbitos tanto económicos y técnicos como sociales. Se examinará el estado del arte, para aportar un mayor contexto al trabajo y conocer mejor la tecnología empleada. Además, se llevará a cabo un análisis de riesgos para comprobar la seguridad de la solución aportada.

El segundo gran apartado es el de Metodología. Aquí se van a describir detalladamente todas las tareas realizadas en el trabajo, y todas las fases por las que ha pasado. Se explicarán los algoritmos y se analizarán los resultados. También se expondrá la planificación seguida en el trabajo.

En el tercer apartado, de aspectos económicos, se analizan tanto el presupuesto como la rentabilidad del proyecto.

Finalmente se añade el apartado de Conclusiones, donde se identifica si se han cumplido los objetivos propuestos en base al análisis de todos los apartados anteriores. También se pueden identificar posibles líneas futuras de desarrollo a implementar en base a este proyecto.

Además de todos estos apartados, se incluirán una bibliografía para identificar las fuentes citadas a lo largo del trabajo, y unos anexos para incluir código y documentos generados en el trabajo.

1.2.CONTEXTO

Este trabajo se encuadra en el nuevo paradigma industrial, que recibe el nombre de Industria 4.0, o industria conectada. En el nuevo marco que se va imponiendo, se exige cada vez mayor flexibilidad a los procesos de producción, menores tiempos de parada, y fallos mínimos y predecibles. En este contexto, los datos están adquiriendo una importancia capital para poder adelantarse a los problemas, predecir como pueden reaccionar las máquinas ante los cambios, y tratar de optimizar los procesos.

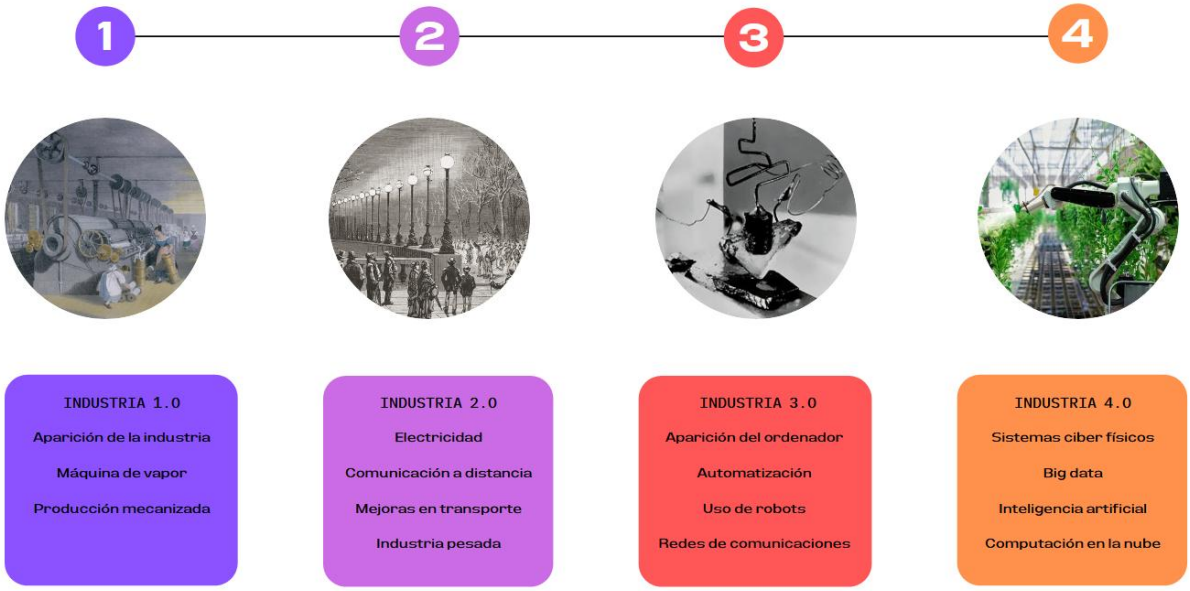


Ilustración 1. Línea de tiempo de la industria

El nombre de revolución industrial se da a una serie de cambios en la tecnología o en la cultura laboral, que permiten dar un gran salto adelante en la industria, respecto a lo que se conocía anteriormente. La primera revolución industrial llegó a finales del siglo XVIII, con una serie de cambios en la forma de trabajar, basándose cada vez mas en máquinas y en formas de producción mecanizadas en lugar de en trabajos manuales. El uso de la fuerza del agua, y mas tarde de la máquina de vapor, fueron esenciales para el desarrollo de esta revolución. Los cambios en la economía fueron notables, con la aparición de empleos industriales, y un sustancial enriquecimiento de la población general.

A mediados del siglo XIX surge la llamada segunda revolución industrial. Esta es probablemente la mas conocida, basada en el desarrollo del ferrocarril, la producción en masa, el desarrollo de la electricidad y las comunicaciones, o la industria pesada. En este punto, las empresas empiezan a profesionalizarse, la formación es cada vez mas

especializada, y se produce un importante desarrollo económico. La sociedad experimentó enormes cambios, con el desarrollo de sistemas de alcantarillado en las ciudades, la posibilidad de obtener empleos estables, de comunicarse a largas distancias gracias al telégrafo, o de esquivar las hambrunas causadas por malas cosechas debido a las mejoras en el transporte.

El paradigma en el que se operaba hasta ahora era el de la tercera revolución industrial. Esta fue un gran salto adelante, debido a la introducción de la automatización en la industria, posibilitada por la introducción del transistor y el circuito integrado. Los autómatas programables empezaron a sustituir a los circuitos eléctricos, y se empiezan a emplear robots. Con la generalización de los ordenadores surgieron las tecnologías de la información, que aportaron una mayor agilidad y control a todas las etapas y niveles de la producción.

Desde el surgimiento de estas tecnologías, se había entrado en una etapa de evolución, en la que la miniaturización de los transistores permitió dotar de mayor potencia a los autómatas y ordenadores, y los robots y máquinas autónomas tomaban cada vez mas protagonismo. A esto hay que añadir la creación de internet tal y como lo conocemos, a principios de los años 90.

Con el tiempo, la reducción de tamaño y precio en los microchips y los sistemas de información empiezan a permitir su uso en lugares en los que anteriormente no se habría pensado, como sensores o elementos básicos de las máquinas. El aumento en la potencia de procesamiento empieza a permitir realizar algoritmos con los datos que antes pertenecían solo al plano teórico. Y en una infraestructura de internet cada vez mas rápida y potente empiezan a surgir conceptos como el del almacenamiento y acceso remoto a servicios.

Llegado un punto, empieza a surgir un nuevo tipo de industria, en la que los límites entre lo digital y lo real se vuelven cada vez mas borrosos. Debido a la proliferación de elementos inteligentes, en la llamada Internet de las Cosas (IoT), hay acceso a cada vez mayores cantidades de datos, por lo que cada vez se pueden realizar modelos que simulen mejor la realidad, con los que, por ejemplo, se pueda predecir el fallo de una máquina, o encontrar la causa de problemas que ya estén ocurriendo.

Debido también al aumento en la capacidad de procesamiento, y a la gran cantidad de datos disponibles, los sistemas de inteligencia artificial están cada vez mas presentes. Ya no solo en el ámbito de la industria, sino en otros como la medicina, la justicia, o el deporte. La posibilidad de analizar todos esos datos y obtener respuestas fiables permite hallar nuevas necesidades o patrones que habían permanecido ocultos, con lo que se puede aumentar la eficiencia y mejorar en aspectos en los que previamente no se

pensaba. Las limitaciones en la capacidad de procesamiento ya no son un problema, debido a que mediante servicios de *cloud computing*, se pueden ejecutar los algoritmos necesarios empleando hardware y software remoto vía internet, sin necesidad de disponer de ello en local, y pagando tan solo por la capacidad que se necesita.

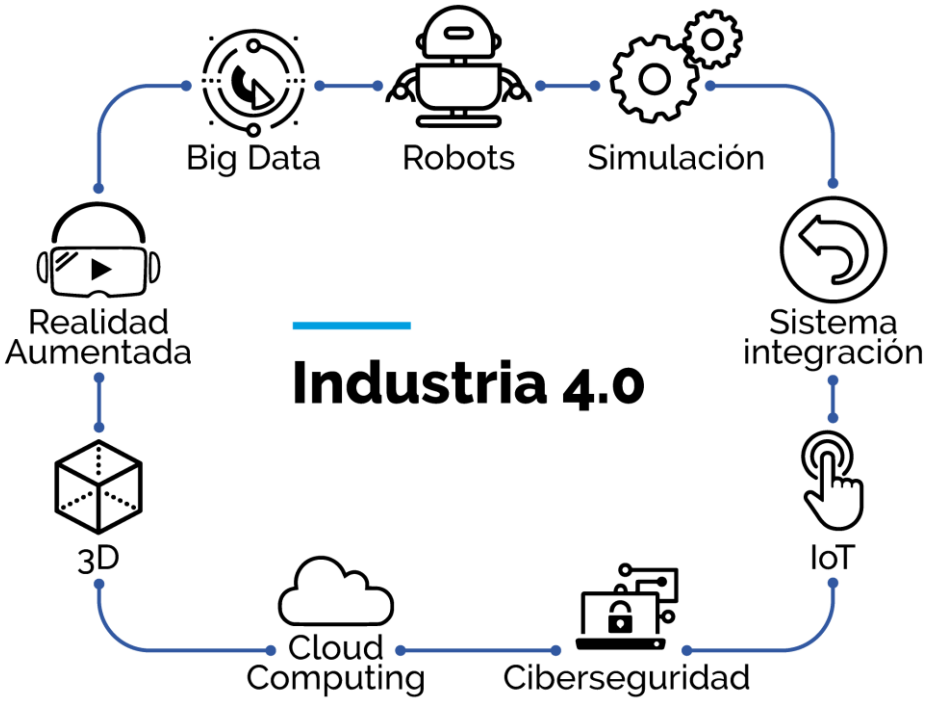


Ilustración 2. Los pilares de la industria 4.0

En el ámbito cultural, se observa una mayor integración, verticalmente entre todos los elementos de una organización, y horizontalmente con proveedores y contratas. Se desarrollan ecosistemas integrados donde se puede conocer el estado del producto en tiempo real, y hacer un seguimiento [1]. El desarrollo de sistemas ciber-físicos permite conocer todos los detalles del producto o del proceso, sin tener que recurrir al elemento real. Esto permite acceder a datos difíciles de conseguir por medios físicos, realizar cambios rápidos en un diseño, o presentar el producto a un cliente sin necesidad de desplazarse.

Se puede observar que los ladrillos que construyen la cuarta revolución industrial son, sin duda, los datos. Asegurar el acceso a datos en buenas cantidades y calidades es necesario para poder llegar a implementar cualquiera de las tecnologías antes mencionadas. Por ello, se han desarrollado protocolos específicos para la transmisión de datos en el ámbito del IoT, como puede ser el MQTT, diseñado para ser ligero, con lo que se puede usar en pequeños sensores y dispositivos sin mucha capacidad de procesamiento [2].

1.3.Objetivos y alcance

En este proyecto se va a trabajar sobre dos grandes objetivos.

El primero consiste en programar una célula robotizada para que realice uno o varios montajes, total o parcialmente. Esta célula se va a programar desde un PLC, en lugar de desde su propio controlador, empleando librerías ofrecidas por el fabricante.

El segundo objetivo consiste en recoger datos del proceso, y subirlos a una base de datos en la nube empleando el protocolo MQTT, desde la que cualquiera con acceso podrá observarlos, descargarlos, y analizarlos o emplearnos en diversos algoritmos

Este proyecto forma parte de un demostrador mas amplio, que incluye varios agentes externos, un gemelo digital de la célula, y la recogida y procesamiento de datos en varias bases de datos. Se comentarán algunos de estos proyectos a lo largo del documento.

1.3.1. Programación de la célula robotizada

La célula robotizada se encarga de realizar un montaje que consta como máximo de cinco piezas: base, rodamiento, bulón, tapa interior y tapa exterior. Las piezas se colocan secuencialmente en ese orden sobre la base. Se puede requerir colocar solo algunas de las piezas, si el montaje llega ya parcialmente montado, o si se planea terminarlo en una estación posterior.

Este objetivo se ha dividido a su vez en los siguientes:

- Identificar todos los servicios posibles examinando las diferentes combinaciones de piezas que se pueden montar.
- Diseñar una estructura de programa de PLC que permita ejecutar todos los servicios en todas las posiciones, siendo lo mas compacto posible.
- Realizar el programa empleando TIA Portal, y las librerías KUKA.PLC mxAutomation.
- Realizar pruebas para asegurar el buen funcionamiento del programa en todas las situaciones.

1.3.2. Registro de datos en la nube

A lo largo de su funcionamiento, el robot envía al PLC varios datos relativos a las operaciones y servicios. En concreto, se envían los números de lote, máquina, servicio, ítem, subproducto y orden. También se transmiten los timestamps de inicio y final del servicio.

Esta información se debe guardar si se quiere hacer un tratamiento posterior, o en tiempo real. Para ello, se ha optado por una solución en la nube, lo que permite gran flexibilidad, y la posibilidad de acceder a los datos desde cualquier lugar con conexión a internet.

Los objetivos identificados en este punto son:

- Emplear el ODK de Siemens para transmitir los datos desde la CPU de PLC al PC del SIMATIC ET 200SP Open Controller.
- Programar un cliente MQTT en el ODK, para publicar los datos en un bróker.
- Poner en marcha un bróker que se ajuste a las capacidades de los sistemas involucrados.
- Establecer un cliente que se suscriba al tema adecuado y reciba los datos del robot, para subirlos a una base de datos en la nube.
- Realizar pruebas para asegurar que los datos requeridos llegan a la base de datos.
- Mostrar los datos en un panel.

1.4.Beneficios que aporta el trabajo

1.4.1. Beneficios económicos

Mediante el uso de células robotizadas se puede asegurar una calidad uniforme del producto, con lo que son necesarias menos inspecciones de calidad, y se reducen las devoluciones por fallos en el producto.

También permiten aumentar la complejidad y la velocidad de las operaciones en la cadena de montaje, lo que aumenta la calidad y la cantidad de productos, repercutiendo ambas cosas muy positivamente en las cuentas de una empresa.

Por ello, pese al elevado coste inicial, invertir en la robotización de una fábrica sale rentable debido a los grandes beneficios que conlleva.

El uso de servicios en la nube hace innecesaria la instalación de servidores o bases de datos en la propia empresa. Estos equipos pueden tener un coste muy alto, y no todas las empresas se lo pueden permitir, o no desean invertir en ello antes de comprobar si la aplicación que se desea implementar cumple su función.

En la nube, además, se puede escalar la aplicación muy fácilmente en el caso de que se desee expandir su alcance. En el caso de disponer de hardware propio, podría ser necesario sustituirlo, con su coste asociado. Por ello, el uso de la nube es beneficioso cuando se está implementando una nueva aplicación, al menos hasta que se haya delimitado su escala y se pueda asegurar qué hardware sería necesario adquirir para llevar a cabo las operaciones en la propia empresa.

1.4.2. Beneficios técnicos

Para una empresa, el empleo de robots puede suponer un importante salto en el ámbito técnico. Para realizar multitud de tareas manuales y repetitivas, los robots son mas fiables, precisos y rápidos que los humanos, lo que sin duda repercute positivamente en la calidad del producto.

Por otra parte, la liberación de personal humano de tareas de bajo valor añadido permite contar con mas personal en otras tareas que puedan aportar mayor valor, como la planificación de mejoras en el producto o en el proceso, de forma que la

empresa estará mas preparada para afrontar la llegada de tecnologías aún mas novedosas.

Para lograr estos objetivos, la aparición de librerías que permiten programar robots desde un PLC simplifica notablemente el proceso. Al homogeneizar tecnologías, ya no es necesario disponer de un especialista en programación de PLCs, y otro en programación de robots, ya que ambas cosas las puede realizar una misma persona.

Por ello, empresas de mediano o pequeño tamaño pueden abordar mas fácilmente la instalación y mantenimiento de una pequeña cantidad de robots empleando los recursos de los que ya disponen. Aun así es necesario comentar que estas librerías permiten controlar un número limitado de robots, por lo que para instalaciones de mayor tamaño será necesario contar con especialistas.

La captación de datos del proceso también puede suponer un importante salto en el ámbito técnico. El análisis de datos permite detectar problemas leves antes de que se conviertan en algo grave, reduciendo tiempos de paro y aumentando la calidad del producto al reducir el tiempo que una máquina puede estar funcionando en mal estado.

También se pueden detectar patrones no deseados o tendencias perjudiciales en el proceso, que una vez eliminados permitirían aumentar la calidad, la eficiencia o la seguridad.

1.4.3. Beneficios sociales

El uso de células robotizadas limita el riesgo de las personas, evitando que formen parte de la cadena de producción, o asistiéndolas en tareas pesadas y repetitivas. También pueden realizar operaciones en áreas de peligro, o manipular sustancias tóxicas o corrosivas, evitando riesgos a los trabajadores humanos.

El uso de servicios en la nube puede tener beneficios para las personas mas allá del ámbito laboral. La mayoría de estos servicios tienen planes de uso gratuitos o a muy bajo coste para usuarios individuales, que pueden acceder a herramientas avanzadas antes limitadas al mundo empresarial.

De esta forma, un usuario puede diseñar y programar pequeños servicios empleando estas herramientas, desde el riego automático de plantas hasta la monitorización y el análisis de actividad deportiva. Con el uso de estas herramientas incluso se pueden

generar ideas que den lugar a nuevas empresas, sin la necesidad de una gran inversión inicial.

1.5.Análisis de riesgos

En este apartado se van a describir varios de los riesgos que se pueden encontrar relacionados con este trabajo; tanto en la parte dedicada a la programación del robot, como a la dedicada a la captación y subida de datos a la nube.

Para clasificar los riesgos posibles, se va a emplear la matriz de riesgos, también conocida como matriz de probabilidad-impacto. En esta matriz, se clasifican los riesgos según dos variables, que son la probabilidad de que suceda, y el impacto que tendría. A continuación, se explican los niveles de estas dos variables.

Clasificación	Probabilidad
A	Extremadamente improbable. Muy escasos antecedentes
B	Remota. Pocos antecedentes
C	Ocasional
D	Común. Fallos conocidos en la mayoría de aplicaciones similares
E	Frecuente. Fallos casi inevitables

Tabla 1. Tabla de probabilidades

Clasificación	Impacto
1	Sin efectos relevantes
2	Efectos menores, sin daños materiales ni personales
3	Leve, con daños escasos
4	Crítico. Pérdida de al menos una función principal. Puede resultar mortal
5	Catastrófico. Destrucción del sistema. Puede resultar mortal

Tabla 2. Tabla de incidencia

1.5.1. Riesgo 1: Fallo en la programación del robot

Debido al gran número de pasos necesarios, y a que desde la programación no es posible saber si los movimientos que el robot va a realizar son correctos, es posible que en las primeras pruebas se comporte de forma extraña, o que haga movimientos no previstos, que puedan llegar a amenazar la integridad de la estación, o del propio robot.

Probabilidad: C-Ocasional. Un desarrollador de software con experiencia tiene cuidado a la hora de realizar programas tan críticos. Sin embargo, debido a la

complejidad del proyecto, es posible que en alguna ocasión no se detecten algunos problemas hasta la fase de pruebas.

Impacto: 4-Crítico. Es necesario gestionar este problema con cautela, ya que un robot industrial puede causar graves daños a material o personas si se mueve de forma descontrolada.

Acciones propuestas: Es necesario llevar a cabo una exhaustiva fase de pruebas antes de la puesta en marcha, con velocidad reducida y extremo cuidado. Si es posible, sería conveniente utilizar un gemelo digital para reducir los riesgos al mínimo.

1.5.2. Riesgo 2: Fallo de comunicación

Ya que, en este caso, la programación del robot se encuentra en un PLC comunicado por Profinet con el controlador del robot, es posible que un fallo en la red de comunicaciones deje al robot sin control.

Probabilidad: B-Remota. En las plantas, se suele contar con redundancias en las comunicaciones, precisamente para evitar este tipo de problemas, que pueden resultar críticos.

Impacto: 3-Leve. Al perder la comunicación, el robot se queda en la posición en la que está. Sin embargo, hay que tener cuidado a la hora de restablecer la comunicación, porque el programa puede estar en un punto diferente al que el robot espera, lo que puede causar movimientos extraños.

Acciones propuestas: Las fibras de comunicaciones en planta deben tener una disposición de anillo, de forma que, si una sección se corta físicamente, se pueda acceder por el otro lado. Es posible que la caída de un servidor central interrumpa las comunicaciones, por lo que deben estar protegidos por un sistema de alimentación ininterrumpida, y si es posible, contar con un servidor de respaldo. Se debe tener cuidado a la hora de restablecer las comunicaciones, para asegurar que el robot se mueva a una posición segura.

1.5.3. Riesgo 3: Ataques informáticos

Uno de los problemas de la nube es que es vulnerable a ataques. Al enviar datos a un servidor externo de la empresa, se pierde el control sobre su ubicación, o su nivel de protección. Por ello, es posible que un agente externo realice un ataque al servidor en el que se alojan, con lo que se puede perder la funcionalidad por un tiempo, se pueden perder los datos allí almacenados, o es posible que los datos sean robados.

Probabilidad: C-Ocasional. Las bases de datos que dan servicio a empresas suelen tener sistemas de protección robustos, ya que su negocio depende de ello. Aun así, existen antecedentes de ataques exitosos, y el robo de datos industriales puede ser una actividad en la que se vean involucrados actores con gran potencial.

Impacto: 3-Leve. Pese a que la pérdida de datos pueda ser un fastidio, y dificultar planificaciones estratégicas, mantenimientos preventivos u otras operaciones; difícilmente supondrá un golpe crítico para el funcionamiento de la planta.

Acciones propuestas: Para datos sensibles o de alta importancia estratégica, es mejor emplear un Fog privado, en lugar de recurrir a la nube. Por ello, una vez que se hayan hecho pruebas en la nube y la aplicación que se desee esté establecida, es conveniente invertir en equipos propios que puedan tener mayor seguridad sin conexión a internet, o con una conexión muy protegida.

1.5.4. Riesgo 4: Pérdida de comunicación con la base de datos

Es posible que se pierda la conexión a internet en algún momento, o que alguna de las herramientas empleadas en el proceso se desconfigure. En este caso, se perderían datos del proceso, hasta que se ponga remedio al problema.

Probabilidad: B-Remota. Si el sistema está configurado correctamente la conexión debería mantenerse. Sin embargo, es posible que ocurra un fallo humano.

Impacto: 2-Efectos menores. El hecho de que no se recojan datos un tiempo no debería ocasionar nada mas que alguna pequeña inconveniencia en su tratamiento. Esto es, suponiendo que se solucione en un espacio de tiempo razonable.

Acciones propuestas: Asegurarse de que la conexión está correctamente configurada, y revisar periódicamente que se reciben los datos correctamente.

1.5.5. Matriz de riesgos

Empleando la matriz de riesgos, se puede evaluar rápidamente la gravedad de los problemas que se pueden producir, lo que puede servir para priorizar acciones y recursos. A continuación, se presentan los cuatro riesgos anteriores en su posición en la matriz.

\Impacto Probabilidad\	1	2	3	4	5
A	Bajo	Bajo	Bajo	Moderado	Alto
B	Bajo	Bajo Riesgo 4	Moderado Riesgo 2	Alto	Inaceptable
C	Bajo	Bajo	Moderado Riesgo 3	Alto Riesgo 1	Inaceptable
D	Bajo	Moderado	Alto	Inaceptable	Inaceptable
E	Moderado	Alto	Inaceptable	Inaceptable	Inaceptable

Tabla 3. Matriz de riesgos

2. METODOLOGÍA

2.1. Descripción de fases, tareas, equipos o procedimientos

2.1.1. Hardware

A continuación, se va a realizar una explicación de los distintos elementos físicos que componen el demostrador.

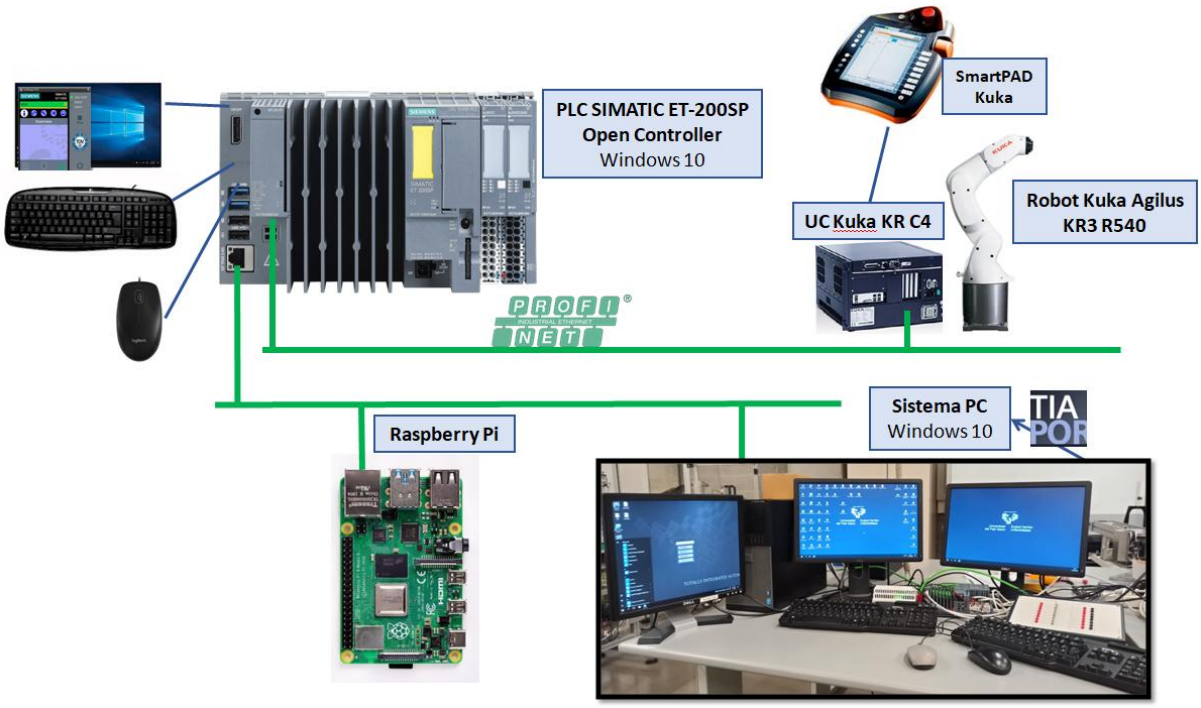


Ilustración 3. Hardware empleado en el proyecto

a) Robot KUKA KR3 R540

El robot empleado en la célula robotizada es el modelo KR3 R540 de la gama AGILUS de KUKA. Se trata de un robot serie de seis ejes. Debido a la ligereza de las piezas empleadas, realizadas mediante impresión 3D, prácticamente cualquier robot serie del mercado podría moverlas, por lo que el factor limitante en este caso es el alcance de la garra, para poder acceder a todos los elementos necesarios situados en la mesa de trabajo.

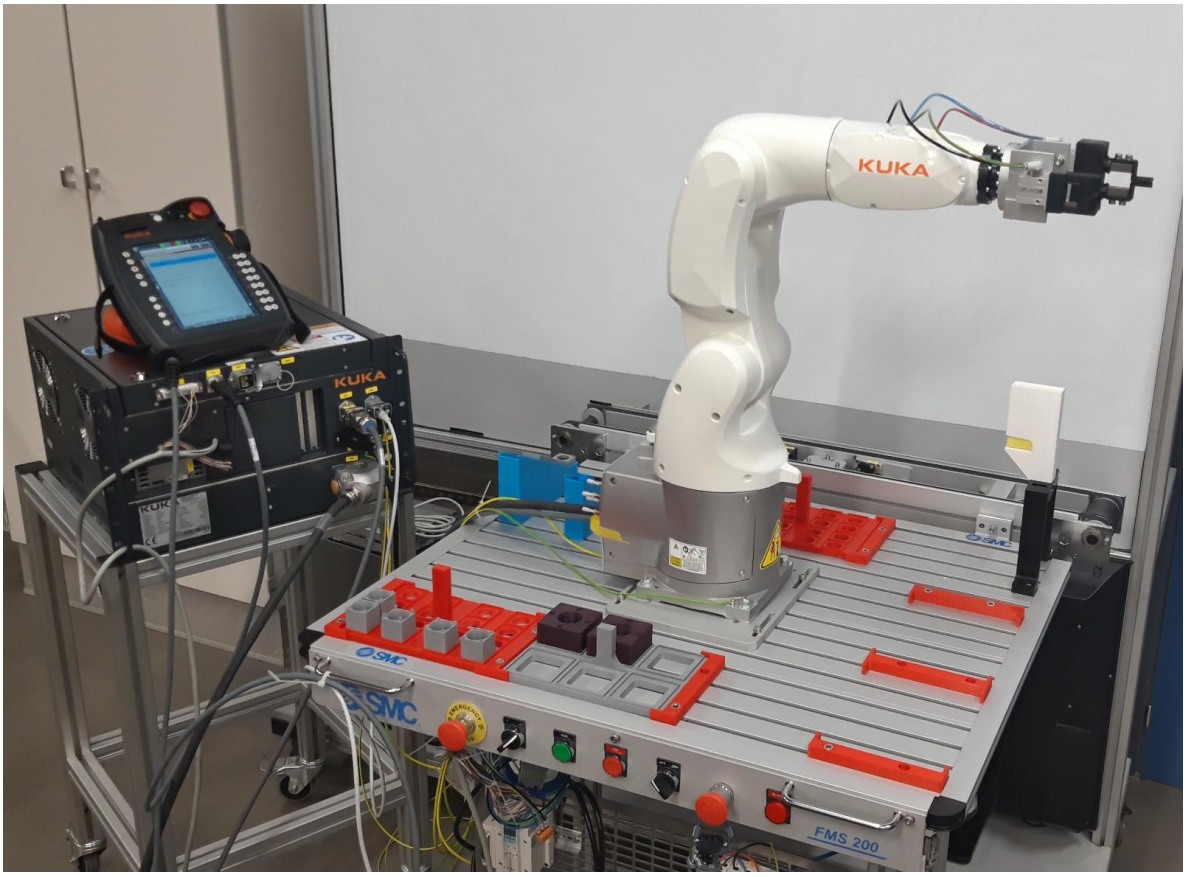


Ilustración 4. Célula robotizada, con su controladora y SmartPad

La mesa empleada es un modelo de SMC, igual que las presentes en la célula de fabricación FMS200 también presente en el laboratorio, lo que permitiría integrar esta estación en la célula en un futuro. Cuenta con varias ubicaciones para colocar pallets. Dado que el robot tiene un alcance máximo de 541 mm (sin contar la garra) [3], se comprueba que alcanza sin problemas cualquier lugar de la mesa si se sitúa en su centro.

Datos técnicos

Alcance máximo	541 mm
Carga máxima	3 kg
Repetibilidad de posición (ISO 9283)	± 0,02 mm
Número ejes	6
Posición de montaje	Suelo; Techo; Pared
Superficie de colocación	179 mm x 179 mm
Peso	aprox. 26,5 kg

Ilustración 5. Datos técnicos del robot KUKA KR3 R540

La garra empleada consta de dos pinzas neumáticas situadas una a cada lado de una estructura metálica, cuyo centro está acoplado a la brida del robot.

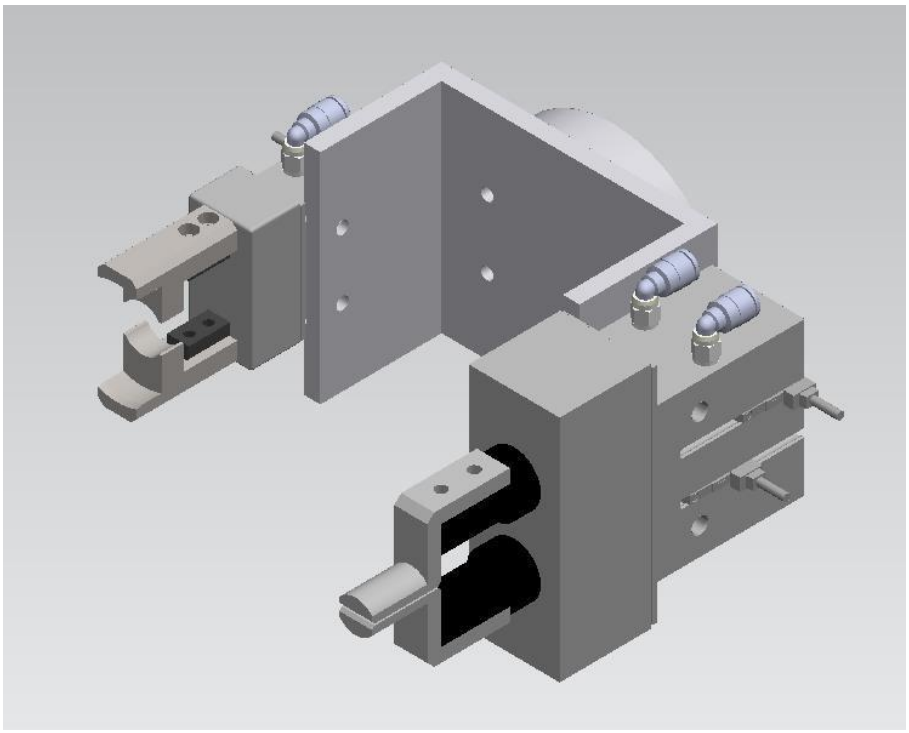


Ilustración 6. Modelo de la garra del robot

Es necesario emplear dos pinzas debido a las diferentes características de los elementos que se van a agarrar.

Una de las pinzas es normalmente cerrada (NC), y se emplea para sujetar las piezas que disponen de un orificio central (rodamiento, tapa interior y tapa exterior).

La otra pinza es normalmente abierta (NA), y es utilizada para sujetar el bulón y los pallets, que cuentan con un agarre cilíndrico.

b) Controladora KR C4 Compact

La controladora escogida para este robot es la KUKA KR C4 Compact. Se trata de la controladora mas pequeña disponible, y también la mas económica. Permite el control de seis ejes, expandibles a otros dos, lo que resulta suficiente para este proyecto.

Características [4]	**KR C4 compact**
Dimensiones (HxLxA)	271 x 483 x 460 mm
Procesador	Tecnología MultiCore
Disco duro	SSD
Punto de conexión	USB3.0, GbE, DVI-I, DisplayPort
Número de ejes (máx.)	6+2 (con caja de ejes adicional)
Frecuencia de red	50/60 Hz ± 1 Hz
Tensión de conexión nominal	CA 200 V a 230 V
Tipo de protección	IP20
Temperatura ambiente	de +5 °C a +45 °C
Peso	33kg
Buses de campo soportados	PROFINET, PROFIBUS, INTERBUS, EtherCAT, Ethernet/IP, DeviceNet, VARANBUS

Tabla 4. Características KUKA KR C4 Compact

c) SmartPad

El SmartPad se incluye junto al robot y la controladora, y se emplea para el diagnóstico, la programación y el mantenimiento del equipo. Desde aquí se comprueban, ajustan y cargan los diferentes programas que puede ejecutar la controladora. También se puede mover el robot directamente de forma manual, y ejecutar una parada de emergencia.

Ilustración 7. SmartPad

d) Controlador PLC SIMATIC ET200SP Open Controller

Este controlador, con CPU 1515SP PC, emplea un concepto diferente al de los autómatas S7 1500 estándar, ya que es un PLC software basado en una plataforma PC que funciona sobre Windows 10. Esto implica que puede operar como PC y como PLC al mismo tiempo, lo que le otorga gran flexibilidad.

Empleando el SIMATIC ODK (open development kit), se pueden implementar funciones especiales que hagan uso de ambas partes del controlador empleando los lenguajes C/C++. En este caso, el ODK se emplea para establecer la conexión MQTT y subir a la nube los datos recibidos por el PLC.

Además, permite comunicaciones empleando Profinet IO, y puede acoplarse a tarjetas de E/S de la gama ET200SP. Operando como PC, dispone de tres puertos USB que se pueden emplear para conectar teclado y ratón, salida de vídeo para conectar una pantalla, y un puerto Ethernet que permite acceso a internet o comunicación con otros dispositivos.

Ilustración 8. Controlador S7 1515SP PC

e) Raspberry Pi

Un ordenador compacto y barato, muy empleado en aplicaciones IoT. Dispone de una CPU de 4 núcleos a 1,5 GHz, y 4 Gb de memoria, y utiliza un sistema operativo basado en Linux. Dispone de 4 puertos USB, una conexión Ethernet, y acceso a redes WiFi. Se utiliza para alojar el bróker Mosquitto, y para comunicarse con internet, de forma que pueda enviar los datos al cliente situado en la nube.

Ilustración 9. Raspberry Pi empleada en el proyecto

2.1.2. Software

En este apartado se presentan los diferentes programas y piezas de software que se han empleado en este proyecto, ya sea como parte integral del demostrador, o como plataforma de desarrollo.

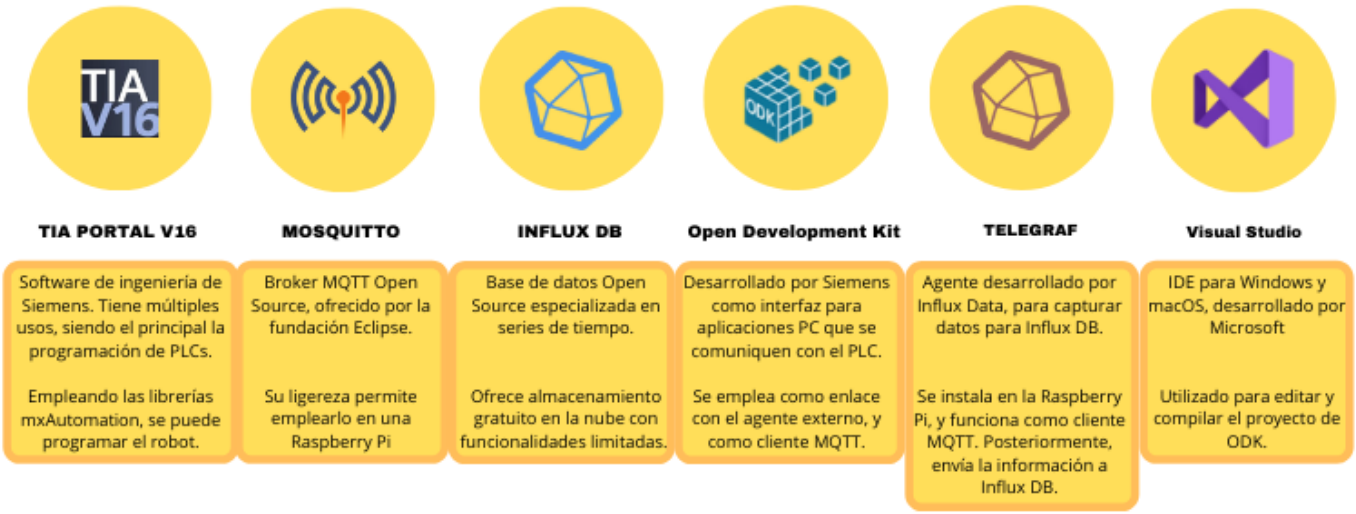


Ilustración 10. Software empleado

a) TIA Portal V16

Herramienta de ingeniería empleada en el desarrollo de la programación del robot. Mediante el uso de la librería KUKA.PLC mxAutomation, en esta herramienta se puede realizar la programación del robot, así como manejarlo mediante un HMI. En esta herramienta también se deben cargar algunos bloques creados en la compilación del programa ODK, para poder transferir los datos del PLC al PC.

Librería KUKA.PLC mxAutomation

Para poder manejar el robot utilizando un autómata SIMATIC S7-1500, es necesario utilizar la librería de bloques KUKA.PLC mxAutomation. Esta librería contiene varios FBs, que proporcionan acceso a las funciones mas importantes del robot. Mediante un fichero GSDML, se puede incluir el robot en la configuración hardware del

proyecto, como un dispositivo de E/S del PLC, comunicando la controladora C4 vía PROFINET. De esta manera, se pueden llegar a controlar hasta 5 robots por cada CPU.

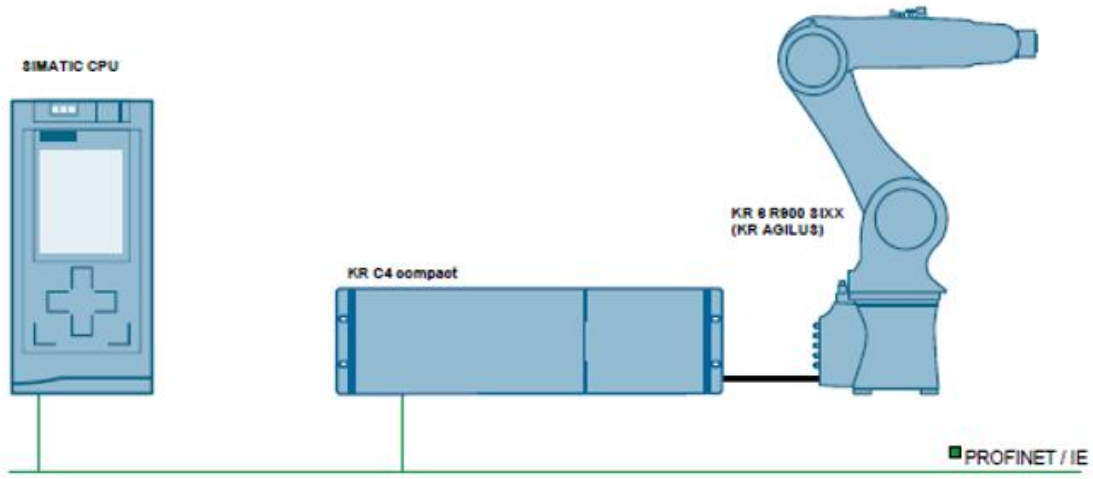


Ilustración 11. Conexión entre autómata y robot

La librería KUKA.PLC mxAutomation contiene dos tipos de bloques de movimiento [5]:

- Los bloques con el prefijo **"KRC_"** hacen referencia a bloques incluidos en las librerías KUKA.PLC mxAutomation (no funcionan de acuerdo con los estándares PLC Open). Se incluyen por razones de compatibilidad.

- Los bloques con el prefijo **"MC_"** se encuentran ubicados en su propio directorio y contienen funcionalidades de Motion Control. Las entradas y salidas de estos bloques se establecen de acuerdo a los estándares PLC Open. [6]

b) Influx DB

Base de datos de código abierto diseñada para gestionar series de tiempos. Es particularmente apropiada para recibir datos de sensores o de otros componentes de un sistema IoT, en los que se reciben datos de forma continua y en los que el tiempo es relevante.

Influx dispone de un agente que ejerce de servidor, llamado Telegraf, que se usa para recolectar datos de diversas fuentes y mandarlos a la base de datos. Mediante plugins, se puede configurar para recoger datos de una gran cantidad de fuentes. Este agente se ha instalado en la misma Raspberry que Mosquitto, y como cliente MQTT recibe los mensajes del tema al que está suscrito, para después enviarlos a la base de datos en la nube.

c) Eclipse Mosquitto

Mosquitto es un broker MQTT de código abierto, perteneciente a la fundación Eclipse. Debido a su ligereza, puede funcionar en dispositivos de baja potencia. Está disponible para su uso en varias plataformas. En este trabajo, se instalará en una Raspberry Pi, que recibirá los mensajes del cliente situado en el PLC.

Como bróker, Mosquitto tiene la tarea de recibir todos los mensajes, filtrarlos, determinar que cliente está suscrito a cada mensaje, y enviar los mensajes a los clientes correctos. Si la conexión está configurada con un cierto nivel de seguridad, el bróker también gestiona la autenticación y autorización de los clientes. En general, el bróker es el corazón del sistema, y por ello debe ser resistente a fallos.

d) ODK (Open Development Kit)

La interfaz ODK ha sido desarrollada por Siemens para su uso específico en los SIMATIC ET 200SP Open Controller, con CPU 1515SP PC. Estos dispositivos, como el usado en este proyecto, tienen funcionalidades de PC y de PLC integradas en la misma máquina. La integración de ambas funcionalidades permite desarrollar soluciones de automatización en un único dispositivo, y utilizar librerías y programas escritos en lenguajes de alto nivel.

En este proyecto, ODK se emplea para programar el cliente MQTT que envía al bróker Mosquitto los datos del robot requeridos. También se emplea para comunicar el PLC con el agente externo.

2.1.3. Comunicaciones

Tras exponer todos los componentes hardware y software del sistema, es necesario aclarar las comunicaciones entre ellos.

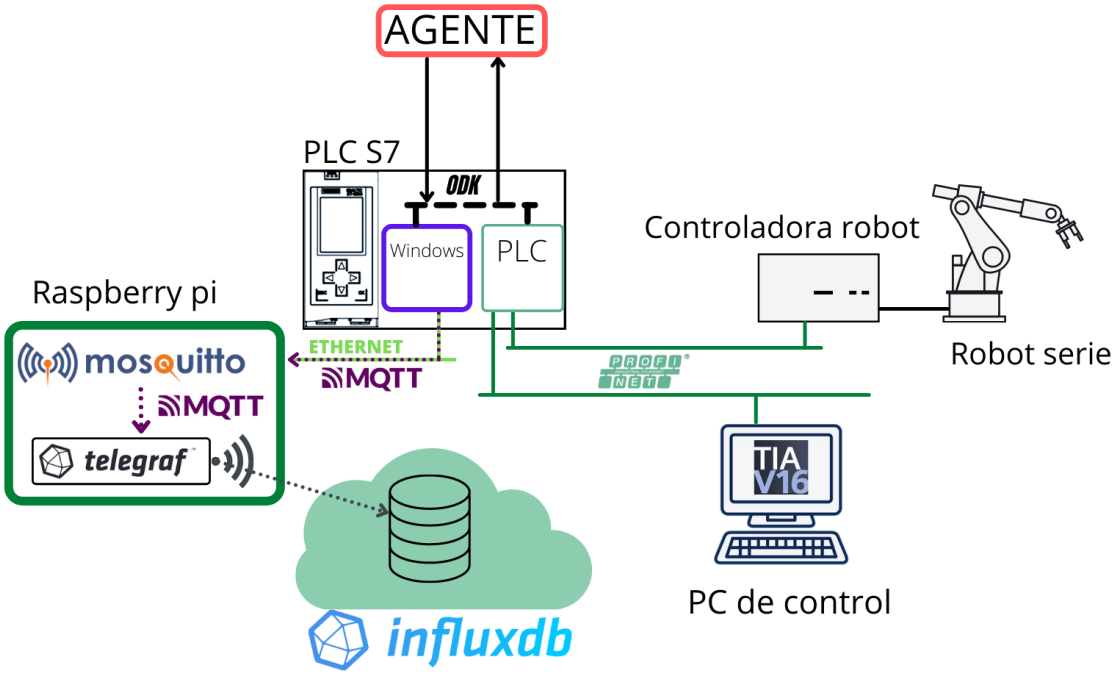


Ilustración 12. Comunicaciones

En primer lugar, la controladora del robot está conectada al PLC mediante una conexión Profinet, al igual que el ordenador con TIA Portal, empleado para programarlo. La Raspberry Pi se conecta al PLC por un cable Ethernet, por el que se envían los datos del cliente MQTT.

El ODK conecta el PLC software con el sistema operativo Windows, y establece la comunicación con el agente externo. También funciona como cliente MQTT, conectando con el bróker Mosquitto alojado en la Raspberry Pi.

Dentro de la Raspberry Pi se aloja también Telegraf, que como cliente MQTT se suscribe al tópico del robot en el bróker Mosquitto. Posteriormente, emplea la conexión WiFi de la Raspberry Pi para establecer una conexión por internet con la base de datos Influx DB en la nube, y enviar los datos del robot.

2.1.4. Montaje

Como se ha explicado previamente, la tarea de la célula robotizada es realizar un montaje consistente en una base, un rodamiento, un bulón, una tapa interna y una externa.

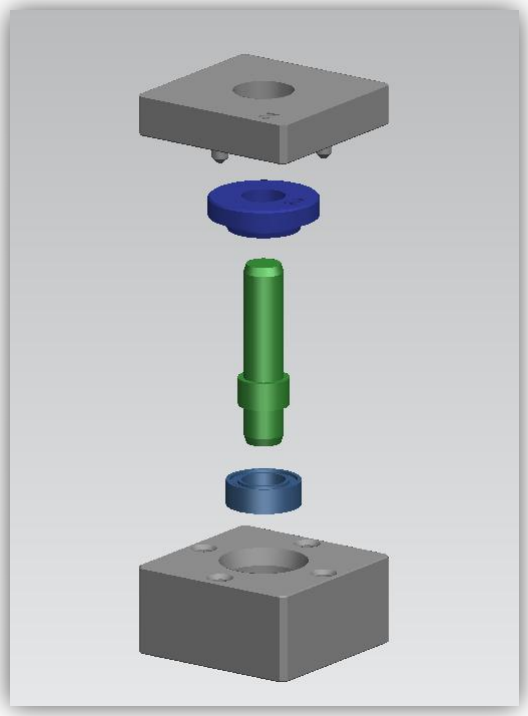


Ilustración 13. Explosión del montaje

Las piezas se deben montar en orden secuencial base – rodamiento – bulón – tapa interior – tapa exterior. Como entrada a la célula, se proporciona siempre una base, que puede incluir ya añadida alguna de las piezas, por lo que el robot deberá montar tan solo las restantes. Se pueden introducir hasta seis bases a la vez, pero todas ellas deberán contar con el mismo número de piezas de entrada. No es necesario que el robot realice el montaje completo, pudiendo colocar una única pieza antes de enviar los montajes a la salida, para que otra estación posterior los complete.

De esta manera, se contabilizan 10 servicios diferentes, dependiendo de la cantidad de piezas que el robot debe colocar y de las que vienen ya colocadas.

Servicio	Base	Rodamiento	Bulón	Tapa I.	Tapa E.
1	Incluida	Colocada			
2	Incluida	Colocada	Colocada		
3	Incluida	Colocada	Colocada	Colocada	
4	Incluida	Colocada	Colocada	Colocada	Colocada
5	Incluida	Incluida	Colocada		
6	Incluida	Incluida	Colocada	Colocada	
7	Incluida	Incluida	Colocada	Colocada	Colocada
8	Incluida	Incluida	Incluida	Colocada	
9	Incluida	Incluida	Incluida	Colocada	Colocada
10	Incluida	Incluida	Incluida	Incluida	Colocada

Tabla 5. Referencia de piezas incluidas de entrada y colocadas por el robot

También hay que tener en cuenta que las bases pueden venir en grupos de seis, por lo que se deben programar 60 combinaciones diferentes de movimientos.

2.2.Estructura del programa

A continuación se expone la estructura que sigue el programa desarrollado. Se puede dividir aproximadamente en tres capas, siendo la primera el Main; posteriormente la 'capa operativa', donde se encuentran las funciones que seleccionan qué se ejecuta y cómo lo hace; y finalmente la capa funcional, donde se encuentran los bloques de función que mueven el robot. En los próximos apartados se explicarán en detalle cada uno de estos bloques y funciones.

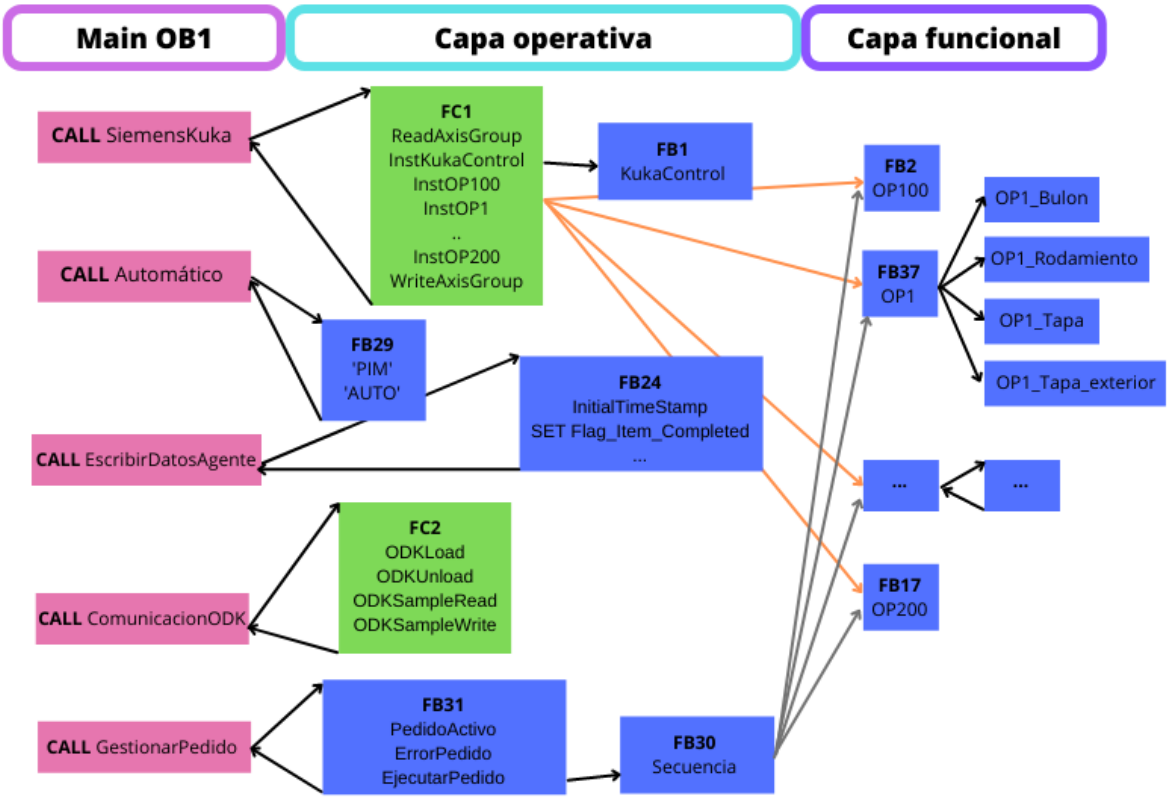


Ilustración 14. Estructura del programa

2.2.1. Configuración Hardware

Antes de empezar con el programa en sí, se va a explicar como se ha realizado la configuración de la controladora del robot, y su conexión.

En primer lugar, se debe obtener el fichero GSDML de la controladora KUKA KR C4 Compact. Este fichero lo proporciona el fabricante en la mayoría de elementos que se pueden comunicar con un PLC, y contiene información sobre el hardware y los

telegramas que se pueden emplear para las comunicaciones. Se pueden importar este tipo de ficheros desde la pestaña 'Opciones', en 'Administrar archivos de descripción de dispositivos'.

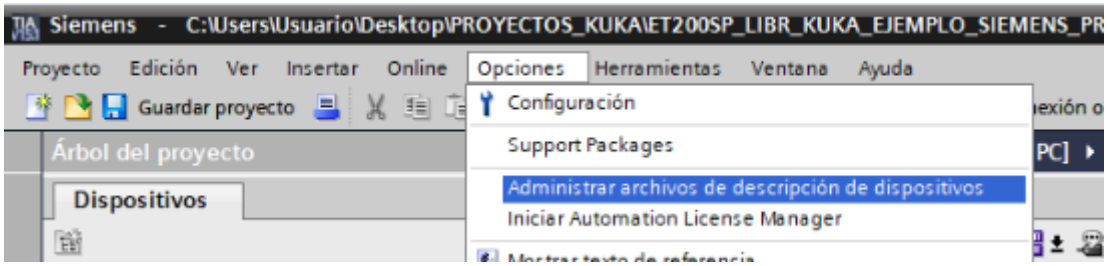


Ilustración 15. Importar GSDML

En la ventana emergente se pueden ver los archivos GSD instalados, entre los cuales se debe seleccionar el correspondiente a la versión de PROFINET instalada en nuestra controladora.

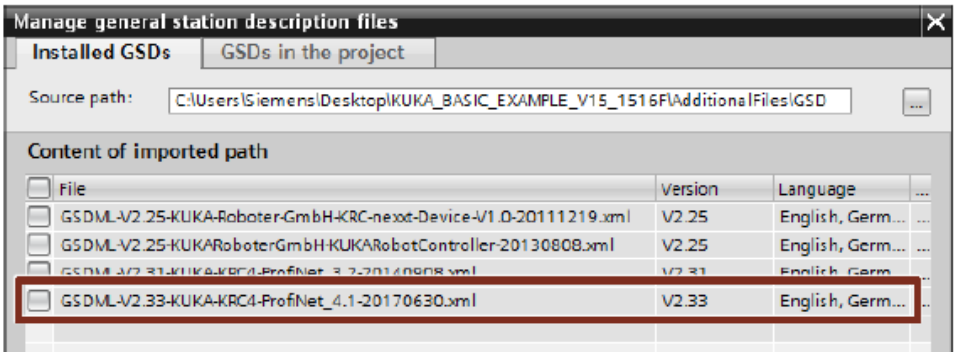


Ilustración 16. Archivos GSD instalados

Una vez se ha importado el archivo, la controladora estará disponible para insertar en el proyecto en la librería de hardware. En este caso, la controladora se llama KRC4-ProfiNet_4.1.

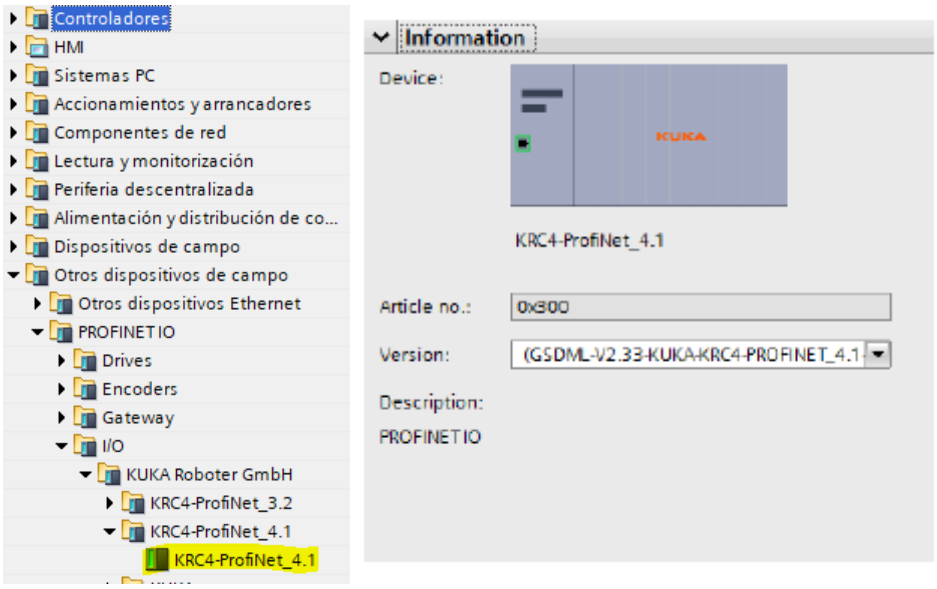


Ilustración 17. Controladora disponible en la librería de dispositivos

Una vez la controladora está integrada, se deben configurar los parámetros de comunicaciones. Para ello, es necesario integrar en la controladora del robot el telegrama de 2032 entradas y salidas digitales. Se debe anotar el valor que toma la dirección inicial de entradas, ya que hará falta posteriormente para programar los FBs de lectura y escritura del robot. En este caso, su valor es 2000.

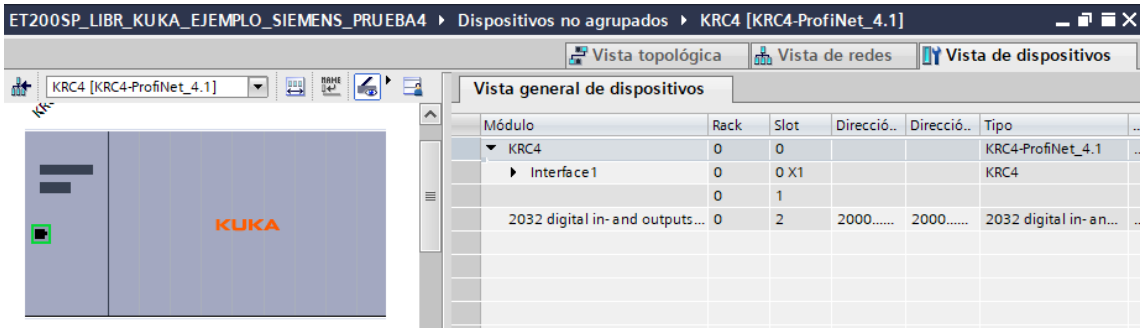


Ilustración 18. Telegrama insertado en la controladora

Finalmente, es necesario conectar la controladora del robot con el PLC S7 mediante una conexión PROFINET, configurando las direcciones IP de tal forma que ambas estén en la misma subred. En la siguiente imagen, se puede ver como se han realizado las conexiones a través de la subred PN/IE_1 siendo la dirección correspondiente al PLC la 192.168.1.1, y la de la controladora del robot la 192.168.1.10. Hay otro componente más en las comunicaciones, el HMI, que se explicará mas adelante.

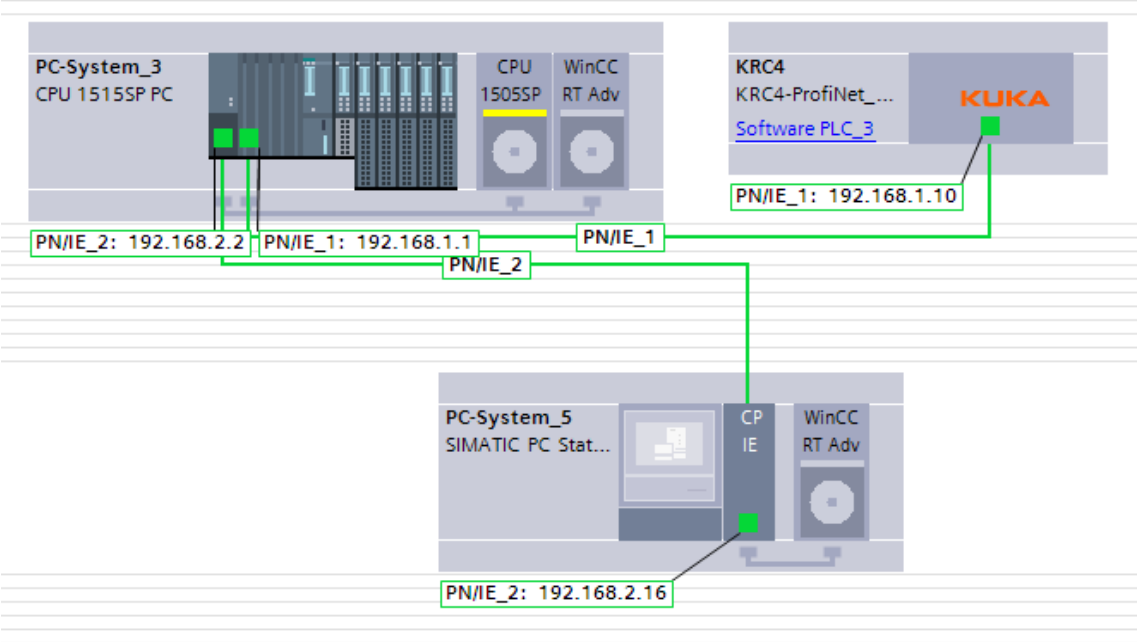


Ilustración 19. Conexiones entre PLC, controladora y HMI

2.2.2. Main[OB1]

En el bloque principal se realizan las llamadas a las siguientes funciones:

- SiemensKuka
- Automático
- EscribirDatosAgente
- ComunicacionODK
- GestionarPedido

2.2.3. Automático[FB29]

En esta función se establecen las dos condiciones mas importantes para permitir el funcionamiento del robot en modo automático.

- PIM: Esta señal indica que el robot se encuentra en la posición HOME. Consta de 6 comparadores de las posiciones, que se activan si el robot se encuentra dentro de un pequeño rango dentro del movimiento permitido de la articulación (en grados). También se incluye una condición mas, la variable *"HMIKuka".robot.*

powerRobot.status.autoExtern.RC_RDY1. Esta variable indica que el robot está preparado para recibir órdenes.

- AUTO: La señal AUTO tan solo se activa cuando se encuentran cerrados los pulsadores de marcha y de paro, lo que indica que se ha ordenado el funcionamiento automático. No se inicia la secuencia si esta señal no está activa.

2.2.4. SiemensKuka[FC1]

El FC SiemensKuka es la función que estructura el programa de control del robot. Consta de los siguientes bloques:

- KRC_ReadAxisGroup

  Mediante este bloque se pueden leer los datos del estado del robot de la controladora, y trasladarlos al almacenamiento interno de la libería mxAutomation. Esto permite que otros bloques de la librería accedan a estos datos para poder realizar sus operaciones.

  Los parámetros que se deben especificar son el grupo de ejes (el robot, en este caso el 1), y el byte inicial de entrada, al que se le ha asignado el valor 2000.

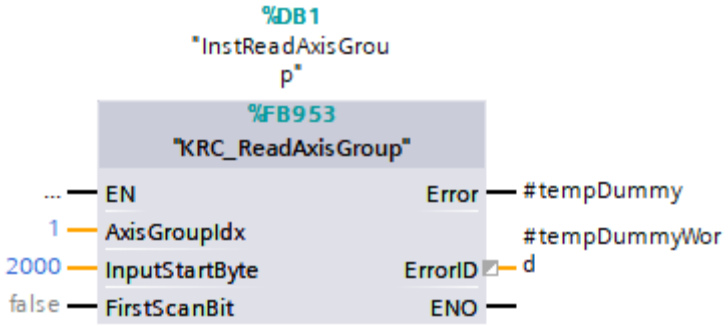


Ilustración 20. Bloque KRC_ReadAxisGroup

- KukaControl

  Aquí se instancia el FB KukaControl. Este FB contiene las funciones básicas del robot. Incluye los bloques que permiten manejar las siguientes funcionalidades:

  - Inicialización
  - Datos de diagnosis
  - Activación del modo Automatic External y encendido del robot
  - Control del override
  - Posición cartesiana actual y posición articular actual
  - Movimiento del robot en modo manual
  - Almacenar posición actual

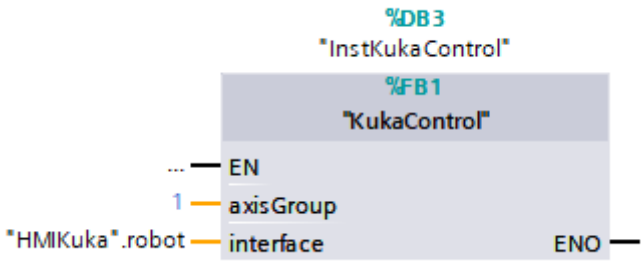


Ilustración 21. Bloque KukaControl

- Operaciones [OPs]

  El programa contiene varias operaciones, en las que se programan los movimientos que debe hacer el robot. Estas operaciones están instanciadas en el bloque SiemensKuka.

  Las operaciones de movimiento están gobernadas por la función 'Secuencia', y por las órdenes enviadas a traves del panel HMI (contenidas en el DB 'HMIKuka').

  También se incluyen las posiciones previamente calculadas a las que se desea enviar al robot, contenidas en el DB 'MxADBPosition_OP1'.

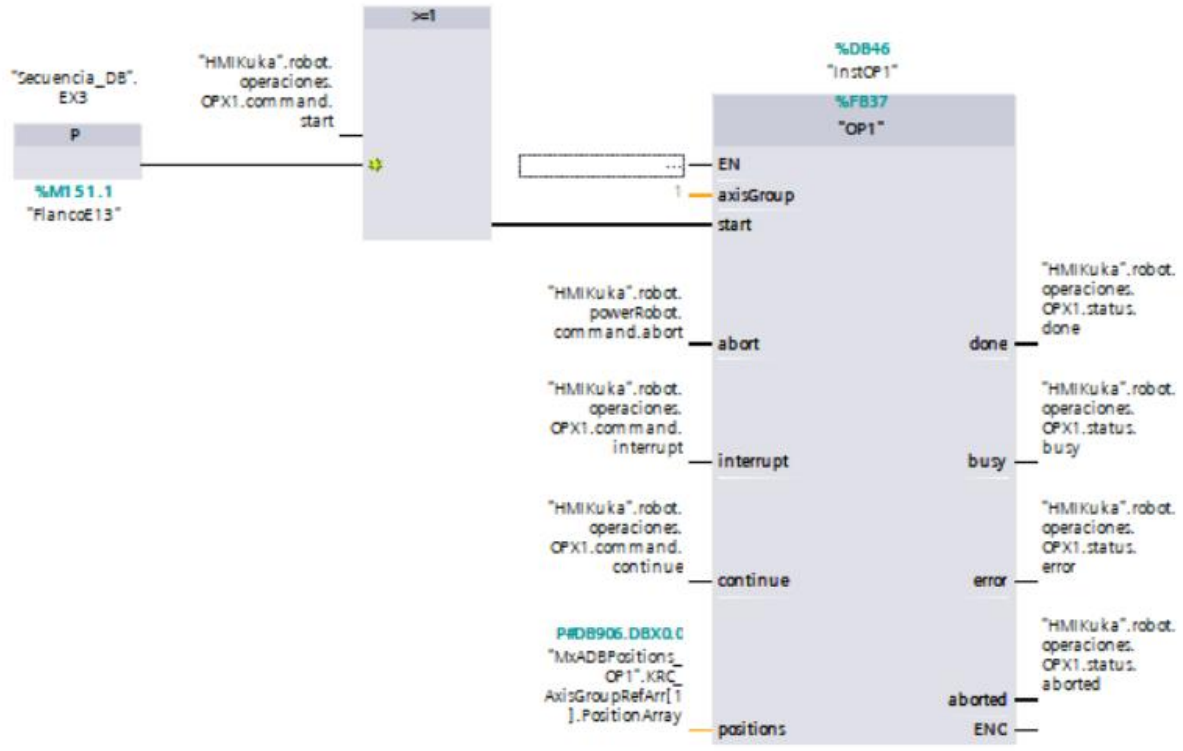


Ilustración 22. Bloque OP1

- KRC_WriteAxisGroup

  Tras llevar a cabo el procesamiento del programa en la CPU SIMATIC, este bloque transfiere los datos al robot.

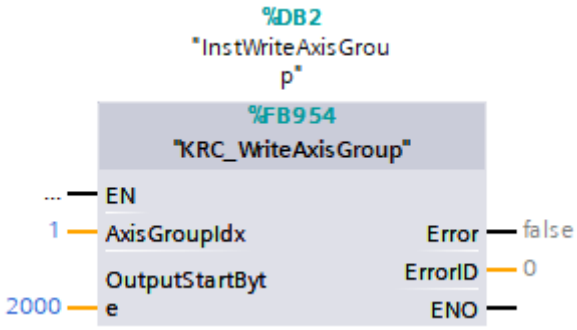


Ilustración 23. Bloque KRC_WriteAxisGroup

2.2.5. GestionarPedido[FB31]

Este bloque comprueba los datos del telegrama que envía el agente para realizar un nuevo pedido, y comprueba las condiciones para mantener el pedido abierto. Si todo es correcto, llama al FB 'Secuencia', que gobierna la ejecución de las operaciones.

a) Pedido activo

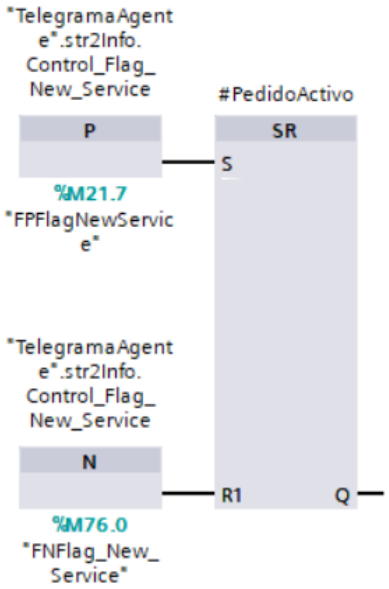


Ilustración 24. Biestable pedido activo

Este biestable permanece activo durante toda la evaluación de errores del FB31, y durante toda la ejecución del pedido. La variable de activación y desactivación es un parámetro del telegrama entre Agente y PLC.

b) Error número de servicio

El segundo segmento del FB31 comprueba si el servicio solicitado está dentro del rango permitido. Solo se dispone de 10 servicios, por lo que un número superior causaría un error, e interrumpiría el programa. El error se puede reconocer desde el HMI, para poder comenzar otro servicio.

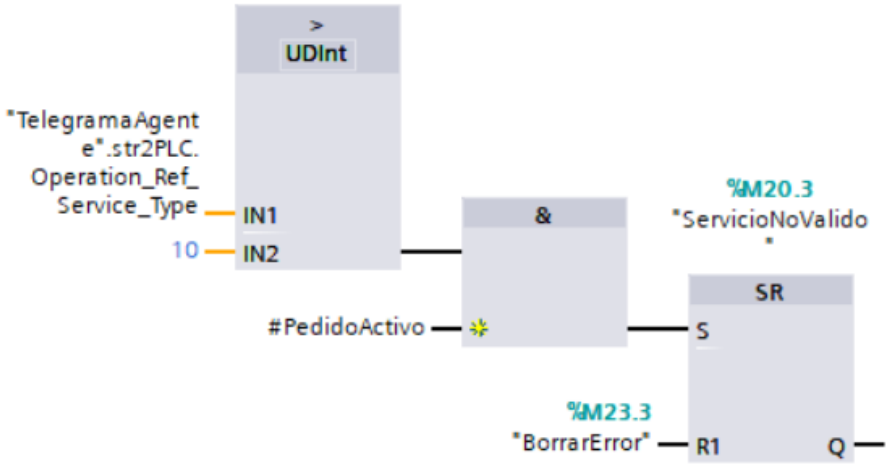


Ilustración 25. Error numero de servicio

c) Error número de ítems

El tercer segmento comprueba si el número de ítems está en rango. Solo se permiten 6 ítems por servicio, por lo que un número mayor causaría un error. Al igual que en el anterior, se puede reconocer el error desde el HMI.

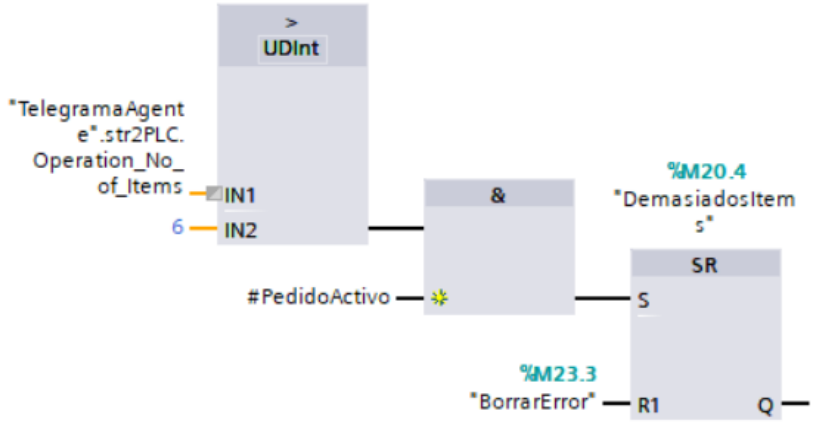


Ilustración 26. Error numero de ítems

d) Ejecutar pedido

Se ejecuta el pedido una vez se ha comprobado que no existen errores.

Ilustración 27. Ejecutar pedido

e) Reseteo flag New Service

El flag 'New Service' es uno de los parámetros del telegrama de comunicaciones entre el Agente y el PLC. Se ha utilizado antes, para la condición de 'Pedido activo'.

Existen cuatro condiciones que resetean el flag.

- Servicio completado: Se resetea el flag una vez el robot termina la ejecución del servicio
- Error pedido: Si ocurre alguno de los dos errores que comprueba el FB31, se activa esta marca.
- Abort: Se finaliza el pedido inmediatamente si se activa la opción de abortar
- ODK Unload: Si se finaliza la conexión con el Agente, se resetea el flag.

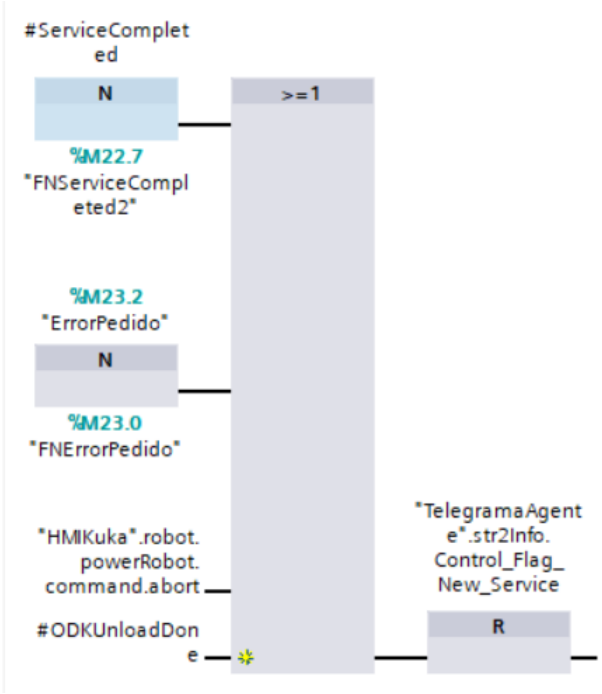


Ilustración 28. Reset flag New Service

2.2.6. EscribirDatosAgente[FB24]

Este FB lleva a cabo las tareas de registro de datos que se llevan a cabo cada vez que se inicia o finaliza un servicio o ítem.

a) Timestamp de inicio de servicio

Esta función lee información del reloj de la CPU, y devuelve un valor en formato LDT. Con el flanco positivo del inicio de un OP100 se produce esta captura.

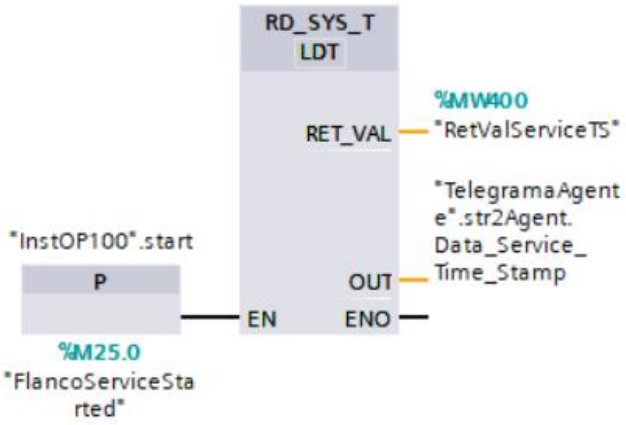


Ilustración 29. Timestamp de inicio de servicio

b) Timestamp de inicio de ítem

Al igual que en el caso anterior, se lee el reloj de la CPU y se envía el dato en formato LDT al agente por medio del telegrama de salida. En este caso, se produce la captura al iniciarse cualquiera de las operaciones OP1-6.

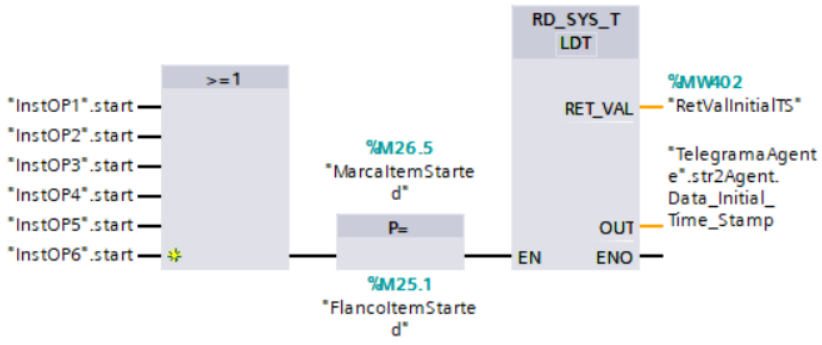


Ilustración 30. Timestamp de inicio de ítem

c) Ítem completado

Una vez se completa un ítem, se activa este segmento. Tiene varios efectos, entre los que está activar el flag de ítem completado.

Además, la marca 'ItemDone' se emplea para trasladar al telegrama de salida todos los datos de la producción (ID de máquina, referencias de orden y de lote, tipos de servicio y subproducto). Además, registra el timestamp de finalización.

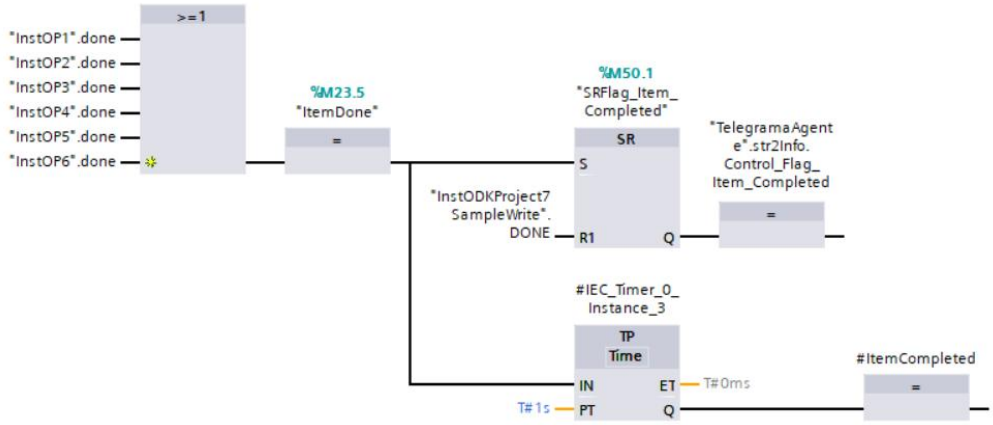


Ilustración 31. Fin de item

Después de este segmento hay otros dos mas, que registran el número del ítem finalizado, y el final de secuencia, aunque no realizan ninguna otra operación relevante.

2.2.7. Secuencia[FB30]

Esta función determina el orden en el que se ejecutan las operaciones, además de discriminar cuales deben activarse, en base al número de piezas solicitadas. Se basa en una sucesión de etapas, que irán activando las operaciones instanciadas en el FC 'SiemensKuka' (ver ilustración 10).

a) Etapa inicial

Ilustración 32. Etapa inicial

Para activarse, la etapa inicial necesita que se cumpla uno de dos pares de condiciones. Ya sean PIM e INIT3 (que indican que el robot está en Home, y que no hay mas etapas activadas), o las dos condiciones que indican que el servicio anterior ha terminado. La etapa se desactiva al activarse la siguiente.

b) OP100

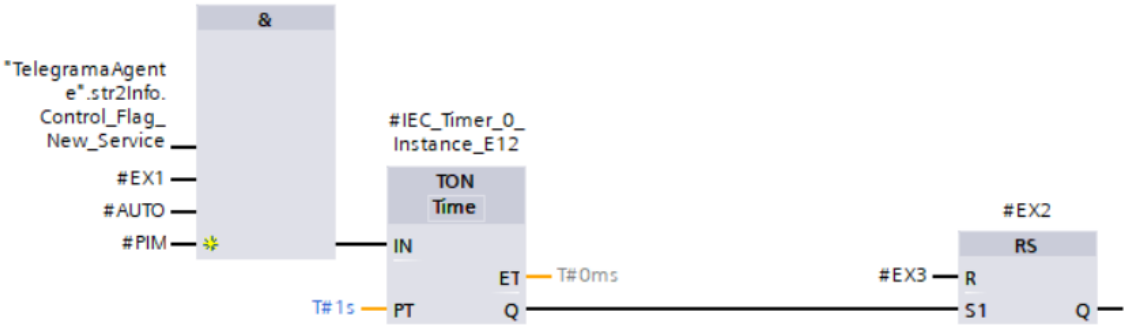


Ilustración 33. Etapa OP100

La siguiente etapa activa la operación OB100. Esta operación es la encargada de mover el pallet de las bases desde la zona de entrada hasta la zona de montaje. Se activa una vez completada la etapa inicial, se solicita un nuevo servicio, el modo automático está activado y el robot está en home.

c) OP1-6

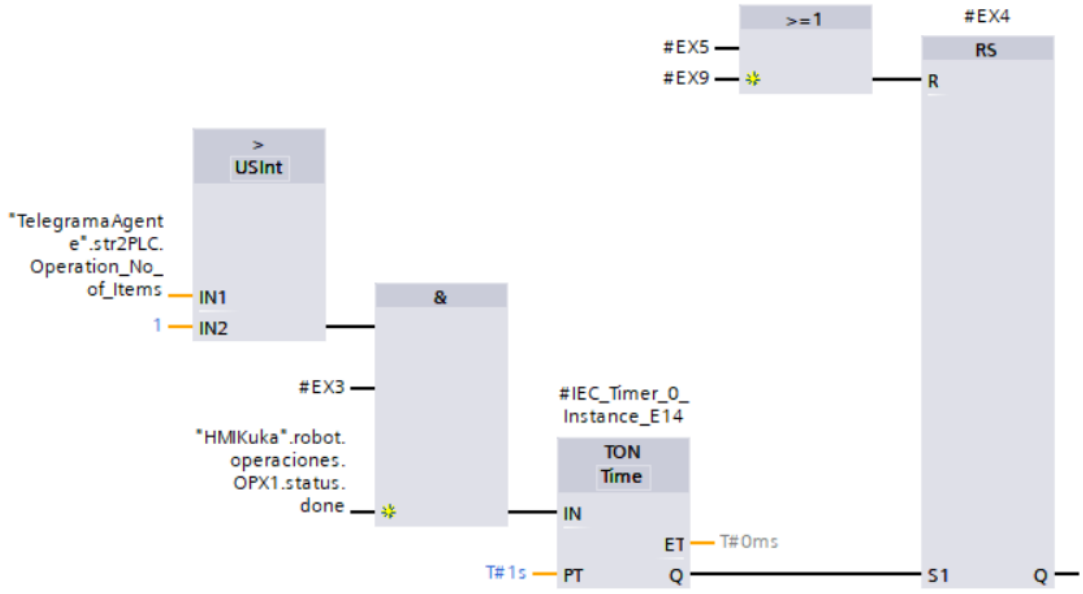


Ilustración 34. Etapa correspondiente a OP2

Los siguientes segmentos van activando las etapas correspondientes a las operaciones OP1 a OP6. La etapa OP1 se activa siempre, pero las siguientes están condicionadas a que haya mas de un número determinado de piezas. La OP2 se activará solo si hay mas de una pieza, la OP3 si hay mas de dos, y así sucesivamente.

Estas etapas se desactivan al activarse la siguiente, o al activarse la etapa 9, llamada 'etapa vacía', que se activa cuando la operación correspondiente al mayor número de piezas ha terminado de ejecutarse. De esta forma, se produce la desactivación aun cuando la siguiente etapa no se active, al no haber suficientes piezas programadas.

Finalmente, la secuencia termina con un contador de 1s, para asegurar que el robot ha tenido tiempo suficiente como para volver a su estado inicial.

2.2.8. Operaciones[FBs]

Se denominan operaciones a las secuencias de movimientos que el robot debe realiza. La numeración, de OP1 a OP6, se corresponde con la posición que las bases tienen en el pallet. También se dispone de las OP100 y OP200, que corresponden al desplazamiento del pallet desde la entrada a la posición de montaje, y desde ésta a la de salida, respectivamente.

Ilustración 35. Pallet de bases, con un montaje completo

4	**5**	**6**
1	**2**	**3**

Tabla 6. Numeración de los huecos del pallet

Cada operación de OP1 a OP6 se ha subdividido en 5 partes, correspondientes al inicio, el rodamiento, el bulón, la tapa interior y la tapa exterior, además de alguna condición de inicio y fin. Se ha diseñado de esta manera para poder activar o desactivar partes dependiendo del servicio que se haya requerido.

a) Segmento inicial

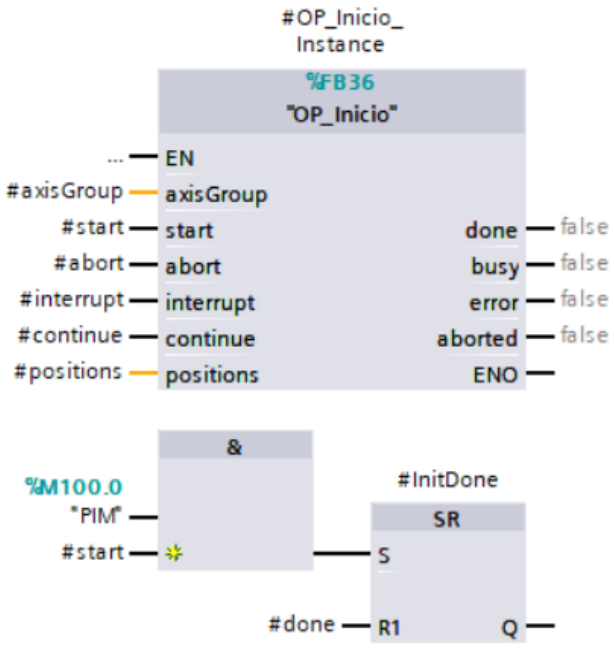


Ilustración 36. Segmento de inicio

El segmento inicial se compone de la FB OP_Inicio, y de la condición inicial, que comprueba si el robot está en posición 'HOME', y si se ha dado la señal de empezar.

El bloque OP_Inicio contiene a su vez varios bloques de la librería mxAutomation, necesarios para la ejecución de la operación, y se mantiene activo hasta el final de la misma.

Los bloques incluidos en OP_Inicio son los siguientes:

- KRC_Abort: Su función es abortar todas las instrucciones activas, y detener todos los movimientos cuya entrada (Execute) sea un flanco positivo.

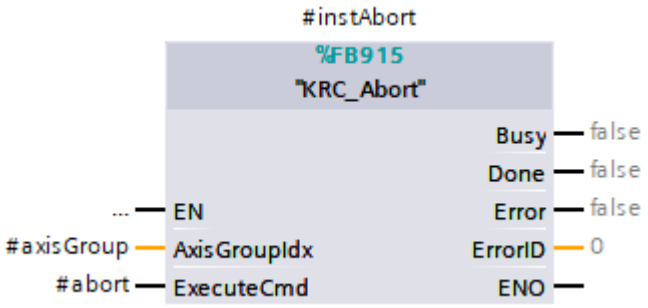


Ilustración 37. KRC_Abort

- KRC_Interrupt: Interrumpe todos los comandos al activar las instrucciones BRAKE, o BRAKE-F en la controladora del robot.

- KRC_Continue: Para continuar un programa interrumpido.

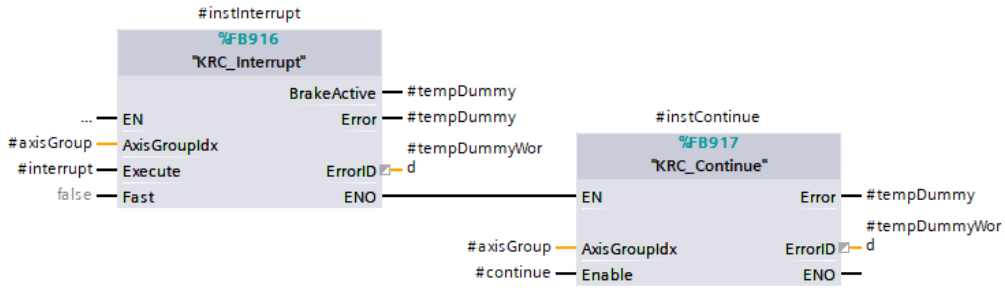


Ilustración 38. KRC_Interrupt y KRC_Continue

- mxA_ValuesToCOORDSYS: Escribe elementos en la estructura “COORDSYS”, que contiene información sobre el sistema de coordenadas seleccionado. Para este caso se definen como 0 las tres variables de herramienta, base, y modo de interpolación. Esto implica que el robot moverá la herramienta sobre una pieza fija.

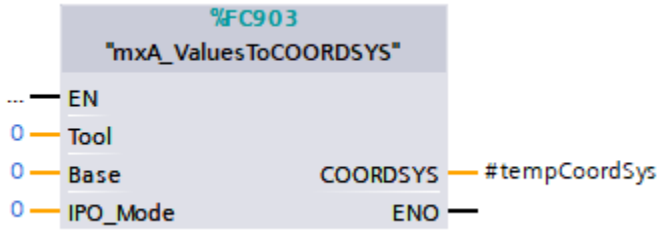


Ilustración 39. mxA_ValuesToCOORDSYS

- mxA_ValuesToAPO: Escritura de elementos en la estructura “APO”, que contiene información sobre la aproximación a puntos intermedios del movimiento del robot. Situar todas las entradas a cero implica que la pinza debe pasar exactamente por cada punto de la trayectoria. En caso contrario, el robot podría realizar una trayectoria mas redondeada, recortando a través de los vértices que conforman los puntos.

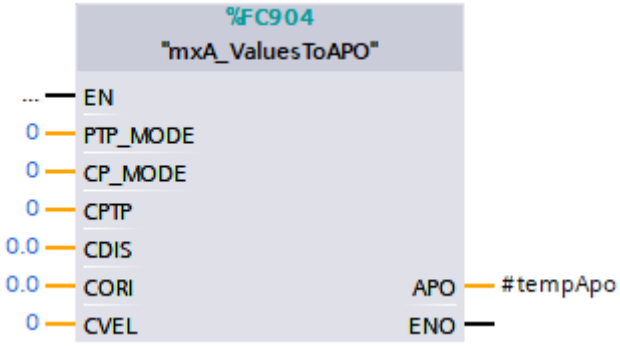


Ilustración 40. mxA_ValuesToAPO

- MC_MoveAxisAbsolute: Este bloque es especialmente relevante, y aparecerá varias veces más en diferentes operaciones. Permite mover el robot a una posición concreta definida por la posición de cada una de las articulaciones del robot, definida con el tipo “E6AXIS”. Las posiciones se guardan en un DB instanciado en la cabecera de la OP, llamado ‘positions’, y la número 1 corresponde a la posición ‘Home’. También se deben definir velocidad y aceleración, y el modo ‘QueueMode’, que en un valor 2 almacena la instrucción en el buffer, para asegurar una ejecución tipo FIFO.

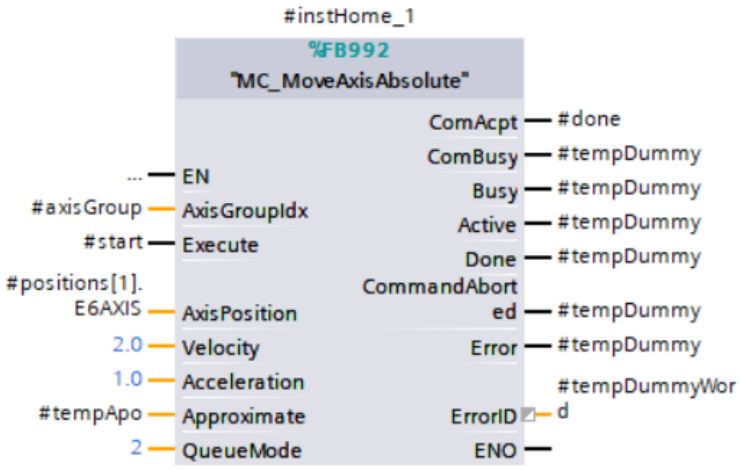


Ilustración 41. MC_MoveAxisAbsolute

b) OP_Rodamiento a OP_Tapa_exterior

A partir de este punto, las operaciones OP1-6 tan solo difieren en los diferentes puntos de las trayectorias, siendo su estructura prácticamente idéntica. Sin embargo difieren en algunos puntos poco relevantes, pero que hace imposible realizar una única operación con diferentes instancias.

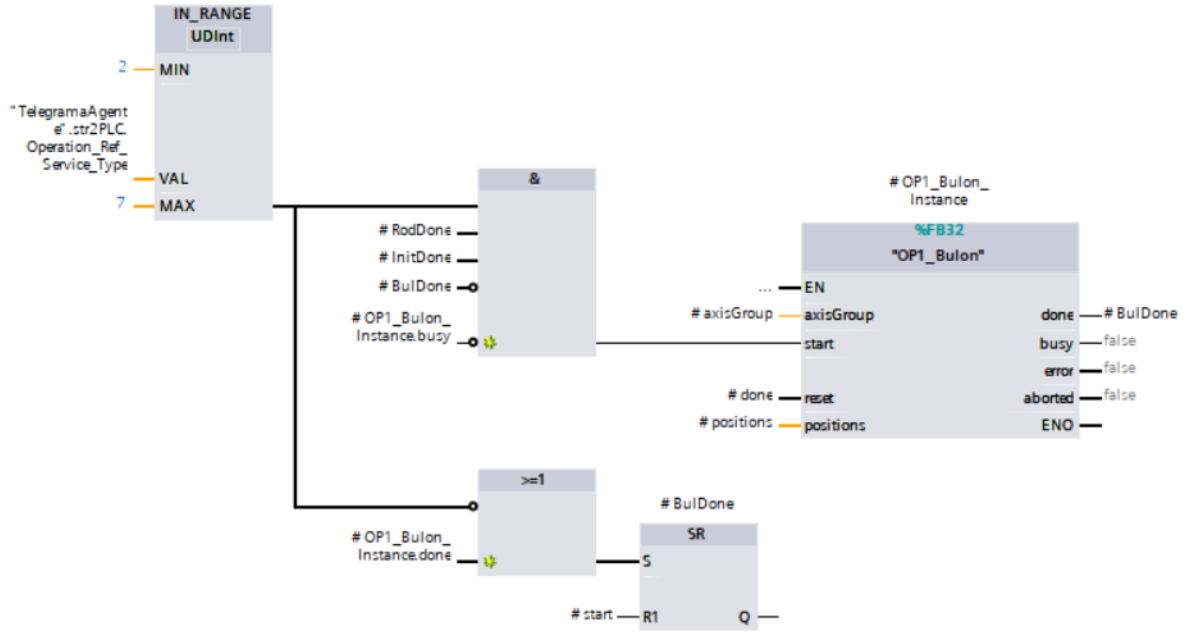


Ilustración 42. OP1_Bulón

Las OPs asociadas a una pieza concreta se instancian como se puede ver en la imagen. Es necesario establecer una serie de condiciones para su activación, siendo la primera de todas que el servicio solicitado incluya la pieza correspondiente. También se requiere que las piezas anteriores hayan sido ya colocadas, y que se haya superado la etapa inicial.

Se programa, además, un biestable que active o desactive la señal 'done'. Se activará en el caso de que el bloque OP acabe su ejecución, o si el servicio solicitado no requiere esta pieza; y se desactivará cuando se active el bloque OP1, para poder empezar de nuevo.

Los bloques de cada una de las piezas contienen la secuencia de movimientos necesaria para recogerla de su almacén y colocarla en la base.

Todos los bloques empleados en estas operaciones están instanciados en la cabecera, por lo que sus DBs asociados no son accesibles externamente. Se encadenan asignando el parámetro 'Done' del bloque anterior a la entrada 'Execute' del siguiente.

A continuación se explican los FBs de MotionControl, empleados en esta secuencia.

- MC_MoveLinearAbolute (sic): Mediante este bloque, el robot puede realizar una trayectoria lineal, empleando la ruta mas corta entre dos puntos. Para ello hay que especificar qué sistema de coordenadas ('COORDSYS'), se desea llevar a un punto especificado, definido en coordenadas cartesianas ('E6POS').

  Este bloque tiene mucha utilidad en las operaciones de pick&place, donde son necesarios movimientos lineales de precisión. En la operación de recogida de la pieza, la entrada 'Execute' se retrasa tres segundos, para asegurar que la pinza ha cogido la pieza.

  Al igual que en el bloque MC_MoveAxis, se deben especificar velocidad, aceleración, y modo de cola. Además, se especifica el parámetro 'OriType', que controla si el punto central de la herramienta (TCP), se mantiene con orientación constante o variable a lo largo del movimiento. En este caso, se ha definido el valor 1, para que la orientación del TCP se mantenga constante.

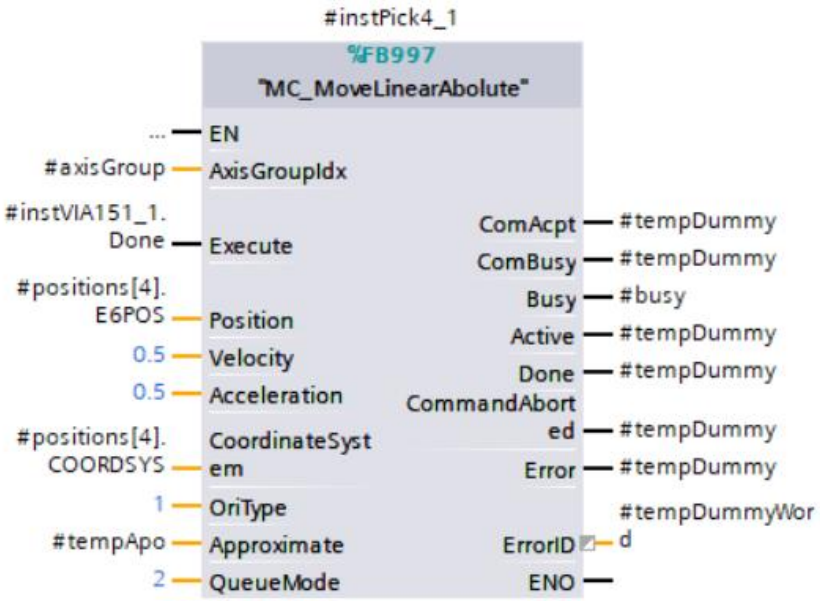


Ilustración 43. MC_MoveLinearAbsolute

- MC_MoveAxisAbsolute: Ya explicado anteriormente, se emplea extensivamente en estos bloques. Se emplea para mover todos los ejes del robot a una posición previamente definida, sin importar su punto de origen (por lo que hay que tener cuidado con su utilización, ya que puede golpear algo al ejecutar trayectorias no controladas). Su ejemplo mas claro es su uso para alcanzar la posición 'Home', aunque en este programa se emplean en la mayoría de trayectorias.

- KRC_WriteDigitalOutput: Este bloque se utiliza para la apertura y cierre de las dos pinzas de la herramienta. Permite escribir sobre una salida digital.

  Mediante la entrada 'Number', se controla cual de las dos pinzas se está controlando. La pinza normalmente abierta se corresponde con una entrada 1, mientras que la normalmente cerrada lo hace con una entrada 2. La entrada 'Value' envía la señal de activación, cerrando la pinza normal abierta, o abriendo la normal cerrada.

  En la siguiente imagen, se observa como el bloque ordena la activación de la pinza normal cerrada (número 2), lo que implica su apertura.

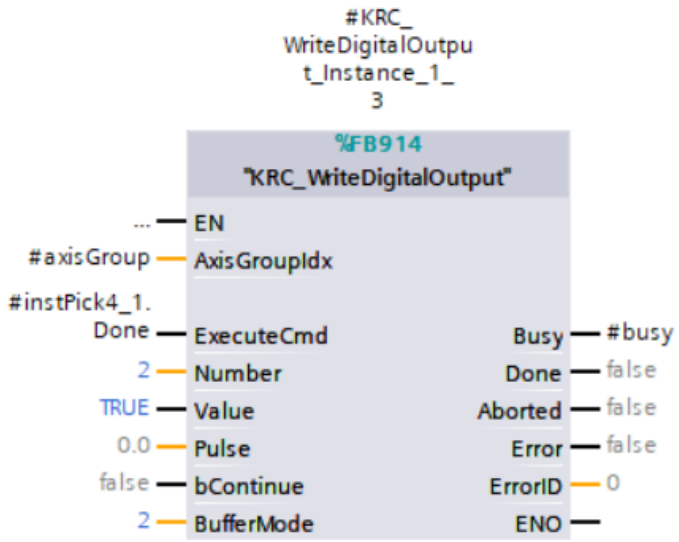


Ilustración 44. KRC_WriteDigitalOutput

Encadenando estos bloques se llevan a cabo los movimientos necesarios para colocar en la base todas las piezas que el servicio requiera.

2.3. HMI

Para realizar el control y supervisión del robot, se va a emplear un HMI de Siemens, hecho con TIA Portal. Se va a emplear una versión modificada del HMI proporcionado en el proyecto de ejemplo de Siemens para las librerías mxAutomation.

2.3.1. Hardware

En primer lugar, es necesario establecer una conexión HMI. Para ello, se debe introducir en el hardware del PLC y del PC el módulo WinCC RT Advanced.

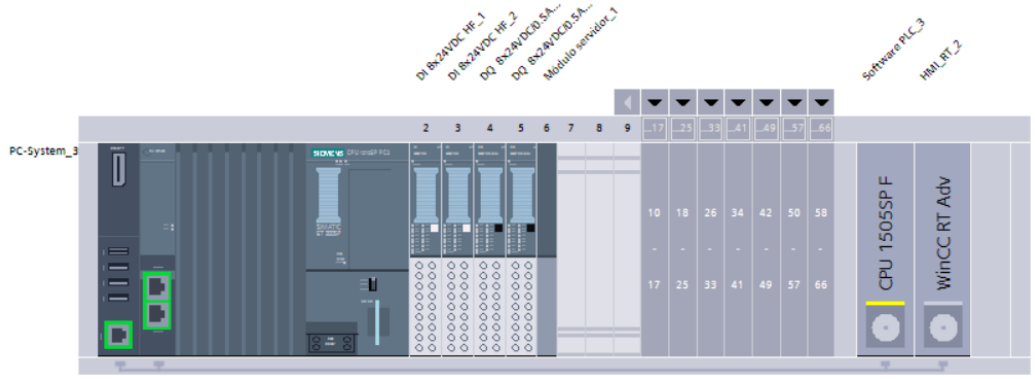


Ilustración 45. Hardware PLC

Posteriormente, se debe seleccionar 'Conexión HMI', en la vista de redes, y conectar los módulos HMI del PC y PLC, y el módulo CPU 1505SP F del PLC.

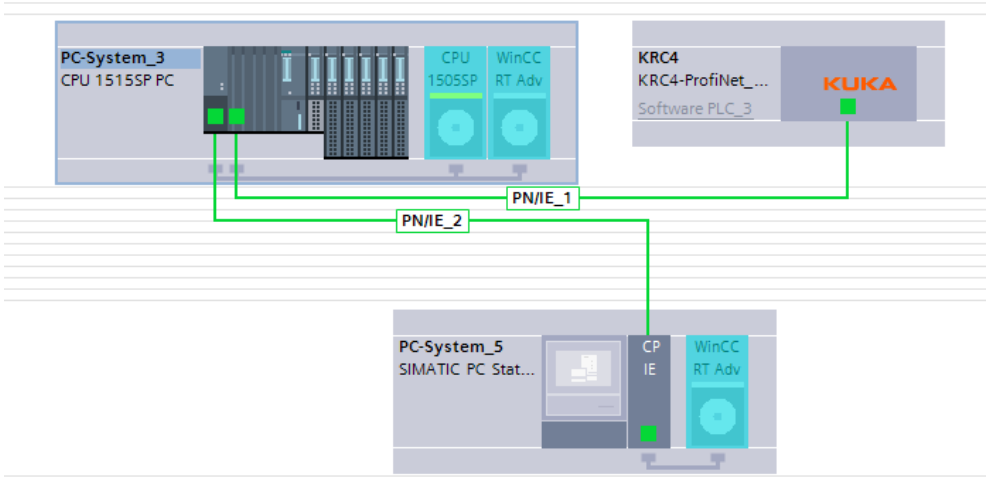


Ilustración 46. Conexiones hardware

2.3.2. Imagen HMI

Ilustración 47.Imagen HMI completa

El HMI tiene numerosas funciones, que se detallan a continuación.

- Controles básicos: Mediante el HMI se enciende y apaga el robot, y se puede resetear y abortar el programa actual. También se puede llevar el robot a Home en cualquier momento.

- Acuse de errores: En la parte superior de la pantalla se encuentran varios mensajes de error, que se vuelven visibles cuando alguno de ellos sucede. El botón de borrar fallo y desechar pedido permite acusar el error, y abortar el pedido fallido.

- Control de velocidad: Se puede configurar manualmente que porcentaje de velocidad se desea emplear, entre el 0% y la máxima programada. Resulta de utilidad para realizar pruebas.

- Visualización: Se dispone de información del estado del robot, de la tensión, y del intérprete. También de las posiciones de los ejes y de la pinza.

- Control de servicios: Se puede simular el agente externo, configurando desde el HMI el tipo de servicio y número de ítems. También se puede activar manualmente cada una de las OP1-6, y las OP100 y OP200.

- Control de las pinzas: Se pueden abrir o cerrar manualmente las pinzas normal cerrado (PC) y normal abierto (PA).

- Modo manual: Se dispone de dos modos de movimiento manual. El primero es Jog in Axis, mediante el cual se puede mover cada uno de los ejes de forma independiente, controlando su posición en grados. El segundo es Jog in Base, que permite controlar el sistema de referencia de la pinza y moverlo según los ejes cartesianos.

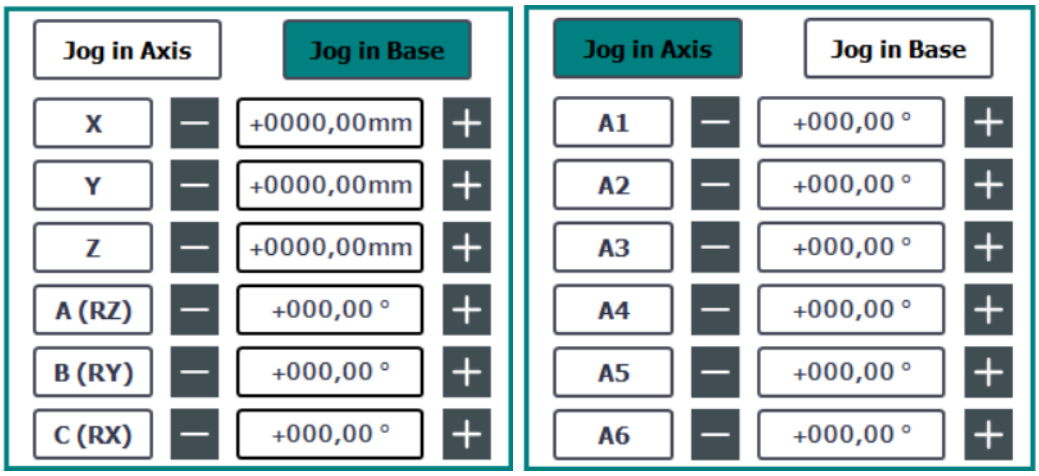


Ilustración 48. Jog in Base y Jog in Axis

En este caso, las funciones añadidas en este trabajo son las relacionadas con el control de servicios. A continuación se va a proceder a explicar cómo se han configurado.

a) Botones

Para la configuración de los botones se va a tomar como ejemplo el botón que permite activar la OP1 una vez. El resto de botones se pueden configurar igual.

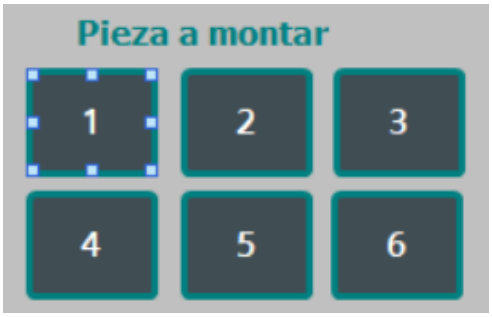


Ilustración 49. Botonera de selección de operación

En primer lugar, se deben configurar como variables HMI las variables sobre las que se vaya a actuar. Mediante la tabla de variables de HMI, se crea una nueva variable, que tras seleccionar la conexión previamente configurada, se puede asociar con una de las variables del PLC. En este caso, se va a establecer una asociación con una variable contenida en el DB HMIKuka. Este DB contiene todos los parámetros relativos al robot y a las operaciones, y entre ellos se encuentran los empleados para almacenar las señales de inicio de las operaciones.

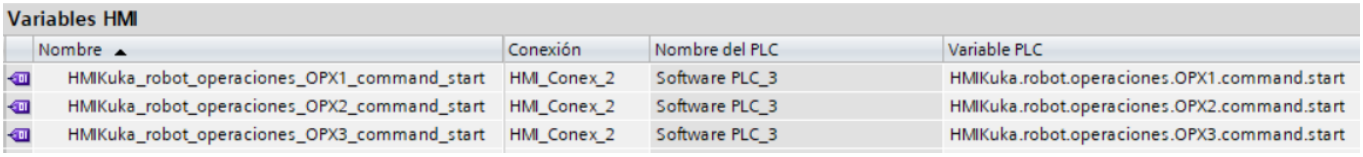


Ilustración 50. Tabla de variables HMIç

La variable a emplear en este caso corresponde a OPX1_command_start. Tras introducir en la imagen del HMI un elemento 'Botón', se accede a la pestaña Propiedades/Eventos, donde se puede configurar la acción que ejecutará el botón ante determinadas interacciones.

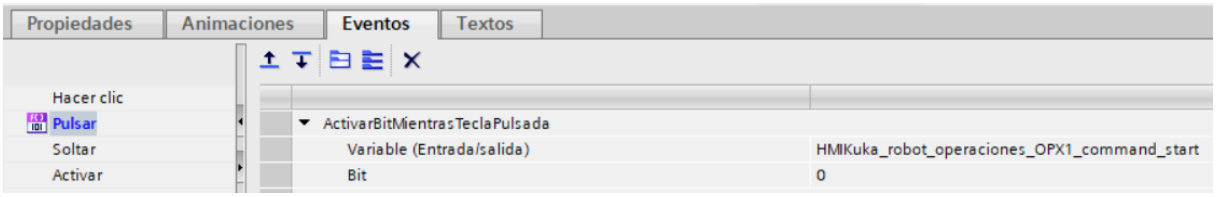


Ilustración 51. Pestaña de eventos

En este caso, se va a configurar la acción 'Activar bit mientras tecla pulsada', al pulsar el botón. Esta acción va a permitir que la variable HMI se active mientras el botón esté pulsado. Al ser necesario tan solo un pulso, no es necesario que la señal se mantenga activada mas tiempo.

Posteriormente, se puede proceder a configurar la apariencia y el texto del botón en la pestaña Propiedades. Se procederá de forma similar con todos los botones, teniendo cuidado de escoger la acción adecuada para cada caso al configurar el evento.

b) Selectores

El HMI cuenta con dos selectores para escoger el tipo de servicio y el número de ítems.

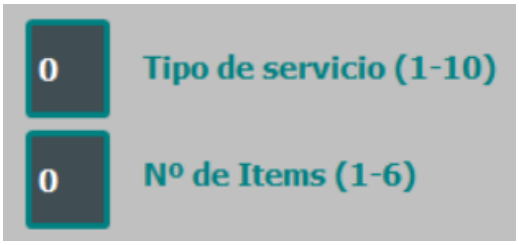


Ilustración 52. Selectores del HMI

Estos selectores sustituyen al agente en el caso de que el ODK no esté activo. Este comportamiento se modela en la FC 'ComunicacionODK', que gobierna las conexiones del ODK con el programa.

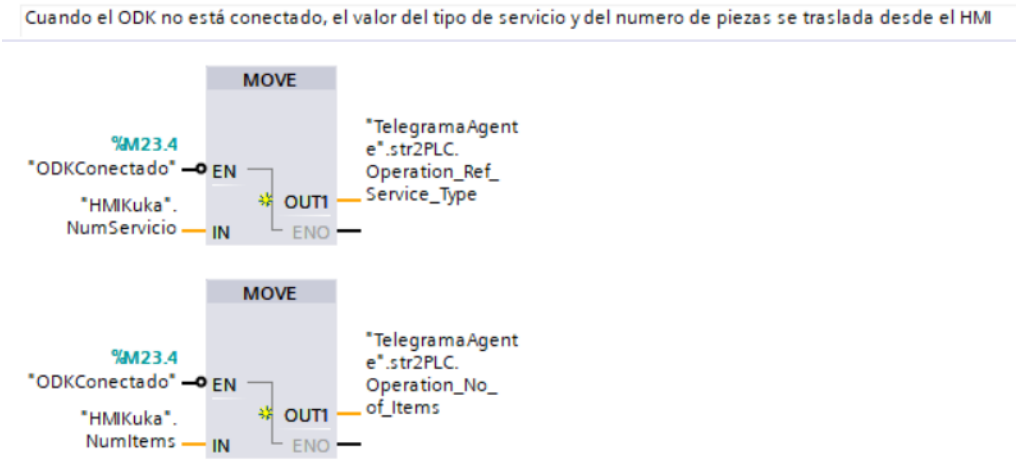


Ilustración 53. Segmento de Comunicación ODK

Al igual que en el caso de los botones, se debe configurar una variable HMI, que en este caso será de tipo INT en lugar de BOOL. Después, se debe introducir en la imagen un campo E/S.

En este caso el selector se asocia directamente a la variable mediante la pestaña Propiedades, en el apartado General. De esta forma, el valor de la variable será siempre el que se escriba en el selector. Al ser un campo E/S, al introducir un número y pulsar Enter, la variable adoptará el valor seleccionado. También se podría usar como campo de salida, pero la variable asignada no cambia de valor por acción del programa, por lo que en este caso no se emplea esa posibilidad.

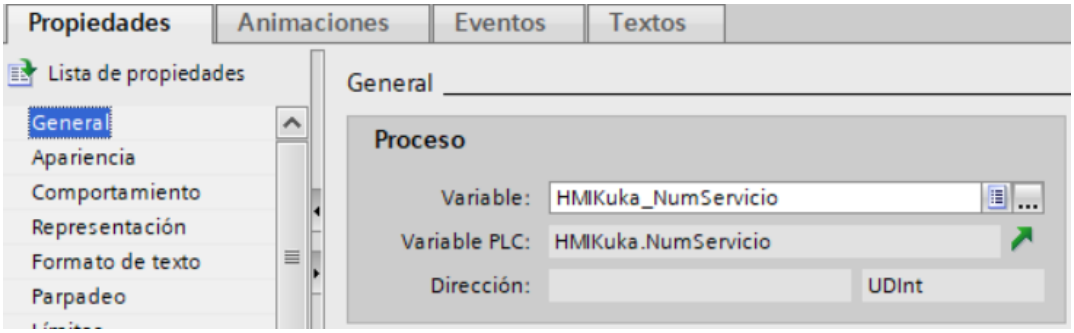


Ilustración 54. Variable de proceso asignada

2.4. Envío de datos por MQTT

En la segunda parte de este trabajo se va a tratar el desarrollo de una aplicación de envío de datos a la nube empleando el protocolo MQTT. En primer lugar, se explicará en qué consiste y como se emplea el ODK. Posteriormente se abordará MQTT, el cliente Paho, y el broker Mosquitto. En último lugar, se explicará el uso de la base de datos Influx, y el asistente Telegraf.

2.4.1. ODK (Open Development Kit)

La interfaz ODK se emplea para facilitar esta integración de lenguajes de alto nivel con el controlador. Se puede llegar a emplear incluso en aplicaciones con requerimientos de tiempo real, aunque el método para emplear esta funcionalidad no se explica aquí. [7]

Para utilizar las aplicaciones de alto nivel, ODK genera automáticamente FBs que permiten cargarlas o descargarlas en el programa de PLC. Para utilizarlas, se deben cargar primero, y se puede finalizar su ejecución al descargarlas.

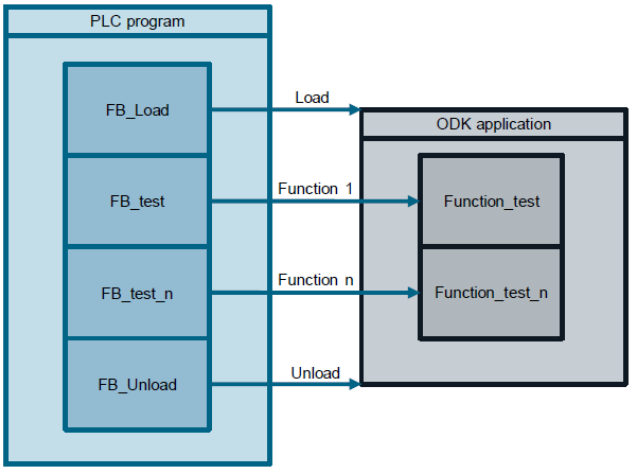


Ilustración 55. Secuencia de utilización de ODK

a) Servidor web

El SIMATIC ET200SP Open Controller dispone de un servidor web, que es necesario activar para acceder a la CPU. Se encuentra en la pestaña de propiedades del dispositivo hardware.

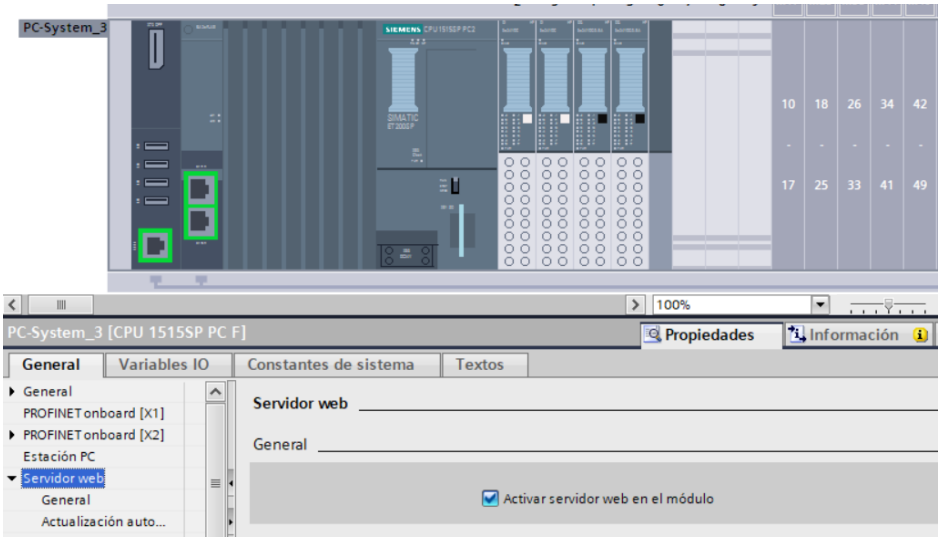


Ilustración 56. Servidor Web

Posteriormente, hay que configurar un usuario y contraseña en el mismo apartado de servidor web, al que hay que otorgar permisos de lectura, escritura y borrado de archivos.

b) Proyecto ODK

Siemens proporciona un proyecto de ejemplo de ODK sobre el que se puede trabajar. Se trata de un programa escrito en C++, que contiene las funciones básicas que debe ejecutar el ODK. En este programa se pueden añadir funcionalidades, como la de comunicación con MQTT que va a emplearse en este proyecto.

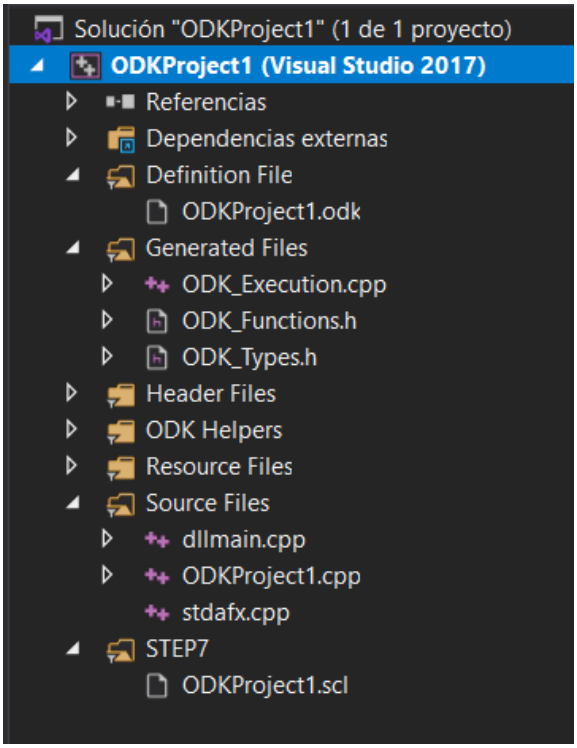


Ilustración 57. Estructura de carpetas del proyecto ODK

Para emplear este proyecto en el controlador, se debe compilar, con las opciones Release, y x86. Tras esta compilación, se generan dos archivos en la carpeta 'Release\ ODK_1500S_Samples_Code' del proyecto, que van a crear nuevos archivos en el programa de TIA Portal.

- Un archivo .dll, que se debe copiar a la carpeta 'C:\ProgramData\Siemens\Automation\ODK1500S'

- Un archivo .scl que se debe llamar desde TIA Portal. Se debe hacer desde la pestaña 'Fuentes externas/Agregar nuevo archivo externo'.

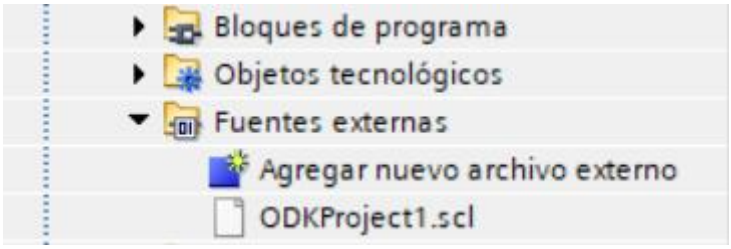


Ilustración 58. Fuentes externas

Después hay que seleccionar el .scl con el botón derecho, y escoger ‘Generar bloques a partir de fuente’. Tras de estas operaciones, se han creado cuatro nuevos FBs. Estos FBs no se deben manipular, y se emplearán en el FC2 ‘ComunicacionODK’, para cargar, descargar, y transmitir los datos entre el ODK y el controlador.

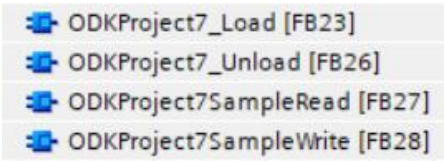


Ilustración 59. FBs de comunicacion con ODK

c) ComunicacionODK [FC2]

Este bloque se crea para gobernar la relación del ODK con el controlador. Aquí se encuentran instanciados los cuatro FBs importados del proyecto, y se asignan sus entradas, salidas, y condiciones.

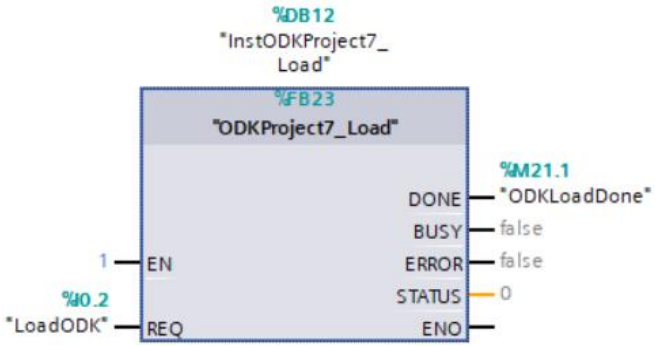


Ilustración 60. Carga de ODK

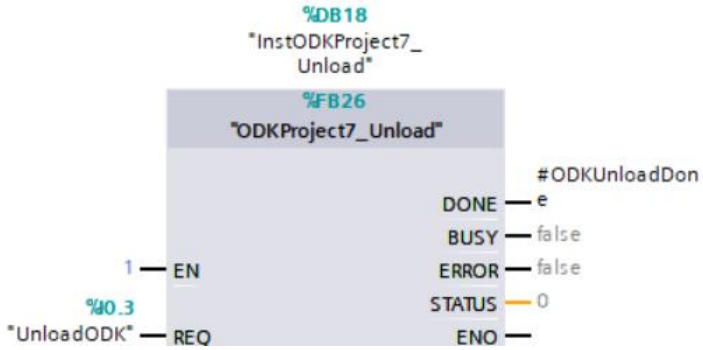


Ilustración 61. Descarga de ODK

Los bloques de carga y descarga de ODK se activan mediante una señal externa, asociada a un interruptor. Una vez se realiza la carga, se ejecuta la conexión, y ya puede utilizarse el ODK. Sus salidas funcionan como entradas del biestable 'ODKConectado'.

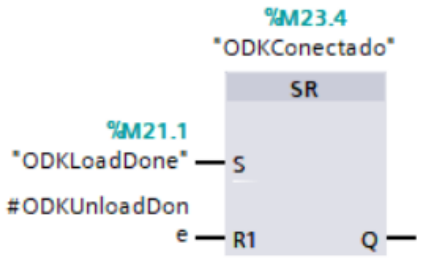


Ilustración 62. ODK conectado

El bloque de lectura solo se activa si no hay ningún pedido activo en el momento, ya que su función es leer el telegrama del agente, que contiene información como el número de ítems o de servicio, que no debe cambiarse en mitad de una operación. Solo se activa con el ODK cargado, y dispone de un retraso de 2s, para dejar un tiempo entre servicios.

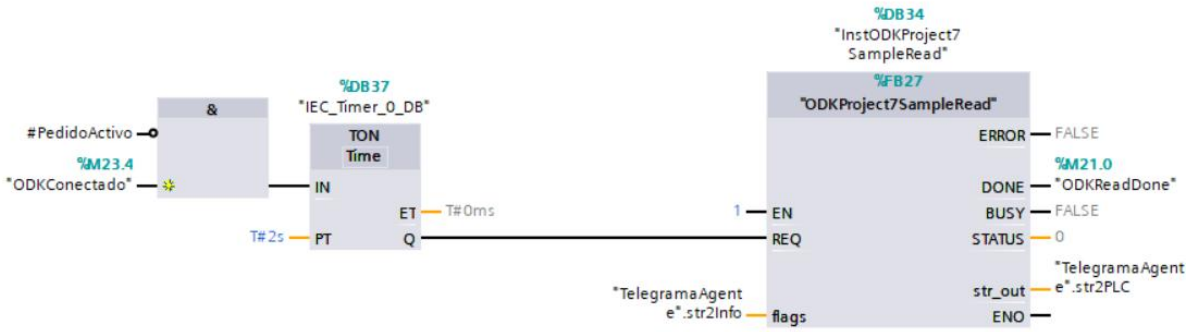


Ilustración 63. Lectura de ODK

El bloque de escritura se activa al terminar un ítem, o al terminar un servicio. Envía al agente la información generada en ese momento, como timestamps, o el número del ítem o servicio recién terminado.

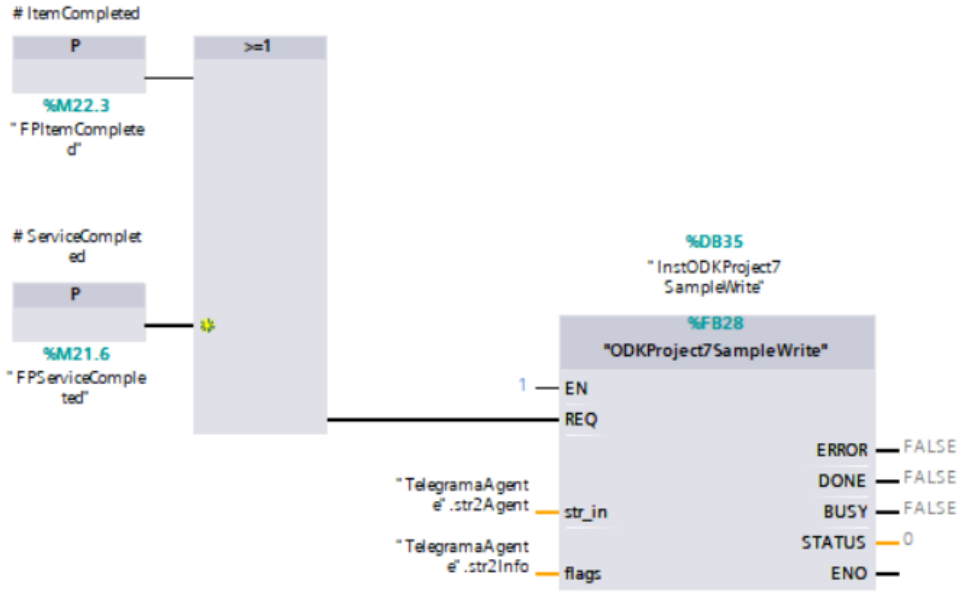


Ilustración 64. Escritura de ODK

En ausencia del agente externo, se dispone de dos archivos .txt que incluyen información de entrada y salida que éste debería gestionar. Su ubicación debe incluirse en el proyecto del ODK.

- ODK2PLC.txt : contiene los valores de las variables que el agente debe enviar al PLC, tales como el número de piezas o de servicio. La acción de lectura lee este archivo.

- PLC2ODK.txt : contiene los datos recogidos por el PLC durante la ejecución del pedido. No recoge timestamps. La acción de escritura vuelca la información del PLC a este archivo.

Cuando el ODK está descargado, se puede especificar el número de ítems o de servicio desde el HMI.

d) TelegramaAgente[DB11]

Los procesos de lectura y escritura involucran al DB11, que también es usado de forma extensiva en otras partes del programa. Este DB replica la estructura del telegrama que envía y recibe el agente. Contiene tres estructuras de datos.

- Str2info: En esta estructura se encuentran las variables de lectura y escritura.

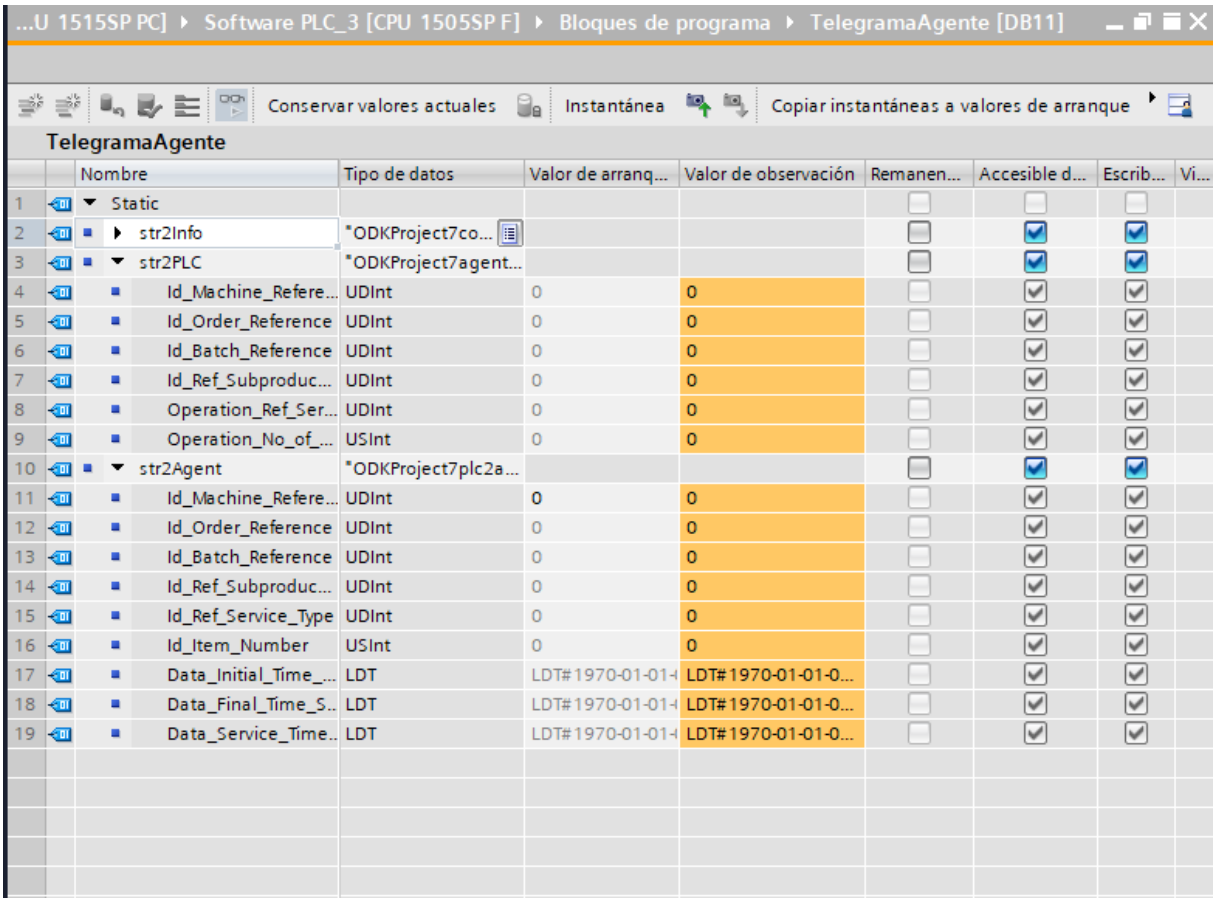
    - Control_Flag_New_Service: Indica la solicitud de un nuevo servicio
    - Control_Flag_Item_Completed: Indica ítem completado
    - Control_Flag_Service_Completed: Indica servicio completado

- Str2PLC: Contiene datos de lectura, del agente al PLC. Entre los datos que se incluyen aquí hay parámetros del proceso (número de ítems y tipo de servicio); y datos identificativos (los que empiezan por Id), que no afectan al funcionamiento, pero son necesarios para identificar el proceso que se va a llevar a cabo.

    - Id_Machine_Reference: Referencia de la máquina
    - Id_Order_Reference: Referencia del pedido
    - Id_Batch_Reference: Referencia del lote
    - Id_Ref_Subproduct_Type: Referencia de subproducto
    - Operation_Ref_Service_Type: Número de servicio a ejecutar
    - Operation_No_of_Items: Número de ítems solicitados

- Str2Agent: Contiene datos de escritura, del PLC al agente. Los datos son los mismos que los de Str2PLC, aunque los datos de ítem y servicio especifican el último terminado. También se incluyen los timestamps de inicio de ítem, fin de ítem, y fin de servicio.

    - Id_Machine_Reference: Referencia de la máquina
    - Id_Order_Reference: Referencia del pedido
    - Id_Batch_Reference: Referencia del lote
    - Id_Ref_Subproduct_Type: Referencia de subproducto
    - Id_Ref_Service_Type: Número de servicio actual
    - Id_No_of_Items: Número del ítem finalizado
    - Data_Initial_Time_Stamp: Timestamp de inicio de ítem
    - Data_Final_Time_Stamp: Timestamp de fin de ítem
    - Data_Service_Time_Stamp: Timestamp de



	Nombre	Tipo de datos	Valor de arranq...	Valor de observación	Remanen...	Accesible d...	Escrib...	Vi...
1	Static							
2	str2Info	"ODKProject7co...						
3	str2PLC	"ODKProject7agent...						
4	Id_Machine_Refere...	UDInt	0	0				
5	Id_Order_Reference	UDInt	0	0				
6	Id_Batch_Reference	UDInt	0	0				
7	Id_Ref_Subproduc...	UDInt	0	0				
8	Operation_Ref_Ser...	UDInt	0	0				
9	Operation_No_of_...	USInt	0	0				
10	str2Agent	"ODKProject7plc2a...						
11	Id_Machine_Refere...	UDInt	0	0				
12	Id_Order_Reference	UDInt	0	0				
13	Id_Batch_Reference	UDInt	0	0				
14	Id_Ref_Subproduc...	UDInt	0	0				
15	Id_Ref_Service_Type	UDInt	0	0				
16	Id_Item_Number	USInt	0	0				
17	Data_Initial_Time_...	LDT	LDT#1970-01-01-(	LDT#1970-01-01-0...				
18	Data_Final_Time_S...	LDT	LDT#1970-01-01-(	LDT#1970-01-01-0...				
19	Data_Service_Time...	LDT	LDT#1970-01-01-(	LDT#1970-01-01-0...				

Ilustración 65. Telegrama Agente

2.4.2. Protocolo MQTT

MQTT es un protocolo de transferencia de mensajes entre dispositivos, diseñado para ser utilizado en el internet de las cosas (IoT). Funciona mediante el sistema publicista/suscriptor, y está diseñado para ser extremadamente ligero, y funcionar con un mínimo ancho de banda y pocas líneas de código. MQTT define las capas 5 a 7 de OSI. Sus comunicaciones están basadas en TCP/IP.

Los mensajes en MQTT van asociados a un tema concreto, al que los clientes pueden suscribirse. De esta forma recibirán todos los mensajes asociados a ese tema, y a los temas de nivel inferior a él. Los clientes también pueden enviar mensajes a un tema sin estar suscritos. La gestión de los mensajes y las suscripciones se realizan a través de un bróker que centraliza los mensajes, de forma que los clientes no pueden comunicarse entre sí directamente [2].

Empleando el ODK, se va a realizar un envío de datos empleando MQTT a una base de datos en la nube.

a) Mosquitto

Mosquitto carece de interfaz, y se controla mediante la línea de comandos. En su versión mas sencilla, basta con lanzar el programa. Por defecto, escucha en el puerto 1883.

```
pi@raspberrypi:~/Documents $ mosquitto
1646155470: mosquitto version 1.5.7 starting
1646155470: Using default config.
1646155470: Opening ipv4 listen socket on port 1883.
```

Ilustración 66. Ejecución de Mosquitto en una Raspberry Pi

Mediante el comando de lanzamiento también se puede configurar el puerto [-p *número de puerto*], el uso como aplicación en segundo plano [-d], y el acceso al archivo de configuración [-c *archivo*]. Si se desea configurar alguna otra cosa, como pueden ser las opciones de autenticación, se debe hacer mediante el archivo de configuración 'mosquitto.conf'.

Por defecto, el bróker envía mensajes al tema $SYS, relativos a su propio funcionamiento. Se pueden observar los temas del bróker mediante la aplicación MQTT Explorer.

Ilustración 67. Tópico $SYS

Una vez el bróker está activo, se le pueden mandar mensajes desde un cliente. Como no se ha configurado ninguna medida de autenticación, basta con conocer la dirección IP del bróker, y disponer de un medio de conexión.

b) Paho

Paho es una librería que proporciona un cliente MQTT. Es un proyecto open-source, disponible para una gran cantidad de lenguajes de programación, y con diversas funcionalidades. En este proyecto, se empleará la librería para C++, ya que se va a ejecutar dentro del programa ODK.

Client	MQTT 3.1	MQTT 3.1.1	MQTT 5.0	LWT	SSL / TLS	Automatic Reconnect	Offline Buffering	Message Persistence	WebSocket Support	Standard MQTT Support	Blocking API	Non-Blocking API	High Availability
Java	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Python	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔	✔	✔	✖
JavaScript	✔	✔	✖	✔	✔	✔	✔	✔	✔	✖	✖	✔	✔
GoLang	✔	✔	✖	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔
C	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
C++	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Rust	✔	✔	✖	✔	✔	✔	✔	✔	✖	✔	✔	✔	✔
.Net (C#)	✔	✔	✖	✔	✔	✖	✖	✖	✖	✔	✖	✔	✖
Android Service	✔	✔	✖	✔	✔	✔	✔	✔	✔	✔	✖	✔	✔
Embedded C/C++	✔	✔	✖	✔	✔	✖	✖	✖	✖	✔	✔	✔	✖

Ilustración 68. Funcionalidades y disponibilidad de Paho

La versión de Paho para C++ se compone de dos carpetas, una para C, y la otra para C++ en sí. Se deben realizar las siguientes operaciones con ellas para poder llamarlas desde el programa, y para que funcionen en ejecución.

- Añadir la dirección de las carpetas en el proyecto ODK, en la sección de Propiedades-C/C++.

- Añadir la dirección de los archivos .lib en la sección de Propiedades-Vinculador-Entrada.

- Añadir la dirección de las carpetas 'bin' al Path del sistema. Esto debe hacerse en el sistema en el que se vaya a ejecutar el ODK; esto es, en el Windows del PLC. Se hace desde la edición de variables de entorno.

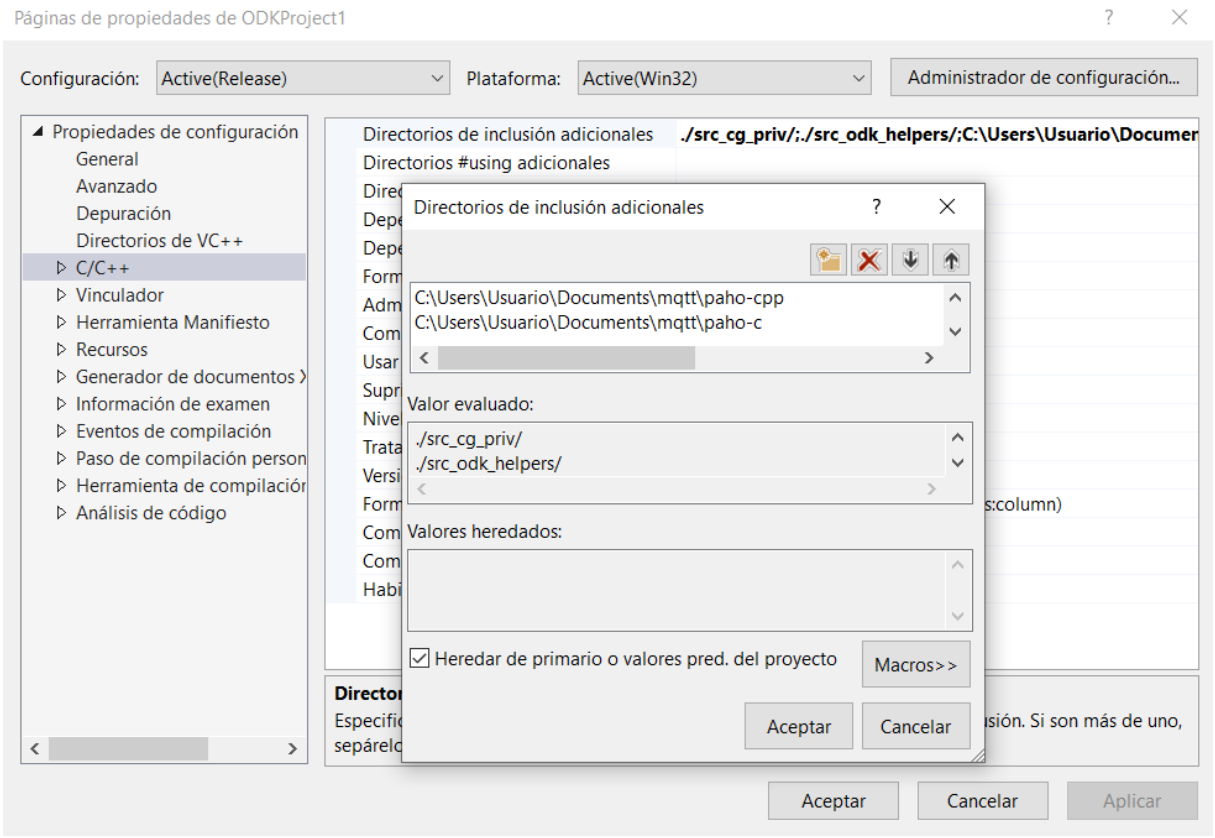


Ilustración 69. Añadir carpetas a Propiedades-C/C++

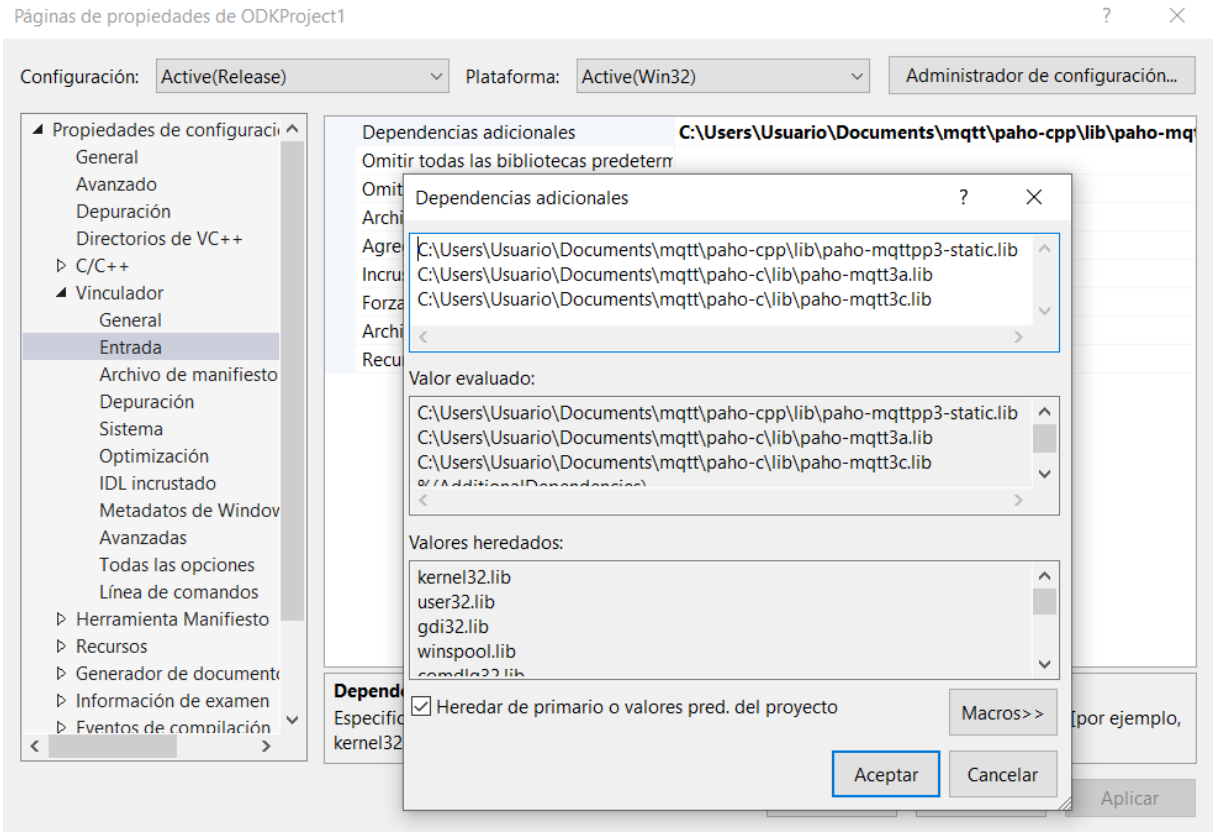


Ilustración 70. Añadir archivos .lib

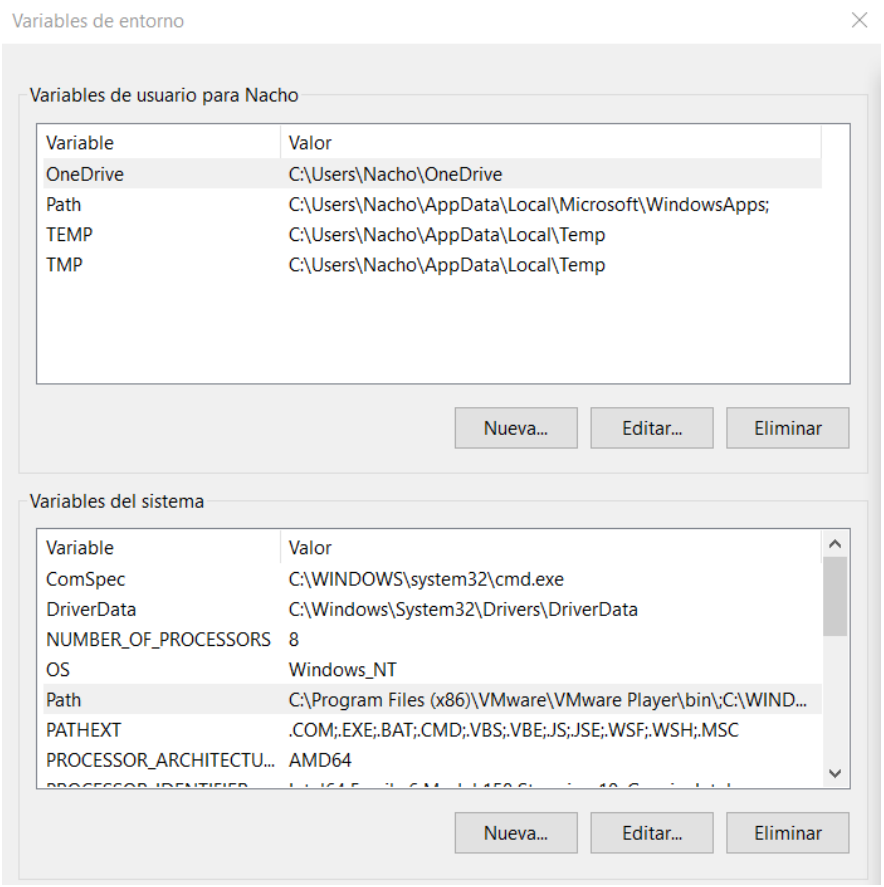


Ilustración 71. Path en variables de entorno

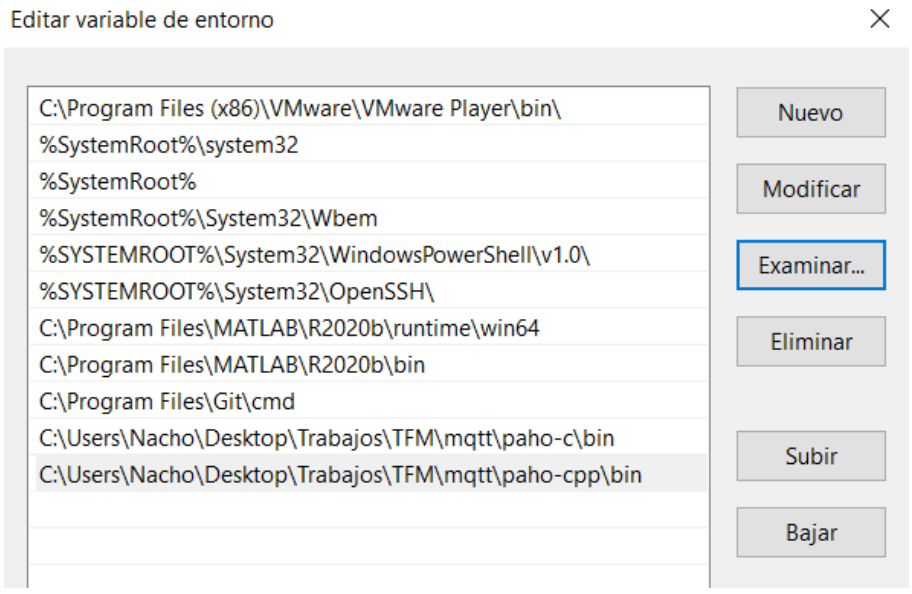


Ilustración 72. Carpetas bin en variables de entorno

Una vez hecho esto, se puede llevar a cabo la programación del cliente MQTT en el ODK. En primer lugar, se debe realizar la llamada a las librerias.

```
#include "stdafx.h"
#include "ODK_Functions.h"
#include "ODK_StringHelper.h"
#include "tchar.h"

#include <iostream>
#include <fstream>
#include <string>

// Include MQTT library
#include "include/MQTTClient.h"

MQTTClient client;
```

Ilustración 73. Llamada a librerías

La programación del cliente MQTT se realiza después de las funciones de ODK. Se deben desarrollar las funciones 'Publish', que mas tarde serán llamadas para publicar los datos. Hay una función por cada dato que se va a subir, ya que cada uno se va a asignar a un tema distinto.

```
void publishMachine(char* payload) {
    char TOPIC[256] = "DISA/robot/machine";

    MQTTClient_message pubmsg = MQTTClient_message_initializer;
    pubmsg.payload = payload;
    pubmsg.payloadlen = strlen(payload);
    pubmsg.qos = 0;
    pubmsg.retained = 0;
    MQTTClient_deliveryToken token;
    MQTTClient_publishMessage(client, TOPIC, &pubmsg, &token);
    MQTTClient_waitForCompletion(client, token, 1000L);
}
```

Ilustración 74. Función Publish de la referencia de máquina

Posteriormente, el ODK adquiere las variables de la estructura del PLC, y las asigna a su propia estructura. Tras esto, ya se puede efectuar la conexión con el bróker elegido, especificando su IP, y el nombre y contraseña si es aplicable.

```
/*Aquí creamos la conexión con el broker MQTT*/
char SERVER_ADDRESS[256] = "192.168.43.45:1883";
char CLIENT_ID[256] = "DISA_Client";                          //Nombre del cliente. Debe ser único para cada uno
char TOPIC[256] = "DISA/robot"; //channels/id del canal/publish/Write API Key


MQTTClient_create(&client, SERVER_ADDRESS, CLIENT_ID, MQTTCLIENT_PERSISTENCE_NONE, NULL);
MQTTClient_connectOptions conn_opts = MQTTClient_connectOptions_initializer;

conn_opts.username = "imanzanares";                           //Username
conn_opts.password = "";                        //MQTT API key

int rc;
if ((rc = MQTTClient_connect(client, &conn_opts)) != MQTTCLIENT_SUCCESS) {
    debugFile << "\n Failed to connect, return code %d\n";
    return ODK_SUCCESS;
}
debugFile << "\n Connection successful!";
```

Ilustración 75. Conexión con el bróker MQTT desde ODK

En este momento, ya se pueden realizar los envíos. Se debe llamar una a una a todas las funciones que se han declarado anteriormente.

```
std::string totalPayload = std::to_string(Id_Machine_Reference);
char* charPayload = (char*)totalPayload.c_str();
publishMachine(charPayload);
```

Ilustración 76. Envío de referencia de máquina

Mediante el MQTT Explorer, se puede comprobar que los datos enviados han sido recibidos por el bróker.

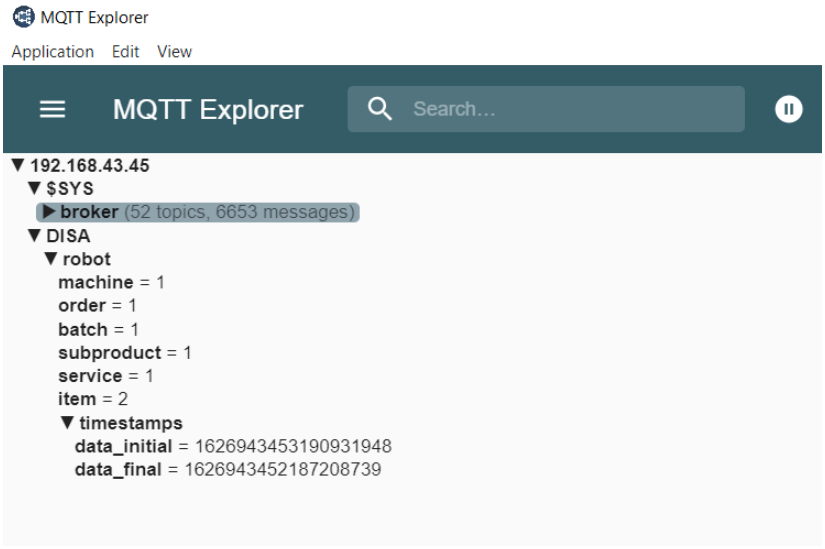


Ilustración 77. Contenido del bróker tras la subida de datos

2.4.3. Base de datos Influx DB

Como se ha comentado anteriormente, esta base de datos es empleada para guardar datos basados en una dimensión temporal. Por ello, es muy utilizada para subir datos de sensores. Es empleada en la industria para gestionar grandes cantidades de datos de fábricas enteras, pero también dispone de un plan gratuito para particulares, que será el empleado aquí. También otorga la posibilidad de construir paneles y gráficos para visualizar los datos. Dispone de una versión local, y otra basada en la nube.

Influx proporciona un agente llamado Telegraf, que recolecta métricas y las añade al contenedor asignado de la base de datos. Empleando el plugin MQTT, puede ejercer de cliente, y solicitar datos al bróker del tema deseado.

El plugin se añade mediante el archivo de configuración. El archivo se obtiene en la propia base de datos, tras crear un contenedor.

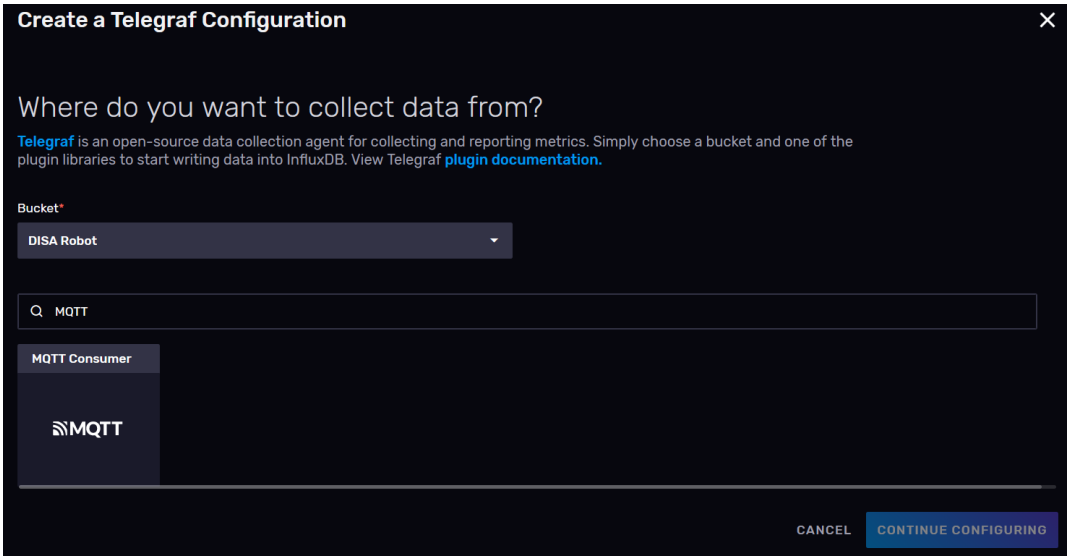


Ilustración 78. Configuración de Telegraf

Este archivo se debe modificar para añadir los datos necesarios para que el agente pueda desempeñar su función. Se configura tanto el bróker desde el que se reciben los datos (IP, temas), como la base de datos a la que se envían (servidor, API, usuario, contenedor). Se debe crear un token API con anterioridad, aunque resulta muy fácil.

```
[[inputs.mqtt_consumer]]
  ## MQTT broker URLs to be used. The format should be scheme://host:port,
  ## schema can be tcp, ssl, or ws.
  servers = ["mqtt://192.168.43.45:1883"]

  ## Topics that will be subscribed to.
  topics = [
    "DISA/robot/#",
  ]
```

Ilustración 79. Conexión al bróker desde Telegraf

```
[[outputs.influxdb_v2]]
 ## The URLs of the InfluxDB cluster nodes.
 ##
 ## Multiple URLs can be specified for a single cluster, only ONE of the
 ## urls will be written to each interval.
 ##   ex: urls = ["https://us-west-2-1.aws.cloud2.influxdata.com"]
 urls = ["https://eastus-1.azure.cloud2.influxdata.com"]

 ## Token for authentication.
 token = "z0gnWDfFOtvj3XdJBEdsex8HiOnHrl-Q7zNLBjNh2Qys-XXXXXXXXXXXXXXXXXXXXXXXXXXX-ttCGeA=="

 ## Organization is the name of the organization you wish to write to; must exist.
 organization = "XXXXXXX@gmail.com"

 ## Destination bucket to write into.
 bucket = "DISA Robot"
```

Ilustración 80. Conexión a la base de datos desde Telegraf

Una vez está realizada la configuración de Telegraf, se debe descargar el archivo, y llamarlo al abrir el programa. Telegraf se pondrá automáticamente a solicitar los datos del bróker según se generen, y a subirlos a la base de datos, si dispone de conexión.

```
pi@raspberrypi:~/Documents $ telegraf --config telegraf.conf
2022-03-01T17:23:40Z I! Starting Telegraf 1.19.1
2022-03-01T17:23:40Z I! Loaded inputs: mqtt_consumer
2022-03-01T17:23:40Z I! Loaded aggregators:
2022-03-01T17:23:40Z I! Loaded processors:
2022-03-01T17:23:40Z I! Loaded outputs: influxdb_v2
2022-03-01T17:23:40Z I! Tags enabled: host=raspberrypi
2022-03-01T17:23:40Z I! [agent] Config: Interval:10s, Quiet:false, Hostname:"raspberrypi", Flush Interval:10s
2022-03-01T17:23:40Z I! [inputs.mqtt_consumer] Connected [mqtt://localhost:1883]
```

Ilustración 81. Llamada a Telegraf desde la Raspberry Pi

Una vez los datos empiezan a llegar a Influx, se puede acceder a ellos desde cualquier dispositivo autorizado con acceso a internet. También se pueden hacer paneles para visualizar los datos de la forma más conveniente.

Ilustración 82. Panel con los datos del robot

3. PLANIFICACIÓN

En este apartado se describen brevemente las tareas realizadas, y el tiempo que ha llevado cada una de ellas.

- **Fase inicial:**

  Diciembre 2020 – Enero 2021

  En esta primera fase, el objetivo es familiarizarse con el robot y el entorno de trabajo, así como definir los objetivos a cumplir en el proyecto.

  - Se lleva a cabo la lectura de trabajos anteriores sobre los que basarse.
  - Se conoce el funcionamiento del robot
  - Se obtiene información sobre MQTT.

  Medios empleados: Ordenador, TIA Portal V16

- **Desarrollo en TIA Portal**

  Febrero 2021 – Abril 2021

  En esta fase se toma como base el programa existente para el control del robot, para realizar la programación completa.

  - Diseño de una estructura mas escalable
  - Desarrollo de servicios restantes

  Medios empleados: Ordenador, TIA Portal V16, robot

- **Pruebas con el robot**

  Mayo 2021

  A lo largo de este mes, se realizan las pruebas con el robot real.

  - Desarrollo de HMI
  - Pruebas con el robot

  Medios empleados: Ordenador, TIA Portal V16, robot

- **Desarrollo ODK**

Junio 2021

Se realiza la adaptación del programa del ODK para la toma de datos mediante MQTT. Mediante pruebas con el robot, se comprueba que el ODK funciona correctamente y transmite los datos a los archivos externos.

- Adaptación del ODK
- Comprobación de la transmisión de datos

Medios empleados: Ordenador, TIA Portal V16, robot, Microsoft Visual Studio

- **Conexión con base de datos**

Julio 2021

Se emplea la Raspberry Pi para efectuar la conexión via MQTT con la base de datos Influx DB.

- Programación del bróker en la Raspberry Pi
- Configuración de Influx y Telegraf
- Pruebas con el robot

Medios empleados: Ordenador, TIA Portal V16, robot, Microsoft Visual Studio, Raspberry Pi

- **Documentación**

Diciembre 2021 – Febrero 2022

Se realiza el documento en el que se recoge la información sobre el proyecto.

Medios empleados: Ordenador, TIA Portal V16, robot, Microsoft Visual Studio

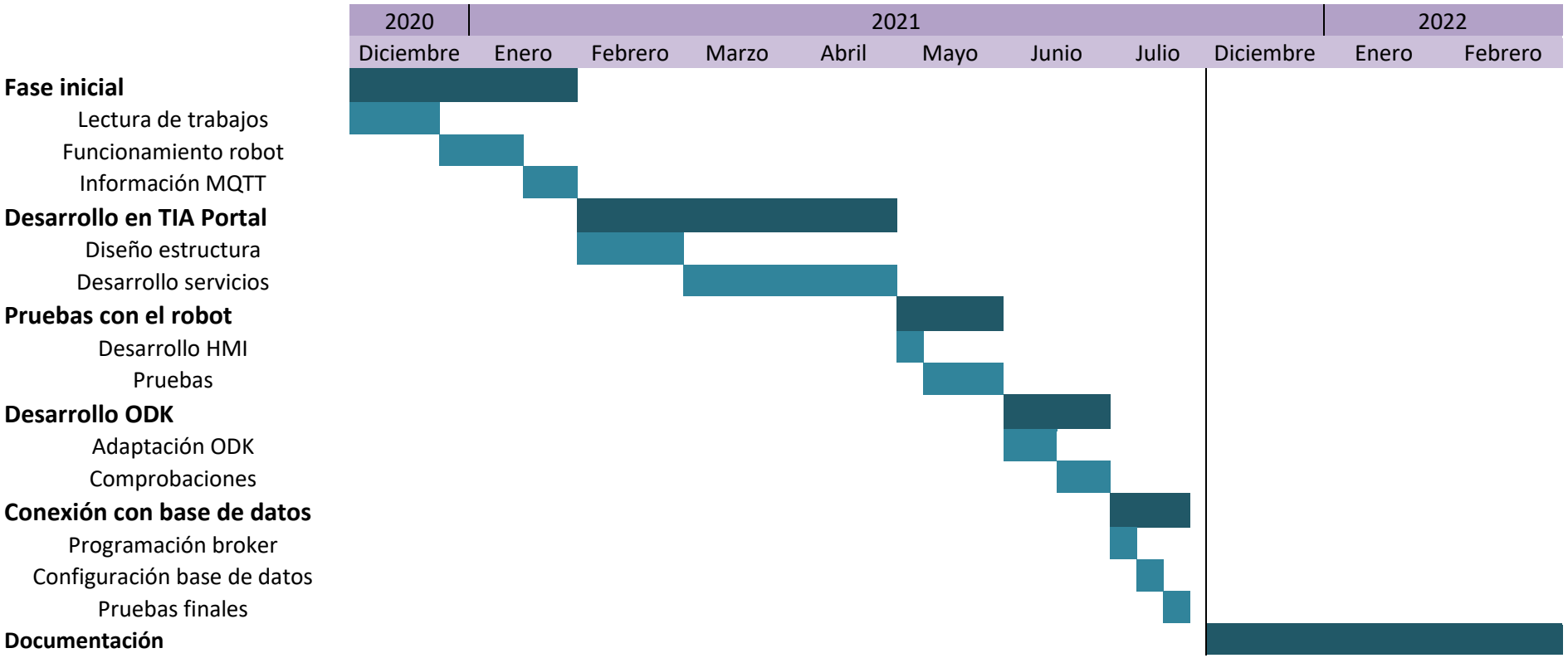


Tabla 7. Diagrama de Gantt

4. DESCARGO DE GASTOS

En este apartado se incluyen los aspectos económicos del proyecto. Primero se ofrecerá una descripción de cada una de las partidas, para después mostrar todos los gastos en una tabla conjunta. También se incluye un diagrama para mostrar los costes relativos de cada partida.

4.1.Horas internas

Este apartado incluye los gastos asociados al personal involucrado en el proyecto. En él han tomado parte un alumno, calculado como ingeniero junior con un coste horario de 20€; y un profesor, considerado ingeniero senior, con una retribución de 50€ por hora. Esta partida supone la gran mayoría de los gastos.

4.2.Amortizaciones

En este apartado se incluye el coste de activos reutilizables, de forma relativa a su tiempo de uso y vida útil. Estos activos no tienen por qué haber sido adquiridos expresamente para el proyecto, y pueden ser usados para otros proyectos posteriormente, hasta el final de su vida útil. Se incluyen aquí las licencias requeridas por los programas no gratuitos empleados, así como el ordenador utilizado para efectuar la programación, el PLC empleado en las pruebas, el robot, la mesa, y otros dispositivos.

4.3.Gastos

Se incluyen aquí los gastos empleados en activos no reutilizables para otros proyectos. En este caso se contabilizan las numerosas copias en papel de toda la documentación utilizada, así como material de oficina.

4.4.Costes indirectos

Estos son costes no imputables a un proyecto concreto, como el gasto eléctrico, limpieza o mantenimiento de equipos. Se calculan como un 10% del total de los demás gastos.

HORAS INTERNAS			
Concepto	Coste por hora	Horas totales	Total horas internas
Ingeniero Senior (Director)	50 €	100	5.000 €
Ingeniero Junior (Alumno)	20 €	600	12.000 €

AMORTIZACIONES				
Concepto	Coste unitario	Vida útil (meses)	Tiempo de uso (horas)	Total amortizaciones
Robot	18.000 €	120	150	31,25 €
Garra	1.200 €	120	150	2,08 €
Mesa de trabajo	1.500 €	120	150	2,60 €
TIA Portal V16	4.000 €	24	180	41,67 €
PLC ET200SP	1.800 €	60	150	6,25 €
Raspberry Pi	60 €	60	50	0,07 €
Ordenador	1.500 €	60	180	6,25 €
Cables de red	5 €	120	180	0,01 €

GASTOS		
Concepto		Total gastos
Material de oficina		5 €
Fotocopias		20 €
	SUBTOTAL:	17.115,17 €
Costes indirectos	10% del subtotal	1.711,52 €
	TOTAL:	18.826,69 €

Tabla 8. Descargo de gastos

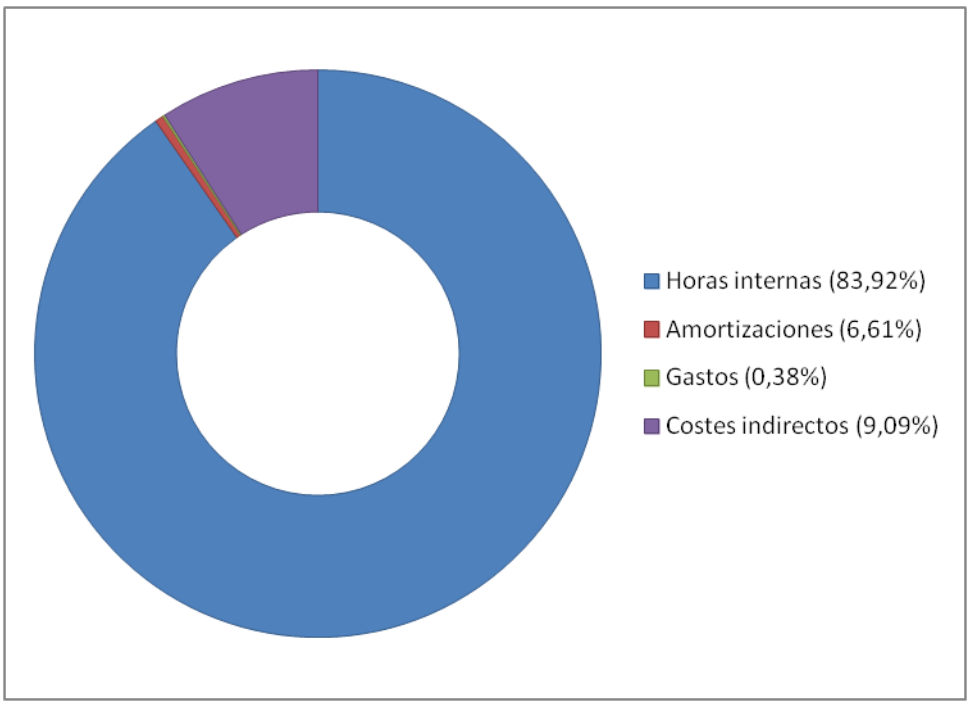


Ilustración 83. Porcentaje de los gastos según partidas

5. CONCLUSIONES

Se considera que en este trabajo de fin de máster se han cumplido satisfactoriamente los objetivos planteados. Se ha logrado realizar una programación funcional del robot en TIA Portal, que permite la operación en modo automático y se adapta sin problemas a los requerimientos del agente externo. También se ha conseguido realizar una conexión mediante MQTT con una base de datos en la nube, que recibe los datos que se envían desde el robot.

El uso de TIA Portal para la programación de robots se ha demostrado como un acierto. La posibilidad de realizar con un programa de uso común entre ingenieros una tarea que normalmente cuenta con una alta barrera de entrada puede permitir el uso de robots en lugares donde antes no se planteaba, y también puede permitir la incorporación de nuevos ingenieros en este mercado. Esto no tiene por qué hacerse a costa de ingenieros especializados en la programación de controladores, ya que esa habilidad seguirá siendo demandada en aquellas situaciones en las que se requiera una mayor complejidad, control, o conocimiento de la máquina.

En lo relativo a la transmisión de datos, se ha mostrado que MQTT es un sistema de transmisión de datos práctico, fiable, y sencillo de configurar. El uso de herramientas de uso no industrial y baja potencia como la Raspberry Pi, además de programas gratuitos o de código abierto, demuestra que la adquisición de datos en la nube no tiene por qué ser algo exclusivo de la industria, sino que también puede resultar útil para realizar pequeñas aplicaciones.

Aun así, algunas herramientas empleadas en este sistema de adquisición de datos ya se han utilizado en grandes aplicaciones industriales. Nervacero SA, utiliza MQTT para enviar datos de hasta 13000 señales de su planta de Trapagaran a una instancia de Influx DB alojada en una nube privada de la compañía matriz Celsa. Estas señales se emplean posteriormente para el desarrollo de aplicaciones de mantenimiento preventivo, y se han desarrollado paneles para mostrar la información mas relevante de una forma atractiva.

En el futuro, el trabajo realizado con el robot es fácilmente ampliable y aplicable a otros proyectos o investigaciones. Se puede ampliar el sistema de adquisición de datos en el ODK, para captar mas señales, ya no solo del proceso, sino del propio robot, para poder conocer en todo momento su estado.

También se pueden emplear esas señales captadas para realizar aplicaciones y análisis, de forma que se pueda optimizar el proceso, o prevenir fallos. Mediante el ODK se pueden diseñar aplicaciones complejas que interactúen con el programa del PLC, para poder

emplear, por ejemplo, sensores de visión artificial que reconozcan en qué fase de la construcción llega el montaje, y escoger de forma acorde las piezas que faltan.

El demostrador en sí forma parte de un proyecto mas grande, en el que varios agentes externos

En general, la combinación de datos, capacidad de procesamiento, y accesibilidad que proporcionan las nuevas herramientas permiten una mayor creatividad a la hora de diseñar aplicaciones, y una ilimitada cantidad de líneas de investigación futuras.

6. Bibliografía

[1] PricewaterhouseCoopers, «Industry 4.0: Building the digital enterprise,» 2016.

[2] OASIS, *MQTT Version 5.0,* 2019.

[3] KUKA, *KR 3 R540 Datasheet,* 2022.

[4] KUKA, *KUKA KR C4 Controllers.*

[5] SIEMENS, *Controlling a KUKA industrial robot using a SIMATIC S7-1500,* 2016.

[6] SIEMENS, *SIMATIC Robot Integrator for KUKA – Getting Started.*

[7] SIEMENS, SIMATIC ODK 1500S Examples V2.0, 2017.