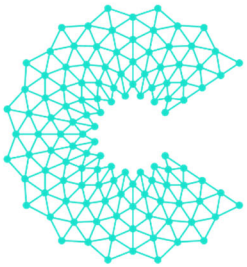


CENID CENTRO DE INTELIGENCIA DIGITAL PROVINCIA DE ALICANTE

5G e Inteligencia Artificial para la Transformación Digital de la Industria

Universidad Miguel Hernández de Elche (UMH)

María del Carmen Lucas Estañ, Javier Gozálvez Sempere, Miguel Sepulcre Ribes, Baldomero Coll Perales

19 de Noviembre de 2021

Contenido

Lista de acrónimos

3GPP	3rd Generation Partnership Project
5GAA	5G Automotive Association
5G-ACIA	5G Alliance for Connected Industries and Automation
5GC	5G Core
5QI	5G QoS Identifier
AAS	Asset Administration Shells
AF	Application Function
AGV	Automated Guided Vehicle
ANN	Artificial Neural Network
APN	Access Point Name
AR	Augmented Reality
ARP	Allocation and Retention Priority
CC	Component Carrier
CCSA	Asociación de Estándares de Comunicaciones de China
CMM	Coordinate-measuring machine
CNC	Central Network Controller
CPSoS	Cyber-Physical Systems of Systems
CQI	Channel Quality Indicator
CU	Centralized Unit
CUC	Centralized User Configuration
DL	Deep Learning
DL	Downlink
DN	Data Network
DNC	Deterministic Network Calculus
DS-TT	Device-side TSN translator
DU	Distributed Unit
EC	Edge Computing
eMBB	enhanced Mobile Broadband
ETSI	European Telecommunications Standards Institute
FC	Fog Computing
FDD	Frequency Division Duplex
FL	Federated Learning
FRER	Frame Replication and Ellimination for Reliability
GBR	Guaranteed Bit Rate
gNB	next generation Node B
gPTP	generalized Precision Time Protocol
HARQ	Hybrid Automatic Repeat Request
IA	Inteligencia Artificial
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IIoT	Industrial Internet of Things
IQN	in-advance QoS Notification
LADN	Soporte de Red de datos de área local
LDPC	Low Density Parity Check
MARL	aprendizaje por refuerzo de múltiples agentes
MC	Mist Computing
MCS	Modulation and Coding Scheme
MDAF	Management Data Analytics Function

MDBV	Maximum Data Burst Volume
MEC	Mobile Edge Computing
MIMO	Massive Multiple Input Multiple Output
ML	Machine Learning
MMRP	Multiple MAC Registration Protocol
mMTC	massive Machine Type Communications
MR	Mixed Reality
MSRP	Multiple Stream Registration Protocol
MVRP	Multiple VLAN Registration Protocol
NEF	Network Exposure Function
NF	Network Functions
NFV	Network Function Virtualization
NIST	National Institute of Standards and Technology
non-GBR	non-Guaranteed Bit Rate
NPN	non-public networks
NWDAF	Network Data Analytics Function
NW-TT	Network-side TSN translator
OEC	Open Edge Computing
OFC	Open Fog Consortium
OFDM	Orthogonal Frequency Division Multiplexing
OOP	Object Oriented Programming
OPC-UA	OLE for Process Control-Unified Architecture
OT	Operational Technology
PCF	Policy Control Function
PCP	Priority Code Point
PDB	Packet Delay Budget
PDCP	Packet Data Convergence Protocol
PDU	Protocol Data Unit
PE	Process Executor
PER	Packet Error Rate
PF	Prediction Function
PM	Process Modeling
PSFP	Per-Stream Filtering and Policing
PTP	Precision Time Protocol
QoS	Quality of Service
RAN	Radio Access Network
RB	Resource Block
RF	Radio Frequency
RIC	RAN Intelligent Controller
RNIB	Radio Network Information Base
SCS	Subcarrier Spacing
SDN	Software Defined Networking
SMF	Session Management Function
SPS	SemiPersistent Scheduling
SRP	Stream Reservation Protocol
SSC	Continuidad de Sesión y Servicio
TDD	Time Division Duplex
TRP	Transmission Reception Point
TSC	Time Sensitive Communications
TSCAI	TSC Assistance Information
TSN	Time-Sensitive Networking
UAV	Unmanned Aerial Vehicle

UE	User Equipment
UHD	Ultra High Definition
UL	Uplink
UNI	User/Network Configuration Interface
UPF	User Plane Function
URLLC	Ultra-Reliable Low Latency Communications
VLAN	Virtual Local Area Network
VR	Virtual Reality
XR	Extended Reality

Lista de figuras

Lista de tablas

Colaboraciones

Para el desarrollo de este proyecto, se ha contado con la colaboración de *Visual Components* (parte de KUKA) empresa desarrolladora del software que lleva el mismo nombre *Visual Components* de diseño y simulación de sistemas de producción y fabricación industrial. *Visual Components* nos ha concedido 4 licencias gratuitas para poder utilizar el software en el marco de este proyecto y poder así desarrollar los modelos digitales de plantas industriales. Además, se ha contado con su apoyo y asesoramiento en el manejo del software, el cual puede ser bastante complejo cuando se quiere implementar nuevas funcionalidades, como ha sido el caso en este proyecto, y hay que introducirse y modificar el código de descripción de los distintos componentes que se integran en el entorno industrial (maquinaria, robots, etc.).

También se ha contado con la colaboración de la empresa Asociación Innovalia. Innovalia ha proporcionado la descripción de una de las líneas de producción para poder desarrollar e implementar en este proyecto su modelo digital en el entorno *Visual Components*. Esta línea de producción se ha implementado en las instalaciones de Innovalia para llevar a cabo el estudio, investigación e implementación de distintas innovaciones tecnológicas relacionadas con la automatización de procesos industriales.

1. Introducción

La industria se encuentra en pleno proceso de transformación digital hacia modelos más productivos, competitivos y sostenibles basados en el concepto de Industria 4.0. La transformación digital de la industria requiere de redes de comunicación inteligentes, programables e inalámbricas que garanticen la capacidad de intercambiar datos de forma robusta y escalable entre nodos industriales ciber-físicos. Además, en el proceso de transformación de la industria hacia el paradigma de la Industria 4.0, las fábricas evolucionan hacia un nuevo modelo de fábrica modular con capacidad de reconfiguración que permita adaptar la producción a la demanda e individualizarla al cliente buscando maximizar la eficiencia de los recursos. Es por ello que las redes de comunicaciones industriales que darán soporte a la Industria 4.0 deberán ser capaces de adaptar su funcionamiento a las necesidades de conectividad y transferencia de datos de los sistemas de producción industriales.

Futuro de la industria europea

La industria de la Unión Europea se caracteriza por la fabricación de productos altamente innovadores y de calidad [1]. La industria europea es actualmente el mayor exportador mundial de productos fabricados, y sus exportaciones representan el 83% de las exportaciones europeas. Gracias a la industria, la Unión Europea alcanza superávits comerciales en el comercio de productos fabricados. Sin embargo, este superávit ya no permite compensar el déficit comercial que genera a la Unión Europea la compra de otros productos como materias primas, energías o servicios, entre otros.

Para mantener la competitividad industrial europea, la Unión Europea ha hecho una apuesta clara por la digitalización industrial, la sostenibilidad y el medioambiente, y el desarrollo de la economía circular [2]. Esta apuesta queda materializada en la hoja de ruta establecida con la nueva plataforma *Made in Europe* que define de forma clara como uno de sus objetivos claves el dúo transición digital-ecológica como aspecto claramente diferenciador de la industria europea. Esta plataforma define una visión para 2030 de la industria europea que pivota sobre el liderazgo tecnológico, la productividad, la retención y atracción del talento, y la sostenibilidad social, económica, y medioambiental. *Made in Europe* identifica claramente la necesidad de que la industria europea, a través de su capacidad innovadora y liderazgo tecnológico, se posicione como una industria totalmente digitalizada y capaz de liderar procesos industriales complejos y altamente diferenciadores, creativos, innovadores, flexibles y con crecientes niveles de personalización. De hecho, existen ya algunos ejemplos de esta capacidad de liderazgo europeo en la fabricación de productos complejos en el contexto actual de crisis de suministro de chips electrónicos con el caso de la multinacional holandesa ASML. Esta empresa está especializada en la fabricación de máquinas para la producción de circuitos integrados y es el mayor proveedor mundial de sistemas de fotolitografía para la industria de los semiconductores; esta tecnología es clave para la producción y miniaturización de chips. ASML proporciona a los fabricantes de chips la maquinaría especializada clave para la fabricación de componentes de alto valor añadido. Si bien Europa ha ido deslocalizando fábricas de semiconductores, retiene la competencia y conocimiento de los aspectos clave para la producción de los mismos. El objetivo de *Made in Europe* es, en línea con el ejemplo de ASML, convertir a la industria europea en el proveedor líder de soluciones tecnológicas, digitalización, eficiencia e implementación del modelo de producción basado en economía circular.

La plataforma *Made in Europe* define ciertos objetivos prioritarios para garantizar la competitividad de la industria europea. Por un lado, identifica como necesario desarrollar

procesos de fabricación *zero-defect* y de alta precisión que incluyan procesos predictivos de calidad y métodos de inspección no invasivos. Otro importante objetivo establecido por *Made in Europe* es desarrollar fábricas y procesos industriales en Europa fácilmente escalables, reconfigurables y flexibles para liderar procesos de fabricación con crecientes niveles de personalización. Además, identifica como objetivo prioritario el liderazgo europeo en el diseño y desarrollo de plataformas digitales y herramientas de ingeniería industrial que faciliten la creatividad e innovación, y mejoren la productividad de los procesos industriales. Los objetivos establecidos para la industria europea en la visión 2030 definida por *Made in Europe* son solo alcanzables con el desarrollo de las capacidades adecuadas en los trabajadores, mayor inversión en I+D, la digitalización de la industria, la flexibilización de los procesos de fabricación, y la integración rápida y efectiva de nuevas tecnologías que permitan desarrollar procesos industriales eficientes basados en la explotación de los datos de forma segura y privada [1].

La flexibilización de los procesos industriales, en parte gracias a una mayor integración de procesos cognitivos y una mayor explotación de los datos industriales, es uno de los principales objetivos que identifica también el grupo de expertos comisionados por el *Lighthouse Industry 4.0* para definir el futuro de la industria europea [3]. Dentro de lo que el *Lighthouse Industry 4.0* califica como *flagship concepts* (o buques insignia) para aprovechar el potencial esperado de la transformación digital de la industria, conviene también destacar los procesos de *virtual commissioning* basados en un mayor uso de procesos digitales de modelado y simulación. Estos procesos permiten diseñar, instalar y probar procesos industriales de forma digital y *off-line* utilizando modelos virtuales. De esta forma, se mejoran los procesos de toma de decisiones y la capacidad modificar o actualizar líneas de producción, máquinas o dispositivos industriales con el menor impacto posible sobre la operatividad de las plantas. A su vez, reducen el tiempo físico de puesta en marcha de nuevas líneas de producción. El *Lighthouse Industry 4.0* identifica también una clara tendencia hacia la descentralización de la inteligencia que debería estar integrada en todos los objetos y componentes industriales de forma que se permita un procesado y actuación local con crecientes niveles de seguridad y privacidad en la explotación de los datos industriales, y una mejor protección de la propiedad intelectual e industrial. La integración de mayores niveles de inteligencia y autonomía en los componentes industriales mejorará su capacidad de interaccionar y cooperar con el entorno industrial (ya sean otros dispositivos o procesos industriales o incluso los propios trabajadores) en procesos cada vez más dinámicos. La colaboración hombre-máquina y la interacción de los trabajadores con los robots colaborativos o cobots será otra tendencia en auge en parte gracias a la integración generalizada de inteligencia y capacidades cognitivas en los componentes industriales. El desarrollo de sistemas de producción cognitivos será un aspecto clave en la transición de la Industria 4.0 a la Industria 5.0. Mientras que la Industria 4.0 se ha caracterizado por crecientes niveles de automatización, conectividad de sistemas ciberfísicos (CPS) industriales, y flexibilización de los sistemas de producción, el concepto de Industria 5.0 busca ir un paso más allá y explotar la digitalización y la integración generalizada de inteligencia y capacidades de procesado de datos para diseñar procesos de producción capaces de analizar y entender el entorno industrial, y realizar las adaptaciones necesarias en sus procesos para una producción más eficiente.

La digitalización de la industria será clave para lograr fábricas y procesos industriales flexibles y ágiles que permitan una rápida respuesta a necesidades cambiantes, fluctuaciones en las características y suministro de componentes, y una cada vez más creciente demanda de productos con mayores niveles de personalización. Para ello, ya no es suficiente una mayor automatización de los procesos industriales, sino que la automatización debe ir acompañada de procesos de gestión y operación digitales de la cadena de valor completa y no sólo de las

fábricas. Para alcanzar la digitalización industrial es necesario integrar y explotar un conjunto de tecnologías habilitadoras entre las que se encuentra las comunicaciones inalámbricas y las redes 5G (incluidas las futuras redes 6G con capacidad táctil para ejecutar, por ejemplo, sistemas holográficos en entornos industriales), la inteligencia artificial (IA) y el *machine learning*, procesos de simulación y emulación que permitan desarrollar *Digital Twins*, el IIoT o internet de las cosas industrial, la realidad aumentada, y las tecnologías *cloud* centralizadas y localizadas (*edge computing*) entre otras.

En el ámbito de la IA, conviene destacar que los primeros usos se han centrado principalmente en monitorización y control en parte para el desarrollo de sistemas predictivos de mantenimiento. Sin embargo, el impacto de la IA en la transformación digital de la industria va más allá del mantenimiento, y mejorará la autonomía y flexibilidad de los sistemas de producción (incluidas las redes 5G que darán soporte a dichos sistemas), y facilitará los procesos de toma de decisión en contextos y entornos industriales complejos y altamente dinámicos. A pesar del potencial de la IA, conviene destacar que su integración efectiva en los procesos industriales es compleja, y en cierta medida lenta por diversas razonas, incluida la definición de interfaces y estándares abiertos e interoperables que faciliten su integración en procesos y dispositivos industriales, la extracción y disponibilidad de datos, y el propio funcionamiento de la IA que a veces dificulta su comprensión.

Un aspecto clave en el proceso de transformación digital de la industria será el desarrollo de los *digital-twins* o gemelos digitales. Un *digital-twin* es la representación digital de los componentes de un sistema de producción industrial. La disponibilidad de *digital-twins* permite entender mejor el funcionamiento de los componentes y predecir su comportamiento tanto de forma aislada como integrado en líneas de producción. La disponibilidad de *digital-twins* reducirá los tiempos de diseño y puesta en marcha de líneas de producción, así como los tiempos de interrupción al integrar nuevos componentes en las líneas existentes. Además, permitirá analizar y validar online de forma extensiva la adaptación o reconfiguración de ciertos procesos antes de su despliegue físico en las fábricas. De forma resumida, es posible decir que los *digital-twins* permitirán a las empresas y desarrolladores entender y predecir mejor como utilizar su infraestructura industrial de la forma más eficiente posible, y como ir integrando nuevas tecnologías con la mínima disrupción en los procesos industriales y la mejora de la productividad y eficiencia de los mismos.

Los grupos de expertos y organizaciones industriales europeas están de acuerdo en identificar la transformación digital de la industria (en binomio con la economía circular y la sostenibilidad) como un aspecto clave para garantizar la competitividad de la industria europea y su posicionamiento estratégico como proveedor clave de tecnologías disruptivas y nuevos procesos industriales centrados en la innovación, la creatividad, la flexibilidad y la personalización. Alcanzar estos objetivos requiere del desarrollo de un conjunto de tecnologías habilitadoras clave entre las que se encuentra el IIoT y la conectividad, sobre todo a través de las redes 5G que ofrecen altas tasas de transmisión (necesarias, por ejemplo, para servicios de realidad aumentada), y comunicaciones de alta fiabilidad y baja latencia (claves para los servicios críticos industriales).

5G para la transformación digital de la industria

5G va a ser una tecnología clave para la transformación digital de la industria, y de hecho Ericsson estima que para 2030 habrá más de 4700 millones de dispositivos inalámbricos en las fábricas. La tecnología 5G ofrece importantes novedades y mejoras con respecto a las

tecnologías celulares anteriores. Por un lado, 5G alcanzará mayores tasas de transmisión y reducirá la latencia de las comunicaciones gracias a una interfaz radio flexible y capaz de adaptar su funcionamiento a las necesidades de los servicios. La reducción de la latencia y el incremento de la fiabilidad de las comunicaciones mejora la capacidad de la tecnología 5G de dar soporte a entornos y servicios críticos como son los industriales. Otro aspecto clave de las redes 5G es la integración de las capacidades de comunicación o conectividad con la capacidad de cómputo no sólo en la nube sino también de forma local a través de la tecnología *edge computing*. La integración de esta tecnología en las redes 5G permite la realización de cálculos y procesado de forma local, es decir más cerca de donde se producen y consumen esos datos (como, por ejemplo, una fábrica), reduciendo así no sólo la latencia y la congestión de las redes, sino también mejorando la seguridad en los entornos conectados y la privacidad en la gestión de los datos (aspecto de hecho clave para la transformación digital de la industria).

La capacidad de garantizar altas tasas de transmisión, conectividad a un gran número de dispositivos, y comunicaciones de baja latencia y alta fiabilidad hará de 5G una tecnología clave para el desarrollo del internet de las cosas industrial (IIoT, *Industrial Internet of Things*) que permitirá que las fábricas del futuro cuenten con herramientas de optimización de la producción y sensores para monitorizar todo el proceso de fabricación. Conectar una planta industrial mejorará la capacidad de modificar y reconfigurar los procesos industriales, permitiendo así a los fabricantes producciones de menor escala y más personalizables. También mejorará la capacidad de monitorizar, controlar e incluso operar de forma autónoma o remota una gran cantidad de máquinas o dispositivos industriales. La 5G también facilitará la implementación de la realidad aumentada, una herramienta muy útil en el entorno industrial para la mejora de la productividad y la precisión, la detección de averías e incluso la formación de los trabajadores. La tecnología 5G mejorará la integración de robots y vehículos autónomos móviles (o AGV) en las plantas industriales mejorando así la productividad, la gestión logística, y la trazabilidad de componentes entre otros. La tecnología 5G será también un facilitador de los robots colaborativos o cobots que interactuarán y ayudarán a los operarios en la realización de ciertas tareas.

Las capacidades de la tecnología 5G es tal que son varios ya los ejemplos de fábricas altamente automatizadas (por ejemplo, fábricas de Bosch Rexroth en Alemania, o Ericsson en Estonia) que integran la tecnología 5G para mejorar y flexibilizar los procesos de monitorización, control y operación de las máquinas y otros dispositivos industriales. Parte de estos despliegues se realizan con lo que se denominan redes 5G privadas (o no públicas). Estas redes son definidas y desplegadas en entornos industriales de forma exclusiva, y requiere de políticas de gestión de espectro que permitan la ausencia de interferencia con las redes 5G públicas, de las cuales también es posible tener cobertura dentro de plantas industriales. De hecho, algunos países como Alemania ya han reservado parte del espectro 5G para ser utilizado de forma exclusiva en redes 5G privadas para entornos industriales.

La capacidad de conectar de forma ubicua y en tiempo real dispositivos industriales entre ellos y de recoger e intercambiar grandes cantidades de datos de los sistemas y dispositivos industriales con alta fiabilidad y baja latencia permitirá a su vez la interconexión continúa entre el mundo físico y digital, y en consecuencia la implementación y ejecución de *digital twins* industriales, que mejoren la flexibilidad y eficiencia de los procesos industriales. Sin embargo, es importante tener en cuenta la complejidad del dimensionado, configuración y operación de las redes 5G. Las altas capacidades y posibilidades de la tecnología 5G derivan en parte de la softwarización y virtualización de las redes, así como de la capacidad de adaptación de su

interfaz radio. Esta flexibilidad y capacidades entraña también un riesgo en la complejidad de la operación de las redes, y es importante entender bien las tecnologías y capacidades de las redes 5G más adecuadas y necesarias para dar soporte a servicios críticos industriales. La complejidad en la gestión y operación de las redes 5G, junto a la necesidad del uso de la tecnología 5G para el desarrollo de los *digital twins* industriales, ha puesto de manifiesto el potencial y necesidad de la creación de *digital twins* de las propias redes 5G, de forma que sea posible en un futuro, una operación más flexible y dinámica de las redes 5G, y también de los *digital twins* industriales que dependen de la tecnología 5G para la conexión entre los sistemas ciberfísicos industriales y la nube (centralizada o local) donde operarán los *digital twins* industriales. Los *digital twins* de las redes 5G emularán de forma precisa los diferentes sub-sistemas que componen las redes 5G, así como el entorno físico de propagación y el comportamiento de los dispositivos 5G [4]. Esto permitirá conocer y anticipar el rendimiento y funcionamiento de las redes 5G en base a su contexto, de forma que se pueda gestionar y configurar las redes 5G de forma proactiva y no reactiva para que proporcionen los niveles y calidad de conectividad necesarios en cada momento. Para ello, la asociación industrial 5G-ACIA (*5G Alliance for Connected Industries and Automation*) propone la definición de un modelo AAS (*Asset Administration Shells*) de las redes 5G que describa sus características y capacidades, y permita así su interacción o integración con los modelos AAS de las plantas industriales [5]. Es importante destacar que la definición de los modelos AAS es un paso relevante en la creación de los *digital twins*, y existen iniciativas para su estandarización en *International Electrotechnical Commission* (IEC) así como el establecimiento de nuevas organizaciones entorno a esta temática como el *Industrial Digital Twin Association*. El AAS es un concepto clave en la transformación digital de la industria pues permite describir digitalmente las capacidades y características de un *asset* (un *asset* puede ser cualquier dispositivo, máquina, software, documento o elemento tanto físico como digital que tenga una función en el entorno industrial), y asegurar así la interoperabilidad dentro de una planta y con el exterior en base a estándares digitales abiertos. La creación de un AAS es un primer paso hacia la creación de *digital twins* interoperables pues permite que *assets* que componen un entorno industrial puedan interactuar e intercambiar datos para la gestión de procesos industriales. Es relevante destacar, en línea con las capacidades y necesidades de autogestión y optimización de las futuras evoluciones de las redes 5G, que existen propuestas de extensión del concepto AAS para redes 5G hacia AAS activos que, además de proporcionar una caracterización digital de las capacidades de la red 5G, permita su autogestión y optimización para adaptarse a las necesidades de los procesos industriales en una mayor simbiosis digital entre la tecnología 5G y los procesos industriales.

La tecnología 5G va a ser una tecnología habilitadora clave para la transformación digital de la industria. Si bien su potencial y capacidades son claras, la tecnología 5G no está exenta de retos para una efectiva y eficiente integración en los sistemas ciberfísicos industriales. Además del posible desarrollo de *digital twins* de redes 5G que interactúan (o sean integrados) con *digital twins* industriales, la tecnología 5G deberá evolucionar y ser adaptada en un futuro a medio plazo (en lo que se denomina redes *Beyond* 5G o 6G) para dar un mejor soporte a las necesidades de digitalización de la industria. Algunos aspectos clave en la evolución de la tecnología 5G incluyen [6]: 1) nuevas funcionalidades para mejor soporte de las redes privadas y un despliegue y operación flexible de las redes 5G, 2) mecanismos avanzados de auto-gestión (o *self-management*) de las redes 5G utilizando, por ejemplo, las capacidades de la inteligencia artificial y el *machine learning*, 3) el diseño de redes más energéticamente eficientes y con menor exposición a campos electromagnéticos, 4) provisión de comunicaciones con latencias inferiores al milisegundo, 5) soluciones para el posicionamiento de alta precisión en entornos de interiores

y desarrollo de capacidades de sensado gracias a las comunicaciones 5G y 6G, 5) e interfaces abiertas y mejor capacidad de interacción/integración con los *digital twins* industriales.

1.1. Objetivos del proyecto

Este proyecto busca contribuir a la integración de la tecnología 5G en la industria como catalizador tecnológico clave para su transformación digital hacia modelos de Industria 4.0. Dicha integración explotará en el futuro la inteligencia artificial para una gestión autónoma, flexible y eficiente de las redes 5G de forma que sean capaces de anticipar las necesidades de conectividad de los sistemas de producción y adaptar así su funcionamiento en base a dichas necesidades.

En este contexto, el trabajo desarrollado en este proyecto se ha desarrollado entorno a estos dos objetivos clave:

1. Estudio técnico sobre necesidades de conectividad de la Industria 4.0, capacidades tecnológicas de 5G para satisfacer estas necesidades, y retos tecnológicos a abordar para una efectiva y eficiente integración de la tecnología 5G en entornos industriales.
2. Generación de modelos digitales de plantas industriales. El objetivo de crear estos modelos digitales es la obtención de bancos de pruebas digitales capaces de emular de forma realista (y configurable) el comportamiento de plantas industriales para poder así conocer y caracterizar la necesidad de conectividad y transferencia de datos dentro de las plantas. La disponibilidad de estos modelos digitales nos permite generar valiosos *datasets* sobre la generación y transferencia de datos industriales en diferentes contextos y configuraciones de las plantas industriales.

Ambos objetivos se han concluido con éxito en el marco del proyecto: el grado de consecución de ambos objetivos es del 100%.

En el desarrollo de este proyecto, el equipo investigador ha contado con la colaboración de los investigadores Rafael Molina Masegosa y Miguel Cuenca Piqueras, y de los estudiantes Francisco José Soler, Pablo Giner y Álvaro Aguilar.

1.2. Principales resultados

Este informe de justificación científica presenta de manera detallada en los siguientes apartados los resultados alcanzados bajo los dos objetivos del proyecto. Esta memoria se ha dividido en dos partes, en las que se describen el trabajo y resultados alcanzados para cada uno de los objetivos identificados, respectivamente.

La parte I se centra en el primer objetivo. En primer lugar, se ha desarrollado un estudio detallado sobre las necesidades de conectividad de la Industria 4.0, detallando los requisitos de comunicación identificados por distintas asociaciones y cuerpos de estandarización para distintos casos de uso y aplicaciones de la Industria 4.0. A continuación, se presentan las capacidades tecnológicas de 5G para satisfacer las necesidades y requisitos demandados por las aplicaciones de la Industria 4.0. Por último, se analizan los retos tecnológicos identificados que todavía deben ser abordados para lograr la eficiente integración de la tecnología 5G en entornos industriales.

La parte II de esta memoria presenta todo el trabajo realizado en la generación e implementación de los modelos digitales de plantas industriales y a generación de los *datasets* con los datos sobre la generación y transferencia de datos industriales en los escenarios industriales modelados. Es importante destacar que la generación de estos *datasets* supone un avance clave en la investigación sobre soluciones de gestión autónoma y flexible de las redes 5G utilizando IA por las siguientes razones principalmente. Por un lado, las técnicas y algoritmos de IA requieren de *datasets* que reflejen el comportamiento real del sistema para su entrenamiento y posterior validación. Por otro lado, la disponibilidad de *datasets* similares en la comunidad es nula dado el carácter confidencial del diseño y configuración de sistemas de producción en plantas industriales reales, lo cual es un factor limitante para el desarrollo de soluciones basadas en inteligencia artificial.

Los resultados obtenidos en este proyecto (en el objetivo 1 y 2) son claves para la comunidad científica, pero también para nuestro grupo de investigación pues nos proporcionan las herramientas necesarias para seguir trabajando en el diseño de técnicas de gestión autónomas y eficientes de las redes 5G *and Beyond* basadas en IA, abordando los desafíos identificados en el marco del primer objetivo. Estas futuras investigaciones de técnicas de gestión de la red 5G *and Beyond* autónomas y eficientes basadas en IA se hacen viables gracias a la disponibilidad de los *datasets* obtenidos en el marco del objetivo 2.

El equipo de investigación de este proyecto realiza activamente un trabajo de difusión y divulgación continuo de sus actividades de investigación, además de la diseminación científica mediante la publicación de sus investigaciones en publicaciones científicas. En cuanto a las actividades de divulgación que realiza el equipo, es importante destacar su participación continua en las Jornadas de Puertas Abiertas organizadas en la UMH para promover las titulaciones de la Escuela Politécnica Superior de Elche (https://epse.umh.es/jornada-de-puertas-abiertas/) y en la Feria de Ciencias y Tecnologías de Elche (FECITELX, http://mireumh.edu.umh.es/es/disfruta-de-actividades-en-nuestro-campus/fecitelx-feria-de-ciencias-y-tecnologias-de-elche/). Desafortunadamente estas Jornadas y Ferias están suspendidas desde 2020 por la situación de pandemia actual. El compromiso del grupo de investigación del presente proyecto es continuar difundiendo y publicitando sus actividades en estos eventos cuando vuelvan a celebrarse.

Además, el equipo difunde sus actividades científicas a través de la página web del laboratorio Uwicore (laboratorio de investigación al que pertenecen los miembros del equipo investigador de este proyecto): http://www.uwicore.umh.es/. En esta página es posible encontrar una descripción de los proyectos de investigación realizados, así como una sección específica con todas las publicaciones científicas realizadas por el grupo (http://www.uwicore.umh.es/publications-all.html). Además, el grupo de investigación también hace uso del Depósito Digital de la UMH (REDIUMH) (http://dspace.umh.es/) para todas aquellas publicaciones cuya política de editorial lo permite.

Parte I
Estudio técnico sobre necesidades de conectividad de la Industria 4.0, capacidades tecnológicas de 5G para satisfacer estas necesidades, y retos tecnológicos a abordar para una efectiva y eficiente integración de la tecnología 5G en entornos industriales

2. Introducción

La tecnología 5G ofrece las capacidades de conectividad, fiabilidad y adaptabilidad necesarias para actuar como catalizador clave de la transformación digital de la industria. De hecho, son varios los países e industrias que apuestan de forma decidida por el despliegue de redes privadas 5G en entornos industriales en sintonía con una mayor automatización industrial y la explotación de datos industriales mediante el uso intensivo de la inteligencia artificial y computación de alto rendimiento. Esta convergencia tecnológica es clave para proporcionar los niveles de servicio deterministas necesarios para la transformación digital de la Industria 4.0, y permitir el desarrollo de servicios digitales innovadores como servicios basados en realidad aumentada, el uso de gemelos digitales (*digital-twins*) para optimización de procesos industriales, y la fabricación autónoma sin defectos, servicios que demandan muy altas tasas de transmisión, bajas latencias y altos niveles de fiabilidad en la conectividad industrial.

Las capacidades tecnológicas que ofrece 5G derivan, en buena medida, de la softwarización y virtualización de las redes 5G, una nueva interfaz radio (5G NR), y la introducción de tecnologías como *network slicing* y *edge computing*. Todas estas tecnologías ofrecen niveles de flexibilidad y reconfiguración sin precedentes, pero a coste de altos niveles de complejidad en la gestión y operación de las redes que pueden reducir su eficacia, e incluso comprometer su eficiencia energéticas. Abordar estos retos requiere de sistemas autónomos de gestión de las redes capaces de integrar procesos de razonamiento gracias a la convergencia de las redes, la computación y los datos, así como la explotación de la inteligencia artificial.

Es importante también destacar que el uso de la tecnología 5G en entornos industriales requiere de su integración efectiva y eficaz con las redes cableadas industriales que proporcionan altos grados de fiabilidad y robustez, y que actualmente están evolucionando hacia la adopción de la tecnología TSN (*Time Sensitive Networking*) basada en Ethernet. TSN proporciona niveles de servicio deterministas inalcanzables hasta la fecha, pero carecen de la flexibilidad de la tecnología 5G. En este contexto, es necesaria la integración de las redes 5G y TSN industriales para garantizar la conectividad necesaria para la transformación digital de la Industria.

En esta primera parte se presenta el trabajo realizado en el marco del primer objetivo de este proyecto. En particular, se analizan las necesidades de conectividad de la Industria 4.0. A continuación se estudian las capacidades tecnológicas de 5G para satisfacer estas necesidades. Y finalmente se analizan los retos tecnológicos que todavía tienen que ser abordados para lograr una eficiente integración de la tecnología 5G en entornos industriales.

3. Requisitos de la Industria 4.0

La transformación digital de la industria resulta en la evolución de entornos de fabricación rígidos en redes de fabricación cada vez más flexibles que interconectan ecosistemas de fabricación modulares que aportan a la industria una alta flexibilidad y capacidad de adaptación eficiente a cambios en la demanda. Además, el aumento de la digitalización y la automatización permite realizar una producción más eficiente, más rápida, más segura, más estable, más económica, más fiable y con una mayor capacidad de gestión eficiente. Sin embargo, al mismo tiempo la complejidad de los productos y sistemas de producción escala exponencialmente, al igual que las capacidades de aprendizaje automático en el entorno industrial. Este aumento de la flexibilidad y de la complejidad del entorno industrial imponen nuevos desafíos en la gestión de las redes de comunicaciones que deben darle soporte.

En este apartado, se presentan las necesidades y requisitos generales de la Industria 4.0. Posteriormente, se analizan los requisitos de servicio en términos del rendimiento de las comunicaciones de los casos de uso más representativos de la Industria 4.0 y de los nuevos servicios emergentes que permitirán lograr esos niveles de autonomía, flexibilidad y eficiencia requeridos por la industria.

3.1. Requisitos generales de la Industria 4.0

En este apartado se describen los principales requisitos generales de la Industria 4.0 impuestos por las actuales tendencias en la transformación digital que sufre la industria:

- Resiliencia y seguridad. Aunque existen diversas definiciones de resiliencia en el entorno industrial, todas ellas comparten un mismo núcleo: “la capacidad de una entidad para anticipar, resistir, absorber, responder, adaptarse y recuperarse de una perturbación” [7]. En el entorno industrial, la resiliencia y la seguridad son dos de las características más demandadas y todavía más actualmente considerando que muchos sistemas industriales están experimentando grandes cambios [8]. Para garantizar estas características de resiliencia y seguridad de las fábricas y procesos de fabricación es necesario que las redes de comunicaciones que los soportan también cumplan estas características ([9]-[11]).
- Transformación verde y eficiencia. Las empresas industriales han aumentado su preocupación sobre el cuidado del medio ambiente. Esto se debe a que las propias empresas son más conscientes de su impacto sobre medio ambiente y que prevén que los clientes no comprarán algo que haya sido dañino para el mismo. Además, existe una clara evidencia de que, al implementar tecnologías y procedimientos operativos más precisos y cuidadosos, todos los indicadores de rendimiento (indicadores técnicos, de rentabilidad y también ecológicos) se optimizan. En este contexto, la Industria 4.0 busca alcanzar una producción sostenible, en la que se debe optimizar la economía de los materiales. Los materiales deben utilizarse eficientemente pues un uso inadecuado de materiales puede resultar no eficiente económicamente y peligroso para el medio ambiente. Una ingeniería y operaciones basadas en el conocimiento, la precisión y bien gestionada son los medios más efectivos para lograr esta eficiencia. Las mayores ineficiencias son debidas a menudo a fallos tecnológicos o paradas imprevistas en la producción, debido al uso de algoritmos y controladores mal ajustados en sistemas de

producción de mala calidad y piezas, componentes y sistemas mal fabricados [3]. En este contexto, las industrias trabajan en alcanzar una producción *zero-defect*, y para alcanzarla, el sistema de comunicaciones que lo soporta juega un papel muy importante.

- Inteligencia distribuida y maquinaria inteligente. En la transformación digital de la industria la inteligencia migra cada vez más cerca de la maquinaria y los procesos, y se está volviendo más local en lugar de global o centralizada [3]. Aunque la capacidad de comunicación global es cada vez mayor, la industria es reacia a difundir sus datos y información crítica a localizaciones remotas fuera de sus instalaciones. Las soluciones actuales de arquitecturas de objetos inteligentes distribuidos en el Edge resultan más atractivas para la Industria 4.0, siendo estas cada vez más viables gracias a las mayores capacidades de computación integradas en las máquinas, robots y procesos. Estos sistemas de computación e inteligencia distribuida en el Edge requieren por tanto del diseño de redes de comunicaciones eficientes y flexibles capaces de soportar el intercambio de datos e inteligencia entre nodos.
- Lugares de trabajo digitales táctiles. La industria se está transformando en un entorno en el que se busca una colaboración, coordinación y cooperación hombre-máquina-máquina más segura y resiliente para impulsar la competitividad en la fabricación. Es necesario ir más allá de los contextos individuales y explotar las redes y servicios inteligentes avanzados para optimizar el funcionamiento en tiempo real de sistemas de sistemas ciber-físicos (*Cyber-Physical Systems of Systems*, CPSoS). La necesidad de centrarse más en el usuario, de aumentar la agilidad de la producción y la transparencia de la cadena de suministro, y mejorar la fabricación y operación remota exige aumentar la ubicuidad, inteligencia y conectividad de la fabricación. En la Industria 4.0, la inteligencia debe poder gestionarse desde cualquier lugar, en cualquier momento y por cualquier persona. En [3] se resalta que las infraestructuras de redes públicas y redes no públicas (NPN) de 5G están respaldando el primer uso de tecnologías inalámbricas y móviles en la automatización de la fábrica y la gestión de la información para respaldar una operación más flexible, modular y remota de los sistemas ciber-físicos (*Cyber-Physical Systems*, CPS). Sin embargo, todavía se requiere la integración de redes inalámbricas avanzadas (5G and *Beyond*) con la inteligencia y la computación para lograr esos entornos de fabricación digital de próxima generación [3].

3.2. Casos de uso y requisitos de servicio

El entorno industrial es diverso y heterogéneo y se caracteriza por una gran cantidad de casos de uso y aplicaciones diferentes con requisitos a veces muy diversos. Por ejemplo, grandes áreas industriales como la fabricación discreta pueden diferir sustancialmente de otras, como puede ser de la industria de procesos. Esto es válido no solo con respecto a los requisitos de calidad del servicio, sino también a los escenarios de implementación. Varias asociaciones y cuerpos de estandarización han analizado los casos de uso típicos de la industria y sus requisitos asociados, como, por ejemplo, el 3GPP (*3rd Generation Partnership Project*), el ETSI (*European Telecommunications Standards Institute*), el NIST (*National Institute of Standards and Technology*) de Estados Unidos, or 5G-ACIA (5G *Alliance for Connected Industries and Automation*) entre otros. En este contexto, este apartado presenta los principales casos de uso de la Industria 4.0 según estas distintas fuentes y sus requisitos de rendimiento.

Antes de comenzar con la presentación de los casos de uso industriales seleccionados y sus requisitos de servicio, se presentan la definición de una serie de parámetros que se utilizarán para definir los requisitos de comunicaciones de los servicios industriales. Estos son:

- Latencia *end-to-end*: latencia máxima de transmisión de un paquete requerida desde la capa de aplicación del transmisor a la capa de aplicación del receptor ([12]).
- Jitter: desviación máxima de la latencia *end-to-end* con respecto a un valor de referencia ([13]).
- *Communication service availability*: indica si el sistema de comunicaciones funciona según los requisitos establecidos ([13]).
- Fiabilidad: probabilidad de fallo de la transmisión debido a que la información se pierde, se recibe de manera posterior a la latencia máxima permitida o se recibe con error. La pérdida se percibe en la interfaz de la capa de aplicación ([12]).
- *Service bit rate*: tasa de datos mínima que el sistema de comunicación garantiza en cualquier momento ([13]).
- Rango: distancia mínima requerida entre dos nodos inalámbricos que forman un único enlace ([12]).
- Tamaño del mensaje o *payload*: cantidad de datos transmitidos en un paquete ([12]).
- *Transfer Interval*: indica el tiempo transcurrido entre dos mensajes consecutivos entregados por la aplicación al sistema de comunicación ([13]).
- Frecuencia de actualización o *update rate*: número de transmisiones que ocurren en un segundo en la capa de aplicación de un dispositivo ([12]).

El 3GPP identifica en distintos estándares ([14][15]) e informes técnicos ([13][16]) los numerosos casos de uso y aplicaciones de la Industria 4.0. Los casos de uso están relacionados con diferentes áreas de aplicación, como automatización de procesos y fábricas, almacenamiento, monitorización y mantenimiento logístico, entre otros. Estos casos de uso tienen diferentes requisitos de comunicación definidos en términos de tasas de datos, latencia, fiabilidad o disponibilidad entre otros. Los casos de uso se clasifican en [14] y [13] en tres clases de tráfico diferentes: determinista periódico, determinista aperiódico y no determinista (periódico o aperiódico). El tráfico periódico determinista se genera periódicamente y debe recibirse dentro de un plazo determinado. El tráfico determinista se caracteriza por una latencia máxima que depende del caso de uso en concreto. El tráfico periódico determinista es la clase de tráfico industrial más común ([14] y [13]). Por ejemplo, se relaciona con casos de uso como control de

movimiento, comunicaciones control a control, comunicación de robots móviles y automatización de procesos, entre otros. El tráfico aperiódico determinista se refiere al tráfico que no se genera periódicamente, pero que cuando se generan paquetes también deben recibirse con una latencia máxima determinada. El tráfico aperiódico determinista es característico de los casos de uso en los que las transmisiones se activan cuando ocurren eventos específicos. Estos eventos se pueden activar, por ejemplo, cuando: 1) una temperatura o un nivel de presión excede o cae por debajo de umbrales predefinidos (eventos de proceso), 2) los sensores detectan fallas o errores de dispositivos o módulos, 3) o en base a información que indique el trabajo de mantenimiento necesario para prevenir fallos (eventos de mantenimiento). El tráfico aperiódico determinista es, por ejemplo, característico de casos de uso relacionados con paneles de control con funciones de seguridad y automatización de procesos. Finalmente, el tráfico no determinista es el tráfico (periódico o aperiódico) que no tiene un plazo de tiempo para recibirlo. El tráfico no determinista es característico de aplicaciones que, por ejemplo, requieren actualizaciones de software o descargas de archivos, entre otros. Estas aplicaciones se pueden encontrar en casos de uso como control de movimiento, paneles de seguridad o automatización de procesos entre otros. A continuación, se presentan las principales características y requisitos de comunicación de algunos casos de uso representativos de la Industria 4.0. Los casos de uso que se presentan a continuación son una selección de los casos de uso definidos en [14] seleccionados con el objetivo de mostrar casos de uso representativos de los distintos tipos de tráfico y basado en la selección de casos de uso realizada en [17] por el 5G-ACIA y en [18]. Estos son:

- Comunicaciones locales control a control o *local control-to-control communication*.
- Control de movimiento.
- Robots móviles y vehículos guiados automatizados (AGV, *Automated Guided Vehicles*).
- Paneles de control móviles con funciones de seguridad (paneles de seguridad).
- Control de bucle cerrado para la automatización de procesos.
- Monitorización remota para la automatización de procesos.

Tal y como se identifica en [17], el uso del término 'local' en la descripción de algunos casos de uso significa que el caso de uso está restringido a un área local y que utiliza una red 5G no pública (ver apartado 4.5). El uso del término 'remoto' en la descripción de los casos de uso denota que el caso de uso requiere una combinación de (múltiples) redes 5G públicas y/o no públicas que permita el acceso remoto.

- <u>Comunicaciones locales control a control o *local control-to-control communication*</u>

  Este caso de uso considera la comunicación entre diferentes controladores industriales. Este tipo de comunicaciones puede ser necesaria para conectar, por ejemplo, máquinas individuales que se utilizan en una línea de montaje para realizar una tarea común. También puede ser necesario sincronizar e intercambiar datos en tiempo real entre diferentes controladores en máquinas grandes (por ejemplo, máquinas de impresión de periódicos). En este tipo de aplicaciones, la comunicación no suele estar basada en IP, ya que las distancias son relativamente cortas. La comunicación control a control generalmente genera tráfico periódico con requisitos de latencia deterministas y casi en tiempo real.

  Un par de ejemplos de este tipo de comunicaciones son los siguientes [17]. El primero de ellos hace referencia a la comunicación control a control en una máquina de embalaje. Las máquinas de embalaje de última generación incorporan sistemas *shuttle* que transportan materiales dentro de la misma máquina o entre varias máquinas. Estos sistemas *shuttle*

incorporan controladores locales que pueden comunicar su posición y otros datos de control a través de la red 5G. dado que estos controladores interactúan con otros dispositivos como robots o máquinas que requieren la posición exacta del sistema *shuttle* en tiempo real, la transmisión fallida o con retardo de los datos correspondientes podría provocar el cese del funcionamiento de la máquina y, por tanto, el tiempo de inactividad.

Otro ejemplo son aplicaciones industriales en los que varios vehículos (dos o cuatro) sin conductor controlados a distancia realizan el transporte o manipulación (de forma colaborativa) de componentes de gran tamaño; este es un caso de uso muy frecuente en industrias de aviación y transporte marítimo. Los vehículos que realizan que realizan el transporte o manipulación de grandes objetivos de forma colaborativa se mueven a lo largo de sus pistas dedicadas y cuyas posiciones relativas se controlan localmente para minimizar la fuerza aplicada al componente que se está transportando.

Los requisitos establecidos para este tipo de aplicaciones por el 3GPP en [13] se presentan en la siguiente tabla:

Tabla 1. Requisitos de servicio para Comunicaciones control a control (en aplicaciones de control de movimiento) [13].

Communication service availability	**Latencia *end-to-end***	**Jitter**	***Service bit rate***	**Tamaño del mensaje [byte]**	***Transfer interval***	**# de UEs**	**Comentarios**
99,9999% to 99,999999%	< *transfer interval*	-	-	1 k	≤ 4 ms	5 a 10	Interacción cíclica, hasta 100 UEs en el futuro.

- Control de movimiento

Un sistema de control de movimiento es responsable de controlar las partes móviles y/o giratorias de las máquinas (por ejemplo, máquinas de impresión, máquinas herramientas o máquinas de embalaje). El control de movimiento genera tráfico periódico con requisitos de latencia deterministas y estrictos. Este caso de uso también puede requerir datos sin requisitos de tiempo real relacionados, por ejemplo, con actualizaciones de software/firmware o información de mantenimiento.

En la Figura 1 se muestra un esquema de un sistema de control de movimiento. Un controlador de movimiento envía periódicamente los puntos de ajuste deseados a uno o varios actuadores (por ejemplo, un actuador lineal o un accionamiento) que, a continuación, realizan una acción correspondiente en uno o varios procesos (en este caso normalmente un movimiento o rotación de un determinado componente). Al mismo tiempo, los sensores determinan el estado actual de los procesos, por ejemplo, la posición actual y/o la rotación de uno o varios componentes, y enviar los valores reales al controlador de movimiento. Esto se hace de una manera estrictamente cíclica y determinista, de modo que durante un ciclo de aplicación el controlador de movimiento envía puntos de ajuste o *set points* actualizados a todos los actuadores, y todos los sensores envían sus valores reales de vuelta al controlador de movimiento. Hoy en día, las tecnologías de Ethernet industrial típicamente se utilizan para los sistemas de control de movimiento.

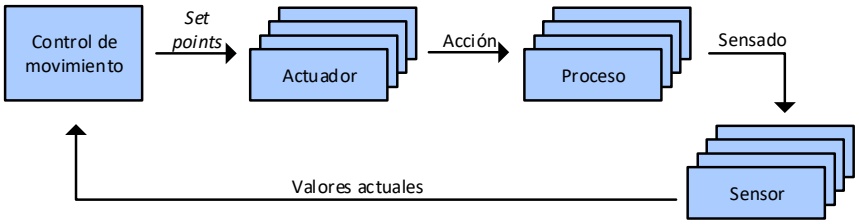


Figura 1. Esquema de un sistema de control de movimiento [14].

En [14] se definen los requisitos de servicio para tres sub-casos de uso dentro del caso de uso de control de movimiento. Estos se presentan en la Tabla 2.

Tabla 2. Requisitos de servicio para casos de uso de robots móviles y AGVs [14].

Sub-caso de uso	*Communication service availability*	Latencia *end-to-end*	Jitter	*Service bit rate*	Tamaño del mensaje [byte]	*Transfer interval*	# de UEs
1	99.999% a 99.99999%	< *transfer interval*	-	–	50	500 µs	≤ 20
2	99.9999% a 99.999999%	< *transfer interval*		–	40	1 ms	≤ 50
3	99.9999% a 99.999999%	< *transfer interval*		–	20	2 ms	≤ 100

- Robots móviles y AGV

Un robot móvil o un AGV es una máquina o robot programable capaz de realizar una gran variedad de tareas, generalmente siguiendo trayectorias programadas. El uso de estos robots móviles y AGV aporta gran flexibilidad al entorno industrial. Los robots móviles normalmente se controlan o supervisan desde un sistema de control de guiado. Las comunicaciones inalámbricas son esenciales para dar soporte a estos robots móviles o AGVs ya que el uso de cables no es viable.

5G-ACIA presenta un par de ejemplos de este tipo de caso en [17]. El primero de ellos es un caso de uso muy común en el que robots móviles son utilizados para realizar el transporte de materiales (recoger y almacenar material) en almacenes y plantas de producción. Los robots recogen los artículos o material de distintas posiciones de almacenamiento y los transportan a un destino predeterminado, como por ejemplo una estación de embalaje o un contenedor. En las plantas de producción, los robots móviles se utilizan para transportar productos u objetos de una fase del proceso de producción a la siguiente. Otro ejemplo es el uso de AGVs en plantas químicas. Estos AGVs son generalmente controlados de forma remota por un operador en una sala de control. El operador observa imágenes capturadas por cámaras montadas en el AGV, las cuales se transmiten de forma inalámbrica. El operador detendrá el AGV de forma inmediata si reconoce un obstáculo en el camino del AGV o cualquier otro mal funcionamiento. Cualquier fallo o retraso en la transmisión de las señales de la cámara puede provocar accidentes graves o, como mínimo, interrupciones innecesarias del funcionamiento del AGV.

Los robots móviles pueden desplazarse en áreas interiores, exteriores o tanto en interiores como exteriores. Normalmente, este tipo de aplicaciones requiere una comunicación determinista y periódica entre el robot y el sistema de control, aunque puede generar otros

tipos de tráfico en función de la aplicación específica admitida por el robot móvil y de las condiciones en las que se implementa.

En [14], el 3GPP establece los requisitos para el caso de uso de robots móviles y AGVs que se muestran en la Tabla 3. En [14], se distingue entre 4 casos de uso diferentes:

- Sub-caso de uso 1: Comunicación periódica para el apoyo del control de movimiento robótico cooperativo preciso (intervalo de transferencia: 1 ms), control de la máquina (intervalo de transferencia: 1 ms a 10 ms), conducción cooperativa (10 ms a 50 ms).
- Sub-caso de uso 2: Comunicación periódica para control remoto operado por vídeo.
- Sub-caso de uso 3: Comunicación periódica para el funcionamiento estándar del robot móvil y la gestión del tráfico.
- Sub-caso de uso 4: Transmisión de datos en tiempo real (datos de vídeo) desde un robot móvil al sistema de control de guiado.

Tabla 3. Requisitos de servicio para casos de uso de robots móviles y AGVs [14].

Sub-caso de uso	*Communication service availability*	Latencia *end-to-end*	Jitter	*Service bit rate*	Tamaño del mensaje [byte]	*Transfer interval*	# de UEs
1	> 99.9999%	< *transfer interval*	-	–	40 a 250	1 ms a 50 ms	≤ 100
2	> 99.9999%	< *transfer interval*		–	15 k a 250 k	10 ms a 100 ms	≤ 100
3	> 99.9999%	< *transfer interval*		–	40 a 250	40 ms a 500 ms	≤ 100
4	> 99.9999%	10 ms		> 10 Mbit/s	–	-	≤ 100

- Paneles de control móviles con funciones de seguridad (paneles de seguridad)

Los paneles de control son dispositivos cruciales para la interacción entre las personas y la maquinaria de producción, así como para la interacción con dispositivos móviles. Estos paneles se utilizan principalmente para configurar, monitorizar, controlar y mantener máquinas, robots, grúas o líneas de producción. Además, los paneles de control (de seguridad) generalmente están equipados con un botón de parada de emergencia y un dispositivo de habilitación, que un operador puede pulsar en caso de un evento de seguridad para evitar daños a personas o a la maquinaria. Cuando se presiona el botón de parada de emergencia, el equipo controlado llega inmediatamente a una posición estacionaria segura. Del mismo modo, si una máquina, robot, etc. opera en el modo especial de “dispositivo habilitado", el operador mantiene manualmente el interruptor del dispositivo de habilitación en una posición estacionaria especial. Si el operador presiona demasiado este interruptor o lo suelta, el equipo controlado llega inmediatamente a una posición estacionaria segura. De esta manera, se puede garantizar que las manos del operador estén en el panel (y no debajo de una prensa de moldeo, por ejemplo), y que el operador, por ejemplo, no sufra ninguna descarga eléctrica o similar. Un ejemplo común para este "modo de dispositivo habilitador" es la instalación, prueba o mantenimiento de una máquina, durante el cual se desactivan otros mecanismos de seguridad (como una valla de seguridad).

Debido a la criticidad de estas funciones de seguridad, los paneles de control de seguridad actualmente utilizan principalmente conexiones cableadas con el equipo que controlan. Además, suele haber muchos paneles de este tipo para las muchas máquinas y unidades de producción que normalmente se pueden encontrar en una fábrica. Con un enlace inalámbrico de baja latencia ultra fiable, sería posible conectar dichos paneles de control móviles con funciones de seguridad de forma inalámbrica, lo cual conllevaría una mayor usabilidad y permitiría la reutilización flexible y fácil de paneles para controlar diferentes máquinas.

Los tiempos de ciclo de la aplicación de control dependen del proceso, maquinaria o equipo cuya seguridad debe garantizarse. Para un robot de movimiento rápido, por ejemplo, las latencias de un extremo a otro son más bajas que para los actuadores lineales de movimiento lento. Este caso de uso requiere la transmisión de datos no críticos (tráfico no determinista) para la configuración, monitoreo y mantenimiento de las máquinas. También requiere la transmisión de datos de seguridad altamente críticos e impredecibles con estrictos requisitos de latencia (tráfico aperiódico determinista) al presionar el botón de parada de emergencia.

En [14] se definen los requisitos de servicio para 3 sub-casos de uso dentro de este caso de uso más general.

- Sub-caso de uso 1: Comunicación periódica bidireccional para control remoto. Ejemplos de unidades controladas: robots de montaje o fresadoras.
- Sub-caso de uso 2: Transmisión de datos periódica en paralelo al control remoto (caso de uso uno).
- Sub-caso de uso 3: Comunicación periódica bidireccional para control remoto. Ejemplos de unidades controladas: grúas móviles, bombas móviles, grúas de pórtico fijo, etc.

Los requisitos se presentan en la Tabla 4.

Tabla 4. Requisitos de servicio para casos de uso de paneles de control móviles [14].

Sub-caso de uso	*Communication service availability*	Latencia *end-to-end*	Jitter	*Service bit rate*	Tamaño del mensaje [byte]	*Transfer interval*	# de UEs
1	99.9999% a 99.999999%	< *transfer interval*	-	-	40 a 250	4 ms a 8 ms	A definir
2	99.9999% a 99.999999%	< *transfer interval*	-	> 5 Mbit/s	-	< 30 ms	A definir
3	99.9999% a 99.999999%	< *transfer interval*	-	-	40 a 250	< 12 ms	A definir

- Control de bucle cerrado para la automatización de procesos

En este caso de uso, varios sensores instalados en una planta realizan mediciones continuas que son transmitidas a un controlador que actúa sobre ciertos actuadores formando un bucle de control cerrado. La latencia y el determinismo en este caso de uso son cruciales. El control de bucle cerrado produce tráfico periódico y aperiódico con estrictos requisitos de latencia (es decir, tráfico determinista). El tráfico es aperiódico si, por ejemplo, el sensor

solo transmite datos cuando se supera un determinado umbral. Será periódico si los datos detectados deben transmitirse periódicamente para mantener activo el proceso industrial.

Un ejemplo de este caso de uso es el siguiente [17]. La creciente necesidad de eficiencia en la producción y calidad del producto exige un control preciso de los procesos de fabricación. Por este motivo, las bombas, válvulas, calentadores, enfriadores, agitadores y otros componentes de un reactor químico son monitorizados continuamente por sensores que miden caudales, temperatura y presión para mantener las condiciones en el reactor dentro de umbrales ajustados. Disponibilidad, fiabilidad, seguridad y confidencialidad de las comunicaciones son cruciales para este caso de uso.

Los requisitos establecidos por el 3GPP en [14] para este caso de uso se presentan en la Tabla 5. En concreto, estos requisitos son establecidos para un caso de uso concreto en el que varios sensores instalados en una planta realizan medidas continuas que son enviadas a un controlador. El controlador configura una serie de actuadores tras procesar los datos recibidos.

Tabla 5. Requisitos de servicio en bucles cerrados de control para la automatización de procesos [14].

Communication service availability	**Latencia *end-to-end***	**Jitter**	***Service bit rate***	**Tamaño del mensaje [byte]**	***Transfer interval***	**# de UEs**
99.9999% to 99.999999%	< *transfer interval*	-	-	20	≥ 10 ms	10 a 20

- Monitorización remota para la automatización de procesos

Para el monitoreo de procesos y activos en el área de automatización de procesos, se instala una gran cantidad de sensores inalámbricos industriales en la planta para brindar información sobre las condiciones ambientales y de proceso, el estado de los activos y el inventario de material. Los datos se transportan a pantallas para observación y/o bases de datos para el registro y análisis de los datos. Ejemplos de sensores utilizados son sensores de temperatura, presión o de flujo para monitorización de procesos, sensores de vibración para monitorización del estado de, por ejemplo, motores o cámaras térmicas para detectar fugas. Estos casos de uso pueden implicar el uso de redes públicas, aunque ciertos subprocesos pueden requerir sus propias redes dedicadas no públicas.

Un ejemplo de este caso de uso lo encontramos en la industria del petróleo y el gas [17]. En estas industrias, los equipos se distribuyen en un área geográfica significativa, como por ejemple en un campo petrolero. Los datos sobre la eficiencia y el estado operativo de los pozos, activos y dispositivos son capturados por los sensores correspondientes para monitoreo remoto. La disponibilidad, la fiabilidad y la seguridad de las comunicaciones son aspectos importantes para toda la cadena de comunicaciones.

Este caso de uso se caracteriza por generar tráfico periódico determinista. Dentro de este caso de uso general, el 3GPP identifica en [14] tres sub-casos de uso más concretos con los siguientes requisitos:

- Sub-caso de uso 1: Los sensores generan mediciones periódicas de un valor continuo (por ejemplo, sensores de temperatura, presión, caudal). El tráfico es de origen predominantemente móvil.
- Sub-caso de uso 2: Los sensores generan medidas de forma de onda (por ejemplo, sensores de vibración). Aunque la medición de la forma de onda es continua, se espera que este tipo de sensores almacenen y transmitan los datos periódicamente (por ejemplo, cada segundo) para ahorrar batería al permitir la transmisión discontinua. El tráfico es de origen predominantemente móvil.
- Sub-caso de uso 3: Cámaras para la monitorización de activos (por ejemplo, para la detección de fugas). Aunque la grabación de video es continua, se espera que este tipo de sensores almacenen y transmitan los datos periódicamente (por ejemplo, cada segundo) para ahorrar batería al permitir la transmisión discontinua. El tráfico es de origen predominantemente móvil.

Tabla 6. Requisitos de servicio para casos de uso de monitorización remota para la automatización de procesos [14].

Sub-caso de uso	*Communication service availability*	Latencia *end-to-end*	Jitter	*Service bit rate*	Tamaño del mensaje [byte]	*Transfer interval*	UE density [UE / m²]
1	99.99%	< 100 ms	-	-	20	100 ms to 60 s	Hasta 1
2	99.99%	< 100 ms	-	-	25 k	≤ 1 s	Hasta 0.05
3	99.99%	< 100 ms	-	-	250 k	≤ 1 s	Hasta 0.05

En el Anexo A.1 se incluye una tabla con los requisitos de servicio para todos los casos de uso de la Industria 4.0 identificados por el 3GPP en [14]. En este anexo, los casos de uso se han agrupado según el tipo de tráfico que demandan: determinista periódico, determinista aperiódico y no determinista.

Como se ha presentado en este apartado y se puede también observar en el Anexo A.1, el 3GPP considera casos de uso con requisitos de fiabilidad (*communication service availability*) de hasta $1\text{-}10^{-6}$ y $1\text{-}10^{-8}$ [14]. Estos requisitos tan estrictos también han sido identificados por otras fuentes. Por ejemplo, el NIST establece en [12] los requisitos para distintas aplicaciones de la Industria 4.0. El NIST sigue la misma clasificación de aplicaciones realizada por el ISA en [19] según el cual las aplicaciones se clasifican en 5 clases:

- Clase 0: *safety*. Incluye las aplicaciones más críticas, como por ejemplo, los sistemas integrados de seguridad. Estos sistemas requieren alta fiabilidad y baja latencia, generalmente con tamaños de carga útil muy pequeños. Las aplicaciones típicas de la Clase 0 son aquellas que se implementan para evitar daños al equipo o al personal.
- Clase 1: Control regulatorio de bucle cerrado. El control regulatorio consta de múltiples lazos de control de una sola entrada y una sola salida, diseñados para regular variables locales como el flujo, la velocidad, etc. control de acoplamiento del robot y control preciso del brazo basado en la posición.
- Clase 2: Control de supervisión de bucle cerrado. Una aplicación típica de Clase 2 es un supervisor basado en PLC. En esta aplicación, los PLC envían comandos a los actuadores para realizar las tareas. Específicamente, en la fabricación discreta, las tareas se realizan

secuencialmente; por lo tanto, el efecto de una mayor latencia de un extremo a otro en el enlace de comunicaciones entre un supervisor y un actuador influirá en la velocidad de producción.

- Clase 3: Control regulatorio de bucle abierto. En la Clase 3, el control se realiza manualmente (mediante operador) en lugar de mediante retroalimentación automatizada. Las aplicaciones típicas de esta clase incluyen el levantamiento de objetos pesados mediante un sistema de pórtico controlado a distancia o la operación manual de equipos rotativos.
- Clase 4: Monitorización. Las aplicaciones típicas de Clase 4 consisten en dispositivos de detección y monitorización. Muchos de estos dispositivos no requieren baja latencia ni alta fiabilidad.

En la Tabla 7 se muestran los requisitos identificados por el NIST en [12] para las distintas clases de aplicaciones de la Industria 4.0. Para cada parámetro, la tabla muestra valores típicos y los valores más estrictos o máximos. Como se puede observar en la Tabla 7, se identifica un requisito de fiabilidad de $1\text{-}10^{-8}$ para las aplicaciones de Clase 0 para aplicaciones de *safety* con requisitos más estrictos y latencias entre 0.5 y 4 ms. En [12] también se establece requisitos de latencia entre 0.25 y 4 ms para aplicaciones de control reglamentario de bucle cerrado y bucle abierto, y entre 4 y 20 ms para aplicaciones de control de supervisión de bucle cerrado; NIST considera un requisito de fiabilidad de $1\text{-}10^{-7}$ para todas estas aplicaciones.

Tabla 7. Requisitos de servicio para la Industria 4.0 identificados por el NIST [12] (se considera una celda de trabajo de 10 m x 10 m).

Parámetro		Clase 0: *Safety*	Clase 1: Control regulatorio de bucle cerrado	Clase 2: Control de supervisión de bucle cerrado	Clase 3: Control regulatorio de bucle abierto	Clase 4: Monitorización
Latencia *end-to-end* (ms)	Típico	4	4	20	4	50
	Estricto	0.5	0.25	4	0.5	4
Fiabilidad	Típico	$1\text{-}10^{-7}$	$1\text{-}10^{-7}$	$1\text{-}10^{-7}$	$1\text{-}10^{-7}$	$1\text{-}10^{-6}$
	Estricto	$1\text{-}10^{-8}$	$1\text{-}10^{-7}$	$1\text{-}10^{-7}$	$1\text{-}10^{-7}$	$1\text{-}10^{-7}$
Número de enlaces	Típico	8	10	10	1	100
	Máximo	24	30	30	4	300
Rango (m)	Típico	10	10	10	10	10
	Máximo	30	30	30	30	30
Tamaño de datos (*payload*) (bytes)	Típico	6	8	8	8	12
	Máximo	24	64	64	64	33K
Update rate (Hz)	Típico	125	125	25	125	10
	Máximo	1000	2000	125	1000	125

Por otro lado, ETSI define algunos de los requisitos de latencia y fiabilidad más estrictos para los casos de uso de la Industria 4.0 en [20]. Por ejemplo, ETSI estima que la fabricación discreta requiere una latencia entre 1 y 12 ms, y aplicaciones de control de movimiento una latencia entre 0.25 y 1 ms; según [20], ambos casos de uso necesitan una fiabilidad de $1\text{-}10^{-9}$. En el Anexo

A.2 se recoge la tabla resumen con los requisitos identificados para los diferentes casos de uso para la Industria 4.0 identificados por ETSI.

Por último, también es interesante resaltar los requisitos de servicios de la Industria 4.0 identificados en otros trabajos, como por ejemplo, en [21], donde se identifican requisitos muy estrictos para aplicaciones de automatización con control de máquinas en tiempo real. Por ejemplo, [21] estima que aplicaciones como máquinas de impresión, máquinas de envasado o celdas de fabricación requieren latencias máximas entre 0.25 y 5 ms y niveles de fiabilidad superiores a $1\text{-}10^{-9}$.

3.3. Servicios emergentes

Los avances en comunicaciones, así como en otras tecnologías como el sensado y detección, imagen, visualización e Inteligencia Artificial o IA empujan el desarrollo de servicios emergentes en el entorno industrial, siendo estos altamente exigentes en términos de comunicaciones. Estos servicios no solo demandan bajas latencias y alta fiabilidad en la comunicación, sino que a la vez requieren elevados anchos de banda. Ejemplos de este tipo de servicios es la Realidad Extendida (*Extended Reality* o XR) y el gemelo digital o *digital twin*, los cuales se presentan a continuación.

XR es un nuevo término que combina VR (*Virtual Reality*), AR (*Augmented Reality*) y MR (*Mixed Reality*) [22]. La XR ha atraído una gran atención y ha abierto nuevos horizontes en el campo de la fabricación y se prevé tendrá un papel muy relevante en las futuras fábricas e instalaciones de producción inteligentes. El desarrollo técnico para realizar XR aún está en progreso, y constantemente aparecen nuevas tecnologías innovadoras [23]. En las fábricas del futuro, las personas seguirán desempeñando un papel importante y sustancial. Debido al alto grado de flexibilidad, versatilidad y adaptabilidad previstas de las fábricas del futuro, los trabajadores de las plantas industriales deberán recibir apoyo para adaptarse rápidamente a nuevas tareas y actividades y garantizar el correcto funcionamiento de nuevas operaciones de manera eficiente y ergonómica. En este contexto, la XR desempeñará un papel fundamental, por ejemplo, para las siguientes aplicaciones:

- seguimiento de procesos y flujos de producción.
- instrucciones paso a paso para tareas específicas, por ejemplo, en lugares de trabajo de montaje manual.
- soporte de un experto remoto, por ejemplo, para tareas de mantenimiento o servicio.

En este sentido, especialmente los dispositivos de XR montados en la cabeza con pantalla transparente son muy atractivos ya que permiten un grado máximo de ergonomía, flexibilidad y movilidad y dejan las manos de los trabajadores libres para otras tareas. Un enfoque muy prometedor en aplicaciones de XR es descargar tareas de procesamiento complejas a la red y reducir la carga de procesado y funcionalidades del dispositivo montado en la cabeza de XR. Esto tiene el beneficio adicional de que la aplicación de XR puede tener fácil acceso a información de contexto diferente (por ejemplo, información sobre el entorno, la maquinaria de producción, el estado actual del enlace, etc.) si se ejecuta en la red.

Este tipo de servicios supone un nuevo desafío para los sistemas de comunicaciones móviles actuales. Por ejemplo, la tecnología AR actual requiere 55.3 Mbps para vídeo con calidad 8K (con un millón de puntos), lo que puede proporcionar suficiente experiencia de usuario en una pantalla móvil [23]. Sin embargo, para proporcionar AR realmente inmersivo, la densidad debe mejorarse en gran medida y requerirá un rendimiento de 0.44 Gbps (con 16 millones de puntos).

Además, la transmisión de medios XR puede tener demandas similares a las de los videos de calidad 16K UHD (*Ultra High Definition*). Por ejemplo, 16K VR requiere un rendimiento de 0.9 Gbps (con una relación de compresión de 1/400).

Otro servicio emergente en la industria es el gemelo digital o *digital twin* cuya integración en la industria resulta de gran interés para la optimización y mantenimiento de los activos físicos, sistemas y procesos de fabricación. El gemelo digital o *digital twin* es una representación digital y dinámica de un activo industrial que permite a las empresas entender y predecir mejor el comportamiento y rendimiento de los procesos [3]. Un artículo de 2018 ([24]) sobre el uso de *digital twins* en la industria predecía: "En 10 años, se generalizarán las réplicas digitales de equipos industriales en industrias que van desde alimentos y bebidas hasta manufactura y atención médica". El gemelo digital o *digital twin* representa la interconexión y convergencia entre un sistema físico y su representación digital creada como una entidad propia [25]. Ambas entidades, el objeto físico y el objeto digital están completamente integrados y pueden intercambiar información en ambas direcciones. Por lo tanto, el objeto digital podría actuar como una instancia de control del objeto físico y viceversa. IoT se utiliza para recopilar automáticamente los datos de fabricación en tiempo real, mientras que el *digital twin* junto con el análisis de *big data* y el uso de la Inteligencia Artificial podría usar los datos para predecir, estimar y analizar los cambios dinámicos dentro del objeto físico. La información o solución optimizada obtenida en el *digital twin*, se retroalimenta al objeto físico que se adaptaría en consecuencia. Esto convierte a la tecnología del *digital twin* en el centro de la transformación y actualización de la fabricación global, ya que tiene el potencial de optimizar todo el proceso de fabricación.

El *digital twin* requiere una gran potencia de computación para disponer de la capacidad de simulación necesaria para que el modelo virtual pueda interactuar en tiempo real con el modelo físico, y la conectividad que se dispone actualmente con la nube es por tanto un recurso sin precedentes. Sin embargo, el *digital twin* supone también un desafío para los sistemas de comunicaciones inalámbricos actuales. Por ejemplo, para duplicar un área de 1 m x 1 m se necesita un Tera-pixel, lo cual requiere un throughput de 0.8 Tbps asumiendo una sincronización periódica de 100 ms y un ratio de compresión 1/300.

4. Tecnologías 5G habilitadoras para la transformación digital de la industria

En este apartado se presenta las principales características y tecnologías implementadas en las redes 5G actuales que serán clave en el proceso de transformación digital de la industria. Para ello, se ha realizado un repaso del estado de desarrollo de los estándares, distintos documentos de trabajo publicados por las principales asociaciones y cuerpos de estandarización relacionados con el desarrollo de las redes 5G (como, por ejemplo, 5G-ACIA, 5GSMA, ETSI, etc.), como de la literatura científica.

4.1. Arquitectura de los sistemas 5G

En primer lugar, se presentan algunos conceptos básicos de la arquitectura del sistema 5G necesarias para entender otras tecnologías presentadas más adelante.

La Figura 2 muestra la arquitectura de un sistema 5G [26]. La red de acceso radio o RAN (*Radio Access Network*) incluye el gNB (*next generation* Node B) y el equipo de usuario UE (*User Equipment*). El gNB y la red troncal o *core network* (CN) de 5G son parte del dominio de la red. En el *core network* de 5G, los elementos hardware utilizados en los sistemas celulares tradicionales se reemplazan por funciones de red NF (*Network Functions*) que se pueden virtualizar e implementar en software. Esto facilita la separación de las NFs del plano de control y de usuario y permite implementaciones de arquitecturas de red flexibles. En una conexión, los datos son enrutados por el *core network* a través de una o varias funciones del plano de usuario o UPFs (*User Plane Functions*). La conexión entre el UE y la red de datos o DN (*Data Network*) se establece a través de una sesión PDU (*Protocol Data Unit*). El plano de control está formado por varias NFs. Entre estas NFs se encuentra la función SMF (*Session Management Function*) que está a cargo de la gestión de las sesiones (establecimiento, modificación y liberación) y configura el direccionamiento del tráfico en el UPF para enrutar el tráfico al destino adecuado. La NF NEF (*Network Exposure Function*) expone las capacidades y eventos de las NF a terceros, como la función de aplicación o AF (*Application Function*), o el MEC. Las entidades del *backend* interactúan con el 5GC para proporcionar servicios utilizando un AF. En [26] se puede encontrar una lista completa y una descripción de las NFs del plano de control.

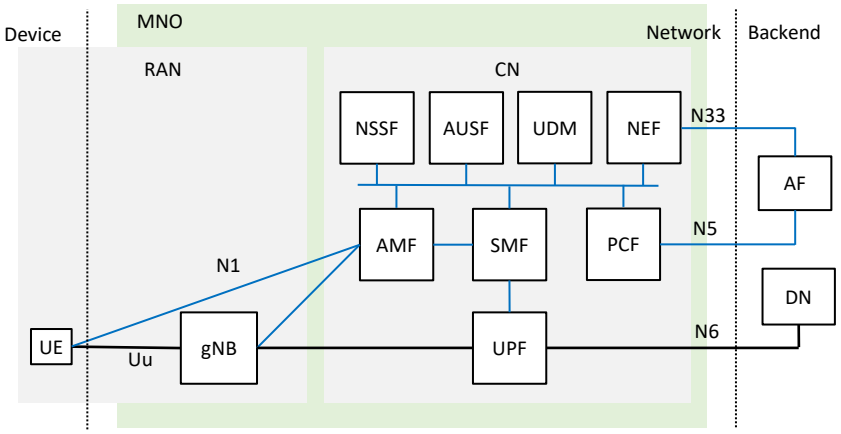


Figura 2. Arquitectura del sistema 5G [26].

4.2. Tecnología 5G para comunicaciones de baja latencia y alta fiabilidad

Una de las características clave de la tecnología 5G es su flexibilidad para adaptarse a requisitos de servicio muy diferentes: satisfacer requisitos de altas tasas de transmisión para servicios eMBB (*enhanced Mobile Broadband*), soportar transmisiones con requisitos de latencia muy bajos y altas demandas de fiabilidad para servicios URLLC (*Ultra-Reliable Low Latency Communications*), o soportar de manera eficiente la conexión de cantidades masivas de nodos a la red para servicios mMTC (*massive Machine Type Communications*). En este apartado nos centramos en estudiar los principales habilitadores introducidos por 5G para dar soporte a comunicaciones URLLC. Estos son la definición de nuevas numerologías que definen recursos radio o *slots* de transmisión con menor duración en el tiempo para reducir la latencia de la comunicación. Además, 5G permite utilizar *slots* completos a solamente algunos símbolos del *slot* (lo que se conoce como *mini-slots*) para reducir todavía más el tiempo de transmisión del paquete. Además, la reducción del *slot* de transmisión también conlleva una reducción de los tiempos de procesado de los datos en el transmisor y el receptor, lo cual también ayuda a la reducción de la latencia en la transmisión de un paquete. 5G NR también define diferentes esquemas de modulación y codificación (*Modulation and Coding Schemes* o MCS) que proporcionan diferente grado de robustez y eficiencia en el uso de los recursos radio. Es importante destacar la definición de MCS que pueden garantizar un BLER igual a 10^{-5} para servicios URLLC si se utiliza en base a los niveles de *Channel Quality Indicators* o CQIs definidos en el estándar. A nivel MAC, 5G NR introduce además del modo de *scheduling* dinámico, la posibilidad de utilizar métodos de *scheduling* con asignación de recursos semi-estática que eliminan la necesidad de solicitar/informar sobre los recursos radio asignados para llevar a cabo una transmisión cada vez que se tenga que transmitir un paquete de datos. Eliminar este intercambio de señalización entre el UE y el gNB antes de la transmisión del propio paquete permite reducir considerablemente la latencia de la transmisión. Además, para lograr una interacción eficiente de servicios eMBB y URLLC, 5G NR introduce *preemptive scheduling* según el cual permite la superposición de transmisiones URLLC más urgentes sobre transmisiones eMBB en curso. Además de los distintos mecanismos y procesos para reducir la latencia, 5G NR también define nuevos procesos para aumentar la fiabilidad. Por ejemplo, 5G NR permite la transmisión de paquetes redundantes mediante el uso de retransmisiones HARQ o transmitiendo varias réplicas del mismo paquete en *slots* consecutivos (*k*-repetitions). Gracias al uso de numerologías con *slots* de menor duración en el tiempo y el uso de *mini-slots*, es posible reducir considerablemente la latencia de los paquetes al utilizar HARQ. Sin embargo, HARQ requiere el intercambio de mensajes entre el transmisor y el receptor que introduce una latencia no despreciable. El envío de varias réplicas del mismo paquete de forma proactiva elimina el intercambio de mensajes entre el transmisor y el receptor, y por tanto la latencia asociada. Además, 5G permite incorporar redundancia adicional para alcanzar los exigentes requisitos demandados por la Industria 4.0 mediante la duplicación y transmisión de los paquetes de datos por caminos redundantes. Todas estos mecanismos y tecnologías habilitadoras para el soporte de comunicaciones URLLC se presentan con mayor detalle a continuación.

4.2.1. Numerología, estructura de trama y duración del slot de transmisión flexible

5G NR se basa en OFDM (*Orthogonal Frequency Division Multiplexing*) similar a LTE, pero introduce el uso de numerologías flexibles. Cada numerología define un espaciado de subportadoras o *Subcarrier Spacing* (SCS) en el dominio de la frecuencia y una duración del *slot* de transmisión en el dominio del tiempo diferente. La Tabla 8 muestra las diferentes

numerologías μ definidas por 5G NR. Para la numerología μ, el SCS es igual a 2^{μ}·15 kHz. Los SCS de 15, 30 y 60 kHz se utilizan para el rango de frecuencia más bajo (FR1, 410 MHz-7.125 GHz), y los SCS de 60, 120 y 240 kHz se utilizan para el rango de frecuencia más alto (FR2, 24.25 GHz-52.6 GHz). Un RB o *Resource Block* está formado por 12 subportadoras consecutivas en el dominio de la frecuencia. En el dominio del tiempo, la duración del intervalo de tiempo disminuye a medida que la numerología aumenta en $1/2^{\mu}$. La duración del *slot* varía desde 1 ms para μ=0 a 0.0625 ms para μ=4. La numerología puede adaptarse flexiblemente en función de los requisitos del tráfico a dar soporte, utilizando *slots* de menor duración para garantizar requisitos de latencia más exigentes. 5G NR utiliza un prefijo cíclico normal para todas las numerologías y puede utilizar un prefijo cíclico extendido para la numerología 2. Un *slot* consta de 14 o 12 símbolos OFDM cuando se usa el prefijo cíclico normal y extendido, respectivamente. Las transmisiones de enlace descendente y ascendente se organizan en tramas con una duración de 10 ms. Cada trama consta de diez subtramas de 1 ms de duración, y cada subtrama consta de un número diferente de *slots* para cada numerología: desde 1 *slot* para μ=0 a 16 *slots* para μ=4.

En 5G NR, se puede utilizar *Time Division Duplex* (TDD) y *Frequency Division Duplex* (FDD). La definición de la trama en modo TDD es muy flexible. En 5G NR, cada símbolo de una trama TDD se puede configurar para que ser utilizado en el enlace descendente o *downlink* (DL), en el enlace ascendente o *uplink* (UL) o de forma flexible. Con FDD, los recursos UL y DL están siempre disponibles. 5G NR permite transmisiones usando *slots* completos (es decir, los 14 o 12 símbolos OFDM de un *slot*), pero también *mini-slots* con 1 a 13 símbolos OFDM en UL y con 2 a 13 símbolos OFDM en DL [27]. La numerología, la estructura de la trama y la duración de la transmisión se pueden adaptar para satisfacer los diversos requisitos de los diferentes servicios de la Industria 4.0.

Tabla 8. Numerologías y sus características [27].

Numerología (μ)	SCS (2^{μ}·15 [kHz])	Prefijo cíclico	Símbolos OFDM por *slot*	Duración del *slot* [ms]	Slots por subtrama
0	15	Normal	14	1	1
1	30	Normal	14	0.5	2
2	60	Normal, extendido	14/12	0.25	4
3	120	Normal	14	0.125	8
4	240	Normal	14	0.0625	16

4.2.2. Tiempos de procesado más cortos

La codificación y decodificación de datos en el transmisor y el receptor incorporan retrasos no despreciables que se denominan *processing delays*. Los *processing delays* en 5G se reducen simplemente debido a la menor duración de las ranuras de transmisión: los retrasos en el procesamiento dependen por tanto de la numerología. Además, 5G NR define dos tipos diferentes de UE en función de sus capacidades de procesamiento. UEs pueden tener capacidad de procesamiento 1 o 2. La capacidad de procesamiento 1 es el requisito mínimo que debe soportar un UE. La capacidad de procesamiento 2 es una capacidad avanzada de procesamiento

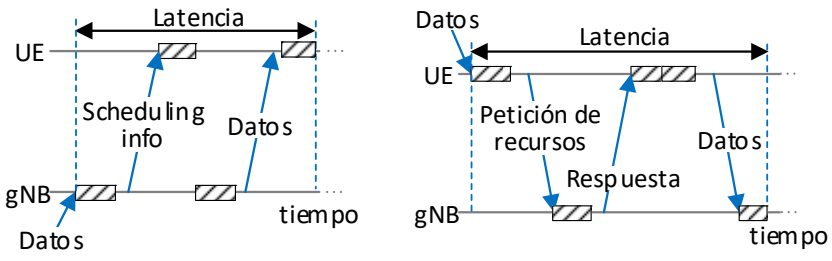
del UE que los UE pueden implementar opcionalmente, y establece tiempos de procesamiento de los canales de datos más cortos para casos de uso y servicios de baja latencia.

4.2.3. Modulación y codificación

5G NR introduce el uso de esquemas de codificación *Low Density Parity Check* (LDPC) para reemplazar los Turbo Codes utilizados en LTE para los canales de datos. LDPC proporciona un mejor rendimiento en términos de tasa de error de bits, así como procedimientos de codificación y decodificación más rápidos en comparación con los Turbo Codes [28]. 5G NR define el uso de modulaciones QPSK, 16QAM, 64QAM o 256 QAM. Sobre la base de estas modulaciones y codificaciones, 5G NR define tres tablas de esquemas de codificación y modulación o MCS diferentes en [29] para la transmisión de datos. La Tabla de MCSs 1 y la Tabla de MCSs 3 definidas en [29] consideran el uso de modulaciones QPSK, 16QAM y 64QAM, mientras que la Tabla de MCSs 2 también considera 256QAM. La Tabla 1 y la Tabla 2 se han definido para proporcionar una alta eficiencia de espectro (hasta 5.5547 y 7.4063 (bit/s)/Hz respectivamente), y un BLER objetivo de 0.1 cuando se utiliza de acuerdo con las tablas de CQI (*Channel Quality Indicator*) 1 y 2 definidas en [29]. La Tabla de MCSs 3 considera el uso de tasas de codificación más bajas en comparación con la Tabla 1 y la Tabla 2 para definir MCS más robustos aunque proporcionan una eficiencia de espectro de 4.5234 (bit/s)/Hz o menor con un BLER objetivo de 10^{-5} si se usa de acuerdo con la tabla de CQI 3 definida en [29].

4.2.4. Esquemas de *scheduling*

5G NR utiliza la asignación de recursos dinámica y semi-estática para la transmisión de datos tanto en DL como en UL para soportar servicios con diferentes requisitos [30]. El *scheduling* dinámico asigna recursos de manera dinámica para cada transmisión. En este caso, el UE y el gNB deben intercambiar mensajes de control para solicitar y/o asignar los recursos radio cuando se genera un paquete (ver Figura 3). El *scheduling* dinámico garantiza una gestión eficiente de los recursos radio, pero el intercambio de señalización entre el UE y el gNB conlleva un retardo que puede comprometer el cumplimiento de requisitos de latencia muy estrictos. 5G NR también introduce el uso de esquemas de *scheduling* semi-estáticos como son *SemiPersistent Scheduling* (SPS) para transmisiones DL y *Configured Grant* para transmisiones UL. Estos esquemas pre-asignan recursos periódicamente a los UEs para la transmisión de datos. Los recursos se asignan previamente cuando el UE se conecta al gNB, es decir, antes de que se generen los paquetes de datos. Por lo tanto, el UE no necesita solicitar recursos cada vez que tiene un nuevo paquete para transmitir, evitando el retardo que supone el intercambio previo de señalización para la solicitud y/o asignación de recursos (ver Figura 4).



a) Transmisión en DL. b) Transmisión en UL.

Figura 3. Latencia radio experimentada al utilizar *dynamic scheduling* (los rectángulos rayados representan tiempos de procesado en el transmisor o receptor).

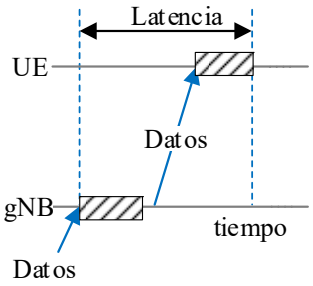


Figura 4. Latencia radio experimentada al utilizar *Configured Grant* in UL (los rectángulos rayados representan tiempos de procesado en el transmisor o receptor).

4.2.5. Retransmisiones HARQ y *k-repetitions*

5G NR permite transmitir copias redundantes del mismo paquete para aumentar la fiabilidad de la transmisión. 5G NR define dos enfoques diferentes. El primero se basa en retransmisiones asincrónicas utilizando *Hybrid Automatic Repeat Request* (HARQ) [30]. Con HARQ, un paquete se retransmite si el receptor informa al transmisor de que el paquete no se recibió correctamente o si el tiempo de espera máximo establecido para recibir una confirmación positiva por parte del receptor se agota. El tiempo entre la primera transmisión y la retransmisión es flexible con HARQ asíncrono en 5G NR. Esto permite optimizar la configuración del proceso HARQ en función de las condiciones del enlace. Las retransmisiones HARQ mejoran la fiabilidad a costa de la latencia de la comunicación debido al intercambio de mensajes entre el transmisor y el receptor. Sin embargo, considerando los menores tiempos de procesado y el uso de numerologías más altas y *mini-slots*, es posible realizar una primera transmisión del paquete en DL y una retransmisión HARQ cumpliendo con los requisitos de 1 ms de latencia para tráfico URLLC utilizando un SCS de 30 kHz y un *mini-slot* de 2 símbolos [31].

5G NR también permite transmitir *k* réplicas del mismo paquete en *slots* consecutivas; *k* puede ser igual a 2, 4 u 8 [32]. Este enfoque es una alternativa proactiva a HARQ que elimina la necesidad de solicitar una retransmisión y, por lo tanto, reduce el retraso asociado. Sin embargo, el uso de *k*-repeticiones puede reducir la eficiencia en el uso de los recursos, ya que algunas de estas réplicas podrían ser innecesarias si se recibió con éxito una réplica transmitida previamente del paquete.

4.2.6. *Preemptive scheduling*

Para garantizar una coexistencia eficiente entre los servicios eMBB y URLLC en el enlace descendente, 5G NR introduce *preemptive scheduling* [31]. *Preemptive scheduling* permite sobrescribir parcialmente una transmisión eMBB en curso con una transmisión URLLC urgente más corta. El gNB puede anunciar tales apropiaciones a los UE afectados que cursan el tráfico eMBB enviándoles una indicación de transmisión interrumpida, de modo que el UE sepa que parte de su transmisión se ha sobrescrito. Con el fin de minimizar el impacto en los UE con tráfico eMBB que pueden verse afectados por este tipo de transmisiones (*preemptive scheduling*), 5G NR introduce las retransmisiones HARQ basadas en grupos de bloques de código. Las retransmisiones HARQ basadas en grupos de bloques de código son un mecanismo de recuperación inteligente con el que solo retransmite los grupos de bloques de código afectados (es decir, un subconjunto del bloque de transmisión completo). Esta es otra de las mejoras de HARQ que introduce 5G NR. Para el enlace ascendente, una solución equivalente se ha estandarizado en Release 16, donde el gNB puede informar a los UE de eMBB que cancelen una

transmisión en curso para liberar rápidamente recursos radio para transmisiones URLLC urgentes.

4.2.7. Duplicación de paquetes de datos y envío por caminos redundantes para el aumento de la fiabilidad

Para aumentar la fiabilidad de las transmisiones y alcanzar los ultra-exigentes niveles de fiabilidad demandados por algunos servicios de la Industria 4.0, 5G NR introduce la posibilidad de duplicar paquetes y enviarlos por caminos redundantes. De este modo, se introduce una redundancia adicional y, por lo tanto, se logra una mayor probabilidad de que el paquete se reciba de forma correcta, suponiendo que haya una baja correlación de errores en los dos cominos. La duplicación de los paquetes puede realizarse a distintos niveles. En primer lugar, se propone la duplicación de paquetes de datos o PDU en la capa *Packet Data Convergence Protocol* (PDCP) para enviarlos al UE a través de dos canales diferentes. La duplicación de paquetes a nivel PDCP puede, por ejemplo, ser aplicado a un UE que está en modo de conectividad dual con dos gNB diferentes, perteneciendo por ejemplo cada gNB a una macro celda y a una *small* celda que operan a diferentes frecuencias. Sin embargo, la duplicación de paquetes conlleva el riesgo del aumento del nivel de interferencias y colas adicionales en los gNB debido al aumento de carga [33].

A nivel de capa MAC/PHY, es posible utilizar multi-TRP (*Transmission Reception Point*). En este caso, se considera que las funciones del gNB se pueden dividir en distintos nodos. En este caso, se considera que las funciones relativas a todas las capas de la pila de protocolos de la RAN a excepción de la capa PHY se implementan en un nodo centralizado denominado *Centralized Unit* o CU, mientras que las funciones de la capa PHY se implementan en *Distributed Units* o DUs que son nodos radio remotos o *remote radio heads*. De esta manera, el paquete o PDU a nivel de MAC se duplica y transmite desde dos nodos diferentes al UE.

Una última opción es duplicar el paquete a nivel de aplicación y enviarlo por dos caminos diferentes a través del *core network* y de la RAN.

4.3. 5G NR mmWave para requisitos de ancho de banda elevados

Como se presentó en el apartado 3, determinados servicios de la Industria 4.0 demandan, además de bajas latencias y alta fiabilidad, altas tasas de transmisión y gran capacidad a las redes de comunicaciones; por ejemplo, este es el caso de la XR o *Extended Reality*. En este contexto, 5G ha sido definido para trabajar en distintas bandas de frecuencia entre 400 MHz y 52.6 GHz, las cuales se muestran en la Tabla 9. Mientras las frecuencias bajas proporcionan buenas condiciones de cobertura, las frecuencias altas denominadas como banda milimétrica o mmWave proporcionan potencialmente altas tasas de transmisión y capacidad. La principal desventaja del uso de estas altas frecuencias son las elevadas pérdidas por propagación, reducida difracción y altas pérdidas por penetración. Sin embargo, el uso de tecnologías MIMO (*Massive Multiple Input Multiple Output*) y técnicas avanzadas de *beamforming* o formación de haces permite aumentar sustancialmente la eficiencia espectral y la cobertura de la red. Gracias al reducido tamaño de las antenas a altas frecuencias (el tamaño de la antena es comparable a la longitud de onda y por tanto el tamaño de la antena es menor a frecuencias más altas), las pérdidas en la señal transmitida se pueden compensar con el uso de haces direccionales. Es importante destacar que gracias al reducido tamaño de las antenas en mmWave, *massive* MIMO o MIMO masivo podrá implementarse no solo en la BS sino también en los dispositivos [34] .

Tabla 9. Rangos de frecuencia definidos para 5G NR en Release 17 [35].

Denominación	Rango de frecuencia
FR1	410 MHz – 7125 MHz
FR2	24250 MHz – 52600 MHz

El espectro mmWave ofrece la posibilidad de utilizar amplios anchos de banda contiguos, hasta 800 MHz de ancho de banda por operador y banda. Con diseños de antenas más avanzados y técnicas de procesamiento de RF (*Radio Frequency*), mmWave puede ofrecer inicialmente velocidades de datos máximas de alrededor de 5 Gbps servicios que demanden requisitos de ancho de banda extremos, y en el futuro se espera llegar hasta los 20 Gbps [36], lo que claramente mejora las tasas alcanzables con 5G NR sub-7GHz. Estas tasas de transmisión más elevadas también se resultan en mejores rendimientos en el UL. Junto con otras tecnologías ya presentadas en apartados anteriores de mejoras del UL, 5G NR mmWave ofrece nuevas oportunidades para una mayor capacidad y una experiencia mejorada para casos de uso como video HD (4k, 8k), XR, AR interactiva, etc. Estos servicios son, sin duda, muy importantes para el desarrollo de la Industria 4.0 [37].

Carrier aggregation es una de las tecnologías más eficientes para conseguir mayores tasas de transmisión y aumentar la eficiencia espectral. *Carrier aggregation* permite agregar varias portadoras para aumentar el ancho de banda asignado a un usuario y es aplicable en las distintas bandas de frecuencia definidas para 5G NR. Actualmente, en las bandas de 5G NR mmWave es posible agregar 8 *Components Carriers* (CC) de 100 MHz de espectro en DL, consiguiendo un total de 800 MHz con una tasa de transmisión pico de 4.2 Gbps, y 2 CC en UL, alcanzando una tasa de transmisión pico de 216 Mbps [38]. Es importante recordar que en la banda de frecuencia FR2 o mmWave, se establece el uso de las numerologías más altas con SCS de 60 y 120 MHz. Gracias a la definición de numerologías flexibles y escalables, se reduce considerablemente la complejidad del procesado necesario con elevados anchos de banda [39].

4.4. *Network Slicing*

Una importante novedad que introducen las redes 5G para poder dar soporte a la creciente digitalización industrial y económica es el paradigma *Network Slicing*, cuyo desarrollo es posible gracias a la integración de las tecnologías de virtualización NFV (*Network Function Virtualization*) y softwarización SDN (*Software Defined Networking*) [40]. El concepto de *Network Slicing* se basa en la creación de múltiples sistemas lógicos o virtuales que se ejecutan sobre una única arquitectura de red física compartida [41]. El principal objetivo detrás del uso de *Network Slicing* es dividir los recursos físicos de la red para agrupar de manera óptima el tráfico y configurar los recursos de la red a un nivel macro [42]. *Network Slicing* divide una única red física común en varias redes virtuales de extremo a extremo (E2E), posibilitando la realización de particiones de la red de forma que cada partición (o *slice*) ofrezca un servicio determinado con requisitos específicos en términos de latencia, ancho de banda y fiabilidad. Cada *slice* deberá estar aislada de forma lógica y disponer de sus propios recursos, mecanismos de seguridad, especificaciones de calidad de servicio, etc. El aislamiento y la independencia de las *slices* son fundamentales para asegurar que el tráfico y servicios que soporta una *slice* no afecten negativamente al resto. Así pues, *Network Slicing* busca asegurar la personalización del servicio (*customization*) y el

aislamiento (*isolation*) sobre una infraestructura común de red de acceso radio física al permitir la separación lógica y virtualización de los recursos de la red.

La arquitectura del sistema y los requisitos para soportar *Network Slicing* en redes 5G se definen en [26][43]. El 3GPP también describe el procedimiento de gestión de *slices* en [43], que divide su ciclo de vida en 4 fases:

- Preparación (*Preparation*): la fase de preparación incluye la evaluación de los requisitos de la *slice*, el diseño y la definición de sus atributos y su incorporación. En esta fase, también se prepara el entorno de red. Las tareas relacionadas con la fase de preparación están a cargo de un nodo orquestador (*orchestrator*), que tiene un conocimiento global de la red. Gracias a la información recopilada de las diferentes celdas (físicas/virtuales), el Orquestador puede evaluar la necesidad de nuevas *slices* para garantizar los requisitos del sistema/red, así como la capacidad de una nueva *slice* para cumplir con los requisitos de comunicación exigidos, basado en la disponibilidad de recursos.
- Puesta en servicio (*Commissioning*): esta fase incluye la creación de la *slice*. Esta tarea también la realiza el orquestador. El orquestador asigna los recursos necesarios a la nueva *slice* y los configura para cumplir con los requisitos establecidos.
- Operación (*Operation*): la fase de operación incluye la activación, supervisión, monitorización de rendimiento, modificación y desactivación de una *slice*. Las principales decisiones sobre la activación o desactivación de una *slice* están a cargo del orquestador. Sin embargo, a nivel local se podrán realizar algunas tareas destinadas a configurar o activar/desactivar los servicios de comunicación a nivel celular.
- Desmantelamiento (*Decommissioning*): esta fase se ejecuta en el orquestador. El orquestador se encarga de eliminar y liberar los recursos de una *slice* cuando ya no se necesita.

Network slicing también permite el uso de redes de múltiples clientes/arrendatarios o *tenants*, en las que varios *tenants* pueden compartir la misma infraestructura al asociar una *slice* a cada uno de ellos, lo que facilita la implementación y el funcionamiento rentables de las futuras redes 5G. Mientras el concepto de *Network Slicing* abarca las conexiones extremo a extremo, muchas investigaciones abordan el problema de RAN *slicing*. Este problema puede ser más complejo que el *slicing* en el *core* de la red, ya que implica la integración de múltiples configuraciones y servicios sobre un conjunto común de recursos radio.

En este contexto, diversos estudios han trabajado en la caracterización de estrategias de gestión de recursos radio (RRM) en escenarios 5G de múltiples *tenants* y servicios. Por ejemplo, el trabajo en [44] presenta un modelo de Markov para RAN *slicing* capaz de caracterizar diversas estrategias RRM considerando servicios de tasa de bits garantizados y no garantizados. El modelo propuesto captura el hecho de que diferentes enlaces radio de diversos usuarios pueden experimentar distintas calidades, lo que permite un modelo preciso de la aleatoriedad asociada a los requisitos reales de los recursos. El modelo integra un procedimiento de asignación de recursos que tiene en cuenta las condiciones de propagación variables a la hora de caracterizar y obtener la distribución de probabilidad de los recursos requeridos y asignados de acuerdo con los parámetros de QoS del servicio. El trabajo estudia el efecto de introducir diferentes funciones RRM mediante la evaluación de diferentes métricas de rendimiento (probabilidad de bloqueo, probabilidad de degradación, *throughput*, ocupación, etc.) en un escenario que considera a dos *tenants* que prestan servicios GBR y no GBR.

La asignación de recursos radio a cada *slice* de acuerdo con sus requisitos es una parte fundamental de *Network Slicing* que generalmente se ejecuta en la RAN. Diversos estudios abordan este problema proponiendo diversos algoritmos. En [45], dicho problema de asignación se formula como un problema de maximización de la tasa de transmisión total sujeto a la restricción de ortogonalidad (es decir, aislamiento de servicios), a la restricción relacionada con la latencia y a la restricción de tasa mínima, mientras mantiene la restricción relativa a la fiabilidad con la incorporación de la modulación y codificación adaptativa (AMC). El objetivo es poder multiplexar usuarios eMBB y URLLC sobre los mismos recursos radio. El problema formulado no es matemáticamente tratable debido a la presencia de una función escalonada asociada con el AMC y una variable de asignación binaria. Para poder resolverlo, los autores realizan una aproximación transformándolo en un problema de programación lineal continua. Asimismo, proponen una solución heurística que trata de aproximarse a la solución óptima. Los resultados obtenidos mediante simulación muestran el compromiso entre el requisito de tasa mínima de los usuarios de eMBB y el requisito de latencia de los usuarios de URLLC. El algoritmo de programación basado en optimización supera al algoritmo de programación basado en heurística porque es capaz de proporcionar la tasa mínima a todos los usuarios de eMBB. Por el contrario, el algoritmo heurístico supera al algoritmo de optimización en términos de latencia y desocupación de las colas de usuarios de URLLC. Además, los resultados muestran que la tasa de transmisión total en ambos esquemas es prácticamente la misma.

Este trabajo [46] ha abordado el problema de la planificación del espectro en RAN *slicing* proponiendo un modelo que tiene en cuenta aspectos relativos al negocio para definir los requisitos de los *tenants* desde una perspectiva de los recursos de la red. Luego, siguiendo este modelo, se han desarrollado diferentes estrategias de planificación del espectro para RAN *slicing* basadas en varios niveles de aislamiento y granularidad de recursos. Estas estrategias se centran en la especificación de capacidad orientada a los recursos (en contraposición a la orientada al rendimiento). Cada estrategia posible le da al proveedor de infraestructura diferentes grados de flexibilidad para asignar recursos. Aprovechando esta flexibilidad, el proveedor puede adaptar eficientemente los recursos de la red a las demandas de tráfico de los sectores. Los resultados muestran que las estrategias sin aislamiento proporcionan el mejor rendimiento de la red debido a un uso más eficiente de los recursos. Las estrategias basadas en una especificación de capacidad por red, en contraposición a una definición por celda, permiten una mejor adaptación a las variaciones de tráfico espacial, lo que resulta en un mayor rendimiento para una baja ocupación de recursos de la red y una alta correlación de tráfico entre las *slices*. Las estrategias con aislamiento intracelular o intercelular proporcionan una protección similar contra la interferencia entre *slices*. Sin embargo, para una alta ocupación de recursos y una baja correlación de tráfico, el aislamiento intracelular da como resultado un mejor rendimiento debido a la mayor flexibilidad para adaptar los recursos a las demandas de los segmentos.

El diseño de una estrategia de asignación de recursos eficiente que combine RAN *slicing* y *scheduling* es importante para garantizar la QoS requerida por los servicios. En este sentido, el trabajo en [47] se centra en garantizar la latencia y fiabilidad del tráfico ascendente URLLC esporádico al tiempo que se mejora la QoS de los servicios eMBB continuos (por ejemplo, la calidad del video). Para ello, proponen un modelo de optimización que considera el consumo de energía y la calidad del servicio para construir su función de coste en el dominio del tiempo y la frecuencia, sujeto a la restricción de latencia. Además, dada su complejidad, se diseña un algoritmo que combina la asignación de ancho de banda a largo y corto plazo. Los resultados de

simulación verifican que el algoritmo propuesto supera a algoritmos existentes en términos de latencia y consumo de energía.

En [48] aplican el concepto de *Network Slicing* a las redes industriales, presentando diferentes métodos para la creación de *slices* con protocolos de comunicación industriales deterministas y no-deterministas. Las propuestas realizadas simplifican la capacidad de administración de redes heterogéneas con diversos requisitos de aplicación. Los autores analizan su rendimiento en cuanto a utilización, fiabilidad y aislamiento. Además, presentan un caso de uso de la Industria 4.0 que ilustra cómo se puede utilizar *Network Slicing* para manejar los diversos requisitos y que utilizan para demostrar cómo se puede utilizar DNC (*Deterministic Network Calculus*) para analizar las latencias de un extremo a otro de la red.

A medida que la red 5G se vuelve cada vez más avanzada para atender casos de uso muy diversos, su complejidad y la cantidad de datos generados hacen que el diseño, la planificación y la gestión de forma manual sean ineficientes. Además, abordar la dinámica de las *slices* mediante la configuración conjunta de multitud de parámetros para lograr un rendimiento óptimo con garantías de servicio requiere una administración de gran complejidad. Resolver estos problemas de forma manual es complicado, por lo que se requieren mecanismos de automatización. Dada la disponibilidad de abundantes recursos de computación y de datos de la RAN, diversos autores proponen el empleo de inteligencia artificial. Empleando inteligencia artificial, el sistema tendría capacidad para aprender, reconocer y adaptarse de acuerdo con los requisitos de la aplicación. En efecto, las soluciones de *slicing* basadas en inteligencia artificial pueden abordar los diferentes y muy complejos problemas que surgen en múltiples niveles, incluido el *scheduling* del tráfico de cada *slice* en la RAN, la asignación de recursos a cada *slice* en el núcleo de la red y el control de admisión de nuevas *slices* [49]. De acuerdo con [50], se aplicarán técnicas de inteligencia artificial para cumplir con los requisitos mencionados anteriormente para una red 6G. Los algoritmos de aprendizaje automático o *Machine Learning* (ML) y aprendizaje profundo o *Deep Learning* (DL) se conciben como las técnicas habilitadas por la IA más populares para redes más allá de 5G. Ello incluye aprendizaje supervisado, aprendizaje no supervisado, aprendizaje por refuerzo y redes neuronales. En [51] se proporciona una descripción general del aprendizaje supervisado y no supervisado y aprendizaje por refuerzo para *Network Slicing*, destacando las ventajas e inconvenientes de cada una de esas técnicas. Dicho estudio también destaca que la predicción de la movilidad probablemente será beneficioso para la gestión de los sistemas Beyond 5G debido a los continuos traspasos o *handover*. La predicción de la movilidad se puede lograr de forma inteligente mediante la aplicación de enfoques de aprendizaje automático dado que la eficiencia de los traspasos puede resultar difícil en la gestión de los requisitos de URLLC, eMBB y MTC. En lo que a gestión de recursos radio respecta, se ha identificado ya el potencial de la inteligencia artificial [51]. En [52] se propone el empleo de aprendizaje por refuerzo para mejorar la política de *slicing* de la red. En [53] los autores presentaron un modelo de predicción de gestión de recursos basado en ANN y agruparon el modelo en tres capas para estimar los requisitos de capacidad de servicio.

El estudio en [54] propone una arquitectura basada en inteligencia artificial para abordar el problema de RAN *slicing* considerando aplicaciones críticas. La arquitectura propuesta hace uso de aprendizaje por refuerzo profundo en el borde (*edge*) de la red para optimizar el proceso de RAN *slicing* y la gestión de recursos radio para aplicaciones sensibles a la latencia. Los resultados confirman que la predicción del tráfico entrante por *slice* no es suficiente para una estrategia de aplicación RAN *slicing* eficaz y el sobre-aprovisionamiento de recursos es notablemente ineficiente. A pesar de que la solución propuesta presenta ventajas, aún existen diferentes retos

no resueltos, como el empleo de transferencia de aprendizaje para garantizar la re-configurabilidad y el despliegue de una solución basada en inteligencia artificial.

Los autores de [55] proponen un marco flexible y configurable para RAN *slicing*, donde los diversos requisitos de las *slices* se tienen en cuenta simultáneamente, y los algoritmos de administración de *slices* ajustan los parámetros de control de diferentes mecanismos de gestión de recursos de radio para satisfacer el acuerdo de nivel de servicio de cada *slice*. Uno de los algoritmos propuestos es heurístico y el otro utiliza una red neuronal artificial (ANN) para predecir el comportamiento de la red y tomar mejores decisiones en el ajuste de los mecanismos RRM. Además, se ha diseñado un mecanismo de protección para evitar que las *slices* se influyan negativamente entre sí. Los resultados de simulación obtenidos demuestran que cuando existe una sobrecarga local o global de una de las *slices*, el método basado en ANN aumenta el número de indicadores clave de rendimiento (KPI) que cumplen con los requisitos establecidos.

El trabajo propuesto en [56] presenta una visión del paradigma RAN *slicing* basado en inteligencia artificial en el que las *slices* personalizadas son arrendadas no solo por los operadores móviles existentes para ofrecer servicios diferenciados, sino también por operadores fijos y similares para aumentar el rendimiento y la cobertura de la red de acceso. Además de acelerar 5G, los beneficios potenciales de este paradigma incluyen un coste de red reducido, una implementación mejorada y una eficiencia operativa, y una mayor oportunidad de reutilización de activos. Sin embargo, este paradigma presenta una serie de retos futuros que merecen una mayor investigación. A diferencia del *slicing* de la red troncal, el RAN *slicing* en sistemas 5G virtualizados todavía no se ha investigado a fondo. La transferencia de un gran volumen de datos, su almacenamiento y procesamiento varios arrendatarios (*tenants*) requieren mecanismos eficientes y rentables para construir la plataforma de análisis de big data para permitir la automatización de la RAN. Otro reto es el diseño de protocolos de señalización de control para una coordinación eficiente entre los diferentes elementos de la red. Además, los beneficios potenciales del proceso de RAN *slicing* solo se pueden lograr si el coste de arrendar las *slices* de la RAN es atractivo en comparación con el coste de actualizar y mantener las redes de acceso fijo, a pesar del hecho de que el enlace inalámbrico arrendado es a menudo menos estable que una conexión alámbrica.

En [57] se propone un marco de implementación basado en el trabajo del 3GPP y la O-RAN Alliance, y sobre esa base se propone y evalúa una solución asistida por inteligencia artificial empleando aprendizaje por refuerzo de múltiples agentes (MARL) basada en la técnica *Deep Q-Network* (DQN). El artículo muestra que la solución propuesta es capaz de ajustar dinámicamente la asignación de recursos de las *slices* a las variaciones del tráfico con el fin de cumplir con los requisitos de los servicios y lograr una alta eficiencia en la utilización de recursos.

4.5. Redes 5G no públicas

Las redes no públicas (*non-public networks* o NPN), también denominadas redes privadas (*private networks*), ofrecen un gran valor a la desarrollo y digitalización de la industria. Una red NPN es una red local implementada para uso exclusivo de una organización (los recursos radio, del *core network* y de transmisión). A diferencia de una red que ofrece servicios de red móvil al público en general, una red NPN proporciona servicios de red 5G a una organización de usuarios o grupo de organizaciones claramente definido. La red NPN 5G se implementa en las instalaciones definidas por la organización, como un campus o una fábrica.

Las redes no públicas pueden ser deseables por varias razones ([58], [59]):

- Requisitos de alta calidad de servicio y/o cumplir con específicos perfiles de rendimiento.
- Garantizar cobertura en lugares con condiciones adversas de operación o de propagación radio, o en lugares en los que la cobertura pública es limitada o no existente.
- Requisitos de alta seguridad, cumplidos con credenciales de seguridad dedicadas. La seguridad y la privacidad de los datos son aspectos críticos en la industria. Por tanto, la capacidad de mantener los datos de operación de forma local es crucial para grandes industrias.
- Aislamiento de otras redes, como forma de protección frente al mal funcionamiento de la red móvil pública. Además, el aislamiento puede ser deseable por razones de rendimiento, seguridad, privacidad y seguridad, lo cual es realmente importante en el entorno industrial.
- Responsabilidad. Una red no pública facilita la identificación de la responsabilidad en cuanto a la disponibilidad, el mantenimiento y la operación de la red.
- Control de la red para aplicar configuraciones que no se soporten por la red pública.

Las redes NPN pueden ser desplegadas en múltiples escenarios. Sin embargo, en este informe nos centramos en la aplicación de este tipo de redes y su despliegue en escenarios industriales. Aun centrándonos en escenarios industriales, estas redes deben dar cobertura y adaptarse a distintos casos de uso, pudiendo utilizar distintas configuraciones de red. Según la configuración de la red NPN, estas pueden clasificarse principalmente en 2 categorías [58]:

- Redes NPN implementadas como redes aisladas e independientes.
- Redes NPN desplegadas junto a o con conexión a una red pública. Según el grado de interacción y nodos compartidos con la red pública, se definen distintas subcategorías que serán presentadas más adelante.

A continuación, se presentan las distintas configuraciones de redes NPN para la industria.

4.5.1. Redes NPN aisladas e independientes

En las redes NPN aisladas e independientes, todas las funciones de red están ubicadas en las propias instalaciones de la fábrica. Además, como se observa en la Figura 5, la red NPN está separada de la red pública. La única ruta de comunicación entre la NPN y la red pública es a través de un firewall. La responsabilidad de gestión y operación de la red NPN y de los servicios ofrecida por esta recae exclusivamente sobre la compañía o gestor OT (*Operational Technology*)[1]. La implementación de la red NPN se basa en tecnologías definidas por el 3GPP y es completamente independiente de la red pública, utilizando su propio ID de NPN dedicado.

Las redes NPN aisladas e independientes pueden incorporar una conexión opcional a la red pública a través del firewall, como se muestra en la Figura 5, para permitir el acceso a los servicios de la red pública, como la voz, mientras se encuentra dentro de la cobertura NPN. Si el propietario y operador de la red NPN así lo desea, la conexión opcional se puede aprovechar para acceder a los servicios NPN a través de la red pública.

[1] OT se refiere a equipos industriales especializados que utilizan software para controlar e interactuar con dispositivos, procesos y eventos. Los ejemplos incluyen líneas de producción robóticas, redes de sensores o equipos de plantas industriales, así como las tecnologías de redes fijas y móviles implementadas.

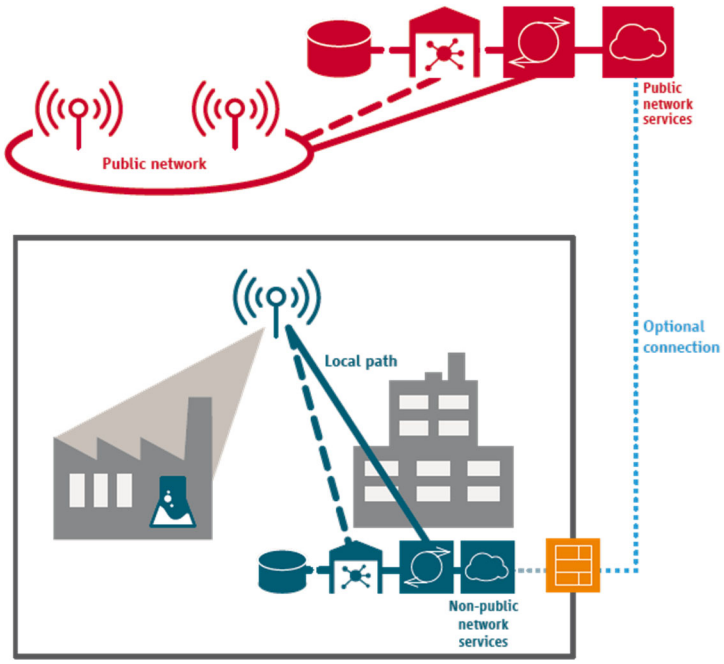


Figura 5. Red NPN aislada e independiente [58].

4.5.2. Redes NPN desplegadas junto a o con conexión a una red pública

Las redes NPN desplegadas junto a o con conexión a una red pública se despliegan en escenarios en los que ciertos casos de uso son soportados completamente por la red pública, mientras que otros requieren de redes NPN dedicadas. Por tanto, se distinguen 2 partes en la red, una pública y una no pública, con el tráfico de datos asignado a cada parte. En función de la configuración de la red, el 5G-ACIA distingue en [58] 3 tipos diferentes:

- Despliegue con RAN compartida (Figura 6).
  En este escenario, la red NPN y la red pública comparten parte de la RAN, mientras otras funciones de red permanecen separadas, tal y como se muestra en la Figura 6. Los flujos de datos relacionados con el tráfico NPN permanecen dentro del perímetro lógico de las instalaciones de la fábrica, y el tráfico de la red pública se transfiere a la red pública. En la Figura 6 solamente se representa una estación base o gNB compartida por simplicidad. Sin embargo, es posible configurar gNBs adicionales a las que solo pueden acceder los usuarios de NPN.

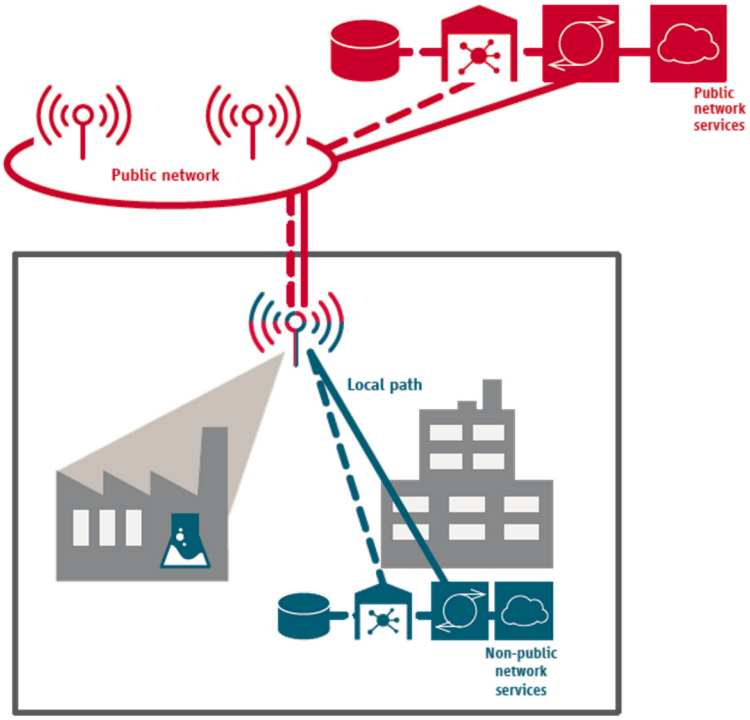


Figura 6. Despliegue de red NPN con RAN compartida [58].

- Despliegue con RAN y plano de control compartidos (Figura 7).
  En este escenario, la NPN y la red pública comparten la RAN y además las tareas de control de la red se realizan en la red pública. No obstante, todos los flujos de tráfico NPN permanecen dentro del perímetro lógico de las instalaciones de la fábrica, mientras que la parte del tráfico de la red pública se transfiere a la red pública.
  Este escenario se puede implementar utilizando *network slicing*, es decir, la creación de redes lógicamente independientes dentro de una única infraestructura física compartida. La separación de las redes públicas y privadas se logra empleando diferentes identificadores de *slices* de la red. Otra manera de lograr la separación de las redes públicas y privadas es utilizando la función definida por el 3GPP denominada *Access Point Name* (APN). El APN denota la red de destino final (dónde enrutar el tráfico), lo que permite la diferenciación entre las porciones de tráfico.
  En este escenario, la NPN está alojada en la red pública y los dispositivos NPN están suscritos a la red pública. Al igual que en el caso anterior, la relación contractual entre la NPN y el operador de la red pública sea más sencilla y se permite que los dispositivos NPN se conecten directamente a la red pública y sus servicios.

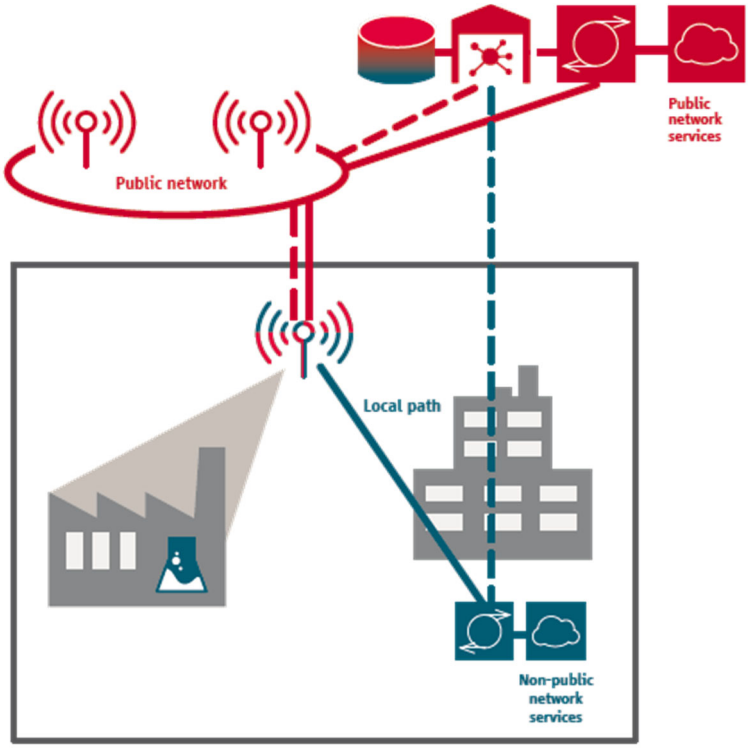


Figura 7. Despliegue de red NPN con RAN y plano de control compartidos [58].

- NPN alojado en la red pública (Figura 8).
  En este escenario, tanto la parte del tráfico de la red pública como la parte del tráfico NPN son externas a las instalaciones de la fábrica, pero se tratan como si fueran partes de redes completamente diferentes. Esto se logra mediante la virtualización de funciones de red en un entorno de la nube (genérico). Estas funciones se pueden utilizar para fines de redes públicas y privadas. Este escenario se puede implementar mediante *network slicing* o la aplicación de la funcionalidad APN.
  En este escenario, los suscriptores de NPN son, por definición, también suscriptores de redes públicas. Dado que todos los datos se enrutan a través de la red pública, el acceso a los servicios de la red pública y la capacidad de itinerancia se pueden implementar fácilmente de acuerdo con el acuerdo entre la NPN y el operador de la red pública. La conexión opcional que se muestra en la Figura 1 en la sección 5.2 no es necesaria en este escenario.

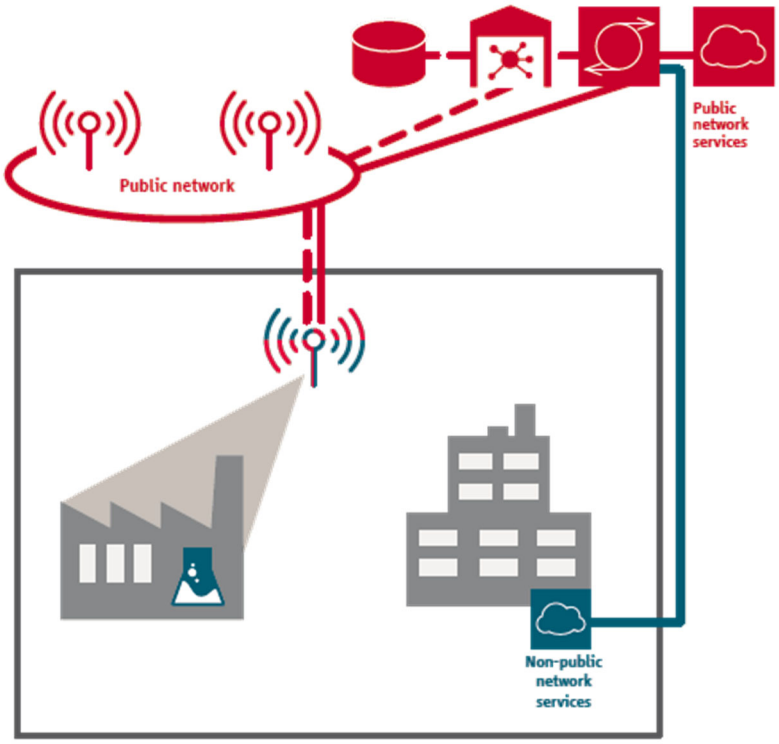


Figura 8. Despliegue de red NPN alojada en la red pública [58].

4.5.3. Espectro para redes NPN

Las redes 5G, y en particular, las redes 5G NPN, pueden utilizar espectro con o sin licencia. El espectro con licencia está regulado por los gobiernos para que los proveedores de servicios y otras organizaciones puedan usar el espectro dedicado sin preocuparse de que otros usen el ancho de banda. El espectro sin licencia, por otro lado, está disponible para que lo use cualquiera. Desde el punto de vista industrial, es importante que el espectro esté disponible, y sujeto a regulaciones estables que permitan una planificación a largo plazo. A continuación, se presentan cuatro opciones de uso del espectro disponibles para las empresas industriales que buscan implementar redes 5G privadas [59]:

- Espectro con licencia propiedad de los operadores. Este es el modelo clásico de concesión de licencias de espectro. El uso protegido hace que este espectro sea atractivo para los usuarios que buscan fiabilidad.
- Espectro dedicado para uso empresarial. Según este modelo, se reserva determinado ancho de banda para uso empresarial o industrial.
- Espectro sin licencia (con uso compartido asincrónico). Se centra principalmente en la banda de 5 GHz. Por regulación, se deberán aplicar técnicas de *listen-before-talk*. Podría resultar útil para redes 5G privadas que no requieren URLLC.
- Espectro sin licencia con uso compartido sincronizado. En nuevas asignaciones de espectro sin licencia (por ejemplo, 6 GHz), existe la oportunidad de introducir nuevos mecanismos de compartición para mejorar el uso compartido, y hacerlo más adecuado para aplicaciones URLLC.

4.5.4. Impacto de la configuración de la red NPN en el rendimiento del servicio

El despliegue y configuración de la red NPN tendrá un impacto en el rendimiento de los servicios soportados por la red [58]. Por ejemplo, el hecho de que una red NPN se implemente o no como parte o como una extensión de una red pública, o si una red NPN tiene la capacidad de interactuar con redes públicas puede tener un impacto en la conectividad del dispositivo, entendido como la capacidad del dispositivo NPN para conectarse a otras redes como, por ejemplo, la red pública. Por ejemplo, los dispositivos de despliegues de red NPN basados en suscripciones a la red pública (como las redes NPN con RAN y plano de control compartidos o despliegue de red NPN alojada en la red pública) tendrán conectividad global, es decir, el dispositivo NPN podrá utilizar servicios de red pública siempre que no se encuentre en el área de servicio de la NPN. El tipo de configuración también definirá la continuidad del servicio cuando un dispositivo se mueve entre la NPN y la red pública.

La calidad de servicio o *Quality of Service* (QoS) experimentada también se verá afectada por el tipo de configuración de la red NPN. En muchos escenarios de implementación, la red NPN y la red pública utilizan la misma infraestructura y recursos. Por este motivo, el tráfico en una red puede afectar al tráfico en la otra red a menos que se proporcione un aislamiento de tráfico adecuado mediante el aislamiento de los recursos de la red. El aislamiento entre las dos redes se puede lograr de dos maneras. En primer lugar, el aislamiento se puede conseguir mediante un aislamiento físico en el que los recursos de red de la red NPN y la red pública están físicamente separados (este es el caso de las redes NPN aisladas e independientes). En segundo lugar, es posible implementar un aislamiento lógico de los recursos de la red. Esto quiere decir que, aunque ambas redes compartan una infraestructura física común, estas no pueden comunicar la una con la otra. Esto puede conseguirse mediante mecanismos como *network slicing*.

El escenario de implementación de NPN específico también puede afectar la forma en que los gestores de OT pueden gestionar y operar la NPN, es decir, si pueden crear, configurar, escalar y operar de forma estática o dinámica funciones de red, y la capacidad de capturar información importante de red y servicios. Esto podría ser necesario, p. ejemplo, para satisfacer las necesidades específicas de un proceso de fábrica automatizado.

Un aspecto crítico en los escenarios industriales es la privacidad y la seguridad para garantizar la confidencialidad y la integridad de los datos. En este contexto, la privacidad significa decidir qué información va a dónde y la seguridad ofrece la capacidad de estar seguro de que se respetan esas decisiones. Las políticas de seguridad y privacidad difieren de unas industrias a otras o de unas fábricas a otras. El grado de privacidad está influenciado principalmente por el grado de aislamiento (tanto físico como lógico) de los datos, el control y la gestión. Por lo tanto, el aislamiento de datos, el control y la gestión se consideran aspectos de servicio para evaluar el cumplimiento de la privacidad de los diferentes escenarios de implementación.

4.6. *Time Sensitive Communications* (TSC)

TSN (*Time Sensitive Networking*) es una evolución de Ethernet que está siendo desarrollado para proporcionar comunicaciones deterministas de baja latencia y ultra alta fiabilidad. TSN proporciona niveles de servicio deterministas inalcanzables hasta la fecha, pero carece de la flexibilidad que ofrecen las tecnologías móviles e inalámbricas. Las redes 5G ofrecen niveles de flexibilidad y reconfiguración sin precedentes gracias a la softwarización y virtualización de las

redes, la nueva interfaz radio 5G NR, y la introducción de tecnologías como *network slicing* y *edge computing*. Además, su capacidad para dar soporte a servicios URLLC (ver apartado 4.2) la hacen la tecnología candidata para dar soporte a la Industria 4.0 en coordinación con las redes TSN. En este contexto, es necesaria la integración de las redes 5G y TSN industriales para garantizar la conectividad necesaria para la transformación digital de la Industria.

Este apartado presenta, en primer lugar, el estándar TSN y las distintas tecnologías que implementa. A continuación, se presenta el *framework* definido por el 3GPP para lograr la eficiente integración de 5G y TSN, y dar soporte a las comunicaciones TSC.

4.6.1. *Time Sensitive Networking*

IEEE (*Institute of Electrical and Electronics Engineers*) 802.1 *Time-Sensitive Networking* (TSN) se está convirtiendo en la tecnología estándar basada en Ethernet para redes convergentes de la Industria 4.0. TSN es un conjunto de estándares que cubren aspectos críticos para permitir el uso de Ethernet en aplicaciones sensibles al tiempo que requieren determinismo tanto en la latencia como en el ancho de banda. Este tipo de aplicaciones y tráfico se soportaban anteriormente por buses de campo (*field bus*) o implementaciones de Ethernet propietarias como PROFINET. TSN habilita un ecosistema estandarizado para lograr este objetivo.

La familia de estándares 802.1 TSN se puede agrupar en cuatro áreas principales relacionadas con un objetivo de rendimiento [60]:

- Sincronización temporal: Con el objetivo de proporcionar una referencia común de tiempo compartido entre todos los nodos participantes en la red TSN, el estándar 802.1AS define *generalized Precision Time Protocol* (gPTP) [61], un perfil del protocolo IEEE 1588 *Precision Time Protocol* (PTP) [62] para lograr una sincronización de tiempo precisa, lo cual es fundamental para garantizar el comportamiento determinista de los nodos finales. Esta sincronización también será utilizada por otros mecanismos de TSN como por ejemplo, el mecanismo de *scheduling* basado en tiempo definido en 802.1Qbv.
- Latencia limitada: Uno de los principales requisitos de las aplicaciones en tiempo real es que el intercambio de mensajes entre dos nodos de la red se debe llevar a cabo con una latencia baja y determinista. TSN define varios métodos para garantizar estas latencias limitadas, como por ejemplo mecanismos para permitir la gestión de colas, el control y la configuración del tráfico (802.1Qav, 802.1Qbv) para garantizar que los paquetes de tiempo crítico reciban prioridad a medida que se reenvían a través de la red. La preferencia de tramas Ethernet o *Ethernet frame preemption*, definida en 802.1Qbu y 802.3br, que permite suspender la transmisión de una trama Ethernet no crítica en favor de una trama crítica, proporcionar una latencia baja y limitada a la vez que aumenta la eficiencia de la red.
- Fiabilidad: La fiabilidad se define como el porcentaje de paquetes que satisfacen el requisito de latencia. La red TSN debe garantizar que todos los paquetes se entreguen dentro de un límite de latencia determinado sin pérdidas de paquetes ni retrasos debido a la congestión. Para ello, TSN define en 802.1CB *Seamless Redundancy* y *Identification for streams*, que permite replicar tramas y enviarlas por 2 o más caminos disjuntos hacia el destino. Las tramas duplicadas son identificadas y eliminadas en el nodo de red anterior al destino.
- Gestión de recursos: La configuración de las capacidades de la red TSN y los recursos de los nodos es fundamental para asegurar el rendimiento extremo a extremo para

servicios o flujos de datos críticos ante la presencia de otro tipo de tráfico. Para ello, TSN permite desde un modelo de configuración de la red totalmente centralizada a un modelo totalmente distribuido (802.1Qcc).

La siguiente tabla resume los principales estándares que integran TSN. Las funcionalidades más relevantes se presentan a continuación de forma más extendida.

Tabla 10. Estándares IEEE 802.1 TSN

Estándar IEEE	Contenido
802.1AS	Sincronización temporal
802.1Qca	Control y reserva de rutas
802.1Qav	Modelado de tráfico (*Credit-based traffic shaping*)
802.1Qbv	*Time-aware scheduling*
802.1Qbu, 802.3br	*Frame preemption*
802.1Qcc	Modelos de configuración
802.1Qci	Filtrado y *policing*
802.1CB	*Seamless Redundancy, Stream Identification*
802.1Qat	*Stream Reservation Protocol* (reserve de recursos distribuida)

4.6.1.1. Configuración de la red TSN

Los componentes principales de una red TSN son [63]:

- Flujo TSN: Este término es utilizado para describir la comunicación crítica en tiempo o *time-critical* entre dispositivos finales o *end devices*. Cada flujo TSN tiene estrictos requisitos de tiempo y es identificado de manera única por los dispositivos de la red.
- *End devices*: Estos nodos son la fuente y destino de un flujo TSN. Los *end devices* ejecutan una aplicación que requiere comunicaciones deterministas. A estos nodos también se les llama *talkers* y *listeners*.
- *Bridges* o puentes: A estos nodos también se les denomina *switches* Ethernet. En TSN, estos *bridges* especiales son capaces de transmitir las tramas Ethernet de un flujo TSN en base a una programación o *scheduling* y recibir tramas Ethernet de un flujo TSN de acuerdo a un *scheduling*.
- Controlador central de la red o *Central Network Controller* (CNC): En TSN, el CNC actúa como un proxy para la red (para los *bridges* TSN y sus interconexiones) y para las aplicaciones de control que requieren comunicaciones deterministas. El CNC define la programación o *Schedule* que sigue la red para transmitir todas las tramas TSN.
- Configuración de usuario centralizada o *Centralized User Configuration* (CUC): El CUC es una aplicación que comunica con el CNC y los *end devices*. El CUC representa las aplicaciones de control y los *end devices*. El CUC solicita al CNC comunicaciones deterministas (flujos TSN) con requisitos específicos para esos flujos.

Muchas de las capacidades que se buscan en las redes TSN, especialmente la redundancia y el *scheduling* en base al tiempo, requieren de una arquitectura de red centralizada en la que controladores centrales recopilan los requisitos de la aplicación y configuran dinámicamente la red para cumplir con esos requisitos. Para dar soporte a estas necesidades. IEEE 802.1Qcc define un nuevo modelo de configuración totalmente centralizado que permite a controladores software centralizados que tienen una visión global de la red recibir los requisitos de los *streams* de los *end devices* (*Talkers and Listeners*) y configurar directamente los *bridges* necesarios para

garantizar estos requisitos. La Figura 9 muestra una representación del modelo de configuración totalmente centralizado definido en IEEE 802.1Qcc. Como muestra la figura, los *Talkers* envían *streams* de datos a los *Listeners* a través de una ruta formada por varios *bridges*. En el modelo de configuración centralizado, las conexiones de *streams* de datos se establecen como se presenta a continuación:

1. Los *end devices* (*Talkers* y *Listeners*) comunican los requisitos de calidad de servicio o QoS del *stream* de datos al CUC utilizando un protocolo de configuración específico de aplicación.
2. El CUC recoge los requisitos de varias aplicaciones en la red y los reenvía agrupados por *stream* (1 *Talker*, múltiples *Listeners*) al CNC utilizando un protocolo *User/Network Configuration Interface* (UNI).
3. El CNC calcula los tiempos de *scheduling* en cada nodo y determina los caminos por los que enviar los datos (entre otros cálculos) para garantizar los requisitos de QoS de cada *stream*.
4. Si el CNC puede satisfacer los requisitos de QoS de los *streams* de los *end devices*, realiza las siguientes acciones:
    a. El CNC configura los *bridges* en la red TSN para cumplir con los requisitos de QoS utilizando un protocolo *Network Management Protocol*.
    b. El CNC envía información relevante sobre la transmisión de los *streams* de los *end devices* al CUC utilizando el protocolo UNI.

    Si el CNC no puede satisfacer los requisitos de QoS, devuelve un error al CUC.
5. En el caso de que sea posible soportar los requisitos de QoS de los distintos *streams*, el CUC configura los *end devices* para la transmisión de los *streams* y la comunicación comienza.

Como se ha explicado, en el modelo totalmente centralizado *Talkers* y *Listeners* envían los requisitos de QoS de sus *streams* al CUC utilizando un protocolo específico de aplicación. Por ejemplo, en una aplicación que utilice un mecanismo de publicación/suscripción basado en OPC-UA[2] (*OLE for Process Control-Unified Architecture*), *Talkers* y *Listeners* pueden comunicar sus requisitos al CUC utilizando un protocolo de configuración de *streams* específico de OPC-UA. Estos protocolos de configuración específicos de las aplicaciones se definen fuera del IEEE 802, permitiendo una migración flexible de las soluciones existentes al uso de redes TSN, a la vez que se optimiza nuevas soluciones para el uso de la red TSN. El estándar IEEE 802.1Qcc-2018 [64] define las estructuras de datos a utilizar para comunicar los requisitos de los *end devices* y toda la información necesaria para la configuración final de los *end devices*. Varias organizaciones, como la *OPC Foundation*, están actualmente definiendo la comunicación entre los *end devices* y el CUC.

[2] OPC-UA es un protocolo de comunicación disponible en abierto diseñado específicamente para la automatización industrial. Permite el intercambio de información y datos en dispositivos dentro de máquinas, entre máquinas y desde máquinas a sistemas.

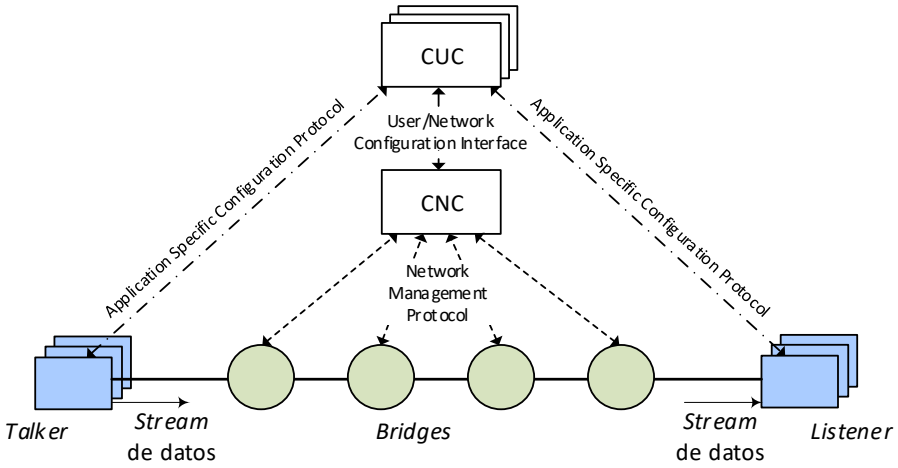


Figura 9. Modelo de configuración totalmente centralizado [65].

Además del modelo de configuración totalmente centralizado presentado en este apartado, otros modelos de configuración también son definidos en TSN. Estos son el modelo de configuración centralizado de red y distribuido de usuario, y el modelo totalmente distribuido. Estos dos modelos utilizan el protocol *Stream Reservation Protocol* (SRP), el cual a su vez utiliza los protocolos *Multiple Stream Registration Protocol* (MSRP), *Multiple VLAN Registration Protocol* (MVRP), y *Multiple MAC Registration Protocol* (MMRP). Sin embargo, según [66], se ha demostrado que SRP no lograr satisfacer las necesidades de las redes de automatización industrial. En este contexto, se están diseñando nuevos protocolos para realizar la reserva de recursos distribuida.

4.6.1.2. Sincronización temporal

La sincronización temporal resulta fundamental para garantizar el comportamiento determinista de los dispositivos. Además, algunos de TSN también requieren que todos los nodos entiendan y compartan una misma referencia temporal, por ejemplo, para realizar la programación o *scheduling* de tráfico (IEEE 802.1Qbv), o para el filtrado y *policing* de los distintos flujos (IEEE 802.1Qci).

El protocolo de sincronización temporal utilizado en TSN conocido como gPTP se define en IEEE 802.11AS [61]. gPTP se construye sobre el estándar PTP definido en IEEE 1588 [62]. IEEE 1588 define un algoritmo distribuido, denominado *Best Master Clock Algorithm* (BMCA) para seleccionar el reloj al cual se sincronizarán los nodos en la red. Con BMCA, los nodos de la red anuncian las características de su reloj de manera que hace un ranking de los relojes en el sistema. El reloj que queda en lo más alto del ranking se denomina *grandmaster*, y el resto de relojes denominados *slaves* se sincronizan a él.

El *grandmaster* envía información sobre la hora sincronizada actual a los *slaves* conectados directamente. Cada una de estas instancias de PTP corrige el tiempo sincronizado recibido agregando el tiempo de propagación desde el *grandmaster*. Si recibe una orden de retransmisión, el *slave* a su vez reenvía la información de tiempo corregida (incluidas las correcciones adicionales por retrasos en el proceso de reenvío) a todos los nodos conectados. La Figura 10 muestra un ejemplo de cómo se transmite la información de sincronización entre 3 nodos. El primer nodo *grandmaster* envía mensajes de "Sync" y "Follow_Up" al segundo nodo PTP en el tiempo del reloj local ‘*i*-1’. El segundo nodo recibe el mensaje "Sync" y marca la

recepción del mensaje, y el tiempo de recepción '*i*'. Después de recibir un mensaje "Follow_Up", el nodo calcula el campo de corrección que incluye el tiempo de residencia del sistema, y el retardo de propagación desde el nodo anterior. Por último, el segundo nodo envía un nuevo mensaje de " Sync" con el campo de corrección recalculado.

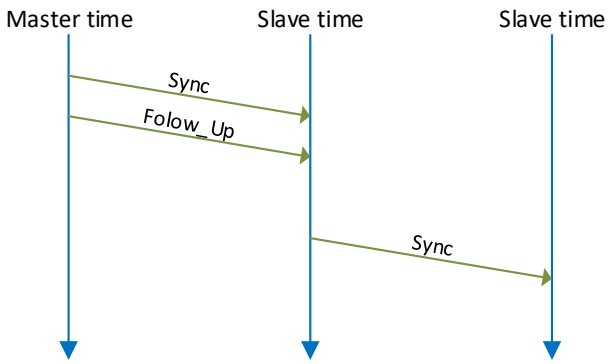


Figura 10. Transmisión de información de sincronización.

El estándar IEEE 802.1AS configura un perfil concreto de las opciones disponibles en IEEE 1588, incluyendo las siguientes [65]:

- Uso del BMCA modificado que permite una convergencia rápida al reloj *grandmaster*.
- Uso de un mecanismo de retardo de pare para medir el retardo de la ruta de la red.
- Participación obligatoria de los puentes, incluida la selección del reloj *grandmaster*.
- Especificación de un requisito de rendimiento tal que cualesquiera dos sistemas de detección de tiempo separados por siete o menos saltos se sincronizarán con una diferencia de 1 µs entre sí.

4.6.1.3. *Scheduling* o programación del tráfico

Algunos sistemas de control industrial requieren una latencia muy baja y determinista, así como una variación de la latencia muy baja ciclo a ciclo. En estos sistemas, múltiples controladores comúnmente distribuidos sincronizan sus operaciones sensor/actuador con otros controladores programando estas operaciones en el tiempo, típicamente usando un ciclo de control repetido. Para garantizar que las aplicaciones que requieren una latencia muy baja y predecible puedan cumplir con sus requisitos de calidad de servicio, TSN define en IEEE 802.1Qbv-2015 un *scheduling* de tráfico basado en el tiempo y que ha sido incluido en IEEE 802.1Q [67].

La Figura 11 muestra la división temporal en múltiples intervalos de tiempo o *time slots* que se repiten cíclicamente. En cada *time slot* se define para la transmisión de determinadas clases de tráfico. Los paquetes que pertenecen a otras clases de tráfico deben esperar hasta el *time slot* en el que se permite la transmisión de esa clase de tráfico. Si en un mismo *time slot* se permite la transmisión de varias clases de tráfico, entonces se aplica un criterio de transmisión adicional, por ejemplo, una priorización estricta. Esta gestión del tráfico se realiza mediante un mecanismo de bloqueo de puertas en función del tiempo que se abren y cierran para el envío de cada tipo de tráfico en base a los criterios definidos. Es importante que este mecanismo de *scheduling* en base al tiempo se aplica en todos los nodos de la red coordinando los *schedulers* de los *bridges* a lo largo de la ruta de los *streams* TSN. Este proceso se ilustra en la Figura 12 y Figura 13. La Figura 12 muestra las distintas puertas implementadas en un *bridge* que preceden la función de

transmisión. Cada puerta corresponde a un tipo de tráfico asociado con una cola específica. Cada puerta puede estar abierta (O) o cerrada (C) permitiendo que la transmisión de cada tipo de tráfico se active o desactive dependiendo de los requisitos de la aplicación. Una lista de control de puertas configurable determina si la puerta de un tipo de tráfico está abierta o cerrada en cada instante, y esta lista de control de puertas se ejecuta en cada ciclo de tiempo. El mecanismo de control de puertas es una aplicación basada en el tiempo PTP que opera en un *bridge* o estación final. Esto permite que la ejecución de este mecanismo de puertas se ejecute de manera precisa en toda la red.

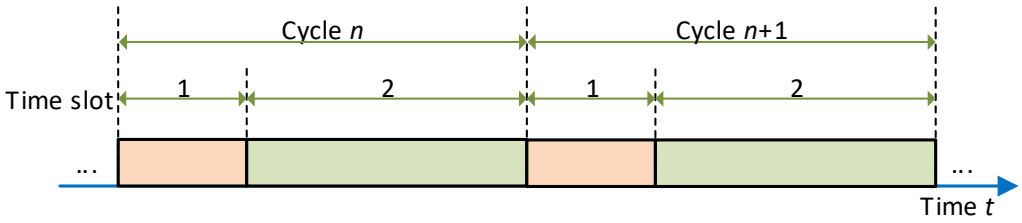


Figura 11. División cíclica del tiempo [66].

El mecanismo de control de puertas permite que una aplicación de control industrial pueda minimizar la latencia y su variación para el tráfico de control mediante la configuración de la lista de control de puertas de manera que el tráfico de control de alta prioridad sea transmitido en intervalos de tiempo conocidos. De manera similar, las operaciones de las puertas para tráfico de baja prioridad o *best effort* se puede asignar a *time slots* no utilizados para el tráfico de control de alta prioridad, asegurando que el flujo de todos los tipos de tráfico. La Figura 13 muestra un ejemplo de la programación del tráfico en distintos *bridges* de una red. Cada nodo es un *bridge* de la red. Las colas de salida se muestran en verde con una flecha negra que apunta al siguiente salto. Las barras de colores representan paquetes de cierta prioridad y visualizan cuando la puerta correspondiente está abierta para el tráfico de dicha prioridad de acuerdo con la lista de control de puerta. Las flechas negras muestran la conexión entre una cola de salida y el siguiente salto. Se permite la transmisión de cualquier otro tráfico cuando no hay paquetes marcados.

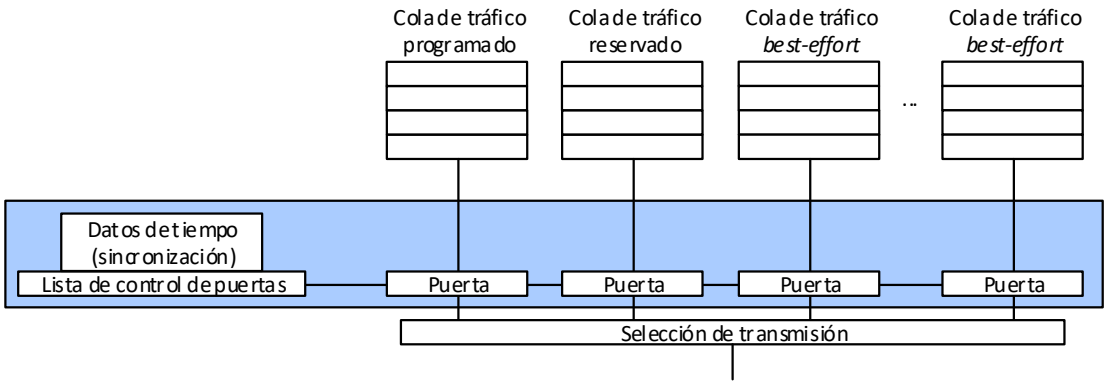


Figura 12. Ejemplo de una lista de control de puertas [66].

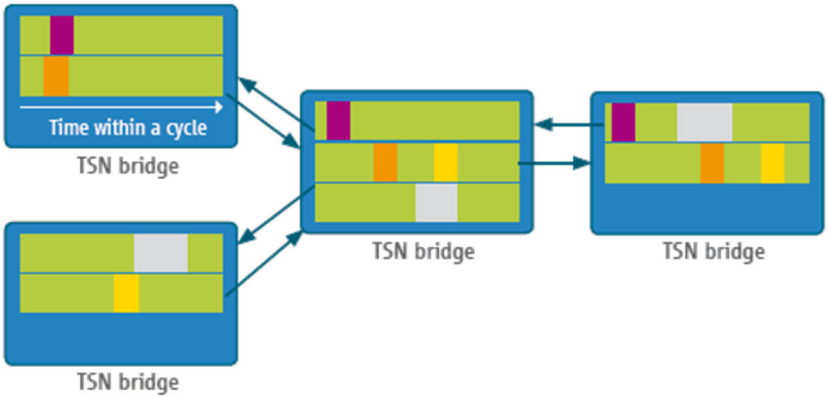


Figura 13. Ejemplo de programación del tráfico en distintos *bridges* en una red esquematizada [66].

4.6.1.4. *Frame preemption* o preferencia de tramas

Cierto tipo de tráfico debe transmitirse a través de la red con una interferencia mínima. *Frame preemption* especifica como los paquetes de este tipo de tráfico de alta prioridad pueden anteponerse al tráfico de menor prioridad para reducir la interferencia. *Frame preemption* puede utilizarse en combinación con el *scheduling* de tráfico para reducir aún más la interferencia. *Frame preemption* se ha estandarizado en IEEE 802.3br (ya incorporado en IEEE 802.3-2018 [68]) y en IEEE 802.1Qbv (ya incluido en IEEE 802.1Q [67]).

4.6.1.5. Diferenciación de tramas

Para asegurar QoS para *streams* sensibles al retardo or *time-sensitive*, *brigdes* y *end stations* deben ser capaces de diferencia *streams* del tipo *time-sensitive* con respecto a otros flujos. El mecanismo fundamental para identificar y diferenciar *streams* del tipo *time-sensitive* de otro tipo de tráfico se presenta en IEEE 802.1Q. Para diferenciar el tráfico IEEE 802.1Q especifica una etiqueta denominada *Virtual Local Area Network* (VLAN), la cual se muestra en la Tabla 11. Esta etiqueta se añade a la cabecera de tramas estándares Ethernet, tal y como muestra la Tabla 12.

Tabla 11. Etiqueta VLAN definida en IEEE 802.1Q [65].

***Tag Protocol Identifier* (TPID)**	***Tag Control Information* (TCI)**		
0x8100 (16 bits)	*Priority Code Point* (PCP) (3 bits)	*Drop Eligible Indicator* (1 bit)	VLAN *Identifier* (VID) (12 bits)

Tabla 12. Campo IEEE 802.1Q VLAN en la trama Ethernet capa 2 [65].

Cabecera Ethernet capa 2					***Payload***	*CRC*
Preámbulo (8 octetos)	Dirección destino (DA) (6 octetos)	Dirección fuente (SA) (6 octetos)	802.1Q VLAN Tag (4 octetos)	EtherType (2 octetos)	(46-1500 octetos)	(4 octetos)

IEEE 802.1Q utiliza clases de tráfico para diferenciar el tráfico. El estándar permite hasta 8 clases de tráfico por puerto, con cada clase de tráfico asociado a una cola dedicada. El campo *Priority Code Point* (PCP) de la etiqueta VLAN determina la clase de tráfico (o lo que es lo mismo, la cola) para una trama Ethernet. De forma específica, los *bridges* se configuran para hacer corresponder

valores PCP a clases de tráfico. De esta manera, cuando llega una nueva trama, el *bridge* la asigna a la cola apropiada en base al valor del campo PCP y la correspondencia entre PCP a clase de tráfico configurada.

4.6.1.6. Redundancia

La redundancia permite alcanzar elevados niveles de fiabilidad requeridos por aplicaciones críticas. *Frame Replication and Elimination for Reliability* (FRER) definido en IEEE 802.1CB [69] es un mecanismo para replicar paquetes de un *stream* y que asegura que la red no se sobrecargue de manera innecesaria debido a la multiplicación de todas las tramas. Además, permite la detección de duplicados y la fusión de *streams*, haciendo la redundancia transparente para la aplicación y asegurando que la redundancia solo se realiza dentro de la red. La Figura 14 muestra el concepto de FRER. La función de replicado envía varias copias o réplicas de una trama o paquete con el mismo número de secuencia por dos o más caminos disjuntos. La función de eliminación elimina los paquetes que tienen el mismo número de secuencia que paquetes recibidos previamente. Las funciones de replicado y de eliminación pueden ejecutarse en los *end devices* o en varios *bridges* de la red.

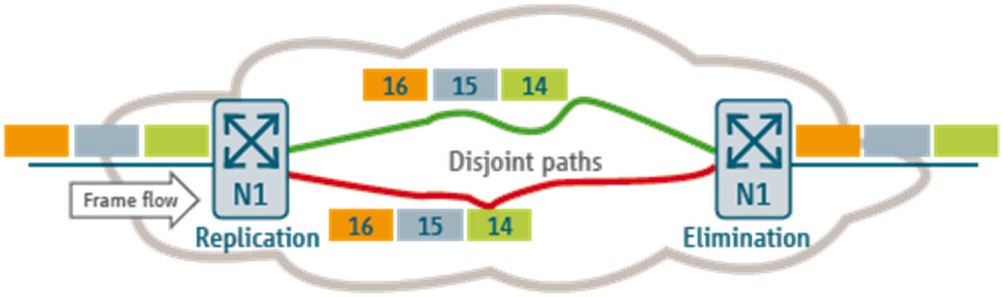


Figura 14. *Frame Replication and Elimination for Reliability* [66].

4.6.2. TSC en 5G

La capacidad de las redes 5G para dar soporte a comunicaciones URLLC (ver apartado 4.2) hace de la 5G una de las tecnologías candidatas para ofrecer comunicaciones inalámbricas deterministas y *time-sensitive*. Como se presentó en el apartado 4.2, la nueva interfaz radio NR definida en 5G incorpora nuevas funcionalidades para lograr bajas latencias en flujos de datos seleccionados. NR permite el uso de *slots* de transmisión de menor duración temporal e introduce el uso de *mini-slots* de manera que las transmisiones pueden iniciarse sin esperar al inicio de un nuevo *slot*, lo que permite reducir la latencia de las transmisiones. Para otorgar un acceso a los recursos radio más rápido y priorizado al tráfico URLLC, 5G NR introduce mecanismos de *preemption*, donde la transmisión de datos de URLLC puede interrumpir y adelantarse a transmisiones en curso que no sean de tipo URLLC. Además, NR acorta los tiempos de procesado, lo cual permite retransmisiones incluso dentro de límites de latencia cortos.

5G define modos de transmisión extra robustos para alcanzar una mayor fiabilidad en canales de control y de datos. La fiabilidad se puede incrementar mediante el uso de distintas técnicas, como la transmisión multi-antena, el uso de múltiples portadoras y la duplicación de paquetes a través de enlaces radio independientes. La sincronización temporal está integrada en los sistemas celulares 5G como una parte esencial de su operación, lo cual ha sido una práctica

común en generaciones anteriores, siendo por tanto una buena base para proporcionar sincronización para aplicaciones críticas en el tiempo o *time-critical*.

Además de estas características de las redes 5G, los sistemas 5G han sido y siguen siendo extendidos para soportar *time sensitive communications* o TSC tal y como se define en los estándares TSN IEEE 802.1. A continuación, se presenta la arquitectura propuesta para la integración de las redes 5G en redes TSN y dar soporte a comunicaciones TSC. También se presenta cómo se lleva a cabo la sincronización temporal de la red 5G con la red TSN, la información sobre los requisitos del tráfico TSN que se transmite a la red 5G, cómo se define el QoS *flow* para el tráfico TSN y las tablas de mapeo de QoS, cómo se debe indicar el retardo del *bridge* virtual 5G a la red TSN y el mecanismo de *Hold and Forward Buffering* definido.

4.6.2.1. Arquitectura para la integración de redes 5G con redes TSN

El 3GPP define en [26] la arquitectura de integración del sistema 5G en una red TSN. Según se propone en [26], el sistema 5G se integra con la red TSN externa como un *bridge* TSN tal y como se muestra en la Figura 15. El sistema 5G se considera un *bridge* TSN lógico, el cual integra la funcionalidad TSN *translator* para la interacción entre el sistema TSN y el sistema 5G tanto en el plano de datos como en el plano de control. La funcionalidad TSN *translator* consta de un TSN *translator* en el lado del dispositivo o *Device-side TSN translator* (DS-TT) y un TSN *translator* en el lado de la red o *Network-side TSN translator* (NW-TT). La función de red TSN AF[3] que forma parte del *Core Network* de 5G o 5G Core (5GC) proporciona la funcionalidad de *translator* en el plano de control, por ejemplo, para la interacción con el CNC de la red TSN.

La red TSN ve al sistema 5G como *bridges* TSN. Se considera un *bridge* TSN lógico por cada UPF tal y como se muestra en la Figura 15. Los procedimientos específicos del sistema 5G tanto en el 5GC como en la RAN, los enlaces inalámbricos, etc. son transparentes para la red TSN. Los puertos de entrada y de salida del *bridge* TSN lógico se implementan a través del DS-TT y del NW-TT. Los DS-TT y NW-TT pueden también implementar funciones de *hold and forward* para minimizar el jitter, y de *per-stream filtering and policing* (PSFP) tal y como se define en IEEE 802.1Q [67]. El NW-TT soporta *link layer connectivity Discovery and reporting* para descubrir los dispositivos Ethernet conectados al NW-TT según se define en IEEE 802.1AB [70]. El DS-TT también puede opcionalmente soportar la función de *link layer connectivity Discovery and reporting* para el descubrimiento de dispositivos Ethernet conectaods al DS-TT tal y como se define en IEEE 802.1AB [70]. Si el DS-TT no soporta esta funcionalidad, el NW-TT se encargará del descubrimiento de los dispositivos Ethernet conectados al DS-TT en su lugar.

La Figura 15 muestra un ejemplo con dos UEs con dos sesiones PDU que soportan dos *streams* TSN redundantes. También es posible despliegues con un único UE físico con dos sesiones PDU que utilizan *dual connectivity* en la RAN. En la Figura 15 se ilustra el caso en el que el sistema 5G conecta un nodo o *end device* a la red TSN, aunque también podría darse el caso en el que el sistema 5G interconecte TSN *bridges*. Todas las sesiones PDU que se conectan a la misma red TSN a través de un UPF específico se agrupan en un único *bridge* 5G. Si un UE establece multiples sesiones PDU que terminan en UPFs diferentes, el UE es representado por múltiples *bridges* 5G.

[3] La función AF interactúa con el *Core Network* para proporcionar servicios, como por ejemplo permitir que la aplicación influencie en el enrutado del tráfico, acceder a la función *Network Exposure Function* (NEF) o interactuar con el *Policy Control Function* (PCF).

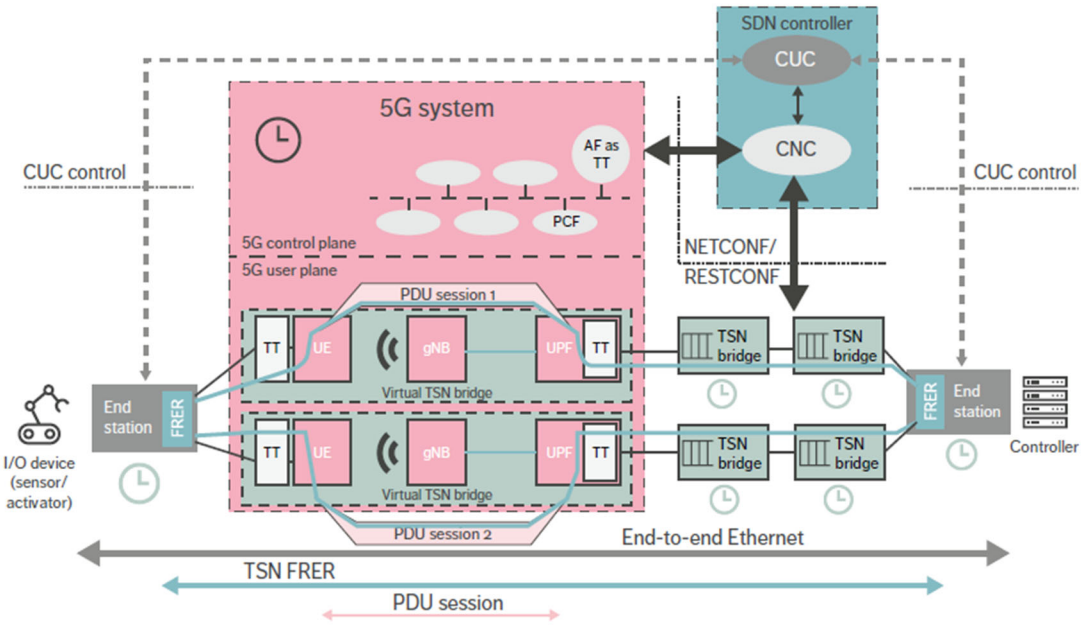


Figura 15. Integración del Sistema 5G en una red TSN [71].

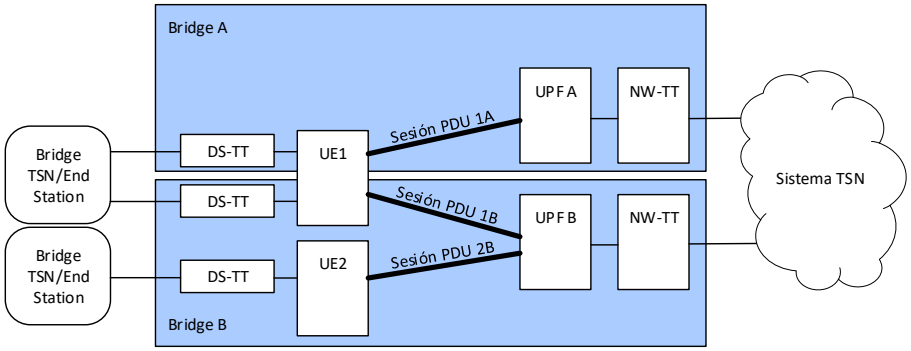


Figura 16. Representación de distintos *bridges* 5G por cada UPF conectado a la red TSN [26].

De los tres modelos de configuración de la red TSN (modelo de configuración totalmente centralizado, modelo de configuración de la red centralizado y de usuarios distribuido, y modelo totalmente distribuido – ver apartado 4.6.1.1), 3GPP Release 16 [26] y 17 [72] solo soportan el modelo de configuración totalmente centralizado.

4.6.2.2. Sincronización temporal con la red TSN

Como se ha presentado en el apartado anterior, los estándares 3GPP actuales consideran que el 5G se integra en una red TSN como un *bridge* TSN lógico. Según esta arquitectura de integración, solamente los *translators* TSN (DS-TT y NW-TT) en los bordes del sistema 5G soportan funciones y operaciones TSN y, por tanto, son los nodos que deben mantener la sincronización temporal con el reloj TSN. Dentro del sistema 5G, los UEs, gNBs, UPFs, DS-TT y NW-TT están sincronizados con el reloj interno 5G. Por tanto, en una red integrada TSN y 5G habrá dos sistemas de sincronización: la sincronización interna del sistema 5G y la sincronización en el dominio de la red TSN. Ambos procesos de sincronización pueden considerarse independientes.

Mientras en *releases* anteriores, los UEs de un sistema 5G se sincronizaban con el reloj del sistema identificando los límites de la trama, en 3GPP Release-16, la red puede informar al UE sobre el tiempo absoluto (UTC), de manera que el UE actualice su referencia de tiempo con mayor precisión con respecto al reloj del sistema 5G. Lograr esta sincronización precisa en el tiempo es un aspecto clave para la integración de las redes 5G en aplicaciones de la Industria 4.0, al mismo tiempo que para lograr la sincronización de los *end devices* para ejecutar de manera satisfactoria mecanismos y procedimientos de TSN. Para proporcionar la referencia temporal, la información de sincronización puede transmitirse en modo *broadcast* a todos los UEs, o la información se puede proporcionar mediante un mensaje RRC o en modo *unicast*. Por otro lado, en los estándares actuales también se define un método diferente para que los dispositivos puedan sincronizarse a un reloj *grandmaster* diferente al del sistema 5G [26]. Según este mecanismo, mensajes gPTP se envían de forma transparente a través del sistema 5G hacia los *end devices*. En los *end devices*, la información de tiempo debe actualizarse considerando los tiempos de residencia o el retardo experimentado en su transmisión a través del sistema 5G.

4.6.2.3. TSC *Assistance Information* (TSCAI)

Una de las características de los flujos de tráfico TSN es que son típicamente estrictamente periódicos y su tiempo de llegada es conocido. En este contexto, un habilitador clave de TSC en sistemas 5G es la introducción de información de asistencia o TSC *Assistance Information* (TSCAI) que proporciona información sobre la dirección (*uplink* or *downlink*), la periodicidad y el tiempo absoluto de offset de la llegada del tráfico para cada flujo de tráfico TSN (la Tabla 13 muestra la información contenida en la TSCAI). Esta información supone un gran beneficio para la configuración del sistema 5G ya que es posible reservar los recursos no solo para el tráfico TSN sino también para liberar recursos para otro tipo de tráfico. Conocer el patrón del tráfico TSN es de gran utilidad para la estación base o gNB para realizar un *scheduling* más eficiente para flujos de tráfico periódicos y deterministas utilizando los distintos tipos de *scheduling* definidos en 5G: *Configured Grant*, *Semi-Persistent Scheduling* o *Dynamic Grant*.

Tabla 13. Información de asistencia TSC o TSCAI [26].

Información	Descripción
Dirección del flujo de tráfico o *flow direction*	Dirección del flujo de tráfico TSC (*uplink* o *downlink*)
Periodicidad	Periodo de tiempo entre el inicio de dos *bursts* consecutivos.
Tiempo de llegada o *Burst Arrival time*	El instante de tiempo en el que el primer paquete del *burst* de datos llega al puerto de entrada de la RAN (en *downlink*) o al puerto de salida del UE (en *uplink*).

La RAN recibe la información TSCAI a través del SMF, por ejemplo, en el establecimiento del QoS *Flow*. Los parámetros TSCAI se establecen acorde con la información proporcionada por el TSN AF. El TSN AF debe ser capaz de identificar el puerto de entrada y por tanto la correspondiente sesión PDU. El TSN AF interactúa con el CNC de la red TSN por los objetos a gestionar con la función *per-stream filtering and policing* o PSFP implementada en los TSN *translators* (en el DS-TT y en el NW-TT). Cuando el CNC proporciona la información PSFP, el TSN AF puede extraer los parámetros que caracterizan el patrón de tráfico de la configuración PSFP; el TSN AF calcula el tiempo de llegada del *bust* de datos con referencia al puerto de entrada y la periodicidad, y obtiene la dirección del flujo de tráfico. En el Anexo I del 3GPP TS 23.501 [26] se describe el proceso seguido para determinar la información sobre el patrón de tráfico seguido por el TSN

AF. Una vez obtenida la información, el TSN AF envía estos parámetros al SMF (via el PCF) en contenedores TSC *Assistance Containers*. El TSN AF puede agregar varios TSN *streams* si estos pertenecen a la misma clase de tráfico, terminan en el mismo puerto de salida y tienen la misma periodicidad y el tiempo de llegada del *burst* es compatible. De esta manera, el TSN AF calculará los parámetros del tráfico para múltiples TSN *streams* de manera que estos TSN *streams* se agreguen en el mismo QoS *flow* (un QoS *flow* es la menor granularidad con la que se pueden distinguir flujos de datos con distintos perfiles de QoS y cada QoS *Flow* está asociado a un perfil de QoS). Una vez recibida la información por el SMF, éste deriva la información TSCAI para cada QoS *flow* y la envía a la RAN. El SMF es responsable de transformar el tiempo de llegada del *burst* y la periodicidad en base al reloj TSN con respecto al reloj 5G. En 3GPP TS 23.501 [26] es posible encontrar más detalles sobre el proceso que realiza el SMF para determinar los parámetros TSCAI.

4.6.2.4. TSC QoS *Flow*

5G gestiona la calidad de servicio o QoS de las comunicaciones a nivel de flujo o *flow level*, denominándolos QoS *flow*. Cada QoS *flow* se asocia con un perfil de QoS que define los parámetros que deben ser garantizados. Este perfil de QoS se define en el momento en el que se establece la sesión PDU y puede ser modificado en el proceso de modificación de la sesión PDU. El perfil de QoS viene definido por los siguientes parámetros:

- El valor 5QI (5G QoS *Indicator*). Es un valor escalar que se utiliza como referencia a características QoS pre-establecidas.
- *Allocation and Retention Priority* (ARP). Determina el nivel de prioridad en base al cual un nuevo flujo de QoS es aceptado o rechazado en la red. Además, también indica la preferencia o *pre-emption capability* de un flujo de datos a obtener recursos que habían sido asignados previamente a otro flujo de datos, y su vulnerabilidad o *pre-emption vulnerability* a perder los recursos asignados para admitir otro flujo de datos.
- Tasa de bit del flujo garantizada y máxima o Guaranteed and Maximum Flow Bit Rate. Estos parámetros se especifican sólo para flujos del tipo GBR.
- Notificaciones de control. Este parámetro solo se define para flujos del tipo GBR. Si este parámetro está activo, la RAN enviará notificaciones cuando los requisitos de QoS dejan de (o vuelven de nuevo a) satisfacerse para un flujo de QoS.

Como se ha introducido previamente, el valor 5QI es un valor escalar que hace referencia a un conjunto de características de QoS con valores predefinidas. Haciendo referencia a un valor de 5QI, no es necesario enviar el valor para cada uno de los parámetros o características de QoS englobadas dentro del 5QI. Las características que engloba el 5QI son las siguientes:

- Tipo de recurso. Que puede ser sin garantía de tasa de transmisión de bits (*non-Guaranteed Bit Rate* o non-GBR), con garantía de tasa de transmisión de bit (*Guaranteed Bit Rate* o GBR) o GBR crítico al retardo (delay-critical GBR).
- Nivel de prioridad. Indica la prioridad en el scheduling.
- *Packet Delay Budget* (PDB). Máxima latencia que debe experimentar un paquete entre el UE y el UPF que se conecta a la red de datos.
- *Packet Error Rate* (PER). Máximo PER a nivel de enlace.
- *Averaging Window*. Periodo sobre el cual se calcula el GFBR y el MFBR.

- Volumen máximo del *burst* de datos o *Maximum Data Burst Volume* (MDBV). Es la máxima cantidad de datos que la RAN debe servir en un periodo igual al PDB de la RAN.

Se pueden utilizar 5QI de diferentes tipos. En primer lugar, se definen una serie de 5QI estandarizados. Para los 5QI estandarizados existe un mapeo uno a uno a valores de características de QoS estandarizadas. Los 5QI estandarizados se presentan en la Tabla XX. También es posible utilizar valores de 5QI pre-configurados, para los cuales, los valores de las características de QoS han sido previamente configuradas en la RAN. Por último, también es posible utilizar valores de 5QI dinámicos, para los cuales es necesario enviar los valores para las distintas características de QoS como parte del perfil de QoS. Para los 5QI estandarizados y preconfigurados no es necesario enviar los valores de las características de QoS. Solamente se enviarían los valores de aquellas características a las que se le quiera modificar su valor.

Los flujos TSC usan recursos del tipo *delay-critical* Guaranteed Bit Rate (GBR) e información TSCAI. Para establecer el perfil de QoS, los flujos de QoS TSC pueden utilizar 5QIs estandarizados, preconfigurados o asignados dinámicamente (estos últimos requieren enviar los valores de las características de QoS como parte del perfil de QoS). De la lista de 5QI estandarizados por el 3GPP en [26], algunos de ellos pueden utilizarse para aplicaciones de automatización industrial, como por ejemplo, los 5QIs numerados de 82 a 85 que utilizan recursos del tipo *delay-critical* GBR. Sin embargo, puede resultar más adecuado utilizar 5QI preconfigurados por el operador de manera que se especifiquen los requisitos exactos de QoS de las aplicaciones industrial de interés [66], como por ejemplo para tipos de tráfico isócrono o cíclico. Para establecer los valores de varios parámetros del perfil de QoS es posible utilizar la información TSCAI. En cada periodo definido por el parámetro periodicidad, los flujos de QoS TSC deben transmitir solamente un *burst* de datos de un tamaño máximo definido (el máximo valor del *bust* de datos o MDBV) tal y como se indica en el perfil del 5QI con recursos del tipo *delay-critical* GBR. El 5QI también debe tener un valor de PDB que satisfaga las capacidades de retardo del *bridge*. La tasa de bit *Maximum Flow Bitrate* calculada por el TSN AF puede ser utilizada para establecer el GBR. En este caso, el valor MBR se configurará igual al GBR. El valor del 5QI se obtiene utilizando unas tablas de mapeo de QoS o QoS *mapping tables* configuradas en el TSN AF (que se describen en el apartado 4.6.2.7) y la información de QoS TSN.

4.6.2.5. Mecanismo *Hold and Forward Buffering*

Los puertos del DS-TT y del NW-TT implementan un mecanismo de *hold and forward* para realizar el *scheduling* del tráfico según se define en IEEE 802.1Q [67], de manera que el sistema 5G participa de manera transparente como un *bridge* en la red TSN. Esta función permite que el comportamiento del *bridge* lógico sea idéntico a realizar un *scheduling* del tráfico utilizando hasta 8 colas y de acuerdo a intervalos de tiempo establecidos en los que las distintas colas de transmisión están abiertas y cerradas.

4.6.2.6. Retardo en el *bridge* 5G

Para que el CNC de la red TSN pueda calcular el *scheduling* de todos los nodos de la red de manera que se satisfagan sus requisitos de latencia y determinismo, es necesario que el *bridge* lógico 5G indique, al igual que el resto de *bridges* de la red TSN, el retardo entre cada par de puertos de entrada y salida para cada clase de tráfico. Para ello, es necesario estimar los siguientes componentes del retardo:

- Tiempo de residencia UE-DS-TT o UE-DS-TT *Residence Time* que representa el tiempo requerido en el UE y el DS-TT para reenviar un paquete entre el UE y el puerto DS-TT. El

tiempo UE-DS-TT *Residence Time* se proporciona por el UE a la red en el momento del establecimiento de la sesión PDU.

- El mínimo y máximo retardo entre el UE y el UPF/NW-TT que conecta con la red de datos (incluyendo los tiempos de residencia o *residence times* en el UPF y el NW-TT) para cada clase de tráfico. Los retardos entre el UE y el UPF/NW-TT por tipo de tráfico son pre-configurados en el TSN AF.

Con estos componentes del delay, el TSN AF calcula los valores para *independentDelayMin* y *independentDelayMax*. Los valores *dependentDelayMin* y *dependentDelayMax* para el *bridge* 5G especifican el rango de tiempo para un octeto de una trama Ethernet desde el puerto de entrada hasta el puerto de salida, e incluye el tiempo requerido para recibir y almacenar cada octeto de la trama, el cual depende de la velocidad del enlace en el puerto de entrada.

El retardo del *bridge* 5G se calcula después del establecimiento de la sesión PDU, ya que éste depende del UE y UPFs que integran la ruta en el plano de usuario.

4.6.2.7. QoS *mapping tables*

Las *mapping tables* entre las clases de tráfico y los perfiles de QoS 5G se crean y utilizan para encontrar el perfil de QoS 5G para transferir tráfico TSN en una sesión PDU [26]. El TSN AF se pre-configura con una *mapping table*. La *mapping table* contiene clases de tráfico TSN, retardos pre-configurados de los *bridges* y niveles de prioridad. Una vez la sesión PDU se ha establecido y tras recibir la información sobre el tiempo de residencia UE-DS-TT, el TSN AF determina el retardo del *bridge* por pares de puertos y clase de tráfico en base al retardo pre-configurado y el tiempo de residencia UE-DS-TT. El TSN AF actualiza los retardos de los *bridges* y los reporta junto a otra información TSN relevante al CNC. Por otro lado, el CNC puede distribuir información y parámetros sobre el *scheduling* y PSFP al *bridge* 5G a través del TSN AF. Esta información puede ser mapeada a requisitos QoS por el TSN AF.

4.7. IA/ML e inteligencia colectiva

La digitalización de la industria está dando lugar a ecosistemas en los que un gran número de nodos (sensores, actuadores, robots, maquinaria, procesos, etc.) generan y consumen grandes cantidades de datos. La disponibilidad de estos datos, junto con el desarrollo de técnicas de IA y ML, está transformando las posibilidades de diseño y gestión de las redes de comunicaciones industriales de modelos estadísticos a un paradigma basado en datos.

La IA y el ML se puede clasificar en [73] aprendizaje supervisado, no supervisado y aprendizaje por refuerzo. En el aprendizaje supervisado, el modelo se aprende presentando muestras de entrada y sus salidas asociadas conocidas. En el aprendizaje no supervisado no hay etiquetas de salida, y el modelo aprende a clasificar muestras de la entrada. En el aprendizaje por refuerzo, un agente interactúa con un entorno y aprende a asignar cualquier entrada a una acción. La selección de la técnica IA o ML generalmente se basa en el estudio del caso particular, los requisitos algorítmicos y las escalas de tiempo de operación. Por ejemplo, el aprendizaje por refuerzo es más adecuado cuando la dinámica temporal del problema a resolver puede adaptarse a una curva de aprendizaje. Algunos ejemplos que utilizan el aprendizaje por refuerzo se presentan en [74] para abordar el control de admisión en RAN *slicing* y la asignación de recursos de radio, y en [75] para mejorar el rendimiento de latencia proporcionado por la programación rápida de recursos del enlace ascendente para comunicaciones masivas de tipo máquina. El aprendizaje profundo es una poderosa herramienta de IA/ML que analiza un

conjunto de datos de entrenamiento, identifica patrones complejos y los aplica a nuevos datos; Las redes neuronales están incluidas en el aprendizaje profundo. Una clara ventaja de las técnicas de IA/ML es que una vez que un modelo aprende las características de un sistema, puede realizar las tareas de manera eficiente utilizando algunos cálculos aritméticos básicos. Sin embargo, la cantidad de datos de entrenamiento aún es limitada; lo cual es un problema más pronunciado en la industria, donde los datos suelen no compartirse por cuestiones de confidencialidad y competitividad [73].

También debe tenerse en cuenta que la mayoría de las soluciones de IA/ML se basan en diseños centralizados donde un solo nodo ubicado en la red central o en la nube tiene acceso completo a todo el conjunto de datos. Esta solución centralizada puede no ser adecuada para satisfacer las crecientes demandas de algunos servicios emergentes en términos de baja latencia, baja complejidad, adaptabilidad e incluso preservación de la privacidad. Además, el tamaño de las redes industriales y el volumen de datos generados por la digitalización de la industria en constante crecimiento llevan a tareas de gestión y optimización de alta complejidad que pueden incluso superar la capacidad informática de un solo nodo. Por todo lo anterior, y dada la creciente capacidad computacional instalada en maquinaria, robots y procesos, la inteligencia se está trasladando (total o parcialmente) de la nube central a los recursos de computación del *edge* en las futuras redes inteligentes [76], lo que se conoce como *edge intelligence*. Mover toda o parte de la inteligencia a los nodos del *edge* locales reduce significativamente la latencia que supone la interacción con un nodo remoto en el *Cloud*, lo cual es crucial para las aplicaciones industriales sensibles al retardo [77]. Además, esto permite aliviar los problemas de privacidad y seguridad, muy importantes en la industria, ya que los datos se mantienen en las instalaciones de la fábrica.

Un requisito básico para adquirir inteligencia basada en el aprendizaje profundo es la disponibilidad de una gran cantidad de datos, lo cual puede no ser posible para los nodos aislados. Este hecho reclama la cooperación de los nodos del *edge* para lograr una Inteligencia Colectiva. La Inteligencia Colectiva es la capacidad de un grupo de nodos independientes e inteligentes para, colectivamente, tomar mejores decisiones (o de una manera más eficiente) que los nodos aislados. Para lograr esta Inteligencia Colectiva a nivel de *edge*, [78] propone el uso de técnicas de transferencia de conocimiento entre los nodos del *edge* y define un marco de intercambio de inteligencia para la gestión de redes vehiculares. Uno de los beneficios más importantes de esta Inteligencia Colectiva es el proceso de formación acelerado del aprendizaje profundo. Otra técnica que está ganando interés en la comunidad investigadora es el aprendizaje federado o *Federated Learning* (FL) [79]. FL es un algoritmo de aprendizaje automático distribuido que permite a los nodos del edge aprender de forma colaborativa un modelo de aprendizaje automático compartido sin intercambiar datos entre ellos. En este caso, hay un nodo central que distribuye un modelo FL entre los nodos del *edge*. Cada nodo del *edge* utiliza sus datos recopilados para entrenar su modelo FL local y envía el modelo FL local entrenado al nodo central. El nodo central integra los modelos FL locales para generar el modelo FL global y lo transmite a todos los nodos del *edge*. Sin embargo, confiar solo en la nube y el paradigma MEC no puede cumplir con los requisitos de latencia extremadamente baja de algunos servicios. Las propuestas recientes aprovechan las capacidades computacionales y de almacenamiento de los terminales de usuario y dispositivos finales actuales para ampliar las capacidades informáticas de borde de la red. Por ejemplo, [80] propone una IA distribuida de varios niveles para proporcionar una gestión inteligente global de las futuras redes de comunicaciones. En particular, se implementa un centro de IA global en la red central, los

centros de IA locales se ubican en estaciones base o servidores MEC, y los terminales de usuario o UEs también tiene herramientas de IA para el entrenamiento de datos en el dispositivo. En [81] se propone una arquitectura multinivel que permite la inteligencia de dispositivos, la inteligencia de borde y la inteligencia de nube para URLLC, y se aplica FL para mejorar la eficiencia del aprendizaje considerando la capacidad de computación limitada en los dispositivos finales y nodos de borde.

4.8. *Predictive* QoS y analítica de datos en 5G

Los mecanismos de predicción de QoS o *predictive* QoS, que fueron introducidos por 5GAA (*5G Automotive Association*), permiten a las redes móviles proporcionar notificaciones anticipadas sobre los cambios de QoS previstos a los nodos interesados para ajustar el comportamiento de las aplicaciones por adelantado [82]. La predicción de QoS permite un comportamiento de red proactivo que tiene claras ventajas respecto al comportamiento reactivo porque brinda a las aplicaciones tiempo suficiente para reaccionar y realizar las adaptaciones en consecuencia. Este mecanismo de *predictive* QoS utiliza el mensaje IQN (*in-advance QoS Notification*) para enviar la predicción de QoS a la entidad que solicita IQN (denominado IQN *consumer*) [82]. El IQN *consumer* utiliza esta información para adaptar el comportamiento de la aplicación. La función de predicción o PF (*Prediction Function*) recopila la información para generar las predicciones mediante, por ejemplo, algoritmos de aprendizaje automatizado. Los parámetros que se suelen monitorizar y predecir incluyen la latencia, fiabilidad, tasa de entrega de paquetes, tasa de datos, utilización de la red, variación de retardo y disponibilidad, entre otros.

Otras organizaciones industriales y de estandarización como 3GPP, 5G-ACIA y ETSI también tienen trabajos en curso para definir mecanismos para realizar la predicción de QoS. Por ejemplo, 3GPP en las Releases 15 y 16 ha especificado un marco para permitir la recopilación de datos y proporcionar análisis a los consumidores. En particular, 3GPP admite el ajuste de la aplicación en caso de que se reciba una notificación sobre QoS *Sustainability Analytics* del 5GS [83]. 3GPP Release 16 ubica la función de predicción en la función NWDAF (*Network Data Analytics Function*) y realiza predicciones en QoS *Sustainability Analytics*. En Release 16, no se permite la entrega de predicciones a la aplicación en el lado del UE, solo funciones conectadas al Core Network de la red (a través de la función de red AF).

Las soluciones actuales de predicción de QoS del 3GPP solo se basan en el análisis de la red troncal (es decir, estadísticas relacionadas con la movilidad del usuario, la carga, los patrones de comunicación, etc.) que recopila la NWDAF. Recientemente, el 3GPP ha introducido una función de análisis de datos de gestión (MDAF, *Management Data Analytics Function*), que facilita el rendimiento agregado y las estadísticas de fallos de red por región o celda. MDAF puede complementar NWDAF permitiendo análisis complejos, por ejemplo, predecir una QoS de UE en una ubicación futura combinando la movilidad con la predicción del rendimiento en células específicas [74]. Según estas predicciones, MDAF puede modificar la configuración de la red (por ejemplo, realizar un ajuste en los recursos de una *slice*) para garantizar que la red cumpla con el acuerdo de servicio establecido. Esto representa un primer esfuerzo del 3GPP para posibilitar decisiones de gestión de red predictivas que puedan mejorar la eficiencia en la utilización de recursos [84]. Una gestión de red más dinámica (por ejemplo, *scheduling* y control de interferencias) requeriría mejorar la analítica de datos mediante estadísticas radio y estadísticas de UE. El 3GPP aún no ha propuesto módulos equivalentes a NWDAF en la RAN, aunque existen iniciativas en curso lideradas por la alianza O-RAN [74]: el RNIB (*Radio Network Information*

Base) recopila información de carga de los diferentes flujos a nivel de RAN, el RIC (RAN *Intelligent Controller*) permite el control casi en tiempo real de los elementos/recursos de RAN. Además, 3GPP Release 17 introducirá mejoras para apoyar la recopilación y utilización de datos proporcionados por el UE en NWDAF con el fin de proporcionar información de entrada para generar información analítica [85].

4.9. *Mobile Edge Computing* (MEC)

4.9.1. Aplicación en la Industria 4.0

Edge Computing (EC) permite a los servicios explotar la proximidad de los dispositivos al proporcionar recursos computacionales más cercanos a los nodos finales, lo que permite una latencia ultra-baja y una comunicación de alta velocidad de datos. Al mismo tiempo, proporciona un medio para controlar y limitar la propagación de datos sensibles. *Multi-access Edge Computing* (MEC) es un estándar del ETSI para redes 5G, entre otras, para descargar el procesamiento y el almacenamiento de datos desde dispositivos móviles e IoT al borde (*edge*) de las redes móviles, en lugar de pasar todos los datos y cálculos a centros de datos o manipulándolos localmente [86]. La *Fog Computing* (FC) y la *Mist Computing* (MC) están estrechamente relacionadas tanto con EC como con CC, ya que pueden interpretarse como computación en la nube cerca del *edge* [87]. EC se refiere principalmente a la infraestructura computacional en el *edge*, FC se refiere principalmente a las arquitecturas lógicas que permiten servicios virtualizados distribuidos en la arquitectura del *edge* utilizando la capacidad de hardware de los nodos EC, y MC. FC generalmente cubre el almacenamiento en caché, el procesamiento de datos y los análisis que ocurren cerca de la fuente de los datos que mejoran el rendimiento en el *edge*, reducen la carga sobre los centros de datos y las redes troncales y mejoran la resistencia frente a problemas en la red [86]. La Figura 17 muestra una descripción general de cómo la computación en el *Cloud*, *Fog*, *Mist* y *Edge* encajan en una estructura en capas, y las operaciones típicas.

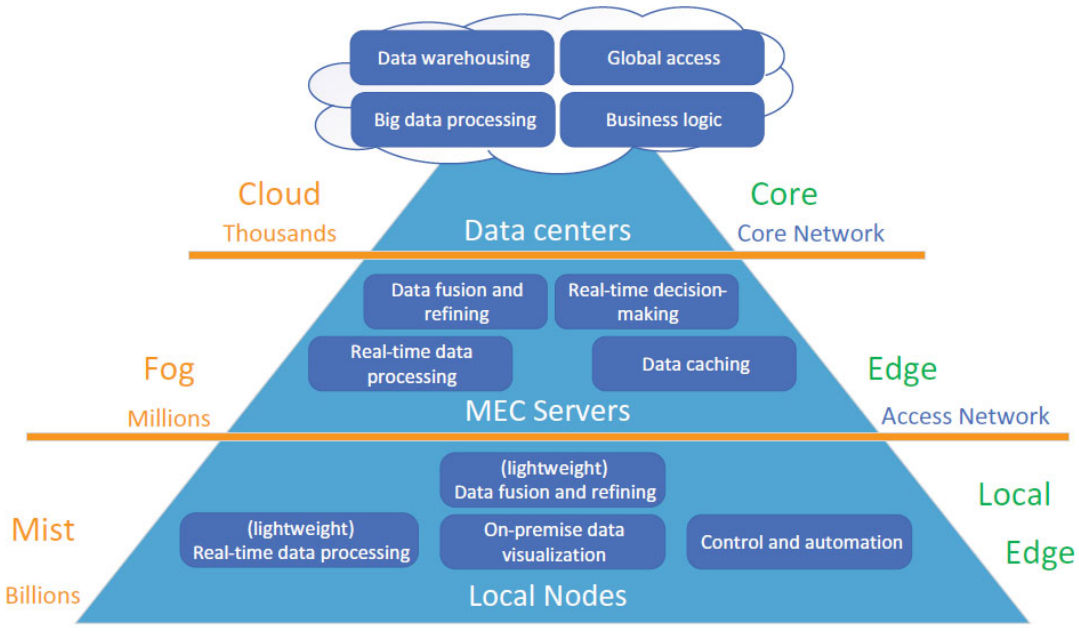


Figura 17. Visión simplificada de las capas *Cloud*, Fog, Mist y Edge [31].

EC es especialmente relevante en escenarios de aplicación donde existe la necesidad de computación y comunicación de baja latencia, alto ancho de banda y alta fiabilidad para permitir una toma de decisiones en tiempo real, inteligente y autónoma. La Industria 4.0 es una de las áreas de aplicación más importantes y, por lo tanto, es vital para definir los requisitos de los sistemas EC. En los últimos años, el paradigma CC se ha abierto camino en la industria de fabricación, abordando la necesidad de procesar una gran cantidad de datos que se originan en una gran cantidad de dispositivos sensores [31]. Ofrece recursos centralizados para realizar operaciones computacionalmente intensivas. Una nueva tendencia en el entorno industrial está representada por diferentes tipos de vehículos móviles, por ejemplo, vehículos aéreos no tripulados (UAV, *Unmanned Aerial Vehicle*) y vehículos guiados automatizados (AGV), que resuelven cooperativamente ciertas tareas. Algunas aplicaciones industriales típicas incluyen servicios avanzados, como robots de producción industrial, donde la baja latencia y la fiabilidad del EC es primordial. La cooperación de brazos robóticos en una línea de producción muestra un claro ejemplo de cooperación robótica en la fábrica. Aquí, EC respalda los sistemas de producción al descargar el análisis de datos y el procesamiento inteligente de datos en las proximidades de las fuentes de datos. Además, la monitorización inteligente del entorno industrial, los controles del sistema de red y las redes de actuadores y sensores inalámbricos masivos autoorganizados siguen atrayendo constantemente la atención de los fabricantes.

Las redes 5G permiten la comunicación de baja latencia, que es crucial en varios escenarios de la industria 4.0 que requieren funcionalidad en tiempo real [88]. En este contexto, MEC (y EC en general) es uno de los principales habilitadores de servicios inalámbricos con requisitos de alta fiabilidad y baja latencia, ya que minimiza la longitud de la ruta entre los nodos locales y los recursos de computación [89]-[92]. EC también ayuda a mejorar otros factores de QoS, ya que es más fácil eliminar y gestionar los cuellos de botella de rendimiento.

4.9.2. Estandarización

Tras la introducción de pequeños centros de datos en la nube cercanos a la fuente de datos denominada *Cloudlets*, se han generado las iniciativas *Open Edge Computing* (OEC) y *Open Fog Consortium* (OFC) para acelerar la estandarización y difusión de la tecnología MEC. A partir de entonces, se han creado múltiples comités, grupos de trabajo y organismos de normalización en todo el mundo. Según [93], los más importantes son los siguientes:

- Iniciativa *Multi-access Edge Computing* como un grupo de especificaciones de la industria dentro de ETSI
- MEC en redes 5G dentro del 3GPP
- *MEC system as service-oriented RAN* dentro de la Asociación de Estándares de Comunicaciones de China (CCSA)

La mayoría de los socios principales de todas las entidades de normalización provienen de la industria de las telecomunicaciones. Esto se refleja en las actividades centrales, así como en los objetivos de los grupos de trabajo. Todos los organismos realizan diferentes trabajos conceptuales, arquitectónicos y funcionales y tienen la intención de desarrollar un entorno abierto y estandarizado que permita la integración eficiente y sin problemas de aplicaciones de terceros a través de plataformas de múltiples proveedores. Sin embargo, la iniciativa ETSI MEC considera la gama más amplia de aplicaciones y escenarios de arquitectura entre todas las entidades de normalización. Esto se refleja en el hecho de que la plataforma EC no está vinculada a ninguna tecnología de acceso, lo que se refleja en el título de MEC. ETSI MEC presenta una

arquitectura de referencia y requisitos técnicos que permiten una ejecución eficiente y sin problemas, así como la interoperabilidad y la implementación de una amplia gama de escenarios EC que incluyen la industria 4.0.

3GPP define en TS 23.501 [94] un conjunto de habilitadores para soportar EC en 5G. El núcleo de la red 5G puede exponer la información y las capacidades de la red a una *Edge Computing Application Function*. La idea es manejar MEC como un 5G AF a través de estos habilitadores. Estos habilitadores se definen para proporcionar un soporte flexible para diferentes implementaciones de MEC y para respaldar MEC en caso de movilidad del usuario. Algunos de estos habilitadores (mencionados por ETSI en [95]) se presentan a continuación para redes SA 5G:

- Enrutamiento local y dirección del tráfico: el 5G *Core Network* proporciona los medios para seleccionar el tráfico que se enrutará a las aplicaciones en la red de datos local. Una sesión de PDU puede tener múltiples interfaces N6 hacia la red de datos. Se dice que las UPF que terminan estas interfaces admiten la funcionalidad de anclaje de sesión de PDU.
- La capacidad de una función de aplicación para influir en la (re) selección de la UPF y el enrutamiento del tráfico directamente a través de la PCF o indirectamente a través de la NEF, según las políticas del operador.
- Los modos de Continuidad de Sesión y Servicio (SSC) para diferentes escenarios de movilidad de aplicaciones y UE.
- Soporte de Red de datos de área local (LADN) por el 5G *Core Network* al proporcionar soporte para conectarse a LADN en un área determinada donde se implementan las aplicaciones. El acceso a una LADN solo está disponible en un área de servicio LADN específica, definida como un conjunto de áreas de seguimiento en la PLMN de servicio del UE. LADN es un servicio proporcionado por la PLMN de servicio del UE.

La Figura 18 muestra un despliegue de MEC integrado en una red 5G [95]. El lado derecho de la figura muestra la arquitectura de un sistema MEC genérico. En un despliegue integrado de MEC en una red 5G, algunas de las entidades funcionales de MEC interactúan con las funciones de red del núcleo 5G. La UPF tiene un papel clave en esta integración. Las UPF pueden verse como un plano de datos distribuido y configurable desde la perspectiva del sistema MEC. En algunas implementaciones específicas, la UPF local puede incluso ser parte de la implementación de MEC como se muestra en la Figura 18.

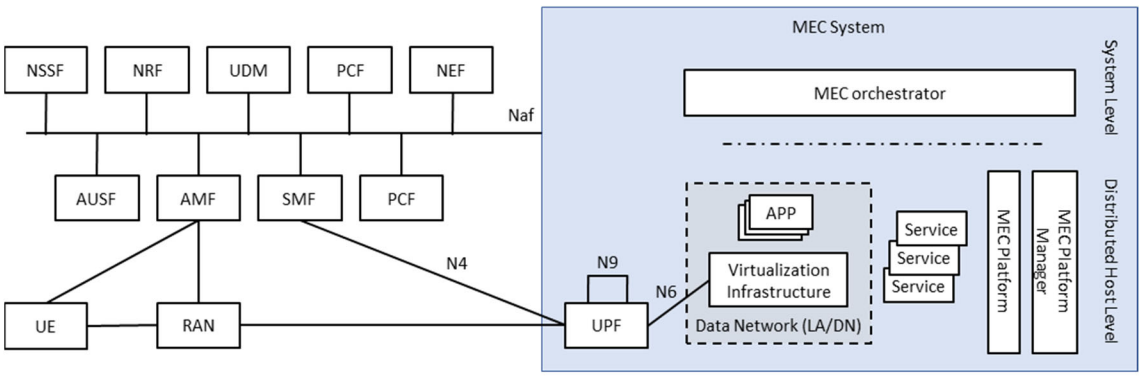


Figura 18. Integración de MEC en red 5G [95].

El orquestador MEC en el lado derecho de la Figura 18 es una entidad funcional a nivel de sistema MEC. Está a cargo de administrar los recursos de cálculo y el funcionamiento de los hosts MEC, y se encarga de la orquestación de aplicaciones MEC que se ejecutan en hosts MEC. Para la red 5G, el orquestador MEC actúa como un AF y puede interactuar con el NEF o, en algunos escenarios, directamente con las *Network Functions* del sistema 5G objetivo. En el nivel de host MEC, la plataforma MEC puede interactuar con las *Network Functions* de 5G en el papel de un AF. El host MEC se implementa con mayor frecuencia en una red de datos. El NEF es una función del núcleo de la red que generalmente se implementa de forma centralizada junto con *Network Functions* similares. Sin embargo, también se puede implementar una instancia de NEF en el borde para permitir el acceso al servicio de baja latencia y alto rendimiento desde un host MEC.

Los hosts MEC se implementan en la red de datos central o de *edge*. Es la UPF la que se encarga de dirigir el tráfico del plano del usuario hacia las aplicaciones MEC específicas en la red de datos. El orquestador de MEC administra y organiza el funcionamiento de los hosts y las aplicaciones de MEC. Puede decidir dinámicamente dónde implementar las aplicaciones MEC. En términos de implementación física, las ubicaciones de los hosts de MEC se pueden decidir en función de varios requisitos operativos, de rendimiento o relacionados con la seguridad. MEC se puede implementar de manera flexible en diferentes ubicaciones, desde cerca del NodoB hasta la red de datos central, pero algo común a las diferentes opciones de implementación es que la UPF se implementa y se utiliza para dirigir el tráfico entre las aplicaciones MEC específicas y la red. Algunos ejemplos de opciones viables para la ubicación física de MEC son [95]:

1. MEC y la UPF local se ubican en el Nodo B.
2. MEC colocado con un nodo de transmisión, posiblemente con una UPF local.
3. MEC y la UPF local colocados con un punto de agregación de red.
4. MEC colocado con las funciones *Core Network* (es decir, en el mismo centro de datos).

5. Retos en el desarrollo de las redes 5G para dar soporte a la digitalización de la industria

5.1. Gestión proactiva de las redes 5G *and Beyond* basada en IA

Como se ha presentado en los apartados anteriores (principalmente apartados 2 y 3.1), las fábricas del futuro serán entornos altamente flexibles, dinámicos y con capacidad de adaptar su configuración a las diferentes y variantes demandas del mercado y clientes. El dinamismo del entorno así como la presencia de nodos móviles (robots móviles, maquinaria y operadores) demandan por tanto redes de comunicaciones que sean capaces de adaptarse de manera dinámica y eficiente a las variaciones de la demanda de comunicaciones a la vez que garantizan de forma ininterrumpida los exigentes requisitos en términos de determinismo, fiabilidad y robustez demandados por las aplicaciones industriales. Como se ha presentado a lo largo de este informe, las redes 5G y su futura evolución (las redes *Beyond* 5G) tienen un gran potencial para dar soporte a este tipo de entornos. Sin embargo, los estrictos requisitos en términos de bajas latencias, alta fiabilidad y determinismo requieren un cambio de paradigma en la gestión de estas redes. Las técnicas y mecanismos de gestión de las redes inalámbricas industriales utilizados hasta ahora reaccionan cuando se detecta un evento (caída de QoS o de la calidad del canal radio, variaciones en la demanda de tráfico, etc.). Sin embargo, en el entorno industrial tan altamente exigente, es necesario anticiparse a estos eventos para poder garantizar sin interrupciones los requisitos de latencia, fiabilidad y determinismo demandados. Por tanto, es necesario diseñar técnicas de gestión de la red que de manera proactiva se antepongan a estos eventos y adapten la configuración de la red de manera adecuada. Para realizar esta gestión proactiva de las redes 5G *and Beyond* es posible aprovechar el rico ecosistema de datos generado por la digitalización de la industria para extraer información útil del comportamiento del sistema y del rendimiento de las comunicaciones. La disponibilidad de estas cantidades crecientes de datos generados por la digitalización de la industria brinda la posibilidad de utilizar técnicas de IA y ML que exploten dichos datos para predecir cambios en el entorno, la demanda de tráfico o incluso en el rendimiento de las comunicaciones 5G, y diseñar técnicas proactivas y predictivas de gestión de la red 5G industrial basadas en datos.

La disponibilidad de datos y los avances en sistemas de computación en el Edge (ver apartado 4.9) hacen también posible la aplicación de técnicas de IA y ML para el propio diseño de técnicas de gestión proactiva que exploten los datos generados en las fábricas digitalizadas. De esta manera, es posible pasar de técnicas de gestión automática basadas en lógica codificada a técnicas de gestión autónoma que aprenden y se adaptan de forma constante y automática a los cambios en la red y el contexto, y permiten una gestión más eficiente de los recursos.

La gestión proactiva y predictiva basada en IA puede ser aplicada para la gestión de distintos aspectos de las redes 5G *and Beyond*. Un campo de aplicación clave de esta gestión proactiva es la gestión de los *slices* de red o *network slices*. Como se ha presentado en el apartado 4.4, las propuestas actuales de RAN Slicing se han centrado principalmente en realizar una asignación de recursos a los *slices* a largo plazo que se mantiene a lo largo del ciclo de vida del *slice*. Sin embargo, realizar una asignación de recursos fija puede resultar en una baja utilización de los recursos o en el incumplimiento de los requisitos de los servicios debido a fluctuaciones del estado de la red, de la calidad de los enlaces o del tráfico a soportar. Para maximizar la eficiencia de los recursos de las redes, la asignación de recursos a los *slices* debe adaptarse de manera dinámica en función de las características espacio-temporales del tráfico. Pero además,

considerando los exigentes requisitos de las aplicaciones industriales, esta gestión dinámica de los *slices* debe realizarse de manera proactiva adaptando y optimizando los recursos radio asignados a cada *slice* en base a un conocimiento anticipado. Varios trabajos han propuesto *frameworks* generales para realizar esta gestión dinámica durante el tiempo de vida de la *slice* [49][56]. Algunos de los *frameworks* propuestos también incorporan el uso de esquemas de IA y ML para la toma de decisiones de gestión eficientes. Sin embargo, todavía existen múltiples desafíos que deben abordarse para una definición adecuada de *slices* dinámicas con el objetivo de maximizar la eficiencia de las redes, como el compartir y aislar recursos entre distintos *slices* para maximizar la eficiencia en el uso de los recursos o la selección del período apropiado para actualizar la asignación de recursos a las *slices*.

Por último, es importante destacar que la disponibilidad de bancos de datos o *datasets* es un requisito para el entrenamiento y validación de mecanismos basados en IA. Sin embargo, la disponibilidad de estos *datasets* en la comunidad es nula dado el carácter confidencial del diseño y configuración de sistemas de producción en plantas industriales reales, lo cual es un factor limitante para el desarrollo de soluciones basadas en inteligencia artificial. La creación de estos *datasets* es otro reto actual y que ha sido abordado en este proyecto (bajo el objetivo 2).

5.2. Gestión conjunta de la red 5G y la inteligencia colectiva

La futura industria digitalizada se caracterizará por el uso generalizado de la IA desde la nube hasta el Edge o más allá del Edge para el control eficiente y óptimo de nodos conectados, como robots, maquinaria y sistemas de producción completos. Como se presentó en el apartado 4.7, la Inteligencia Colectiva tiene el potencial de mejorar la eficiencia del aprendizaje en técnicas de IA y ML [78][79]. Sin embargo, la transferencia de conocimiento entre los distintos nodos inteligentes y distribuidos requiere la implementación de métodos de comunicación adecuados entre los diferentes nodos que faciliten la transmisión eficiente de inteligencia. Como se presenta en [96], el ancho de banda limitado se convierte en el principal cuello de botella para la Inteligencia Colectiva cuando integra nodos móviles. En este contexto, la red inalámbrica también debe diseñarse para proporcionar la infraestructura necesaria para dar soporte de manera eficiente y permitir una integración transparente de la inteligencia colectiva. Pero además, esta inteligencia colectiva también debe diseñarse considerando la infraestructura de comunicación que le dará soporte. Por tanto, es todavía un desafío llevar a cabo una planificación y gestión conjunta de la red 5G *and Beyond* y la inteligencia distribuida y colectiva considerando las dependencias entre comunicaciones e inteligencia.

5.3. Integración eficiente de redes 5G y TSN

Como se ha presentado en el apartado 4.6, el 3GPP ha definido el *framework* y arquitectura de red para la integración de las tecnologías 5G y TSN en una red de convergencia que soporte los requisitos de determinismo y flexibilidad, entre otros, demandados por la transformación digital de la industria. Sin embargo, hay aspectos que requieren seguir siendo estudiados y mejorados para conseguir una integración real y eficiente que cubra todos los escenarios posibles en la industria. Por este motivo, el 3GPP tiene varios *Work Items* activos en Release 17 en los que sigue estudiando la integración de 5G y TSN para dar soporte a la digitalización de la industria. Estos *Work Items* son:

- *Support of Enhanced Industrial IIoT* (IIoT)
- *Study on enhanced support of Industrial IoT* (FS_IIoT)
- *Enhanced Industrial Internet of Things (IoT) and ultra-reliable and low latency communication (URLLC) support for NR* (NR_IIOT_URLLC_enh)

Los estándares del 3GPP (hasta Release 16) consideran la integración del sistema 5G como un *bridge* virtual o lógico dentro de la red TSN. En este modelo de *bridge* virtual, la comunicación TSC siempre se realiza entre un UE y un nodo en el *backbone* de la red 5G, es decir, la comunicación se realiza entre un UE y el UPF que se conecta a una red de datos (ver Figura 15 en el apartado 4.6.2.1). Sin embargo, no se ha considerado el caso de comunicaciones UE a UE tal y como se representa en la Figura 19. Este es uno de los aspectos clave a estudiar identificados en [97]. Para este tipo de comunicaciones TSC de UE a UE, se debe estudiar cómo identifica el sistema 5G el par de UEs/DS-TT entre los que se realiza la comunicación TSC o cómo calcula el sistema 5G el retardo del *bridge* virtual entre los dos UEs.

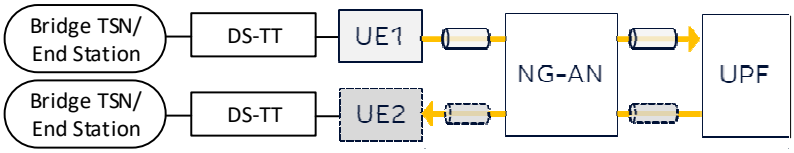


Figura 19. Comunicación TSC de UE a UE [97].

El *framework* de integración de las redes 5G y TSN definido por el 3GPP considera que las redes TSN siguen un modelo de configuración de la red totalmente centralizado. Los modelos centralizados pueden presentar problemas de escalabilidad. Como se presentó en el apartado 4.6.1.1, el estándar TSN 802.1Qcc también propone un modelo de configuración totalmente distribuido en el que los mensajes de configuración se distribuyen a través de los distintos caminos entre el *Talker* y el *Listener*. En este contexto, sería interesante considerar y estudiar las posibilidades de integración de 5G en una red TSN con modelo de configuración totalmente distribuido. Con este modelo de configuración, el sistema 5G debería ser capaz de leer la información de configuración que se transmite en los mensajes de registro TSN y modificar estos mensajes de igual manera que un *bridge* TSN añadiendo información sobre las capacidades del sistema 5G [97].

Además, todavía quedan retos por abordar considerando el actual *framework* de integración definido por el 3GPP en Release 16. Por ejemplo, un desafío importante es la consideración de nodos móviles. En este caso, el retardo que se puede garantizar al nodo móvil en el sistema 5G (el retardo del *bridge* virtual 5G) variará según la posición actual en la que se encuentre el nodo móvil (al cambiar las condiciones de propagación y por tanto la calidad del enlace). Estas variaciones en las capacidades del *bridge* virtual 5G deberían notificarse de manera dinámica al CNC de la red TSN. En base a las nuevas condiciones del *bridge* virtual 5G, la configuración de la red TSN debería adaptarse para seguir cumpliendo los requisitos del tráfico TSN.

Otro desafío es la coordinación de los *schedulers* de las redes 5G y TSN. El estándar define las herramientas necesarias para realizar esta coordinación, pero queda fuera de su ámbito de trabajo definir cómo realizarla. Por ejemplo, pueden existir inconsistencias entre la periodicidad demandada por el tráfico TSN y las que 5G puede soportar (cuando la periodicidad demandada no es múltiplo de las que 5G puede gestionar). Para dar solución a este problema, 5G NR permite asignar a un mismo usuario diferentes recursos de manera periódica, es decir, distintas configuraciones de *Configured Grant,* aunque solamente una de ellas puede estar activa en cada

momento. De esta manera, el UE podrá elegir la configuración a utilizar para adaptarse mejor al periodo requerido por el tráfico TSN. Sin embargo, cuándo activar/desactivar cada configuración para lograr un uso eficiente de los recursos a la vez que se garantizan los requisitos del tráfico TSN necesita todavía ser estudiado.

Por último, la mayoría de los estudios siempre consideran tráfico TSN periódico por ser el mayoritario en el entorno industrial. Sin embargo, el tráfico aperiódico también es un tipo de tráfico presente en el entorno industrial y además importante ya que este está relacionado con alarmas o eventos que requieren una comunicación determinista con requisitos de latencia muy bajos. Por tanto, dar soporte a este tipo de tráfico, para el cual no se puede prever su llegada, es un desafío importante para las redes 5G y TSN actuales, las cuales están basadas en el conocimiento preciso del patrón del tráfico y una planificación previa en base al mismo.

5.4. Implementación de MEC en el entorno industrial

La fiabilidad ante problemas en la infraestructura de red y de computación es una de las áreas de investigación más importantes de EC/MEC [98]. En la Industria 4.0, el procesamiento de los datos de los sensores, al menos cierto grado de la lógica de toma de decisiones y el control de los actuadores, podría realizarse localmente, porque la conexión con la red de acceso ocasionalmente puede volverse poco fiable o tener un rendimiento bajo [92]. Por lo tanto, es clave incorporar algo de capacidad de MEC también dentro de los clústeres locales. Los desafíos relacionados con la fiabilidad de los sistemas MEC se refieren a la capacidad de adaptarse a situaciones dinámicamente cambiantes, relacionadas, por ejemplo, con la movilidad, fallos de red, perturbaciones y fallos de hardware. El sistema MEC debe gestionar, analizar y optimizar automáticamente su funcionamiento, incluida la ubicación de las tareas de gestión de datos y computacionales, en función de la situación actual y los cambios previstos. Con respecto a la disponibilidad, las preguntas críticas son dónde están ubicados los componentes del sistema y quiénes/dónde están los usuarios. La disponibilidad se convierte en un problema particular en los casos en que los diferentes actores de los servicios están distribuidos lógica y geográficamente.

En la Industria 4.0, se recopila información de un gran número de dispositivos conectados [99]. Además, dispositivos como sensores avanzados, los sistemas de control, la transmisión de vídeo de vigilancia y los dispositivos de captura de imágenes fijas son capaces de producir enormes cantidades de datos para procesar [100]. En los sistemas tradicionales, todo este procesamiento de datos y la lógica de toma de decisiones relacionada se ha manejado en los centros de datos, lo que se está volviendo problemático desde el punto de vista de la escalabilidad, el rendimiento y la fiabilidad. En este contexto, los sistemas MEC pueden ayudar al proporcionar capacidad computacional cerca de las fuentes de los datos, lo que permite varias funciones de preprocesamiento, refinamiento y análisis de datos para reducir la cantidad de datos que se envían a los servidores en la nube y, por lo tanto, reducir la carga infligida a las redes centrales y centros de datos. El importante reto investigador en esta área es desarrollar algoritmos inteligentes para decidir en qué nivel administrar las diferentes funciones y priorizar las tareas cuando los recursos limitados no permiten soluciones óptimas a nivel global

6. Referencias

[1] *Made in Europe-The manufacturing partnership in Horizon Europe (2021 – 2027)*, European Factories of the Future Research Association (EFFRA).

[2] *Made in Europe-The Future of European Manufacturing?*, European Commission Directorate-General for Research and Innovation, Febrero 2020.

[3] *Industry 4.0 Whitepaper*, Industry4.E Lighthouse, Junio 2021.

[4] H. X. Nguyen, R. Trestian, D. To and M. Tatipamula, "Digital Twin for 5G and Beyond", *IEEE Communications Magazine*, vol. 59, no. 2, pp. 10-15, February 2021.

[5] *Using Digital Twins to Integrate 5G into Production Networks*, 5G-ACIA Whitepaper, 2021.

[6] Our view on the Evolution of 5G towards 6G, 5G-ACIA Position paper, 2021.

[7] J. L. Carlson et al., "Resilience: Theory and application", Argonne National Lab., Argonne, IL, Tech. Rep. ANL/DIS-12-1, 1044521, Feb. 2012. Disponible en http://www.osti.gov/servlets/purl/1044521/ (último acceso 19/11/2021).

[8] O. Lucia *et al*., "Emerging Trends in Industrial Electronics: A Cross-Disciplinary View", *IEEE Industrial Electronics Magazine*, vol. 15, no. 1, pp. 127-139, March 2021.

[9] K. E. Benson, G. Wang, N. Venkatasubramanian, Y.-J. Kim, "Ride: A resilient IoT data exchange middleware leveraging SDN and edge cloud resources", in *Proc. 2018 IEEE/ACM 3rd Int. Conf. Internet-of-Things Design Implementation IoTDI*, Orlando, FL, pp. 72–83.

[10] W.-P. Chen, A.-H. Tsai, and C.-H. Tsai, "Smart traffic offloading with mobile edge computing for disaster-resilient communication networks", *J. Netw. Syst. Manage.*, vol. 27, no. 2, pp. 463–488, Apr. 2019.

[11] E. Gourdin, D. Medhi, and A. Pattavina, "Design of reliable communication networks," *Ann. Telecommun.*, vol. 73, no. 1–2, pp. 1–3, Feb. 2018.

[12] K. Montgomery et al, "Wireless User Requirements for the Factory Workcell", *NIST Advanced Manufacturing Series 300-8*, R1, Nov. 2020.

[13] 3GPP; Technical Specification Group Services and System Aspects; Study on Communication for Automation in Vertical Domains (Release 16), TR 22.804 v16.3.0, julio 2020.

[14] 3GPP; Technical Specification Group Services and System Aspects; Service requirements for cyber-physical control applications in vertical domains; Stage 1 (Release 17), TS 22.104 v17.7.0, sept. 2021.

[15] 3GPP; Technical Specification Group Services and System Aspects; Service requirements for the 5G system; Stage 1 (Release 17), TS 22.261 v17.8.0, sept. 2021.

[16] 3GPP; Technical Specification Group Services and System Aspects; Study on enhancements for cyber-physical control applications in vertical domains; Stage 1 (Release 17), TR 22.832 v17.4.0, marzo 2021.

[17] *Key 5G Use Cases and Requirements from the Viewpoint of Operational Technology Providers*, 5G-ACIA Whitepaper, Diciembre 2020.

[18] Jan García-Morales, M. Carmen Lucas-Estañ, and Javier Gozalvez, "Latency-Sensitive 5G RAN Slicing for Industry 4.0", *IEEE Access*, vol. 7, pp. 143139-143159, 2019.

[19] ISA, "ISA-TR100.00.03-2011 - Wireless User Requirements for Factory Automation", 2011.

[20] ETSI, “Reconfigurable Radio Systems (RRS); Feasibility study on temporary spectrum access for local high-quality wireless networks”, ETSI TR 103 588 V1.1.1, Feb. 2018.

[21] P. Schulz et al., “Latency Critical IoT Applications in 5G: Perspective on the Design of Radio Interface and Network Architecture”, *IEEE Communications Magazine*, vol. 55, no. 2, pp. 70-78, February 2017.

[22] S. Doolani, et.al, “A Review of Extended Reality (XR) Technologies for Manufacturing Training”, *Technologies* 8, no. 4, pp 77.

[23] *6G The Next Hyper-Connected Experience for All*, Samsung Research, Julio 2020.

[24] S. Castellanos, *Digital Twins Concept Gains Traction Among Enterprises*, Sept. 2018. Disponible en https://blogs.wsj.com/cio/2018/09/12/digital-twins-concept-gains-tractionamong-enterprises/.

[25] M. Raza, P. M. Kumar, D. V. Hung, W. Davis, H. Nguyen and R. Trestian, “A Digital Twin Framework for Industry 4.0 Enabling Next-Gen Manufacturing”, in *Proc. of the 9th International Conference on Industrial Technology and Management (ICITM)*, 2020, pp. 73-77.

[26] 3GPP; Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 16), TS 23.501 v16.10.0, sept. 2021.

[27] 3GPP TS 38.211 V16.1.0, “Technical Specification Group Radio Access Network; NR; Physical channels and modulation (Release 16)”, March 2020.

[28] K. Arora, J. Singh, Y. Singh Randhawa, “A survey on channel coding techniques for 5G wireless networks”, *Telecommunication Systems*, vol. 73, pp. 637-663, 2020.

[29] 3GPP TS 38.214 V16.1.0, “Technical Specification Group Radio Access Network; NR; Physical layer procedures for data (Release 16)”, March 2020.

[30] 3GPP TS 38.300 V16.4.0, “Technical Specification Group Radio Access Network; NR; NR and NG-RAN Overall Description; Stage 2 (Release 16)”, Dec. 2020.

[31] N.H. Mahmood, N. Marchenko, M. Gidlund, P. Popovski, *Wireless Networks and Industrial IoT. Applications, Challenges and Enablers*, Springer, 2021.

[32] 3GPP TS 38.331 V16.3.1, “Technical Specification Group Radio Access Network; NR; Radio Resource Control (RRC) protocol specification (Release 16)”, Jan. 2021.

[33] M Centenaro, et al., “System-level study of data duplication enhancements for 5G downlink URLLC”, *IEEE Access,* vol. 8, pp. 565-578, 2020.

[34] H. Holma, A. Toskala, T. Nakamura, *5G Technology. 3GPP New Radio*, Wiley, 2020.

[35] 3GPP TS 38.211 V16.1.0, “Technical Specification Group Radio Access Network; NR; Base Station (BS) radio transmission and reception (Release 17)”, Sept. 2021.

[36] *Making 5G a reality: Addressing the strong mobile broadband demand in 2019 & beyond*, Qualcom & Nokia, Sept. 2017.

[37] *Leveraging the potential of 5G millimiterwave*, Ericsson, 2021.

[38] *What, Why and How: the Power of 5G Carrier Aggregation*, Ericsson, Junio 2021. Disponible en https://www.ericsson.com/en/blog/2021/6/what-why-how-5g-carrier-aggregation (último acceso 16/11/2021).

[39] *Making 5G NR a commercial reality*, Qualcomm, Dic. 2017.

[40] F J. Ordonez-Lucena, et al., “Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges”, IEEE Communications Magazine, vol. 55, Issue 5, pp. 80-87, May 2017.

[41] P. Rost et al., “Mobile network architecture evolution toward 5G,” *IEEE Commununications Magazine*, vol. 54, no. 5, May 2016, pp. 84–91.

[42] Fibocom, “5G Network Slicing: Empowering Vertical Industries”, White Paper, Sept. 2021. Online: https://www.fibocom.com/en/Whitepaper/5g-network-slicing-empowering-vertical-industries.html

[43] 3GPP; Technical Specification Group Services and System Aspects; Service requirements for the 5G system (5GS); Stage 1 (Release 16); TS 22.261 v16.5.0, 2018.

[44] I. Vilà, J. Pérez-Romero, O. Sallent and A. Umbert, "Characterization of Radio Access Network Slicing Scenarios With 5G QoS Provisioning," *IEEE Access*, vol. 8, pp. 51414-51430, 2020.

[45] P. Korrai, E. Lagunas, S. K. Sharma, S. Chatzinotas, A. Bandi and B. Ottersten, "A RAN Resource Slicing Mechanism for Multiplexing of eMBB and URLLC Services in OFDMA Based 5G Wireless Networks," *IEEE Access*, vol. 8, pp. 45674-45688, 2020.

[46] P. Muñoz, ñ. Adamuz-Hinojosa, J. Navarro-Ortiz, O. Sallent and J. Pérez-Romero, "Radio Access Network Slicing Strategies at Spectrum Planning Level in 5G and Beyond," *IEEE Access*, vol. 8, pp. 79604-79618, 2020.

[47] L. Feng, Y. Zi, W. Li, F. Zhou, P. Yu and M. Kadoch, "Dynamic Resource Allocation With RAN Slicing and Scheduling for uRLLC and eMBB Hybrid Services," *IEEE Access*, vol. 8, pp. 34538-34551, 2020.

[48] A. E. Kalør, R. Guillaume, J. J. Nielsen, A. Mueller and P. Popovski, "Network Slicing in Industry 4.0 Applications: Abstraction Methods and End-to-End Analysis," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 12, pp. 5419-5427, Dec. 2018.

[49] D. Bega, M. Gramaglia, A. Garcia-Saavedra, M. Fiore, A. Banchs and X. Costa-Perez, "Network Slicing Meets Artificial Intelligence: An AI-Based Framework for Slice Management," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 32-38, June 2020.

[50] K.B. Letaief, W. Chen, Y. Shi, J. Zhang, Y.J.A. Zhang, “The roadmap to 6G: AI empowered wireless networks”, *IEEE Communications Magazine*, vol. 57, pp. 84–90, 2019.

[51] R.M. Sohaib, O. Onireti, Y. Sambo, M.A. Imran, “Network Slicing for Beyond 5G Systems: An Overview of the Smart Port Use Case”, *Electronics*, 10, 1090, 2021.

[52] D. Bega, M. Gramaglia, A. Banchs, V. Sciancalepore, K. Samdanis, X. Costa-Perez, “Optimising 5G infrastructure markets: The business of network slicing”, *Proceedings of the IEEE INFOCOM 2017—IEEE Conference on Computer Communications*, Atlanta, GA, USA, pp. 1–9, 1–4 May 2017.

[53] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, X. Costa-Perez, “DeepCog: Cognitive network management in sliced 5G networks with deep learning”, *Proceedings of the IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, Paris, France, pp. 280–288, 29 April–2 May 2019.

[54] S. Martiradonna, A. Abrardo, M. Moretti, G. Piro, G. Boggia, “Deep reinforcement learning-aided RAN slicing enforcement supporting latency sensitive services in B5G networks”, *Internet Technology Letters*, Vol. 4, Issue 6, November/December 2021.

[55] B. Khodapanah, A. Awada, I. Viering, A. n. Barreto, M. Simsek and G. Fettweis, "Framework for Slice-Aware Radio Resource Management Utilizing Artificial Neural Networks," *IEEE Access*, vol. 8, pp. 174972-174987, 2020.

[56] T. C. Chuah and Y. L. Lee, "Intelligent RAN Slicing for Broadband Access in the 5G and Big Data Era," *IEEE Communications Magazine*, vol. 58, no. 8, pp. 69-75, August 2020.

[57] R. Ferrús, J. Pérez-Romero, O. Sallent, I. Vilà, R. Agustí “Machine Learning-Assisted Cross-Slice Radio Resource Optimization: Implementation Framework and Algorithmic Solution”, *ITU Journal on Future and Evolving Technologies*, Volume 1 (2020), Issue 1, 18 December 2020.

[58] *5G Non-Public Networks for Industrial Scenarios*, 5G-ACIA Whitepaper, Julio 2019.

[59] *Private 5G Mobile Networks for Industrial IoT*, Qualcomm, Julio 2019.

[60] *Wireless TSN - Definitions, Use Cases & Standards Roadmap*, Avnu Alliance® White Paper, v1.0, Marzo 2020.

[61] IEEE Standard 802.1AS-2020, “IEEE Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications”, published Junio 2020.

[62] IEEE Standard 1588-2019, “IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems”, Noviembre 2019.

[63] Time-Sensitive Networking: A Technical Introduction, White Paper, Cisco Public, 2017.

[64] IEEE 802.1Qcc-2018, “Standard for Local and metropolitan area networks - Bridges and Bridged Networks - Amendment: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements“, Octubre 2018.

[65] Theory of Operation for TSN-enabled Systems Applied to Industrial Markets , rev. 1.0, Avnu Alliance Best Practices, 2017.

[66] *Integration of 5G with Time-Sensitive Networking for Industrial Communications*, 5G-ACIA Whitepaper, Diciembre 2020.

[67] IEEE Standard 802.1Q-2018, “IEEE Standard for Local and metropolitan area networks: Bridges and Bridged Networks”, Julio 2018.

[68] IEEE Standard 802.3-2018, “IEEE Standard for Ethernet”, published August 2018.

[69] IEEE 802.1CB-2017, “IEEE Standard for Local and metropolitan area networks-Frame Replication and Elimination for Reliability“, Octubre 2017.

[70] IEEE Std 802.1AB-2016: "IEEE Standard for Local and metropolitan area networks -- Station and Media Access Control Connectivity Discovery".

[71] *5G-TSN integration for industrial automation*, Ericsson Technology Review, Julio 2019.

[72] 3GPP; Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 16), TS 23.501 v17.2.0, sept. 2021.

[73] 6G Flagship, “White Paper on Machine Learning in 6G Wireless Communication Networks”, 6G Research Visions, No. 7, June 2020

[74] D. Bega, et al. “AI-Based Autonomous Control, Management, and Orchestration in 5G: From Standards to Algorithms”, *IEEE Network*, vol. 34, no. 6, pp. 14-20, Dec. 2020

[75] S. Ali, W. Saad, and D. Steinbach, “White Paper on Machine Learning in 6G Wireless Communication Networks”, White Paper 6G Research Vision, no. 7, Jun. 2020

[76] 6G Flagship, “6G White Paper on Edge Intelligence”, 6G Research Visions, No. 8, June 2020

[77] Y. Xu, F. Yin, W. Xu, C. -H. Lee, J. Lin and S. Cui, “Scalable Learning Paradigms for Data-Driven Wireless Communication”, *IEEE Communications Magazine*, vol. 58, no. 10, pp. 81-87, October 2020.

[78] J. Guo, W. Luo, B. Song, F. R. Yu and X. Du, “Intelligence-Sharing Vehicular Networks with Mobile Edge Computing and Spatiotemporal Knowledge Transfer”, *IEEE Network*, vol. 34, no. 4, pp. 256-262, July/August 2020.

[79] Y. Liu, X. Yuan, Z. Xiong, J. Kang, X. Wang and D. Niyato, “Federated learning for 6G communications: Challenges, methods, and future directions”, *China Communications*, vol. 17, no. 9, pp. 105-118, Sept. 2020.

[80] S. Zhang, et al., “Envisioning Device-to-Device Communications in 6G”, *IEEE Network*, vol. 34, no. 3, pp. 86-91, June 2020.

[81] C. She et al., “Deep Learning for Ultra-Reliable and Low-Latency Communications in 6G Networks”, *IEEE Network*, vol. 34, no. 5, pp. 219-225, September/October 2020.

[82] 5GAA, “Making 5G Proactive and Predictive for the Automotive Industry”, Whitepaper, Dec. 2019.

[83] 3GPP TS 23.288 (v16.2.0), “Architecture Enhancements for 5G System (5GS) to Support Network Data Analytics Services (Release 16)”, Dec. 2019.

[84] A. Banchs, et al., "A 5G Mobile Network Architecture to Support Vertical Industries", IEEE Communications Magazine, vol. 57, no. 12, pp. 38-44, Dec. 2019.

[85] 3GPP TR 23.700-91 (v2.0.0), “Study on enablers for network automation for the 5G System (5GS); Phase 2 (Release 17)”, Nov. 2020.

[86] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, E. Riviere, “Edge-centric computing: vision and challenges”, *SIGCOMM Comput Commun Rev* 45(5), pp. 37–42, 2015.

[87] C. Puliafito, E. Mingozzi, F. Longo, A. Puliafito, O. Rana, “Fog computing for the Internet of Things: a survey”, *ACM Trans Internet Technol*, vol. 19(2): 18:1–18:41, 2019.

[88] L. Chettri, R. Bera, “A comprehensive survey on Internet of Things (IoT) toward 5G wireless systems”, *IEEE Internet of Things Journal*, vol. 7(1):16–32, 2020.

[89] P. Porambage, J. Okwuibe, M. Liyanage, M. Ylianttila, T. Taleb, “Survey on multi-access edge computing for Internet of Things realization”, *IEEE Communications Survey and Tutorials*, vol. 20(4), pp. 2961–2991, Fourthquarter 2018.

[90] T.X. Tran, A. Hajisami, P. Pandey, D. Pompili, “Collaborative mobile edge computing in 5G networks: new paradigms, scenarios, and challenges”, *IEEE Communications Magazine*, Vol. 55(4), pp. 54–61, 2017.

[91] P. Mach, Z. Becvar, “Mobile edge computing: a survey on architecture and computation offloading”, IEEE Communication Survey and Tutorials, vol. 19(3), pp. 1628–1656, thirdquarter 2017.

[92] E. Harjula, P. Karhula, J. Islam, T. Leppadnen, A. Manzoor, M. Liyanage, J. Chauhan, T. Kumar, I. Ahmad, M. Ylianttila, “Decentralized IoT edge nanoservice architecture for future gadget-free computing”, IEEE Access, vol. 7, pp. 119856–119872, 2019.

[93] Y. Ai, M. Peng, K. Zhang, “Edge computing technologies for Internet of Things: a primer”, *Digital Communications and Networks*, vol. 4(2), pp. 77–86, 2018.

[94] 3GPP, Technical Specification Group Services and System Aspects; System architecture for the 5G System (5GS); Stage 2 (Release 16), 3GPP TS 23.501, v16.3.0, Dec. 2019.

[95] ETSI, MEC in 5G networks (White Paper No. 28), June 2018.

[96] K. B. Letaief, W. Chen, Y. Shi, J. Zhang and Y. A. Zhang, “The Roadmap to 6G: AI Empowered Wireless Networks”, *IEEE Communications Magazine*, vol. 57, no. 8, pp. 84-90, August 2019.

[97] 3GPP, Technical Specification Group Services and System Aspects; Study on enhanced support of Industrial Internet of Things (IIoT) in the 5G System (5GS) (Release 17), 3GPP TR 23.700-20, v17.0.0, Marzo 2021.

[98] J. Gubbi, R. Buyya, S. Marusic, M. Palaniswami, “Internet of Things (IoT): a vision, architectural elements, and future directions”, Future Generation Computer Systems, vol. 29(7), pp. 1645–1660, 2013. Including special sections: cyber-enabled distributed computing for ubiquitous cloud and network services & cloud computing and scientific applications – big data, scalable analytics, and beyond.

[99] O.B. Sezer, E. Dogdu, A.M. Ozbayoglu, “Context-aware computing, learning, and big data in Internet of Things: a survey”, *IEEE Internet of Things Journal*, vol. 5(1), pp. 1–27, 2018.

[100] G. Xu, E.C. Ngai, J. Liu, “Ubiquitous transmission of multimedia sensor data in Internet of Things”, IEEE Internet of Things Journal, vol. 5(1), pp. 403–414, 2018.

[101] 3GPP TS 38.300 V16.4.0, “Technical Specification Group Radio Access Network; NR; NR and NG-RAN Overall Description; Stage 2 (Release 16)”, Dec. 2020.

Anexo A.1. Requisitos de servicio para los casos de uso de la Industria 4.0 definidos por el 3GPP en [14]

En este anexo se presentan los requisitos de servicio identificados por el 3GPP para los distintos casos de uso definidos en [14]. Los casos de uso se agrupan en 3 tablas diferenciando los casos de uso con tráfico determinista periódico, tráfico determinista aperiódico y tráfico no determinista.

Tabla 14. Requisitos de servicio para casos de uso con comunicaciones deterministas periódicas [14].

Caso de uso	*Communication service availability*	Tiempo medio entre fallos	Latencia *end-to-end*	*Service bit rate*	Tamaño del mensaje [byte]	*Transfer interval*	*Survival time*	Velocidad del UE	# de UEs	Área de servicio
Motion control	99.999 % to 99.999 99 %	~ 10 years	< transfer interval value	–	50	500 µs	500 µs	≤ 75 km/h	≤ 20	50 m x 10 m x 10 m
Motion control	99.999 9 % to 99.999 999 %	~ 10 years	< transfer interval value	–	40	1 ms	1 ms	≤ 75 km/h	≤ 50	50 m x 10 m x 10 m
Motion control	99.999 9 % to 99.999 999 %	~ 10 years	< transfer interval value	–	20	2 ms	2 ms	≤ 75 km/h	≤ 100	50 m x 10 m x 10 m
Control-to-control in motion control	99.999 9 % to 99.999 999 %	~ 10 years	< transfer interval value		1 k	≤ 10 ms	10 ms	-	5 to 10	100 m x 30 m x 10 m
Wired-2-wireless 100 Mbit/s link replacement	99.999 9 % to 99.999 999 %	~ 10 years	< transfer interval value	50 Mbit/s		≤ 1 ms	3 x transfer interval	stationary	2 to 5	100 m x 30 m x 10 m
Wired-2-wireless 1 Gbit/s link replacement	99.999 9 % to 99.999 999 %	~ 10 years	< transfer interval value	250 Mbit/s		≤ 1 ms	3 x transfer interval	stationary	2 to 5	100 m x 30 m x 10 m
Control-to-control in motion control	99.999 9 % to 99.999 999 %	~ 10 years	< transfer interval value		1 k	≤ 50 ms	50 ms	-	5 to 10	1,000 m x 30 m x 10 m
Mobile robots	> 99.999 9 %	~ 10 years	< transfer interval value	–	40 to 250	1 ms to 50 ms	transfer interval value	≤ 50 km/h	≤ 100	≤ 1 km^2
Mobile control panels – remote control	99.999 9 % to 99.999 999 %	~ 1 month	< transfer interval value	–	40 to 250	4 ms to 8 ms	transfer interval value	< 8 km/h (linear movement)	TBD	50 m x 10 m x 4 m
Mobile Operation Panel: Emergency stop	99.999 999 %	1 day	< 8 ms	250 kbit/s	40 to 250	8 ms	16 ms	quasi-static; up to 10 km/h	2 or more	30 m x 30 m

Caso de uso	*Communication service availability*	Tiempo medio entre fallos	Latencia *end-to-end*	*Service bit rate*	Tamaño del mensaje [byte]	*Transfer interval*	*Survival time*	Velocidad del UE	# de UEs	Área de servicio
Mobile Operation Panel: Safety data stream	99.999 99 %	1 day	< 10 ms	< 1 Mbit/s	<1024	10 ms	~10 ms	quasi-static; up to 10 km/h	2 or more	30 m x 30 m
Mobile Operation Panel: Control to visualization	99.999 999 %	1 day	10 ms to 100 ms	10 kbit/s	10 to 100	10 ms to 100 ms	transfer interval	stationary	2 or more	100 m² to 2,000 m²
Mobile Operation Panel: Motion control	99.999 999 %	1 day	< 1 ms	12 Mbit/s to 16 Mbit/s	10 to 100	1 ms	~ 1 ms	stationary	2 or more	100 m²
Mobile Operation Panel: Haptic feedback data stream	99.999 999 %	1 day	< 2 ms	16 kbit/s (UL) 2 Mbit/s (DL)	50	2 ms	~ 2 ms	stationary	2 or more	100 m²
Mobile control panels -remote control	99.999 9 % to 99.999 999 %	~ 1 year	< transfer interval	–	40 to 250	< 12 ms	12 ms	< 8 km/h (linear movement)	TBD	typically 40 m x 60 m; maximum 200 m x 300 m
Process automation – closed loop control	99.999 9 % to 99.999 999 %	≥ 1 year	< transfer interval value	–	20	≥ 10 ms	0	typically stationary	typically 10 to 20	typically ≤ 100 m x 100 m x 50 m
Mobile robots – video-operated remote control	> 99.999 9 %	~ 1 year	< transfer interval value	–	15 k to 250 k	10 ms to 100 ms	transfer interval value	≤ 50 km/h	≤ 100	≤ 1 km²
Mobile robots	> 99.999 9 %	~ 1 year	< transfer interval value	–	40 to 250	40 ms to 500 ms	transfer interval value	≤ 50 km/h	≤ 100	≤ 1 km²
Plant asset management	99.99 %	≥ 1 week	< transfer interval value	–	20 to 255	100 ms to 60 s	≥ 3 x transfer interval value	typically stationary	≤ 10,000 to 100,000	≤ 10 km x 10 km x 50 m
Cooperative carrying – fragile work pieces	99.999 9 % to 99.999 999 %	~ 10 years	< 0.5 x transfer interval	2.5 Mbit/s	250 500 with localisation information	> 5 ms > 2.5 ms > 1.7 ms	0 transfer interval 2 x transfer interval	≤ 6 km/h (linear movement)	2 to 8	10 m x 10 m x 5 m; 50 m x 5 m x 5 m
Cooperative carrying – elastic work pieces	99.999 9 % to 99.999 999 %	~ 10 years	< 0.5 x transfer interval	2.5 Mbit/s	250 500 with localisation information	> 5 ms > 2.5 ms > 1.7 ms	0 transfer interval 2 x transfer interval	≤ 12 km/h (linear movement)	2 to 8	10 m x 10 m x 5 m; 50 m x 5 m x 5 m

Tabla 15. Requisitos de servicio para casos de uso con comunicaciones deterministas aperiódicas [14].

Caso de uso	*Communication service availability*	Tiempo medio entre fallos	Latencia *end-to-end*	*Service bit rate*	Tamaño del mensaje [byte]	*Survival time*	Velocidad del UE	# de UEs	Área de servicio
Mobile robots – video streaming	> 99.999 9 %	~ 1 week	10 ms	UL: > 10 Mbit/s			≤ 50 km/h	≤ 100	≤ 1 km^2
Mobile control panels - parallel data transmission	99.999 9 % to 99.999 999 %	~ 1 month	< 30 ms	> 5 Mbit/s			< 8 km/h (linear movement)	TBD	TBD
Mobile Operation Panel: Emergency stop (emergency stop events)	99.999 999 %	1 day	<8 ms	250 kbit/s	40 to 250	16 ms	quasi-static; up to 10 km/h	2 or more	30 m x 30 m
Augmented reality; bi-directional transmission to image processing server	> 99.9 %	~ 1 month	< 10 ms				< 8 km/h (linear movement)	≥ 3	20 m x 20 m x 4 m
Wired-2-wireless 100 Mbit/s link replacement	99.999 9 % to 99.999 999 %	~ 10 years	< 1 ms	25 Mbit/s			stationary	2 to 5	100 m x 30 m x 10 m
Wired-2-wireless 1 Gbit/s link replacement	99.999 9 % to 99.999 999 %	~ 10 years	< 1 ms	500 Mbit/s			stationary	2 to 5	100 m x 30 m x 10 m

Tabla 16. Requisitos de servicio para casos de uso con comunicaciones no deterministas [14].

Caso de uso	Tiempo medio entre fallos	*Service bit rate* Tamaño del mensaje [byte]	*Survival time*	Velocidad del UE	# de UEs	Área de servicio
Motion control - software updates	~ 1 month	DL: ≥ 1 Mbit/s		~ 0 km/h ≤ 75 km/h	≤ 100	50 m x 10 m x 10 m
Mobile robots; real-time video stream		UL: > 10 Mbit/s		≤ 50 km/h (linear movement)	≤ 100	≤ 1 km^2

Anexo A.2. Requisitos de servicio para los casos de uso de la Industria 4.0 definidos por el ETSI en [20]

En este anexo se presentan los requisitos de servicio identificados por el ETSI para los distintos casos de uso definidos en [20]. No se incluye en este documento la definición de los distintos casos de uso por ser estos bastante similares a los definidos por el 3GPP y presentados en el apartado 3.2; la definición de los casos de uso se encuentra disponible en [20].

Tabla 17. Requisitos de servicio para casos de uso con comunicaciones deterministas aperiódicas [20].

	Monitorización y diagnóstico		Fabricación discreta		Logística y almacén			Automatización de procesos	Realidad aumentada	Seguridad funcional (*Safety*)
Parámetro	**General**	**Monitorización o *condition monitoring***	**General**	**Control de movimiento**	**General**	**AGVs**	**Grúas**			
Latencia/ciclo	>20 ms	100 ms	1-12 ms	250µs-1 ms	>50 ms	15-20 ms	15-20 ms	50 ms	10 ms	10 ms
Fiabilidad	$1 - 10^{-4}$	$1 - 10^{-5}$	$1 - 10^{-9}$	$1 - 10^{-9}$	$>1 - 10^{-2}$	$>1 - 10^{-6}$	$>1 - 10^{-6}$	$1 - 10^{-5}$	$1 - 10^{-5}$	$1 - 10^{-9}$
Tasa de transmission	kbit/s - Mbit/s	kbit/s	kbit/s - Mbit/s	kbit/s - Mbit/s	kbit/s - Mbit/s	kbit/s - Mbit/s	kbit/s - Mbit/s	kbit/s	Mbit/s - Gbit/s	kbit/s
Tamaño de paquete	> 200 bytes	1-50 bytes	20-50 bytes	20-50 bytes	< 300 bytes	< 300 bytes	< 300 bytes	<80 bytes	> 200 bytes	<8 bytes
Rango	< 100 m	100 m – 1 km	< 100 m	< 50 m	< 200 m	~ 2 m	< 100 m	100 m-1km	< 100 m	< 10 m
Mobilidad del UE	0 m/s	< 10 m/s	< 10 m/s	< 10 m/s	< 40 m/s	< 10 m/s	< 5 m/s	Generalmente estático	<3 m/s	< 10 m/s
Densidad de UEs	$0.33\text{-}3m^{-2}$	$10\text{-}20\ m^{-2}$	$0.33\text{-}3m^{-2}$	$<5\ m^{-2}$	$\sim 0.1m^{-2}$	$\sim 0.1m^{-2}$	$\sim 0.1m^{-2}$	10000/fábrica	$>0.33\text{-}0.02m^{-2}$	$>0.33\text{-}0.02m^{-2}$
Eficiencia energética	n/a	10 años	n/a	n/a	n/a	<8h	n/a	10 años	1 día	n/a

Parte II
Generación de modelos digitales de plantas industriales

7. Introducción

En esta segunda parte del informe, se presenta el trabajo y resultados obtenidos en el marco del segundo objetivo 'Generación de modelos digitales de plantas industriales'.

Como se ha presentado en la primera parte de este informe, la tecnología 5G tiene un gran potencial para cumplir con los requisitos de la Industria 4.0: 5G es altamente flexible y tiene una gran capacidad de adaptación y reconfiguración, a la vez que implementa los mecanismos necesarios para dar soporte a comunicaciones de baja latencia y alta fiabilidad (URLLC). Sin embargo, para conseguir una integración eficiente de la tecnología 5G en la industria como catalizador tecnológico clave para su transformación digital hacia modelos de Industria 4.0 es necesario todavía seguir trabajando para superar algunos retos todavía presentes. Uno de estos retos es diseñar mecanismos de gestión autónoma y flexible de las redes 5G para que estas sean capaces de adaptarse de manera autónoma a los cambios de contexto en el entorno industrial y de la demanda de tráfico para garantizar los niveles de resiliencia y fiabilidad del entorno industrial, y satisfacer sin interrupciones los exigentes requisitos de QoS de las aplicaciones industriales.

La digitalización de la industria está dando lugar a ecosistemas en los que un gran número de nodos (sensores, actuadores, robots, maquinaria, procesos, etc.) generan y consumen grandes cantidades de datos. La disponibilidad de estos datos hace que la aplicación de técnicas de inteligencia artificial o IA se convierta en una tecnología clave para el diseño de estos mecanismos de gestión autónoma y flexible de la red 5G. Sin embargo, el diseño de técnicas de gestión basada en IA requiere de grandes bancos de datos que reflejen el comportamiento del sistema industrial para el entrenamiento y validación de las técnicas diseñadas. Sin embargo, la disponibilidad de estos *datasets* en la comunidad actualmente es nula dado el carácter confidencial del diseño y configuración de sistemas de producción en plantas industriales reales, lo cual es un factor limitante para el desarrollo de soluciones basadas en inteligencia artificial.

En este contexto, en este proyecto se han implementado modelos digitales de plantas industriales capaces de emular de forma realista (y configurable) el comportamiento de plantas industriales para poder así conocer y caracterizar la necesidad de conectividad y transferencia de datos dentro de las plantas. La disponibilidad de estos modelos digitales permitirá generar valiosos *datasets* sobre la generación y transferencia de datos industriales en diferentes contextos y configuraciones de las plantas industriales. La disponibilidad de los modelos digitales de fábrica y los *datasets* supone un gran hito y es clave para el entrenamiento y validación de soluciones basadas en inteligencia artificial. De esta manera, como objetivo futuro se prevé explotar dichos modelos digitales y *datasets* para generar soluciones predictivas basadas en IA de gestión autónoma de las redes 5G de forma que sea posible predecir la demanda de conectividad de los sistemas industriales, y adaptar de forma proactiva la configuración y operación de la red 5G en base a dicha demanda.

A continuación, se presenta el trabajo realizado. En primer lugar, el apartado 8 presenta una descripción de los escenarios de plantas industriales seleccionados para ser modelados digitalmente. En el apartado 9, se detalla la generación de datos en los distintos escenarios y las comunicaciones que se llevan a cabo en el escenario para la transmisión de dichos datos entre distintos componentes del escenario. A continuación, se presenta cómo se ha realizado el modelado digital de los escenarios en el apartado 10. En el apartado 10, se presenta en primer lugar la herramienta utilizada para ello, el software *Visual Components* de modelado 3D de entornos de fabricación, y a continuación se detalla el proceso seguido para la implementación de los modelos digitales. El apartado 10 además presenta los nuevos componentes y funciones que se han tenido implementar en *Visual Components* para emular la generación de datos en el escenario ya que actualmente *Visual Components* no incorpora dicha funcionalidad. Por último, en el apartado 11 se describen los *datasets* generados y la nueva funcionalidad implementada en *Visual Componentes* para recopilar la información y crear los *datasets* en tiempo de simulación.

8. Escenarios industriales

Este apartado presenta una descripción general del funcionamiento de los escenarios industriales seleccionados para ser modelados en el software *Visual Components*. En concreto, en este proyecto se han implementado 2 escenarios diferentes. El primero de ellos implementa una línea de producción cuya finalidad es el moldeado de láminas de acero. Esta línea de producción incluye en su parte final un sistema de telemetría y telecontrol de calidad realizado de manera remota en nodos de computación implementados en el *edge*. Esta línea de producción está implementada en un centro tecnológico de la empresa Innovalia Association localizada en el País Vasco. El segundo escenario considera una planta industrial cuya función es el moldeado de láminas de acero para puertas de automóviles. En esta planta industrial se implementan varias líneas de producción en paralelo y además tiene integrados 2 con almacenes con gestión automatizada. Este segundo escenario está basado en la implementación previamente existente en el software *Visual Components*. Este escenario corresponde a un centro de producción real, aunque por temas de confidencialidad, *Visual Components* no aporta datos identificativos de dicha industria.

8.1. Escenario 1. Línea de prensado de láminas de acero con control de calidad basado en telemetría y telecontrol

El primer escenario comprende una única línea de producción cuya finalidad es moldear láminas de acero para ser utilizadas con distintos fines. Esta línea de producción está compuesta de distintas celdas de procesado en las que se realizan tareas concretas: prensado, soldadura, medición y control, y almacenamiento. A continuación, se presenta el funcionamiento general de la línea.

Esta línea de producción está dividida en distintas celdas. En particular, la línea consta de una celda de prensado, en la que las láminas de acero son sometidas a un proceso de prensado para darle la forma deseada. Una vez moldeadas, las láminas se transportan a la celda de soldadura en la que se someten a un proceso para soldar pequeñas partes a la lámina principal. Completado este proceso de soldadura, las láminas pasan a la celda de telemetría o medición, en la que se realiza el proceso de control de calidad al producto. Para ello, se utiliza una máquina de medición por coordenadas o CMM (*Coordinate-measuring machine*) que inspecciona cada lámina. Los datos de la medición son enviados a un ordenador remoto situado en una sala de control donde se comprueba que dichos datos estén dentro de los valores umbrales definidos por el usuario para el moldeado de lámina que se esté fabricando. Si los datos están dentro de los valores umbrales, el resultado del control de calidad es apto, y las piezas finalmente se trasladan al almacén para su almacenamiento. En caso contrario las láminas son desechadas. Más adelante en este apartado se presenta de forma detallada el funcionamiento de cada una de estas celdas y el flujo de trabajo de la planta. El transporte de las láminas entre celdas es realizado por un AGV (*Automated Guided Vehicle*). Todos los procesos se monitorizan desde la sala de control central situada en el mismo escenario. La Figura 20 muestra una representación del escenario descrito en las que se puede observar las distintas celdas que forman la línea de producción.

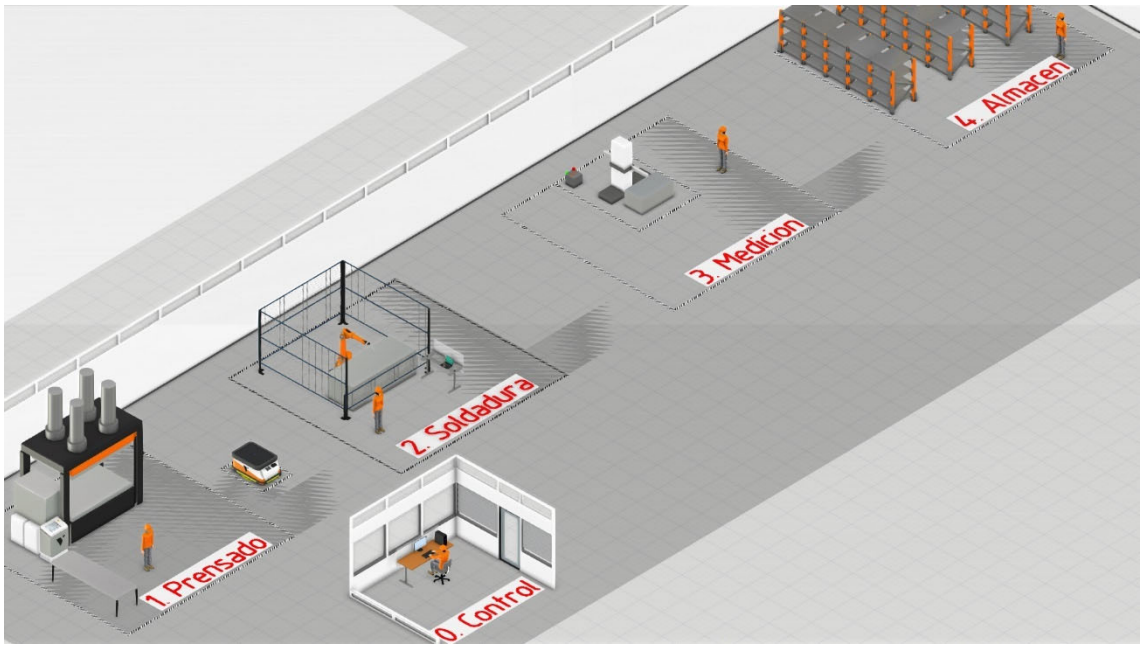


Figura 20. Representación del escenario 1 - Línea de prensado de láminas de acero con control de calidad basado en telemetría y telecontrol.

Como se ha comentado anteriormente, esta línea de producción ha sido desarrollada y por la Asociación Innovalia[4], una unidad empresarial de I+D privada que tiene como principal propósito fomentar la innovación tecnológica en las pequeñas y medianas empresas. Esta línea de producción se ha desarrollado en las propias instalaciones de Innovalia para llevar a cabo el estudio, investigación e implementación de distintas innovaciones tecnológicas relacionadas con la automatización de procesos industriales. Entre estas investigaciones, cabe destacar el uso de redes 5G para dar soporte a las comunicaciones que tienen lugar en el escenario, entre las que se encuentran comunicaciones con requisitos de baja latencia entre las máquinas y procesos locales y procesos remotos implementados de forma centralizada en la nube o *Cloud*, o en un nodo de computación en el *edge* (el *edge* representa nodos de computación distribuida que pueden estar implementados en las propias instalaciones de la fábrica y reducen la latencia o tiempos de transmisión de la información entre el proceso local y el nodo de computación remoto). En este contexto, el modelado digital de este escenario y la simulación de la generación de datos a transmitir entre distintos elementos de la línea de producción resulta de gran interés para Innovalia, ya que utilizando el modelado digital del escenario podrán estudiar y dimensionar la red 5G para dar soporte a las necesidades de comunicación del escenario.

A continuación, se describe de forma detallada el funcionamiento de cada una de estas celdas y el flujo de trabajo de la planta.

8.1.1. Celda de prensado

La celda de prensado constituye el inicio del flujo de trabajo, este se lleva a cabo gracias a un operario humano, una prensa industrial y su controlador y la asistencia de un AGV (la Figura 21 muestra la celda de prensado). El proceso se inicia con la colación de las láminas de acero (una a una) en la prensa por parte del operario. Una vez colocada la lámina en la prensa, el operario inicial el proceso desde el controlador de la misma situado junto a la prensa (ver Figura 21). Cuando el proceso de prensado finaliza, el controlador de la prensa notifica al AGV para que

[4] https://innovalia.org/

transporte el producto hasta la siguiente celda. En este caso, el operario se encarga de depositar el material (las láminas prensadas) encima del AGV. Una vez el AGV detecta el producto en su superficie, inicia el transporte.

Adicionalmente, la prensa envía datos de monitorización al centro de control remoto, estando así el proceso de prensado monitorizado en todo momento. Todas las comunicaciones existentes en esta celda quedan recogidas en el apartado 9.1.2 de este documento.

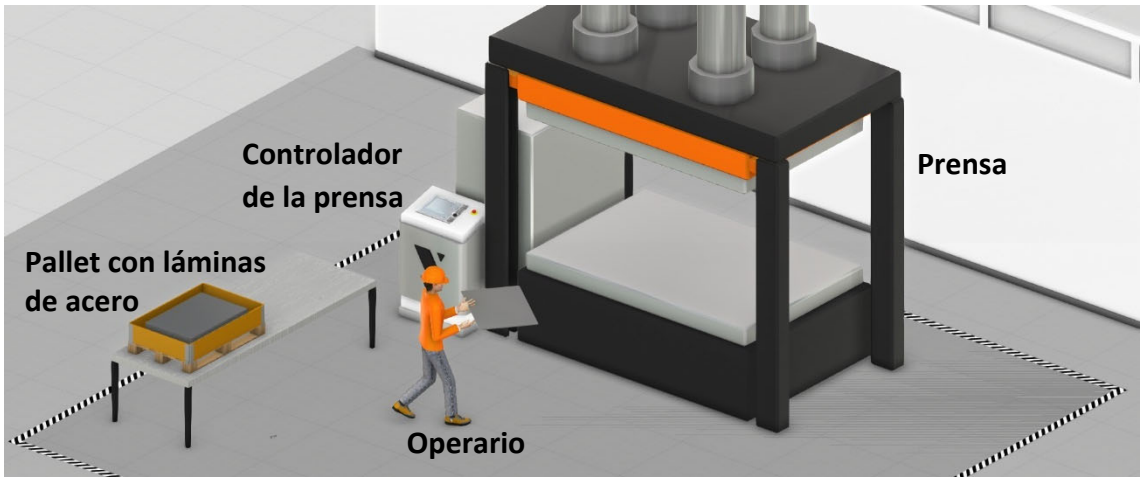


Figura 21. Representación gráfica de la celda de prensado.

8.1.2. Celda de soldadura

La celda de soldadura alberga el siguiente proceso en la línea de producción tras el prensado. Este proceso se realiza de manera similar al que se lleva a cabo en la celda de prensado y en él participan un operario humano y una serie de elementos mecatrónicos que asisten su trabajo. En esta celda se llevan a cabo las soldaduras necesarias en las láminas de acero ya moldeadas. Este proceso está automatizado mediante un robot articulado cuyo funcionamiento se activa por el operario a través del ordenador situado en la misma celda. La Figura 22 muestra una representación de la celda de soldadura.

El proceso se inicia cuando el AGV llega con el producto (la lámina moldeada) desde la celda de prensado hasta la ubicación indicada en la celda de soldadura. En ese momento, el operario coge la lámina y la coloca en la ubicación específica para que el robot pueda realizar la soldadura. Hecho esto, el operario inicia el ciclo de soldadura mediante el PC situado sobre la mesa próxima al robot. Una vez terminada la soldadura, el robot notifica que ha finalizado el proceso, indicando así al AGV que debe transportar el producto (la pieza soldada) hasta la siguiente celda. Cuando el AGV alcanza la posición deseada, el operario coloca el producto sobre la superficie del AGV. Tras detectar un producto sobre su superficie, el AGV inicia el transporte hacia la siguiente celda. De igual modo que en la celda anterior, desde la celda de soldadura se envían datos de monitorización al centro de control. Toda la información detallada de las comunicaciones de esta celda queda recogida en el apartado 9.1.3.

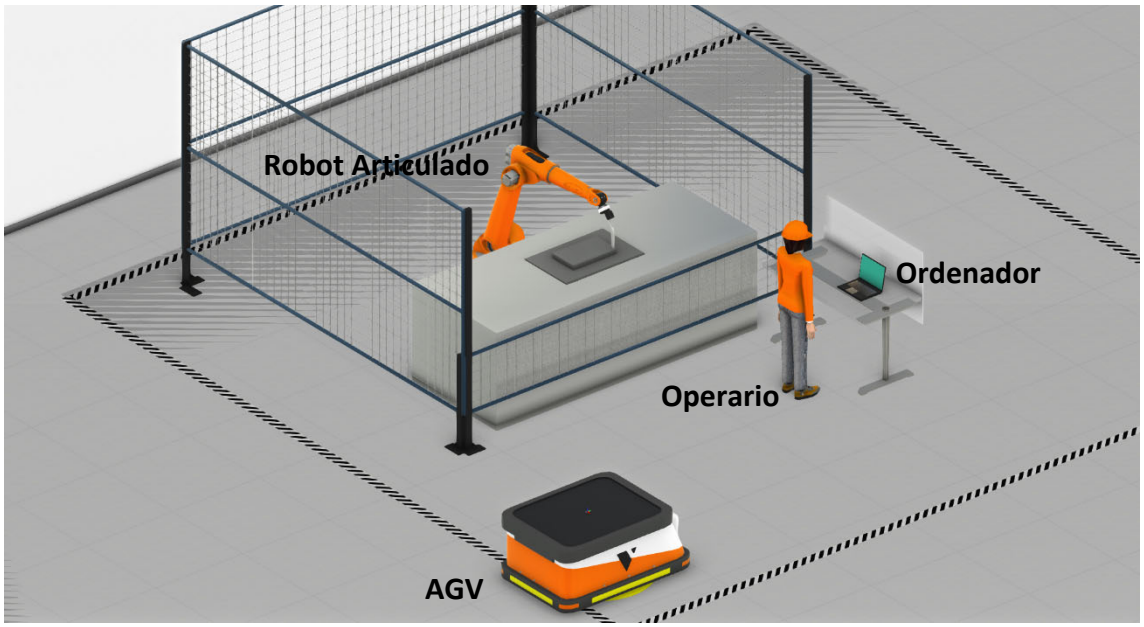


Figura 22. Representación gráfica de la celda de soldadura.

8.1.3. Celda de medición

Tras el proceso de soldadura, el producto se somete a un proceso de medición para el control de calidad. En este proceso de control de calidad se realizan una serie de mediciones del producto finalizado para asegurar que cumple con las especificaciones dadas. Este proceso de medida se realiza mediante dispositivo de medición por coordenadas o CMM.

Las láminas llegan sobre el AGV desde la celda de soldadura y mediante un operario son colocadas en la ubicación indicada para su medición. Una vez hecho esto, el AGV queda libre para realizar otras tareas de transporte. El proceso de medición del producto se inicia de manera remota desde la celda de control. Dado que la cantidad de puntos que es capaz de medir el CMM es muy extensa, es necesario realizar un proceso de muestreo y procesado de los datos. El procesado de los datos y medidas se realiza en el edge. mediante un *edge*. Como ya se explicó anteriormente, pero es importante recordar, el *edge* representa nodos de computación distribuida que pueden estar implementados en las propias instalaciones de la fábrica y reducen la latencia o tiempos de transmisión de la información entre el proceso local y el nodo de computación remoto. Al mismo tiempo que el CMM obtiene las coordenadas de los puntos estos son enviados al *edge* para su procesado. A medida que los datos son procesados en el *edge*, los datos procesados son enviados al ordenador central de la planta desde el cual se realizará monitorización de estos y se envían nuevas órdenes de medida al CMM. Debido a la baja latencia requerida para la comunicación entre CMM y el *edge*, el *edge* se sitúa en el propio armario de control del CMM para realizar una conexión cableada. Una vez terminado el ciclo de medición, el operario coloca la lámina sobre el AGV y este la lleva a la celda siguiente. Las comunicaciones que tienen lugar en esta celda se detallan en el apartado 9. En la Figura 23 se muestra dicha celda con sus elementos etiquetados.

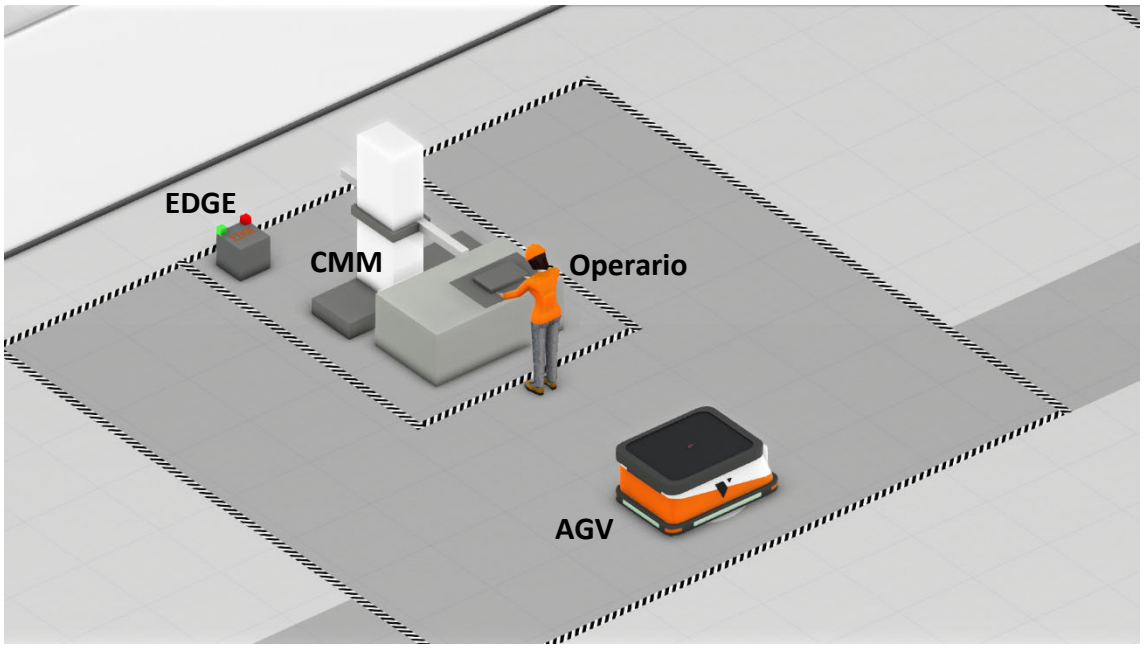


Figura 23. Representación gráfica de la celda de medición.

8.1.4. Almacén

El proceso termina con el transporte del producto al almacén. El AGV transportará el producto hasta la ubicación indicada en el almacén donde un operario se encarga de colocar dicha lámina en las estanterías.

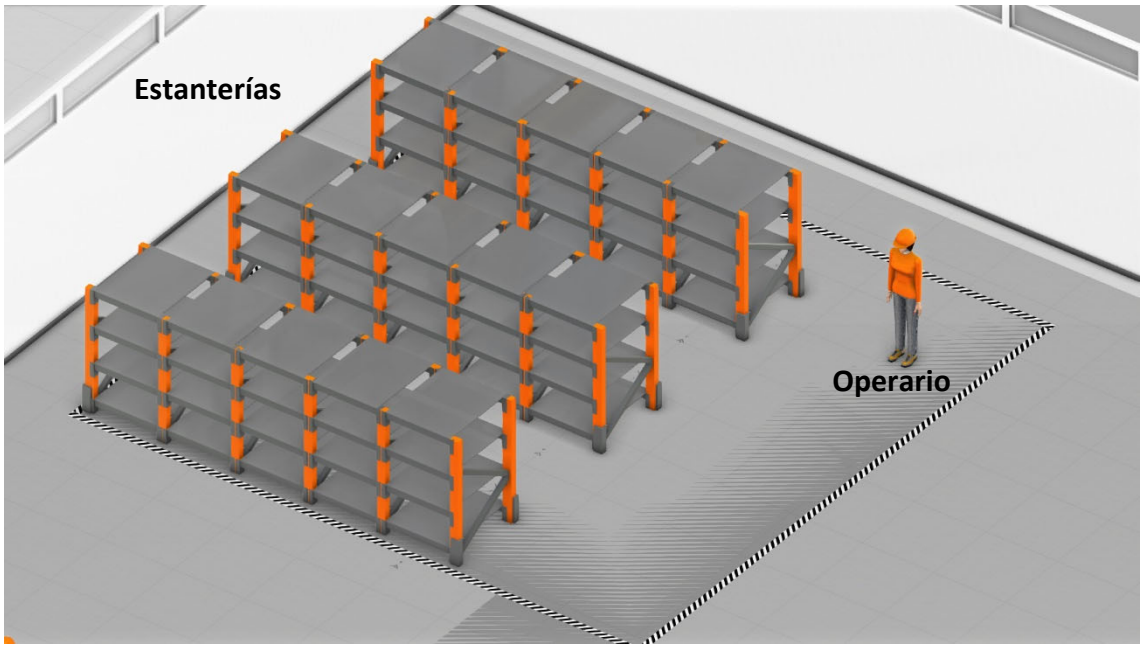


Figura 24. Representación gráfica del almacén.

8.1.5. Celda de control

Esta celda está constituida básicamente por un ordenador central de control y un operario que supervisa los distintos procesos realizados en la línea. El ordenador central de control recibe información de los distinto procesos y componentes del escenario. En base a los datos recibidos, genera instrucciones o comandos de control para indicar a cada componente las acciones que ha de realizar. Desde las celdas de prensado y soldadura se envían datos de los distintos procesos que se realizan en ellas para su monitorización desde el ordenador central de control situado en la sala de control. La configuración, control y manejo del CMM se realiza de forma remota desde el ordenador central (telemetría y telecontrol). El ordenador central envía comandos de configuración y funcionamiento al CMM (por ejemplo, indica cuándo debe comenzar y finalizar el proceso de captura y medida, indica posiciones en las que debe realizar nuevas medidas y el ángulo con el que debe realizarlas, etc.). Desde el CMM se envían las medidas realizadas al ordenador central. En el ordenador central de control también se implementa el controlador del AGV, por lo que las órdenes de movimiento son enviadas desde el ordenador central de control al AGV. Aunque el AGV es capaz de calcular de forma autónoma la trayectoria a seguir para alcanzar el punto destino, el AGV envía de manera periódica información sobre su posición y estado de batería actual al ordenador central de control, para que este monitorice su posición en tiempo real. En la Figura 25 se muestra una representación de la celda de control.

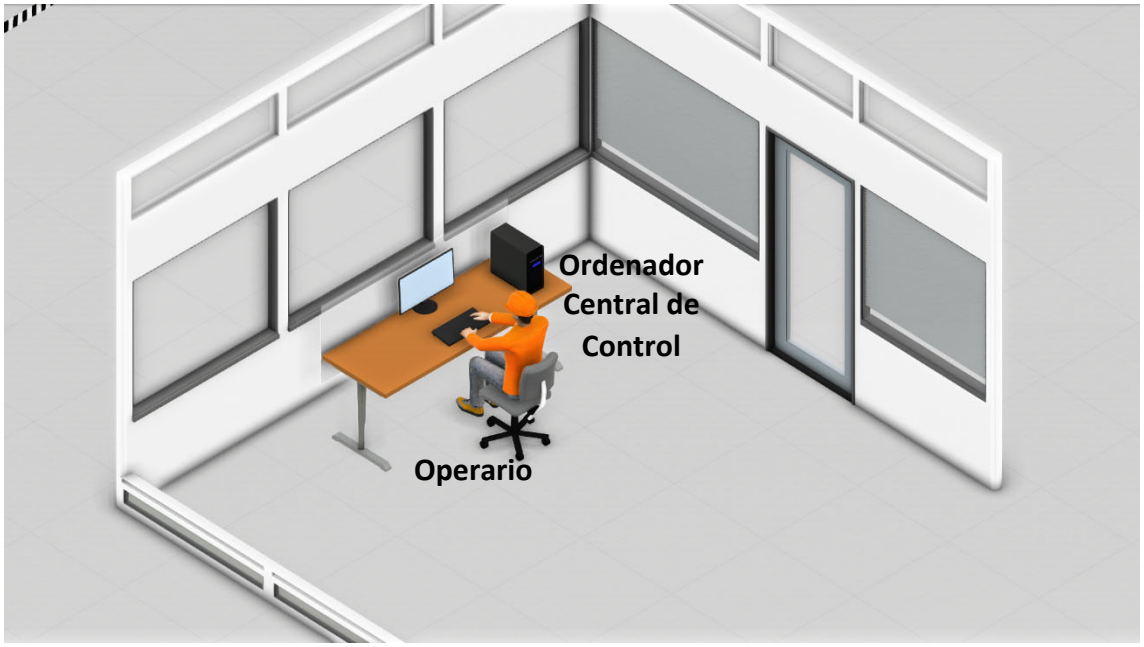


Figura 25. Celda de control.

8.2. Escenario 2. Planta de prensado de láminas de acero para puertas de automóviles

A diferencia del primer escenario que representa una línea de producción individual, en este segundo escenario se ha buscado representar una planta de producción completa que incorpora varias líneas de fabricación en paralelo. El escenario escogido corresponde a una planta de producción cuyo objetivo es el prensado de láminas de acero para la producción de puertas de

automóviles. Este escenario se basa en un diseño o *layout* previamente implementado en *Visual Components* y que representa una planta real. Por temas de confidencialidad, *Visual Components* no aporta información identificativa sobre la planta industrial concreta representada en este *layout*.

Es importante destacar que, aunque *Visual Components* ya incorpora este *layout* en su catálogo, simplemente se ha tomado como referencia de modelo de industria real dada su complejidad y posibilidades de actualización a Industria 4.0. Para el presente proyecto se ha desarrollado un *layout* desde cero añadiendo funciones adicionales como los almacenes automáticos o el transporte de productos mediante AGVs con el objetivo de representar una planta completa. La decisión de modelizarlo desde cero, a pesar de que ya existiese parte de él en el catálogo del software, se ha tomado en base a la complejidad del diseño y la necesidad de conocer al detalle cómo se implementa la interacción entre los distintos elementos que componen el escenario y el flujo de trabajo.

La Figura 26 muestra una imagen de la planta de prensado modelada en *Visual Components*. La planta está dividida en tres zonas bien diferenciadas (identificadas en la Figura 27):

1. Almacén de entrada con gestión autónoma de stock. En el almacén de entrada se lleva a cabo la recepción y almacenaje de la materia prima, en este caso, planchas de acero. El almacén de entrada dispone de un controlador local de almacén que conoce y gestiona en tiempo real el estado de stock del almacén. Desde el almacén de entrada, una serie de AGVs se encargan de transportar el material necesario hasta el inicio de las líneas de prensado en las que se dará la forma requerida a las planchas.
2. Celda de prensado. Para dar la forma requerida a las planchas en base al diseño del producto final, las planchas de acero se someten a un proceso de prensado a través de una serie de prensas. En concreto, la celda de prensado en la que se lleva a cabo este proceso está formada por 3 líneas de fabricación en paralelo. La inyección de material a cada línea, así como el traspaso de las planchas de una prensa a otra, se realiza mediante brazos robóticos de manera autónoma. Tras el proceso de prensado, se realiza un control de calidad de la plancha ya procesada (con la forma requerida). Las planchas con defectos de producción son descartadas mediante un brazo robot, mientras que las que cumplen los criterios de calidad continúan hasta el final de la cinta transportadora donde son depositadas por operarios en pallets para ser trasportadas mediante una carretilla elevadora al almacén de salida.
3. Almacén de salida con gestión autónoma. En el almacén de salida se lleva a cabo la recogida y almacenaje del producto finalizado. Al igual que en el almacén de entrada, la gestión de los productos en el almacén se lleva a cabo de manera autónoma mediante un gestor local del almacén.

Además, la planta incorpora un sistema de monitorización central que recoge datos del estado, funcionamiento y operación de la celda de prensado y de los almacenes de entrada y de salida. Este sistema de monitorización proporciona una visión global del funcionamiento y operación de la celda y permite la identificación de problemas o ineficiencias, así como sus causas.

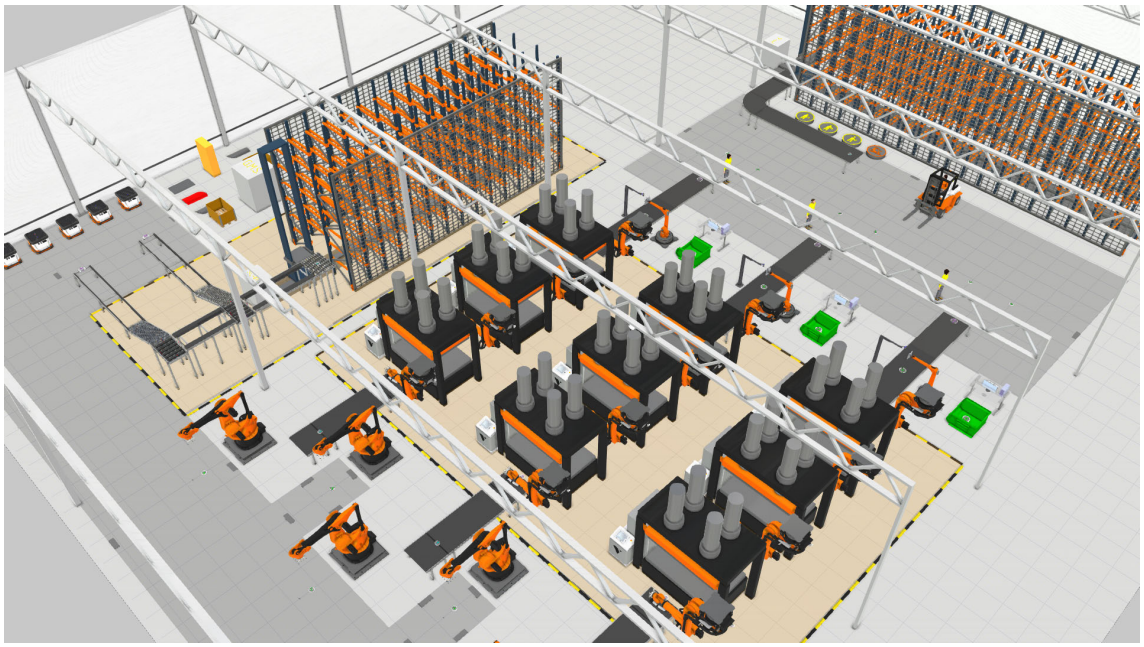


Figura 26. Representación del escenario 2 - Planta de prensado de láminas de acero para puertas de automóviles.

En la Figura 27 se observa la distribución en planta. El cuadro (1) muestra el almacén de entrada, el cual está conectado mediante un pasillo adaptado para los AGVs a la celda de prensado del cuadro (2). Una vez se le ha dado forma al material es llevado al almacén final, representado en el cuadro (3).

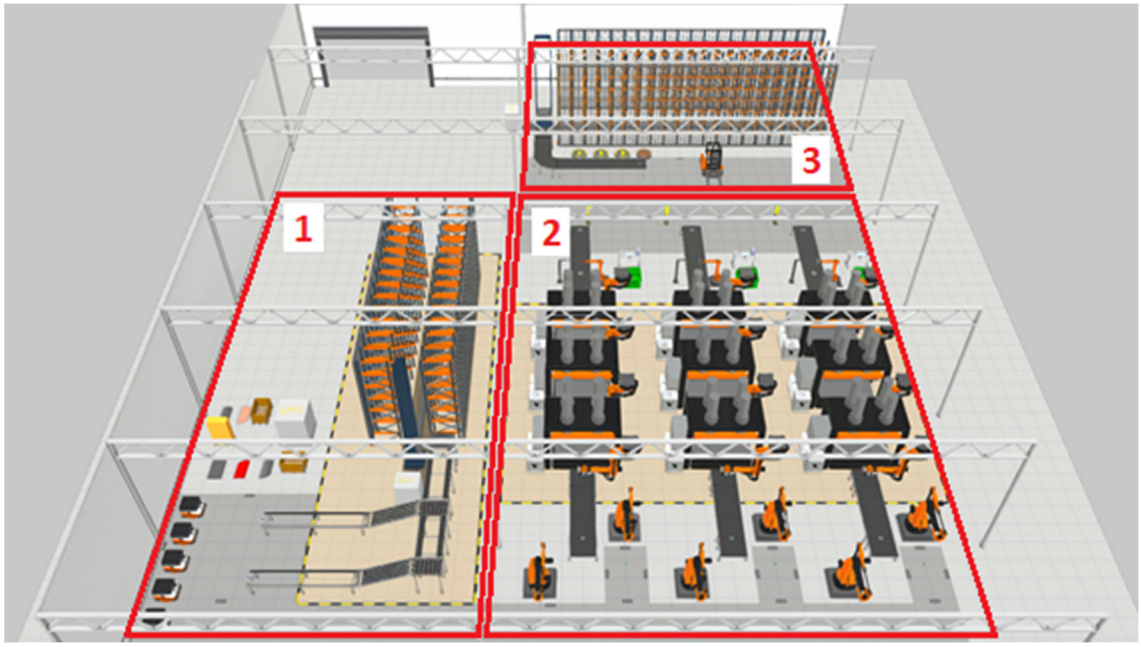


Figura *27*. Celdas que forman la planta de prensado de láminas de acero para puertas de automóviles (Escenario 2).

8.2.1. Almacén de entrada con gestión autónoma de stock

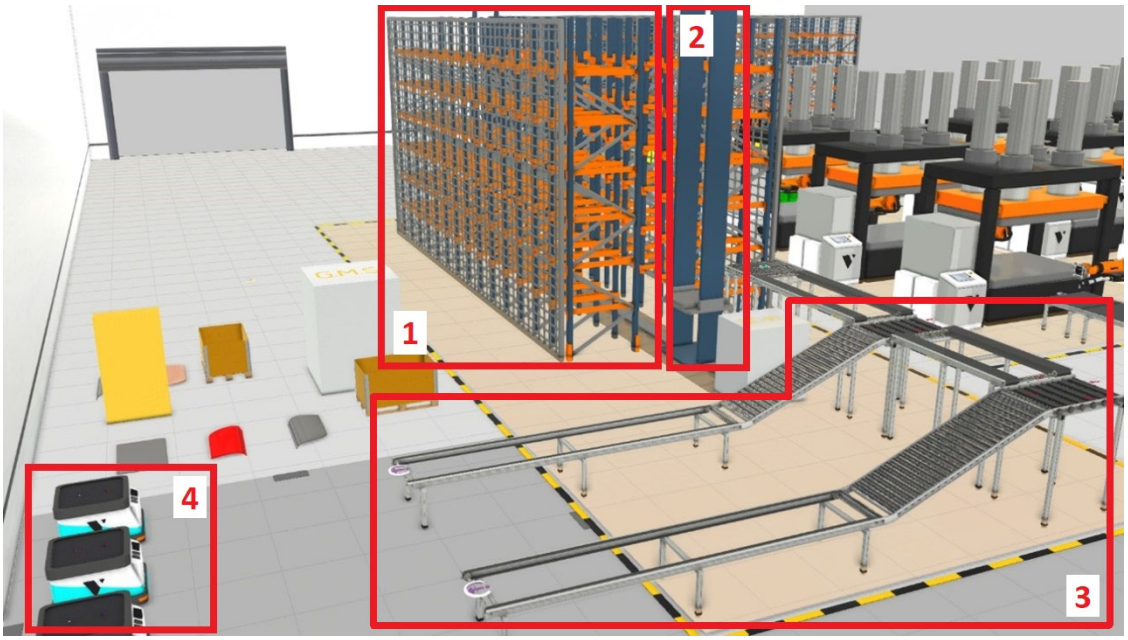
El almacén de entrada realiza la recogida y almacenaje de las planchas de acero, que son la materia prima utilizada en esta planta industrial. Este almacén consta de varios elementos enumerados en el párrafo siguiente, pero predominan dos estanterías metálicas con dimensiones de 13 x 6,4 m, las cuales cuentan cada una con 6 bahías (celdas individuales de almacenaje) de altura y 14 de largo, lo que le otorga una capacidad total para almacenar 168 pallets.

La Figura 28 muestra una vista general del almacén de entrada en la que se pueden observar los diferentes elementos que lo integran. Estos elementos son:

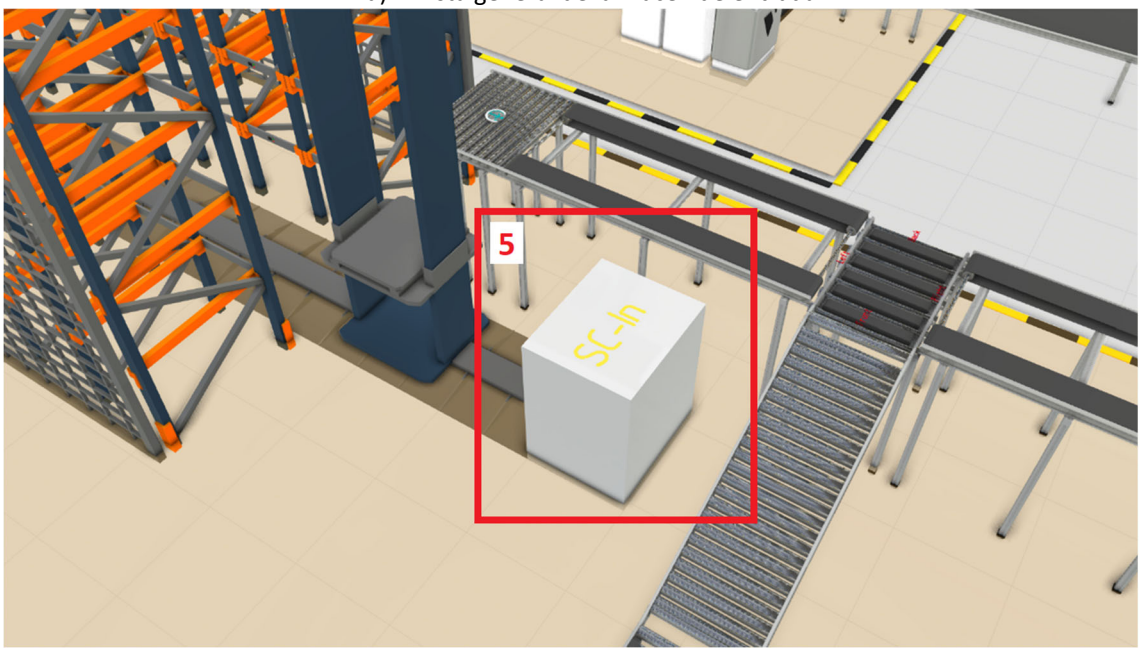
- Estanterías en las que se deposita el material de entrada identificadas en el recuadro con numeración (1).
- Grúa que recoge el material de la estantería para situarlo en las cintas transportadoras (identificadas en el recuadro con numeración (2)).
- Cinta transportadora que traslada el material depositado por la grúa hasta el punto de recogida de los AGVs (recuadro (3)).
- AGVs encargados de transportar el material desde el almacén hasta el comienzo de las distintas líneas de prensado (recuadro (4)).
- Controlador local de almacén que conoce y gestiona en tiempo real el material disponible en el almacén para cubrir las necesidades de material de entrada a la celda de prensado. Además, mantiene sincronizados y operativos los AGVs (recuadro (5)).

El gestor local del almacén es una aplicación software instalada en un servidor junto a la grúa (Figura 28.b). Este está compuesto por dos módulos software: el gestor del material y el controlador de los AGVs. El gestor del material conoce y gestiona en tiempo real el material disponible en el almacén para cubrir las necesidades de material de entrada a la celda de prensado. El controlador de los AGVs tiene conocimiento del estado, posición y tareas de los distintos AGVs en todo momento. En base a esta información, identifica y notifica al AGV que debe realizar cada transporte de material necesario desde el almacén de entrada hasta la línea de prensado correspondiente.

Cuando se produce la entrada de material en el almacén, éste es depositado en las bahías de las estanterías, las cuales poseen un tamaño adecuado para los pallets de 1,2 x 1 m. Las estanterías incorporan en cada bahía un sensor (Figura 29) que indica si está ocupada o no. Cada vez que algún sensor de estas baldas sufre un cambio de valor, se notifica al gestor local del almacén mediante un mensaje donde se le indica el cambio que ha sufrido dicho sensor. Cuando el gestor local del almacén recibe una solicitud de material desde una línea de prensado, envía la orden a la grúa para que saque material. Este material será transportado por los AGVs hasta la línea de prensado que ha solicitado material. El AGV se mueve a través de un pasillo reservado para ello (Figura 30), hasta los puntos de descarga de material, que se encuentra al inicio de cada línea de prensado (señalados en rojo en la Figura 30). De esa forma, la materia prima estará disponible para que el brazo robot correspondiente lo coja para iniciar el proceso de prensado.



a) Vista general del almacén de entrada.



b) Gestor local del almacén.

Figura 28. Almacén de entrada del escenario 2.

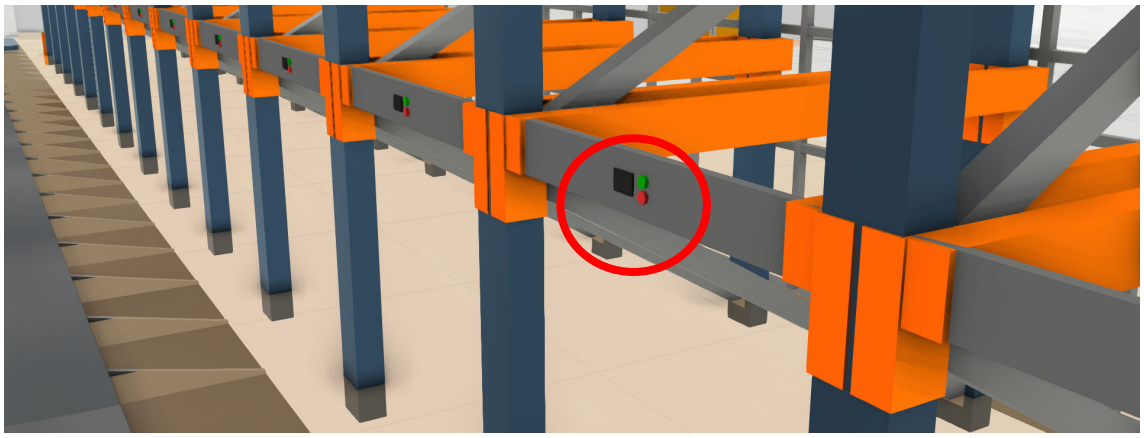


Figura 29. Sensores de las celdas del almacén automático.

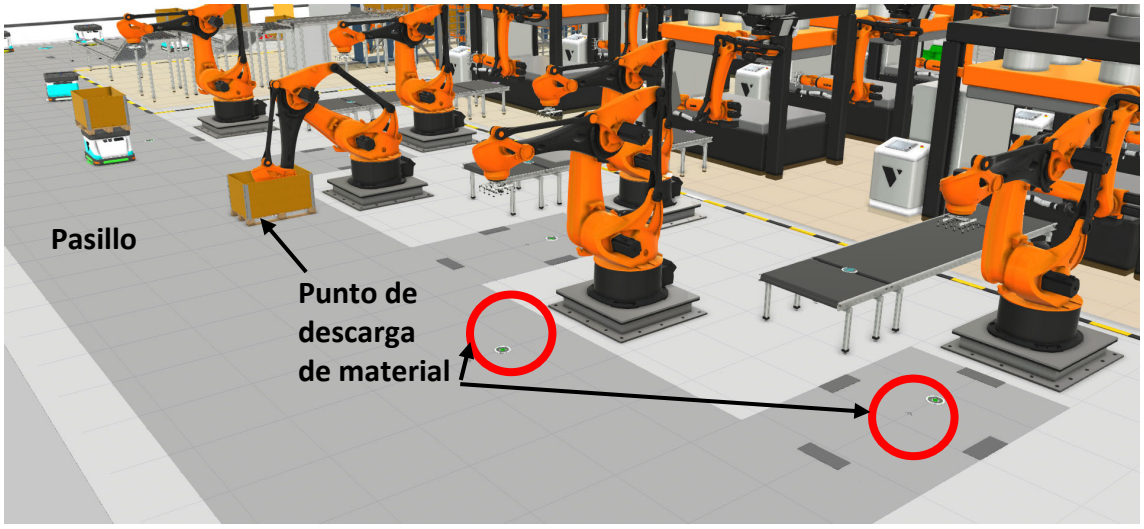


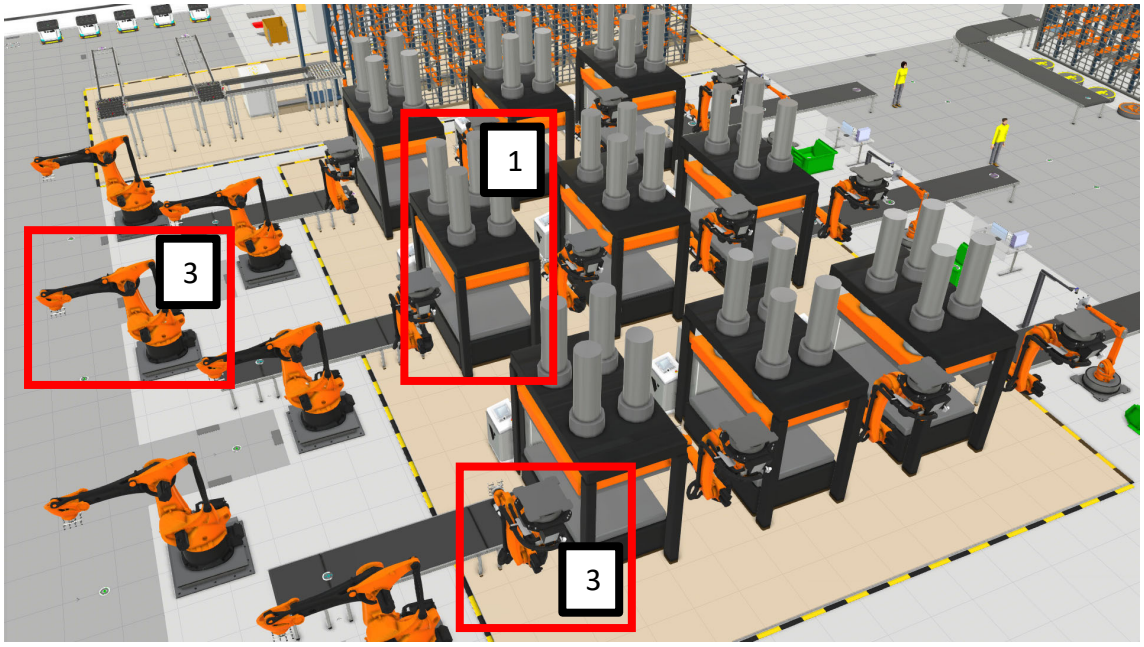
Figura 30. Área de operación de los AGVs.

8.2.2. Celda de prensado

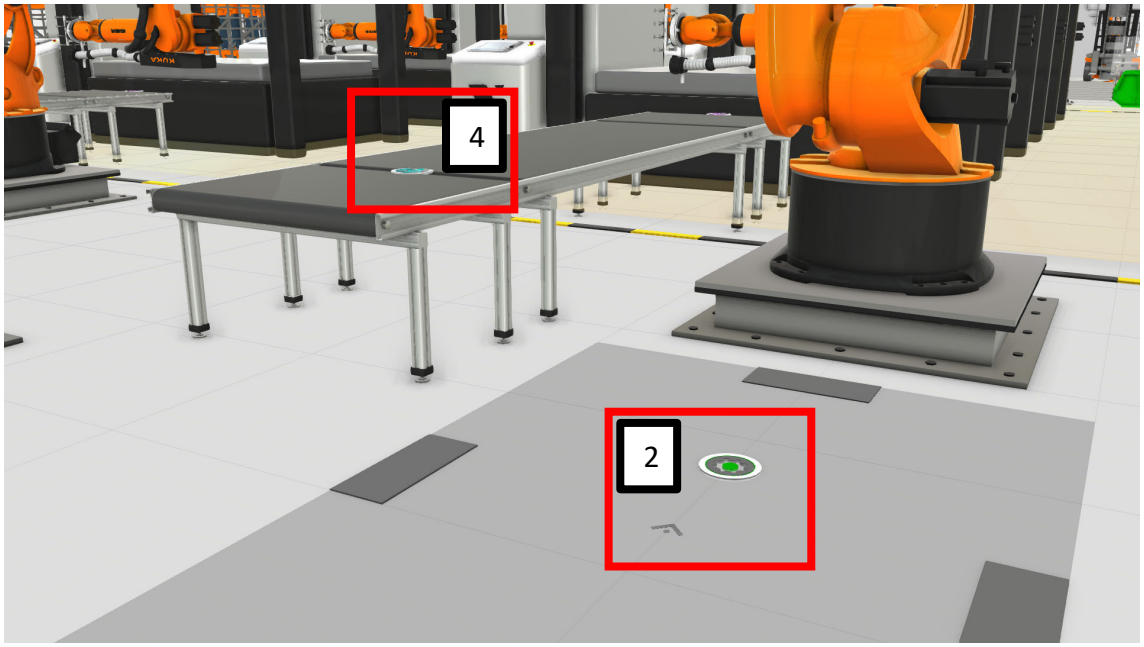
La celda de prensado recibe las planchas de acero provenientes del almacén para ser prensadas y convertidas en puertas de automóviles. La celda consta de 3 líneas de prensado idénticas que trabajan en paralelo. La Figura 31 muestra las líneas de prensado. Cada línea está compuesta por los siguientes elementos:

- 3 prensas en las que se les da forma a las planchas de acero (recuadro (1) en Figura 31). Las prensas tienen unas dimensiones de 4 x 5 x 2 m, dotadas de un espacio interior adecuado al material con el que se trabaja.
- Puntos de descarga en los que los AGVs provenientes del almacén depositan los pallets con el material a una distancia operativa para los robots. Estos puntos poseen sensores que se encargan de avisar al gestor local del almacén si hace falta más material y mantener informados a los controladores de los robots de esta celda, condición necesaria para que estos actúen o no (recuadro (2) en Figura 31).
- Brazos robóticos que se encargan de mover el producto de una prensa a la siguiente (recuadro (3) en Figura 31).

- Cintas transportadoras que dirigen el material hasta el siguiente punto de trabajo. Tiene unos sensores al inicio y al final que son los encargados de indicar a los controladores de los robots si hay o no material (recuadro (4) en Figura 31).



a) Vista general de la celda de prensado.



b) Punto de descarga de material para cada línea de prensado.

Figura 31. Celda de prensado en el escenario 2.

Al inicio de cada línea de prensado, se sitúan 2 brazos robóticos que se encargan de introducir el material en la prensa (ver Figura 32). Estos 2 brazos robóticos trabajan de manera sincronizada. Para lograr esta sincronización, los robots intercambian información de su posición

de forma periódica. De esta manera tienen la capacidad de decidir cuándo recogen material y lo depositan en la cinta.

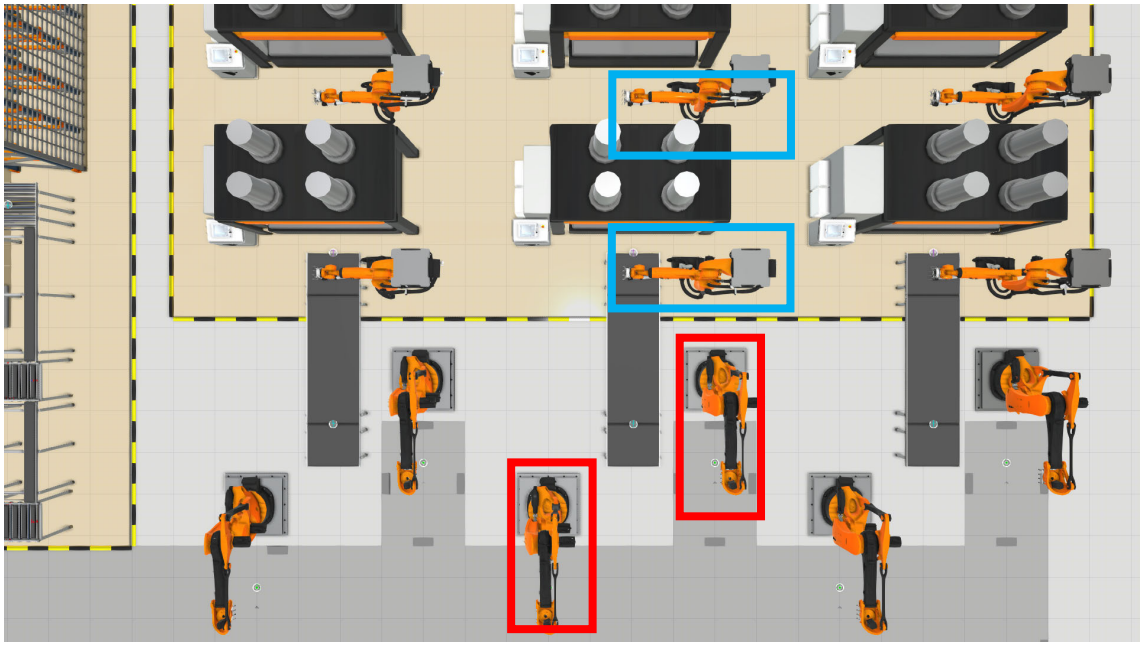


Figura 32. Vista de planta de la celda de prensado. Enmarcados en rojo, los robots que depositan planchas en la cinta en la línea central. En azul, los robots que mueven las planchas entre prensas en la línea central.

Cuando las planchas depositadas en la cinta llegan al final de esta, el sensor ubicado al final (Figura 33) enviará una señal al siguiente robot que recoge el material y lo introduce en la prensa (ver Figura 32 y Figura 34).

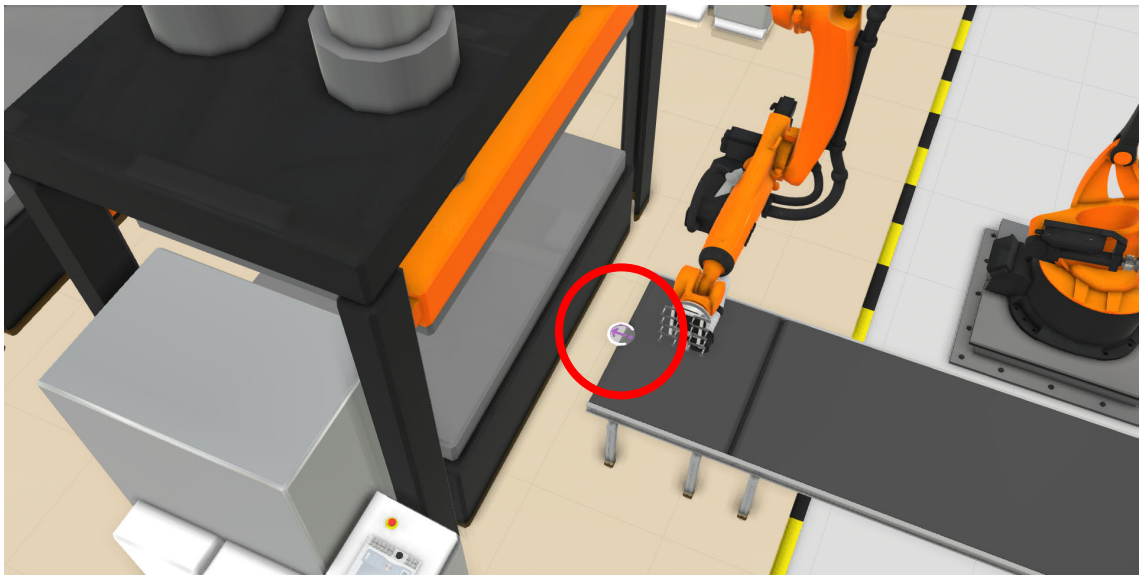


Figura 33. Sensor al final de la cinta transportadora.

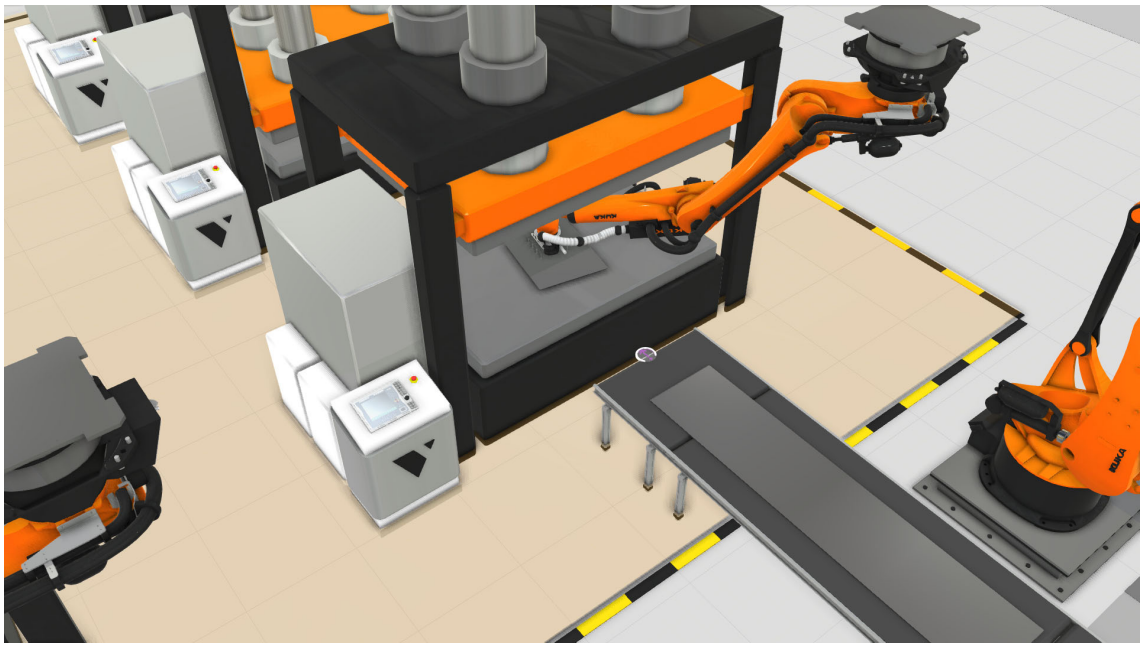


Figura 34. Robot introduciendo una plancha en la primera prensa.

Una vez introducido, la prensa detecta, mediante sus propios sensores, el material e inicia el proceso que demora 3 segundos de bajada, 1 segundo de prensado y finalmente otros 3 de subida, volviendo a la posición inicial. Finalizado el primer prensado, se sucede este proceso 2 veces más en las otras 2 prensas restantes. La prensa libera el producto y envía la señal al controlador del brazo robot siguiente para que la extraiga y la introduzca directamente en la segunda prensa (Figura 35).

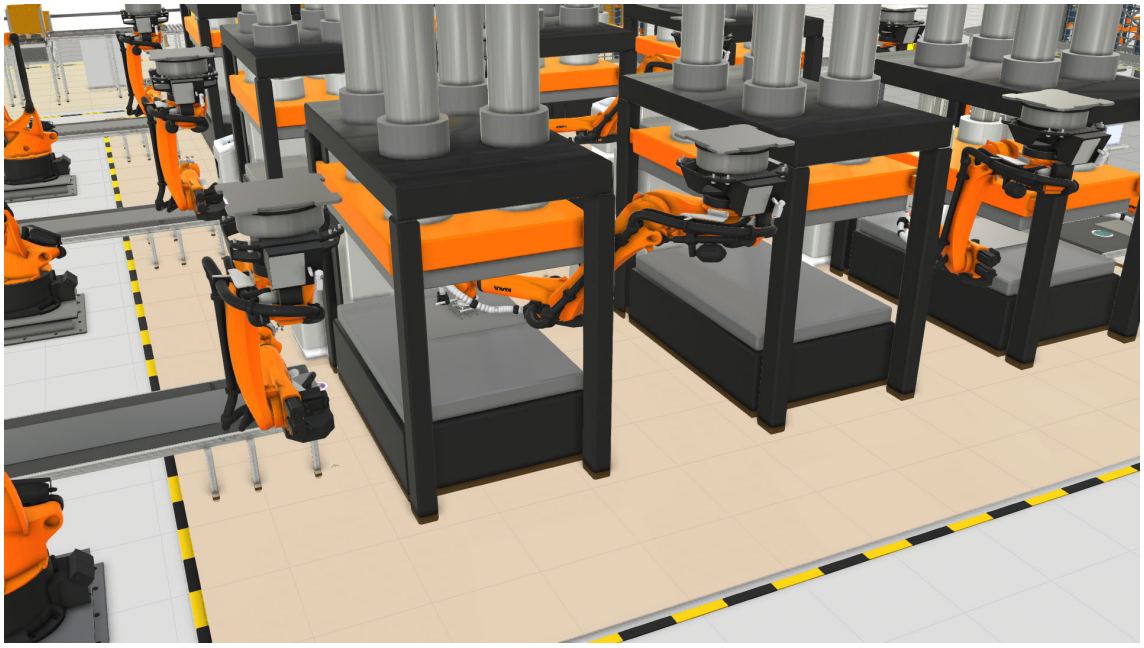


Figura 35. Robot sacando una plancha tras el primer prensado para introducirla en la segunda prensa.

Al final de cada línea de prensado se realiza un control de calidad en el cual se desechan las piezas que presenten defectos. A la salida de la tercera prensa, el producto ya acabado es

colocado en la cinta transportadora que lo llevará a la zona de control de calidad (Figura 36). El control de calidad durante la producción es de suma importancia, ya que permite encontrar de forma rápida y con precisión fallos en los procesos a los que se somete el material. Sin este control, un porcentaje de las piezas que salen de las prensas seguirían adelante en la cadena, a pesar de presentar defectos, y pueden poner en riesgo la integridad del producto final.

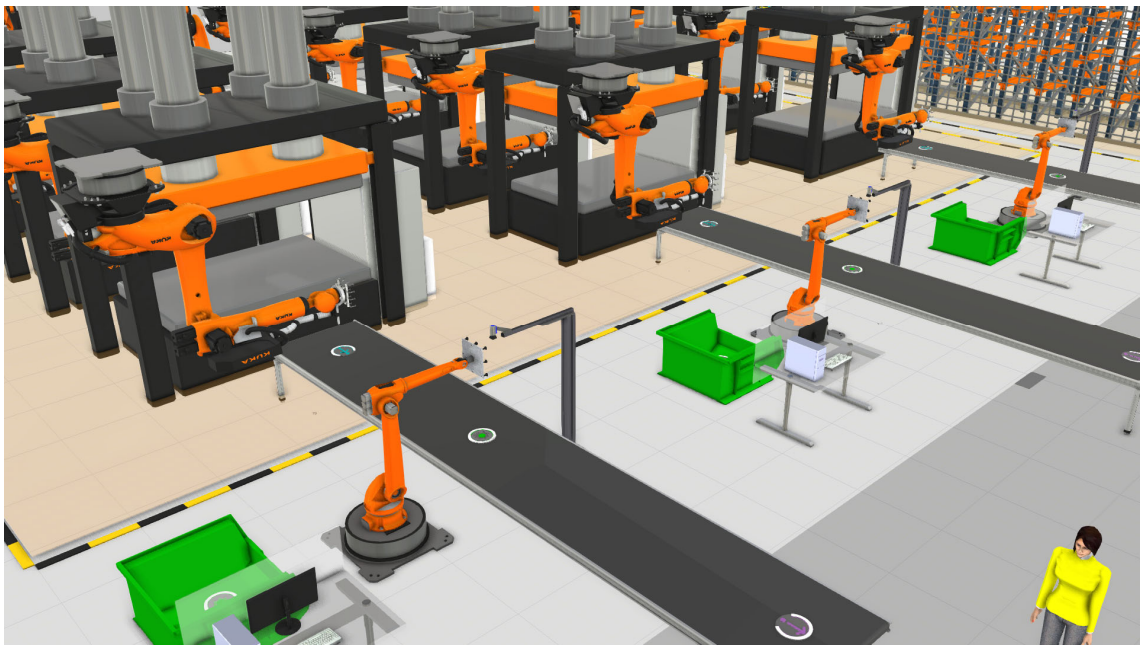


Figura 36. Salida de las prensas e inicio del control de calidad.

La Figura 37 muestra la zona en la que se realiza el control de calidad y se identifica los distintos elementos que participan en este proceso. Estos elementos son:

- Cintas transportadoras, que van trasladando el producto a lo largo del proceso de control de calidad hasta llegar al operario humano. Estas cintas, también poseen sensores al inicio y al final para detectar si hay producto o no en ellas (recuadro (1) en Figura 37).
- Cámaras detectoras de fisuras y pequeños defectos (recuadro (2) en Figura 37).
- Brazos robots encargados de coger los productos defectuosos y sacarlos de la línea de producción (recuadro (3) en Figura 37).
- Contenedores para los productos defectuosos (recuadro (4) en Figura 37).
- Operarios cuya tarea es poner los productos sin defectos en pallets (recuadro (5) en Figura 37).

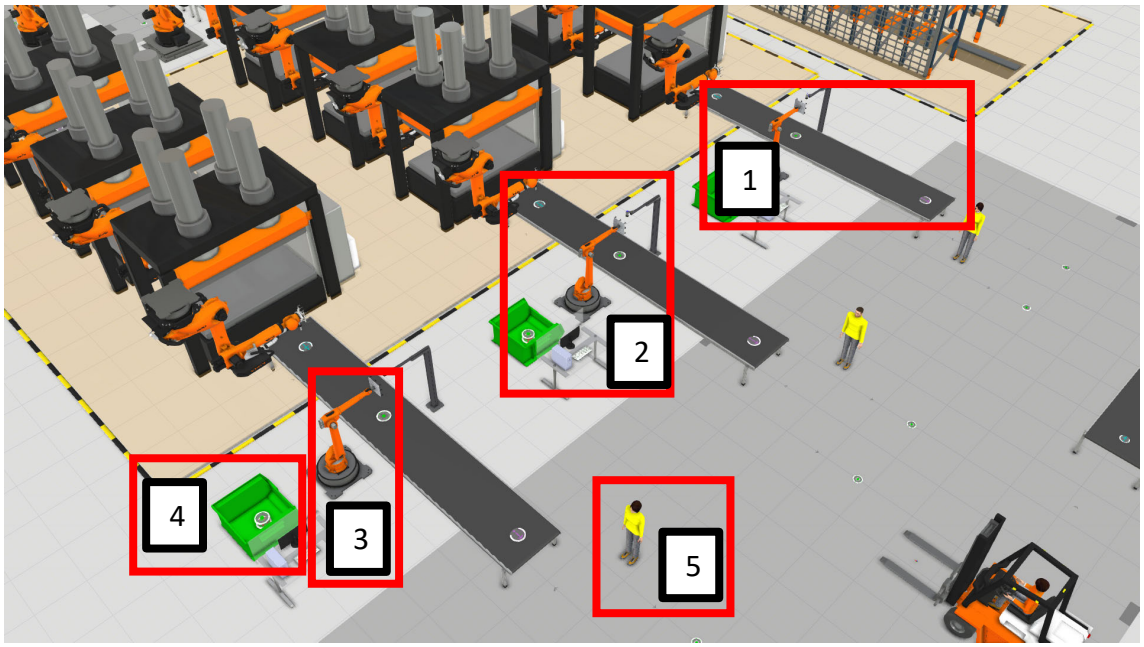


Figura 37. Vista general del proceso de control de calidad.

Continuando el flujo de trabajo a la salida de las prensas, cuando el último robot deposita el producto en las cintas este acaba llegando al nodo de control señalado en la Figura 38. Este nodo tiene un sensor para identificar que el producto ha llegado, parar el movimiento de la cinta y avisar a la cámara del control de calidad (Figura 39) para empezar a tomar imágenes del producto para su procesado, y así encontrar defectos si los tuviera. Una vez acabada la revisión, se envía una señal al brazo robot que tiene en frente para retirar el producto si el resultado del control de calidad es negativo. En ese caso, el producto se desecha en el contenedor verde (Figura 40). Cuando el brazo levanta el producto de la cinta, el nodo detecta la no presencia del mismo y vuelve a activar el movimiento de la cinta. En caso contrario, si el resultado de la revisión es favorable, se continúa con el proceso.

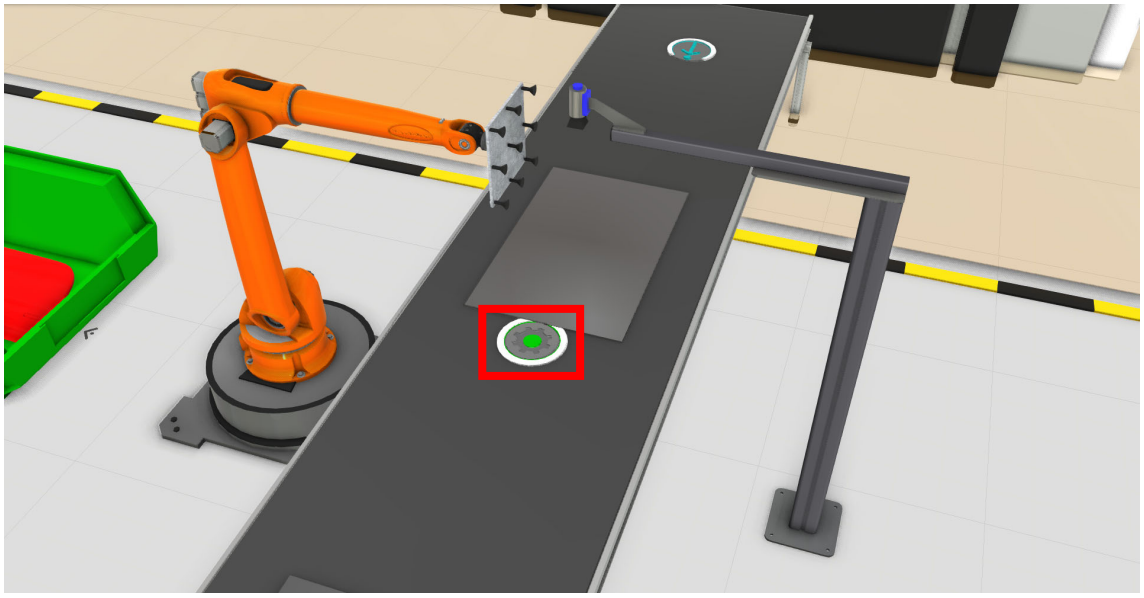


Figura 38. Nodo de control de calidad

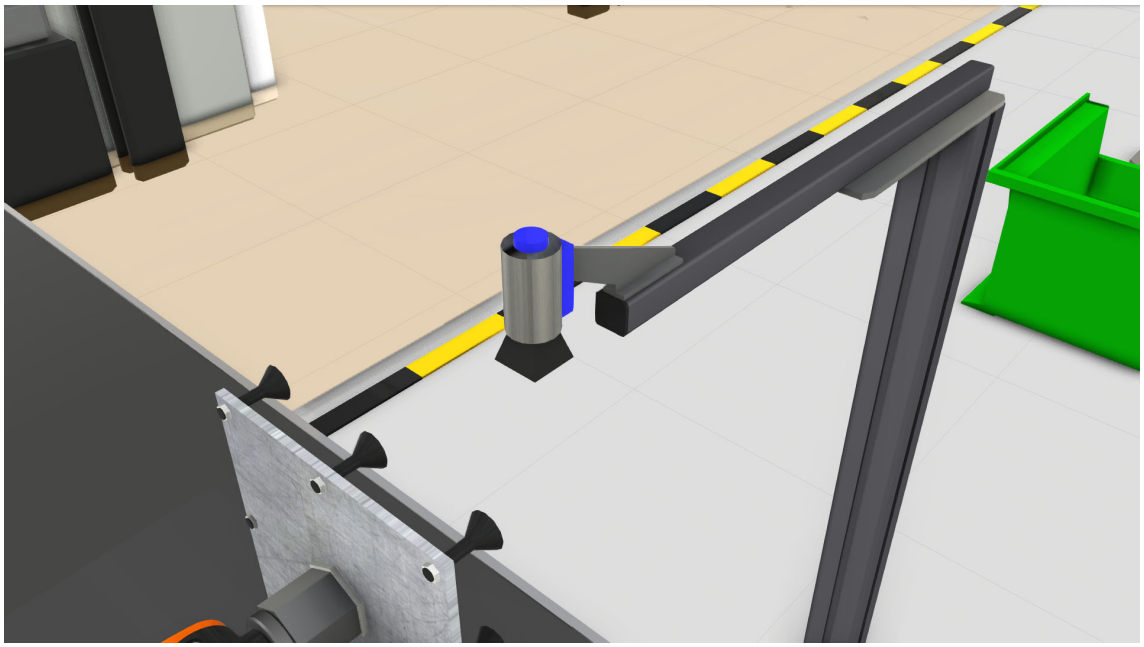


Figura 39. Cámara detectora de fisuras y pequeños defectos.

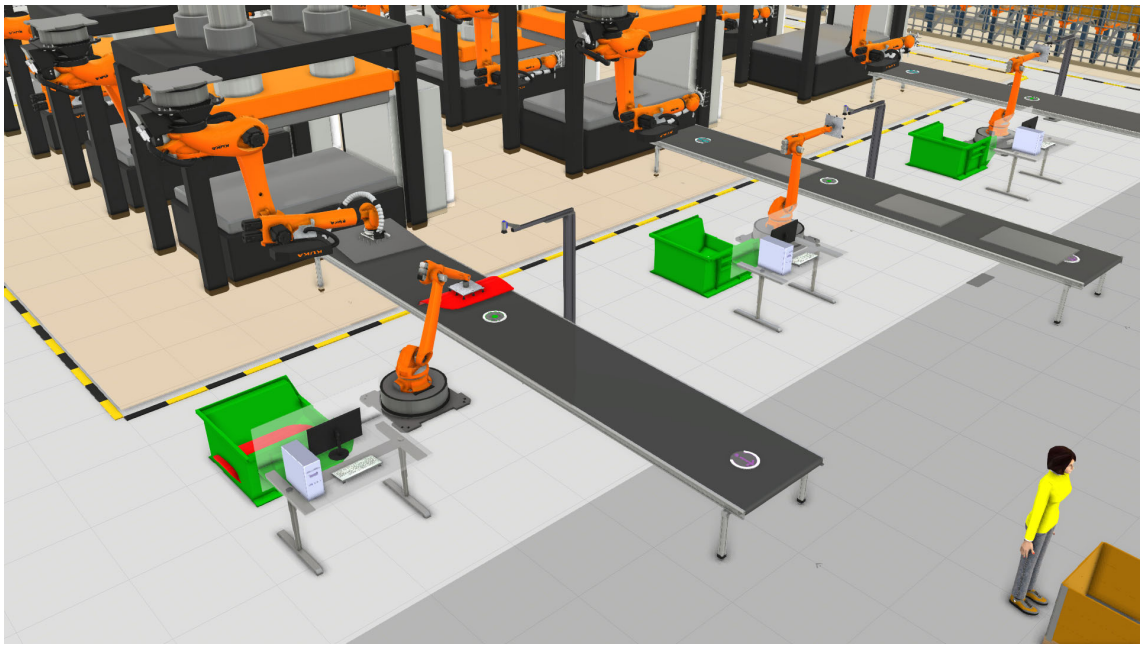


Figura 40. Una de las líneas desecha un producto por presentar defectos. El producto defectuoso es marcado es rojo.

Si el producto pasa de manera exitosa el control de calidad, continúa por la cienta hasta el final, donde un operario la retira y la deposita en un pallet tal y como se representa en la Figura 41 y Figura 42. Cuando el pallet esté lleno, la carretilla elevadora controlada por otro operario lo recogerá para llevarlo, ahora sí, a la celda final de la línea de producción: el almacén de salida con gestión autónoma (Figura 43).

Figura 41. Operario retira el producto de la cinta transportadora para paletizarlo.

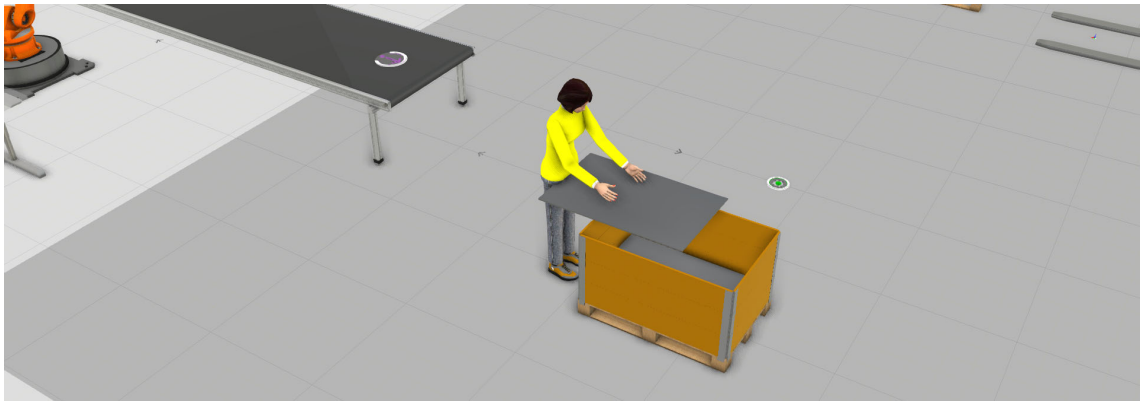


Figura 42. Operario depositando el producto en el pallet final.

Figura 43. Pallet llevado al almacén de salida.

8.2.3. Almacén de salida con gestión autónoma

Por último, y como ya se ha mencionado anteriormente, se encuentra el almacén de salida el producto final. La Figura 44 muestra una vista general del almacén de salida en la que se puede observar los diferentes elementos que lo integran. Estos elementos son:

- 2 estanterías en las que se deposita el producto finalizado. Cada estantería tiene unas dimensiones de 18,5 x 6,4 metros, las cuales cuentan cada una con 6 bahías (celdas individuales de almacenaje) de altura y 20 de largo, lo que le otorga una capacidad total para almacenar 240 pallets.
- Grúa que recoge el material de la estantería para situarlo en las cintas transportadoras.
- Cinta transportadora que dirige el material depositado por la carretilla elevadora hasta la grúa del almacén.
- Carretilla elevadora operada por un operador.
- Controlador de almacén local que conoce y gestiona en tiempo real la cantidad de producto almacenado y el espacio libre disponible.

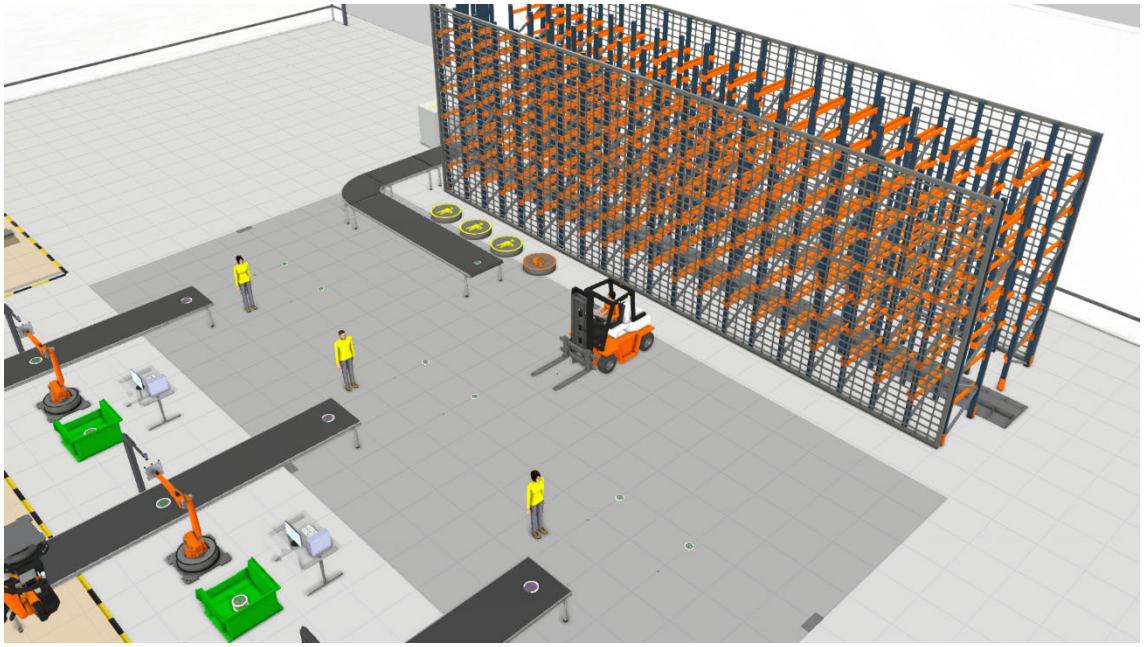


Figura 44. Vista general del almacén de salida con gestión autónoma del escenario 2.

El flujo de trabajo en esta celda es el siguiente. El operario con la carretilla deposita los palets con el producto finalizado en la cinta transportadora. Gracias a los sensores que incorpora al inicio de la cinta (Figura 45), ésta se activa cuando el producto se deposita sobre ella. de esta planta de fabricación finaliza aquí, en esta celda. A través de la cinta, el producto se dirige hacia la grúa del almacén para situarlos en la estantería.

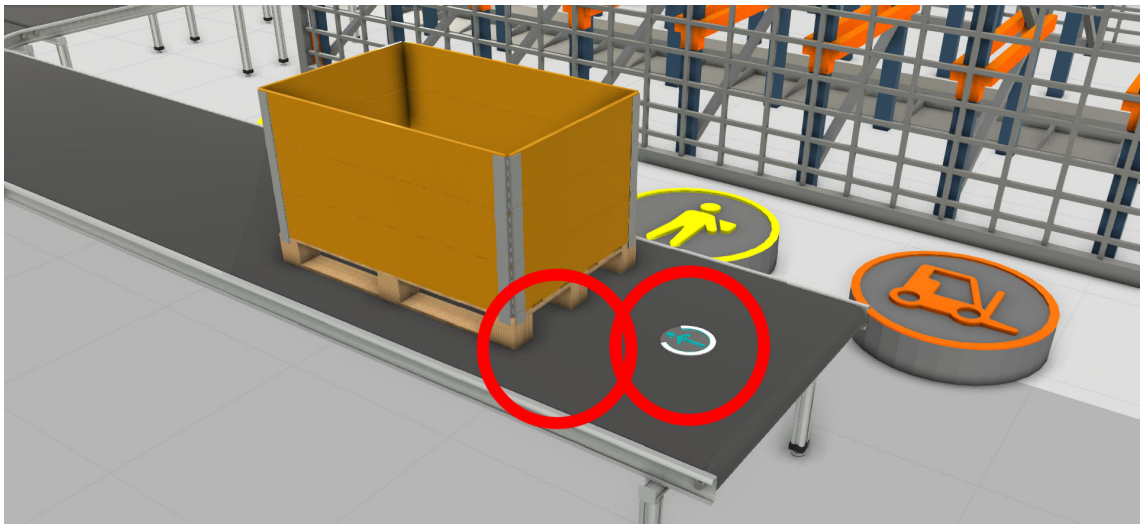


Figura 45. Sensor de la cinta transportadora.

8.2.4. Sistema de monitorización central

El sistema de monitorización central (mostrado en la Figura 46) recibe información sobre el funcionamiento y operación de los procesos que se llevan a cabo en las distintas celdas de la fábrica. Todos esos datos se envían a un servidor situado en una pequeña habitación junto al AS/RS de entrada, donde se encuentra un operario para supervisar y monitorizar los datos recibidos.

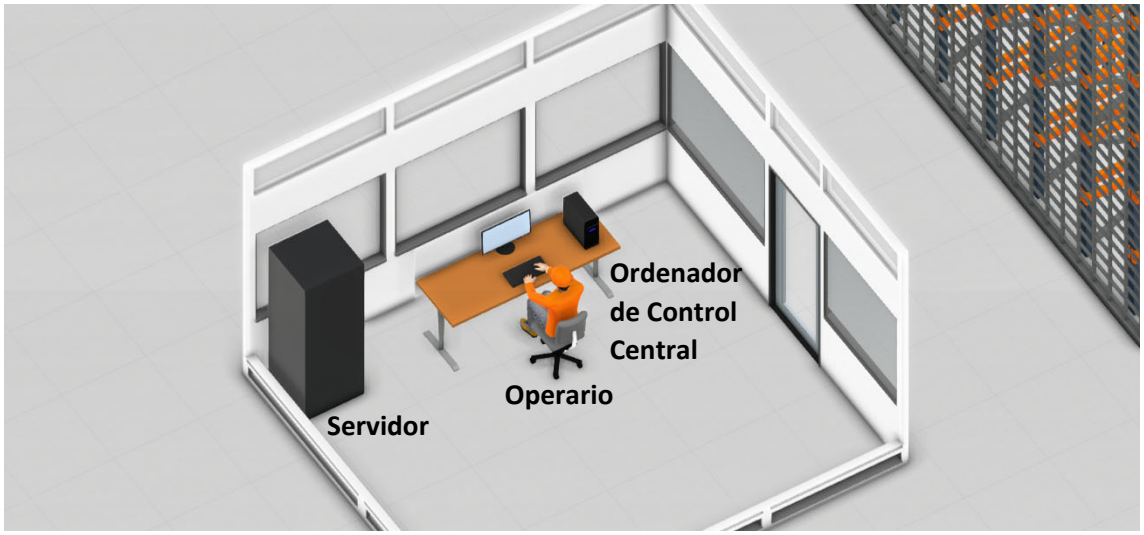


Figura 46. Sistema de monitorización central.

9. Comunicaciones inalámbricas en escenarios industriales

En este apartado se describen las distintas comunicaciones que tienen lugar entre los distintos elementos que componen cada escenario industrial. En concreto, este apartado se centra en las comunicaciones que se llevarían a cabo de forman inalámbrica utilizando potencialmente 5G. Para cada comunicación se especifica el origen o fuente y el destino de la comunicación, la cantidad de datos a transmitir, si el mensaje se envía de forma periódica o no, la periodicidad si es el caso, y los requisitos de latencia y fiabilidad de la comunicación. Los valores para los distintos parámetros que caracterizan el tráfico (tamaño de datos a transmitir, latencia, fiabilidad, etc.) se han establecido en base a los modelos de tráfico definidos en [1] y [14].

9.1. Comunicaciones inalámbricas en el escenario 1

Como se ha presentado en el apartado 8.1, en el escenario 1 'Línea de prensado de láminas de acero con control de calidad basado en telemetría y telecontrol' se implementa una línea de prensado de láminas de acero en la que el producto pasa de una celda a otra en la que se le aplican distintos procesos. Para el transporte del producto de una celda a otra se utilizan robots móviles o AGVs. Para eliminar, o minimizar lo máximo posible, los tiempos de espera en estos transportes del producto, los distintos procesos deben estar coordinados con los AGVs. Para ello, debe haber una comunicación fiable y robusta entre los distintos elementos (maquinaria y controladores) de la línea con los AGVs. Además, los distintos procesos envían información a un centro de control en el cual se monitoriza el correcto funcionamiento de los mismos. Por último, el proceso de control de calidad basado en telemetría y telecontrol que se realiza al final de la línea, conlleva el intercambio continuo de datos entre el CMM, el ordenador de control remoto y el *edge* en el que se realiza las tareas de computación, para lo que se necesita una conexión que además de fiable y robusta, proporcione bajas latencias de transmisión.

A continuación, se presentan los datos que se generan en los distintos elementos (controladores, maquinaria, robots, etc.) que componen el escenario 1 y que deben transmitirse de manera inalámbrica, particularmente utilizando tecnología 5G. Para cada comunicación se especifica la siguiente información:

- Mensaje #: Identificador empleado para cada tipo de mensaje.
- Origen → Destino: El origen identifica al elemento que crea y emite el mensaje, y el destino identifica al elemento que recibe el mensaje.
- Tamaño: Número de bytes de datos que ocupa el mensaje.
- Tipo: Indica si el mensaje es periódico o aperiódico.
- Periodo: En caso de ser un mensaje periódico, indica la periodicidad en milisegundos.
- Latencia: Tiempo máximo en el que el mensaje debe ser transmitido entre el origen y el destino.
- Fiabilidad: Indica el porcentaje mínimo de mensajes que deben ser transmitidos correctamente entre el origen y el destino. Si el mensaje tiene requisitos de latencia, los mensajes deben transmitirse por debajo de la latencia máxima establecida.

9.1.1. Comunicaciones para la gestión de los transportes de piezas por los AGVs

El controlador del AGV (o en el futuro de los AGVs) se implementa en el ordenador central de control. Por lo tanto, el ordenador central de control y el AGV se comunican a través de una

conexión inalámbrica. A continuación, se presentan los distintos mensajes que se intercambian entre ellos:

1. Solicitud de transporte. Cuando el proceso de prensado/soldadura ha finalizado y el ordenador central de control es informado de ello, solicita a un AGV que realice el transporte de la pieza hasta la siguiente celda.
    - Mensaje 1: 'Solicitud de transporte a AGV' (se le indica al robot la coordenada a la que debe desplazarse y la orientación con la que se debe situar en el destino)
      PC Control → AGV
      Tamaño: 128 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%
    - Mensaje 2: 'ACK'
      AGV → PC Control
      Tamaño: 64 bytes. Estos paquetes suelen tener un tamaño muy pequeño, de unos pocos bytes. Sin embargo, se asume que se transmiten como tramas L2 con un tamaño mínimo de 64 bytes. Esto es aplicable para todas las transmisiones.
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%
2. Informe periódico del estado, posición y orientación del AGV. El AGV informa de manera periódica al ordenador central de control sobre su estado, nivel de batería, si está ocupado realizando un transporte, si lleva carga sobre él, su posición y orientación.
    - Mensaje 3: 'Informe de estado del AGV'
      AGV → PC Control
      Tamaño: 250 bytes
      Tipo: periódico, determinista
      Periodo: 250 ms
      Latencia: < periodo
      Fiabilidad: 99.9%
3. Transporte finalizado. Cuando el AGV llega a su destino, envía un mensaje al PC de control para informarle.
    - Mensaje 4: 'Informe de transporte finalizado'
      AGV → PC Control
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%
    - Mensaje 5: 'ACK'
      PC Control → AGV
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%

9.1.2. Comunicaciones en la celda de prensado

En la celda de prensado tienen lugar el siguiente intercambio de información entre máquinas, dispositivos y nodos de computación a través de comunicaciones inalámbricas. En primer lugar, la prensa comunica de manera inalámbrica con el ordenador central de control situado en la Sala de Control. La información intercambiada es la siguiente:

1. Inicio del proceso de prensado. Cuando el operario sitúa la plancha en la prensa, este presiona la orden de inicio del proceso de prensado utilizando un HMI instalado en la prensa. Esta acción envía un mensaje al PC de Control que debe contestar con un ACK a dicha orden.
    - Mensaje 6: 'Inicio Proceso Prensado'
      Prensa → PC Control
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%
    - Mensaje 7: 'ACK'
      PC Control → Prensa
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%
2. Fin del proceso de prensado. Cuando el proceso finaliza, la prensa informa al PC de Control. En este momento, el PC de control solicita a un AGV que se desplace a recoger la pieza.
    - Mensaje 8: 'Fin Proceso Prensado'
      Prensa → PC Control
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%
    - Mensaje 9: 'ACK'
      PC Control → Robot de soldadura
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%

9.1.3. Comunicaciones en la celda de soldadura

En esta celda, el robot que se encarga de realizar la soldadura comunica con el ordenador central de control. Esta interacción es muy similar a la realizada entre la máquina de prensado y el ordenador central de control. La información intercambiada entre el robot de soldadura y el ordenador central de control es la siguiente:

1. Inicio del proceso de soldadura. Cuando el operario sitúa la plancha en el robot, este presiona la orden de inicio del proceso de soldadura utilizando un HMI instalado junto al robot. Esta acción envía un mensaje al PC de Control que debe contestar con un ACK a dicha orden.

- Mensaje 10: ‘Inicio Proceso Soldadura’
  Robot soldadura → PC Control
  Tamaño: 64 bytes
  Tipo: aperiódico (basado en eventos), no determinista
  Latencia: 500 ms
  Fiabilidad: 99%
- Mensaje 11: ‘ACK’
  PC Control → Robot de soldadura
  Tamaño: 64 bytes
  Tipo: aperiódico (basado en eventos), no determinista
  Latencia: 500 ms
  Fiabilidad: 99%

2. Fin del proceso de soldadura. Cuando el proceso finaliza, el robot de soldadura informa al ordenador central de control. En este momento, el ordenador central de control solicita a un AGV que se desplace a recoger la pieza.
   - Mensaje 12: ‘Fin Proceso Soldadura’
     Robot soldadura → PC Control
     Tamaño: 64 bytes
     Tipo: aperiódico (basado en eventos), no determinista
     Latencia: 500 ms
     Fiabilidad: 99%
   - Mensaje 13: ‘ACK’
     PC Control → Robot de soldadura
     Tamaño: 64 bytes
     Tipo: aperiódico (basado en eventos), no determinista
     Latencia: 500 ms
     Fiabilidad: 99%

9.1.4. Comunicaciones en la celda de telemetría y control de calidad

El AGV transporta el producto hasta la CMM. El operario pone la pieza sobre el CMM y desde el ordenador central de control se inicia el proceso de medida. En esta celda, las distintas máquinas, PCs y controladores se comunicación de la siguiente forma. Actualmente la comunicación entre el Edge y el CMM es cableada, por lo que no se incluye en este apartado.

1. Comunicación entre el CMM y el ordenador central de control. Cuando la pieza se sitúa en el CMM, el CMM avisa al ordenador central de control de que está listo para iniciar el proceso:
   - Mensaje 14: ‘Ready to measure. Waiting for commands…’
     CMM → PC Control
     Tamaño: 64 bytes
     Tipo: aperiódico (basado en eventos), no determinista
     Latencia: 100 ms
     Fiabilidad: 99.99%
   - Mensaje 15: ‘ACK’
     PC Control → CMM
     Tamaño: 64 bytes
     Tipo: aperiódico (basado en eventos), no determinista
     Latencia: 100 ms

Fiabilidad: 99.99%

2. Comunicación entre el ordenador central de control y el edge. El ordenador central de control envía mensajes de configuración, la orden de comenzar/finalizar el proceso de medición, órdenes de movimiento (a determinadas coordinadas), giros, etc.
    - Mensaje 16: 'Operation commands'
      PC Control → Edge
      Tamaño: hemos pedido información para modelarlo
      Tipo: aperiódico
      Latencia: 100 ms
      Fiabilidad: 99.99%
    - Mensaje 17: 'ACK' ¿se envían mensajes de ACK a cada comando?
      CMM → PC Control
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.99%
3. El EDGE envía las medidas realizadas que recibe del CMM al ordenador central de control.
    - Mensaje 18: 'Measurements reports for Monitoring'
      Edge → PC Control
      Tamaño: hemos pedido información para modelarlo
      Tipo: periódico, determinista
      Periodo: 250 ms
      Latencia: 100 ms
      Fiabilidad: 99.99%

9.1.5. Comunicaciones en el almacén

El AGV transporta el material desde la celda de telemetría hasta el almacén. Cuando llega a su destino, informa al PC de control. Cuando llega a su destino, el operario coge la pieza y la almacena en las estanterías correspondientes. Por tanto, en esta celda no hay comunicaciones adicionales.

9.2. Comunicaciones inalámbricas en la planta de prensado de láminas de acero para puertas de automóviles (escenario 2)

Tal y como se ha presentado en la sección 8.2, la planta de prensado de láminas de acero implementa procesos de control y procesado autónomos mediante el uso de distintos controladores locales en la planta y el sistema de monitorización central. Para poder llevar a cabo estos procesos autónomos, es necesario que los distintos controladores dispongan de información actualizada tanto del estado de los procesos, así como de la maquinaria, robots y dispositivos que participan en los distintos procesos. Para dar soporte a estos procesos autónomos, es necesario la implementación de comunicaciones fiables y robustas entre los controladores, maquinaria, robots, sensores y dispositivos que participan en los distintos procesos. Para garantizar la flexibilidad y adaptabilidad de la planta, este tipo de comunicaciones se desean implementar mediante comunicaciones inalámbricas, particularmente utilizando tecnología 5G.

A continuación, se describen las distintas comunicaciones que tendrán lugar en la planta entre distintos elementos (controlador, maquinaria, robot, sensor, etc.). La descripción de cada mensaje se realiza de igual manera que para el escenario 1.

9.2.1. Comunicaciones en el almacén de entrada

Todas las comunicaciones que tienen lugar en el almacén de entrada tienen como origen o destino el gestor local del almacén. Estas comunicaciones soportan el envío de comandos de control y operación desde el gestor local del almacén a la grúa, sensores en los estantes y AGVs, el envío de información sobre el estado de los procesos y de la propia maquinaria, robots y otros elementos al gestor local del almacén, y el envío de información de monitorización desde el gestor local del almacén al sistema de monitorización central. A continuación, se describen con más detalles cada una de estas comunicaciones.

1. Tal y como se describió en el apartado 8.2, las bahías de las estanterías disponen de sensores que envían su estado actualizado al gestor local del almacén cuando estos son ocupados o liberados. El gestor local del almacén contesta con un ACK.
    - Mensaje 19: '*TransitMaterial*'
      Sensor en estante → Gestor almacén (incluye controlador de la grúa)
      Tamaño: 64 bytes
      Tipo: aperiódico
      Latencia: 100 ms
      Fiabilidad: 99.99%
    - Mensaje 20: '*ACK*'
      Gestor almacén → Sensor en estante
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.99%
2. El gestor local del almacén envía la orden de recogida de material a la grúa. En esta orden le indica en qué estante debe recoger el material.
    - Mensaje 21: '*ServeMaterialCommand*'
      Gestor almacén → Grúa
      Tamaño: 64 bytes
      Tipo: aperiódico
      Latencia: 50 ms
      Fiabilidad: 99.99%
    - Mensaje 22: '*ACK*'
      Grúa → Gestor almacén
      Tamaño: 64 bytes
      Tipo: aperiódico
      Latencia: 50 ms
      Fiabilidad: 99.99%
3. La grúa envía su estado actualizado (su posición, si está realizando una tarea o en reposo y si transporta carga) al gestor local del almacén.
    - Mensaje 23: '*CraneStatusReport*'
      Grúa → Gestor almacén
      Tamaño: 64 bytes

Tipo: periódico
Periodo: 5 s
Latencia: 1 s
Fiabilidad: 99%

- Mensaje 24: '*ACK*'
  Gestor almacén → Grúa
  Tamaño: 64 bytes
  Tipo: periódico
  Periodo: 5 s
  Latencia: 1 s
  Fiabilidad: 99%

4. Cuando la línea de prensa se queda sin material, envía una solicitud de material al gestor local del almacén.
   - Mensaje 25: '*MaterialRequest*'
     Línea Prensa X → Gestor almacén
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
   - Mensaje 26: 'ACK'
     Gestor almacén → Línea Prensa X
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
5. El gestor local del almacén (en particular sería el controlador de los AGVs implementado dentro del gestor local del almacén) envía los comandos necesarios para que un determinado AGV (AGV Y) realice el transporte de un pallet desde la salida del almacén hasta la línea de prensa necesitada de material.
   - Mensaje 27: '*TransportCommand*'
     Gestor del almacén (controlador de AGVs) → AGV Y
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
   - Mensaje 28: 'ACK'
     AGV Y → Gestor almacén
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
6. Informe constante del estado, posición y orientación del AGV Y. El AGV Y informa de manera periódica su estado, nivel de batería, si está ocupado realizando un transporte, si lleva carga sobre él, posición y orientación al Gestor local del almacén.
   - Mensaje 29: *'AGVStatusReport'*
     AGV Y→ Gestor almacén
     Tamaño: 250 bytes

Tipo: periódico, determinista
Periodo: 250 ms
Latencia: < periodo
Fiabilidad: 99.9%

7. Transporte finalizado. Cuando el AGV llega a su destino, envía un mensaje al gestor del almacén para informarle.
    - Mensaje 30: '*TransportCompleted*'
      AGV Y → Gestor almacén
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%
    - Mensaje 31: 'ACK'
      PC Control → AGV
      Tamaño: 64 bytes
      Tipo: aperiódico (basado en eventos), no determinista
      Latencia: 100 ms
      Fiabilidad: 99.9%
8. El gestor local del almacén envía cada minuto al sistema de monitorización central en un único mensaje sobre el estado del stock en el almacén (puede ser una matriz con 0s y 1s que indican si cada estante está ocupado o libre) y estadísticas de los trabajos realizados por la grúa (número de trabajos realizados en el último periodo, tiempo en espera, tiempo realizando trabajos).
    - Mensaje 32: *'StorageStatistics'*
      Gestor almacén → Sistema de monitorización central
      Tamaño: 1 kbytes
      Tipo: periódico
      Periodo: 60 s
      Latencia: 500 ms
      Fiabilidad: 99%
    - Mensaje 33: 'ACK'
      Sistema de monitorización central → Gestor almacén
      Tamaño: 64 bytes
      Tipo: periódico
      Latencia: 500 ms
      Fiabilidad: 99.99%
9. El gestor local del almacén envía cada minuto al sistema de monitorización central en un único mensaje información sobre los trabajos de transporte realizados por los AGVs (número de transportes realizados en el último periodo por cada AGV, número de transportes totales, tiempo realizando un transporte para cada AGV, tiempo en espera para cada AGV).
    - Mensaje 34: *'TransportStatistics'*
      Gestor almacén → Sistema de monitorización central
      Tamaño: 1 kbytes
      Tipo: periódico
      Periodo: 60 s
      Latencia: 500 ms

Fiabilidad: 99%

- Mensaje 35: 'ACK'
  Sistema de monitorización central → Gestor almacén
  Tamaño: 64 bytes
  Tipo: periódico
  Latencia: 500 ms
  Fiabilidad: 99.99%

9.2.2. Comunicaciones en una línea de prensa

En este apartado se listan las comunicaciones implementadas para una de las líneas de prensado, siendo equivalentes las comunicaciones para el resto de las líneas.

1. Un sensor situado en el "*conveyor"* inicial de cada línea de prensa se emplea para notificar cuándo se puede cargar una nueva lámina de acero en la línea. Cada vez que este sensor sufra un cambio de estado, enviará la información del estado actual directamente a los controladores de los robots encargados de mover las planchas desde el pallet a la línea. Los controladores serán los encargados de decidir cuándo cargar una nueva plancha en función de la información proporcionada por el sensor del "*conveyor*".
   - Mensaje 36: *'SensorState'*
     Sensor conveyor entrada línea → Controlador Robot
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99%
   - Mensaje 37: 'ACK'
     Controlador Robot → Sensor *conveyor* entrada línea
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
2. Al alimentar la misma línea de prensa mediante 2 robots distintos, los controladores de estos han de comunicarse entre sí para evitar colisiones y funcionamientos erróneos. Por ello, de forma periódica cada robot envía al otro robot su estado actual y posición y viceversa.
   - Mensaje 38: *'RobotState'*
     Controlador brazo robot *1* → Controlador brazo robot *2*
     Controlador brazo robot *2* → Controlador brazo robot *1*
     Tamaño: 250 bytes
     Tipo: periódico
     Periodo: 250 ms
     Latencia: 100 ms
     Fiabilidad: 99%
3. Un sensor situado previo a la primera prensa en el "*Conveyor"* inicial de cada línea, notifica su estado al controlador del robot encargado de transportar la plancha de acero desde el *"conveyor"* al interior de la prensa. El sensor envía su estado actual al controlador cada vez que sufre un cambio.
   - Mensaje 39: *'SensorState'*
     Sensor previo prensa → Controlador Robot

Tamaño: 64 bytes
Tipo: aperiódico
Latencia: 100 ms
Fiabilidad: 99%

- Mensaje 40: 'ACK'
  Controlador Robot → Sensor previo prensa
  Tamaño: 64 bytes
  Tipo: aperiódico
  Latencia: 100 ms
  Fiabilidad: 99.99%

4. Cada prensa envía su estado actualizado a los controladores que tiene asociados, tanto al encargado de cargar nuevos productos, como al encargado de extraerlos. Mediante esta información los controladores son capaces de decidir cuándo ejecutar una determinada acción.
   - Mensaje 41: *'PressState'*
     Prensa → Controlador Robot Carga + Controlador Robot Descarga
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99%
   - Mensaje 42: 'ACK'
     Controlador Robot Carga + Controlador Robot Descarga→Prensa
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
5. De igual manera que el sensor previo a la primera prensa notifica al controlador asociado su estado, el sensor tras la última prensa, situado en el Conveyor de salida, envía su estado actualizado al controlador asociado. determinada acción.
   - Mensaje 43: *'SensorState'*
     Sensor tras última prensa → Controlador Robot
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99%
   - Mensaje 44: 'ACK'
     Controlador Robot Carga + Controlador Robot Descarga→Prensa
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
6. En la zona del control de calidad, un sensor situado en el centro del conveyor de salida detecta cuando hay una plancha sobre él. Este sensor notifica su estado actual al controlador de la cámara que realiza en control de calidad.
   - Mensaje 45: *'QualityNodeState'*
     Sensor ubicado en control de calidad → Controlador cámara
     Tamaño: 64 bytes

Tipo: aperiódico
Latencia: 100 ms
Fiabilidad: 99%

- Mensaje 46: 'ACK'
  Controlador cámara → Sensor ubicado en control de calidad
  Tamaño: 64 bytes
  Tipo: aperiódico
  Latencia: 100 ms
  Fiabilidad: 99.99%

7. En función de los datos recibidos del entorno el controlador de la cámara para el control de calidad, situado en un PC próximo a la línea, enviará los comandos necesarios a la cámara para iniciar una medición.
   - Mensaje 47: *'MeasuringCommands'*
     Cámara → Controlador cámara
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99%
   - Mensaje 48: 'ACK'
     Controlador cámara → *Cámara*
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
8. Una vez la cámara ha realizado la medición devuelve la imagen capturada de la plancha para que el controlador pueda analizar si los puntos de control detectados se encuentran dentro de los valores permitidos.
   - Mensaje 49: *'QualityCameraData'*
     Cámara → Controlador cámara
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99%
   - Mensaje 50: 'ACK'
     Controlador cámara → Cámara
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%
9. Una vez recibidos los datos de la cámara, el controlador enviará resultado al controlador del robot encargado de extraer la plancha en caso de que el resultado fuese fallido.
   - Mensaje 51: *'QualityResult'*
     Controlador cámara → Controlador del robot
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99%

- Mensaje 52: ‘ACK’
  Controlador del robot → Controlador cámara
  Tamaño: 64 bytes
  Tipo: aperiódico
  Latencia: 100 ms
  Fiabilidad: 99.99%

9.2.3. Comunicaciones con el sistema de monitorización central

En este apartado se recogen todos los mensajes que se envían al sistema de monitorización central para enviar información estadística del funcionamiento de los distintos procesos.

1. Cada controlador de robot envía de forma periódica sus estadísticas al sistema de monitorización central.
   - Mensaje 53: *‘RobotStatistics’*
     Controlador del robot → sistema de monitorización central
     Tamaño: 64 bytes
     Tipo: periódico
     Periodo: 60.000 ms
     Latencia: 500 ms
     Fiabilidad: 99%
   - Mensaje 54: ‘ACK’
     sistema de monitorización central → Controlador del robot
     Tamaño: 64 bytes
     Tipo: periódico
     Periodo: 60.000 ms
     Latencia: 500 ms
     Fiabilidad: 99.99%
2. Cada prensa envía de forma periódica sus estadísticas al sistema de monitorización central.
   - Mensaje 55: *‘PressStatistics’*
     Prensa # → sistema de monitorización central
     Tamaño: 64 bytes
     Tipo: periódico
     Periodo: 60.000 ms
     Latencia: 500 ms
     Fiabilidad: 99%
   - Mensaje 56: ‘ACK’
     sistema de monitorización central → Prensa #
     Tamaño: 64 bytes
     Tipo: periódico
     Periodo: 60.000 ms
     Latencia: 500 ms
     Fiabilidad: 99.99%
3. Cada controlador de cámara envía de forma periódica sus estadísticas al sistema de monitorización central.
   - Mensaje 57: *‘QualityStatistics’*
     Controlador de cámara → sistema de monitorización central
     Tamaño: 64 bytes

Tipo: periódico
Periodo: 60.000 ms
Latencia: 500 ms
Fiabilidad: 99%

- Mensaje 58: 'ACK'
  sistema de monitorización central → Controlador de cámara
  Tamaño: 64 bytes
  Tipo: periódico
  Periodo: 60.000 ms
  Latencia: 500 ms
  Fiabilidad: 99.99%

9.2.4. Comunicaciones en el almacén de salida

En este caso las comunicaciones son prácticamente iguales que las del almacén de entrada. A continuación, se indica la que difiere.

1. Un sensor previo al almacén automatizado indica al controlador de este su estado actual, informando así al controlador del almacén si tiene producto para ser almacenado o no.
   - Mensaje 59: *'SensorState'*
     Sensor previo almacén salida → Gestor local del almacén de salida
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99%
   - Mensaje 60: 'ACK'
     Gestor local del almacén de salida → Sensor previo almacén salida
     Tamaño: 64 bytes
     Tipo: aperiódico
     Latencia: 100 ms
     Fiabilidad: 99.99%

10. Modelado digital de escenarios industriales

Para desarrollar los modelos digitales que se tratan en esta memoria se ha empleado el software *Visual Components Premium 4.3* [3]. *Visual Components* es un software de simulación 3D para la fabricación, que incluye herramientas para el diseño, construcción, simulación de la producción, obtención de estadísticas, programación fuera de línea y verificación de PLC de fábricas.

Previamente a describir el desarrollo de los escenarios, se dedica el apartado 10.1 a explicar el funcionamiento del programa y las herramientas que ofrece para la implementación de modelos digitales de fábricas. En este apartado se detallarán dos de los aspectos más importantes a conocer sobre *Visual Components*: Modelado de Procesos (apartado 10.1.1) y Modelado de Componentes (apartado 10.1.2). El primero de ellos, describe la forma de distribuir productos, procesos y flujos de procesos en un diseño. Por otro lado, el modelado de componentes recoge las herramientas para crear nuevos componentes (como, por ejemplo, máquinas que no existen en el catálogo de componentes de *Visual Components*) o modificar componentes ya existentes (prensas, AGV, controladores de AGV, etc.).

10.1. *Visual Components*

Visual Components alberga, como se ha dicho anteriormente, herramientas suficientes para diseñar y representar rápidamente soluciones de producción. A continuación, se presentan algunas de las funcionalidades más importantes de *Visual Components:*

- Simulación de fábrica. La simulación de fábrica ofrece todas las herramientas esenciales para diseñar y configurar rápidamente las simulaciones de fabricación en un entorno interactivo y fácil de usar, así como exportar y compartir fácilmente los resultados. Algunas de estas herramientas son:
    - Configuración de diseño 3D. Crear diseños rápidamente arrastrando componentes desde el catálogo electrónico directamente al mundo 3D y conectando componentes compatibles a través de la función *plug-and-play* (PnP).
    - Modelado de procesos. Es una forma simple y visual de configurar la simulación de sus diseños que ofrece el software. Defina productos, procesos y flujos de producción con flujos de trabajo visuales.
    - Catálogo electrónico. Contiene más de 2500 componentes predefinidos y listos para usar, una biblioteca sólida de modelos virtuales de robots, máquinas y equipos de docenas de marcas líderes en automatización industrial.
    - Importación de archivos CAD. Importar archivos CAD directamente al mundo 3D. *Visual Components* admite tipos de archivos CAD de muchos de los principales proveedores de CAD (Autodesk, CATIA, SolidWorks, etc.).
    - Dibujos 2D y BoM. Importar dibujos 2D de nuestros diseños de fábrica actuales al mundo 3D y configurar las simulaciones. La función también admite la exportación de dibujos 2D de diseños 3D, incluida la lista de materiales.
    - Estadísticas. Visualizar estadísticas de simulación utilizando gráficos de líneas, áreas, barras o circulares. Crear, modificar y visualizar datos de simulación personalizados en el panel de estadísticas. Cuando se requiera un análisis más profundo, los datos de simulación se pueden exportar fácilmente en formatos de datos PDF o Microsoft Excel.

- Puesta en servicio virtual básica. Se pueden probar y validar la funcionalidad de los modelos digitales conectando las simulaciones con sistemas de control reales. De esta forma nos podremos asegurar de que nuestros sistemas de producción están bien integrados y funcionan de manera eficiente antes de la ejecución física (PLC *Connectivity*).
- Programación básica de robots. Permite enseñar y programar rápidamente los robots y cobots para su uso en un entorno virtual utilizando herramientas de programación de robots sin conexión, lo cual ahorra tiempo y costes de integración de robots y cobots para sus aplicaciones.
- Convertir datos CAD en modelos de simulación. Permite diseñar y simular aplicaciones de fabricación realistas con funciones de modelado fáciles de usar. Con esta herramienta es posible crear geometrías 3D desde cero, importar datos CAD existentes o modificar cualquier componente disponible de la biblioteca y convertirlo en modelos listos para la simulación.
- Modelado de componentes. Puede agregar estructuras cinemáticas y comportamientos funcionales a modelos CAD importados, modificar detalles de características de modelos existentes y crear su propia biblioteca personalizada de componentes.
- Asistentes prediseñados para un flujo de trabajo más rápido. Los complementos de funcionalidad prediseñados para el modelado de componentes agilizan el proceso de desarrollo de componentes al solicitar entradas simples y configurar automáticamente los comportamientos de los componentes.
- Geometría sólida básica. Con esta herramienta es posible crear nuevas geometrías 3D o realizar actualizaciones rápidas de los modelos 3D importados, directamente dentro del producto utilizando el kit de herramientas de modelado.
- Simplificación de geometría. Es posible mejorar el rendimiento de la simulación simplificando y eliminando detalles innecesarios de sus modelos y reduzca el tamaño de los archivos.
- Puesta en servicio virtual mejorada. El software permite conectar las simulaciones con una gama de controladores físicos y virtuales específicos de determinados proveedores para planificar, depurar y verificar sus programas con imágenes poderosas y mejorar las posibilidades de puesta en marcha virtual.
- Programación avanzada de robots. Esta herramienta permite programar rápidamente robots con cinemática detallada y visualizarlos en simulación para diversas aplicaciones prácticas como soldadura, sellado, corte, pintura, manipulación de materiales, etc.
- Declaraciones de ruta. El software permite enseñar y simular rápidamente trayectorias de posiciones con robots. Las herramientas de selección de bordes y curvas lo ayudan a seleccionar y editar las trayectorias de los robots con aplicaciones prácticas para soldadura de robots, cuidado de máquinas, pintura y operaciones de chorro de agua, sellado y corte.

En el sitio web *Visual Components* (see [4]) hay disponible multitud de tutoriales, cursos y lecciones que ofrecen una formación a varios niveles, empezando por cursos básicos hasta los más avanzados.

Un aspecto muy destacado del software es la biblioteca *Process Modeling*, diseñada para la simulación de fábrica, que define de manera muy visual los procesos para cada grupo de productos a lo largo de la línea de producción. Los diseños de simulación de fábrica y los datos estadísticos relacionados se pueden exportar posteriormente a diferentes formatos de archivo compatibles.

10.1.1. Modelado de procesos

Para facilitar el modelado de plantas industriales *Visual Components* dispone de un conjunto de bibliotecas de componentes que están disponibles en el *eCatalog*. Existen principalmente dos bibliotecas para crear una planta industrial en *Visual Components.* La primera de ellas es la biblioteca *Process Modeling* (*PM*). Y la segunda, es la biblioteca *Works*.

La diferencia entre ellas es el modo en el que está creado cada uno de sus componentes y como se trabaja con ellos. En este proyecto se ha empleado únicamente la biblioteca de Modelado de Procesos *(Process Modeling)* dada su mayor sencillez y número de componentes disponibles. Esta función agiliza el proceso de planificación y optimización del diseño con una configuración de la simulación rápida en el software *Visual Components*. Algunas definiciones de palabras clave son las siguientes:

- Producto: cualquier entidad que pasa por un proceso particular en un diseño.
- Proceso: conjunto de declaraciones que asignan cierto comportamiento a un proceso.
- Flujo: La secuencia de procesos que sigue el producto en un diseño durante la simulación (ver Figura 47).

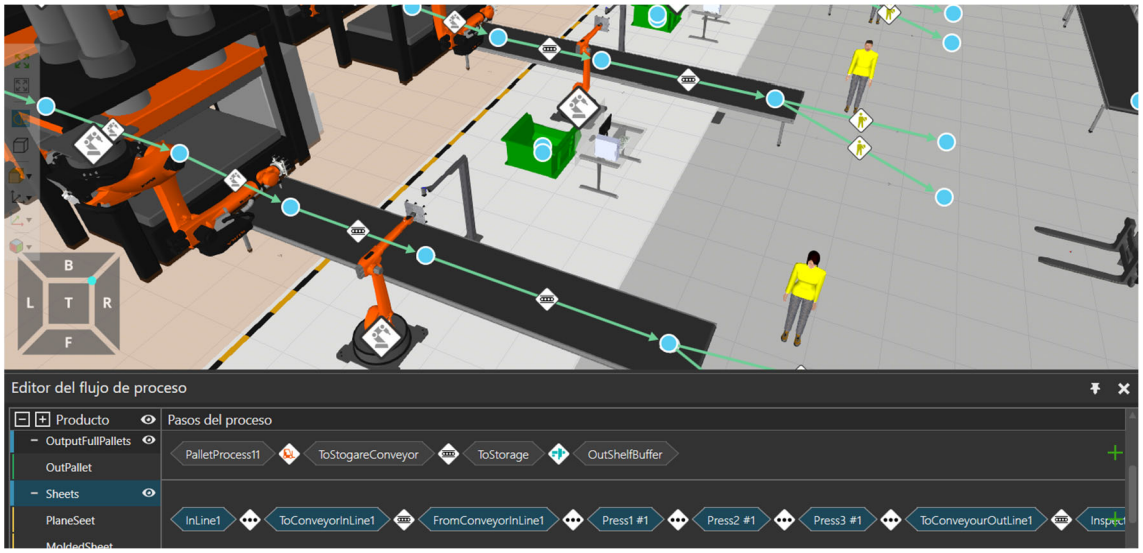


Figura 47. Flujos de trabajo correspondientes a las planchas de acero transformadas en la celda de prensado.

Mediante el Modelado de Procesos podemos encontrar formas de optimizar los procesos de fabricación y liberar recursos, tarea que no es fácil. La automatización puede ayudar en forma de robots móviles. Un robot móvil está diseñado para moverse en un entorno y realizar tareas. Por ejemplo, podemos asignar tareas de entrega a robots móviles, lo que permitiría a los trabajadores concentrarse en tareas específicas y evitar interrupciones, como dejar su estación de trabajo para llevar cosas de un lugar a otro. Es posible asignar prioridad a recursos y tareas, el transporte múltiple y la navegación a todo tipo de recursos: operarios humanos, robots móviles, brazos robot, etc.

10.1.2. Modelado de componentes

El Modelado de Componentes es la otra herramienta fundamental que se ha utilizado en este proyecto, ya que con ella se ha podido modelar nuevos componentes y crear bibliotecas propias de estos. En él se aprenden los conceptos básicos para modelar un componente y desarrollar una comprensión de los datos de los componentes, su estructura y capacidad de reutilización. Por ejemplo, se puede importar archivos CAD para máquinas, modelarlas para simular cómo funcionan y luego usarlas en diferentes diseños para visualizar, integrar y optimizar soluciones de fábrica.

El modelado de componentes es flexible. Es decir, existen muchos enfoques para modelar un componente de principio a fin. Puede simplificar las cosas utilizando un componente del catálogo electrónico (*eCatalog*) como plantilla para modelar un nuevo componente, como un robot. También podemos utilizar asistentes integrados para acelerar el proceso de modelado. Por ejemplo, hay herramientas que rápidamente preparan la producción de transportadores, posicionadores, dispositivos IO y robots.

El modelado de componentes es un apartado imprescindible si queremos desarrollar nuevos modelos digitales de plantas industriales. Esta opción del programa permite modificar un componente, crearlo desde cero, añadirle funcionalidades, etc. En este apartado tan sólo se describen unas pocas herramientas que ofrece el programa para crear nuevos componentes y definir su funcionamiento.

Antes de comenzar a explicar conceptos del modelado de componentes, se considera que es importante saber cómo funciona y cómo está desarrollado *Visual Components.* Este software aplica *OOP* (*Object Oriented Programming*) para identificar cada uno de los elementos del escenario. En este contexto, es importante realizar la definición de "componente" en *Visual Components*. Un componente es cada elemento que podemos incluir en un escenario de *Visual Components*. Estos componentes son objetos[5] de una determinada clase[6] que describe las propiedades, métodos y eventos de dicho objeto. A su vez, cada componente está compuesto por más objetos, en este caso denominados comportamientos, y de igual modo que para los objetos componente, los objetos comportamiento pertenecen a una clase.

Todas las clases posibles de objetos en *Visual Components* están documentadas en la opción "Python API" que podemos encontrar dentro de la pestaña de "AYUDA" del propio programa (ver Figura 48). Esta documentación es imprescindible como veremos más adelante para trabajar con comportamientos de tipo *Script* (programación en Python).

Las principales clases empleadas en el desarrollo de ambos escenarios se nombran a continuación, de manera que cuando se haga referencia a alguno de estos términos de aquí en adelante, el lector identifique que se está hablando de una clase o tipo de objeto:

- *vcComponent*: define el elemento primario (componente).
- *vcBehaviour*: define objetos denominados comportamientos encontrados en un componente.
- *vcScript*: define el comportamiento que permite ejecutar códigos Python.

[5] En la programación orientada a objetos, un objeto es un ente orientado a objetos (programa de computadoras) que consta de un estado y de un comportamiento, que a su vez constan respectivamente de datos almacenados y de tareas realizables durante el tiempo de ejecución.

[6] En informática, una clase es una plantilla para la creación de objetos de datos según un modelo predefinido. Cada clase es un modelo que define un conjunto de variables, métodos y eventos apropiados para operar con dichos datos.

- *vcSignal*: define el comportamiento que modela señales.
- *vcProcessExecutor*: define el comportamiento que modela una secuencia de acciones.
- *vcTransportNode*: define el comportamiento que modela una ubicación en el entorno.
- *vcTransportLink*: define el movimiento de productos entre dos *vcTransportNode.*
- *vcPythonTransportController*: define un comportamiento de tipo *vcScript.* Este comportamiento controla el recurso empleado para realizar el transporte de un producto a través de un *vcTransportLink.* Por ejemplo, AGV es un recurso, controlado por un *vcPythonTransportController,* para realizar un transporte a través de un *vcTransportLink.*

Para obtener más información sobre cada una de estas clases se debe entrar a la pestaña AYUDA/Python API como se muestra en la Figura 48.

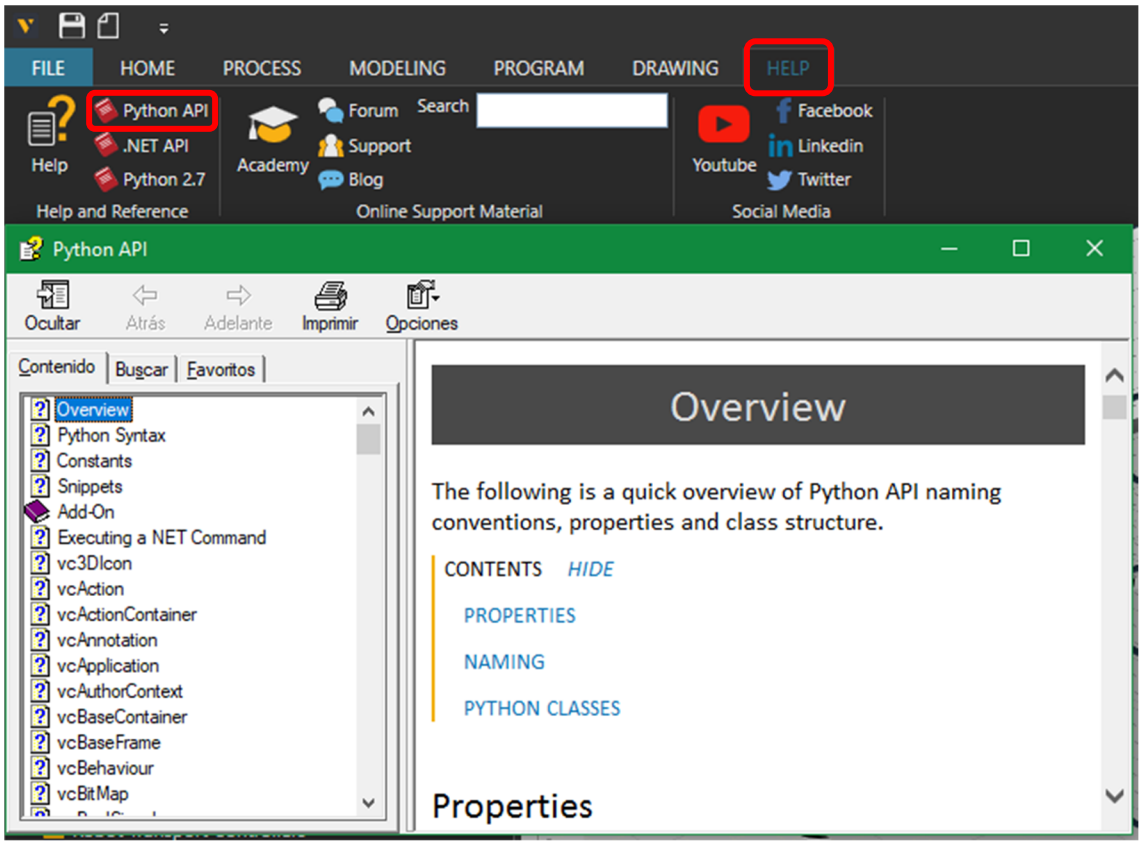


Figura 48. Acceso a la documentación de la API en Python para *Visual Components*.

El modelado de componentes se realiza desde la pestaña denominada "MODELADO" o "MODELING" en inglés. La mejor opción para identificar los objetos que forman un componente es acceder a la pestaña modelado, seleccionar el componente deseado y observar el panel situado a la izquierda.

10.1.2.1. Componentes

Los componentes, objetos de tipo *vcComponent,* son la unidad principal que se emplea para desarrollar un escenario. Cada componente queda definido por un conjunto de propiedades, comportamientos y articulaciones.

- Propiedades: objetos tipo *vcProperty* que definen las características del componente.

- Comportamientos: objetos tipo *vcBehaviour* que modelan el funcionamiento del componente.
- Articulaciones: objetos tipo *vcNode* que permiten a un componente tener elementos móviles.

De ahora en adelante cuando se hable de propiedades, comportamientos o articulaciones, se estará hablando de objetos que pertenecen a cada una de las clases correspondientes. Adicionalmente, se ha de distinguir entre propiedades de un componente (objetos de tipo *vcProperty*) y propiedades de un objeto (propiedades o atributos que tiene la clase a la que pertenece dicho objeto).

A pesar de ser igual de importantes los tres conceptos anteriores nos centraremos mayoritariamente en los comportamientos, dado que se han usado estos para implementar los módulos de comunicaciones. Toda la documentación sobre las propiedades, comportamientos, etc., se puede encontrar en el apartado '*Documents*' dentro de la opción Ayuda de la pestaña de AYUDA (ver Figura 49).

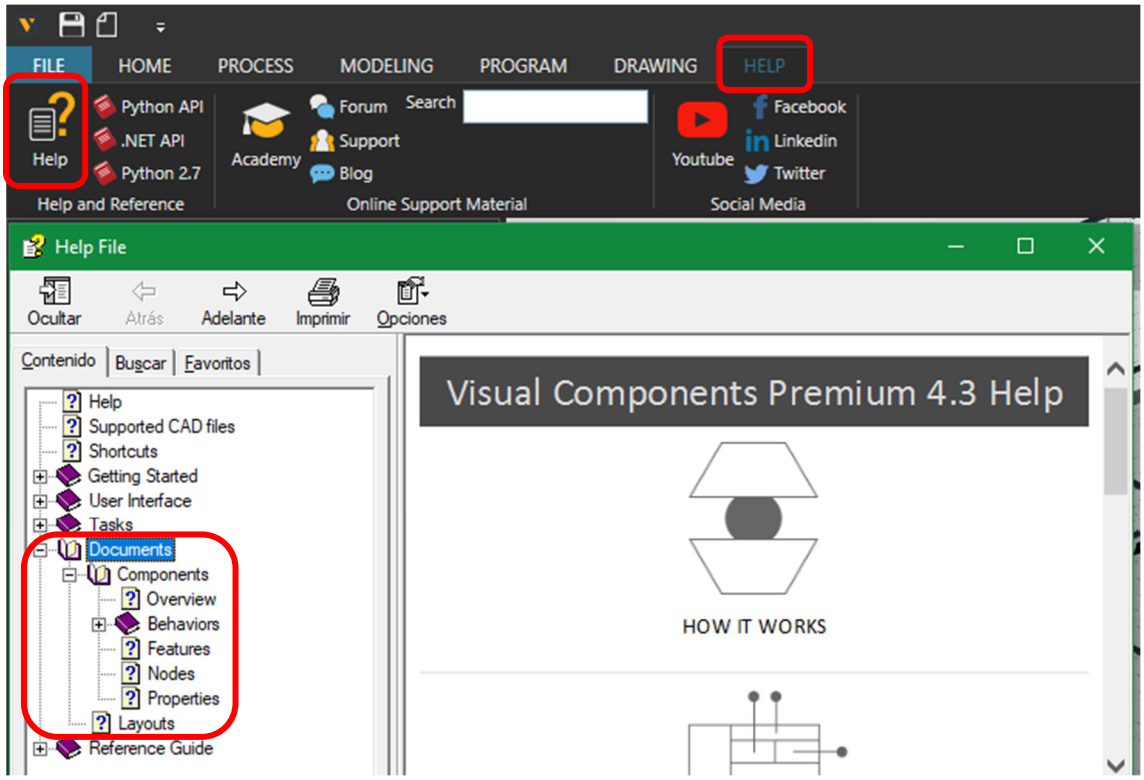


Figura 49. Acceso al apartado documentos de la pestaña de ayuda.

Para visualizar un componente y todas las propiedades, comportamientos y articulaciones de las que está compuesto, se ha de acceder a la pestaña de "MODELADO", seleccionar el componente deseado, y en el panel izquierdo denominado "Gráfico de Componentes" se muestran todos los elementos que conforman el componente (ver Figura 50).

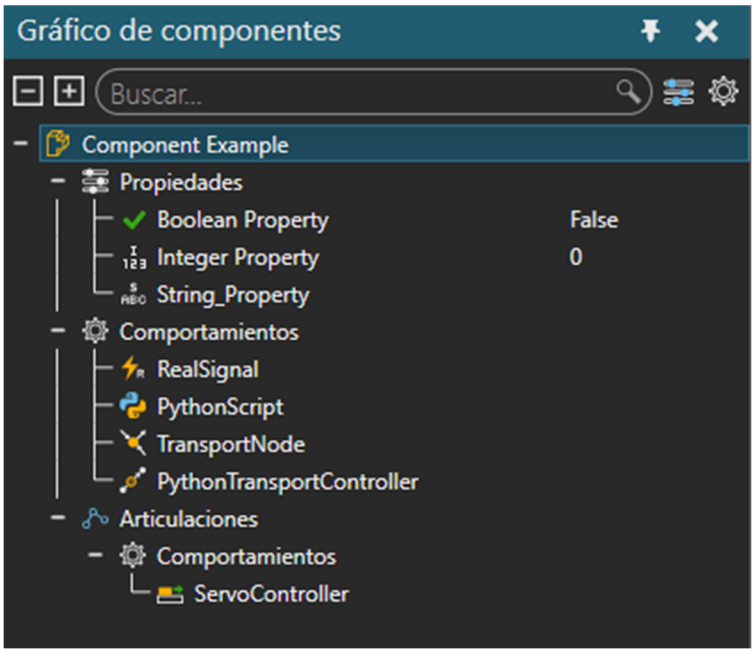


Figura 50. Ejemplo del recuadro de Gráfico de Componentes.

Tal y como se observa en la Figura 50 y se ha descrito anteriormente, el componente denominado "Component Example" está formado por una serie de objetos de tipo propiedad, comportamiento y articulación. Seleccionando cada propiedad, comportamiento o articulación desde el gráfico de componentes, se pueden acceder a algunas de las propiedades que tienen asociadas ese tipo de objetos (objeto propiedad, objeto comportamiento, objeto articulación).

La única manera de acceder a todas las propiedades, métodos o eventos que ofrece cada uno de los objetos descritos como propiedades, comportamientos o articulaciones, es a través de un objeto de tipo *vcScript*, de ahí la importancia de conocer la documentación recogida en la opción "Python API" de la pestaña de ayuda.

10.1.2.2. Propiedades

Las propiedades como ya se ha descrito, son objetos de tipo *vcProperty,* que permiten establecer valores o parámetros de un componente. La clase *vcProperty* tiene a su vez muchas clases heredadas que concretan más el tipo de propiedad que es. En este apartado no se tratan en profundidad en todos los tipos de propiedades existentes, pero sí se comentan los más importantes. Los tipos de propiedad que se consideran más importantes son: booleanas, string, enteras, reales, botón y distribución. En la Figura 51 se muestran todos los tipos de propiedades disponibles. Los principales tipos de propiedades son:

- Booleana: propiedad que puede almacenar un valor True o False.
- String: propiedad que almacena una cadena de caracteres.
- Entera (*Integer*): propiedad que almacena un valor entero.
- Real: propiedad que almacena un valor real.
- Botón: propiedad emula un botón o pulsador en la GUI. Se puede emplear para ejecutar un código cuando ese "botón" es pulsado.
- Distribución: propiedad que modifica su valor durante la simulación dentro de una distribución normal especificada.

Como norma general las propiedades de un componente se emplean para parametrizar o configurar dicho componente.

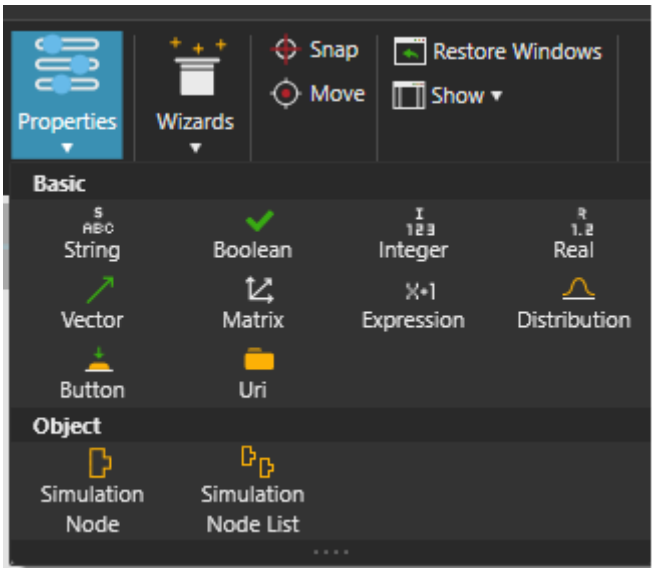


Figura 51. Tipos de propiedades disponibles.

10.1.2.3. Comportamientos

En este apartado se describen los comportamientos en mayor profundidad. Como se ha indicado anteriormente los comportamientos son objetos de tipo *vcBehaviour*. La clase *vcBehaviour* tiene un conjunto de clases heredadas que permiten concretar el tipo de comportamiento que es. Los principales comportamientos empleados son los siguientes:

- *Signal String*: comportamiento de tipo *vcStringSignal* que permite enviar señales cuyo valor contiene una cadena de caracteres.
- *Script* en Python: comportamiento de tipo *vcScript* con el cuál podemos obtener acceso a todos los elementos del escenario por completo, ya sean componentes, comportamientos, *vcTransportLinks,* etc. Cualquier objeto de *Visual Components* se puede gestionar desde un *vcScript*, obtener sus propiedades, usar sus métodos y eventos.
- Nodo de Transporte: objeto de tipo *vcTransportNode* que permite definir la posición de puntos por los que ha de pasar un producto.
- Ejecutor de Proceso: objeto de tipo *vcProcessExecutor* que asociado a un *vcTransportNode* define una serie de instrucciones a realizar cuando un producto llega a dicho *vcTransportNode.*

Cada uno de estos comportamientos tiene una serie de propiedades, métodos y eventos asociados al tipo de objeto al que pertenecen. Toda esta información se encuentra detallada tal y como se ha indicado anteriormente en la API de Python.

A continuación, se presentan las principales propiedades, métodos y eventos de los comportamientos *Signal String* y *Script* en Python, puesto que mediante estos dos comportamientos se han implementado los módulos de comunicaciones.

La señal de tipo *string* es un comportamiento de tipo *vcStringSignal.* A través de dicho comportamiento se pueden enviar mensajes en formato de cadena de caracteres. De la clase *vcStringSignal* destaca la importancia de las propiedades, métodos y eventos siguientes:

- Propiedad *Connections:* atributo que contiene una lista de los comportamientos (principalmente *vcScript*) a los que notificar cuando la señal cambia su valor.

- Método *signal("value")*: método que permite enviar un valor a través de dicho comportamiento notificando de este cambio a aquellos comportamientos contenidos en la propiedad anterior (*Connections).*
- Evento *OnValueChange*: evento que devuelve la señal cada vez que dicha señal cambia su valor. Permite ejecutar funciones definidas por el usuario cuando ocurre este evento.

El comportamiento de *Script* en Python pertenece a la clase *vcScript.* Gracias a este comportamiento podemos ejecutar códigos Python y programar funcionamientos y cualquier cosa que se nos ocurra. Al igual que para el comportamiento anterior, se destaca la importancia de las propiedades, métodos y eventos siguientes:

- Propiedad *script*: permite leer o escribir el código Python.
- Método *delay*(): retrasa la ejecución del script un determinado tiempo.
- Método *triggerCondition:* mantiene la ejecución del script en dicha línea hasta que una determinada función devuelva un valor verdadero.
- Evento *OnStart*: primer tramo de código a ejecutar cuando se inicia una simulación. Se ejecuta previo al OnRun. Permite configurar variables antes de que comience la simulación.
- Evento *OnRun*: cuerpo principal del script, es el único tramo de código donde se puede emplear método *delay*().
- Evento *OnSignal*: tramo de código que se ejecuta cuando algún comportamiento de tipo señal, en cuya propiedad *Connections* se encuentre el *vcScript* actual, sufre algún cambio.

La Figura 52 muestra los distintos comportamientos disponibles en *Visual Components*.

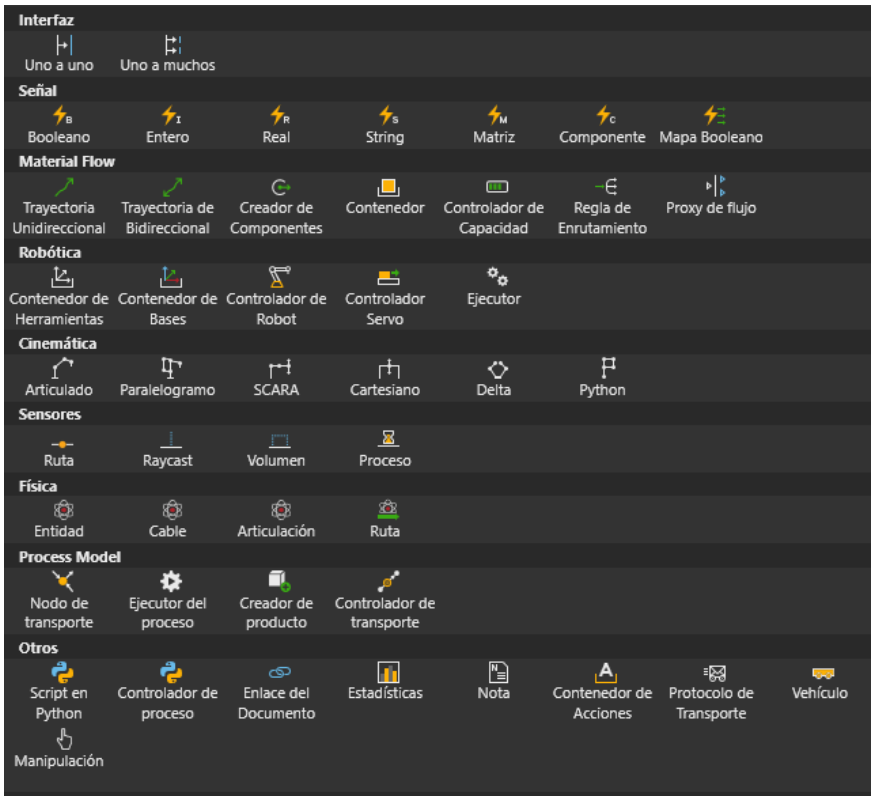


Figura 52. Tipos de comportamientos disponibles.

10.1.2.4. Articulaciones

Las articulaciones son objetos de tipo *vcNode* que se emplean para definir elementos móviles dentro de un componente. Dada su baja importancia para el presente proyecto no se redactan en mayor profundidad, pero se considera necesario saber al menos de su existencia.

10.2. Implementación de los escenarios

En este apartado se documenta en profundidad la implementación realizada para cada uno de los escenarios implementados en el presente proyecto. En concreto, se identifican los componentes de mayor relevancia empleados para cada uno de los escenarios, su funcionamiento en caso de haber sido modificado, y el flujo de los productos correspondiente.

10.2.1. Implementación de escenario 1: Línea de prensado de láminas de acero con control de calidad basado en telemetría y telecontrol

Este escenario se ha implementado usando la biblioteca de componentes "*Process Modeling*". Tal y como se ha descrito en el apartado 8.1, consiste en una única línea de producción cuya finalidad es moldear láminas de acero para ser utilizadas con distintos fines, y de las cuales se realiza una medición por coordenadas individualmente, con el fin de realizar un control de calidad.

A continuación, se detallan tanto los componentes empleados, como los flujos que describen el movimiento de productos a través de este escenario.

10.2.1.1. Componentes del escenario 1

Para la descripción de los componentes más relevantes del escenario 1, se ha creado la Tabla 18. Esta tabla está dispuesta de manera que en la primera columna contiene los nombres que tiene cada uno de los componentes en *Visual Components.* En la segunda columna se hace una breve descripción de la función de dicho componente y por último se muestra una pequeña imagen de dicho componente.

A todos los componentes de la Tabla 18 se les ha añadido los comportamientos desarrollados en el apartado 10.3 para simular el módulo de comunicaciones. Adicionalmente, algunos de los siguientes componentes han tenido que ser modificados o creados específicamente para este escenario. Aquellos componentes con modificaciones adicionales al módulo de comunicaciones o creados específicamente para este escenario se describen tras la Tabla 18.

Tabla 18. Lista de componentes en el Escenario 1.

Identificador	Descripción	Imagen
Press #1	Prensa encargada de moldear las láminas de acero. Se ha partido de un componente del *eCatalog* perteneciente a la biblioteca "*Works"* y se le han añadido los comportamientos necesarios para trabajar con ella en *ProcessModeling.* Este componente ha sido modificado.	

AGV #1	AGV o robot móvil encargado de transportar las láminas de acero de una celda a otra.	
AGV Controller #1	Controlador del robot móvil o AGV. Este componente ha sido modificado.	
ControlPC	Situado en las celdas de soldadura y medición. Su función es la de permitir al operario indicar el inicio de uno de estos procesos. Cada celda tiene su propio ControlPC. Este componente ha sido modificado.	
WeldingRobot #1	Robot articulado encargado de realizar las soldaduras en las láminas de acero.	
RobotController #1	Controlador de robot.	
CMM	CMM mediante el cual se realizan las medidas de las láminas de acero. Este componente ha sido modificado.	
EDGE	Componente intermediario entre las comunicaciones emitidas por el ControlPC de la celda de medición y el CMM. Este componente ha sido modificado.	
Central ControlPC	Ordenador situado en la celda de telemetría y telecontrol encargado de realizar las tareas que comprende dicha celda.	

Los componentes que han sido modificados (más allá de la adición del módulo de comunicaciones) o creados desde cero son: prensa, controlador de AGVs, ordenador de control local, *CMM* y *EDGE*.

- Prensa. Para el componente que simula la prensa del escenario, se ha partido del componente que proporciona *Visual Components* en su *eCatalog* denominado "*Works Press*". Este componente está diseñado por defecto para trabajar con componentes de la biblioteca *Works,* pero dado que para este escenario se emplea la biblioteca PM*,* se ha modificado para trabajar mediante PM.

  Para poder emplear la prensa en *PM* se le han añadido dos comportamientos, uno de tipo *vcTransportNode* y otro de tipo *vcProcessExecutor* (ver Figura 53). Mediante la adición de estos dos comportamientos ya se puede trabajar con la prensa en PM. Las instrucciones que contiene el *ProcessExecutor* se describen en el apartado 10.2.1.3 donde se tratan los flujos del escenario.

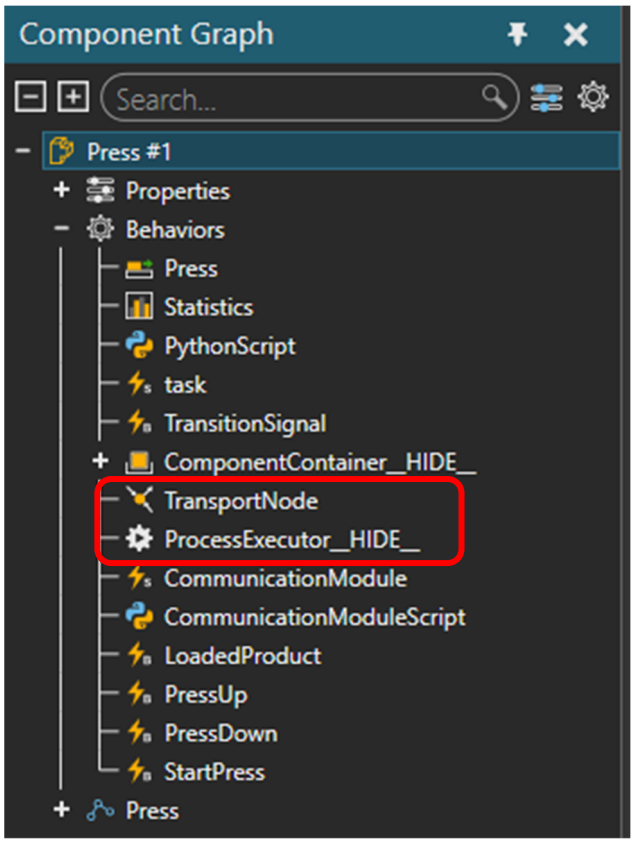


Figura 53. Gráfico de componentes de la prensa en el escenario 1.

- Controlador AGVs. El controlador de AGVs tan sólo ha recibido un par de cambios en uno de sus scripts, "*TaskLogic*", este código gestiona la asignación de AGVs para realizar un determinado transporte entre otras muchas más cosas.

  La modificación surge por la necesidad de identificar el instante en el que el controlador envía las instrucciones de transporte e identificar que recurso, o AGV va a realizar dicho transporte. Para ello, en este caso se ha añadido un comportamiento denominado "*ResourceAssignedByController*" de tipo *vcComponentSignal.* Este comportamiento contiene un objeto de tipo *vcComponent*, el cual se corresponde con el recurso o AGV

asignado para realizar un transporte. Cuando *ResourceAssignedByController* sufre un cambio, se entiende que el controlador ha asignado un transporte y que el recurso designado para realizar dicho transporte está contenido en el valor de *ResourceAssignedByController.*

En la Figura 54 se remarcan los comportamientos involucrados en esta modificación.

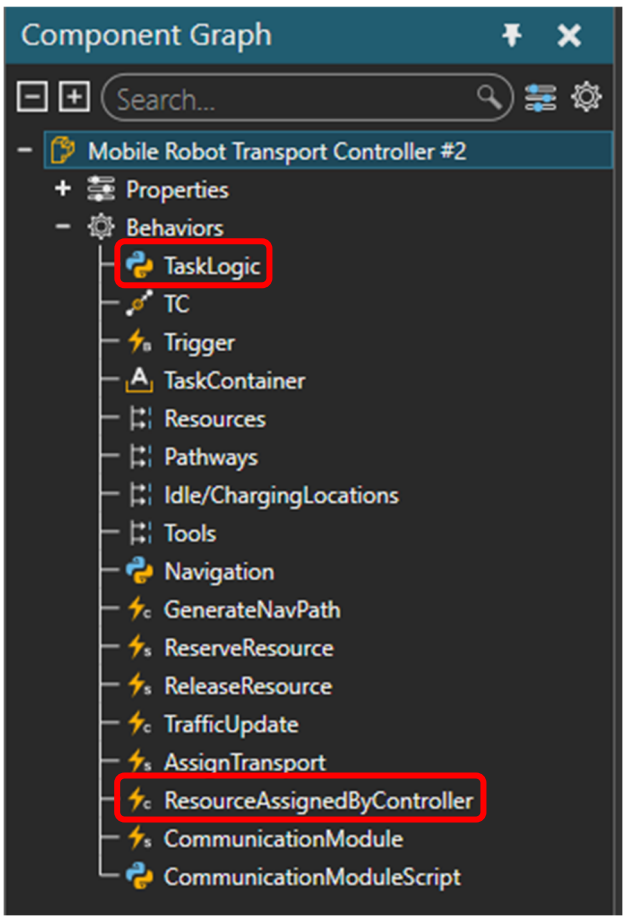


Figura 54. Gráfico de componentes del controlador de AGVs en el escenario 1.

La asignación del valor a *ResourceAssignedByController* se hace desde el script *TaskLogic*. Por lo tanto, se ha modificado el comportamiento por defecto del controlador de AGVs y se le han añadido las siguientes líneas de código a *TaskLogic*.

Las líneas que se han añadido tienen al final el comentario "*#Added"* de manera que usando la herramienta de búsqueda se pueden localizar fácilmente. Casi al final de este script, en la zona donde se obtienen los comportamientos necesarios para el funcionamiento del script se ha añadido la línea 1435, donde se obtiene el objeto que representa el comportamiento *ResourceAssignedByController* para poder asignarle valores (ver Figura 55).

```
1432    ## BEHAVIOR HANDLES
1433    reserve_signal = comp.findBehaviour('ReserveResource')
1434    release_signal = comp.findBehaviour('ReleaseResource')
1435
1436    resourceAssigned = comp.findBehaviour('ResourceAssignedByController') #Added
1437
1438    trigger_signal = comp.findBehaviour('Trigger')
1439    resource_interface = comp.findBehaviour('Resources')
1440    tool_interface = comp.findBehaviour('Tools')
1441    idle_interface = comp.findBehaviour('Idle/ChargingLocations')
1442    task_container = comp.findBehaviour('TaskContainer')  # Type: vcActionContainer
```

Figura 55. Obtención del comportamiento *ResourceAssignedByController*.

Una vez ya se tiene el objeto que permite dar valores a la señal *ResourceAssignedByController,* se puede pasar a asignarle valores. La asignación de valor se realiza dentro de la función "*allocate_task"* en la línea 826 (ver Figura 56). Esta función se ejecuta cuando el controlador asigna un transporte de producto a algún AGV disponible.

```
804  def allocate_task(self, task, resources):
805      # type: (TaskManager.Task, ResourceManager.Resource) -> None
806
807      # used only for reserving a resource to the nodes (according to multitransport strategy)
808      source_node = task.get_source_node()
809      destination_node = task.get_destination_node()
810
811      check_tool = task.tool_name != '' and task.tool_name != '-' and type(task) != TaskManager.Assist
812      for resource in resources:
813          tool_comp = None
814          if check_tool:
821
822          # create vcAction
823          self._generate_action(task, tool_comp)
824          resource.allocate_task(task, source_node, destination_node, task.tool_name, self.reserve_subscribers)
825
826          resourceAssigned.signal(resource.component) #Added
827
828          self.tasks.remove(task)
829          return
```

Figura 56. Asignación de valores al comportamiento *ResourceAssignedByController*.

- Ordenadores para control local. Estos ordenadores controlan el inicio del proceso que tiene asociado cada célula, por ejemplo, el comienzo del proceso de soldadura. Inicialmente estos componentes no realizaban ninguna función, pero se les han añadido dos comportamientos, con el fin de simular el trabajo del operario al acercarse al ordenador y lanzar los comandos para iniciar el proceso deseado. Estos componentes son los mismos que para la prensa anteriormente indicada, un componente de tipo *vcTransportNode* y un *vcProcessExecutor.* El *vcTransportNode* permite indicar la ubicación donde se realizará el trabajo de lanzar los comandos, y permite también adjuntarle el *vcProcessExecutor* para definir cuando ha de lanzar el operario los comandos. En la Figura 57 se muestra su gráfico de componentes.

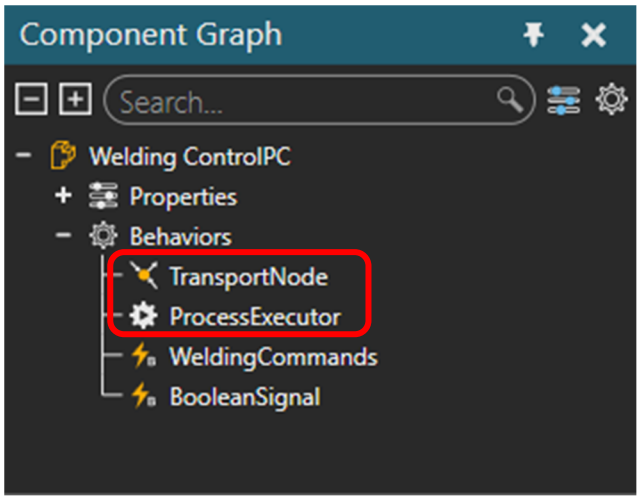


Figura 57. Gráfico de componentes del ordenador de control local de cada celda.

De igual modo que para la prensa, las instrucciones contenidas en el *ProcessExecutor* se describen en el apartado 10.2.1.3 de este mismo capítulo.

- CMM (*Coordinated Measuring Machine*). Para desarrollar este componente se ha partido de la plantilla de componente proporcionada por *Visual Components* en su *eCatalog* denominada "*Template 3-Axis Cartesian*". Esta plantilla parte de un componente que contiene las articulaciones y controladores de estas para poder realizar movimientos tridimensionales. Redimensionando este componente y modificando su aspecto se ha llegado al mostrado en la imagen correspondiente de la Tabla 18. Lista de componentes en el Escenario 1.

  Para poder trabajar además con este componente junto con la biblioteca PM, se han añadido los mismos comportamientos que para la prensa, *vcTransportNode* y *vcProcessExecutor*. De forma adicional, se ha añadido un comportamiento de tipo *vcComponentContainer,* este último almacena de forma temporal los productos que llegan al vcTransportNode del CMM (ver Figura 58).

  Como en los componentes anteriores su funcionamiento queda descrito en las instrucciones del *vcProcessExecutor* y se define en el apartado 10.2.1.3.

  Por último, en el CMM se ha creado una propiedad para indicar cuando se ha terminado de realizar una medida. Esta propiedad denominada "*MeasureDone*" valdrá True cuando el CMM termina de medir una pieza y False el resto del tiempo.

  El funcionamiento del CMM queda descrito por las instrucciones de su *vcProcessExecutor* documentadas en el apartado 10.2.1.3. El funcionamiento de las comunicaciones producidas en el CMM, por el contrario, se realizan desde su *CommunicationModuleScript*. En este último comportamiento encontramos las siguientes zonas diferenciadas.

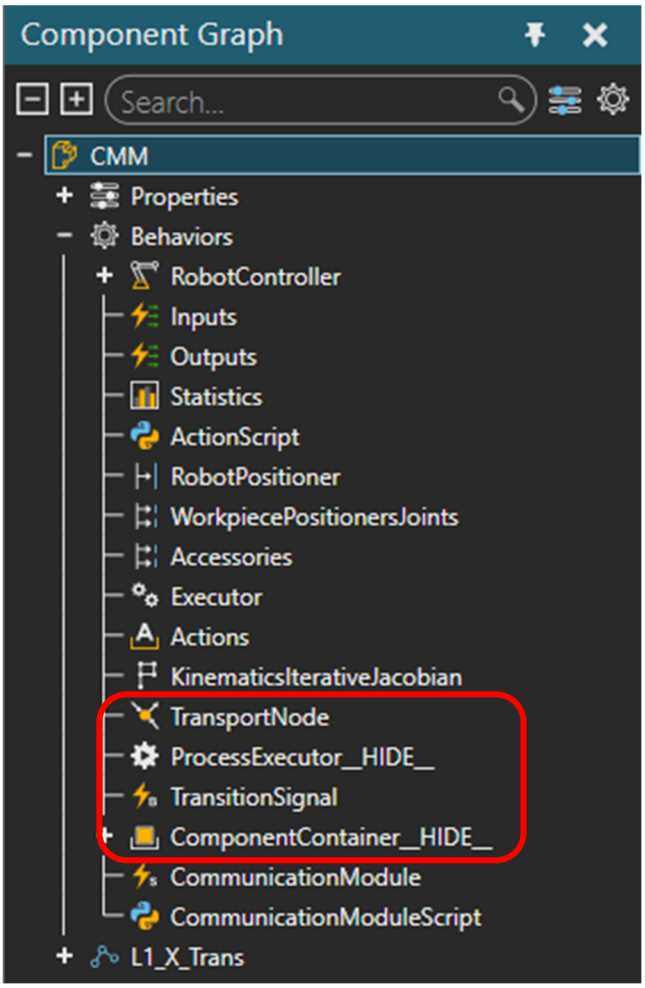


Figura 58. Gráfico de componentes del CMM.

- Función *OnStart*(): obtiene una lista de los *vcTransportLinks* que salen del *CMM* y crea una lista con los controladores encargados de realizar el transporte a través de dichos *vcTransportLinks* (ver Figura 59).

```
14  def OnStart():
15      global tOutLinks, tControllers
16      tOutLinks = [n for n in tNode.TransportLinks if n.Destination != tNode]
17
18      for z in tOutLinks:
19          tControllers.append(z.Implementer)
```

Figura 59. Función *OnStart*() del CommunicationModuleScript del CMM.

- Función *OnRun()*: espera a recibir la señal con los comandos procesados provenientes del *EDGE* y mientras que no se termine la medición de una pieza envía dos mensajes periódicamente, uno emula el envío de la posición del brazo del *CMM* y otro emula los datos de la medición (ver Figura 60).

```
## COMUNICACIONES ENTRE COMPONENTES ##
def OnRun():
    global cmmComModule, edgeComModule, measureDone, cycleSize
    while True:
        triggerCondition(lambda: getTrigger() == edgeComModule and edgeComModule.Value.split(';')[3] == 'ProcessedCommandSignal')

        while measureDone.Value == False:
            delay(2)
            cmmComModule.signal(comp.Name + ';' + edge.Name + ';' + str(74) + ';' + 'PosMonitorSignal')
            delay(0.1)
            cmmComModule.signal(comp.Name + ';' + edge.Name + ';' + str(74) + ';' + 'ScanDataSignal')
```

Figura 60. Función *OnRun*() del *CommunicationModuleScript* del CMM.

- Función *sendRequestTransport()*: función que se ejecuta cuando se ha de realizar un transporte de producto y envía una señal para emular la solicitud de transporte de dicho producto (ver Figura 61).

```
## COMUNICACIONES DE TRANSPORTE ##
def sendRequestTransport(tuple):
    global cmmComModule, tOutLinks
    link = tuple[0][0]
    if link in tOutLinks:
        cmmComModule.signal(link.Source.Component.Name + ';' + link.Implementer.Component.Name + ';' + str(74) + ';' + 'RequestTransport')
```

Figura 61. Función *sendRequestTransport*() del *CommunicationModuleScript* del CMM.

- Zona de declaración de variables: se obtienen los objetos de componentes, propiedades y comportamientos necesarios para las funciones anteriores (ver Figura 62).
- Zona de conexión de señales y eventos: en esta zona se conectan las señales del *EDGE* a este script para que detecte cambios en dichas señales. En segunda instancia se detectan los eventos que suceden en los controladores de transporte (detecta cuando se asigna un transporte) para llamar a la función *sendRequestTransport* (ver Figura 63).

```
## HANDLERS PARA LOS COMPONENTES - PROPIEDADES - COMPORTAMIENTOS NECESARIOS ##
# COMPONENTES #
edge = app.findComponent('EDGE')

# PROPIEDADES #
  #CMM
measureDone = comp.getProperty("Communication:: MeasureDone")

# COMPORTAMIENTOS #
  #CMM
thisScript = comp.findBehaviour("CommunicationModuleScript")
cmmComModule = comp.findBehaviour("CommunicationModule")
tNode = comp.findBehavioursByType(VC_TRANSPORTNODE)[0]
tOutLinks = []
tControllers = []
  #EDGE
edgeComModule = edge.findBehaviour("CommunicationModule")
```

Figura 62. Declaración de variables en el *CommunicationModuleScript* del CMM.

```
60    ## SIGNAL CONNECTIONS ##
61    if thisScript not in edgeComModule.Connections:
62      edgeComModule.Connections = edgeComModule.Connections + [thisScript]
63
64    ## EVENT TRIGGERS ##
65    for x in tControllers:
66      x.OnBeginTransport = sendRequestTransport
```

Figura 63. Declaración conexiones y eventos en el *CommunicationModuleScript* del CMM.

- EDGE. El componente que emula el *EDGE* se ha creado como un cubo, y cuyo funcionamiento se describe en el comportamiento de tipo *vcScript "CommunicactionModuleScript"* dado que sólo emula la emisión y recepción de mensajes no requiere de *vcTranportNode* ni *vcProcessExecutors.*

  El fragmento de código que describe el funcionamiento de este componente se muestra en la Figura 64. De forma descriptiva, el código de su *CommunicactionModuleScript* funciona de la siguiente manera:

  1. Constantemente espera que se reciba un mensaje desde el ordenador de control central. Este mensaje representa los comandos para la medición del *CMM.*
  2. Tras un determinado tiempo el *EDGE* devuelve al *CMM* a través de su *CommunicationModule* un mensaje que emula los comandos procesados y contiene los parámetros que modelan dicha comunicación. Esta comunicación es alámbrica puesto que requiere muy baja latencia.
  3. Mientras que no se haya completado la medición de la pieza, cada vez que *EDGE* reciba un mensaje proveniente del *CMM* con los datos respectivos e la medición, el *EDGE* envía a través de su *CommunicationModule* (ComModule) de forma periódica los datos procesados al ordenador del control central.
  4. Una vez terminada la medición se vuelve al punto 1, en el que se espera la recepción de comandos para realizar una nueva medición.

```
6   def OnRun():
7     global edgeComModule, boolSignal, cmmComModule, pcNodeComeModule
8     while True:
9       triggerCondition(lambda: getTrigger() == pcNodeComModule)
10      delay(10)
11      edgeComModule.signal(comp.Name + ';' + cmm.Name + ';' + str(74) + ';' + 'ProcessedCommandSignal')
12      boolSignal.signal(True)
13
14      while measureDone.Value == False:
15        triggerCondition(lambda: getTrigger() == cmmComModule and cmmComModule.Value.split(';')[3] == 'PosMonitorSignal')
16        delay(1)
17        edgeComModule.signal(comp.Name + ';' + pcNode.Name + ';' + str(74) + ';' + 'ProcessedPosMonitorSignal')
18        delay(0.1)
19        edgeComModule.signal(comp.Name + ';' + pcNode.Name + ';' + str(74) + ';' + 'ProcessedScanDataSignal')
```

Figura 64. Función *OnRun*() del *CommunicationModuleScript* del EDGE.

Por otro lado, en la Figura 65 de este mismo script, se muestra una zona donde se obtienen las propiedades y comportamientos necesarios para describir su funcionamiento.

```
## HANDLERS PARA LOS COMPONENTES - PROPIEDADES - COMPORTAMIENTOS NECESARIOS ##
# COMPONENTES #
pcNode = app.findComponent("PC Node")
cmm = app.findComponent("M3 Gage CMM")

# PROPIEDADES #
  #CMM
measureDone = cmm.getProperty("Communication:: MeasureDone")

# COMPORTAMIENTOS #
  #EDGE
thisScript = comp.findBehaviour("CommunicationModuleScript")
edgeComModule = comp.findBehaviour("CommunicationModule")
boolSignal = comp.findBehaviour("BooleanSignal")
  #PC Node
pcNodeComModule = pcNode.findBehaviour("CommunicationModule")
  #CMM
cmmComModule = cmm.findBehaviour("CommunicationModule")
```

Figura 65. Obtención de propiedades y comportamientos del *CommunicationModuleScript* del EDGE.

Por último, en la Figura 66 se muestran las líneas de código que se emplean para conectar las señales del ordenador de control y del *CMM* al script del *EDGE*.

```
## CONEXIONES DE SEÑALES ##
  #PC Node
if thisScript not in pcNodeComModule.Connections:
  pcNodeComModule.Connections = pcNodeComModule.Connections + [thisScript]
  #CMM
if thisScript not in cmmComModule.Connections:
  cmmComModule.Connections = cmmComModule.Connections + [thisScript]
```

Figura 66. Conexión de señale al *CommunicationModuleScript* del EDGE.

10.2.1.2. Productos del escenario 1

Para este escenario se han definido una serie de productos. Es necesario definir dichos productos para indicar el flujo de estos a través de los componentes del escenario. Para acceder a la lista de productos de un escenario hay que acceder a la pestaña “PROCESO” y una vez ahí seleccionar del panel superior la opción “Productos”. Esta opción muestra una ventana a la izquierda de la pantalla denominada Editor de Producto (ver Figura 67).

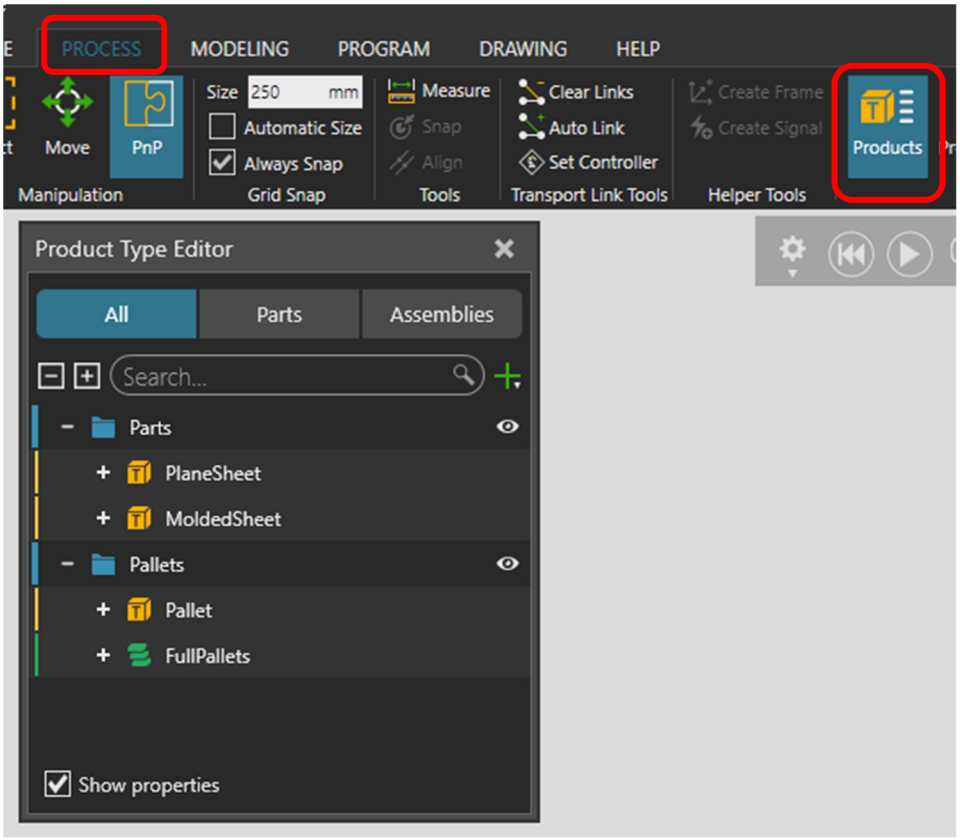
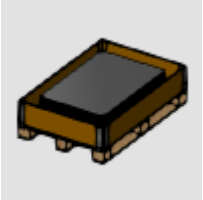


Figura 67. Ventana de editor de producto en el Escenario 1.

En la Tabla 19 se muestran de forma ordenada el nombre indicado para cada producto, una breve descripción de estos y una pequeña imagen.

Tabla 19. Lista de productos definidos en el Escenario 1.

Producto	Descripción	Imagen de producto
PlaneSheet	Láminas de acerdo planas. Materia prima inicial.	
MoldedSheet	Láminas de acero moldeadas.	

Pallet	Pallet vacío.	
FullPallets	Montaje de un pallet con 10 láminas de acero planas.	

10.2.1.3. Flujos en el escenario 1

A continuación, se presentan los flujos programados en este escenario. Un flujo establece cómo se desplazan los productos de un componente a otro y quién o qué recurso (operario, AGV, brazo robótico, etc.) realiza el transporte. En el presente escenario existen dos flujos denominados:

- "*Pallets*": Este flujo define el desplazamiento que sufren los productos "*FullPallets*" (pallet con láminas de acero) y el producto "*Pallet*" (pallet sin láminas de acero).
- "*Parts*": Este otro flujo, define el movimiento que sufren de las láminas de acero (planas o moldeadas) a través de todo el escenario.

La Figura 68 muestra la ventana de Editor de Flujo, desde la que se pueden ver todos los flujos del escenario y se accede desde la pestaña "PROCESO" con la opción "Flujo".

Figura 68. Ventana de editor de flujo en el Escenario 1.

A continuación, se describe en profundidad e individualmente cada uno de los flujos.

- Flujo Pallets. El grupo de flujo denominado *"Pallets"* representa el movimiento que sufren los pallets cuando estos últimos se quedan sin láminas. Dada esta situación, el operario coge el pallet y lo mueve al nodo 2 donde el pallet es eliminado y automáticamente se genera un nuevo pallet con láminas en el nodo 1. El tipo de controlador que se emplea para realizar el transporte se representa mediante el icono que se ve en el centro del *TransportLink.* Como se observa en la Figura 69 el controlador

de transporte es de tipo humano, por lo que se emplean humanos para realizar dicho transporte.

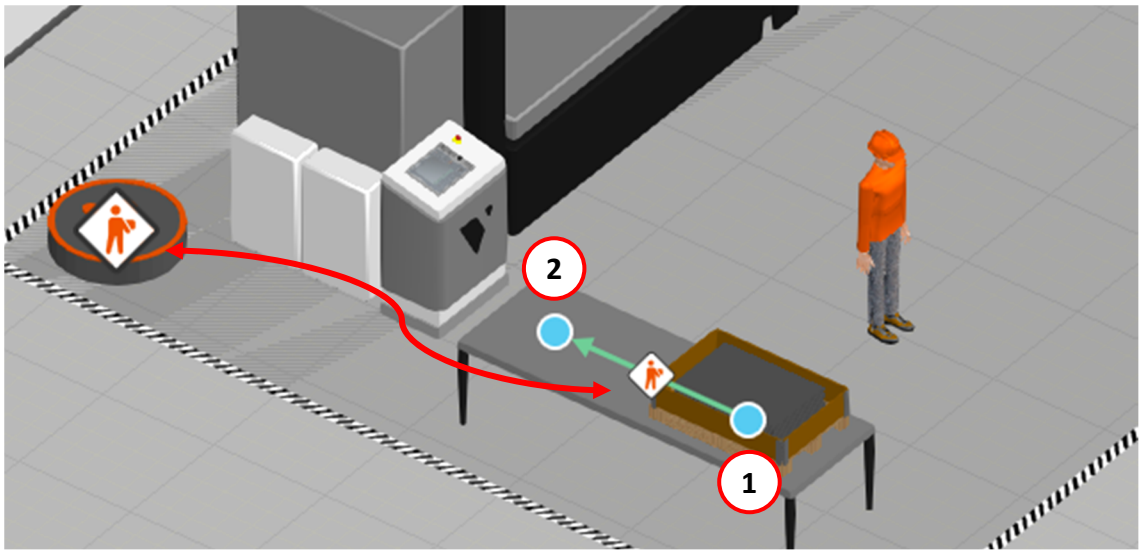


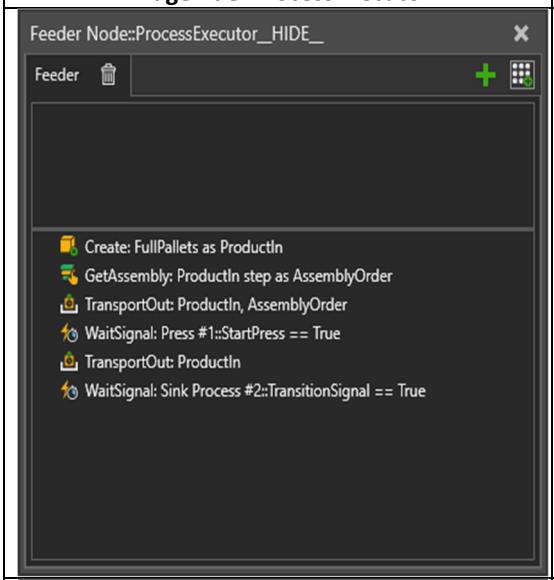
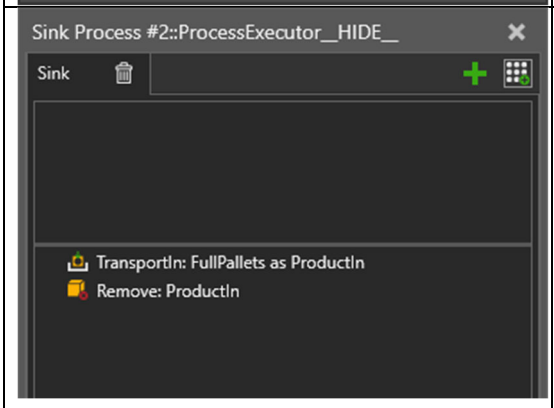
Figura 69. Flujo pallets del Escenario 1.

Para describir el funcionamiento de este flujo se emplean los *ProcessExecutor* "*Feeder*" y "*Sink*" adjuntos a los nodos 1 y 2 respectivamente. A continuación, se comenta detalladamente el funcionamiento de dichos *ProcessExecutors* (PE).

- *Feeder:* este PE tiene la función de crear un pallet lleno de láminas de acero, solicitar el transporte de estas a la prensa, y cuando se vacíe el pallet solicitar el transporte de este para su eliminación. A este flujo sólo le afectan las instrucciones de transporte cuando el pallet está vacío, las instrucciones de transporte de planchas afectan al flujo "*Parts*" que veremos más adelante.
- *Sink:* en este PE se está esperando constantemente la recepción de un pallet vacío, y cuando ocurre esto simplemente el pallet es eliminado.

En la Tabla 20 se muestra a la izquierda la imagen del PE y a la derecha una descripción que numera cada instrucción en orden de ejecución, siendo la primera instrucción en ejecutarse la situada más arriba. Las instrucciones de un PE se ejecutan de forma secuencial, hasta que no termina la instrucción actual no pasa a la siguiente.

Tabla 20. *ProcessExecutors* que intervienen en el flujo *Pallets*.

Imagen del *ProcessExecutor*	Descripción
	*Feeder* 1. Create: crea un Pallet lleno de láminas 2. GetAssembly: obtiene variable necesaria para poder transportar las láminas de una en una. 3. TransportOut: se transportan de una en una todas las láminas de acero del pallet al siguiente proceso que acepte láminas de acero. (Una vez se transportan todas las láminas pasa a la siguiente instrucción.) 4. WaitSignal: espera a que se haya iniciado el ciclo de prensado. 5. TransportOut: El pallet vacío se lleva transporta al siguiente proceso que acepte pallets. 6. WaitSignal: Espera a que se elimine el pallet para reiniciar este proceso y crear un nuevo pallet lleno.
	*Sink* 1. TransportIn: permite la entrada de productos del tipo *Pallet*. 2. Remove: Elimina el producto recibido.

- Flujo Parts. Este flujo establece el desplazamiento de las láminas de acero a través del escenario. Como se presentó en el apartado 8.1, el desplazamiento de las láminas se realiza por los operarios como por el AGV. Dentro de cada celda, el operario sitúa cada lámina en el emplazamiento y posición correspondiente para ser procesada. Por ejemplo, dentro de la celda de prensado, el operario coge cada lámina una a una desde el lugar en el que están almacenadas inicialmente y las introduce en la prensa para comenzar el proceso de prensado. Cuando finaliza este proceso, el operario sitúa la lámina sobre el AGV, el cuál trasporta la lámina prensada hasta la celda de soldadura para continuar con el proceso de fabricación. Procesos similares tienen lugar en cada celda, los cuales se describen a continuación de forma más detallada. Cada uno de estos desplazamientos realizados por los operarios o por el AGV se define mediante un *TransportLink*. En la Figura 70 se muestran numerados los *TransportLinks* del flujo *Parts.* Estos son:

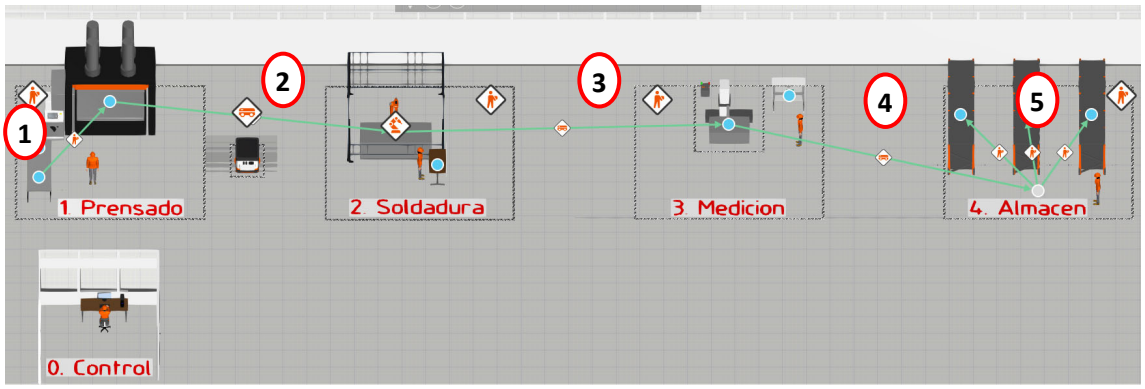


Figura 70. Flujo Parts del Escenario 1.

1. *TransportLink* 1: Desplazamiento entre el punto de almacenamiento inicial y la prensa.

   Este *TransportLink* describe el funcionamiento que sigue el programa para introducir las láminas de acero en la prensa. El *TransportLink* 1 une el *ProcessExecutor* inicial "*Feeder*" con el *ProcessExecutor* final "*Press*". El encargado de realizar este desplazamiento es el operario que se encuentra en la celda, tal y como se indica por el icono que se sitúa en el centro de la flecha del *TransportLink*.

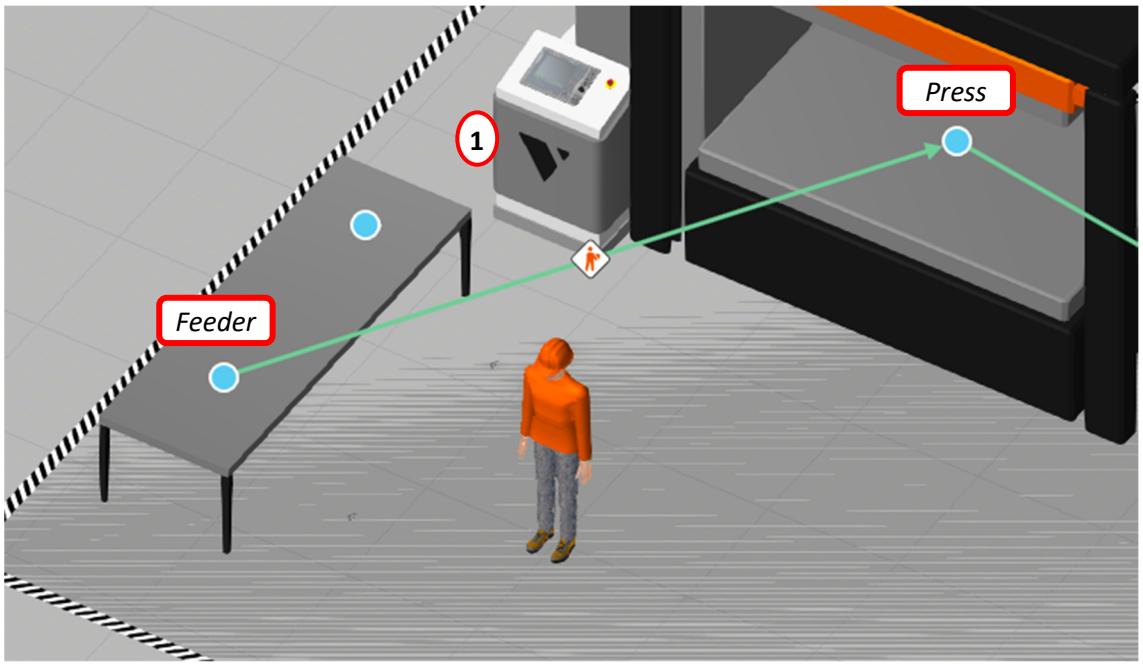


Figura 71. *TransportLink* 1: Desplazamiento entre el punto de almacenamiento inicial y la prensa.

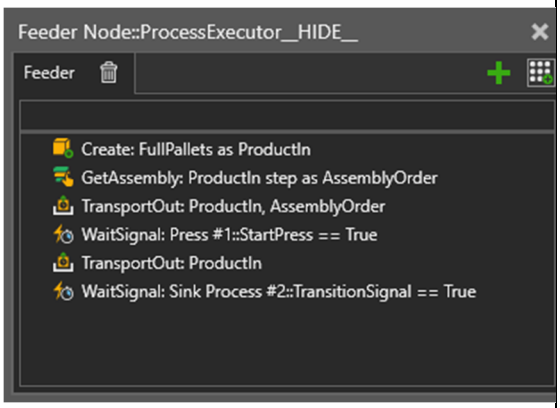
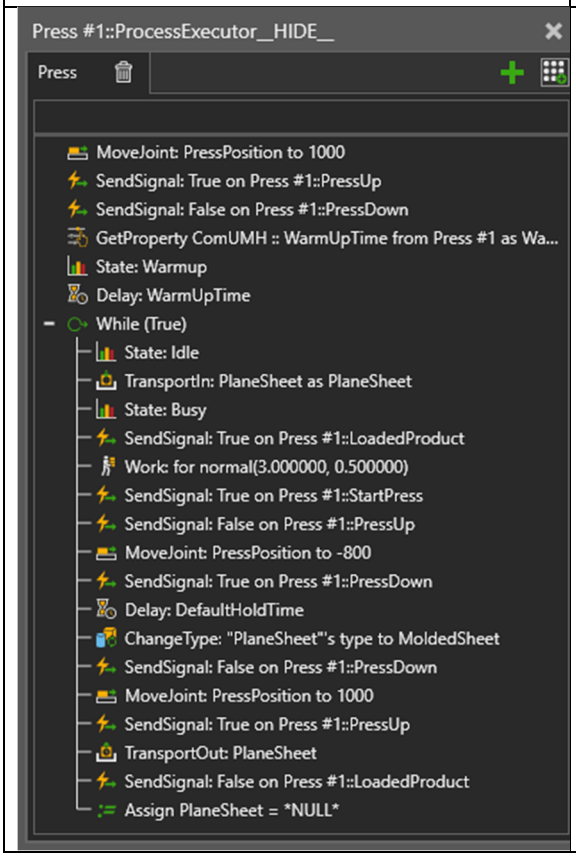
Los *ProcessExecutors* que intervienen en este *TransportLink* son los siguientes:

- Feeder: este PE tiene la función de crear un pallet lleno de láminas de acero, solicitar el transporte de estas a la prensa, y cuando se vacíe el pallet solicitar el transporte de este para su eliminación. A este flujo sólo le afectan las instrucciones de transporte de láminas a la prensa.

- *Press:* este *PE* controla en funcionamiento de la prensa. Al comienzo de este se reinician las variables de la prensa para que se encuentre en su estado inicial. Por ejemplo, independientemente de la posición en la que esté la prensa, al iniciar la simulación la prensa se situará en la posición más alta. Tras colocar la prensa en su estado inicial, entra en un bucle. En dicho bucle, es donde está el funcionamiento de la prensa, donde solicita productos, cuando estos entran la prensa baja y una vez terminado el prensado solicita que se extraigan.

A continuación, se describen las instrucciones que ejecutan los *ProcessExecutor* inicial y final que intervienen en el *TransportLink 1.*

Tabla 21. *ProcessExecutors* que intervienen en el *TransportLink1 del flujo Parts*.

Imagen del *ProcessExecutor*	Descripción
	*Feeder* 1. Create: Se crea el producto "*FullPallets*". 2. GetAssembly: Se obtiene la variable necesaria para poder extraer las láminas una a una del montaje que define el pallet con 10 láminas de acero. 3. TransportOut: Se transportan todas las láminas del pallet de una en una hacia la prensa. (Una vez no queda láminas en el pallet) 4. WaitSignal: Se espera a que el operario inicie el ciclo de prensado de la última lámina. 5. TransportOut: Se transporta el pallet hasta el *ProcessExecutor* denominado "Sink" para ser eliminado 6. WaitSignal: Se espera a que el pallet sea eliminado para reiniciar este *ProcessExecutor.*
	*Press* 1-6. Al inicio de la simulación se establecen los valores iniciales para la prensa, tanto su posición, sus señales y estado. -Las instrucciones "State" cambian el estado de la máquina, pero no altera su funcionamiento. 9. TransportIn: El nodo permanece a la espera de la entrada de productos del tipo PlaneSheet. 11. SendSignal: Una vez recibido producto, notifica a través de la señal *"LoadedProduct"* que hay un producto en la prensa. 12. Work: Instrucción que provoca que el operario se dirija al HMI de la prensa para emular el comportamiento de la vida real. 13. SendSignal: Se envía el mensaje "True" a través de la señal "*StartPress*" para indicar el comienzo del ciclo de prensado. 14. SendSignal: Se notifica que la prensa ya no está arriba. 15. MoveJoint: Se mueve la prensa hacia abajo. 16. SendSignal: Se notifica que la prensa está abajo. 17. Delay: Mantiene un determinado tiempo la prensa abajo. 18. ChangeType. Cambia el tipo de producto de *PlaneSheet* a *MoldedSheet.*

	19-21. Se notifican que la prensa no está abajo, se sube y se notifica que la prensa está arriba. 22. TransportOut: El nodo permanece a la espera de que se extraiga el producto. 23. SendSignal: Se notifica la extracción de producto. 24. Assign: Se reinicia la variable PlaneSheet.

2. *TransportLink* 2: Desplazamiento entre la prensa y el nodo donde se realiza la soldadura.

   El transporte de las planchas ya moldeadas se realiza mediante el AGV a través del *TransportLink2* que se muestra en la Figura 72. El transporte se da cuando el nodo de inicio espera que se extraiga un producto y el de destino espera la entrada de este.

   Este *TransportLink* queda definido por el *ProcessExecutor* de la prensa (“*Press*”) y el del nodo de soldadura (“*Weld*”). Además, se involucra el *ProcessExecutor “Weld ControlPC”* con la finalidad de emular el PC que emplea el operario para iniciar un proceso de soldadura.

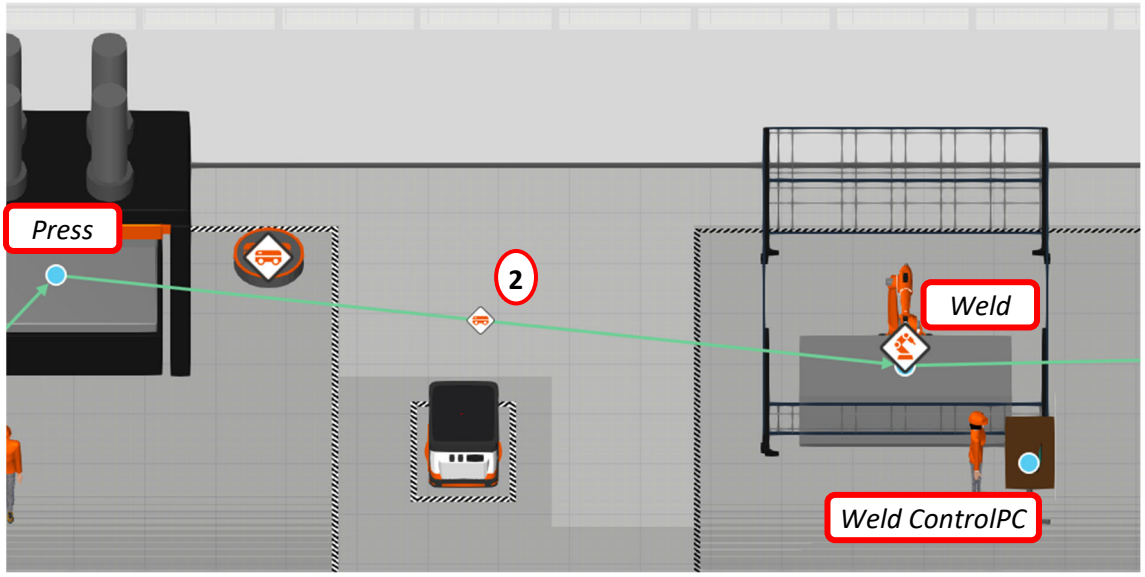


Figura 72. *TransportLink* 2: Desplazamiento entre la prensa y el nodo de soldadura.

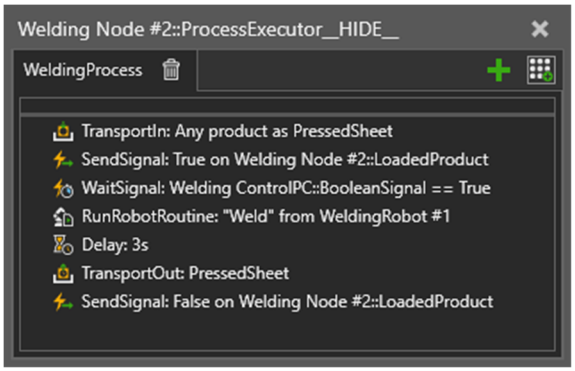
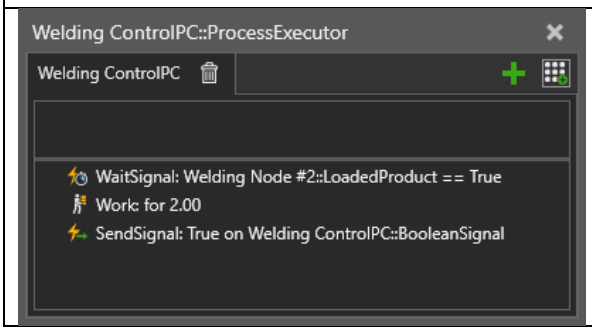
Dado que el PE *“Press”* ya se ha descrito en la Tabla 21, solo se describen el resto de PE involucrados en este *TransportLink2*.

- *Weld:* este PE describe las acciones que realiza el nodo de soldadura. Que serían recibir productos, realizar la soldadura de estos y solicitar su extracción. En primer momento espera a que se le introduzca un producto, una vez hecho esto notifica que está ocupado, espera que se le indique el inicio del ciclo de soldadura, una vez se le indica que inicie la soldadura ejecuta un programa para soldar los puntos especificados, solicita la extracción de producto y notifica que está libre para recibir nuevos productos.

- *Weld ControlPC:* este PE emula la situación en la que cuando hay una pieza cargada en el nodo de soldadura, el operario indica el inicio del ciclo de soldadura mediante el PC que se encuentra en dicha celda.

En la Tabla 22 se describe en mayor profundidad cada instrucción de los *PE* mencionados.

Tabla 22. *ProcessExecutors* que intervienen en el *TransportLink2* del flujo *Parts*.

Imagen del *ProcessExecutor*	Descripción
	*Weld* 1. TransportIn: Espera entrada de producto. 2. SendSignal: Notifica mediante una señal que se ha cargado un producto. 3. WaitSignal: Espera a que se envíe la señal de comandos proveniente del *ControlPC.* 4. RunRobotRoutine: Ejecuta los movimientos del robot para realizar la soldadura. 5. Delay: Espera 3 segundos. 6. TransportOut: Espera a que se extraiga el producto. 7. SendSignal: Notifica que el nodo está libre.
	*Weld ControlPC* 1. WaitSignal: Espera a que se introduzca un producto en el nodo de soldadura. 2. Work: Instrucción que provoca que el trabajador permanezca durante 2 segundo frente al *ControlPC* para emular el envío de comandos al robot articulado. 3. SendSignal: Envía la señal que emula los comandos de movimiento del robot.

3. *TransportLink 3*: *Desplazamiento* entre nodo de soldadura y el nodo de medición.

El tercer *TransportLink* es el *que* une la celda de soldadura y la de medición. Cuando la celda de soldadura termina su proceso, el operario de dicha celda, colocará la lámina de acero soldada sobre el AGV y este llevará el producto hasta la celda de medición. Una vez allí, se coloca dicha lámina en la ubicación indicada para la medición mediante la ayuda del operario y este mismo es el encargado de iniciar el ciclo de medida mediante el ordenador que está junto al CMM.

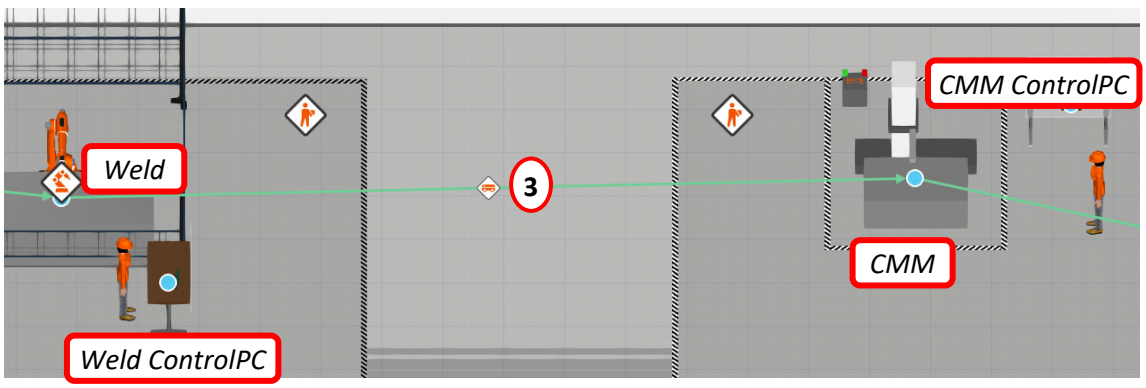


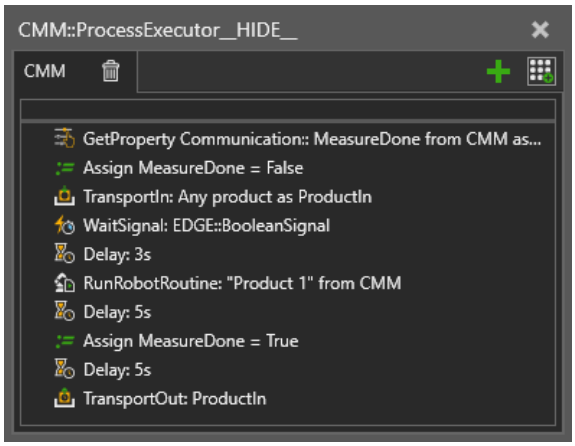
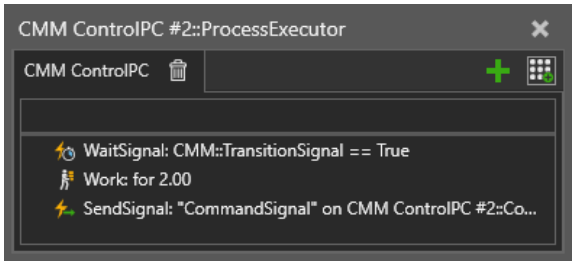
Figura 73. *TransportLink* 3: Desplazamiento entre el nodo de soldadura y el nodo de medición.

Los PE que intervienen en este *TransportLink* son el "*Weld*", "*CMM*" y "*CMM ControlPC*". Puesto que el PE "*Weld"* ya se ha descrito para el *TransportLink2* no se repetirá en este apartado.

- CMM**:** en este *PE* se lleva a cabo la solicitud de entrada de producto, una vez hecho esto espera hasta que el operario inicia el ciclo de medición y ejecuta la instrucción necesaria para que el propio componente CMM realice la medición. Finalizada la medición se indica mediante una propiedad que se ha terminado dicha medición y se solicita un transporte de salida.
- CMM ControlPC: su funcionamiento es exactamente igual que para el ControlPC de la celda de soldadura. Este *PE* emula la situación en la que cuando hay una pieza cargada en el CMM, y el operario indica el inicio del ciclo de medición mediante el PC que se encuentra en dicha celda.

La imagen y descripción de cada instrucción de los PE anteriores se desarrolla en la Tabla 23.

Tabla 23. *ProcessExecutors* que intervienen en el *TransportLink3* del flujo *Parts*.

Imagen del *ProcessExecutor*	**Descripción**
	<u>*CMM*</u> 1. GetProperty: obtiene la propiedad que indica si la medición se ha finalizado. 2. Assign: indica que la medición no se ha realizado. 3. TransportIn: queda a la espera de recibir algún producto. 4. WaitSingal: espera a recibir la señal del EDGE con los comandos para iniciar la medición. 5. Delay: espera 3 segundos. 6. RunRobotRoutine: inicia la medición de la pieza. 7. Delay: una vez terminada la medición de la pieza espera unos segundos. 8. Assign: indica que se ha completado la medición de la pieza. 9. Delay: mantiene ese valor durante unos segundos. 10. TransportOut: espera a que se extraiga el producto.
	<u>*CMM ControlPC*</u> 1. WaitSignal: Espera a que se introduzca un producto en el nodo de soldadura. 2. Work: Instrucción que provoca que el trabajador permanezca durante 2 segundo frente al *ControlPC* para emular el envío de comandos al robot articulado. 3. SendSignal: Envía la señal que emula los comandos de movimiento del robot.

4. *TransportLink 4 y 5*: Desplazamiento entre nodo de medición y la celda almacén.

   Los dos últimos *TransportLink,* el 4 y el 5 se describen juntos dada su simplicidad. El 4 parte de la celda de medición y mediante el AGV la lámina de acero es

transportada a la celda de almacén. Una vez en la celda de almacén, no existe un *ProcessExecutor* que gestione que acciones realizar, directamente el operario coge la lámina del AGV y la lleva hasta las estanterías, a través del *TransportLink 5.* Por *TransportLink 5* se entienden las 3 flechas que unen la ubicación desde donde el AGV lleva las piezas, hasta las estanterías. Para el programa, estas flechas actúan como una sola. El transporte se lleva a cabo siempre y cuando las estanterías tengan huecos libres.

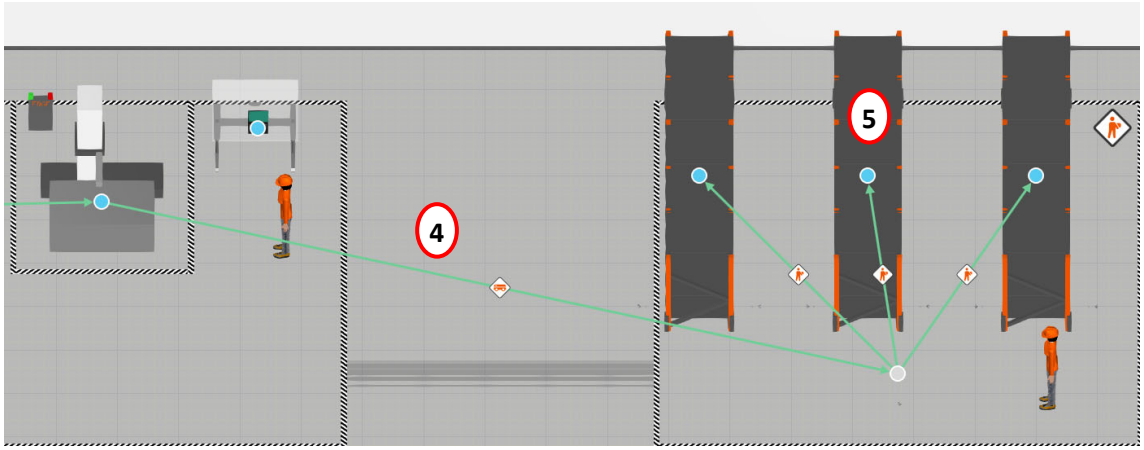


Figura 74. *TransportLink* 4-5: Desplazamiento entre la celda de medición y el almacén.

Con el transporte de las láminas a las estanterías concluiría el flujo “Parts” de este escenario.

10.2.2. Implementación del escenario 2: Planta de prensado de láminas de acero para puertas de automóviles

Este escenario, el cual se presentó en el apartado 8.2, consiste en el prensado de láminas de acero para puertas de automóviles. Se pueden distinguir tres bloques dentro de él, el primero sería referente las tareas de gestión del almacén de entrada, en el que se engloba el almacén, los AGVs encargados de transportar los productos a las líneas de prensa y los controladores necesarios para llevar a cabo las tareas anteriores. En segundo lugar, se identifica una zona de prensado, donde existen 3 líneas paralelas que mediante varias prensas dan la forma deseada a las láminas de acero. Por último, se dispone de una zona de control de calidad y almacén de salida.

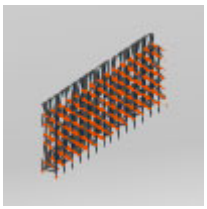
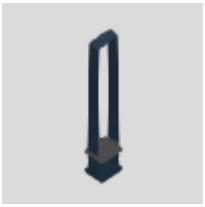
En esta sección nos centraremos en el modelado de este escenario y como se ha desarrollado. Al igual que en el apartado anterior, se exponen los componentes que intervienen en este escenario, sus productos y sus flujos.

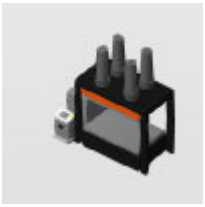
10.2.2.1. Componentes del escenario 2

Para identificar los componentes de este escenario se ha seguido un determinado criterio, de tal manera que se pueda identificar o ubicar el componente a través de su nombre. Este criterio se ha aplicado a componentes repetidos (tres líneas de prensa iguales), en los cuales el número después del “#” indica la línea de prensa a la que pertenecen (1, 2 o 3). Por ejemplo, para las prensas, dado que hay tres en la primera línea sus nombres serían Press 1 #1, Press 2 #1 y Press 3 #1. Mediante estos identificadores podemos saber de qué línea se está hablando (#1) y de que elemento (1, 2 o 3).

Los componentes más representativos del escenario se muestran en la Tabla 24.

Tabla 24. Lista de c*omponentes del Escenario 2*.

Identificador	Descripción	Imagen
GlobalMonitorSystem	Componente que emula un servidor al que se le envían los datos de estadísticas del resto de componentes del escenario.	
InboundShelf # - OutboundShelf #	Estanterías que almacenan ya sean los productos de entrada o los de salida (Inbound, Outbound).	
SC In – Crane Controller - SC Out – Crane Controller	Componente que implementa el control de la grúa que extrae o almacena productos en las estanterías.	
Single Stacker Crane – In - Single Stacker Crane – Out	Grúa empleada para realizar los transportes hacia fuera o hacia dentro de las estanterías.	
AGV #	Robot móvil encargado de transportar las láminas de acero hasta las líneas de prensa.	
SC – AGV Controller	Componente que implementa el control de los AGVs.	
StorageController - In - StorageController - Out	Este componente emula un controlador local de almacén que contiene tanto al controlador de grúa como al controlador de AGVs mencionados más arriba.	
FeedLineRobot 'X' #	Robot situado al inicio de cada línea de prensa y las alimenta. Se encarga de coger láminas planas y cargarlas a la cinta transportadora.	

FeedPressRobot ‘X’ #	Robot situado al inicio o fin de cada prensa. Se encarga de introducir o extraer dichas láminas de la prensa más cercana. Están boca abajo como se ve en la imagen.	
FailLineRobot #	Robot situado en el nodo de control de calidad. Se encarga de extraer las láminas defectuosas.	
RobotController ‘X’ #	Componente encargado de controlar los movimientos de un robot. Se sitúa justo bajo cada robot.	
Press ‘X’ #	Prensa encargada de moldear las láminas de acero entrantes.	
QualityCamera #	Componente que emula una cámara en el control de calidad.	
CameraController #	Componente que emula un PC que contiene el controlador de la cámara de calidad. Se sitúa en una mesa contigua a la salida de cada línea de prensa.	

Los componentes mencionados en la tabla anterior han recibido modificaciones respecto a su estado por defecto. Todos ellos han sufrido la adición de los comportamientos necesarios para implementar el módulo de comunicaciones que se describe en el apartado 10.3. Para conocer cuáles son dichos comportamientos y su función dirigirse al apartado ya mencionado 10.3.

Por otro lado, algunos de dichos componentes han sufrido modificaciones adicionales y son las que se comentan a continuación. Lo componentes con modificaciones adicionales son: controladores (ya sean de AGV, de robots o de grúa) y las prensas.

- Controladores – AGV, robots y grúa. Todos los controladores que empleen un recurso para realizar un transporte han sufrido una modificación en su script de funcionamiento que permite identificar el instante en el que el controlador asigna un transporte a uno de sus recursos y el recurso al que se lo asigna. Esta información se requiere para emular las comunicaciones producidas en la vida real. Para esta modificación se emplean

algunos de los comportamientos del módulo de comunicaciones inalámbricas. Dichos comportamientos son el *ComTriggerSignal,* que contiene la situación que se da en el escenario, en este caso una asignación de transporte, y como segundo comportamiento añadido encontramos *ComInfoSignal,* que contiene información necesaria para el mensaje que queremos enviar, en este caso el recurso (AGV, robot o grúa) que realizará el transporte.

Los scripts de funcionamiento de todos los controladores tienen la misma forma, se comentan las zonas donde se han realizado las modificaciones y se indican las líneas modificadas del script para cada controlador.

Los comportamientos script modificados para el controlador de AGVs y robots se denominan "*TaskLogic*" y en el caso del controlador de la grúa se denomina "*MainLogic*", pero como ya se ha comentado están estructurados de la misma manera y su código es muy similar. Todas las modificaciones realizadas tienen al final el comentario "#UMH" que permite localizar rápidamente las líneas modificadas.

Las modificaciones realizadas a dichos scripts son las siguientes:

- Obtención de objetos desde el script. Para poder asignar valores a un comportamiento en primer lugar hay que obtener dicho comportamiento (objeto) desde el código para tener acceso a él. Esto se hace casi al final del código en un apartado denominado "## BEHAVIOUR HADLES". Aquí se obtienen los objetos de los comportamientos que deseamos emplear. Los identificadores de los objetos son *"comTriggerSignal"* para el comportamiento "*ComTriggerSignal*" y "*comInfoSignal*" para el comportamiento "*ComInfoSignal*". En la Figura 75 se muestra como se obtienen dichos objetos en el código.

```
## BEHAVIOR HANDLES
trigger_signal = comp.findBehaviour("Trigger")
resource_interfaces = [comp.findBehaviour("CraneA"), comp.findBehaviour("CraneB")]
task_container = comp.findBehaviour("TaskContainer")

comTriggerSignal = comp.findBehaviour('ComTriggerSignal')    #UMH
comInfoSignal = comp.findBehaviour('ComInfoSignal')          #UMH
```

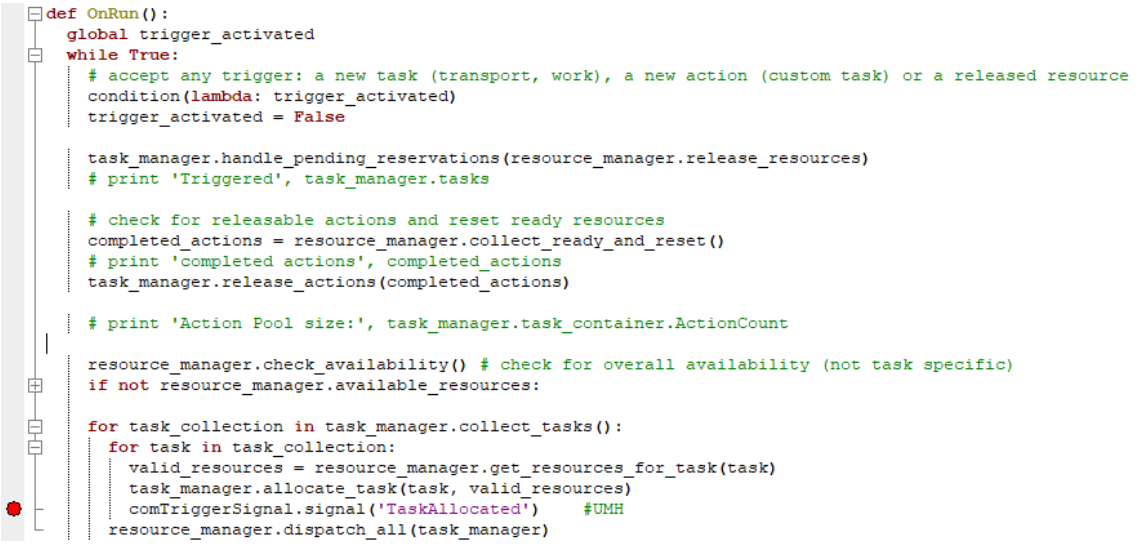
Figura 75. Obtención de comportamientos desde un script.

El número de línea en el que se encuentran estas modificaciones son:

  - Controlador de la grúa: 1112-1113
  - Controlador robots: 1596-1597
  - Controlador AGVs: 1440-1441

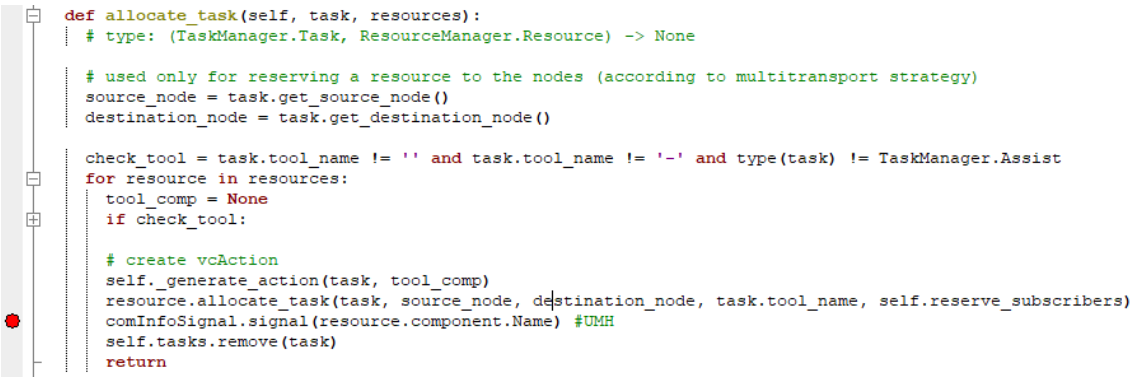
- Asignación de valores a los objetos ya declarados. Una vez ya tenemos acceso a los objetos que permiten dar valores a los comportamientos, es necesario saber en qué momento se han de asignar dichos valores. Ahora es necesario ubicar las secciones de código donde primero, se asigna un transporte y segundo, a quién se asigna dicho transporte. En primer lugar, la asignación de transporte se realiza dentro de la función *OnRun()* de los scripts mencionados, siendo en esta función donde se usa *comTriggerSignal* para notificar que se ha asignado un transporte (ver Figura 76). En segundo lugar, la decisión de que recurso se va

a emplear para realizar el transporte se localiza dentro de la función *allocate_task* de la clase *TaskManager.* Dentro de esta función se utiliza *ComInfoSignal* para notificar el recurso que se va a emplear para el transporte (ver Figura 77).



```
def OnRun():
    global trigger_activated
    while True:
        # accept any trigger: a new task (transport, work), a new action (custom task) or a released resource
        condition(lambda: trigger_activated)
        trigger_activated = False

        task_manager.handle_pending_reservations(resource_manager.release_resources)
        # print 'Triggered', task_manager.tasks

        # check for releasable actions and reset ready resources
        completed_actions = resource_manager.collect_ready_and_reset()
        # print 'completed actions', completed_actions
        task_manager.release_actions(completed_actions)

        # print 'Action Pool size:', task_manager.task_container.ActionCount

        resource_manager.check_availability() # check for overall availability (not task specific)
        if not resource_manager.available_resources:

        for task_collection in task_manager.collect_tasks():
            for task in task_collection:
                valid_resources = resource_manager.get_resources_for_task(task)
                task_manager.allocate_task(task, valid_resources)
                comTriggerSignal.signal('TaskAllocated')    #UMH
            resource_manager.dispatch_all(task_manager)
```

Figura 76. Notificación de una asignación de transporte.



```
def allocate_task(self, task, resources):
    # type: (TaskManager.Task, ResourceManager.Resource) -> None

    # used only for reserving a resource to the nodes (according to multitransport strategy)
    source_node = task.get_source_node()
    destination_node = task.get_destination_node()

    check_tool = task.tool_name != '' and task.tool_name != '-' and type(task) != TaskManager.Assist
    for resource in resources:
        tool_comp = None
        if check_tool:

        # create vcAction
        self._generate_action(task, tool_comp)
        resource.allocate_task(task, source_node, destination_node, task.tool_name, self.reserve_subscribers)
        comInfoSignal.signal(resource.component.Name) #UMH
        self.tasks.remove(task)
        return
```

Figura 77. Notificación de recurso empleado para un transporte.

El número de línea en el que se encuentran estas modificaciones son:

- Controlador de la grúa:
  - Notificación de transporte: 47
  - Información de recurso empleado: 540
- Controlador robots:
  - Notificación de transporte: 94
  - Información de recurso empleado: 875
- Controlador AGVs:
  - Notificación de transporte: 92
  - Información de recurso empleado: 825

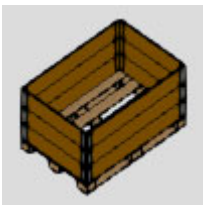
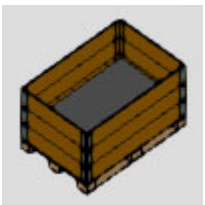
- Prensas. Las prensas empleadas en este escenario son las mismas que la del escenario 1. Básicamente son los comportamientos necesarios para trabajar con este tipo de prensa en *ProcessModeling.* La descripción de sus modificaciones se encuentra en profundidad en el apartado 10.2.1.1.

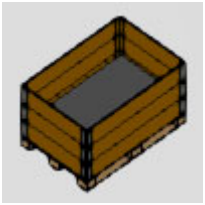
10.2.2.2. Productos del escenario 2

Para este escenario se han definido una serie de productos que permiten simular el funcionamiento el prensado de láminas, láminas fallidas, etc. Los productos se pueden observar como ya se ha comentado desde la pestaña de procesos con la opción productos (ver Figura 78).

La lista de productos definidos está recogida en la Tabla 25.

Tabla 25. Lista de productos definidos en el Escenario 2.

Producto	Descripción	Imagen de producto
PlaneSheet	Láminas de acerdo planas. Materia prima inicial.	
MoldedSheet	Láminas de acero moldeadas.	
FailSheet	Láminas de acero que no pasan el control de calidad.	
Pallet	Pallet vacío.	
InBoundPallet	Montaje de un pallet con 10 láminas de acero planas.	

OutPallet	Montaje de un pallet con 10 láminas de acero moldeadas. Permite modificar el número de láminas que caben en el pallet de salida respecto del de entrada.	

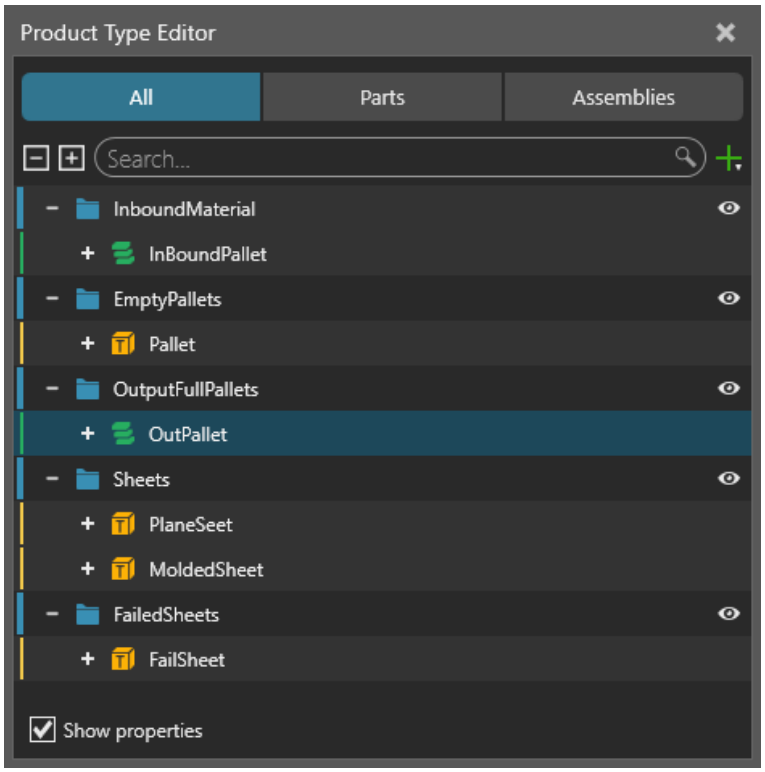


Figura 78. Ventana de Editor de producto del Escenario 2.

10.2.2.3. Flujos del escenario 2

En el escenario actual se pueden diferenciar cuatro zonas: una zona inicial de gestión de almacén de entrada, una segunda zona de producción, una tercera donde se lleva a cabo un control de calidad y por último una de gestión del almacén de salida. Para cada una de estas zonas se ha implementado un flujo distinto, que describe el movimiento de los productos por cada una de dichas zonas. En la Figura 79 se muestra el Editor de Flujo donde se muestran todos ellos y algunos de los *ProcessExecutors* que intervienen.

Figura 79. Editor de flujo del Escenario 2.

Se va a realizar la descripción del flujo de una sola línea de prensas, dado el gran número de *TransportLinks* que tiene este escenario y sería confuso mostrarlos todos.

- *Inbound Pallets*: Primer flujo del escenario cuya función es llevar pallets llenos de láminas planas a las entradas de las líneas de prensa. Una vez se solicite material en las líneas de prensa, el pallet viajará desde el almacén hasta el inicio de la cinta transportadora mediante la grúa. Una vez depositado el producto en el inicio de la cinta, este viajará hasta el final de la cinta. Llegado el pallet al final de la cinta se solicita que un *AGV* transporte el pallet hasta la ubicación de la línea de prensa con necesidades de material (ver Figura 80).

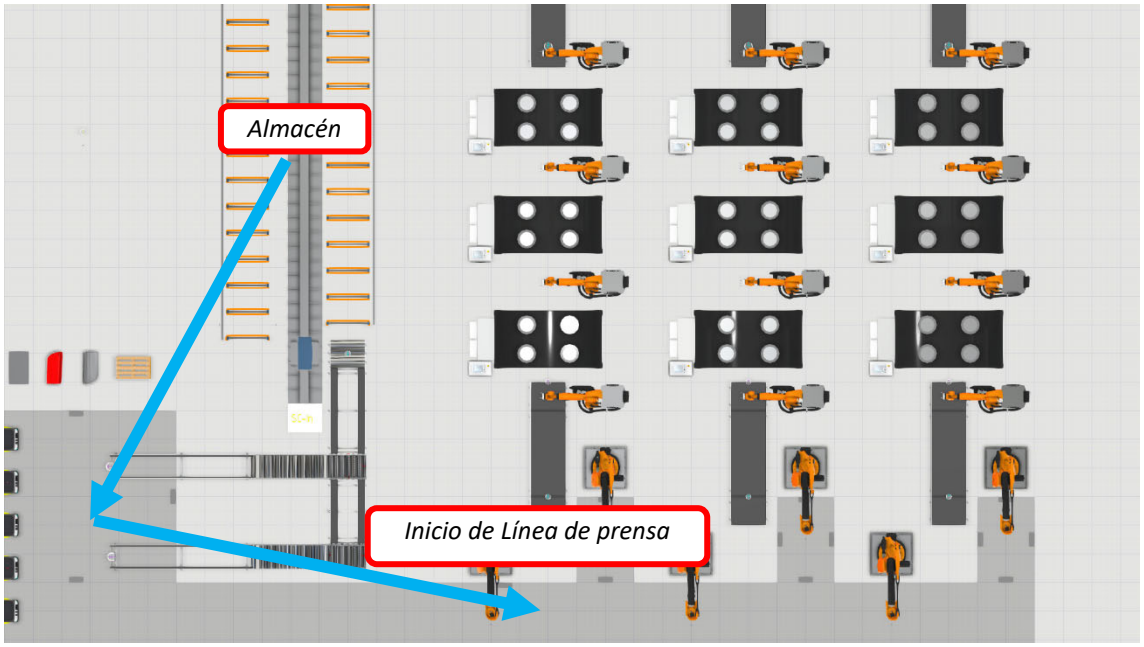


Figura 80. Flujo simplificado del almacén de entrada del escenario 2.

- *Sheets*: Este flujo representa el camino que siguen las láminas de acero a través del escenario. Parte de los nodos que alimentan cada línea de prensa y llega hasta el final de dichas líneas donde las láminas son paletizadas para su almacenamiento (ver Figura 81).

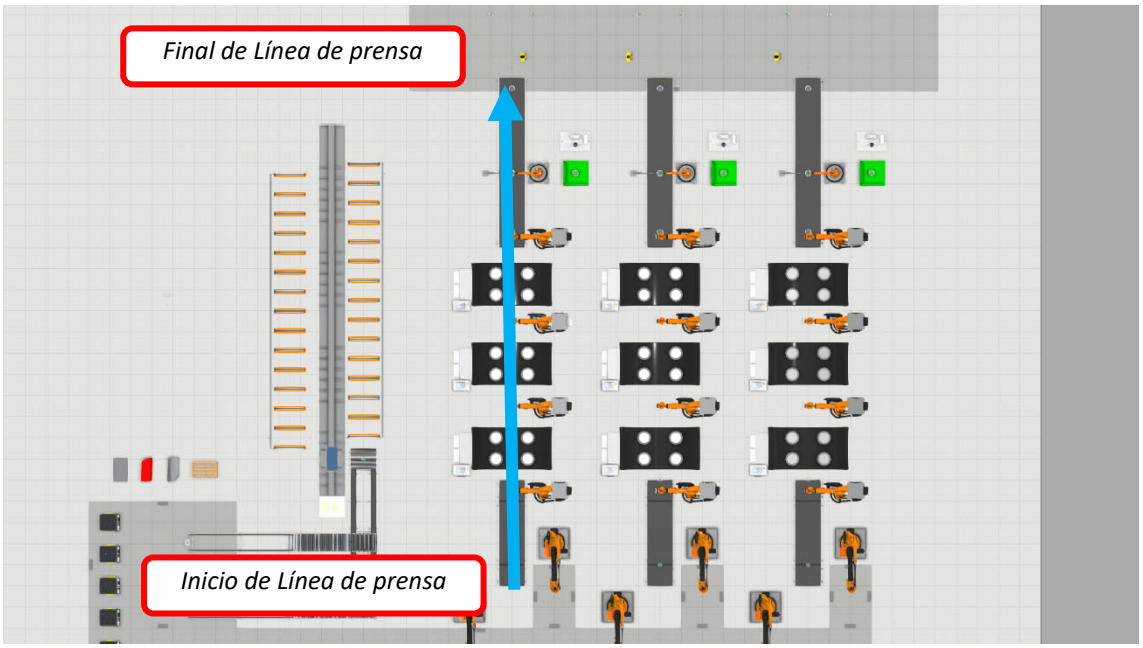


Figura 81. Flujo simplificado de las láminas de acero a través del escenario 2.

- *Fail_sheets:* Este flujo describe el movimiento de las láminas que no pasan el control de calidad. Cada lámina es comprobada individualmente y en caso de no cumplir los parámetros establecidos es transportada mediante un robot articulado a un contenedor de deshechos (ver Figura 82).

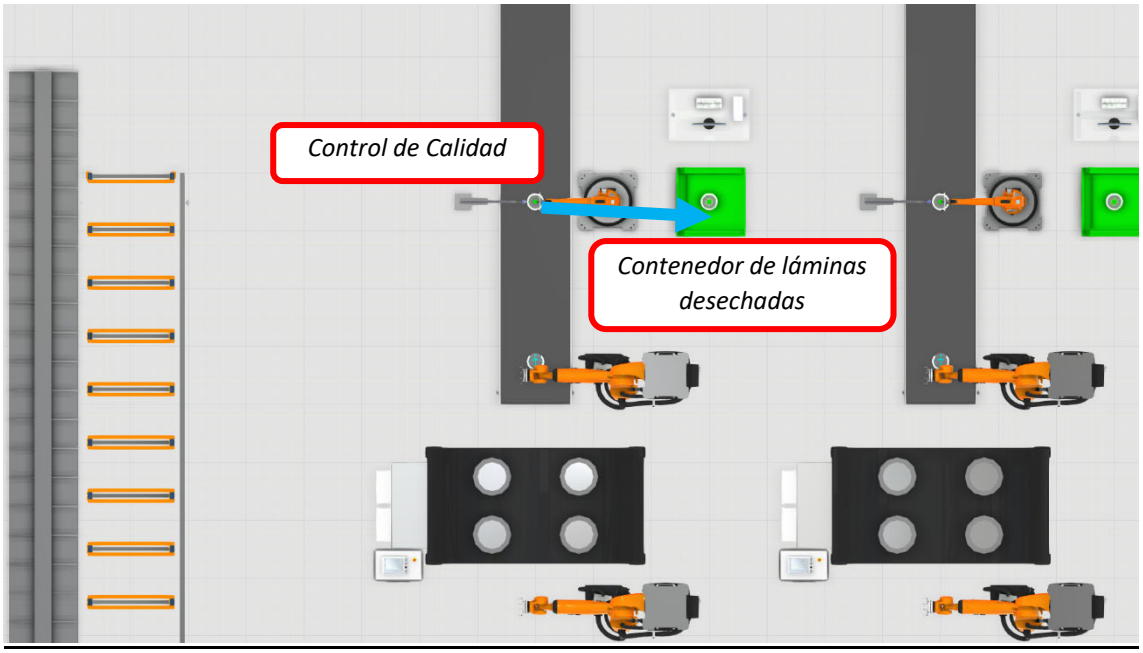


Figura 82. Flujo simplificado de las láminas de acero fallidas del escenario 2.

- *Outbound_Pallets:* Último flujo del escenario, una vez un pallet a la salida de una línea de prensa está lleno, este se transporta hasta el almacén de salida (ver Figura 83).

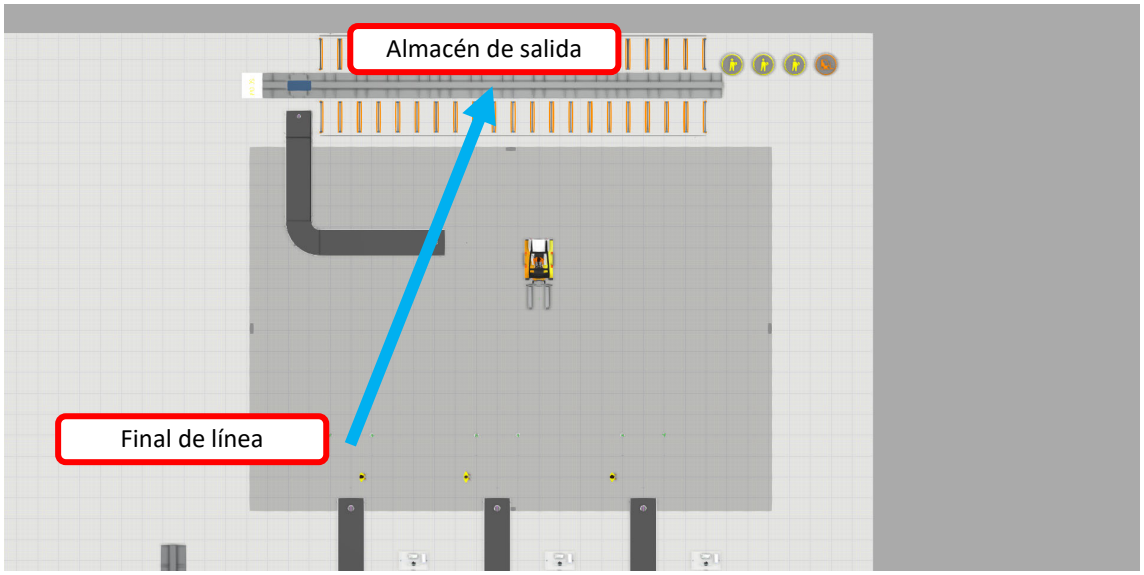


Figura 83. Flujo simplificado del almacén de salida del escenario 2.

A continuación, se describen más detalladamente los *ProcessExecutors* que intervienen en cada flujo mencionado. Los *ProcessExecutors* se observan como hexágonos achatados en la ventana de editor de flujo (ver Figura 79).

Antes de comenzar a explicar cada *ProcessExecutor,* se describen dos de ellos, los cuales se repiten en muchas ocasiones y función es la de simular un sensor y notificar mediante una señal cuando cambia el estado de ese sensor. Dichos *ProcessExecutors* son los denominados "*FromConveyor*" y "*ToConveyor*". Estos PE tienen la forma definida en la Figura 84.

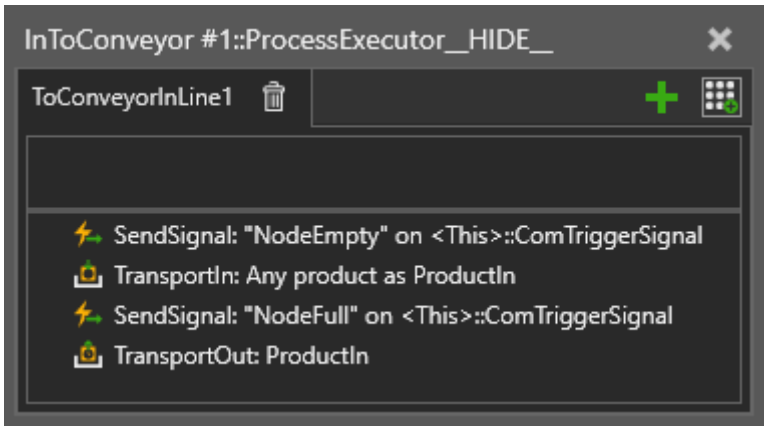


Figura 84. Forma general de los PE *FromConveyor* y *ToConveyor* del escenario 2.

En la Figura 84 se observan las siguientes instrucciones:

- *SendSignal*: envía el mensaje "*NodeEmpty*" a través del comportamiento *ComTriggerSignal.* Con el envío de esta señal se pretende notificar el estado del sensor.
- *TransportIn*: esta instrucción mantiene la ejecución en espera de recibir un producto.

- *SendSignal*: una vez el nodo recibe un producto, envía el mensaje "*NodeFull*" a través del comportamiento *ComTriggerSignal.* Con el envío de esta señal se pretende notificar el estado del sensor.
- *TransportOut*: por último, mantiene la ejecución en espera de que el producto sea extraído.

De ahora en adelante, aunque estos *ProcessExecutors* intervengan en el flujo que se esté describiendo, no se repetirá su explicación. Ahora sí, se describen cada uno de los *ProcessExecutors* que intervienen en cada flujo.

- *Inbound_Pallets.* Tal y como se ha descrito en páginas anteriores, este flujo describe el camino que siguen los pallets llenos desde el almacén hasta el inicio de las líneas de prensa. Las ubicaciones de estos PE se pueden observar en la Figura 85.
    - Feeder: crea pallets llenos de láminas que llevan el almacén de entrada automáticamente.
    - InShelfBuffer: *PE* ubicado en las estanterías que realiza la función de almacenar los pallets llenos de láminas y permitir su salida.
    - InLine1: *PE* ubicado al inicio de las líneas de prensa que cuando está vacío, solicita un pallet lleno de láminas y cuando se vacía el pallet, lo elimina y solicita un nuevo pallet lleno de láminas.

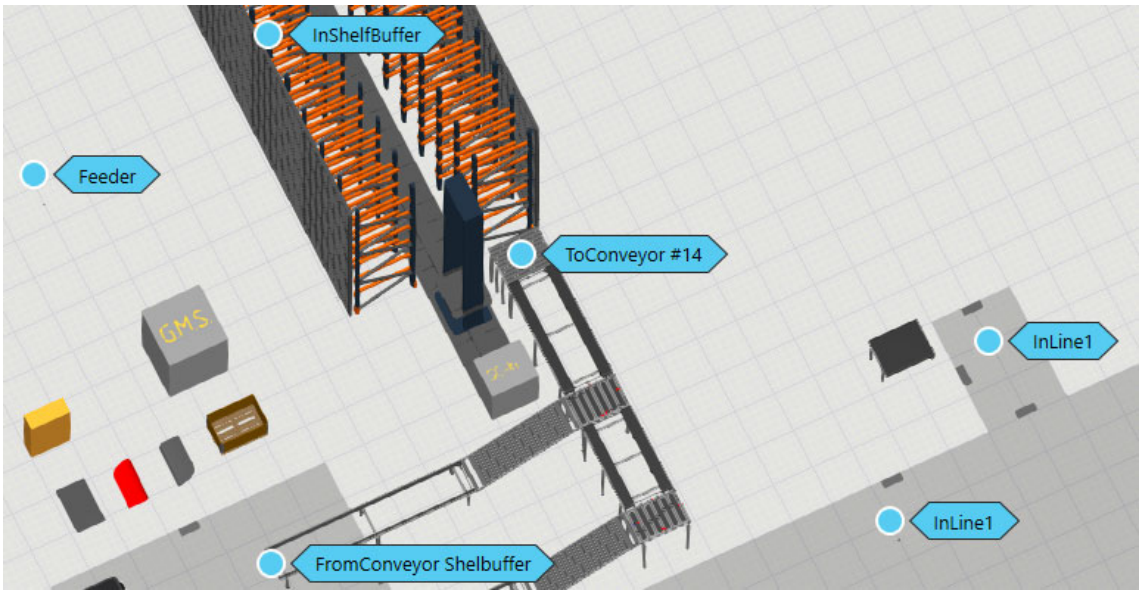


Figura 85. *ProcessExecutors* del flujo del almacén de entrada del escenario 2.

- *Sheets.* El flujo *Sheets* describe el movimiento de las láminas de acero planas a través de las líneas de prensa. Durante este flujo se llevan a cabo los siguientes *ProcessExecutors*.
    - InLine1: PE ubicado en los nodos al inicio de las líneas de prensado, el cual, cuando está vacío, solicita un pallet lleno de láminas y cuando se vacía este, lo elimina y solicita un nuevo pallet lleno de láminas.
    - *Press #*: ejecuta las acciones que permiten a la prensa funcionar y dar forma al material.
    - *Inspection*: PE ubicado en el nodo del control de calidad. Este es el encargado de comunicar al controlador de la cámara detectora de defectos que haga el análisis sobre la plancha.

- PalletProcess11: PE mostrado en la Figura 86 que crea pallets vacíos para ser llenados por los operarios. Una vez que el pallet está lleno avisa al controlador del vehículo para que los retire.

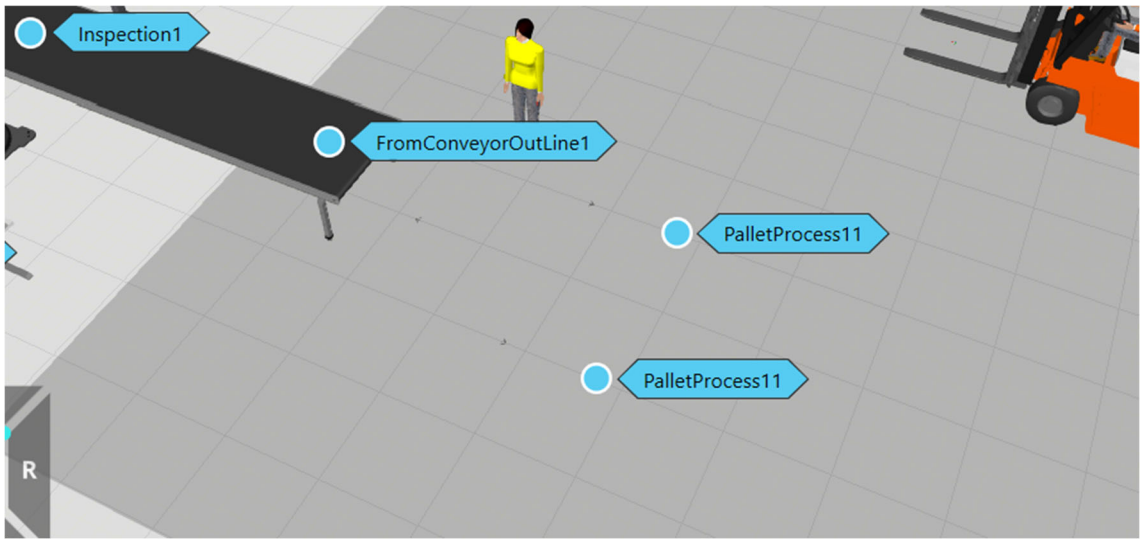


Figura 86. *ProcessExecutors* finales de la zona de prensado.

- *Fail_Sheets*
  - Inspection: PE ubicado en el nodo del control de calidad. Este es el encargado de comunicar al controlador de la cámara detectora de defectos que haga el análisis.
  - ScrapPoint: PE ubicado en el contenedor de material defectuoso que se encarga de recibir el material depositado mediante el brazo robot.
  - BrokenSint: PE dedicado a hacer desaparecer de la simulación el objeto desechado y evitar acumulación de objetos virtuales que ralenticen la simulación.
- *Outbound_Pallets*
  - OutShelfBuffer: *PE* ubicado en las estanterías que realiza la función de almacenar los pallets llenos de láminas y permitir su salida.

10.3.Módulo de comunicaciones inalámbricas

Una aportación muy relevante de este proyecto ha sido el modelado del intercambio de información entre distintos elementos que integran los escenarios industriales modelados. Este intercambio de información se realizará utilizando una tecnología inalámbrica, siendo una candidata con gran potencial la tecnología 5G. La información y mensajes a intercambiar entre distintos elementos del escenario se identificó y se describió en el apartado 9. En este apartado se presenta cómo se ha implementado esta nueva funcionalidad dentro del software Visual Components. Es importante también destacar el valor que añade este trabajo a la herramienta software Visual Components, ampliando sus funcionalidades.

Esta nueva funcionalidad se ha implementado mediante la definición de nuevos comportamientos a cada componente. La definición de estos nuevos comportamientos es común para todos los componentes para que exista una uniformidad en la definición de los

mismos. En función de las necesidades concretas, cada componente puede utilizar funciones específicas dentro de cada comportamiento. De esta manera, todos los componentes dispondrán de todas las herramientas disponibles para implementar las comunicaciones deseadas en cualquier momento, pudiendo hacer uso de ellas en función de las necesidades concretas. Los comportamientos implementados son de tipo señal y de tipo script, las clases correspondientes a estos objetos son *vcSignal* y *vcScript*. Los comportamientos añadidos se describen brevemente a continuación, estando totalmente desarrollados en los siguientes apartados.

- *CommunicationModule:* comportamiento de tipo señal, más concretamente señal de tipo "*string"*. Esta señal contiene los parámetros que definen el mensaje a transmitir (por ejemplo, fuente y destino de la comunicación, tamaño del mensaje, latencia requerida, etc.).
- *CommunicationModuleScript:* comportamiento de tipo script. Este script monitoriza la señal *ComTriggerSignal*. Cuando *ComTriggerSignal se activa*, asigna el valor correspondiente a la señal *CommunicationModule.* Por ejemplo, cuando se asigna un valor "*TransportRequest*" a *ComTriggerSignal,* el *CommunicationModuleScript* envía el mensaje correspondiente por la señal *CommunicationModule*.
- *ComTriggerSignal:* comportamiento de tipo señal, concretamente "*string*". Se emplea para notificar al *CommunicationModuleScript* las condiciones de disparo que provocan el envío de un determinado mensaje*.* Por ejemplo, cuando un nodo se queda sin productos, envía el mensaje "*TransportRequest*" a través de *ComTriggerSignal.*
- *ComInfoSignal:* comportamiento de tipo señal, concretamente "*string*". Se emplea para notificar al *CommunicationModuleScript* determinada información de forma dinámica, es decir durante la simulación, para actualizar campos del mensaje, por ejemplo. Por ejemplo, cuando un controlador de AGVs asigna el recurso encargado de realizar un transporte, se notifica al *CommunicationModuleScript* mediante *ComInfoSignal* el nombre del recurso.
- *ComConditionScript:* comportamiento de tipo script, empleado para enviar a través de *ComTriggerSignal* las condiciones de disparo, o envío de un determinado mensaje. Su función es la de monitorizar situaciones específicas del escenario que requieran el envío de un mensaje. Para enviar dicho mensaje se notifica la situación mediante *ComTriggerSignal* y *CommunicationModuleScript* se encarga de enviar el mensaje correspondiente en la señal *CommunicationModule.*

En la Figura 87 se muestran los distintos comportamientos que conforman el módulo de comunicaciones inalámbricas.

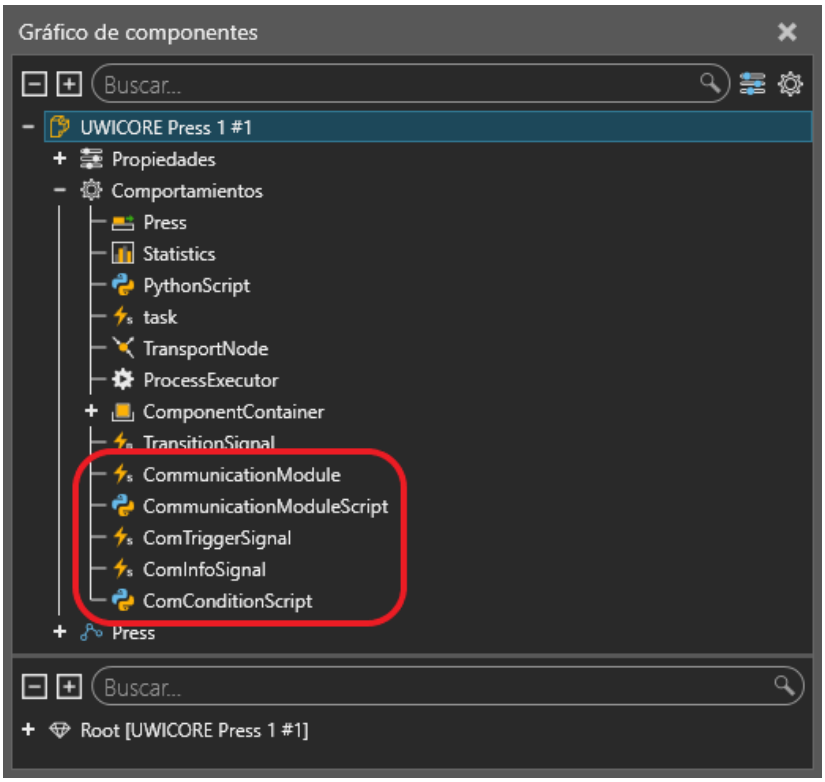


Figura 87. Ejemplo de comportamientos que forman el Módulo de Comunicaciones Inalámbricas.

10.3.1. CommunicationModule

El *CommunicationModule* es un comportamiento de tipo *vcStringSignal* que contiene los parámetros que modelan una comunicación inalámbrica. El mensaje contenido en *CommunicationModule* (parámetros que describen la comunicación) tiene un formato específico. Dicho formato consiste en separar cada parámetro por un ";" y que los parámetros siempre estén en el mismo orden. El formato tendría el siguiente aspecto:

(SENDER ; RECEIVER ; MSG TYPE ; COMMUNICATION TYPE ; PERIOD ; MSG SIZE ; LATENCY ; RELIABILITY ; TECH USED ; MSG VALUE; ACK REQUIRED)

Cada parámetro contenido en el mensaje es una cadena de caracteres y tiene el siguiente significado:

- `SENDER`: contiene el nombre del componente que emite el mensaje.
- `RECEIVER`: contiene el nombre del componente al que va dirigido el mensaje.
- `MSG TYPE`: contiene un identificador de mensaje. Por ejemplo, cuando un controlador manda comandos para realizar un transporte tomaría el valor "*TransportCommands*".
- `COMMUNICATION TYPE`: contiene una "p" o una "a" dependiendo si la comunicación que se está emulando es periódica o aperiódica respectivamente.
- `PERIOD`: contiene el valor del periodo del mensaje en milisegundos.
- `MSG SIZE`: contiene el valor del tamaño del mensaje en bytes.
- `LATENCIA`: contiene el valor de la latencia correspondiente a la comunicación emulada en milisegundos.
- `RELIABILITY`: contiene el valor en tanto por ciento de la fiabilidad de la comunicación.

- TECH USED: contiene un “wl” (*wireless*) o “wd” (*wired*) dependiendo si se quiere modelar la comunicación como inalámbrica o alámbrica respectivamente. Permite describir todas las comunicaciones existentes en un escenario y filtrar si se quieren modelar como inalámbricas o alámbricas.
- MSG VALUE: contiene el valor del mensaje que se enviaría en la comunicación modelada. Puede contener el valor de una coordenada si la comunicación emula el envío de la posición de un componente. Si no se le quiere dar ningún valor se le asignará un “-1”.
- ACK REQUIRED: contiene un “True” o un “False” en función de si la comunicación emulada necesita de un mensaje de confirmación por parte del receptor o no.

Es importante atenerse a este formato dado que, si se cambia, los *datasets* descritos en el capítulo 11 no se crearán correctamente.

10.3.2. CommunicationModuleScript

El *CommunicationModuleScript* es el comportamiento encargado de percibir las situaciones del entorno que requieren una comunicación. Esta percepción de las situaciones del entorno es gracias al *ComTriggerSignal,* que le indica al *CommunicationModuleScript* cuando aparece una situación que requiere comunicación y qué situación es concretamente. Además, cuando recibe a través de *ComTriggerSignal* una nueva situación del entorno envía a través de *CommunicationModule* el mensaje que describe los parámetros de dicha comunicación.

En este apartado, se documenta el script encargado de emular las comunicaciones en cada componente. El *CommunicationModuleScript*, tiene la misma forma para todos los componentes, teniendo simplemente que modificar algunas variables o parámetros para configurar las comunicaciones de cada componente.

El *CommunicationModuleScript* tiene siete zonas diferenciadas que se describen a continuación:

1. INITIAL DESCRIPTION

   Esta zona se encuentra al inicio del script y contiene una breve explicación del funcionamiento del script y el formato de mensaje a emplear (ver Figura 88).

```
#-----------------------------------------------------------------------------------------------------------------------------------------------
# Things to know to use this Script:
#  -Use the ComTriggerSignal as a Trigger to send a certain msgType, the msgType depends on the value of ComTriggerSignal
#  -Set the different msgTypes at the end of the Script
#  -Set the pair of ComTriggerSignal value and the msgType to send in the last Dictionary 'conditionsDic'
#  -All the fields of the msg are passed in a unique string divided by ';' as you can see below
#
# Format of the msg sent:
# (SENDER ; RECEIVER ; MSG TYPE ; COMMUNICATION TYPE ; PERIOD ; MSG SIZE ; LATENCY ; RELIABILITY ; TECH USED ; MSG VALUE; ACK REQUIRED)
#
# Ejemplo:
#   SENDER: Mobile Robot Transport Controller
#   RECEIVER: Mobile Robot Resource
#   MSG TYPE: TransportCommand
#   COMMUNICATION TYPE: PeriOdic
#   PERIOD: 100 ms
#   MSG SIZE: 64 bytes
#   LATENCY: 100 ms
#   RELIABILITY: 99.9%
#   TECH USED: WireLess
#   MSG VALUE: [x,y,z,R,P,Y], (-1 = Null)
#   ACK REQUIRED: True
# -----------------------------------------------------------------------------------------------------------------------------------------------
```

Figura 88. Ejemplo de la descripción de un *CommunicationModuleScript*.

2. DEFAULT FUNCTIONS

En esta zona, se encuentran las funciones que ofrece por defecto un objeto de tipo vcScript en Visual Components:

- *OnStart():* tras pulsar el botón para iniciar la simulación, se ejecuta esta función y a posteriori se inicia la simulación. Esta función se emplea para actualizar variables u objetos antes de cada simulación. De este modo, si se realizan cambios en algún componente, la próxima vez que se inicie la simulación se ejecutará en primer lugar el *OnStart*, actualizando las variables u objetos antes de que se inicie la simulación. En el caso de este script, la actualización de las variables u objetos se hace llamando a unas funciones definidas por el usuario que describiremos más adelante (ver Figura 89).

```
################################################################
## DEFAULT FUNCTIONS ###########################################
################################################################

def OnStart():
    #Call to Custom Functions
    if comp.Name != 'Communication Controller v0.6':
        getPosibleDestinationMsgs()
        getTriggers()

    try:
        if comModuleScript not in comTriggerSignal.Connections:
            comTriggerSignal.Connections = comTriggerSignal.Connections + [comModuleScript]
    except:
        pass
```

Figura 89. Función *OnStart*() del *CommunicationModuleScript*.

- *OnRun():* esta función se emplea para enviar mensajes periódicos dado que es la única que permite hacer un *delay()* en el tiempo. El resto de funciones están pensadas para que se ejecuten inmediatamente de forma que aunque tengan un *delay()* este será ignorado (ver Figura 90).

```
def OnRun():
    if True:  ##Change False for True if you need to implement Periodic Msgs!!!##
        #Declare multiple periods and its counters(tmp)
        tmp1 = 0
        tmp2 = 0
        period1 = periodicMsg1['period']
        period2 = periodicMsg2['period']

        while True:
            ## UNCOMENT WHICH OPTION YOU NEED

            #Multiperiod Permanent Streaming (period must be in ms)
            '''
            if tmp1 == period1:
                sendMsg(periodicMsg1)
                tmp1 = 0

            if tmp2 == period2:
                sendMsg(periodicMsg2)
                tmp2 = 0

            tmp1 += 1
            tmp2 += 1
            delay(0.001) #ms
            '''

            #Monoperiod Permanent Streaming
            '''
            sendMsg(periodicMsg)
            delayTime = float(periodicMsg['period'])/1000
            delay(delayTime)
            '''

            #Conditional Streaming
            '''
            triggerCondition(lambda: startPeriodicMsgDetected())
            while comTriggerSignal != 'StopPeriodicMsg':
                sendMsg(periodicMsg)
                delayTime = float(periodicMsg['period'])/1000
                delay(delayTime)
            '''
    else:
        pass
```

Figura 90. Función *OnRun*() del *CommunicationModuleScript*.

- *OnSignal():* esta función se ejecuta cada vez que un comportamiento conectado al *CommunicationModuleScript* varíe. De esta manera el script es capaz de detectar cuando varía *ComTriggerSignal* y enviar el mensaje correspondiente a la condición de disparo a través de la señal/comportamiento *CommunicationModule*. Además, esta función detecta cuando se envía un mensaje cuyo destino es el componente actual y si dicho mensaje requiere *ACK,* en ese caso envía el *ACK* correspondiente (ver Figura 91).

```
def OnSignal(signal):

    #Case send a communication msg
    if signal.Name != 'CommunicationModule':
        if signal.Value in conditionsDic.keys():
            messageType = conditionsDic[signal.Value]
            sendMsg(messageType)

    #Case posible ACK required
    else:
        signalList = signal.Value.split(';')
        ackRequerido = signalList[10]
        destinatario = signalList[1]

        if ackRequerido == 'True' and destinatario == comp.Name:
            ackMsg['receiver'] = signal.Component.Name
            sendMsg(ackMsg)
```

Figura 91. Función *OnSignal*() del *CommunicationModuleScript*.

3. CUSTOM FUNCTIONS

   En este apartado se describen una serie de funciones desarrolladas en este proyecto, las cuales facilitan la implementación de las comunicaciones (ver Figura 92):

```
#####################################################################
## CUSTOM FUNCTIONS #################################################
#####################################################################

def sendMsg(msgArg):

def startPeriodicMsgDetected():

def infoSignalDetected(signal): #Used to update values of msgTypes during Simulation

def getPosibleDestinationMsgs():

def connectComModuleToReceiverScript(destComp):

def getTriggers():
```

Figura 92. Funciones de usuario desarrolladas para el *CommunicationModuleScript*.

- *sendMsg():* esta función se encarga de enviar un mensaje a través del *CommunicationModule* con el formato indicado tomando como argumento un diccionario que contiene el valor de cada uno de los campos del mensaje.
- *startPeriodicMsgDetected():* esta función comprueba el valor de *ComTriggerSignal* cada vez que sufre un cambio. Esto permite que en la función *OnRun()* se envíe un mensaje periódico si se recibe una *ComTriggerSignal* con valor *'StartPeriodicMsg'*.
- *infoSignalDetected():* se ejecuta cuando la señal / comportamiento *ComInfoSignal* sufre un cambio. Se puede usar para actualizar valores de los

mensajes. Por ejemplo, se usa para actualizar el campo del mensaje '*receiver*', al nombre del AGV al cual se le envían los comandos de transporte.

- *getPosibleDestinationMsgs():* obtiene una lista de componentes, los cuales son candidatos a ser receptores de mensajes del componente actual. Esto es útil para parametrizar emisor y receptor en los mensajes o para conocer los posibles componentes que recibirán mensajes del componente actual.
- *connectComModuleToReceiverScript():* esta función coge como argumento la lista de posibles componentes que reciben mensajes de este componente y conecta el *CommunicationModule* de este componente al *CommunicationModuleScript* de los posibles receptores. Esto se emplea para que los componentes sean capaces de devolver un *ACK* cuando reciben un mensaje del cual son el receptor y requiere *ACK*.
- *getTriggers():* función donde declarar los disparadores deseados. Permite una función determinada cuando sucede algún evento. Por ejemplo, cuando *ComInfoSignal* sufre un evento (cambia su valor), se ejecuta la función *infoSignalDetected().*

4. VARIABLE / OBJECT DECLARATION

   En esta zona se declaran los comportamientos que se emplean en el script. Además, sería la zona indicada para declarar variables globales en caso de ser necesario (ver Figura 93).

```
########################################################################
## VARIABLE / OBJECT DECLARATION ######################################
########################################################################

# GLOBAL VARIABLES


# BEHAVIOURS
comModule = comp.findBehaviour('CommunicationModule')
comModuleScript = comp.findBehaviour('CommunicationModuleScript')

comTriggerSignal = comp.findBehaviour('ComTriggerSignal')
comInfoSignal = comp.findBehaviour('ComInfoSignal')
```

Figura 93. Zona de declaración de objetos necesarios del *CommunicationModuleScript*.

5. FORMAT & ORDER OF MESSAGE FIELDS

   En este bloque se describe una lista, la cual define el formato de mensaje que se envía. Contiene los nombres, en el orden específico, que definen el formato de mensaje. Esto se usa dado que los mensajes están definidos por diccionarios y no se puede acceder a los campos de estos de forma ordenada, en cambio en una lista sí (ver Figura 94).

```
######################################################################
## FORMAT & ORDER OF MESSAGE FIELDS ##################################
######################################################################

# List of Keys
msgKeys = [
    'emitter',
    'receiver',
    'msgType',
    'comType',
    'period',
    'dataSize',
    'latency',
    'reliability',
    'comTech',
    'msgValue',
    'ackRequired']
```

Figura 94. Declaración de los atributos del mensaje contenido en *CommunicationModule*.

6. MESSAGE TYPES

   En esta zona se define cada tipo de mensaje que envía el componente mediante un diccionario. Al principio se encuentra una plantilla para añadir tantos diccionarios como tipos de mensajes envíe un componente. De forma predeterminada cada componente tiene tres diccionarios, uno para mensajes Default, uno para ACKs y uno para mensajes Periódicos (ver Figura 95).

```
######################################################################
## MESSAGE TYPES #####################################################
######################################################################

#Each Message Type must have its own Dictionary

'''
TEMPLATE FOR NEW DICTIONARIES
dicTemplate = {
    'emitter'     : comp.Name,
    'receiver'    : controller.Name,
    'msgType'     : 'PalletRequest',
    'comType'     : 'a',     # 'p' = periodic  /  'a' = aperiodic
    'period'      : -1,      #ms
    'dataSize'    : 64,      #bytes
    'latency'     : 100,     #ms
    'reliability' : 99.9,    #%
    'comTech'     : 'wl',    # 'wl' = WireLess  /  'wd' = Wired
    'msgValue'    : -1,      #-1 = None
    'ackRequired' : True }   #variable ack
'''

defaultMsg = {

ackMsg = {

periodicMsg = {
```

Figura 95. Zona y formato de declaración de tipos de mensaje del *CommunicationModuleScript*.

7. CONDITIONS DICTIONARY

   Aquí se define un diccionario, que recoge los pares de 'condición de disparo' : 'tipo de mensaje a enviar'. Es decir, si la condición para que se envíe el mensaje 'msg1', es recibir por *ComTriggerSignal* un valor 'cond1', entonces el diccionario quedaría así:

   conditionsDic = {'cond1': msg1}

   Para cada tipo de mensaje que necesite una condición para ser enviado, se ha de colocar en el diccionario la condición que se ha de recibir, y el diccionario que define el mensaje a enviar (ver Figura 96).

```
################################################################
## CONDITIONS DICTIONARY #######################################
################################################################

## Put as key the value of the ComTriggerSignal needed to send a certain msgType
## Put as value the Dictionary above that describes this msgType

conditionsDic = {
    'Default': defaultMsg,
    'Periodic': periodicMsg }
```

Figura 96. Declaración de relación entre condiciones del entorno y tipo de mensaje a enviar.

10.3.3. ComTriggerSignal

Este comportamiento de tipo *vcStringSignal* notifica mediante su contenido las condiciones que se dan en el escenario y mediante las cuales el *CommunicationModuleScript* decide que mensaje ha de enviar. Su valor está constantemente cambiando, dependiendo de la situación en la que se encuentre el escenario. Cuando sucede una determinada situación en el escenario, se notifica a través de *ComTriggerSignal* la condición correspondiente y el *CommunicationModuleScript* advierte que *ComTriggerSignal* ha modificado su valor, entonces en función del nuevo valor de *ComTriggerSignal* envía el mensaje correspondiente a través del *CommunicationModule.* Su función es la de notificar al *CommunicationModuleScript* cuando y que mensaje ha de enviar, por lo tanto, siempre que queramos enviar un mensaje a través de *CommunicationModule,* habrá que notificar al *CommunicationModuleScript* mediante un determinado valor en *ComTriggerSignal.*

10.3.4. ComInfoSignal

ComInfoSignal es un comportamiento de tipo *vcStringSignal* cuyo fin es contener valores necesarios para actualizar atributos del mensaje durante la simulación. Por ejemplo, cuando controlador de AGVs indica que AGV va a realizar un determinado transporte, el nombre del AGV correspondiente se pasa a través de *ComInfoSignal.* El *CommunicationModuleScript* recibe el cambio de valor en *ComInfoSignal* y actualiza el campo de receptor del mensaje al contenido de *ComInfoSignal*.

10.3.5. ComConditionScript

En ocasiones no es trivial identificar el instante en que se requiere enviar una comunicación y es necesario acceder a propiedades, métodos o eventos de un objeto que no se pueden acceder desde la *GUI.* Para estas situaciones se ha creado el *ComConditionScript* un comportamiento de

tipo *vcScript* desde el que poder acceder a las propiedades, o eventos específicos de un objeto que determinan el instante en el que se ha de emular una comunicación. Una vez identifico el instante en que se ha de emular una comunicación la idea es que el *ComConditionScript* notifique a través de *ComTriggerSignal* dicha condición, haciendo así que *CommunicationModuleScript* envíe el mensaje correspondiente a dicha condición a través de *CommunicationModule.*

Este comportamiento no tiene un código de base, y no siempre se utiliza, pero se ha creado para tener localizadas las condiciones bajo las que se ha de enviar un mensaje (cuando para ello es necesario acceder a propiedades y eventos de un objeto) en un mismo sitio.

11. Creación de los datasets

Tras modelar los escenarios industriales de interés y desarrollar los comportamientos de *Visual Components* necesarios para modelar la generación e intercambio de datos entre elementos del escenario (datos que serán transmitidos entre los nodos origen y destino utilizando una tecnología de comunicaciones inalámbrica), el objetivo final es crear *datasets* o bancos de prueba digitales con la información de los datos generados en el escenario industrial y otras datos de eventos e información sobre el estado y distribución de los distintos elementos en el escenario. La disponibilidad de estos *datasets* es clave para conocer y caracterizar la necesidad de conectividad y transferencia de datos dentro de las plantas con el fin de diseñar y aplicar técnicas de inteligencia artificial y *machine learning* para la gestión autónoma y flexible de los futuros sistemas de comunicaciones 5G industriales. Estas técnicas de inteligencia artificial y *machine learning* necesitan grandes cantidades de datos sobre el comportamiento real del escenario para poder ser entrenados y posteriormente validados. Es importante destacar la ausencia de *datasets* similares en la comunidad dado el carácter confidencial del diseño y configuración de sistemas de producción en plantas industriales reales. Por tanto, la generación de estos *datasets* es un hito muy relevante en este campo, siendo clave para el entrenamiento y validación de las soluciones basadas en inteligencia artificial.

Para registrar la información de interés en los *datasets*, se ha creado un nuevo componente denominado '*Data Collector*'. Con este nuevo componente, es posible leer y capturar datos que describen el funcionamiento del escenario y el comportamiento de los distintos elementos en el mismo. Los datos del escenario se guardan en archivos del tipo .csv. En los siguientes apartados, se describen los *datasets* creados y la información registrada en cada uno de ellos. A continuación, se describe de forma detallada las funciones que realiza el *Data Collector* y los distintos comportamientos que se le han implementado.

11.1. Información registrada en los *datasets*

Para cada escenario se realiza el registro de información que describe el comportamiento y funcionamiento de la planta o proceso industrial representado, y la generación e intercambio de datos entre elementos del escenario. En concreto, se registra información sobre la posición de los distintos componentes que integran el escenario en cada momento, de los distintos estados en los que se encuentra cada componente (por ejemplo, activo o inactivo, fallo, sin batería, etc.), e información sobre los datos generados en cada componente y que son transmitidos de manera inalámbrica. Estos datos se guardan en tres ficheros diferentes del tipo .csv dentro de la carpeta '*Datasets*' que se situará en la ruta indicada por el usuario (esto se explica en los siguientes subapartados). A continuación, se describen los distintos ficheros .csv generados y la información registrada en cada uno de ellos.

11.1.1. Posición de los componentes

La información sobre la posición de los distintos elementos o componentes (utilizando la terminología de *Visual Components*) que forman el escenario a lo largo de la simulación se guarda en el fichero *data_position*.csv. La flexibilidad y la capacidad de reconfiguración son requisitos clave de las plantas y líneas de producción en el marco de la Industria 4.0. Además, la integración de robots y maquinaria móvil, así como la interacción con operarios que se desplazan por la fábrica también será algo intrínseco de la Industria 4.0. Por tanto, es importante conocer la posición actualizada de los distintos componentes que integran el escenario. La

información sobre la posición se registrará al comienzo de la simulación para todos los componentes. Cuando un componente cambie de posición (componentes móviles) durante la simulación, se realizará el registro de su posición actualizada. El registro de la nueva posición se realizará cuando la distancia entre la posición actual y la anterior sea mayor a una distancia umbral establecida (este parámetro es configurable y se puede especificar en el script RegistroPosiciones.py en el '*Data Collector*' y que se explica en el apartado 11.2).

La información en el archivo .csv se guarda de la siguiente manera. La información relativa a cada componente en un instante de tiempo determinado (cada entrada en el fichero) se escribe en líneas diferentes. Para cada entrada, se guardan distintos datos que se presentan a continuación. La separación entre datos se realiza mediante ",". En la primera fila del fichero se indica el nombre de cada campo registrado. En cada entrada del fichero se guarda la siguiente información:

- Índice del registro.
- *Component_Name*: identifica de manera única el componente en el escenario.
- *Timestamp*: Indica los segundos transcurridos desde el inicio de la simulación.
- *Position*_X Posición en el eje X del componente. Se indica en milímetros utilizando 3 decimales. El origen del sistema de coordenadas se sitúa en el centro del escenario.
- *Position*_Y: Posición en el eje Y del componente. Se indica en milímetros utilizando 3 decimales.
- *Position*_Z: Posición en el eje Z del componente. Se indica en milímetros utilizando 3 decimales.
- *Orientation*_X**:** Orientación en el eje X del componente. Se indica en grados utilizando 3 decimales. La orientación 0.0 significa que el objeto se sitúa orientado en la misma posición que la posición global.
- *Orientation*_Y**:** Orientación en el eje Y del componente. Se indica en grados utilizando 3 decimales.
- *Orientation*_Z**:** Orientación en el eje Z del componente. Se indica en grados utilizando 3 decimales.

En la Figura 97 se muestra un extracto del fichero '*data_position*.csv' obtenido para el escenario 2. Por ejemplo, en la línea 375 podemos leer que el componente *Mobile Robot Resource* en el instante de tiempo 201.0 segundos se sitúa en la posición (-20225.922, 2978.510, 0.0) mm y una orientación (0.0, 0.0, 90.0) grados. En la línea 377 se puede leer que este mismo componente en el instante 201.5 segundos se ha desplazado hasta la posición (-20225.922, 2771.660, 0.0) mm manteniendo la orientación (0.0, 0.0, 90.0) grados.

```
374,Human (Otto),201.0,-21225.922,3000.000,0.000,-0.000,-0.000,-76.591
375,Mobile Robot Resource,201.0,-20225.922,2978.510,0.000,0.000,-0.000,90.000
376,Human (Otto),201.5,-21369.528,2816.253,1.144,0.000,-0.000,-128.009
377,Mobile Robot Resource,201.5,-20225.922,2771.660,0.000,0.000,-0.000,90.000
378,Human (Otto),202.0,-21753.774,2324.601,4.205,0.000,-0.000,-128.009
379,Human (Otto),202.5,-22108.463,1870.768,7.031,0.000,-0.000,-128.009
380,Mobile Robot Resource,202.5,-20225.922,2564.810,0.000,0.000,-0.000,90.000
```

Figura 97. Ejemplo del registro de información en el fichero *data_position*.csv.

11.1.2. Generación de datos en el escenario

Este fichero guarda información sobre los datos generados en cada componente y que se transmitirán de forma inalámbrica. En particular, es de interés conocer la fuente y destino de la

información, el instante temporal en el que se generan los datos y la cantidad de datos a transmitir. Si estuviera disponible esta información, también se registrará los requisitos de latencia (es decir, el tiempo máximo en el que los datos deben ser entregados en el destino) y de fiabilidad.

Con esta información, y relacionándola con la información de posición y distribución de los distintos componentes en el escenario, se obtiene información sobre la generación espacio-temporal de datos en el escenario. Esta información se registrará en el fichero 'data_communications.csv'. La información registrada es la siguiente:

- *Iteration*: Indica el número del registro que se realiza. Cada registro representa un proceso de comunicación.
- *Timestamp*: Tiempo transcurrido desde el inicio de la simulación en segundos.
- *Source*: Nombre del componente que genera y transmite los datos.
- *Destination*: Nombre del componente destino al que se dirigen los datos.
- *Message* Id: Tal y como se presentó en el apartado 9, es el identificador empleado para cada tipo de mensaje.
- *Communication Type*: Identifica se la comunicación es periódica (representado por 'p') o aperiódica (representado por 'a').
- *Period*: En caso de ser un mensaje periódico, indica la periodicidad en milisegundos. Si el mensaje es aperiódico, se indicará el valor '-1'.
- *Data Size*: Tamaño de los datos generados y que se enviarán en el mensaje.
- *Latency*: Tiempo máximo en el que los datos deben entregarse en el destino. Si la información de latencia requerida no está disponible, se registra el valor -1.
- *Reliability*: Indica la fiabilidad requerida en la transmisión de estos datos. En concreto se calcula como el porcentaje de mensajes entregados en el destino cumpliendo con la latencia requerida entre el total de mensajes enviados. Si la información sobre nivel de fiabilidad requerido no está disponible, se registra el valor -1.
- *Communication Tecnology*: Indica si la comunicación es inalámbrica (representado por 'wl') o cableada (representado por 'l').
- *Message Value*: Campo sin valor que ha sido dejado por si en un futuro se quisiera transmitir otra información, por el momento se mantendrá el valor -1.
- *AckRequired*: Indica si el mensaje transmitido requiere confirmación o no.

La Figura 98 muestra, a modo de ejemplo, un extracto del fichero '*data_communications*.csv'. En este ejemplo se puede observar el registro de los distintos atributos explicados anteriormente.

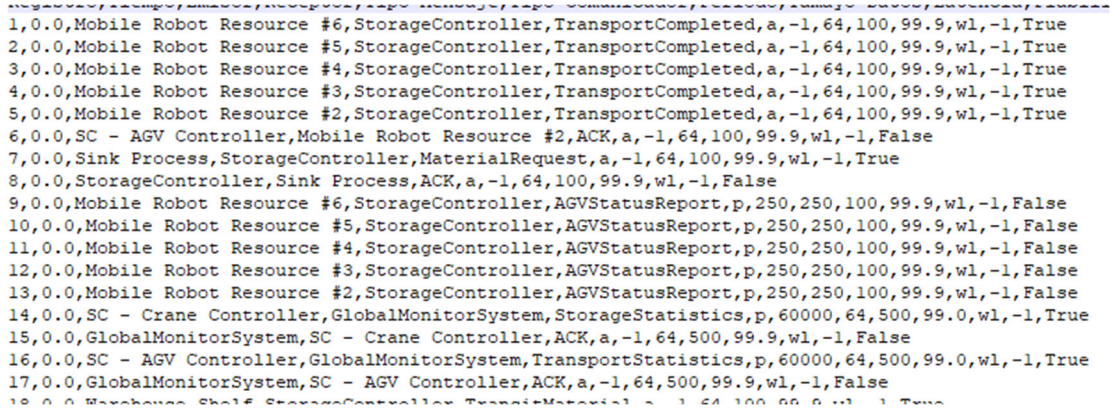


Figura 98. Ejemplo del registro de información en el fichero *data_communications*.csv.

11.1.3. Estado de los componentes

A lo largo de la simulación los componentes pueden cambiar de estado, por ejemplo, pueden estar 'Ejecutando proceso', en 'Espera', 'Roto', 'Reparándose', etc. Cada componente tendrá definida la lista de estados por los que puede pasar. Pasar de un estado a otro puede desencadenar una serie de acciones que impliquen un mayor intercambio de información y datos entre componentes y, por tanto, que cambie el patrón de generación de datos en el escenario. Por este motivo, resulta también interesante el registro de la información sobre el estado en el que se encuentra cada componente. El registro del estado de cada componente se realiza en el fichero data_states.csv. Al inicio de la simulación se realiza el registro del estado de cada componente y se realizará una nueva entrada cada vez que un componente cambio de estado.

Para pasar por estos estados el componente debe de tener en sus comportamientos el objeto '*Statistics*', que es el que recoge todos los estados que este tiene. En la Figura 99 se muestra el comportamiento '*Statistics*' dentro del componente '*Generic Hydraulic Press*' utilizado como ejemplo.

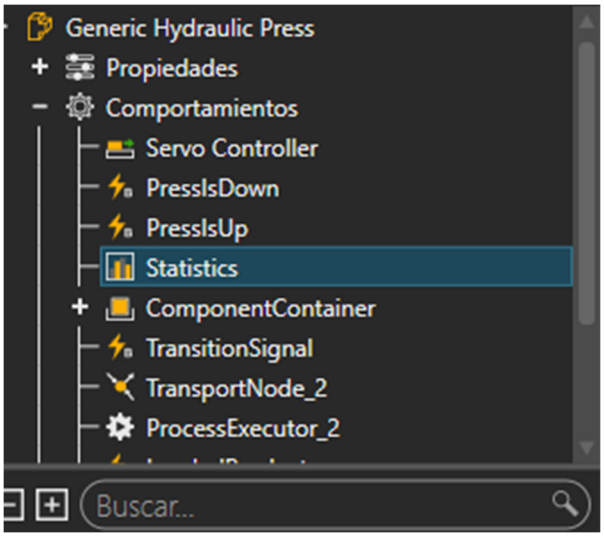


Figura 99. Vista del comportamiento '*Statistics*'

En la Figura 100 vemos como una vez que tenemos seleccionado el objeto '*Statistics*' nos aparecerá en la ventada de la derecha, abajo del todo, los estados que tiene implementado por defecto y que hayamos creado nosotros.

Figura 100. Vista de estados en el comportamiento '*Statistics'*.

No todos los componentes en *Visual Components* tienen implementado el comportamiento '*Statistics'*. Además, los que sí implementan el comportamiento '*Statistics'* pueden tener distintos estados. Para conocer los distintos estados por los que puede pasar un componente, debemos recorrer uno a uno cada componente accediendo al menú correspondiente. Para hacer este proceso más ágil, se ha implementado un script, denominado '*ComponentStatesInfo'* que recorre uno a uno cada componente del escenario y lee los estados por los que puede pasar a lo largo de la simulación. Esta información la guarda en el fichero 'component_info_states.csv', en el que cada entrada corresponde a los estados por los que puede pasar cada componente. La Figura 101 muestra un ejemplo de este fichero para el escenario 2. En la figura podemos observar que el componente '*Mobile Robot Resource*' puede pasar por los estados '*Processing*, *Repairing*, *Transporting*, *Picking*, *Placing*, *Moving*, *CollectingTool*, *ReturningTool*, *Idle*, *Blocked*, *Bypassing*, *Charging*, Break', mientras que el componente '*Feeder Process*' solamente puede pasar por los estados '*Warmup*, *Break*, *Idle*, *Busy*, *Blocked*, *Broken*, *Repair*, *Setup*'.

```
1  Components:,States
2  Weld Button Node,Warmup,Break,Idle,Busy,Blocked,Broken,Repair,Setup
3  Feeder Process,Warmup,Break,Idle,Busy,Blocked,Broken,Repair,Setup
4  Generic Hydraulic Press,Warmup,Break,Idle,Busy,Blocked,Broken,Repair,
5  Human (Otto),Processing,Repairing,Transporting,Picking,Placing,Movin
6  Mobile Robot Resource,Processing,Repairing,Transporting,Picking,Plac
7  Welding Node,Warmup,Break,Idle,Busy,Blocked,Broken,Repair,Setup
8  RobotController,Idle,Blocked,BusyHandling,BusyWorking,ChangingTool,M
9  Generic Articulated Robot,Warmup,Break,Idle,Busy,Blocked,Broken,Repa
10 Human (Anna),Processing,Repairing,Transporting,Picking,Placing,Movin
11 Human (Anna) #3,Processing,Repairing,Transporting,Picking,Placing,Mo
12 Warehouse Shelf,Warmup,Break,Idle,Busy,Blocked,Broken,Repair,Setup
13 Press Button Node,Warmup,Break,Idle,Busy,Blocked,Broken,Repair,Setup
14 PC Node,Warmup,Break,Idle,Busy,Blocked,Broken,Repair,Setup
15 Human (Anna) #2,Processing,Repairing,Transporting,Picking,Placing,Mo
16 M3 Gage CMM
17
```

Figura 101. Ejemplo del registro de información en el fichero component_info_states.csv.

Los estados por los que pasa cada componente durante la simulación se registran en el fichero '*data_states*.csv'. En concreto, se registra la siguiente información:

- *Timestamp*: tiempo de la simulación en el que se registra la posición. El tiempo se indica en milisegundos, siendo el tiempo 0.0 el inicio de la simulación.
- *Component*: Nombre del componente que sufre un cambio de estado.
- *State*: Estado en el que se encuentra el componente.

Finalmente, es importante decir que, aunque en el marco de este proyecto se han desarrollado las herramientas (nuevos comportamientos y scripts de Python) para registrar los estados en los que se encuentra cada componente a lo largo de la simulación del proceso de producción, en los escenarios implementados no se simulan los cambios de estado de los componentes, es decir, cada componente se mantiene en el mismo estado durante toda la simulación. Por este motivo, no se ha podido obtener esta información en los escenarios simulados, aunque las herramientas están preparadas para poder hacerlo. Para simular estos cambios de estados es necesario estudiar las características de los objetos reales que representa cada componente y, en base a estas características, establecer y definir los momentos y en base a qué circunstancias se producen los cambios de estado para cada componente.

11.2.*Data Collector*

Para realizar el registro de la información y crear los *datasets*, se ha creado un componente nuevo denominado '*Data Collector*'. Es importante destacar que este componente es un componente ficticio que no forma parte del escenario real y cuya única función es la de realizar la lectura de los datos de interés y su posterior registro en uno o varios ficheros. El nuevo componente parte de una figura geométrica, particularmente un cubo, al que se le han definido una serie de propiedades y comportamientos para realizar la lectura y registro de los datos. En la Figura 102 muestra una vista del escenario 2 en la que se puede observar el componente '*Data Collector*'.

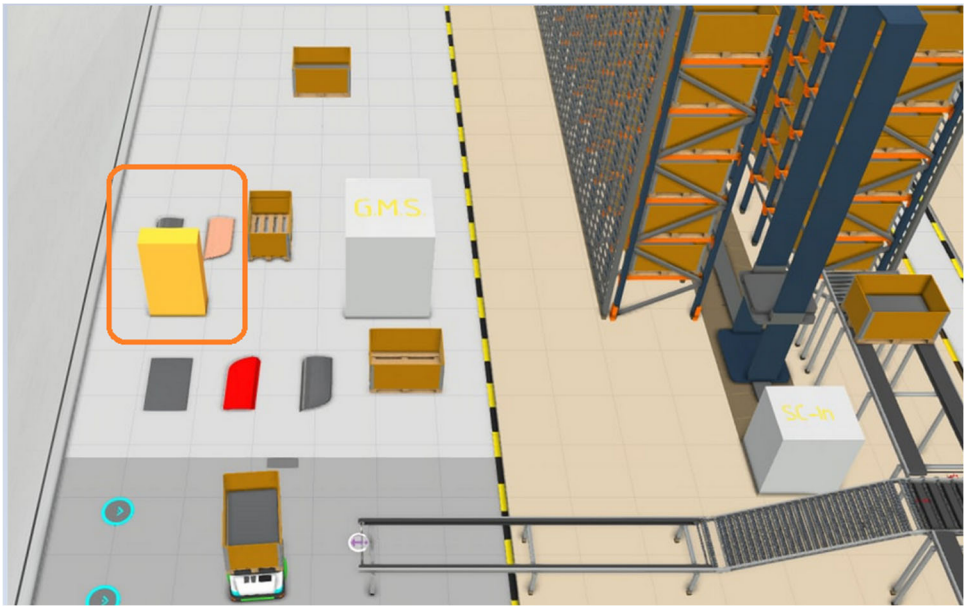


Figura 102. Vista del escenario 2 en la que se puede observar el nuevo componente '*Data Collector*'.

Las propiedades y/o comportamientos definidos para el componente '*Data Collector*' son:

- *PositionsRecording*: Comportamiento de tipo script. Monitoriza la posición de cada componente durante la simulación y guarda dicha información en el fichero 'data_position.csv'.
- *Get & Connect CommunicationModules*: Comportamiento de tipo script, que conecta los componentes que incluyen el comportamiento '*Communication Module*' al script '*DataToTransmitRecording*'. Esto es necesario para que el script '*DataToTransmitRecording*' pueda acceder al '*Communication Module*' de cada componente y capturar la información sobre los datos generados para su posterior registro.
- *DataToTransmitRecording*: Comportamiento de tipo script. Monitoriza los datos que se generan en cada componente y que se transmitirán de forma inalámbrica. Además, realiza el registro de estos datos en el archivo 'data_communications.csv'.
- *GetStates*: Comportamiento de tipo script. Registra los distintos estados que tiene implementados un componente en su comportamiento '*Statistics'*. Con este objeto se pueden asignar distintos estados por los que pueda pasar una máquina, por ejemplo, de estar en 'esperar' a pasar a estar 'ejecutandose'. Este script recoge todos los estados por los que podría pasar ese componente.
- *StatesRecording*: Comportamiento de tipo script, que registra los distintos cambios de estados que se producen en cada componente. Indicaría por ejemplo en un cierto instante de tiempo cuando por ejemplo un componente pasa del estado 'roto', al estado 'reparandose', para finalmente volver al estado 'ejecutándose' o 'esperando'. También indicaría cuando por ejemplo un robot esta con la batería baja o cualquier otro evento. A continuación, se presentan y describen con detalle los scripts implementados para definir cada comportamiento.

11.2.1. PositionsRecording

El script '*PositionsRecording'* se encarga de registrar la posición y la orientación de cada componente respecto al centro del escenario cada vez que la alguno de estos valores cambie. Los datos se registran en el fichero 'data_positions.csv'. A continuación, se describe el funcionamiento del script.

- Especificar la ruta en la que se guardará el fichero.

En la Figura 103 realizamos la especificación del nombre de la carpeta donde queremos guardar el archivo que creemos (línea 17), obtenemos el nombre del usuario que tenga abierta la sesión (línea 19) y con ello obtenemos la ruta donde estará la carpeta en la que guardaremos el archivo (línea 21), por defecto hemos puesto que sea en el escritorio principal.

```
16  #Nombre de la carpeta donde almacenaremos el csv con los datos
17  nombre_carpeta = "Datasets"
18  #Obtenemos nombre de usuario
19  username = os.getenv("USERPROFILE")
20  #obtenemos la ruta donde crearemos la carmpeta
21  path = os.path.join("C:\\Users\\", username, "Desktop")
```

Figura 103. Especificación de ruta en script *PositionsRecording*.

- Variables

Las variables que hemos usado para este script se pueden observar en la Figura 104. Primero hemos declarado las constantes correspondientes a: el intervalo de iteraciones 'INTERVALO_ITERACIONES' (línea 25), que se encarga de comprobar que la lista no recoge en exceso cantidades grande de información, sin registrarlo en el fichero; y al intervalo de distancia mínima llamado 'INTERVALO_DISTANCIA' (línea 27). Hemos usado en este script dos contadores, uno para asignar el orden del registro (línea 31) y otro para controlar la cantidad de registros que llevamos recogidos (línea 33) en la lista que guarda las posiciones (línea 43). En la línea 37 recogemos el objeto del entorno de la aplicación, en la línea 38 el objeto referente a la simulación y en la línea 41 creamos una variable diccionario que usaremos para ir comparando la posición actual del componente con respecto a su posición anterior registrada.

```
23  #CONSTANTES
24  #Indica el intervalo de cantidad de
25  INTERVALO_ITERACIONES = 100
26  #Indica el intervalo de la distanci
27  INTERVALO_DISTACIA = 1
28
29  #CONTADORES
30  #Contador de todos los registros de
31  contador_registro = 0
32  #Contador que controla en todo mome
33  contador_intervalo = 0
34
35
36  #Declaramos las variables que repre
37  app = getApplication()
38  sim = app.getSimulation()
39
40  #diccionario que recoje la última p
41  diccionario_ultimo_registro = {}
42  #Lista que recoje los datos que esc
43  lista_posiciones = []
44
```

Figura 104. Variables usadas en script *PositionsRecording*.

- Función *exportData*

En la Figura 105 se muestra la implementación de la función '*exportdata*' que primero abrirá el fichero de modo que escribirá a continuación de lo que ya había en el fichero con anterioridad (línea 48), en la línea 50 indicamos el delimitador de la información que escribiremos, es decir, esta estará separada por lo que le indiquemos. Posteriormente, escribimos todos los elementos de la lista que ha recogido las posiciones y orientaciones de los componentes (línea 52 y53). Por último, cerramos el fichero (línea 55) y vaciamos la lista de posiciones (línea 57).

```
45   #Esta funcion exporta a csv los datos recogidos en la variable "lista
46   def exportData():
47       #abrimos fichero
48       f = open(path + "\\"+ nombre_carpeta + "\\data_position.csv", "a")
49       #establecemos el delimitador del csv
50       state_writer = csv.writer(f, delimiter=",", lineterminator='\n')
51       #Escribimos en el csv todas las posiciones recogidas
52       for fila in lista_posiciones:
53           state_writer.writerow(fila)
54
55       f.close()
56       #vaciamos el contenido de la lista
57       del lista_posiciones[:]
```

Figura 105. Función '*exportData'* encargada de registrar la información del script *PositionsRecording*.

- Función *exportDataUnique*

En la Figura 106 mostramos la función '*exportDataUnique'* que usaremos para recoger los datos que almacenaremos posteriormente en el fichero y llamar al final de su ejecución a la función que se encarga de exportar los datos al fichero. En la variable 'columnas' (línea 62) recogemos las columnas que guardaremos en nuestro fichero. Posteriormente intentaremos crear la carpeta '*Dataset*' (línea 65) que en caso de que no exista, se avisará de que ya existe (línea 67). Necesitaremos usar la lista de componentes del escenario, así que lo almacenamos en una lista (línea 71). Antes de empezar a registrar la información abriremos el fichero (línea 74) para guardar en la primera fila el nombre de las columnas (línea 81) y lo volveremos a cerrar (línea 82).

```
59   #De forma continua vamos recogiendo las distintas posiciones de los comp
60   def exportDataUnique():
61       #variables
62       columnas = ['Iteration','Component_Name' ,'Timestamp', 'Position_X', '
63       global contador_registro, contador_intervalo, lista_posiciones, diccio
64       #creamos directorio
65       try:
66           os.mkdir(path + "\\"+ nombre_carpeta)
67       except:
68           print("Ya existe el directorio")
69
70       #Obtenemos una lista con los objetos componentes
71       listComponents = app.Components
72
73       #abrimos fichero
74       f = open(path + "\\"+ nombre_carpeta + "\\datos_posiciones.csv", "w")
75
76       #establecemos el delimitador del csv
77       state_writer = csv.writer(f, delimiter=",", lineterminator='\n')
78
79       #escribimos primera fila donde indicamos el nombre de cada columna
80
81       state_writer.writerow(columnas)
82       f.close()
83
```

Figura 106. Parte de la función '*exportDataUnique'* en el script *PositonsRecording*.

A continuación, en la Figura 107 mostramos la implementación de un bucle infinito (línea 85) que se va a encargar de recorrer toda la lista de componentes (línea 86), para comprobar si su posición u orientación ha variado. Para ello, lo primero que tenemos que hacer es obtener los datos de las posiciones y orientaciones actuales (línea 89-94), luego las agrupamos (líneas 96 y 97) para que a la hora de hacer la comparación con su anterior posición resulte más sencillo.

```
84      #Bucle infinito
85      while True:
86          for componente in listComponents:
87              #if not (componente.BOMdescription == "Human"):  #esta linea es u
88              #accedemos al objeto del componente que indica las posiciones y o
89              pos_x = "{:.3f}".format(componente.WorldPositionMatrix.P.X)
90              pos_y = "{:.3f}".format(componente.WorldPositionMatrix.P.Y)
91              pos_z = "{:.3f}".format(componente.WorldPositionMatrix.P.Z)
92              ori_x = "{:.3f}".format(componente.WorldPositionMatrix.WPR.X)
93              ori_y = "{:.3f}".format(componente.WorldPositionMatrix.WPR.Y)
94              ori_z = "{:.3f}".format(componente.WorldPositionMatrix.WPR.Z)
95              #unimos en un mismo string la informacionpara que luego sea mas f
96              posiciones = str(pos_x) + " " + str(pos_y) + " " + str(pos_z)
97              orientaciones = str(ori_x) + " " + str(ori_y) + " " + str(ori_z)
```

Figura 107. Asignación de variables de posición y orientación en el script *PositionsRecording*.

En la Figura 108 estamos realizando la comparación de la posición actual con la anterior (línea 101), que está guardada en el diccionario 'diccionario_ultimo_registro' donde accedemos mediante el nombre del componente que es la clave y comparamos tanto la posición como la orientación guardada anteriormente en las variables 'orientaciones' y 'posiciones', que representa su ubicación actual. En el caso de que las ubicaciones no sean iguales se guardarán las posiciones y orientaciones últimas (línea 102-105) y se restarán en valor absoluto con las distancias actuales, en el caso de que la distancia absoluta recorrida, sea superior al intervalo marcado (línea 107) registramos la nueva posición (línea 110), sino no.

```
98  #Al comprobar por primera vez el diccionario dará error porque no hay nada aún, en ese caso lo metemos dentro
99  try:
100     #comparamos la posicion actual del componente con la anterior que tendremos guardada
101     if not (diccionario_ultimo_registro[componente.Name][2] == posiciones and diccionario_ultimo_registro[componente.Name][3] == orientaciones):
102         posiciones_ultimas = diccionario_ultimo_registro[componente.Name][2].split(';')
103         pos_x_aux = float(posiciones_ultimas[0])
104         pos_y_aux = float(posiciones_ultimas[1])
105         pos_z_aux = float(posiciones_ultimas[2])
106         #comprobamos el intervalo de las distancias, en caso de que sea menor al intervalo no escribiremos este
107         if (abs(pos_x - pos_x_aux) >= INTERVALO_DISTANCIA or abs(pos_y - pos_y_aux) >= INTERVALO_DISTANCIA or abs(pos_z - pos_z_aux) >= INTERVALO_DISTANCIA ):
108             contador_registro += 1
109             #añadimos a la lista de las posiciones un nuervo registro que posteriormente añadiremos al csv
110             lista_posiciones.append([contador_registro,componente.Name,str(sim.SimTime),str(pos_x),str(pos_y),str(pos_z),str(ori_x),str(ori_y),str(ori_z)])
111             contador_intervalo += 1
```

Figura 108. Comprobación de intervalos de las posiciones que se registran.

En la Figura 109 mostramos como en caso de que al acceder al diccionario no esté la clave que buscamos, esto generará un error que lo capturaremos (línea 112) y que siempre actuará cuando se recoja la ubicación de un componente por primera vez, por lo que, si este fragmento de código se activa, su función será recoger la posición actual de dicho componente (línea 116). Comprobaremos el número de registros que llevamos recogidos

(línea 119) y en caso de que sea mayo al permitido los exportamos al fichero (línea 120). Y por último meteremos un retardo para que la comprobación de la ubicación de los componentes sea cada cierto tiempo intervalo configurable en el script (línea 123), ya que, si no realizaría la comprobación en cada instante lo que podría ralentizar la recogida de registros por el aumento del tiempo de ejecución, dando lugar a pérdidas de información.

```
112     except:
113         diccionario_ultimo_registro[componente.Name] = [componente.Name,str(sim.SimTime),posiciones,orientaciones]
114         contador_registro += 1
115         #añadimos a la lista de las posiciones un nuervo registro que posteriormente añadiremos al csv
116         lista_posiciones.append([contador_registro,componente.Name,str(sim.SimTime),str(pos_x),str(pos_y),str(pos_z
117         contador_intervalo += 1
118     #Cada un número determinado de registros que se hagan se irá guardando de forma automática
119     if contador_intervalo == INTERVALO_ITERACIONES:
120         exportData()
121 #Tiempo de retraso que realiza una vez acabada la iteración del bucle while. Si queremos que el intervalo de ti
122 #Teniendo en cuanta la velocidad de ciertos componentes se aconseja que cuanto mas rapidos sean, menor debe ser
123 delay(0.5)
```

Figura 109. Registro de las posiciones y comprobación de intervalo de registros.

- Evento *OnRun*

En la Figura 110 mostramos el evento *OnRun* en el script '*PositionsRecording'* que se activará mientras el script esté ejecutándose, llamando una única vez a la función '*exportDataUnique'* (línea 127) que es el que se encarga de estar comprobando de forma constante las posiciones y orientaciones de los componentes.

```
125 #Cada vez que la simulación arranque ejecutará su contenido
126 def OnRun():
127     exportDataUnique()
```

Figura 110.Evento *OnRun* en script *PositionsRecording*.

- Evento *OnStop*

En este evento siempre que paremos la simulación llamaremos a la función *exportData* (línea 130) por lo que abrirá el fichero correspondiente y guardará todo lo que tenga almacenado en dicho momento. En la Figura 111 se muestra este evento implementado.

```
128 #cada vez que la simulación se pare ejecutará su contenido
129 def OnStop():
130     exportData()
```

Figura 111. Evento *OnStop* en script *PositionsRecording*.

11.2.2. *Get& Connect CommunicationModules*

A continuación, se explica el script desarrollado *Get& Connect CommunicationModules.* Como se introdujo al inicio del apartado 11.2, para realizar el registro de los datos generados en cada componente, es necesario de manera previa conectar el script que realizará el registro de estos datos con el comportamiento '*ComunicationModule*' de cada componente. Para ello una vez que hayamos listado todos los componentes, comprobaremos si tienen el comportamiento

'*ComunicationModule*'. La Figura 112 muestra el código del script usado para conectar los componentes al script *DataToTransmitRecording*. Primero cogemos la lista de componentes (obtenida en la línea 14), la recorremos (línea 20), comprobamos si tienen o no el comportamiento que actúa de comunicador que se llamaría '*CommunicationModule'* (línea 21), en caso de que el componente lo tenga lo añadiremos a la lista '*comModules'* (línea 22), posteriormente recorremos esta lista y elemento por elemento (línea 29), comprobamos si está conectado en las conexiones, que no es más que una lista propia, (línea 30), en caso de no estar en las conexiones la añadimos (línea 31).

```
12  #GENERICO DE TODO EL LAYOUT
13  app = getApplication()
14  components = app.Components
15
16  comModules = []
17
18  #Crea una lista con todos los behaviours que coinciden
19  #en nombre con algún elemento de la lista SIGNALS
20  for compo in components:   #Para cada componente del layout
21      if compo.findBehaviour("CommunicationModule") != None:
22          comModules.append(compo.findBehaviour("CommunicationModule"))
23
24  #PROPIO DEL COMPONENTE
25  comp = getComponent()
26  this = comp.findBehaviour("DataToTransmitRecording")
27
28  #Conecta todas las señales que queremos registrar a este Script
29  for x in comModules:
30      if this not in x.Connections:
31          x.Connections = x.Connections + [this]
```

Figura 112. Vista de script *Get& Connect CommunicationModules*.

11.2.3. DataToTransmitRecording

En la Figura 113 indicamos los pasos para concretar la ruta donde guardaremos la información referente los datos generados en los distintos componentes del escenario. Primero, vamos a especificar donde queremos guardar el fichero, en nuestro caso le hemos dicho que coja la carpeta del usuario actual y vaya a su escritorio donde creará la carpeta llamada '*Dataset'* (línea 16), obtenemos el nombre del usuario (línea 18) más adelante guarda en una variable la ruta que usaremos para llegar a la carpeta '*Dataset*' (línea 20).

```
14  #CREACION DE CARPETA Y RUTA DE ARCHIVO
15  #Nombre de la carpeta donde almacenaremos el csv con los datos
16  nombre_carpeta = "Datasets"
17  #Obtenemos nombre de usuario con el que hemos iniciado Windows
18  username = os.getenv("USERPROFILE")
19  #obtenemos la ruta donde crearemos la carpeta
20  path = os.path.join("C:\\Users\\", username, "Desktop")
21
```

Figura 113. Especificación de ruta en el script *DataToTransmitRecording*.

En la Figura 114, indicamos el nombre de las variables que usaremos en el script. Primero se indica la constante que usaremos para controlar que cada cierto de número de registros se realice un guardado automático (línea 33) que explicaré más adelante. Luego la lista 'columnas_comunicaciones' (línea 37) indica los nombres de las columnas que guardaremos en

el fichero; la siguiente lista, 'lista_comunicaciones' (línea 39), guardará la información de las comunicaciones, donde cada fila representaría la información recogida de una comunicación.

Hemos usado variables a modo de contador (líneas 43 y 45), donde usaremos la primera, 'contador_registros', para contabilizar los registros que vamos guardando dentro de la variable 'lista_comunicaciones'; por otro lado, la variable 'contador_iteraciones' indicará el número de comunicación que ha llegado, será distinto para cada fila que posteriormente guardemos en el fichero. Por último, la variable 'comprobacion_fichero' la usaremos para identificar si el fichero ya ha sido abierto y si tenemos que escribir la fila correspondiente a los nombres de las columnas del fichero.

```
30  #VARIABLES
31  #Consantes
32  #Indica cada cuantas iteraciones guardaremos de forma automáti
33  REINICIO_ITERACIONES = 100
34
35
36  #Lista con lo campos que habrá en cada fila en el csv
37  columnas_comunicaciones = ['Iteration','Timestamp','Transmitte
38  #Esta la lista la iremos llenando con los distinto registros q
39  lista_comunicaciones = []
40
41  #CONTADORES
42  #Contador usado para el control de registros máximos que almac
43  contador_registros = 0
44  #Contador que indicará el orden de llegada de la comunicación
45  contador_iteraciones = 1
46
47  #Variable usada para controlar si se ha escrito o no la primer
48  comprobacion_fichero = 0
```

Figura 114. Variables que usamos en el script DataToTransmitRecording

En la Figura 115 empezamos a explicar la función principal que se encargará de exportar los datos al dataset llamado '*exportDataComunication*'. Primero intentaremos crear la carpeta, en caso de que ya exista dará un error que será capturado y mostrará el mensaje de que ya existe (líneas 134-137)

```
129  #Exporta a csv los registros guardados en la variables "lista_comunicaciones"
130  def exportDataComunication():
131      global comprobacion_fichero
132
133      #creamos directorio si no existe y creamos el archivo csv
134      try:
135          os.mkdir(path + "\\"+ nombre_carpeta)
136      except:
137          print("Ya existe el directorio")
```

Figura 115. Comienzo de función *exportDataComunication*.

En caso de que el fichero se abra por primera vez (línea 139-141 en la Figura 116), deberemos escribir la primera fila del archivo que contiene el nombre de las distintas columnas (línea 147), y una vez hecho eso iremos escribiendo uno por uno todos los elementos de la variable 'lista_comunicaciones' (línea 150-152), donde cada elemento es un registro de una comunicación. También vaciaremos la lista del contenido que haya recogido, para vaciar el buffer(línea 154), y aumentaremos en una unidad la variable 'comprobacion_fichero' para tener

en cuenta posteriormente que el archivo ya se ha abierto(línea 156), por último cerraremos el fichero (línea 157).

```
138
139 if comprobacion_fichero == 0: #el fichero lo abrimos por primera vez
140     #abrimos fichero
141     f = open(path + "\\"+ nombre_carpeta + "\\data_communications.csv", "w")
142
143     #establecemos el delimitador del csv
144     state_writer = csv.writer(f, delimiter=",", lineterminator='\n')
145
146     #escribimos primera fila donde indicamos el nombre de cada columna
147     state_writer.writerow(columnas_comunicaciones)
148
149     #recorremos la lista que contiene los registros y los vamos guardando uno
150     for fila in lista_comunicaciones:
151         state_writer.writerow(fila)
152
153     #una vez que hemos acabado vaciamos la variable lista
154     lista_comunicaciones[:] = []
155     #como es la primera vez que abrimos el fichero lo indicamos en la variabl
156     comprobacion_fichero += 1
157     f.close()
```

Figura 116. Comprobación de fichero y registro de comunicaciones.

En caso de que anteriormente ya hayamos abierto el fichero (línea 160 en la Figura 117) le indicamos que solo escriba las filas registradas en la variable 'lista_comunicaciones' (líneas 166-167) y posteriormente la vacíe (línea 170). Finalmente cerramos el fichero (línea 172).

```
158 else:   #solo añadimos filas al fichero
159     #abrimos fichero
160     f = open(path + "\\"+ nombre_carpeta + "\\datos_comunicaciones.c
161
162     #establecemos el delimitador del csv
163     state_writer = csv.writer(f, delimiter=",", lineterminator='\n')
164
165     #recorremos la lista que contiene los registros y los vamos guar
166     for fila in lista_comunicaciones:
167         state_writer.writerow(fila)
168
169     #una vez que hemos acabado vaciamos la variable lista
170     lista_comunicaciones[:] = []
171     #cerramos el fichero una vez que hemos acabado
172     f.close()
```

Figura 117. Registro de información en el caso de que se haya abierto anteriormente el fichero.

Respecto a los eventos que usaremos para manejar el script durante la ejecución de la simulación están:

- *OnStop*: Cada vez que paremos la simulación el script abrirá el dataset y registrará todo lo que haya recogido al llamar a la función encargada de ello '*exportDataComunication*' (línea 56). La Figura 118 muestra la implementación del evento.

```
51  #Evento que ejecutará todo lo que tenga dentro
52  #siempre que paremos la simulación
53  def OnStop():
54      #Cuando paramos la simulación se iniciará
55      #la exportación del contenido de la variable "
56      exportDataComunication()
```

Figura 118. Evento *OnStop*.

- *OnStart*: Solo la usaremos para inicializar las variables 'comprobacion_fichero' (línea 64) y 'contador_iteraciones' (línea 65). Aunque al inicio del script las hayamos inicializado pueden dar error si no se hace también de esta forma. La Figura 119 muestra la implementación del evento.

```
58    #Evento que ejecutará lo que tenga dentro cuando se i
59    def OnStart():
60        #Por algún problema en el manejo de memoria cuando
61        #de las varibles pueden ser lo mismos de la anterio
62        #para que no ocurra eso las inicilizamos siempre qu
63        global comprobacion_fichero, contador_iteraciones
64        comprobacion_fichero = 0
65        contador_iteraciones = 1
66
```

Figura 119. Evento *OnStart*.

- *OnSignal*: Es el encargado de manejar la entrada de datos de la señal. Tal y como hemos explicado anteriormente en el script *'Get & ConnectCommunicationModules'*. Este evento siempre se activará cuando en el escenario se envíe una señal conectada a este script. En la Figura 120 mostramos como hemos declarado las variables que usaremos para registra cada filara que represente una comunicación. En la variable *'timestamp'* hemos guardado el tiempo en segundos que transcurre desde el inicio de la simulación (línea 71). Seguidamente accedemos a la información que se envía en la señal (línea 75) y se la asignamos a una lista que separará por elementos toda la información, tomando como valor de separador el carácter de ','. De las líneas 79 a la 89 hemos ido asignando a las correspondientes variables la información separada, en caso de que quisiéramos que llegará en otro orden esto debería ser modificado de forma manual seleccionando el comportamiento de los respectivos componentes en los escenarios.

```
67    #Este evento ejecutará lo que tenga dentro cuando se reciba una
68    def OnSignal(signal):
69        global contador_iteraciones, contador_registros
70        #tiempo de simulación en el que recibe la comunicación
71        timeStamp = sim.SimTime
72
73        #Dividiemos en varios campos la información recibida en la se
74        #en la información de la señal
75        signalList = signal.Value.split(';')
76
77        #Las varibales que se recogen a continuación son las que hay
78        #para modificar lo que se recoge en cada posición habría que
79        emitter = signalList[0]  #componente emisor
80        receiver = signalList[1] #componente receptor
81        msgType = signalList[2]  #tipo de mensaje
82        #comType = signalList[3]  #tipo de comunicación
83        #period = signalList[4] #señal periodica o aperiodica
84        datasize = signalList[5] #tamaño de la información que transm
85        latency = signalList[6] #latencia de la ocmunicacion
86        reliability = signalList[7] #porcentaje de fiabilidad de dich
87        #comTech = signalList[8]  #Indica si la comunicación es inalá
88        #msgValue = signalList[9] #Campo sin valor que ha sido dejado
89        #ackRequired = signalList[10] #Es para saber si el mensaje tr
90        #en caso de que queramos usar este último campos lo trasnform
91
```

Figura 120. Asignación de variables en evento *OnSignal*.

La Figura 121 muestra cómo se simplifica la información que posteriormente registraremos en el fichero, y es que para datos que aportan información binaria fácilmente la podremos representar en el fichero con '1's o '0's en caso de que queramos guardar esta información. En la variable '*ackrequired*' si el valor es 'True' lo guardaremos como '0', en caso de que sea 'False' se guardará como '0' (líneas 94-97). Haremos lo mismo con la variable '*comTech*', en caso de que el tipo de tecnología sea inalámbrica 'wl' lo guardaremos en el fichero con '1', en el caso contrario 'w' la guardamos como ''0' (líneas 100-103). Por último, haremos lo mismo con la variable 'comType', en caso de ser periódica 'p' se guardará como '1', en caso de ser aperiódica 'a', se guardará como '0'.

```
92
93      #Para reducir tamaño del fichero, que e
94      if ackRequired == 'True':
95          ackRequired = 1
96      else:
97          ackRequired = 0
98
99      #Para reducir el tamaño del fichero, qu
100     if comTech == 'wl':
101         comTech = 1
102     else:
103         comTech = 0
104
105     #Para reducir el tamaño del fichero, qu
106     if comType == 'p':
107         comType = 1
108     else:
109         comType = 0
110
111
```

Figura 121. Simplificación de variables en el evento *OnSignal*.

En la Figura 122 se muestra cómo se realiza el registro de la información recogida (línea 117), donde usaremos un condicional para coordinar la entrada de las informaciones recogidas de componentes que necesitemos que cumplan un orden (línea 115). Si fueran otros los componentes que necesitamos coordinar la entrada de las comunicaciones deberíamos de usar este condicional. Luego incrementamos en una unidad la variable encargada de contabilizar la cantidad de información que vamos recogiendo en la lista que recoge la información (línea 120). Más adelante realizamos la comprobación de la cantidad de información registrada por el script (línea 123), en caso de que exceda esta cantidad se encargará de llamar a la función '*exportDataComunication*' que se encarga de abrir el fichero y escribir en él la información (línea 124), posteriormente pondría el contador de nuevo a 0 (línea 125), e incrementará el número de iteraciones (línea 127) que indican el orden de registro en el fichero.

```
112
113     #El objetivo de usar este condicional es evitar que se solapen las comunicaciones en los componentes EDGE y M
114     #registrada en el csv sería erronea con respecto a estos componentes, en caso de que todo esté correcto añadi
115     if not ((emisor == 'EDGE' or emisor == 'M3 Gage CMM') and (receptor == 'EDGE' or receptor =='M3 Gage CMM')):
116         #en esta linea añadimos las variables que queremos exportar al csv, solo es necesario anadir o quitar las v
117         fila = [contador_iteraciones,timeStamp, emitter, receiver, msgType,datasize,latency,reliability]
118         lista_comunicaciones.append(fila)
119     #añadimos al contador de registros uno mas
120     contador_registros += 1
121
122     #Cada cierto número de iteraciones, se exportará los datos guardados en la lista
123     if contador_registros == REINICIO_ITERACIONES:
124         exportDataComunication()
125         contador_registros = 0
126
127     contador_iteraciones += 1
```

Figura 122. Registro de información en la lista que recoge los registros.

11.2.4. *GetStates* y *StatesRecording*

En la Figura 123 realizamos la especificación del nombre de la carpeta donde queremos guardar el fichero que crearemos (línea 22), obtenemos el nombre del usuario que tenga abierta la sesión (línea 24) y con ello obtenemos la ruta donde estará la carpeta en la que guardaremos el fichero (línea 26), por defecto hemos puesto que sea en el escritorio principal.

```
19  #+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
20  #CREACION DE CARPETA Y RUTA DE ARCHIVO
21  #Nombre de la carpeta donde almacenaremos el csv con los datos
22  nombre_carpeta = "Datasets"
23  #Obtenemos nombre de usuario
24  username = os.getenv("USERPROFILE")
25  #obtenemos la ruta donde crearemos la carmpeta
26  path = os.path.join("C:\\Users\\", username, "Desktop")
27
```

Figura 123. Especificación de ruta en script *GetStates*.

En la Figura 124 hemos declarado las variables que usaremos en el script. Necesitaremos el objeto del escenario (línea 29), la lista de componentes (línea 30), y una variable que apunte al objeto '*statistics'* del componente '*Data Collector*' (línea 31), esto se hace solo para referenciar un objeto de este tipo y que más adelante al intentar acceder no de error. Usaremos también un diccionario llamado '*dict_states*' que se encargue de guardar los distintos estados que tienen cada componente siendo el nombre de este la clave para acceder (línea 34), y por otro lado tendríamos la lista '*list_states*' que la usaremos para guardar los estados de los componentes que posteriormente insertaremos en el diccionario.

```
28  #Accedemos a los objetos de lista de componentes, estadist
29  app = getApplication()
30  components = app.Components
31  statistics = getComponent().findBehaviour('statistics')
32
33  #Diccionario que guarda los distintos estados que tiene ca
34  dict_states = {}
35  #Lista auxiliar que recoje los estados y luego los añade
36  list_states = []
```

Figura 124. Declaración de variables en script *GetStates.*

La funcionalidad de este script la hemos implementado en el evento *OnRun* tal y como muestra la Figura 125. Para ello hemos abierto el fichero en la ruta que hemos obtenido anteriormente (línea 42) y una vez abierto escribiremos la primera fila que hace referencia a las columnas de componentes y sus respectivos estados (línea 45), separados por el delimitador que le indiquemos que en este caso es el carácter ‘,’ (línea 44).

```
38  def OnRun():
39      global statistics, dict_states, list_states
40
41      #abrimos fichero
42      f = open(path + "\\"+ nombre_carpeta + "\\data_states.csv", "w")
43      #establecemos el delimitador del csv
44      state_writer = csv.writer(f, delimiter=",", lineterminator='\n')
45      state_writer.writerow(['Components:', 'States'])
46
```

Figura 125. Evento *OnRun* del script *GetStates*.

Con la Figura 126 mostramos la última parte del script, donde recorremos todos los componentes (línea 48), comprobamos si tiene el componente ‘*statistics*’, que guarda los estados (línea 50), y en caso de que sí que lo tenga accedemos a él (línea 52), posteriormente accedemos a los estados que guarda el objeto (línea 54) y los recogemos línea (línea 57-59). Por último, lo escribimos todo en el fichero (línea 63) y lo cerramos (línea 66).

```
46
47      #recorremos toda la lista de componentes
48      for component in components:
49      #comprobamos que tenga el objeto que contiene los distin
50        if component.findBehaviour('Statistics'):
51          #apuntamos al objeto que contiene los estados del co
52          statistics = component.findBehaviour('Statistics')
53          #apuntamos a los estados
54          states = statistics.States
55
56          #recogemos en la lista el nombre del componente y po
57          list_states.append(component.Name)
58          for state in states:
59            list_states.append(state[0])
60            #print list_states
61
62          #escribimos en el fichero la información recogida
63          state_writer.writerow(list_states)
64          dict_states[component.Name] = list_states
65          del list_states[:]
66      f.close()
```

Figura 126. Creación de la lista de estados posibles para cada componente en el script *GetStates*.

De manera similar a *GetStates*, el script *StatesRecording* accede al componente ‘*statistics*’ para leer en tiempo real el estado actual de cada componente. Sin embargo, los cambios de estado no se han simulado en el escenario tal y como se presentó en el apartado 11.1.3.

12. Referencias

[1] *A 5G Traffic Model for Industrial Use Cases*, 5G-ACIA Whitepaper, Noviembre 2019.

[2] 3GPP; Technical Specification Group Services and System Aspects; Service requirements for cyber-physical control applications in vertical domains; Stage 1 (Release 17), TS 22.104 v17.7.0, sept. 2021.

[3] Visual Components website: https://www.visualcomponents.com/. Último acceso 18/11/2021.

[4] Visual Components Academy: https://academy.visualcomponents.com/. Último acceso 18/11/2021.