UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

SMARTIOT4INDUSTRY

Trabajo Fin de Máster

Máster de Ingeniería y Tecnología de Sistemas Software

Autor: Alejandro Martinez Blanco

Tutor/es: Joan Fons Cors

Vicente Pelechano Ferragud

2019/2020

Agradecimientos

Después de un intenso período de varios meses, hoy es el día: escribo este apartado de agradecimientos para finalizar mi trabajo de fin de máster. Ha sido un período de aprendizaje intenso, no solo en el campo científico, pero también a nivel personal. Escribir este trabajo ha tenido un gran impacto en mí y es por eso por lo que me gustaría agradecer a todas aquellas personas que me han ayudado y apoyado durante este proceso.

Primero de todo, me gustaría agradecer a los tutores de máster por su colaboración. Me habéis apoyado enormemente y siempre habéis estado ahí para ayudarme cuando lo necesitaba. Me gustaría agradeceros vuestra cooperación y daros las gracias por todas las oportunidades que me habéis dado durante la investigación sobre mi trabajo. Definitivamente me habéis brindado todas las herramientas necesarias para completar mi trabajo de fin de máster satisfactoriamente.

También me gustaría agradecer a mi chica Belén por su apoyo, sabios consejos y su comprensión.

¡Muchas gracias a todos!

Resumen

El término “Industria 4.0” fue utilizado por primera vez en la Feria de Hannover en el 2011. El concepto de Industria 4.0, engloba la intención de conectar personas, máquinas y productos para la producción eficiente y procesos de valor agregado a la red digital también conocida como Fábrica Inteligente o Smart Factory (CATANIA). Por lo tanto, el concepto explota la idea de máquinas altamente conectadas y automatizadas, que usan recursos e interfaces de red para monitorear de forma autónoma su trabajo y el entorno circundante.

Uno de los principales problemas a los que se enfrenta actualmente esta novedosa idea de “Industria 4.0”, es que trata de aunar tecnologías complemente diferentes entre sí, que por sí solas y sin la interacción con los diferentes componentes del sistema productivo y con otras tecnologías desplegadas no son capaces de aportar el valor añadido buscado, ya que sin una arquitectura de comunicación y procesos estándares definidos es complicado que todos los sistemas se integren correctamente en la plataforma de control industrial.

Tomando como bases las ideas de la industria 4.0 y partiendo de una arquitectura IoT, a lo largo de este documento se ha diseñado una arquitectura de comunicación donde los diferentes actores del sistema productivo junto con la introducción de nuevas disciplinas en el ámbito de la computación autónoma y adaptativa puedan comunicarse de manera natural y dotando al sistema de auto adaptación evolucionando la Industria 4.0 desde “el proceso de automatización industrial” a la “industria IoT autónoma”.

Palabras clave: Industria 4.0, interfaces multimodales, IoT, auto adaptación, lenguaje natural, industria IoT autónoma.

Abstract

The term "Industry 4.0" was first used at the Hannover Fair in 2011. The concept of Industry 4.0 encompasses the intention to connect people, machines and products for efficient production and value-added processes to the digital network also known as Smart Factory [1]. Therefore, the concept exploits the idea of highly connected and automated machines, which use network resources and interfaces to autonomously monitor their work and the surrounding environment.

One of the main problems currently faced by this novel idea of "Industry 4.0" is that it tries to bring together technologies that are completely different from each other, which by themselves and without the interaction with the different components of the production system and with other deployed technologies are not capable of providing the desired added value, since without a defined communication architecture and standard processes it is complicated for all systems to be correctly integrated into the industrial control platform.

Taking as a basis the ideas of the industry 4.0 and starting from an IoT architecture, along this document a communication architecture has been designed where the different actors of the productive system together with the introduction of new disciplines in the field of autonomous and adaptive computing can communicate in a natural way and providing the system with self-adaptation evolving the Industry 4.0 from "the industrial automation process" to the "autonomous IoT industry".

Keywords: Industry 4.0, multimodal interfaces, IoT, self-adaptation, natural language, autonomous IoT industry.

Índice de abreviaturas

AI	Artificial intelligence
AmI	Ambient Intelligence
API	Application Programming Interface
ARN	Amazon Resource Name
ASK	Alexa Skills Kit
AWS	Amazon Web Service
CoAP	(Constrained Aplication Protocol)
CPU	Central Processing Unit
GB	GigaByte
HTTP	Hypertext Transfer Protocol
IoT	Internet of things
IP	Internet Protocol
JSON	JavaScript Object Notation
LTE	Long Term Evolution
M2M	Machine to Machine
MQTT	Message Queue Telemetry Transport
MVVM	Model–View–ViewModel pattern
PHP	Hypertext Preprocessor
PLC	Programmable Logic Controller
PNL	Programación Neuro-Lingüística
RAM	Random Access Memory
SAPI	Microsoft Speech API
SD	Secure Digital
SDK	Software Development Kit
TICs	Tecnologías de la Información y la Comunicación
TCP	Transmission Control Protocol
TFM	Trabajo Fin de Máster
UI	User interface
UML	Unified Modeling Language

Tabla de contenidos

Tabla de Ilustraciones

Tabla de Tablas

1. **Introducción**

En este documento se expone el trabajo de fin de máster llamado “SmartIoT4Industry” correspondiente al Máster Universitario de Ingeniería y Tecnología de Sistemas Software de la Universidad Politécnica de Valencia.

En este capítulo se describirán los motivos para la elaboración de este documento, los objetivos que se esperan lograr, el impacto que se espera conseguir con el cumplimiento de estos objetivos, la metodología empleada para el desarrollo del proyecto, la planificación seguida y la estructura utilizada durante todo el proceso de desarrollo de esta memoria.

Cuando se piensa en un entorno industrial y de fabricación, a menudo se recurre a la idea de producción en masa, con maquinaria enorme y un grupo masivo de personas que realizan una multitud de trabajos. Aunque muchas empresas industriales y fábricas han estado utilizando formas de tecnología digital durante tiempo, solo en los últimos años esto ha comenzado a ser más generalizado y demandado por la misma.

Actualmente la mayoría de las empresas están considerando en automatizar al máximo las tareas que se realizan dentro de las fábricas, desde que entra la materia prima, hasta finalizar todos los procesos requeridos para convertir dicha materia en un producto final. Este concepto esta conocido como Industria 4.0 (industria inteligente).

Términos como Internet de las cosas (IoT por sus siglas en inglés), Inteligencia Artificial (IA por sus siglas en inglés) y computación autónoma cada día están más ligados al concepto de Industria 4.0, aplicando a la industria manufacturera un concepto de integración de maquinaria de fabricación inteligente, automatización impulsada por inteligencia artificial y análisis avanzado para ayudar a que cada trabajador y cada fábrica sean más eficientes.

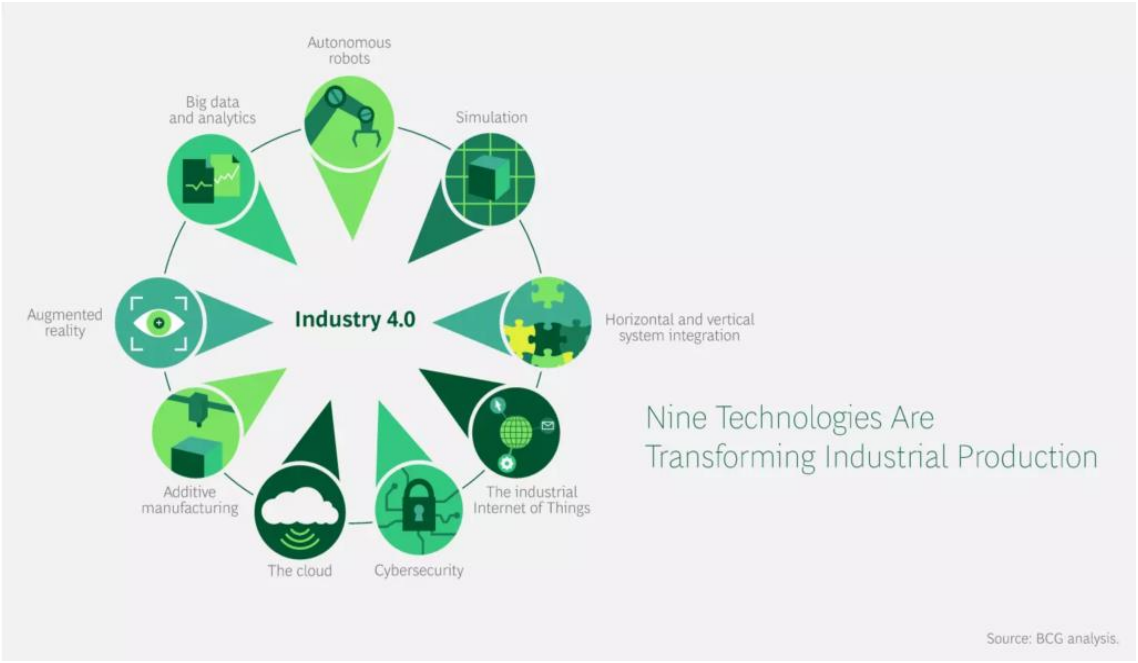


Ilustración 1 Industria 4.0

Así mismo, los nuevos sensores y dispositivos del IoT, hacen uso y propulsan el desarrollo de nuevas tecnologías, sobre todo en el campo de las telecomunicaciones: protocolos, modulaciones, etc. Esto ocurre en particular en dispositivos de difícil acceso o que no pueden estar cableados. Algunas de las más conocidas son tecnologías de comunicación como Zigbee, LTE, LoRa, etc. y protocolos de comunicación como MQTT.

En este contexto se enmarca este Trabajo de Fin de Máster (TFM) que se centrará precisamente en el análisis y diseño de un marco teórico/practico de comunicación bidireccional entre humanos y dispositivos IOT aplicados a la gestión de máquinas de producción dentro del ámbito de la industria 4.0. Además de ello, el desarrollo y despliegue de un prototipo aplicado de la solución que permitirá de forma sencilla e intuitiva la validación del modelo propuesto.

1.1 **Motivación del proyecto**

La motivación principal de este trabajo es la integración dentro del marco de la industria 4.0 de tecnologías nuevas emergentes relacionadas por un lado con la comunicación humano-maquina a través del procesamiento del lenguaje natural permitiendo dicha interacción dentro de un entorno de Inteligencia Ambiental para la industria a través del habla. Para ello tomando como referencia las actuales soluciones tecnológicas de interacción por voz como son ‘Ok Google’, Alexa o Siri, y sus dispositivos específicos asociados. Por otro lado, la introducción de un paradigma de computación autónoma y la implementación de un modelo mínimo de dicho bucle de control en el ámbito del prototipo a desarrollar.

1.2 **Objetivos del proyecto**

El TFM propuesto se centra en el desarrollo de un marco teórico/practico que proporcionará una interfaz de comunicación bidireccional entre humanos y dispositivos IoT, desplegados en la plataforma AWS IoT de Amazon. La comunicación de llevará a cabo a través de interfaces basadas en lenguaje natural dentro del ámbito industrial de la Industria 4.0 La funcionalidad del sistema englobará el envío de órdenes desde el usuario final a la máquina, la consulta del estado de funcionamiento a través de los diferentes sensores de monitorización, la implantación, a través de un sistema de control basado en un bucle MAPE-K, de un sistema de alertas al usuario final con diferentes niveles de notificación, aportando incluso autoadaptación, si se ha producido un cambio del estado en los sensores de los dispositivos que afecten al correcto funcionamiento del proceso de producción monitorizado.

La plataforma proveerá al usuario final de un sistema diálogo multimodal que le permita la interacción con un entorno de Inteligencia Ambiental para una industria a través del habla, tomando como referencia las actuales soluciones tecnológicas de interacción por voz como son ‘Ok Google’, Alexa o Siri, y sus dispositivos específicos asociados.

Para alcanzar el objetivo principal, se procederá a diseñar y desarrollar diferentes componentes software:

- Se diseñará una solución arquitectónica que abordará estos retos definiendo la infraestructura necesaria para poder comunicar de manera digital todos los procesos de una cadena de producción (máquinas personas, ordenes de

producción), desde el procesamiento de la materia prima hasta la creación del producto final, la seguridad de las infraestructuras y dotar al sistema de funcionalidades inteligentes que prevengan perdidas de materia prima, mejora en el mantenimiento tanto preventivo como correctiva y reducción del tiempo de acción de las personas en las diferentes líneas de trabajo de la empresa.

- Se analizarán las diferentes arquitecturas tecnológicas y se implementará la que más se ajuste a las necesidades del proyecto.

- Se diseñará e implementará una skill de Amazon la cual permitirá la interacción del usuario de línea trabajo a través del lenguaje natural con la funcionalidad del sistema. Sera el punto de entrada de las ordenes humanas al sistema, las cuales se canalizarán hasta la controladora de la maquina gracias a la arquitectura diseñada.

- Se implementará una función lambda que se ejecutará dentro del Core de Amazon AWS la cual recibirá órdenes desde la skill de Alexa a través de las "invocation order" y contactará con las diferentes controladoras de las maquinas a desarrollar a través del Broker MQTT asociado para hacerle llegar las ordenes tanto de funcionalidad como de ordenes de trabajo para que las maquinas ejecuten dichas órdenes.

- Se implementará una versión mínima (debido a las limitaciones temporales y no ser el scope central del proyecto) de un bucle de control MAPE-K. Este bucle de control analizara de manera autónoma ciertas condiciones del estado de las máquinas de producción y reaccionara de manera concreta según se ha definido en el plan de acción de este para evitar o dar solución a situaciones inesperadas en el funcionamiento de las líneas de producción.

- Se realizará una pequeña aproximación a modo de prototipo para poder realizar una de validación de la propuesta arquitectónica a través del desarrollo de un caso de estudio concreto y real, con dispositivos que podrían utilizarse a nivel industrial. Dentro del ámbito de este prototipo de validación se van a desarrollar los siguientes componentes:

    - Se desarrollará un simulador de controladora de máquina de línea de producción. Los simuladores recibirán los órdenes a través del broker MQTT y ejecutarán las ordenes como si de la maquina real se tratase. Este simulador aparte de recibir las órdenes del broker y simular las acciones que se realizarían sobre el PLC de esta, también simula la producción de cartoneras según la orden de trabajo que tiene asociada y la velocidad a la cual trabaja la máquina para poder tener un escenario completo de testeo de la solución propuesta.

    - Se desarrollará una aplicación de monitorización visual y de interacción vocal del estado de las diferentes líneas de trabajo para poder ver como las diferentes líneas de trabajo reaccionan a las diferentes ordenes que le llegan desde el usuario final a través de la skill desarrollada o desde la aplicación de computación autónoma (bucle de control MAPE-K)

1.3 **Metodología**

Para el desarrollo de este caso de estudio, se ha decidido elegir una metodología basada en un desarrollo de ciclo de vida de espiral definido por primera vez por Barry Boehm en 1986 (Boehm, 1986). El modelo en espiral es una combinación entre el modelo lineal o de cascada y el modelo iterativo o basado en prototipos.

El modelo de desarrollo en espiral se caracteriza por los siguientes ciclos:

- Definición de objetivos: Los objetivos se determinan juntamente con el cliente. Al mismo tiempo, se discuten posibles alternativas y se especifican las condiciones marco (por ejemplo, sistemas operativos, entornos y lenguajes de programación).

- Análisis y diseño de la solución: En este ciclo se definen los diferentes componentes funcionales, diseño de la arquitectura, análisis protocolos de comunicación y posibles prototipos. También se evalúan las alternativas existentes y evaluación de riesgos. Los riesgos son registrados, evaluados y luego reducidos utilizando prototipos, simulaciones y softwares de análisis.

- Desarrollo y prueba: Los prototipos se amplían y se añaden funcionalidades. El código real es escrito, probado y migrado a un entorno de prueba varias veces hasta que el software pueda ser implementado en un entorno productivo.

- Planificación del siguiente ciclo: El siguiente ciclo se planifica al final de cada etapa. Si se producen errores, se buscan soluciones, y si una alternativa es una mejor solución, se prefiere en el siguiente ciclo.

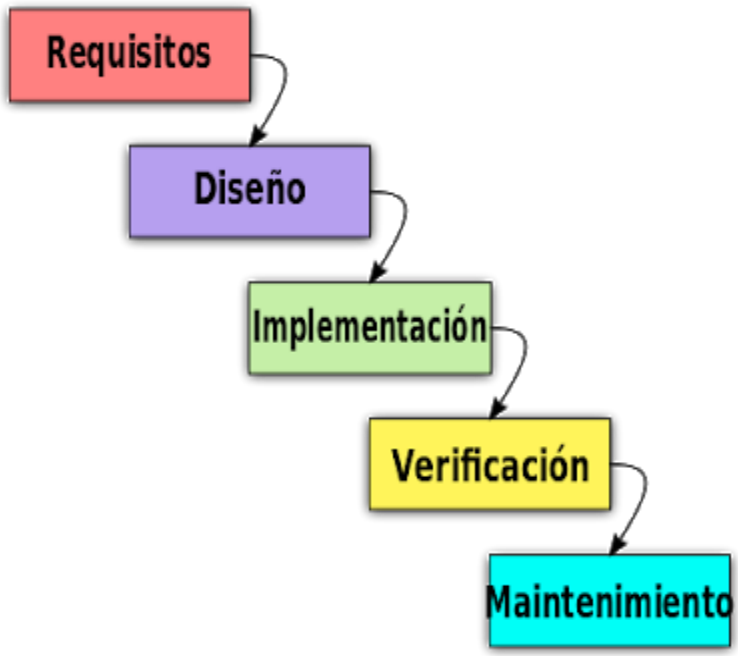


Ilustración 2 Modelo de desarrollo de ciclo de vida en espiral

En primer lugar, se han fijado los objetivos y requisitos principales que debe cubrir la solución a desarrollar acordándolas con los tutores de este proyecto, para establecer las directrices del desarrollo del trabajo.

Una vez definidos los objetivos a alcanzar dentro del ámbito del propuesto TFM se arranca la fase de análisis y diseño de la arquitectura de la aplicación. Dentro de esta fase de análisis teórica se englobará 1) el análisis de la integración de dispositivos IoT dentro de la infraestructura IoT seleccionada, 2) estudio y diseño de la interacción humano-plataforma a través de las interfaces de lenguaje natural disponibles actualmente, 3) diseño de arquitectura de comunicación y de mensajería para la interacción de los diferentes componentes de la solución y 4) el estudio y diseño de un bucle control en el proceso de producción para aportar cierta autoadaptación a la solución propuesta la cual pueda dar respuesta a errores críticos que pudieran ocurrir en el sistema productivo.

Una vez superada la fase de diseño, la más importante de este trabajo, se propone el desarrollo de un prototipo de la solución diseñada que permita validar la solución propuesta. En esta fase de implementación se llevará a cabo el desarrollo de una skill que junto con algún dispositivo de interacción de lenguaje natural permitirá la interacción de manera bidireccional del usuario con los diferentes controladores de las maquinas a través de la plataforma AWS IoT. En esta fase se llevarán a cabo las labores de integración de que verifican la calidad y el correcto funcionamiento del sistema en su conjunto, como la preparación de la mayoría de la documentación asociada al mismo.

Por último, en una fase de verificación se realizarán las pruebas necesarias para validar el funcionamiento correcto y corregir los posibles errores que vayan surgiendo durante esta etapa de validación. Al final de esta fase se espera tener una versión final y estable del prototipo de validación propuesto y con ella una validación de la arquitectura propuesta durante la fase de análisis y diseño.

1.4 **Planificación**

Uno de los aspectos más importantes en el momento de realizar un proyecto es seguir una planificación adecuada que permita satisfacer de forma correcta todas las pretensiones definidas dentro del marco de este proyecto de TFM.

Siendo la planificación aplicada a este proyecto la siguiente:

- ***Preparación de la memoria***: Durante la fase de ejecución del proyecto se ha ido desarrollando la memoria de forma paralela para documentar las diferentes fases de este.
- ***Investigación previa***: Debido al uso de nuevas tecnologías como skill de Alexa, Amazon AWS, paradigmas recientes como la computación autónoma y el procesamiento del lenguaje natural ha sido necesario esta fase de investigación para la documentación y compresión de forma correcta de todas las tecnologías citadas.
- ***Caso de estudio***: Una vez llevado a cabo ese proceso de investigación previa, se ha procedido al planteamiento del caso de estudio a realizar y los diferentes escenarios a desarrollar para cumplir las soluciones propuestas.
- ***Diseño***: En esta fase se ha abordado el diseño arquitectónico de la solución, se ha definido la librería de mensajería para la comunicación entre los

diferentes componentes del sistema, se ha definido los “topics” del bróker mqtt y se ha diseñado los diferentes componentes del bucle mape-k propuesto.

- ***Implementación***: Una vez diseñada la solución, en esta fase se ha desarrollado el prototipo piloto que nos ha servido para la validación de la solución dentro del marco tecnológico propuesto
- ***Pruebas***: Esta fase es la de realizar el testing de cada módulo que compone el proyecto con el fin de certificar su correcto funcionamiento y la interacción correcta entre los diferentes componentes softwares.

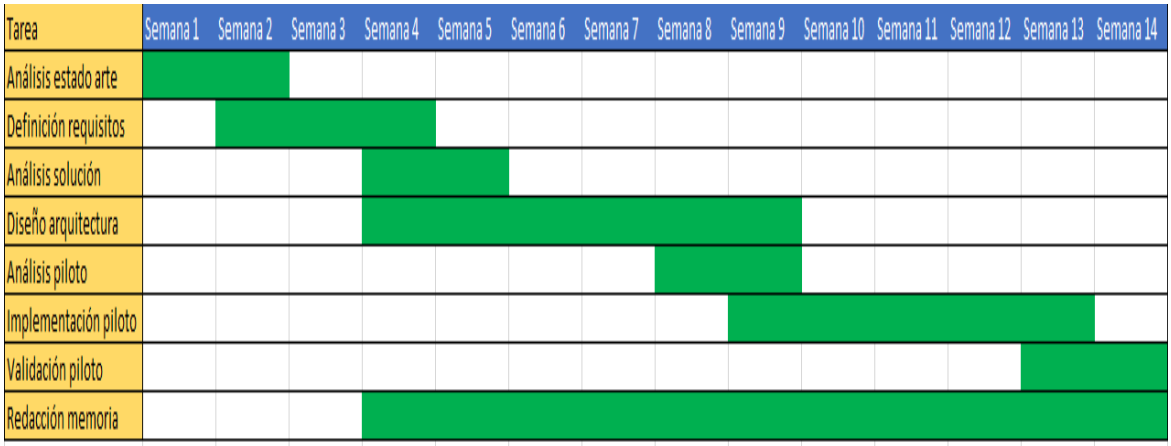


Ilustración 3 Diagrama de Gantt del proyecto

2. **Contexto tecnológico**

Tras la introducción donde se han definido los principales objetivos y la motivación de este trabajo, en este capítulo se realizará una revisión de los principales conceptos, disciplinas, herramienta tecnologías y protocolos que se utilizaran durante la fase de diseño y desarrollo del proyecto. Las citadas herramientas no son las únicas que podrían utilizarse para la elaboración del proyecto, pero tras una fase previa de análisis donde se han barajado diferentes alternativas se ha determinado por motivos de adaptabilidad, eficiencia, disponibilidad y facilidad para construir el proyecto propuesto.

2.1 **¿Por qué IoT en el entorno industrial?**

La palabra IoT viene del inglés "Internet of Things" y se refiere a la interconexión de dispositivos a través de una red, donde los objetos pueden comunicarse e interactuar desde cualquier lugar del mundo sin necesidad de intervención humana.

La interoperabilidad es uno de los aspectos clave de IoT que contribuyen a su creciente popularidad. Los dispositivos conectados o "inteligentes", tienen la capacidad de recopilar y compartir datos de sus entornos con otros dispositivos y redes. Pero la idea más interesante de este paradigma y que encaja a la perfección con los objetivos de este proyecto, tiene que ver no solamente con la obtención de la información acerca del entorno, sino que dichos datos nos permitirán controlarlo y actuar sobre él.

Ilustración 4 IoT en el entorno industrial

La introducción del IoT en la industria puede dotar al sector de diferentes beneficios para lograr el objetivo de cualquier proceso logístico: maximizar la rentabilidad reduciendo los costes operativos. Entre sus múltiples ventajas para la industria, se podría distinguir:

- **Aumento de la eficiencia y del rendimiento del almacén**: la interconexión entre personas y maquinaria facilitará la optimización de cada proceso logístico.
- **Mantenimiento predictivo de la maquinaria**: las máquinas contarán con sensores que, sin intervención humana, avisarán a los operarios de cualquier riesgo de avería antes de que se produzca. Gracias a esta función, se acrecentará la productividad de cada operativa.
- **Nuevas oportunidades de negocio**: monitorizar cada dato que produce la compañía permitirá anticiparse a las nuevas tendencias del mercado y prever qué segmento de negocio puede ser más interesante para la empresa.
- **Mejora de la eficiencia energética**: la incorporación del internet de las cosas cambiará el paradigma de eficiencia energética, puesto que dotará al almacén de sensores que ayudarán a optimizar los consumos energéticos.
- **Seguridad industrial**: la conectividad pasa por ser la mejor aliada de la seguridad en el almacén. Los procesos en los que intervengan elementos IoT contarán con sensores y monitores para prevenir cualquier accidente que pudiera ocurrir en la instalación.

Alguna de las ventajas enumeradas anteriormente junto a otras que se identificaran a lo largo del siguiente capítulo gracias a la introducción de nuevos paradigmas como adaptación autónoma y de la introducción de comunicación bidireccional Humano-Maquina serán las principales ventajas de las que podrá beneficiarse cualquier empresa que introduzca la solución propuesta, la cual está basada en una arquitectura IoT, dentro de su proceso productivo.

2.1.1 ¿Qué es MQTT? Su importancia como protocolo IoT

MQTT (ISO/IEC, 2016-06) es un protocolo de red abierto, ligero, de publicación y suscripción estándar que transporta mensajes entre dispositivos. El protocolo generalmente se ejecuta sobre TCP / IP; sin embargo, cualquier protocolo de red que proporcione conexiones bidireccionales ordenadas y sin pérdidas puede admitir MQTT.

El protocolo MQTT define dos tipos de entidades de red: un intermediario de mensajes “Message Broker” y uno o varios clientes. Un broker de MQTT es un servidor que recibe todos los mensajes de los clientes y luego los enruta a los clientes de destino apropiados. Un cliente MQTT es cualquier dispositivo (desde un microcontrolador hasta un servidor completo) que ejecuta una biblioteca MQTT y se conecta a un agente MQTT a través de una red.

La información está organizada en una jerarquía de “Topics”. Cuando un cliente quiere publicar un nuevo elemento de datos para distribuir, publica un nuevo mensaje de control con los datos en el Broker. El broker luego distribuye la información a cualquier cliente que se haya suscrito a ese tema. El cliente publicador no necesita tener ningún dato sobre el número o la ubicación de los suscriptores, y los suscriptores a su vez no tienen que estar configurados con ningún dato sobre el cliente que publico dicho mensaje.

Los principios de diseño son minimizar el ancho de banda de la red y los requisitos de recursos del dispositivo, al mismo tiempo que intentan garantizar la confiabilidad y cierto grado de seguridad de la entrega. Estos principios también hacen

que el protocolo sea ideal para el emergente "máquina a máquina" (M2M) o "Internet de las cosas" IoT del mundo de los dispositivos conectados y para las aplicaciones móviles donde el ancho de banda y la energía de la batería son importantes.

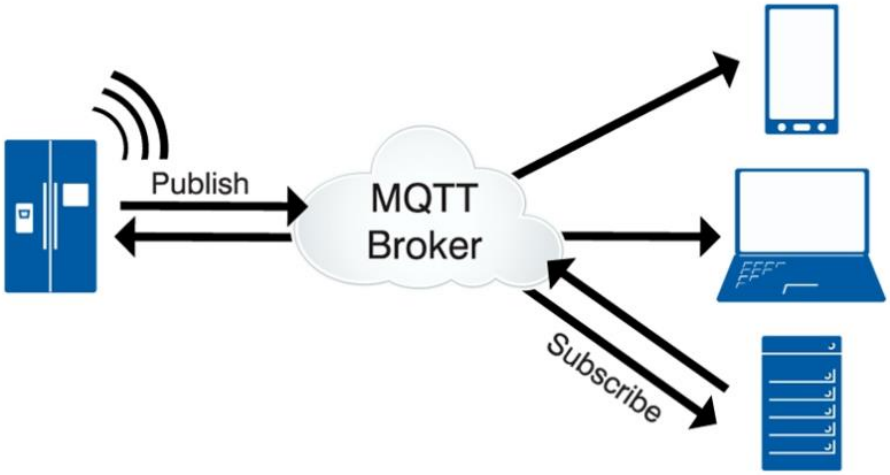


Ilustración 5 Arquitectura MQTT (infoplc)

El protocolo MQTT ha sido elegido el protocolo de comunicación de la arquitectura diseñada para realizar el proyecto, por su paradigma de comunicaciones, publicador/suscriptor, y porque es uno de los más empleados en la industria debido a la baja sobrecarga del canal que aporta el protocolo, lo que hace que sea muy interesante su uso en dispositivos que tienen poca capacidad computacional como las controladoras de la maquinaria que se comunicaran con el sistema propuesto.

2.1.2 **Estructura de datos: JSON sobre MQTT**

Se ha decidido utilizar JSON como estructura de modelo de datos. JSON por sus siglas en inglés (JavaScript Object Notation), es un estándar de datos basado en texto plano para el intercambio de información, por lo que se usa en muchos sistemas que requieren mostrar o enviar información para ser interpretada por otros sistemas.

Una de las más grandes ventajas que aporta el uso de este standard, es que tiene un formato totalmente independiente de cualquier lenguaje de programación. Esto permite que los servicios puedan compartir información sin necesidad de hablar el mismo idioma (lenguaje de programación). Con lo que emisor y receptor pueden utilizar lenguajes distintos, pero son capaces de decodificar cadenas de JSON gracias a su librería propia. Esta ventaja de independencia de lenguaje de programación dota a la solución propuesta de una gran adaptabilidad a diferentes escenarios donde diferentes actores de la arquitectura general pueden estar implementados con diferentes lenguajes de programación y aun así poder comunicarse de manera transparente gracias a la versatilidad de este estándar de datos.

2.2 **Disciplinas tecnológicas adaptables al entorno industrial**

Mark Weiser en (The computer for the 21st Century, 2002) introdujo el concepto de computación ubicua en el año 1988 para referirse al proceso por el cual

los ordenadores se integran perfectamente en un mundo físico. A partir de ese momento, numerosos investigadores han desarrollado multitud de dispositivos móviles y aplicaciones que facilitan la interacción computacional en tiempo real. Se ha generado una gran proliferación de dispositivos físicos, donde la presencia de los ordenadores es menos visible, la nueva tecnología se entremezcla discretamente en nuestro día a día, a través de dispositivos integrados en los objetos más cotidianos La visión de Weiser se dirigía hacia la investigación en la creación de dispositivos de interacción que fueran sensibles al contexto, robustos y capaces de capturar datos naturales respecto a los siguientes ámbitos:

- Interfaces naturales multimodales: El deseo de interactuar con interfaces naturales que faciliten una mayor capacidad de comunicación entre los seres humanos y la computación.
- Inteligencia ambiental: La necesidad de que las aplicaciones de computación ubicua sean sensibles al contexto y que adapten su comportamiento a la información detectada del entorno físico y computacional que les rodea.
- Computación ubicua: El esfuerzo por automatizar la captura de experiencias en vivo y proporcionar después acceso flexible y universal a dichas experiencias.

El objetivo de estas disciplinas se centra en entender cómo se pueden mejorar las tareas cotidianas y en qué grado se alteran, con la introducción de tecnologías ubicuas. La computación cotidiana, con la que promueven la idea de la necesidad de impulsar esa disponibilidad de interacción hombre-máquina, 24 horas al día, los 7 días de la semana, los 365 días del año. Si bien es cierto que el PC, Internet y la Web han influido ya en muchos aspectos de nuestro mundo (por ejemplo, en la convergencia de los medios de comunicación, el entretenimiento, la electrónica de consumo, las telecomunicaciones, etc.), la siguiente ola de la revolución tecnológica nos afectará mucho más directamente en el proceso productivo con el surgimiento de la industria 4.0. El desarrollo de nuevos marcos de comunicación y gestión de la industria, como el propuesto en este documento, permitirá a las empresas que los introduzcan en su proceso productivo una mayor producción orientada a demanda, uso más eficiente de los recursos, reducción de tiempos de mantenimiento, incremento de la productividad, reducción de costos, ciclo de desarrollo de productos más cortos, mayor competitividad, optimización de los procesos, autonomía en la toma de decisiones y cadenas de suministro más integradas. Para conseguir dotar al entramado industrial con los objetivos tan ambiciosos deseados no basta solo con la introducción de sensorización y comunicación a través una arquitectura IoT de los diferentes componentes de la industria, sino que se deben aprovechar los avances de diferentes disciplinas en pleno crecimiento para dotar al entorno de cierta auto adaptabilidad, inteligencia y la introducción de interfaces multimodales y naturales con el deseo de facilitar y mejorar la comunicación humano-maquina a través de interfaces basadas en lenguaje natural.

2.2.1 **Interfaces multimodales**

En la actualidad, las interfaces más utilizadas para la comunicación hombre-máquina son el teclado, el ratón o tableta táctil. Estos dispositivos representan la adaptación de los seres humanos a las limitaciones de los ordenadores en lugar de una comunicación natural con el ordenador. En los últimos años, ha aparecido un nuevo requisito y es que los seres humanos necesitan comunicarse con las máquinas

de la misma manera como lo hacen entre sí: por el habla, mímica y gestos, ya que éstos conciben mucha más información que los anteriores dispositivos periféricos. Esto nos lleva al término interfaz multimodal (MMI - multimodal interface) (Juraj Kačur, Primera edición, 2017).

La computación con interfaces naturales supone alejarse del escritorio del ordenador e interactuar con los sistemas computacionales como los seres humanos interactúan con el mundo físico. Esto plantea dos problemas más en los que pensar: Cómo tratar con tipos de datos naturales, como nuevo sistema de datos de entrada, y cómo manejar los errores producidos por sistemas basados en el reconocimiento de formas, del habla, de imágenes, etc. La captura electrónica del movimiento de la mano es relativamente fácil usando pens, inks y tablets especiales. Pero interpretar y manipular gestos como palabras y números u otros caracteres es mucho más difícil. En cuanto a la detección y manejo de errores con datos naturales, no se pueden tratar como datos anómalos y eliminarlos, por lo que es necesario un nuevo sistema para manejarlos, así como una infraestructura que permita la reutilización de dicho manejador de errores.

Uno de los principales objetivos de las interfaces Multimodales es que “deben llevar al usuario a la acción desde el primer momento”, en este punto es importante recalcar que para que los sitios o aplicaciones diseñados con multimodalidad deben ser testeados considerando no solamente la usabilidad tradicional, sino que también el correcto funcionamiento de las distintas interfaces, la comodidad física del usuario y la precisión de las instrucciones iniciales. También los recursos gráficos deben ser un aporte fundamental en cuanto al dinamismo del sitio y a la diferenciación de objetos y colores, de preferencia deben ser formas básicas, colores contrastantes que faciliten la lectura y selección de elementos.

La demanda a nivel industrial de este tipo de interfaces multimodales para la comunicación hombre-maquina ha crecido recientemente en la última década de la mano de los avances tecnológicos en visión artificial, procesamiento lenguaje natural, inteligencia ambiental, computación en la red etc. La introducción de este tipo de interfaces multimodales está especialmente dirigido a aquellas industrias que hasta ahora han sido reticentes a la utilización de soluciones robóticas, debido a que los procedimientos empleados son demasiado complejos para su automatización. El nuevo sistema ofrecerá soluciones para automatizar los procesos más exigentes, conocerá el contexto del mantenimiento y ejecución de las ordenes de trabajo, será seguro para los trabajadores, flexible, adaptable a los cambios dinámicos y rentable.

2.2.2 **Procesamiento del lenguaje natural**

La forma en que interactuamos con las computadoras ha sufrido cambios importantes desde el inicio de la computación; que van desde la utilización de tarjetas perforadas, hasta los esfuerzos de hoy en día, para que interactuemos sin dispositivos informáticos aparentes, con los que realicemos la comunicación.

A este tipo de interacción, se le ha dado el nombre de: Interfaces Naturales de Usuario (NUI por sus siglas en inglés). La computación con interfaces naturales supone alejarse del escritorio del ordenador e interactuar con los sistemas computacionales como los seres humanos interactúan con el mundo físico, es decir, una interacción natural y fluida, como hablar, escribir, hacer un gesto, etc. Esto plantea diversos aspectos/problemas más en los que pensar:

- Cómo tratar con tipos de datos naturales, como nuevo sistema de datos de entrada.
- Cómo manejar los errores producidos por sistemas basados en el reconocimiento de formas, del habla, de imágenes, etc.
- Facilidad para determinar qué acciones son posibles en cada momento.
- Facilidad de evaluación el estado actual del sistema.
- Alineamiento natural entre la intención y acción del usuario; entre acción y resultado; y entre información visible e interpretación del estado del sistema.

Uno de los campos de las ciencias de la computación en el que se basa este tipo de interfaces naturales para las interacciones entre los computadores y el humano es el procesamiento del lenguaje natural (PNL). Los fundamentos de PNL se encuentran en una serie de disciplinas tan variadas como informática y ciencias de la información, lingüística, matemática, ingeniería eléctrica y electrónica, inteligencia artificial y robótica, psicología, etc. Esta área de investigación y aplicación explora cómo las computadoras se pueden utilizar para comprender y manipular texto o lenguaje en lenguaje natural para hacerlas útiles. Los investigadores de la PNL tienen como objetivo reunir conocimiento sobre cómo los seres humanos entienden y usan lenguaje para que se puedan desarrollar herramientas y técnicas apropiadas para crear sistemas informáticos comprender y manipular los lenguajes naturales para realizar las tareas deseadas (Chowdhury, 2003).

Las aplicaciones de PNL incluyen una serie de campos de estudio, como la máquina traducción, procesamiento y resumen de texto en lenguaje natural, interfaces de usuario, multilingüe y recuperación de información en varios idiomas reconocimiento de voz, inteligencia artificial y ambiental entre otras. Como veremos en a lo largo de este documento, para el desarrollo de este proyecto, se han utilizado dos técnicas dentro del ámbito de las interfaces naturales que son Alexa skill (sistemas de reconocimiento de voz y comprensión del lenguaje natural) y Microsot Speech Api (aplicación de reconocimiento de voz y síntesis de voz).

2.2.3 **Inteligencia ambiental**

La Inteligencia Ambiental (AmI por sus siglas en inglés) constituye una evolución de las tecnologías TICs respondiendo a una demanda actual de servicios y de información omnipresentes. La Inteligencia Ambiental engloba una amplia variedad de tecnologías con capacidad sensorial, de procesamiento, de razonamiento que facilitan la comunicación entre diferentes dispositivos todo ello en un entorno distribuido y con capacidad de interacción con los usuarios (López, 2017)

.

La visión principal de la Inteligencia Ambiental presenta al usuario rodeado de interfaces inteligentes e intuitivas, integradas en los objetos cotidianos de su entorno de forma transparente. Estas interfaces poseen capacidad para reconocer la presencia de diferentes usuarios, y modificar su comportamiento en función de la identidad de dicho usuario, sus necesidades y las características del contexto o entorno donde se encuentren. Especial atención merecen aspectos como la facilidad de uso, el soporte eficiente de los servicios y la posibilidad de mantener interacciones naturales con las personas (Altimir).

Las características principales que cualquier sistema de Inteligencia artificial debe cumplir son los siguientes:

- Computación ubicua
- Información sensible al contexto
- Aprendizaje autónomo
- Interacción natural / transparente
- Espacios inteligentes

La mayoría de las investigaciones en curso sobre Inteligencia Ambiental se focalizan en el ámbito del hogar digital, pero la aplicación de este tipo de sistemas en el entorno industrial puede permitir la creación de entornos inteligentes que se adapten automáticamente al tipo de usuario que esta interaccionando con ellos, sin necesidad de una identificación previa en el sistema. Estos sistemas se pueden aplicar en el proceso de producción o en el proceso de distribución de los productos dentro de nuestra industria objetivo. En ambos casos, estos sistemas pretenden ayudar a los trabajadores en sus tareas diarias, facilitando su trabajo al darles información precisa sobre él, aportando valor añadido a la solución diseñada embebiendo ciertos tipos de sistemas de Inteligencia Ambiental dentro de la arquitectura propuesta.

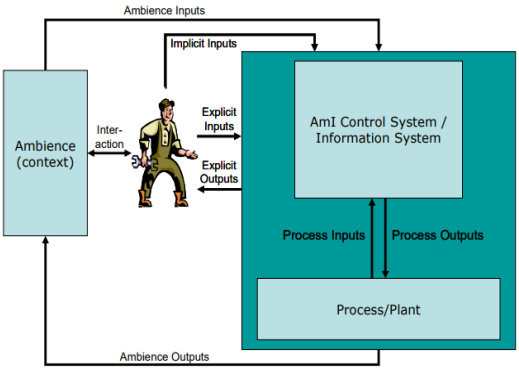


Ilustración 6 Sistema de inteligencia ambiental (Dr. Dragan Stokic)

2.2.4 **Computación autónoma**

Los sistemas de software modernos suelen funcionar de forma dinámica en entornos cambiantes y lidian con condiciones operativas muy cambiantes: los componentes pueden aparecer y desaparecer, pueden encontrarse temporal o permanentemente no disponibles, puede cambiar su comportamiento, etc. La autoadaptación es ampliamente reconocida como un enfoque efectivo para tratar con la creciente complejidad, incertidumbre y dinámica de estos sistemas.

Un enfoque de ingeniería bien reconocido de la autoadaptación es la introducción de autoadaptación en los sistemas TIC mediante la implantación de un sistema independiente de control de retroalimentación loop llamado MAPE-K (ver Ilustración 7). Este bucle es concebido como una secuencia de cuatro subsistemas:

- Monitorización : Se encarga de recoger uy filtrar los datos recogidos por los diferentes sensores del sistema.
- Análisis: Compara el funcionamiento del sistema respecto a los valores normales y objetivos definidos.
- Planificación: Determina las acciones correctivas a realizar para mitigar o eliminar las consecuencias de las situaciones anormales detectadas
- Ejecución: Ejecuta las acciones de reparación o reconfiguración planificadas en la fase anterior basada en el análisis del problema.

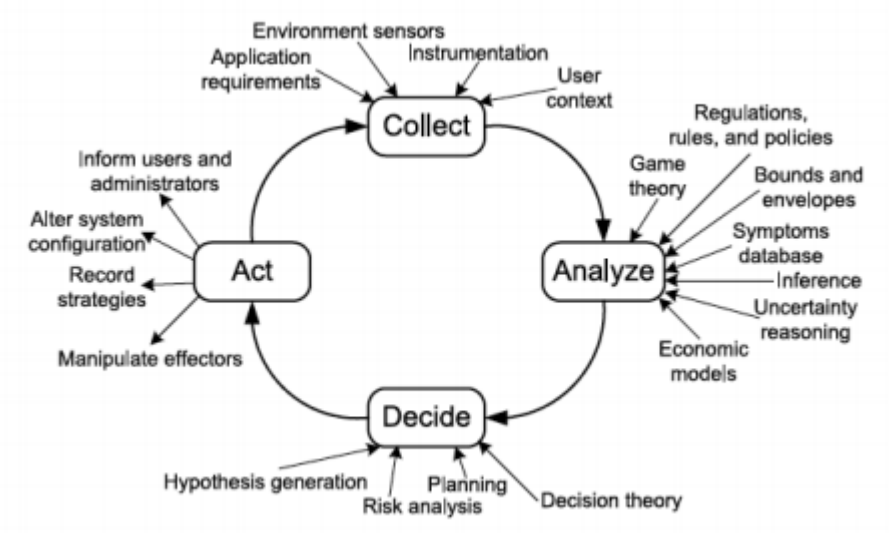


Ilustración 7 Bucle de control tipo MAPE-K

Estos avances tecnológicos relacionados con la autoadaptación aportan un gran valor añadido a sistemas cada vez más grandes y complejos donde la gestión manual se convierte en un proceso cada vez más inviable y donde la automatización de tareas coge cada vez una mayor relevancia. La solución propuesta por este tipo de tecnologías se basa en la creación de procesos informáticos que tengan la capacidad de ejecutarse sin la necesidad de la interacción humana, capaces de detectar eventos imprevistos en el funcionamiento normal del sistema y tener la habilidad de autorregularse para mitigar o eliminar las consecuencias de las situaciones anormales detectadas. La introducción de este tipo de tecnologías dentro de la solución propuesta en este proyecto, y en general en los sistemas TIC encargados de los procesos productivos en las industrias automatizadas 4.0 puede aportar grandes beneficios a dichos sistemas, entre ellos se incluyen:

- Evitar los problemas técnicos que causan costosos cortes de producción.
- Dotar de un alto grado de confiabilidad y estabilidad en los diferentes sistemas TIC.
- Mantener la integridad del proceso de producción con menos supervisión humana.
- La capacidad de los sistemas físicos cibernéticos de tomar decisiones por sí mismos y realizar sus tareas de la manera más autónoma y eficaces posibles.

2.3 **Herramientas de desarrollo y simulación**

En este apartado se definen qué tipo de tecnologías se han utilizado para llevar a cabo el diseño, desarrollo y testeo del proyecto propuesto.

2.3.1 **Infraestructura IoT. Amazon IoT Core**

AWS IoT Core es un servicio en la nube administrado que permite a los dispositivos conectados interactuar de manera fácil y segura con las aplicaciones en la nube y otros dispositivos. AWS IoT Core admite miles de millones de dispositivos y billones de mensajes, y es capaz de procesarlos y direccionarlos a puntos de enlace

de AWS y a otros dispositivos de manera fiable y segura. Con AWS IoT Core, las aplicaciones pueden realizar un seguimiento de todos los dispositivos y comunicarse con ellos en todo momento, incluso cuando no están conectados.

AWS IoT Core facilita la utilización de servicios de AWS y Amazon, como AWS Lambda para crear aplicaciones de IoT que recopilen, procesen, analicen y utilicen datos generados por dispositivos conectados sin tener que administrar ninguna infraestructura.

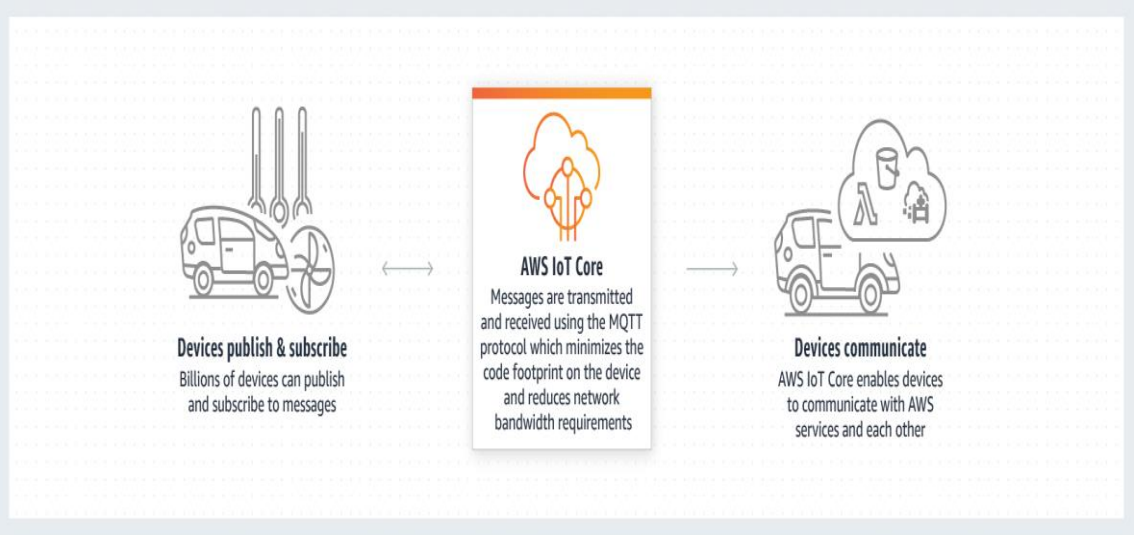


Ilustración 8 Amazon IoT Core

2.3.2 **Servicio servesless. AWS Lambda**

AWS es una plataforma de computación sin servidor que permite ejecutar código sin aprovisionar ni administrar servidores. AWS Lambda ejecuta el código bajo demanda y escala automáticamente los recursos, desde unas pocas solicitudes por día hasta miles por segundo.

Con AWS Lambda, se puede ejecutar código para prácticamente cualquier tipo de aplicación o servicio de back-end, todo con cero administraciones. AWS Lambda ejecuta su código en una infraestructura de cómputo de alta disponibilidad y realiza toda la administración de los recursos de cómputo, incluido el mantenimiento del servidor y del sistema operativo, el aprovisionamiento de capacidad y el escalado automático, el monitoreo y el registro del código. (Amazon Inc)

2.3.3 **Asistente virtual controlado por voz**

Alexa es un asistente virtual desarrollado por Amazon, su primer uso fue dentro de la gama de altavoces inteligentes “Amazon Echo”

El servicio de voz de Alexa permite acceder a las capacidades ofrecidas por Alexa basadas en la nube con el soporte de las API de “Alexa Voice Service”. Este servicio permite a los desarrolladores integrar Alexa directamente en sus productos, proporcionando que se puedan controlar estos productos mediante la voz. “Alexa Voice Service Device SDK” (Mvelegon) proporciona librerías en C++ que aprovechan la API de “Alexa Voice Service” para crear el software adecuado para productos habilitados de Alexa. Se puede modular, proporcionando diversas componentes para manejar las funciones como el reconocimiento de la voz. Soporta diferentes idiomas

entre ellos inglés, alemán, japonés, francés, italiano y español. Para la conversión de la voz a texto

Amazon adquirió Ivona (Barr), que es donde se encuentra la API Amazon Polly que es la que permite convertir su audio a texto o viceversa. Tiene un abanico de voces diferentes para ciertos idiomas, entre otras funciones tiene la de ajustar la velocidad de la voz a la de la imagen que se proyecta, es decir si el vídeo está en inglés y se quiere traducir al alemán esta aplicación corregirá la velocidad automáticamente para que dure igual.

Es compatible con varias plataformas y lenguajes de programación como Java, JavaScript, PHP, entre otros o SDK para móviles como iOS y Android. Amazon también soporta una API de HTTP para así poder implementarse, con el uso de "Alexa Web Services" y derivados la propia capa de acceso.

Amazon Echo es un altavoz inteligente de la compañía Amazon. Este altavoz funciona con AWS que, como se ha comentado en el punto anterior, se trata de los servicios de computación en la nube que tiene Amazon. El dispositivo por defecto está a la escucha de la conversación hasta que escucha la palabra Alexa.

2.3.4 **Como interaccionar con Alexa**

Alexa Skills Kit es un kit de desarrollo de software (SDK) que permite a un desarrollador desarrollar habilidades, también llamadas aplicaciones de conversación, en el asistente de inteligencia artificial Amazon Alexa.

El kit de habilidades de Alexa se compone de herramientas, interfaces de programas de aplicación (API), muestras de código y documentación que permite a un desarrollador agregar habilidades a las capacidades de reconocimiento de voz de más de 10,000 disponibles en Alexa.

Con el Kit de habilidades de Alexa, puede crear habilidades con un modelo de interacción personalizado. Implementa la lógica de la habilidad y también define la interfaz de voz a través de la cual los usuarios interactúan con la habilidad. Para definir la interfaz de voz, asigna la entrada hablada de los usuarios a las intenciones que su servicio basado en la nube puede manejar. Para declarar esta asignación, proporcione las siguientes entradas:

- **Intents**: un intento representa una acción que cumple la solicitud oral de un usuario. Los intents pueden tener opcionalmente argumentos llamados espacios. Los intents se especifican en una estructura JSON llamada esquema de intents.
- **Sample utterances**: un conjunto de posibles frases habladas asignadas a los intentos. Esto debe incluir tantas frases representativas como sea posible.
- **Custom slot types**: una lista representativa de valores posibles para un espacio. Los tipos de espacios personalizados se utilizan para listas de elementos que no están cubiertos por uno de los tipos de espacios integrados de Amazon.
- **Dialog model**: una estructura que identifica los pasos para una conversación de varias vueltas entre su habilidad y el usuario para recopilar toda la información necesaria para cumplir con cada intento.

2.3.5 **Sintetizador de voz. Microsoft Speech**

La interfaz de programación de aplicaciones de voz o SAPI (abreviatura en inglés de Microsoft Speech API) es una API desarrollada por Microsoft para permitir el uso de aplicaciones de reconocimiento de voz y síntesis de voz dentro de aplicaciones de Windows (Wikipedia).

Hasta la fecha, se han lanzado una serie de versiones de la API, que se han distribuido ya sea como parte de un Speech SDK , o como parte de Windows en sí. Las aplicaciones que utilizan SAPI incluyen Microsoft Office, Microsoft Agent y Microsoft Speech Server.

En general, todas las versiones de la API se han diseñado de tal manera que un desarrollador de software puede escribir una aplicación, para llevar a cabo el reconocimiento de voz y síntesis, mediante el uso de un conjunto estándar de interfaces, accesibles desde una variedad de lenguajes de programación. Además, es posible que una tercera empresa tercera producir sus propios motores de reconocimiento de voz y de texto o adaptar los motores existentes para trabajar con SAPI. En principio, siempre y cuando estos motores se ajustan a las interfaces definidas, pueden ser utilizados en lugar de los motores suministrados por Microsoft.

La API de voz es un componente de libre distribución que se puede enviar en cualquier aplicación de Windows que desea utilizar la tecnología de voz. Muchas versiones (aunque no todos) de los motores de reconocimiento de voz y síntesis también son de libre distribución.

2.3.6 **Raspberry Pi**

La Raspberry Pi es una placa de ordenador simple compuesto por un SoC, CPU, memoria RAM, puertos de entrada y salida de audio y vídeo, conectividad de red, ranura SD para almacenamiento, reloj, una toma para la alimentación, conexiones para periféricos de bajo nivel, reloj, etc. Para ponerlo en marcha tenemos que conectar periféricos de entrada y salida para poder interactuar como una pantalla, un ratón y un teclado y grabar un sistema operativo para Raspberry Pi en la tarjeta SD. Ya solo queda conectarlo a la corriente y estamos listos para funcionar.

Ilustración 9 Placa base - Raspberry, Pi 4 Modelo B 4GB

3. **Caso estudio**

3.1 **Introducción**

Tras la introducción de las diferentes tecnologías relacionadas con el concepto de la IOT, del tratamiento del lenguaje natural, la computación autónoma, de la auto adaptación y de la industria 4.0, la principal idea de este caso de estudio es la fusión de estas diferentes tecnologías para definir y desarrollar un marco teórico/practico que proporcionará una interfaz de comunicación bidireccional entre humanos y dispositivos IOT, desplegados en la plataforma AWS IOT de Amazon. La comunicación de llevará a cabo a través de interfaces basadas en lenguaje natural dentro del ámbito industrial de la Industria 4.0

La funcionalidad del sistema englobará el envío de órdenes desde el usuario final a la máquina, la consulta del estado de funcionamiento a través de los diferentes sensores de monitorización, la implantación, a través de un sistema de control basado en un bucle MAPE-K, de un sistema de alertas al usuario final con diferentes niveles de notificación, aportando incluso autoadaptación, si se ha producido un cambio del estado en los sensores de los dispositivos que afecten al correcto funcionamiento del proceso de producción monitorizado.

La idea de este proyecto de enfoque practico es incorporar este marco teórico/practica de la industria 4.0 a una empresa española real. La empresa seleccionada ha sido Embalpack. Embalpack se dedica a la fabricación de cantoneras de cartón y están especializados en la fabricación, transformación y distribución de productos para todo tipo de embalaje con la máxima calidad y protección de su contenido.

Las empresas como Embalpack, tratan de aprovechar la tecnología más avanzada en sus líneas de producción con el fin de obtener un producto de alta calidad, capaz de adaptarse a las necesidades de nuestros clientes. Las fábricas que se aprovechan de Iot y forman parte de la industria 4.0 ya obtienen diversos beneficios. La aplicación de procesos basados en internet y nuevas tecnologías permite tomar decisiones consensuadas respecto a la monitorización de procesos físicos, ya que se detectan antes los fallos y errores y facilita las posibles correcciones. Además, posibilita conectar diferentes etapas de producción y obtener información a tiempo real y anticiparse a las necesidades de los compradores.

La aplicación de nuevas tecnologías en el sector de la manufactura permite a las empresas ser competitivos en un terreno en pugna constante por ofrecer la mejor calidad y fidelizar al consumidor. Un ecosistema dominado también por una profunda globalización. Las principales tendencias en fabricación y manufacturación pasan por la disminución de errores en procesos de fabricación y la creación de fábricas más eficientes.

3.2 **Problemática presente en Embalpack**

Una de las principales problemáticas a las que se enfrenta la empresa Embalpack en su día a día dentro del proceso de producción de las cartoneras de papel es 1) la gestión de las ordenes de trabajo dentro de las diferentes líneas de producción de la fábrica y 2) el mantenimiento de las máquina de cada una de las

líneas de producción las cuales se encargan de transformar la materia prima previamente modificada (en ese momento de la producción al producto se le denomina torta) y convertirla en el producto final tal y como se ha definido en la orden de trabajo asociada.

Actualmente tanto las ordenes de trabajo como la asignación de estas a las diferentes líneas de producción de la fábrica se realizan de forma manual. Esto requiere que un operario vaya configurando las máquinas para ajustarlas a la orden de trabajo designada y las ordenes de trabajo se actualicen de forma manual si alguna lo requiere una vez definidas.

Por otro lado, la modificación del estado de funcionamiento actual de la maquina desde el inicio y parado, pasando por la velocidad de producción de la orden de trabajo, hasta la pausa o reanudación de la producción de la maquina actualmente es una proceso manual, donde los operarios de línea tiene que interaccionar físicamente con la máquina para realizar cualquier cambio de estado de la misma, desde la parada controlada si se queda sin torta dentro de la bobina, o una parada de emergencia si una situación inesperada aparece.

Otro de los principales problemas a los que se enfrentan diariamente los operarios de las líneas de producción es la parada de emergencia de una maquina si la cuchilla de corte ya no está suficientemente afilada para poder cortar la torta y esto provoca que las cajetillas no se corten correctamente y se tenga que parar la producción, perder materia prima que haya salido de la bobina y volver a ajustar la orden de trabajo con la cantidad de cajetillas correctamente generadas.

Por último, se ha pensado que se es importante tener una excelente trazabilidad de los procesos dentro de la compañía, ya que de esta forma podremos saber cosas como quién ha manipulado alguna materia o maquinaria de la empresa. Esta es una manera de obtener información para sacar estadísticas sobre el tiempo de acción y producción dentro de la fábrica y poder optimizar los recursos y hacer a la empresa más eficiente.

Dar una solución tecnológica que sirva para la detección autónoma y preventiva de errores y una manera mucho más rápida y ágil de proceder con acciones correctivas puede generar un marco muy interesante para que Embalpack pegue el salto a la nueva industria, la industria 4.0.

3.3 **Requisitos iniciales de la solución**

Para conseguir que Embalpack o cualquier otra empresa que implante esta arquitectura de comunicación y monitorización pueda dar el salto a la empresa 4.0 se definen una serie de requisitos iniciales que se basan en que la empresa pueda:

- Sacarle el máximo rendimiento al personal y a las líneas de producción
- Minimizar las pérdidas de materia prima, reducir al máximo las paradas de línea debido a mantenimientos correctivos

Haciendo en general los procesos productivos más eficientes. Para ello se han identificado diferentes funcionalidades básicas (las cuales podrán ser extendidas en un futuro cercano debido a la flexibilidad de la solución propuesta) que permitirán a los diferentes componentes de la solución intercambiarse mensajes definidos para facultar a los actores activos del sistema (usuarios de control de línea y bucle de control MAPE-K) a realizar acciones sobre las diferentes líneas de producción.

Para el objetivo de la mejora en el rendimiento del personal y de las líneas de producción se han identificado las siguientes ordenes sobre las líneas de producción:

- *Arrancar la línea de producción N*: Con este mensaje la maquina arranca y queda a la espera la carga de una orden de trabajo sobre ella para empezar la producción.

- *Parar la línea de producción N*: Con este mensaje la maquina se apaga deja de producir la orden de trabajo si estaba en un proceso de producción sobre alguna de ellas.

- *Aumentar la velocidad de producción a X en la línea de producción N:* Con este mensaje la maquina aumenta en X velocidad la producción de unidades de la orden de trabajo que se está ejecutando en el momento de recibir la orden.

- *Disminuir la velocidad de producción a X en la línea de producción N:* Con este mensaje la maquina disminuye en X velocidad la producción de unidades de la orden de trabajo que se está ejecutando en el momento de recibir la orden.

- *Añade orden de trabajo Q sobre la línea de producción N:* Con esta orden, la máquina que tiene acceso a las líneas de trabajo carga los valores de esta en su sistema, ajusta los parámetros de la maquina automáticamente dejando así lista la línea de producción para cuando el operario de línea decida arrancar la producción de dicha orden de trabajo.

- *Inicia/Reanuda/Pausa/Termina la orden de trabajo Q en la línea de producción N:* Con este mensaje la máquina, a la cual previamente se le ha cargado una línea de trabajo, dependiendo de la orden arrancar o detiene la producción de la orden de trabajo actual si el operario lo requiere debido a una parada prevista o imprevista.

- Añade X unidades producidas a la orden actual Q: Con este mensaje, el operario de la línea puede aumentar las unidades de la orden de trabajo que ya se está procesando si la operativa lo requiere. Este mensaje será enviado a todas las líneas de producción y las controladoras serán las encargadas de discriminar dependiendo de si esa máquina esta justamente produciendo la orden de trabajo Q o no.

- Añade medición a la orden de trabajo Q: Con este mensaje, el operario de la línea puede añadir ciertas anotaciones a la orden de producción actual para su registro posterior una vez procesada la totalidad de las unidades definidas en dicha orden de trabajo . Este mensaje será enviado a todas las líneas de producción y las controladoras serán las encargadas de discriminar dependiendo de si esa máquina esta justamente produciendo la orden de trabajo Q o no.

Por otro lado, para minimizar en la empresa las pérdidas de materia prima y reducir al máximo las paradas de línea debido a mantenimientos correctivos, se han identificado las siguientes ordenes sobre las líneas de producción:

- *Reporte estado línea de producción N:* Este mensaje que se emitirá periódicamente contiendo información acerca del estado de funcionamiento de

la línea de producción junto al estado de la orden de trabajo que se está procesando dentro de la misma si la maquina está en funcionamiento en ese momento, se podrá monitorizar el estado de ciertos componentes o estados de la maquina como una cuchilla poco afilada, velocidad excesiva de producción, tortas de cartón a punto de acabar, temperatura excesiva, etc.

- *Parara emergencia la línea de producción N*: Con este mensaje la maquina se apaga y deja de producir la orden de trabajo si estaba en un proceso de producción sobre alguna de ellas debido a una situación inesperada detectada por el bucle de control autónomo.

4. **Análisis de la solución**

4.1 **Descripción contexto del caso de estudio**

Embalpack es una empresa especializada en la fabricación, transformación y distribución de productos para todo tipo de embalaje con la máxima calidad y protección de su contenido. Teniendo la capacidad de fabricar un gran abanico de referencias combinando diversos anchos, espesores, longitudes y tipos de papel. Las diferentes cartoneras que actualmente puede producir la empresa son cantonera blanca, cantonera blanca brillo, cantonera Kraft o marrón y cantonera troquelada. Todos estos tipos de cartoneras están realizados en cartón estratificado 100% reciclable y biodegradable, de gran resistencia. Son disponibles en medidas variables tanto para formas circulares, cuadradas o rectangulares. Con distintos anchos de ala (de 33 a 80 mm) y distinto espesor (de 2 a 6 mm) según demanda.

Los pedidos de los diferentes clientes se subdividen a su vez en diferentes ordenes de trabajo dependiendo de los tipos de cartoneras y los espesores demandados. Estas órdenes de trabajo son las unidades mínimas de información que las controladoras de las diferentes líneas de producción manejan para llevar a cabo la transformación de la materia prima denominada torta en las diferentes cartoneras. En cada proceso productivo participa típicamente uno o varios operarios, la materia prima a convertir, una orden de trabajo específica y una máquina de producción de cartoneras de la fábrica. A través de una carretilla mecánica, cada operario carga la materia prima de un espesor especifico en los diferentes rodillos de las maquinas dependiendo que orden de trabajo va a ejecutar. Actualmente el proceso de asignación, procesamiento y seguimiento del proceso productivo de las ordenes de trabajo en cada máquina, como la modificación de otro valor de dichas maquinas se realiza manualmente y con la necesidad de que los operarios interaccionen con las maquinas directamente a través de su interfaz gráfica o botones de señales analógicas.

El objetivo de aplicar la solución propuesta en este proyecto a esta empresa es convertirla en una empresa un paso más allá del proceso de conversión a una empresa automatizada, sino acercarla cada vez más al concepto de empresa autónoma, elevando la automatización de procesos a otro nuevo nivel. Con la ayuda de la implantación de este sistema, las diferentes líneas de producción, los dispositivos de entrada/salida, sensores e interfaces podrán interactuar entre sí de manera cada vez más autónoma permitiendo en la empresa un aumento de la eficiencia en el control de las máquinas y los procesos productivos que se ejecutan en ellas. El despliegue de esta solución en el entorno industrial traerá más agilidad a la operación de las fábricas, disponiendo rápidamente de más información y análisis de datos. Esto ayuda tanto a la toma de decisiones de negocio como al mantenimiento predictivo de los equipos industriales. El uso de aprendizaje automatizado podría perfeccionar el software que determina cuáles componentes de la maquinaria necesitan ser sustituidos.

Ahora bien, para implantar la solución propuesta existen diferentes deficiencias en el estado de automatización de la empresa ya que con las maquinas actuales de Embalpack no hay posibilidad de automatización completa, por lo tanto, para llevar a cabo una validación total del sistema dentro de la empresa lo que haremos es una simulación de las controladoras de la maquinaria y virtualizaremos la ordenes de

trabajo que transforman la materia prima en el producto final. Así queda un escenario de la siguiente manera:

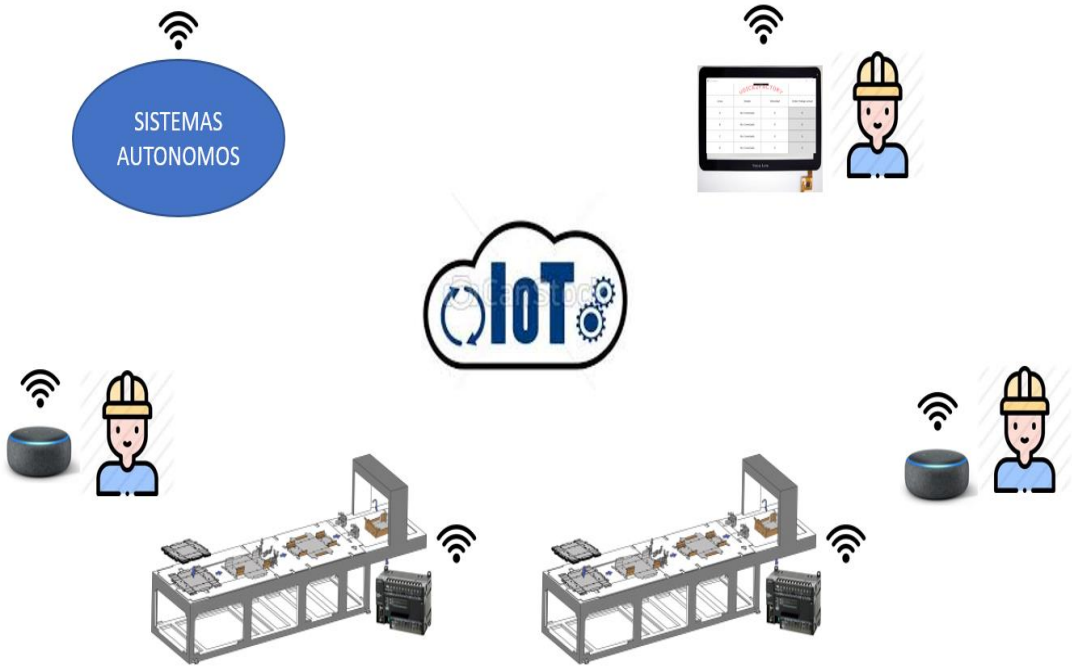


Ilustración 10 Esquema virtual Embalpack

Como se muestra en la imagen anterior, cada una de las diferentes máquinas de producción de cartoneras, concretamente las controladoras simuladas, estarán conectadas por red inalámbrica a internet o a la red local de la empresa según la implantación local o descentralizada que se realice. La implantación del broker de comunicaciones y lógica de control centralizada con la función lambda podrá desplegarse en entornos IoT comerciales como Amazon AWS o de manera local en servidores propietarios de la empresa Embalpack.

Por otro lado, los diferentes operarios encargados del control del proceso productivo tendrán a su mano un altavoz inteligente (Amazon Echo) para poder interactuar con el sistema a través de la skill VoiceToFactory emitiendo las ordenes que tome oportunas para la eficiencia del proceso productivo y una Tablet con la herramienta de reporting la cual le ofrecerá de manera intuitiva y rápida una visión general del estado de las diferentes máquinas de la fábrica y el estado del proceso productivo total que se está generando en dichas máquinas, ofreciéndole señales sonoras si se detecta alguna situación inconsistente en el sistema productivo o en el estado de las máquinas como una cuchilla poco afilada, una bobina a punto de acabar, atascos de la cinta de transporte de las cartoneas, sobrecalentamientos, etc.

Por ultimo y no por ello menos importantes, dentro de la arquitectura implantada, existe el componente que añade cierta autoadaptibilidad e inteligencia al sistema. Estos sistemas son independientes y a través de la infraestructura de comunicación propuesta y gracias a la librería de mensajería diseñada, pueden analizar la información que fluye desde todas las controladoras de la fábrica e interferir en el proceso productivo si así lo determinan basados en las reglas definidas.

4.2 **Marco conceptual del dominio**

Tras una descripción del contexto del caso de estudio, dentro de la fase de análisis de la solución, para poder realizar una correcta modelización de la solución propuesta, definiendo los diferentes subcomponentes que la definen y como están relacionados entre sí abordando la complejidad del sistema, se han definido diferentes tipos de diagrama de tipo UML que muestran las diferentes vistas de la arquitectura del sistema a través de una representación gráfica del conjunto de subsistemas del modelo y sus relaciones.

4.2.1 **Casos de uso**

Los siguientes diagramas de casos de uso identifican las diferentes funciones del sistema que utilizarán el sistema y sus acciones asociadas. Los actores involucrados se describen a continuación:

- **Operario/s:** la persona que utiliza la aplicación y tiene acceso a la comunicación con el dispositivo echo de Amazon para la invocación de las ordenes definidas.

- **Función lambda:** Interpreta las órdenes del usuario que se envían a través de alguno de los “intent” definidos dentro de la skill desarrollada.

- **Broker mensajería:** envía datos desde la función lambda a los diferentes controladores de las máquinas de las líneas de trabajo

- **Controlador línea:** Por un lado, recibe los mensajes del broker mqtt que se envían desde la función lambda o desde el bucle de control MAPE-k, modifica el estado de funcionamiento de la maquina cartonera y/o modifica las ordenes de trabajo que se ejecutan en la cartonera de dicha controladora. Por otro lado, envía reportes periódicos a través del broker mqtt para que la interfaz de control de estado de líneas actualice periódicamente el estado de las diferentes líneas de trabajo.

- **Reporting tool:** Herramienta de reporting que notifica a nivel gráfico y vocal el estado de trabajo y de las diferentes ordenes de trabajo que se están procesando en ese momento de las diferentes líneas de producción de la fábrica.

- **Bucle MAPE-K:** El bucle de control se encarga de monitorizar de manera autónoma el estado de diferentes variables como el estado de afilado de la cuchilla de corte de la máquina. Estas variables se van actualizando periódicamente con el mensaje de “ReportStatus” que emiten las diferentes controladoras a través del broker MQTT.

Para formar una idea global de la aplicación, se extraen de los requisitos los casos de uso siguientes:

Identificador:	**1.0**
Titulo:	Caso de uso general.
Descripción:	El caso de uso muestra una versión general de las diferentes interacciones de los actores dentro del sistema definido.
Actores:	Todos los actores del sistema.
Precondiciones:	N/A
Postcondiciones:	N/A
Frecuencia de uso:	N/A

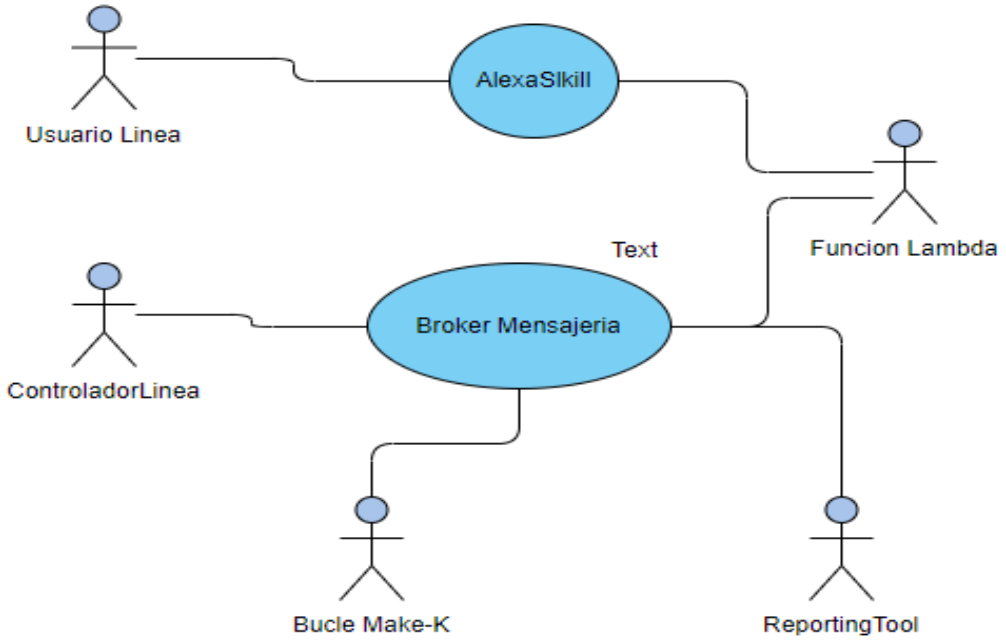


Ilustración 11 General de la solución

Identificador:	**2**
Titulo:	Envió ordenes función lambda desde operario de fábrica.
Descripción:	El caso de uso muestra la interacción entre el operario de fábrica y la función lambda a través de la skill de Alexa definida.
Actores:	Operario y función lambda.
Precondiciones:	El operario a través del dispositivo Amazon echo invoca la skill de VoiceToFactory.
Acciones:	El operador a través del lenguaje de la skill envía una orden dentro de los “Intents” definidos hacia la función lambda.
Postcondiciones:	La función lambda recibe la invocación desde la skill, interpreta la información que se ha enviado y contesta con un mensaje de SkillResponse al usuario a través del dispositivo de Amazon Echo. Tras ello, construye un mensaje que enviara a través del broker a las controladoras.
Frecuencia de uso:	Cada vez que el usuario decide enviar una orden a las diferentes controladoras de las líneas de producción de las fábricas.

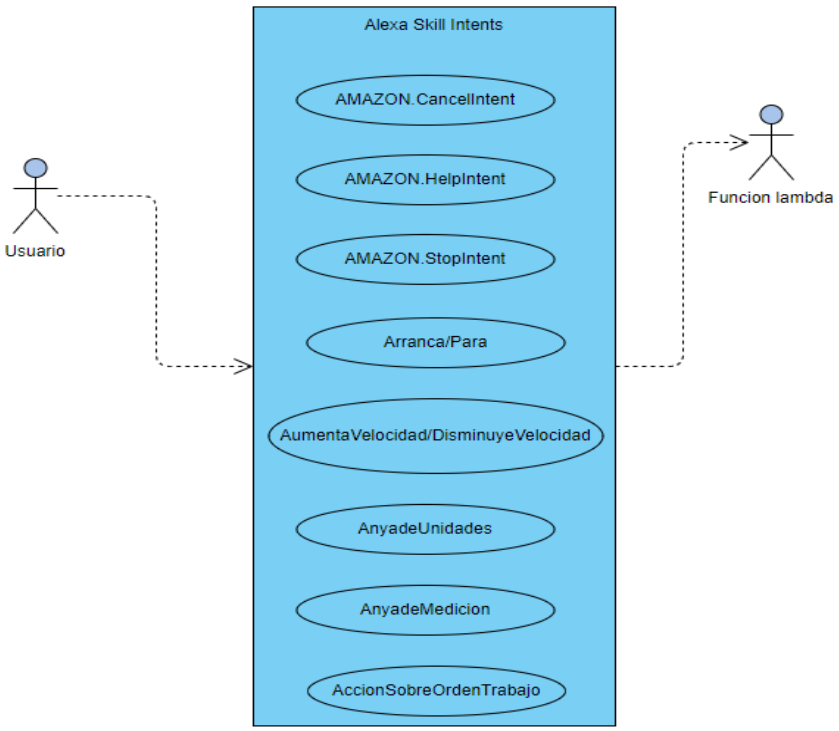


Ilustración 12 Envió ordenes función lambda desde operario de fábrica

Identificador:	**3**
Titulo:	Envío de órdenes a controladoras desde función lambda.
Descripción:	El caso de uso muestra la interacción entre la función lambda y las diferentes controladoras de las máquinas de línea a través del broker MQTT.
Actores:	Función lambda, Broker MQTT y controladora de línea.
Precondiciones:	El operario a través de la skill de VoiceToFactory ha enviado una orden a una de las máquinas de producción especifica.
Acciones:	La función lambda interpreta la orden a través del intent enviado por la skill, compone el mensaje de la orden específica y lo publica dentro del broker MQTT en el topic especifico.
Postcondiciones:	La controladora de la maquina destino de la orden consume el mensaje del broker, lo interpreta y ejecuta la acción especifica definida en el mensaje.
Frecuencia de uso:	Cada vez que el usuario decide enviar una orden a las diferentes controladoras de las líneas de producción de las fábricas.

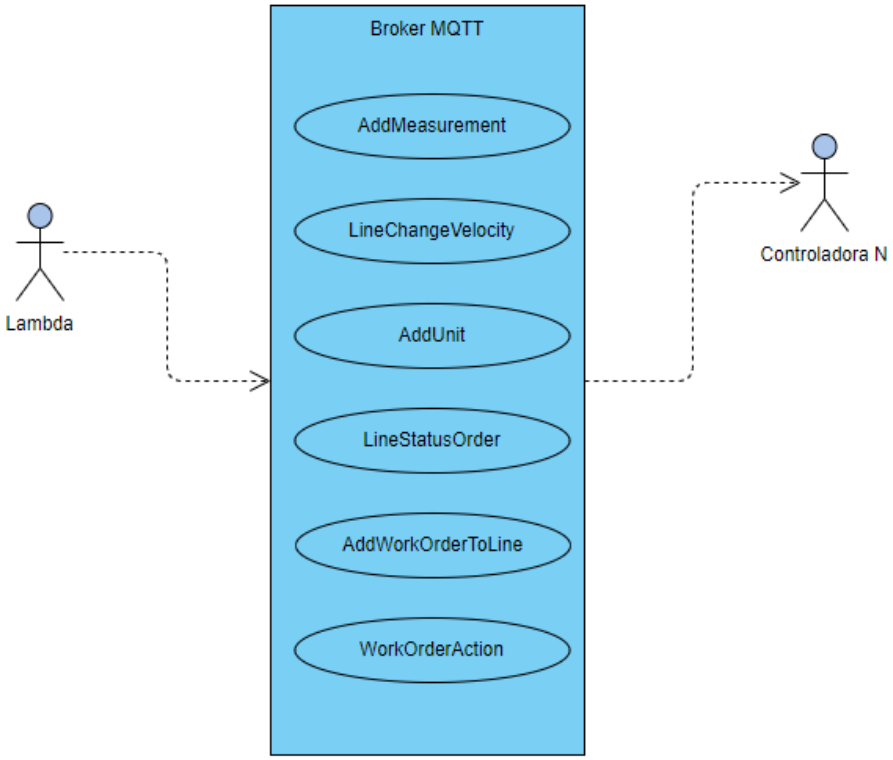


Ilustración 13 Envío de órdenes a controladoras desde función lambda

Identificador:	**4**
Titulo:	Envío reporting desde controladora a la reporting tool
Descripción:	El caso de uso muestra la interacción entre las diferentes controladoras de máquinas de producción con la herramienta de reporting gráfica y el sistema de control auto adaptativo (bucle mape-k)
Actores:	Controladoras máquinas, Herramienta de reporting y bucle de control mape-k
Precondiciones:	Cada controladora, internamente recoge información del estado de esta y de la orden de trabajo que está produciendo si está realizando un proceso de producción activo en ese momento.
Acciones:	La controladora prepara el mensaje de tipo ReportStatus y lo publica en el topic especifico. Tras ellos los diferentes subscriptores consumen ese mensaje y lo procesa para realizar sus acciones especificas
Postcondiciones:	Cada sistema reportado ejecuta acciones internas para la detección de errores imprevistos y/o informar al usuario del estado general del proceso productivo
Frecuencia de uso:	Periódicamente cada x segundos

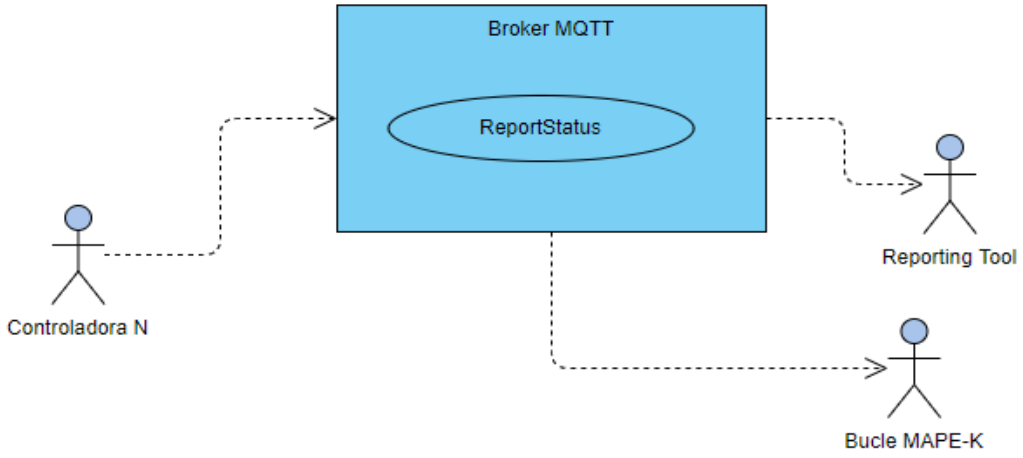


Ilustración 14 Envío reporting desde controladora a la reporting tool

Identificador:	**5**
Titulo:	Envío ordenes desde bucle mape – k a controladora N
Descripción:	El caso de uso muestra la interacción entre sistema de control auto adaptativo (bucle mape-k) y las diferentes controladoras de máquinas de producción cuando el sistema auto adaptativo encuentra alguna situación inconsistente en el sistema productivo.
Actores:	Controladoras máquinas y bucle de control mape-k
Precondiciones:	El sistema auto adaptativo encuentra alguna situación inconsistente en el sistema productivo.
Acciones:	El sistema auto adaptativo una vez ha planificado la acción correctiva, en la fase de ejecución, manda un mensaje a través de broker en el topic a la controladora específica para parar la producción si el problema no es recuperable de manera automática o ajustar el sistema si el sistema se puede auto adaptar al evento inesperado
Postcondiciones:	La controladora de la maquina ejecuta las ordenes planificados por el sistema de bucle de control.
Frecuencia de uso:	Cada vez que se encuentra alguna situación inconsistente en el sistema productivo.

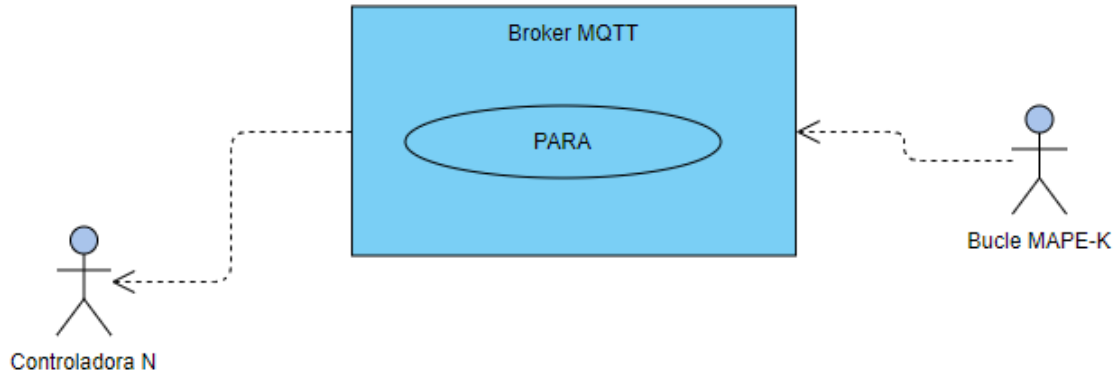


Ilustración 15 Envío ordenes desde bucle mape – k a controladora n

4.2.2 **Módulos principales del sistema**

El siguiente diagrama representa el sistema propuesto dividido en sus diferentes módulos, mostrando las interacciones y las dependencias entre estos componentes. Este diagrama nos ayuda a modelar y documentar la arquitectura del sistema propuesto generando la vista estática y dinámica del sistema como se presentará más adelante en la sección de diseño de la solución. Los principales módulos identificados tras esta fase de análisis de este sistema son los siguiente:

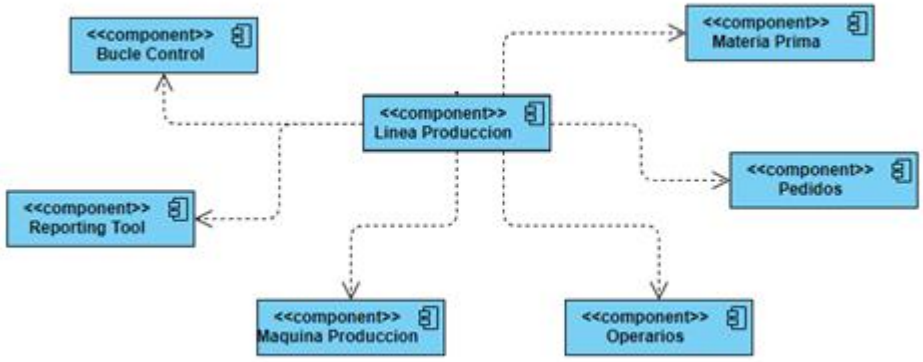


Ilustración 16 Diagrama de componentes del sistema

4.2.3 **Diagrama de clases**

Tras la definición de las diferentes funcionalidades del sistema a través de casos de uso y ya identificados los módulos más importantes del sistema, se pasa a determinar las diferentes características, relaciones, asociaciones y dependencias entre los diferentes subsistemas que la forman. A través del siguiente diagrama de clases se establece el marco conceptual del dominio de la solución, sobre el cual, se centrará el diseño de la solución final propuesta (ver sección 5.4) .

El diagrama de clases propuesto presenta una entidad central llamada línea de producción, la cual agrupa los diferentes elementos del proceso productivo que se han modelado como diferentes clases.

El proceso de fabricación se divide en diferentes tareas de transformación como son el rebobinado de la torta de materia prima y el cortado de la misma en cartoneras según los parámetros definidos en la orden de trabajo la cual se ejecuta en la máquina de producción asociada con la línea de trabajo. Por último, asociado a cada línea de producción, se encuentra asociado uno o varios operarios que controlan y monitorizan el proceso productivo, asociando a cada línea de trabajo con las diferentes ordenes de trabajo y realizando tanto los mantenimiento productivos y preventivos requeridos. Los diferentes componentes que componen la línea de producción se han modelado como diferentes entidades, como se presentan en el siguiente diagrama.

Por último, asociado con la entidad de máquina de producción, se han definido dos entidades diferentes. Por un lado, la entidad Sensor, la cual modeliza los diferentes sensores que posee la máquina de producción como son el estado de la bobina de materia prima, sensor de temperatura o el sensor de presión de corte. Por

otro lado, la entidad SmartThing (controladora inteligente), será el componente encargado de recibir las órdenes del operario a través de la arquitectura de comunicación, ejecutarlas y generar un reporte periódico del estado de la línea de producción para dar trazabilidad al operario asociado con la línea de producción.

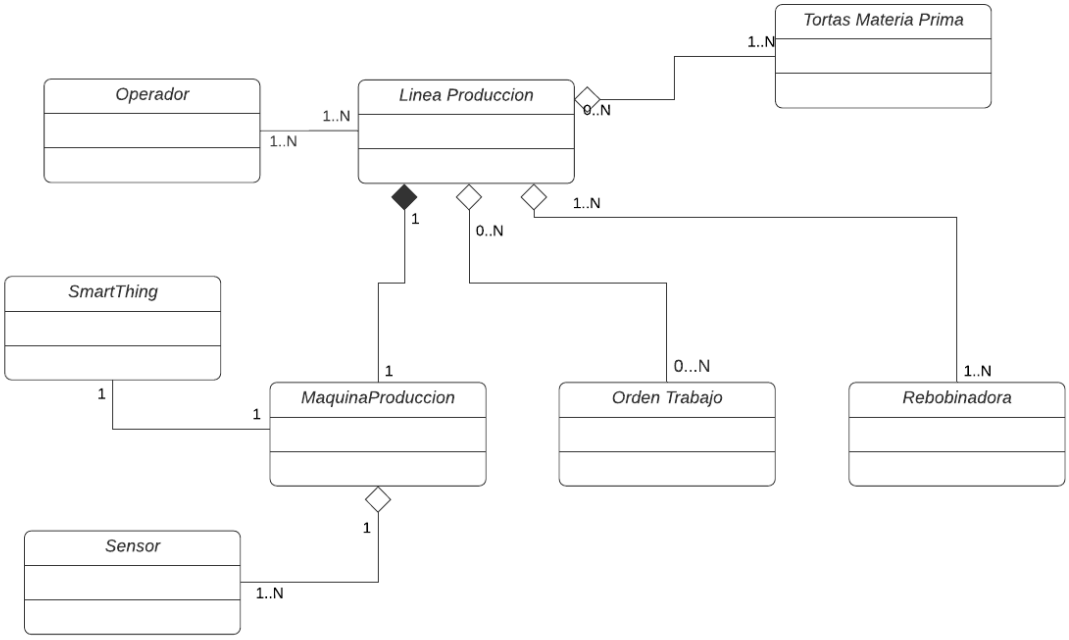


Ilustración 17 Marco conceptual del dominio. Diagrama de clases

4.2.4 **Diagrama de secuencia**

El diagrama de secuencia diseñado muestra como los procesos y los objetos coexisten, interaccionan y en qué orden, dentro de la arquitectura propuesta. También se muestran los diferentes mensajes intercambiados entre ellos y la línea de vida de cada una de las funciones.

El usuario a través de un altavoz inteligente interacciona con la Skill de VoiceToFactory emitiendo a través del lenguaje natural la orden que desea enviar a una o varias controladoras de máquinas de producción. Una vez la skill ha reconocido la orden y validado con el usuario que la orden es correcta, está a través del enlace de ARN que las conecta, construye el objeto de invocación de la función lambda. La función interpreta la orden obteniendo los diferentes parámetros y la función específica a ejecutar. La función lambda construirá un mensaje especifico basado en la información enviada por la skill y lo publicará en el topic específico dentro del broker de comunicaciones del sistema y enviará la skill un objeto de tipo Lambda Response con la información del resultado de la ejecución, la cual, a través del módulo de sinterización de voz del altavoz inteligente, será enviada al usuario que emitió la orden para conocer si la orden se llevó a cabo u ocurrió un error en la misma. Las diferentes

controladoras que están subscritas al topic donde se publican las ordenes reciben el mensaje publicado por la función lambda, interpretan la información y deciden si la orden recibida deben aplicarla o esa orden no aplica para ellas.

Por otro lado, periódicamente las diferentes controladoras de las máquinas de producción publican en el topic de “measurements” un reporte del estado de los diferentes sensores de la maquina como la presión de corte, el estado de la bobina, temperatura interna, etc, junto al estado de la orden de trabajo que está ejecutando. Este mensaje publicado llegara a dos componentes de la arquitectura diferentes, por un lado, a la herramienta de reporting que a través de diferentes interfaces tanto visual como auditiva, y por otro lado a los diferentes componentes de auto adaptabilidad introducidos en la arquitectura. Si alguno de estos escenarios de adaptabilidad detecta algún estado anormal en el sistema, se planificará una acción preventiva o correctiva dependiendo del nivel del estado anormal detectado, tras ello, enviaran a través del broker de comunicaciones un mensaje a la controladora especifica con la información del plan de acción definido. La siguiente imagen muestra de manera visual el flujo de comunicación definido entre los diferentes componentes de la arquitectura.

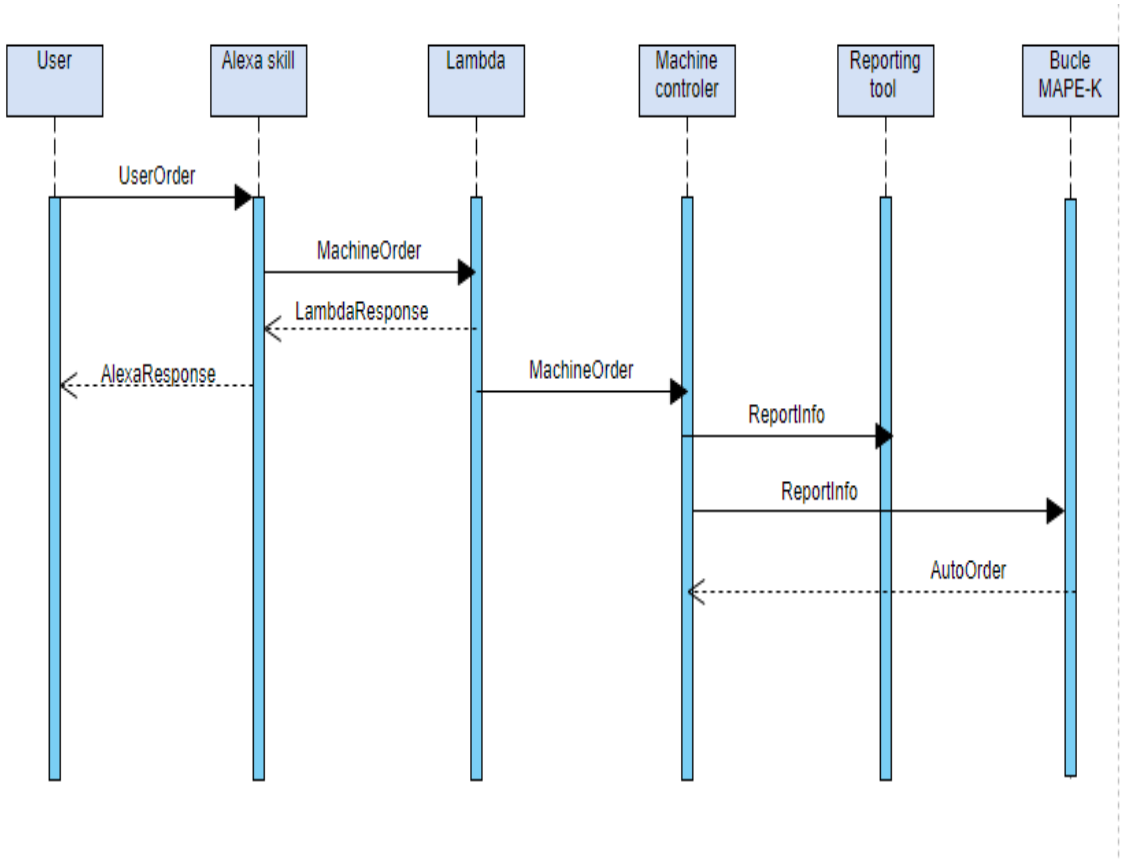


Ilustración 18 Diagrama de secuencia de comunicación del sistema.

5. **Diseño de la solución**

Este capítulo parte de un análisis profundo del problema, siendo su objetivo la realización del diseño de la solución para posteriormente determinar todos los elementos necesarios para realizar la implementación de la solución.

5.1 **¿Qué arquitectura software utilizar?**

Uno de los aspectos más importantes para llevar a cabo una implementación en todo proyecto software, es el de utilizar una arquitectura software adecuada para desarrollar el proyecto. La arquitectura de software es la organización fundamental de un sistema encarnado en sus componentes, las relaciones entre ellos y el ambiente, además de los principios que orientan su diseño y evolución.

Existen diferentes modelos de arquitectura software que pueden aplicarse a diferentes tipos de soluciones softwares. Las más importantes son las siguientes:

- **Arquitectura en capas**: Este patrón se puede usar para estructurar programas que se pueden descomponer en grupos de subtareas, cada uno de los cuales se encuentra en un nivel particular de abstracción. Cada capa proporciona servicios a la siguiente capa superior.

- **Arquitectura cliente/servidor:** Este patrón consta de dos partes; Un servidor y múltiples clientes. El componente del servidor proporcionará servicios a múltiples componentes del cliente. Los clientes solicitan servicios del servidor y el servidor proporciona servicios relevantes a esos clientes. Además, el servidor continúa escuchando las solicitudes de los clientes.

- **Arquitectura Maestro/Esclavo:** Este patrón consta de dos partes, maestro y esclavos. El componente maestro distribuye el trabajo entre componentes esclavos idénticos y calcula un resultado final a partir de los resultados que devuelven los esclavos.

- **Arquitectura Pipeline:** Este patrón puede usarse para estructurar sistemas que producen y procesan un flujo de datos. Cada paso de procesamiento está encerrado dentro de un componente de filtro. Los datos para procesar se pasan a través de tuberías. Estas tuberías se pueden usar para almacenamiento en búfer o para fines de sincronización.

- **Arquitectura P2P:** En este patrón, los componentes individuales se conocen como pares. Los pares pueden funcionar como un cliente, solicitando servicios de otros pares y como un servidor, proporcionando servicios a otros pares. Un par puede actuar como cliente o como servidor o como ambos, y puede cambiar su función dinámicamente con el tiempo.

- **Arquitectura Modelo-Vista-Controlador:** Este patrón, también conocido como patrón MVC, divide una aplicación interactiva en 3 partes, esto se hace para separar las representaciones internas de información de las formas en que la información se presenta y se acepta del usuario. Desacopla los componentes y permite la reutilización eficiente del código.
    - Modelo: contiene la funcionalidad y los datos principales

- Vista: muestra la información al usuario (se puede definir más de una vista)
- Controlador - maneja la entrada del usuario

- **Arquitectura Broker:** Este patrón se utiliza para estructurar sistemas distribuidos con componentes desacoplados. Estos componentes pueden interactuar entre sí mediante invocaciones de servicios remotos. Un componente de intermediario es responsable de la coordinación de la comunicación entre los componentes. Los servidores publican sus capacidades (servicios y características) en un corredor. Los clientes solicitan un servicio del intermediario, y el intermediario redirige al cliente a un servicio adecuado desde su registro.

Para la implementación de la solución se ha decidido utilizar una arquitectura tipo bróker para la comunicación desacoplada entre los diferentes componentes de la solución. Se utilizan diferentes "topics" para la publicación de los mensajes en el bróker y los clientes consumen los mensajes de las colas asociadas.

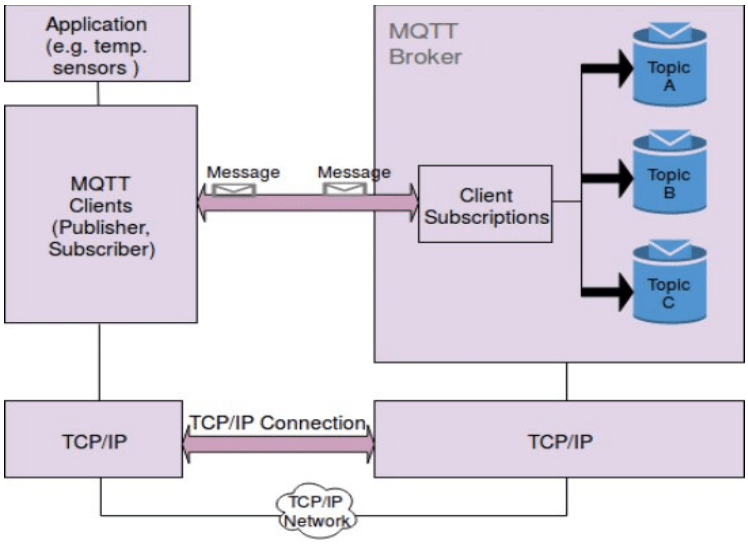


Ilustración 19 Arquitectura Broker MQTT (Soni, 2017)

5.2 ¿Comunicación síncrona o asíncrona?

Con la arquitectura software seleccionada, la siguiente pregunta que se plantea en el diseño de la solución es que tipo de comunicación usar, síncrona o asíncrona. Cuando se habla de comunicación sincrónica y asincrónica se hace referencia a dos formas diferentes de intercambio de la información en función de la simultaneidad con la que se envía y ofrece el un determinado mensaje.

La principal diferencia entre ambos patrones de comunicación se basa en que mientras que la comunicación asíncrona no espera una respuesta del receptor, la ejecución síncrona necesita que ambas partes comuniquen simultáneamente, por lo que la simultaneidad en la comunicación es la clave principal de distinción entre ambos tipos de comunicación. Aunque esta es la principal distinción entre ambas existen otro tipo de diferencias que se presenta a continuación:

- En la comunicación sincrónica, los datos se transfieren en forma de tramas, mientras que, en la asincrónica, los datos se envían de un byte en un byte.
- La transmisión sincrónica necesita una señal de reloj entre el emisor y el receptor para informar al segundo sobre la llegada del nuevo byte o mensaje. En cambio, en la asincrónica, no se requiere esta señal de reloj externa, puesto que los datos se sincronizan a través de señales, que indican el inicio del nuevo byte o mensaje.
- La velocidad de transferencia de la comunicación asincrónica es más lenta que la de la transmisión sincrónica.
- Por el contrario, la transmisión asincrónica es más simple y económica que la sincrónica.
- La comunicación sincrónica es más eficiente y tiene una sobrecarga menor, en comparación con la asincrónica.

Tras el análisis de ambas posibilidades y como encajan estas con el modelo de arquitectura software broker y con una solución definida sobre una infraestructura IoT, se considera que la asincronía basada en eventos como un marco de comunicación, encaja perfectamente con entornos IoT.

5.3 Protocolos de comunicación asíncrona, ¿Cuál utilizar?

Una vez seleccionada la comunicación asíncrona como estrategia de comunicación de la solución, se ha de decidir el protocolo de comunicación asíncrono a utilizar en los diferentes procesos comunicativos de la solución.

Con el auge del IoT ha sido necesario el desarrollo de protocolos de comunicación que se adecúen a las necesidades de este tipo de aplicaciones. Dichos protocolos permitirán enviar a través de Internet la información recogida por los sensores o equipos de una forma segura, simple y eficazmente. Muchos de estos protocolos son protocolos basados en envío de mensajes, como los utilizados en aplicaciones de mensajería instantánea. Entre ellos destacan dos los protocolos entre los demás

- **MQTT**: es un protocolo de mensajería abierto, simple y ligero, basado en publicaciones y subscripciones a mensajes en formato JSON, entre un cliente y un broker. El protocolo trabaja sobre TCP/IP, pero existen otras variantes como MQTT-SN que funcionan sobre redes no TCP/IP. Este protocolo en particular ha sido diseñado para localizaciones remotas con ancho de banda limitado.
- **CoAP** (Constrained Aplication Protocol): es un protocolo de la capa de aplicación para la utilización en dispositivos con recursos limitados, tales como las redes de sensores. CoAP está diseñado para que pueda ser traducido fácilmente a HTTP para simplificar su integración en la web, y también cumple con requisitos como capacidad de multicast, cabeceras muy pequeñas y simplicidad

Ambas soluciones proveen mecanismos asíncronos de comunicación, trabajan sobre IP y son más adecuados para entornos con recursos limitados como son los escenarios de infraestructura IoT. En cuanto a la compatibilidad con las otras tecnologías utilizadas, la alternativa MQTT cuenta con ciertas ventajas frente a CoAP, ya que MQTT implementa una solución para recoger datos mediante la subscripción a

un bróker MQTT. Mientras que con CoAP haría falta adquirir estos datos con alguna solución alternativa y convertirlos para poder enviarlos mediante CoAP.

Tras este análisis de estos dos protocoles de comunicación se ha decidido que MQTT es una solución más adecuada para el desarrollo propuesto en este trabajo, ya que proporciona una mayor facilidad para integrarla con otras de las alternativas tecnológicas utilizadas, ofrece una complejidad de implementación y mantenimiento adecuada.

5.4 **Arquitectura general del sistema**

La Ilustración 20 presenta una visión general de la arquitectura final definida al acabar la fase de análisis y diseño de la solución. La arquitectura del proyecto "SmartIoT4Industry" tiene como base principal una arquitectura IoT tradicional junto a la fusión de disciplinas en pleno auge como la introducción de interfaces multimodales, el uso del lenguaje natural para una comunicación humano-máquina más natural y efectiva y la introducción de los paradigmas de computación ubicua y autónoma para lograr dotar a la empresa 4.0 del futuro de un marco teórico practico autónomo el cual permitirá que las empresas que implante este tipo de sistemas para pasar desde "el proceso de automatización industrial" a la "industria IoT autónoma".

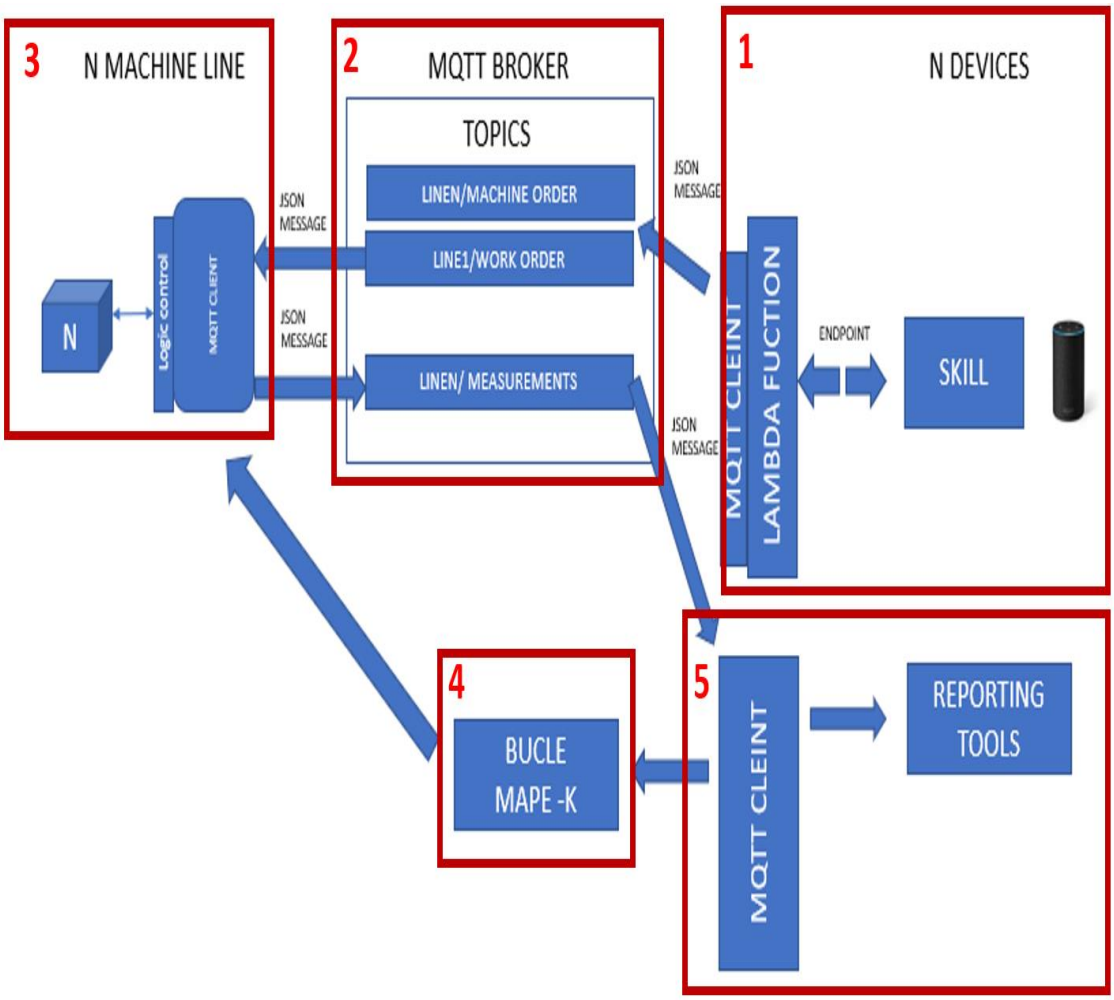


Ilustración 20 Arquitectura general del sistema

La arquitectura definida se ha subdivido en cinco subsistemas identificados numéricamente en la imagen anterior. A continuación, se describen en orden numérico las características principales de los subsistemas definidos:

- ***Comunicación humano-sistema a través de interfaces del lenguaje natural***: El principal objetivo de esta parte de la arquitectura es lograr una comunicación de ordenes fluida desde los diferentes operarios y las máquinas de producción de la fábrica. A través la skill se intenta simular un comportamiento lingüístico humano y, para ello, se ha de tener muy presente las estructuras propias del lenguaje y las normas generales del discurso, dos elementos básicos por los que los humanos que participan en un diálogo son capaces de combinar las palabras para formar una oración, conocer los significados de estas, deducir cómo afecta cada una al significado total de la oración y tener un conocimiento del mundo general. Otra de las características de la aplicación de PLN sobre el sistema es el procesamiento de los mensajes de voz humana, transformándolos en texto, interpretando y comprendiendo la intencionalidad de los mismos (Sidorov, 2016), y tras la generación de dicho texto, se envía las órdenes a la función lambda, la cual interpreta esta ordenes, obtiene la información necesaria enviada por el usuario y la traduce a mensajes que se envían a través del broker para hacerla llegar a los distintos clientes subscritos al topic especifico, en este caso, las diferentes controladoras de las máquinas de la fábrica. La plataforma proveerá al usuario final de un sistema diálogo multimodal que le permita la interacción con un entorno de Inteligencia Ambiental para una industria a través del habla, tomando como referencia las actuales soluciones tecnológicas de interacción por voz como son 'Ok Google', Alexa o Siri, y sus dispositivos específicos asociados.

- ***Comunicación entre los diferentes componentes de la solución a través de una arquitectura de comunicación de broker:*** Tras el análisis de las diferentes arquitecturas de comunicación introducidas en el capítulo 5.1, se ha determinado que una arquitectura tipo bróker para la comunicación desacoplada entre los diferentes componentes del sistema es la solución óptima para los diferentes procesos comunicativos definidos en el capítulo anterior, siguiendo una topología tipo estrella, donde existe un nodo central o broker con capacidad para trabajar con un gran número de clientes. El broker de tipo MQTT nos ofrece protocolo de comunicación M2M (machine-to-machine) donde los dispositivos se comunican entre sí utilizando un patrón publicación/suscripción. Dentro de la arquitectura definida, es muy importante el concepto "topic" ya que a través de estos "topics" se articula la comunicación puesto que emisores y receptores deben estar subscritos a un "topic" común para poder entablar la comunicación. Como se muestra en la figura anterior, los topics definidos son dinámicos y tienen una estructura jerárquica gracias a la cual podemos establecer relaciones padre-hijo y, al subscribirnos a un topic padre, recibir también la información de sus hijos. La siguiente figura describe la estructura de topics definida en el broker de comunicaciones.

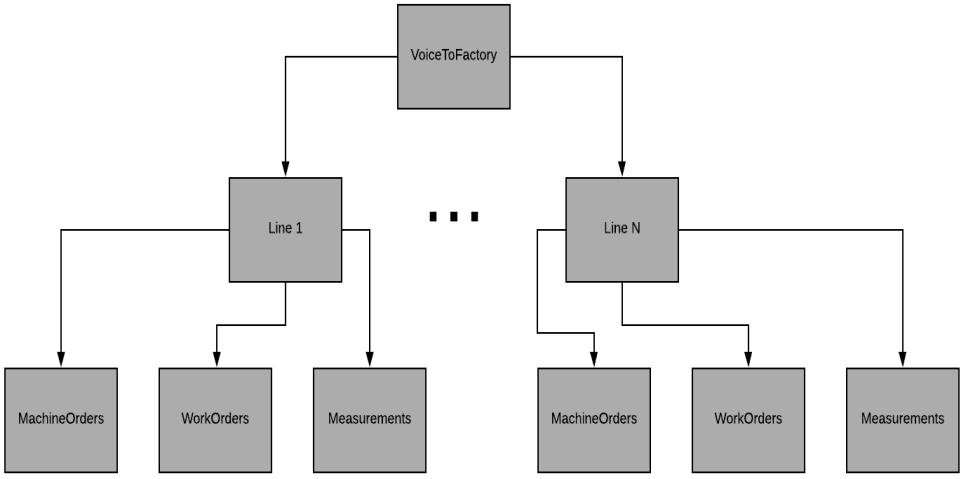


Ilustración 21 Estructura jerárquica de topics definidos

- ***Controladoras de máquinas de producción que son capaces de interactuar de manera transparente con el sistema de manera bidireccional:*** Uno de los requerimientos más importantes de la solución propuesta, es la comunicación de las ordenes desde el usuario a las diferentes máquinas de producción de la empresa y por otro lado el reporte periódico del estado de dichas máquinas, que permitirán el seguimiento del estado en tiempo real del proceso productivo. Normalmente el dispositivo de control de una maquina suelen ser un controlador lógico programable, más conocido como PLC (Programmable Logic Controller, debido a sus siglas en inglés). Este es básicamente una computadora que se utiliza en la ingeniería de automatización para las industrias, es decir, para el control de la maquinaria de una fábrica o de situaciones mecánicas. Para la integración de este tipo de controladores en los sistemas como el propuesto, se necesitan requisitos innovadores en la configuración en relación con la infraestructura de base, lo que impulsa aún más la fusión del nivel de IT y de automatización. En la arquitectura definida, se pretende ampliar las funcionalidades de estos controladores programables para que intervengan activamente en el proceso de comunicación como un actor más recibiendo las órdenes, interpretándolas y aplicando los cambios que el usuario emite a través de la skill o las ordenes de reconfiguración del bucle de control y por otro lado aportando periódicamente información relevante acerca del estado de la maquina y de la orden de trabajo que está procesando en ese momento. Todo ello interaccionando con el broker de comunicaciones para ciertos escenarios con el rol de subscriptor y para otros con el de publicador.

- ***Componentes de auto adaptabilidad y control:*** La introducción de diferentes disciplinas emergentes orientadas a la auto adaptabilidad, la inteligencia ambiental y la computación dentro de la arquitectura propuesta, acerca a la industria cada vez más al concepto de “smart factory” que representa la adaptabilidad de la industria al contexto volviéndose más autónoma flexible e interconectada . A través de este tipo de sistemas de monitorización, las nuevas fábricas serán capaces de evolucionar y optimizarse por sí mismas, estando en constante aprendizaje y analizan cada dato que los sensores captan para tomar las mejores decisiones dotando gracias a ello de adaptabilidad a las condiciones cambiantes de la propia fábrica, otorgando una

flexibilidad que mejora el rendimiento de los procesos y ayuda a reducir costes. Actualmente existen sistemas de monitorización ya implementados en la industria, pero lo que realmente crea valor añadido en la implantación de este tipo de arquitecturas de comunicación como la definida en este documento, es el análisis de los datos recogidos. De nada sirve tener toda la información del mundo si no se es capaz de conseguir entenderla y sacarle rendimiento de manera autónoma, por eso es por lo que esta arquitectura de comunicación junto a los sistemas auto adaptativos y las personas encargadas de la interpretación del análisis de datos serán los actores más importantes dentro de la implantación de este sistema en la futura "industria IoT autónoma".

- ***Herramientas de reporting a través de interfaces multimodales***: En el sistema propuesto, capaz de una gestión del proceso productivo cada vez más autónomo y descentralizado, la gestión del rendimiento y monitorización del estado del proceso productivo, en un mundo de Industria 4.0 será exponencialmente más compleja de lo que es hoy en día. Este problema de falta de visibilidad es abordado en la arquitectura propuesta con la introducción de una plataforma de visibilidad a través de interfaces multimodales que proporcione una visión de extremo a extremo de las interacciones de los usuarios con no sólo una aplicación, sino toda la infraestructura que soporta esa aplicación (Álvaro Hernández). Esto permite al equipo de operarios de la empresa aprovechar las herramientas de diagnóstico de manera más eficaz para reducir tiempos de mantenimiento tanto preventivos como correctivos y a su vez tener una visión general de que ordenes de trabajo se están produciendo en cada una de las máquinas del proceso productivo.

6. **Implementación del prototipo de validación de la arquitectura diseñada**

Este capítulo se centra en detallar los pasos seguidos para la implementación de la arquitectura software del proyecto presentada en la sección 5.4 y que se ha descrito a lo largo de los capítulos anteriores.

6.1 **Selección del lenguaje de programación**

Primero de todo, antes de iniciar el desarrollo del prototipo de validación se ha de seleccionar un lenguaje de programación que encaje con los siguientes criterios:

- Funciona en Raspberry Pi para los controladores de máquinas de línea
- Compatibilidad con protocolos IoT
- Soporte de AWS SDK
- Soporte MQTT
- Entorno de diseño gráfico para la implementación de interfaces de usuario visuales

Los lenguajes de programación candidatos han sido:

- **Go**: Es un proyecto de código abierto desarrollado por un equipo de Google y muchos colaboradores de comunidad de open source. Se considera Go principalmente porque es rápido, tiene un gran soporte para subprocesos múltiples y sintaxis fácil de leer. Go es un lenguaje compilado, lo que significa que se ejecuta directamente en el subyacente hardware. Go fue construido con multihilo en mente. Tiene “gorutinas” en lugar de hilos, que proporciona una gestión de memoria flexible y evita tener que recurrir al bloqueo de mutex al compartir estructuras de datos. Go también tiene una biblioteca estándar fuerte para el desarrollo

- **Node.js**: Es un entorno de tiempo de ejecución JavaScript multiplataforma de código abierto que ejecuta JavaScript código fuera de un navegador. Node.js fue considerado principalmente debido a su gran soporte para integración API de desarrollo de aplicaciones, y está bien equipado para manejar una gran cantidad de solicitudes a través de flujos. Se instala fácilmente, por ejemplo, en una Raspberry Pi y tiene un gran soporte para sockets y MQTT

- **Python**: Es un lenguaje de programación dinámico de alto nivel, interpretado y de propósito general que se enfoca en la legibilidad del código. Se considera Python principalmente por su facilidad de uso, legibilidad y excelente soporte con los marcos web. La sintaxis en Python ayuda a los programadores a codificar en menos pasos en comparación con otros idiomas.

- **C / C ++**: Funciona muy bien para aplicaciones que se ejecutan en dispositivos donde la potencia informática suele ser bastante limitado, como un Raspberry Pi. C funciona bien debido a su acceso de bajo nivel a la memoria de la computadora y no requiere mucha potencia de procesamiento. C ++ se deriva directamente de la C lenguaje, lo que significa que comparte muchas propiedades con C al tiempo que agrega mejoras y soporte para la

programación orientada a objetos. Tanto C como C ++ tienen poco soporte para frameworks web, lo que hace Desarrollo de una API muy desafiante.

- .**Net Core**: es uno de los lenguajes de programación más utilizados y una opción destacada para los ingenieros de IoT. Ha sido la columna vertebral de muchas tecnologías emergentes de IoT. Una de las razones principales para elegir .Net sería su flexibilidad, versatilidad y su capacidad de ejecución multiplataforma.

El lenguaje seleccionado para la implementación del proyecto prototipo ha sido .NET Core ya que cumple con los requisitos mínimos evaluados y además me es un lenguaje de programación familiar ya que trabajo con él diariamente.

6.2 Interacción usuario con el sistema de control de la fabrica

Partiendo de la subdivisión de la arquitectura presentada en el capítulo anterior concretamente en la parte de la interacción vocal entre los usuarios de planta con la función lambda a través de un altavoz inteligente y una se ha diseñado y desarrollado una Skill de Alexa (ver sección 2.3.4), la cual permite al usuario, utilizando para ello el lenguaje natural, interaccionar con las diferentes máquinas de las líneas de producción de una manera intuitiva y sencilla a través de pequeños comandos de voz predefinidos en la estructura de dialogo de la skill desarrollada.

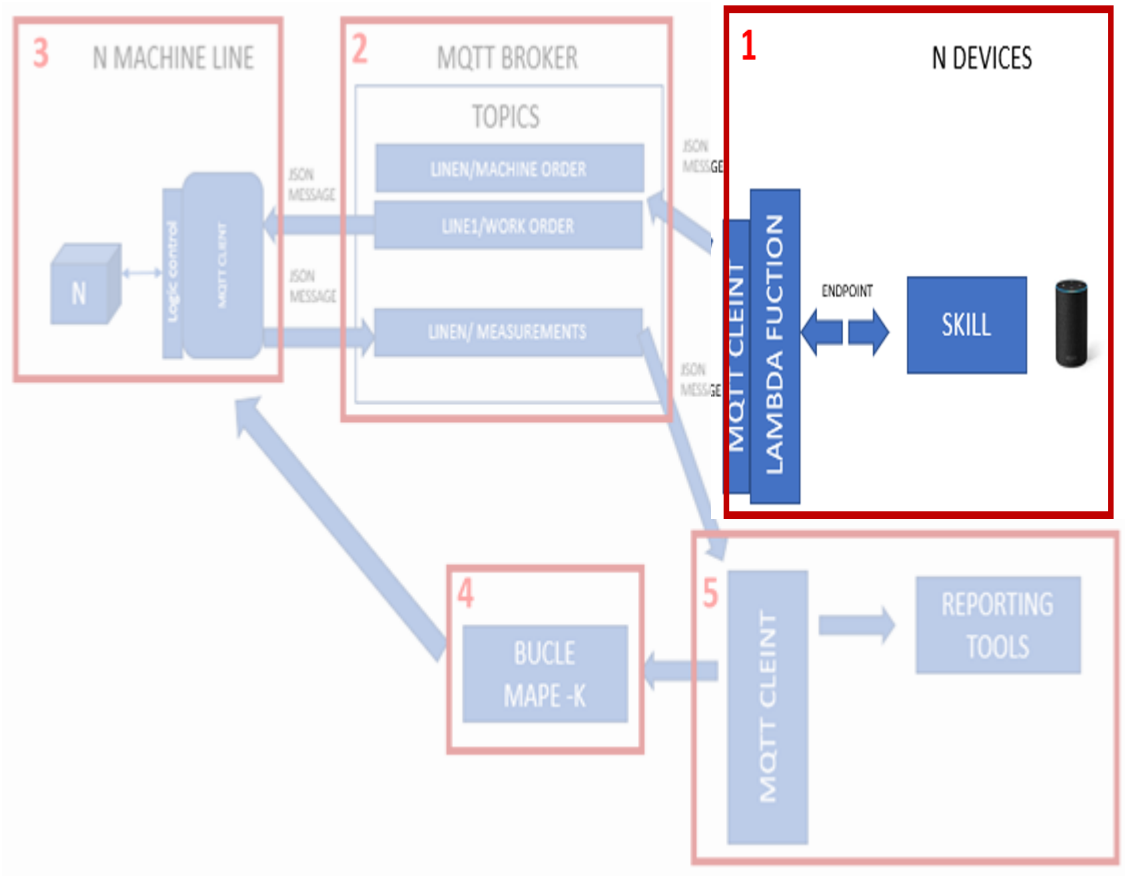


Ilustración 22 Arquitectura general del sistema. Sección Interacción Vocal

Para poder interactuar con la funcionalidad de la skill, el operario de la línea de producción simplemente debe invocar la skill a través del “Skill Invocation Name” usando para ello el comando de voz “Alexa inicia Voice To Factory”. Esto envía al servicio de la skill un mensaje de tipo LaunchRequest, el cual hará que el dispositivo Amazon Echo a partir de ese momento cargue la funcionalidad y la exponga la funcionalidad definida en el caso de 4.2.1

Para poder facilitar la interacción vocal entre el usuario y la skill, se han definido y agregar tipos propios de valores personalizados a un subconjunto de los tipos de datos ya predefinidos dentro del SDK de la skill denominados “SlotsType”.

Los tipos de valores creados para esta solución son los siguientes:

-
  - Arranca
  - Pausa
  - Reanuda
  - Termina
- Líneas de producción
  - A
  - B
  - C
  - D

- Medicion sobre orden de trabajo
  - Estado del Corte
  - Velocidad Media de producción
  - Temperatura
  - Estado de la bobina

El siguiente ejemplo muestra un fragmento de código de la skill donde se define un “slotType” concretamente el de acción sobre orden de trabajo.

```
"types": [
    {
        "name": "Accion",
        "values": [
            {
                "name": {
                    "value": "Arranca"
                }
            },
            {
                "name": {
                    "value": "Pausa"
                }
            },
            {
                "name": {
                    "value": "Reanuda"
                }
            },
            {
                "name": {
                    "value": "Termina"
                }
            }
        ]
    },
```

Ilustración 23 Definición de slotType acción sobre orden de trabajo

Tras la definición de los SlotsType específicos para la implementación de la solución, se han definidos un conjunto de intents. Los intents definidos dentro de la skill son las acciones que se le permiten al usuario al interaccionar con ella. Estos intents se componen internamente de un nombre y de uno o varios slots que sirven para introducir valores dinámicos en las órdenes del usuario. El tipo de valores de estos slots pueden ser los genéricos del SDK o los personalizados como los

introducidos en el párrafo anterior. Para la skill “Voice To Factory” se han definidos los siguientes intents y slots internos:

- Amazon.CancelIntent
- Amazon.HelpIntent
- Amazon.StopIntent
- Amazon.NavigateHomeIntent
- Arranca la producción de la línea de trabajo
    - Line (Línea de producción)
- Para la producción de la línea de trabajo
    - Line(Línea de producción)
- Aumenta la velocidad de producción de la línea de trabajo
    - Line (Línea de producción)
    - Velocidad (Valor numérico)
- Disminuye la velocidad de producción de la línea de trabajo
    - Line (Línea de producción)
    - Velocidad (Valor numérico)
- Acción sobre orden de trabajo
    - OrdenTrabajo (Valor numérico)
    - Acción (Acción sobre orden de trabajo)
    - Line (Línea de producción)
- Añade Unidades a la orden de trabajo
    - OrdenTrabajo (Valor numérico)
    - Unidades (Valor numérico)
- Añade Medicion a la orden de trabajo
    - OrdenTrabajo (Valor numérico)
- Medicion (Medicion sobre orden de trabajo)

E

l siguiente ejemplo muestra un fragmento de código de la skill donde se define un “Intent” concretamente el de acción sobre orden de trabajo.

```
{
    "name": "AccionSobreOrdenTrabajo",
    "confirmationRequired": false,
    "prompts": {},
    "slots": [
        {
            "name": "OrdenTrabajo",
            "type": "AMAZON.NUMBER",
            "confirmationRequired": false,
            "elicitationRequired": false,
            "prompts": {}
        },
        {
            "name": "Accion",
            "type": "Accion",
            "confirmationRequired": false,
            "elicitationRequired": false,
            "prompts": {}
        },
        {
            "name": "Line",
            "type": "Line",
            "confirmationRequired": false,
            "elicitationRequired": false,
            "prompts": {}
        }
    ]
}
```

Ilustración 24 Definición de Intent: Acción sobre orden de trabajo

Por último, una vez definidos los intents y los tipos de datos personalizados se ha definido para enriquecer más la comunicación humano-maquina directivas de tipo “dialog” que permiten gestionar una conversación de múltiples turnos entre la skill y el usuario para permitir al usuario validar una orden asociada a algunos de los intents anteriores o recibir información extra. El siguiente ejemplo muestra un fragmento de código de la skill donde se define un “Intent” concretamente el de acción sobre orden de trabajo.

```
"prompts": [
    {
        "id": "Elicit.Intent-Arranca.IntentSlot-Line",
        "variations": [
            {
                "type": "PlainText",
                "value": "¿Que linea desea que empiece a trabajar?"
            }
        ]
    },
    {
        "id": "Confirm.Intent-Arranca.IntentSlot-Line",
        "variations": [
            {
                "type": "PlainText",
                "value": "¿Desea empezar trabajo en la linea {Line}?"
            }
        ]
    },
```

Ilustración 25 Fragmento de Dialog skill Elicit y Confirm

El operario a través del dispositivo echo ha emitido una orden sobre una de las máquinas de la línea de producción. El dispositivo ha procesado las palabras del operario y las transcribe a lenguaje máquina, concretamente en formato JSON. A través del nombre de la invocación, Alexa sabe a qué punto extremo vincular y qué expresiones y espacios debe enviar al punto final. En nuestro caso, el punto final es una función Lambda que ahora procesará la solicitud. Este proceso esta resumido en la siguiente ilustración.

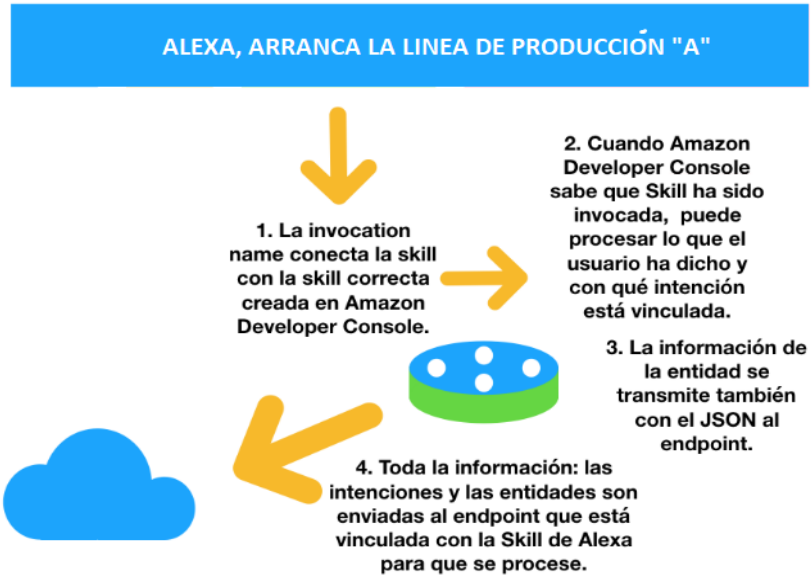


Ilustración 26 Comunicación humano - función lambda. Skill Voice To Factory

La función lambda (ver sección 2.3.2) es la encargada de recibir las ordenes desde la skill desarrollada y traducirlas a órdenes a las distintas controladoras de las máquinas de producción. Como se muestra en la subsección 2 de la Ilustración 20 Arquitectura general del sistema, esta comunicación se lleva a cabo a través de los diferentes topics definidos dentro del broker MQTT utilizando para ello una librería de comunicación que se presentara en el capítulo 6.3.La función lambda se ha implementado en el lenguaje de programación .Net Core 2.1 y una vez implementado se ha desplegado en la plataforma de Amazon AWS. Como base para el desarrollo se ha usado el SDK que nos ofrece Alexa para así poder recibir las peticiones de la skill y enviar respuestas completas a todos los dispositivos que la estén usando dentro de la fábrica.

El código desarrollado y cargado en la plataforma serverless de AWS Lambda se ejecuta como una función única. Existen dos palabras clave asociadas con la función Lambda, "Handler" y " LambdaContext ". Cuando se invoca la función lambda, se genera un objecto de tipo LambdaContext (este objeto proporciona métodos y propiedades que facilitan información acerca de la invocación, la función y el entorno de ejecución). Dicho objeto se pasa al método “Handler” definido en la clase principal Function.cs. El método “Handler” es el método de la función Lambda que procesa eventos. Cuando invoca una función, el motor de ejecución ejecuta el método del “Handler”. La cabecera de la función del manejador, que es el punto de entrada tras la invocación de la skill tiene la siguiente estructura:

public SkillResponse Handler(SkillRequest input, ILambdaContext ctx)

La función para Alexa acepta un objecto de tipo SkillRequest y devuelve un SkillResponse. La función lambda obtiene del objeto SkillRequest el “intent” que el usuario final ha utilizado para mandar la orden deseada a la maquina a través de la solución propuesta. Los diferentes posibles “intent” han sido definidos en la skill del apartado anterior. Una vez determinada la intent que el usuario desea ejecutar, la función lambda analiza los diferentes slots para construir el mensaje que se enviara a la controladora específica a través del broker MQTT. Un ejemplo de cómo obtener la información desde el objecto SkillRequest se muestra a continuación:

```
private string SetMachineVelocity(string action, LineData data, SkillRequest request)
{
    try
    {
        if (request.Intent.Slots.ContainsKey("Line") && request.Intent.Slots.ContainsKey("Velocidad"))
        {
            if (request.Intent.Slots.TryGetValue("Line", out var slot)
                && request.Intent.Slots.TryGetValue("Velocidad", out var velocidad)
                && int.TryParse(velocidad.Value, out var vel))
            {
                if (_sender.SendLineVelocityMessage(slot.Value.ToLower(), vel, action))
                {
                    return $"La velocidad de la linea {slot.Value} se ha {action} en {vel} unidades " + factdata.AskMessage;
                }
                return InvalidSlotMessage(_lineData);
            }
        }
        return string.Join(";", request.Intent.Slots.Select(x => x.Key + "=" + x.Value.Value).ToArray());
    }
    catch (Exception e)
    {
        return e.Message;
    }
}
```

Ilustración 27 Intents y slots asociados al objecto SkillRequest

La arquitectura final de interacción entre el operario de la fábrica y la función lambda la cual interpreta las ordenes captadas a través de la skill y la distribuye a los objetos específicos del sistema productivo se muestra en la siguiente imagen:

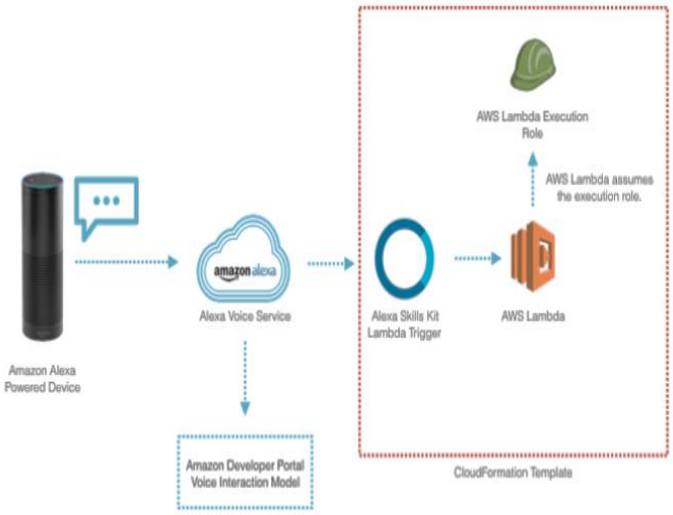


Ilustración 28 Arquitectura comunicación desde skill a funcion lambda

6.3 **Broker de comunicaciones**

Para llevar a cabo la comunicación entre la función lambda y las diferentes controladoras de máquinas de producción de la fábrica, dentro de la arquitectura del sistema (ver subsección 2 de la Ilustración 20) se ha definido una comunicación asíncrona a través de un broker de comunicaciones MQTT, concretamente para la implementación de este prototipo se ha elegido el broker que nos ofrece la plataforma AWS IoT.

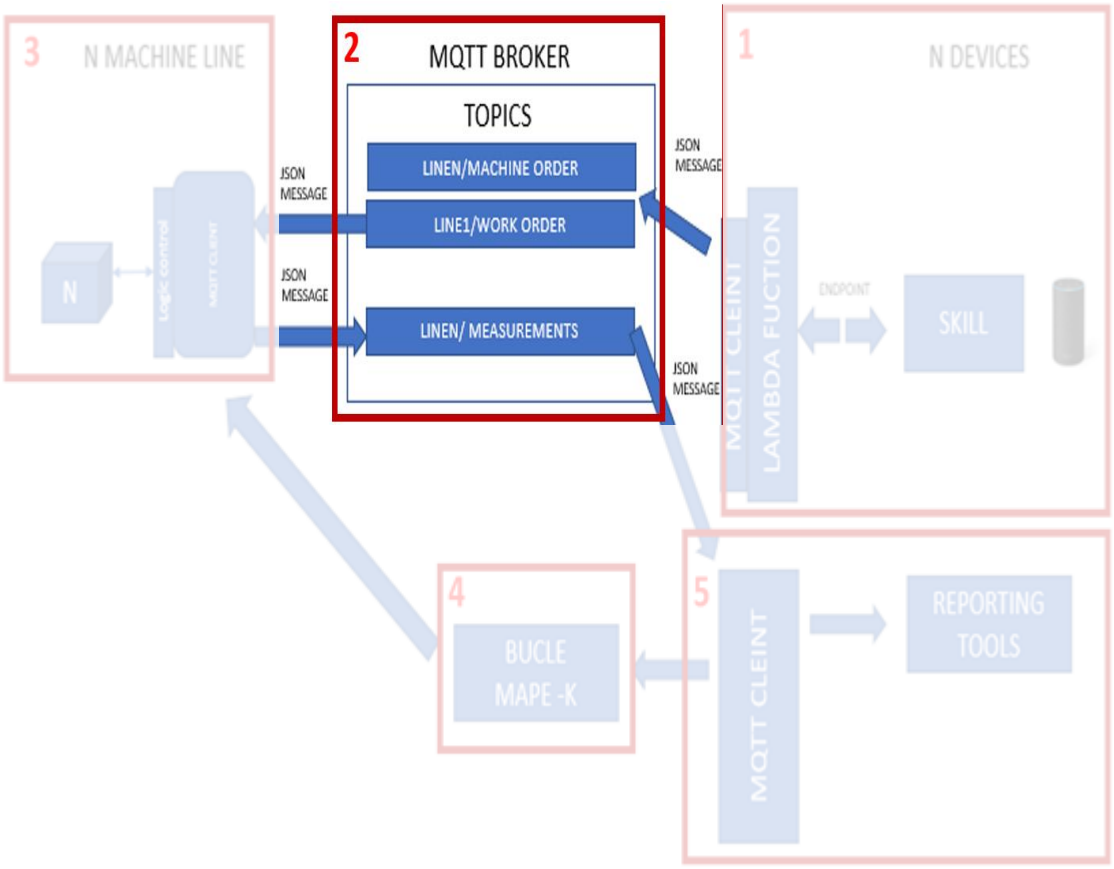


Ilustración 29 Arquitectura general del sistema. Sección broker comunicaciones

El procesamiento de mensajes a través del agente de mensajes del broker MQTT permite definir reglas que pueden iniciar más acciones en función del contenido de los mensajes. Los nombres de tema y los filtros de tema son cadenas codificadas UTF-8. Pueden representar una jerarquía de información utilizando el carácter de barra diagonal (/) para separar los niveles de la jerarquía. Gracias a esta funcionalidad de topics jerarquizados, se dota al sistema de adaptabilidad al crecimiento si se desea añadir más máquinas de producción sin tener que alterar ningún componente de la arquitectura implementada. Los topics definidos para la comunicación entre los diferentes actores de la solución que permiten el intercambio total de mensajes en todos los escenarios de comunicación diseñados son los siguientes:

- ***Line{line}/machineOrder:*** En este topic se publicarán todos los mensajes relativos a acciones sobre la línea de trabajo como iniciarla, pararla, aumentar o disminuir la velocidad de producción, etc

- ***workorder/workorder:*** En esté topic se publicarán todos los mensajes relativos a las órdenes de trabajo, como son, añadir orden de trabajo a línea de producción, modificar unidades de la orden de trabajo, iniciar, pausar reanudar o terminar la producción de una orden de trabajo dentro de una línea de producción.

- ***Line{line}/Measurements:*** En esté topic se publicarán periódicamente mensajes de reporting con las mediciones llevabas a cabo por cada una de las líneas de trabajo a través de los diferentes sensores de estás. La información recogida es relativa al estado de corte, la temperatura interna de la máquina, nivel del sensor de la bobina, etc junto al estado de la orden de trabajo que se esta ejecutando en la máquina que reporta.

Para la implementación del prototipo se ha utilizado la plataforma Amazon AWS. Para poder dar de alta un bróker MQTT dentro de la infraestructura de AWS IoT, se debe de dar de alta un objecto (“thing”). Un objeto es la representación y el registro de su dispositivo físico en la nube. Cualquier dispositivo físico necesita un registro de objeto para poder funcionar con AWS IoT.

A través de la consola de administración de Amazon AWS se ha creado un objecto o “thing” llamada 'dotnetdevice' como muestra la siguiente imagen.

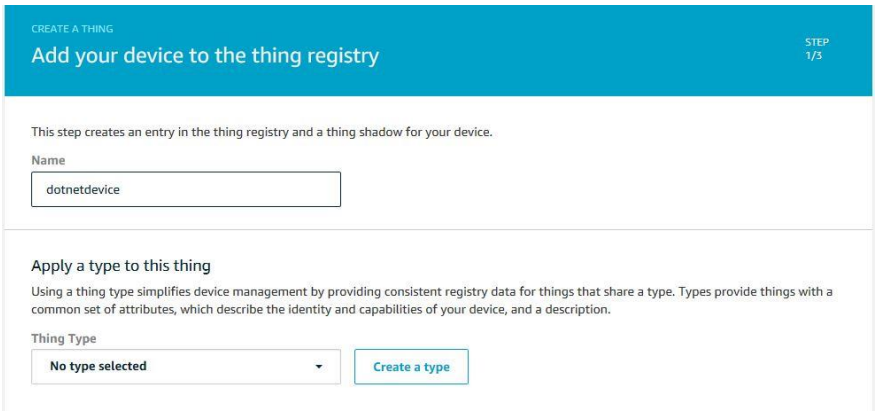


Ilustración 30 AWS IOT Device Manager

Una vez creado el dispositivo, se ha creado una política de acceso, esta política es un requisito necesario para poder utilizar el broker asociado al dispositivo IoT. Las políticas de AWS IoT permiten controlar el acceso al plano de datos de AWS IoT. El plano de datos de AWS IoT consta de operaciones que le permiten conectarse al agente de mensajes de AWS IoT, enviar y recibir mensajes MQTT y obtener o actualizar la sombra de un dispositivo. En efecto se especifica la acción que la política permite o niega.

La política definida para el dispositivo IoT de la solución es la siguiente:

```
{
  "Version": "2019-10-17",
  "Statement": [
    {
      "Action": [
        "iot:Publish",
        "iot:Subscribe",
        "iot:Connect",
        "iot:Receive"
      ],
      "Effect": "Allow",
      "Resource": [
        "*"
      ]
    }
  ]
}
```

Ilustración 31 Política de seguridad AWS IoT definida

Una vez definida la política de acceso al dispositivo, para poder conectar en el dispositivo a través del protocolo MQTT se obtienen los siguientes cuatro artefactos de seguridad necesarios para iniciar la comunicación a través del broker MQTT.

- Certificado de dispositivo
- Clave pública del dispositivo
- Clave privada del dispositivo
- Certificado raíz AmazonRootCA1.pem.

El siguiente fragmento de código muestra un ejemplo de cómo se ha implementado la conexión en modo publicador con el bróker mqtt del dispositivo utilizando los certificados y claves publica/privada definidas anteriormente.

```
string iotEndpoint = "<<iot-AWS-endpoint>>";
Console.WriteLine("AWS IoT Dotnet message publisher starting..");
int brokerPort = 8883;
string topic = "Line{line}/machineOrder";
string message = "Test message";
var caCert = X509Certificate.CreateFromCertFile(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "AmazonRootCA1.crt"));
var clientCert = new X509Certificate2(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "certificate.cert.pfx"), "pass");
var client = new MqttClient(iotEndpoint, brokerPort, true, caCert, clientCert, MqttSslProtocols.TLSv1_2);
string clientId = Guid.NewGuid().ToString();
client.Connect(clientId);
Console.WriteLine($"Connected to AWS IoT with client id: {clientId}.");
int i = 0;
while (true)
{
    client.Publish(topic, Encoding.UTF8.GetBytes($"{message} {i}"));
    Console.WriteLine($"Published: {message} {i}");
    i++;
    Thread.Sleep(5000);
}
```

Ilustración 32 Conexión en modo publicador con el bróker MQTT

Por otro lado, para poder consumir los mensajes publicados en los diferentes topics de la arquitectura definida, se ha implementado este fragmento de código que permite subscribirse al topic y consumir los mensajes.

```
string iotEndpoint = "<<your-iot-endpoint>>";
int brokerPort = 8883;
Console.WriteLine("AWS IoT dotnet message consumer starting..");
var caCert = X509Certificate.CreateFromCertFile(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "AmazonRootCA1.crt"));
var clientCert = new X509Certificate2(Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "certificate.cert.pfx"), "MyPassword1");
var client = new MqttClient(iotEndpoint, brokerPort, true, caCert, clientCert, MqttSslProtocols.TLSv1_2);
client.MqttMsgPublishReceived += Client_MqttMsgPublishReceived;
client.MqttMsgSubscribed += Client_MqttMsgSubscribed;
string clientId = Guid.NewGuid().ToString();
client.Connect(clientId);
Console.WriteLine($"Connected to AWS IoT with client ID: {clientId}");
string topic = "Line{line}/machineOrder";
client.Subscribe(new string[] { topic }, new byte[] { MqttMsgBase.QOS_LEVEL_AT_LEAST_ONCE });
// Keep the main thread alive for the event receivers to get invoked
KeepConsoleAppRunning(() => {
    client.Disconnect();
    Console.WriteLine("Disconnecting client..");
});
```

Ilustración 33 Conexión en modo subscriptor con el bróker MQTT

Una vez dado de alta el broker MQTT y establecida la comunicación con este, se ha procedido a la definición de mensajes que los diferentes actores de la arquitectura intercambiaran para que la información fluya correctamente desde el emisor a los diferentes receptores a través de dicho broker. Se ha definido una librería de mensajería que cuenta con una interfaz base la cual implementan los diferentes mensajes de dicha librería. Los mensajes definidos son los siguientes (ver Ilustración 34):

- AddMeasurement: Este mensaje se envía desde la función lambda cuando el usuario final del sistema decide introducir una nueva medición de algún tipo de parámetro asociado a la maquina como el estado de la cuchilla o la velocidad de la línea.
- LineChangeVelocity: Este mensaje se envía desde la función lambda cuando el usuario final del sistema decide incrementar/disminuir la velocidad de producción de la línea.
- AddUnit: Este mensaje se envía desde la función lambda cuando el usuario final del sistema decide incrementar en X unidades las unidades a producir en el marco de una orden de trabajo.
- LineStatusOrder: Este mensaje se envía desde la función lambda cuando el usuario final del sistema decide arrancar/parar la producción en la línea seleccionada.
- AddWorkOrderToLine: Este mensaje se envía desde la función lambda cuando el usuario final del sistema decide cargar en la máquina de la línea de producción seleccionada la orden de trabajo asociada con el identificador de la orden de trabajo del mensaje.
- WorkOrderAction: Este mensaje se envía desde la función lambda cuando el usuario final del sistema decide realizar una acción sobre la orden de trabajo que previamente se ha cargado en una línea de trabajo. La orden puede ser inicia, parada, pausada y reanudada.

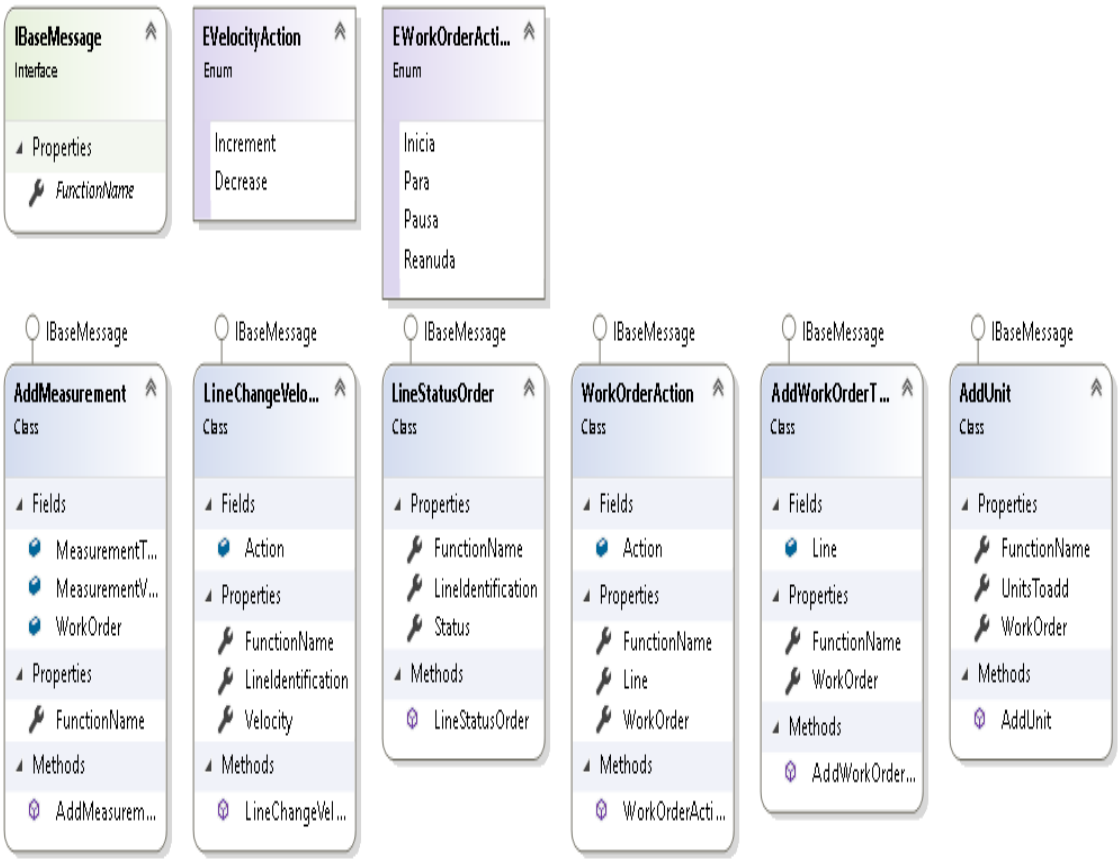


Ilustración 34 Diagrama de clases de la librería de mensajería SmartIoT4Industry

6.4 **Simulación de controlador de maquina**

El principal objetivo de esta tesis es el desarrollo teórico/practico de una arquitectura y solución para la integración de tecnologías de síntesis de voz, computación en la red y computación autónoma dentro de un marco industrial como la empresa Embalpack. Para poder desarrollar el prototipo de validación de la solución, se ha desarrollado un simulador de controladora de maquina la cual recibe la información del broker mqtt y simula las acciones que se realizarían a través de la comunicación con el dispositivo de control de la máquina de cartonaje (normalmente un controlador lógico programable PLC) como se muestra en la subsección 3 de la arquitectura propuesta.

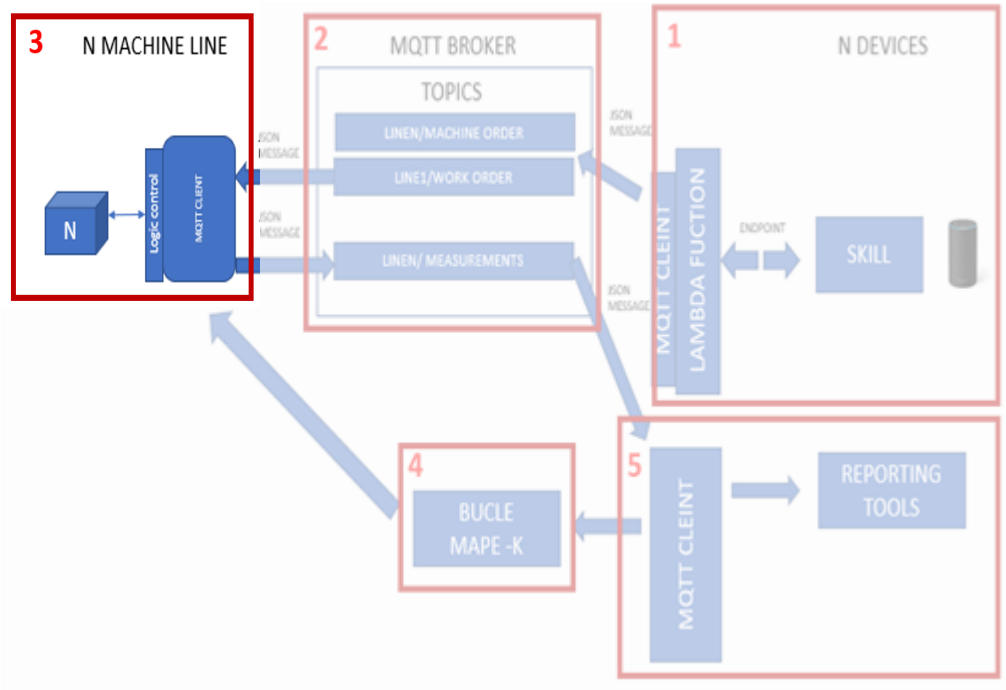


Ilustración 35 Arquitectura general. Sección controladora inteligente

El simulador implementado tiene un módulo de conexionado con el broker de comunicaciones que le permite subscribirse a los específicos para la línea de producción a la que está asociada. Cuando se recibe un mensaje desde el broker de comunicaciones, la controladora interpreta dicha orden y ejecuta la acción requerida basándose en el siguiente diagrama de actividad:

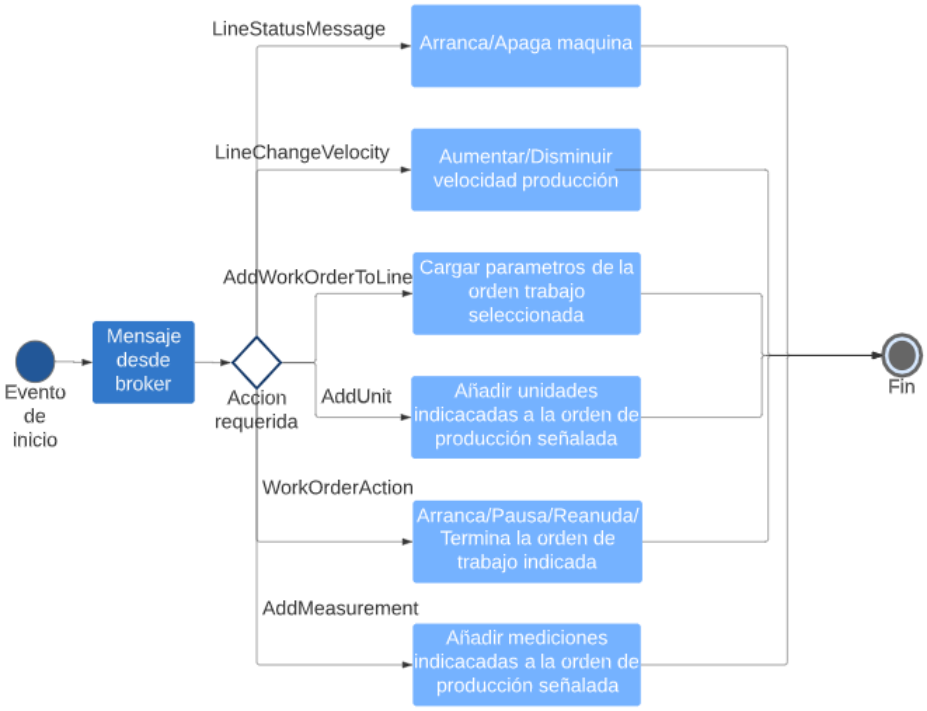


Ilustración 36 Diagrama actividad controladora maquinaria

Este dispositivo aparte de recibir las órdenes del broker y simular las acciones que se realizarían sobre el PLC de esta, para tener una funcionalidad completa del sistema productivo también simula la producción de cartoneras según la orden de trabajo que tiene asociada y la velocidad a la cual trabaja la máquina. La información de los diferentes sensores de la máquina, como son el sensor de presión de corte de la cuchilla, el sensor de temperatura interna de la máquina y el sensor de control de cantidad de materia prima en la bobina de la maquina es recogida por la controladora implementada y almacenada para enviar periódicamente a través del broker MQTT concretamente en los topics “Line{line}/Measurements”, como se ha descrito en el caso de uso (ver Ilustración 14) , un mensaje llamado “ReportStatus”. Este mensaje se encuentra definido dentro de la librería de mensajería propuesta dentro de la solución. El mensaje de ReportStatus contiene información acerca del estado de funcionamiento de la línea de producción junto al estado de la orden de trabajo que se está procesando dentro de la misma si la maquina está en funcionamiento en ese momento y de la información recopilada por los diferentes sensores.

La estructura de información definida para este reporte es el siguiente:

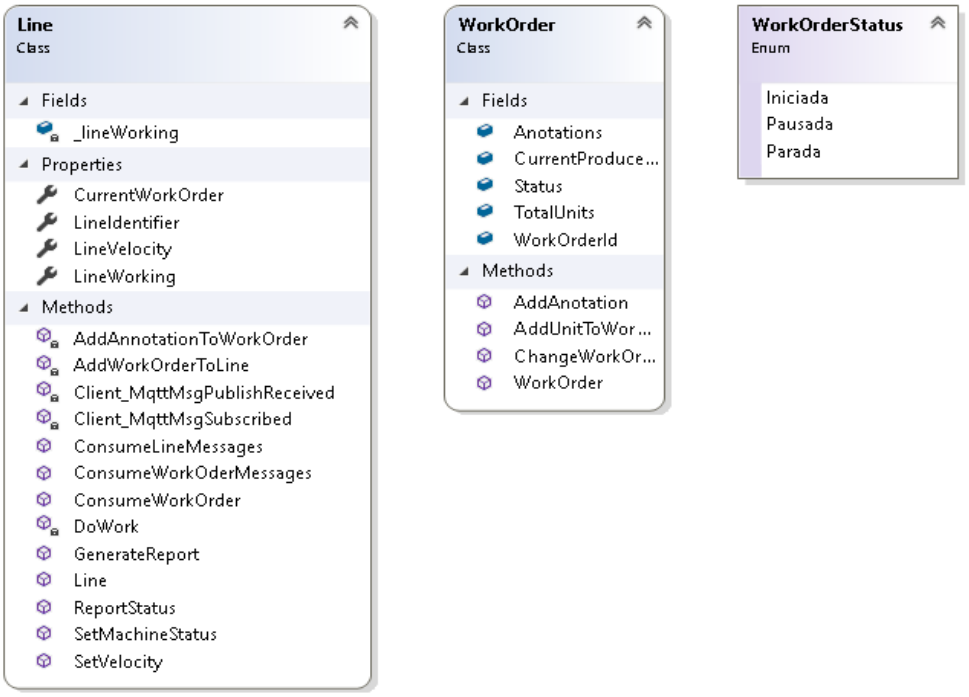


Ilustración 37 Estructura del mensaje ReportStatus

6.5 Auto adaptación del sistema. Bucle de control MAPE-K

La autoadaptación se está convirtiendo en un requisito necesario e incluso indispensable en los enormes sistemas software industriales. Los sistemas autoadaptables son capaces de adaptar dinámicamente su comportamiento y estructura en ejecución en respuesta a variaciones en el sistema y el entorno productivo. Los enfoques actuales de autoadaptación tienden a asumir un mundo cerrado, en el que todos los fenómenos de interés son previstos en tiempo de diseño. Sin embargo, algunos comportamientos adaptativos no se pueden prever a priori. En

este ámbito, se requieren técnicas que den soporte a la evolución en tiempo de ejecución de sistemas autoadaptables (María Gómez). Por lo tanto, la introducción de un bucle de retroalimentación tipo MAPE-K es crucial para la implementación de sistemas auto adaptativos como se presenta en la siguiente imagen:

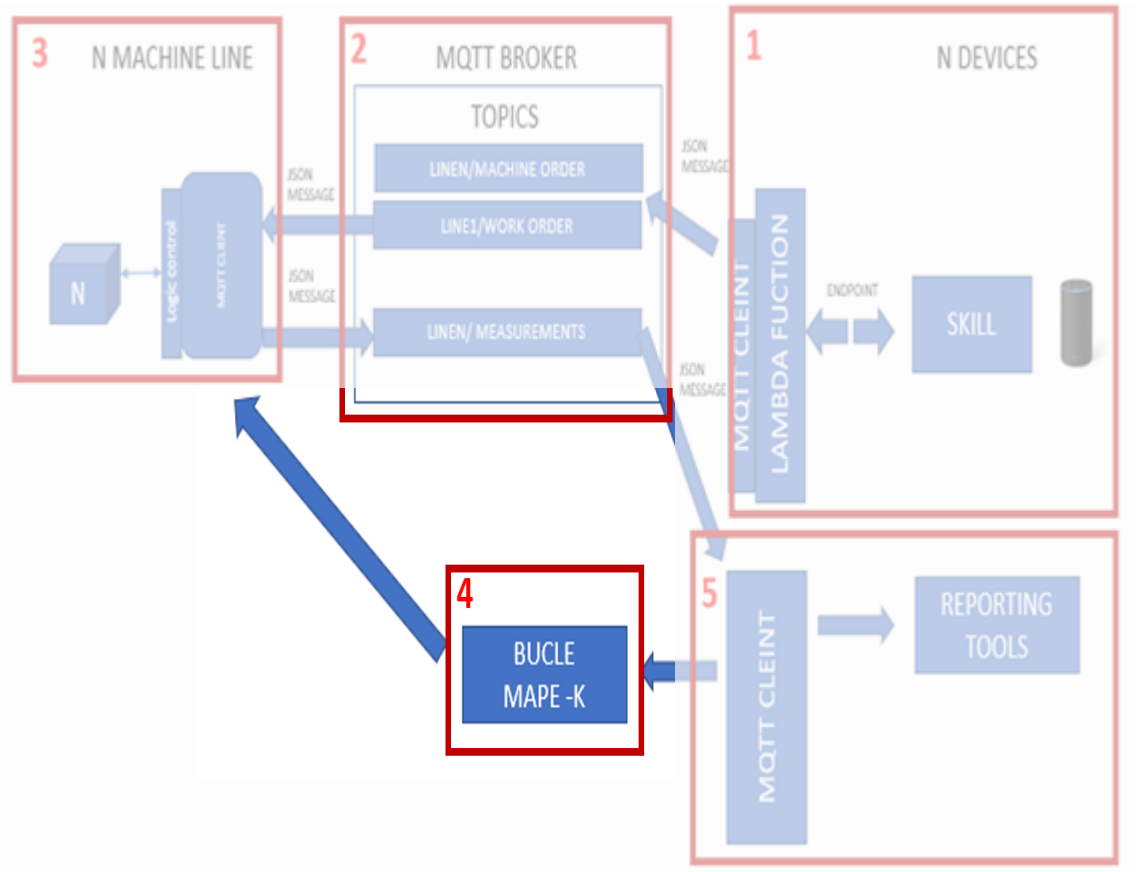


Ilustración 38 Arquitectura general. Sección sistema auto adaptativo

El ciclo de control de este tipo de sistemas auto adaptativos comprende los siguientes cuatro procesos:

- El proceso de monitoreo es responsable de recopilar y correlacionar los datos del software y su entorno
- El proceso de análisis (detección) es responsable de analizar los síntomas adaptativos mediante monitoreando los datos.
- El proceso de planificación (decisión) es responsable de determinar los cambios requeridos y sus métodos de ejecución (es decir, es responsable de determinar las estrategias de adaptación).
- El proceso de ejecución (actuación) es responsable de aplicar la estrategia de adaptación.

El desarrollo e implantación completo e integral de un sistema de retro alimentación para el control del proceso productivo conlleva consigo el estudio pormenorizado de dichos procesos productivos, el análisis y detección de las situaciones y eventos inesperados, la introducción de técnicas de inteligencia artificial y el desarrollo completo del mismo. El desarrollo de este tipo de sistemas escapa al

objetivo principal del proyecto, el cual trata de demostrar como la introducción de este tipo de sistema auto adaptativos pueden introducir un valor añadido aportando una solución tecnológica que se encargue de la detección autónoma y preventiva de errores e introduzca una manera mucho más rápida y ágil de proceder con acciones correctivas generando un marco muy interesante para que empresas como Embalpack pegue el salto a la nueva industria, la industria 4.0. Para ello se han definido tres escenarios para valorar el impacto de la adaptabilidad dentro del sistema diseñado. Los tres escenarios definidos son los siguientes:

- Escenario 1. La máquina está produciendo una orden de trabajo, pero la temperatura de la maquina empieza a subir: En este escenario de adaptabilidad, el bucle de control se encarga de monitorizar la temperatura interna de las diferentes máquinas de producción de la fábrica. Si se detecta que la temperatura empieza a subir y se acerca al máximo de temperatura recomendada por el fabricante, el bucle de control planificara una acción preventiva ordenando a través de la arquitectura de comunicación propuesta que reduzca la velocidad de producción en X unidades para intentar que la máquina recupere una temperatura dentro del rango ideal. Si, aun así, tras la reducción de la velocidad, la temperatura sigue aumentando, llegando al máximo de temperatura definido por el fabricante, el bucle de control aplicara una acción correctiva para evitar que la maquina se sobrecaliente y se estropee, deteniendo el proceso productivo totalmente en dicha máquina y notificando el inesperado evento al usuario. La siguiente imagen muestra el diagrama adaptativo del este escenario.

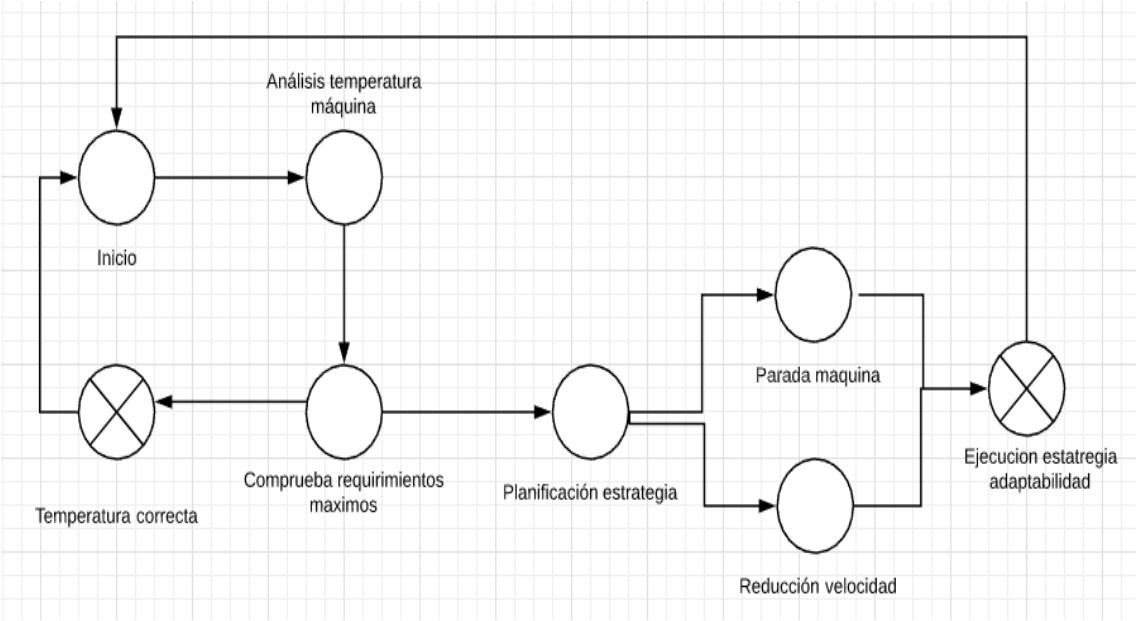


Ilustración 39 Adaptación sobre temperatura en máquina de producción.

- Escenario 2. La máquina está produciendo una orden de trabajo, pero la cuchilla de corte empieza a perder filo: En este escenario de adaptabilidad, el bucle de control se encarga de monitorizar la presión de corte de cada una de las cuchillas de las diferentes máquinas de producción de la fábrica. Las diferentes máquinas de la fábrica, basándose en los parámetros definidos en las ordenes de trabajo que están produciendo, cortan la materia prima en

diferentes formas a través de las cuchillas de corte de cada una de ellas. Estas cuchillas a lo largo del tiempo van perdiendo filo poco a poco hasta que su filo no es suficiente y ya no son capaces de cortar la materia prima lo que conlleva perdida de esta, hasta que el operario de la fábrica detecta el problema, detiene la maquina y reemplaza la cuchilla. Gracias al escenario de adaptabilidad definido, el cual es capaz de a través de los datos de presión de corte de cada una de las máquinas, detectar en primer caso una pérdida de filo que aunque no bloquea el productivo, es un indicador de un posible fallo en un corto periodo de tiempo, por lo que el bucle de control planificara una acción preventiva de reducción de velocidad y notificará a través de la arquitectura de comunicación propuesta y la herramienta de reporting el estado de la cuchilla para que el operario de fabrica planifique un reemplazo de la cuchilla previo al fallo total de la misma. Si el operario, no reemplaza la cuchilla a tiempo, llegara un momento que la presión de corte bajara de la mínima para poder corte la materia por lo que podría ocasionar perdida de materia prima y pérdida de tiempo en el proceso productivo. Para evitar esta circunstancia, el bucle de control planificara una acción correctiva, mandando a la maquina con la cuchilla defectuosa detener el proceso productivo inmediatamente y notificando al usuario del inesperado evento. La siguiente imagen muestra el diagrama adaptativo del este escenario.

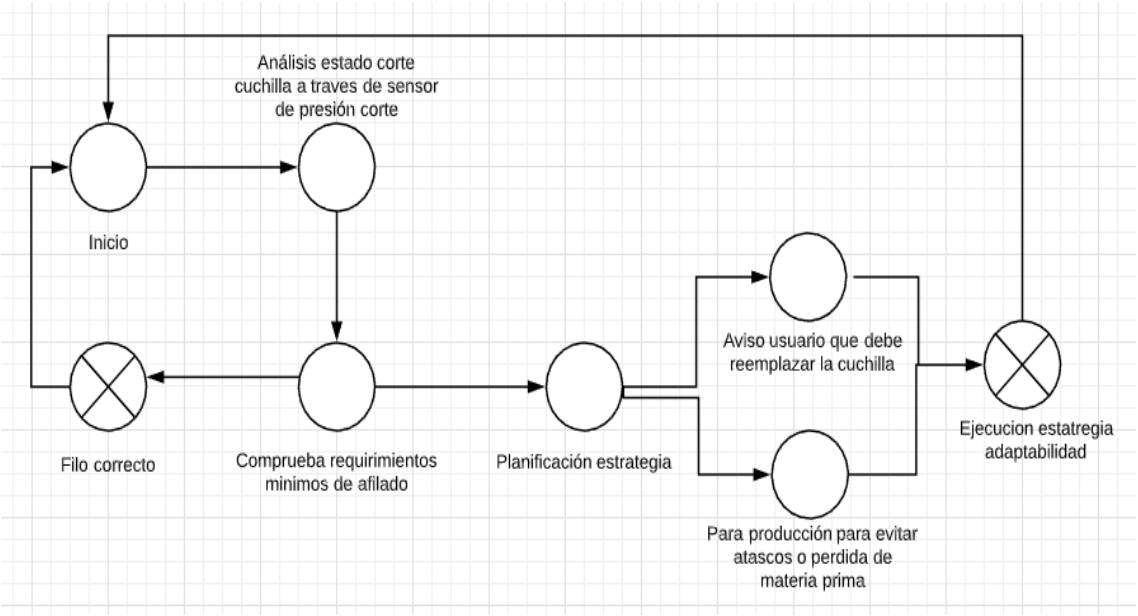


Ilustración 40 adaptación perdida de filo en la cuchilla de máquina de producción.

- Escenario 3. La máquina está produciendo una orden de trabajo, pero la bobina de materia prima empieza a quedarse vacía: En este escenario de adaptabilidad, el bucle de control se encarga de monitorizar el sensor de cantidad de materia prima de cada una de las bobinas de las diferentes máquinas de producción de la fábrica. Las diferentes máquinas de la fábrica poseen una bobina donde se cargan las tortas de materia prima que después se convertirán en cartoneras al final del proceso productivo. Estas bobinas son reemplazadas por los operarios de la fábrica cuando se han acabado o visualmente detectan que están acabándose. A veces, las bobinas se acaban

completamente y hasta que el operario de la fábrica detecta esta situación, detiene la maquina y reemplaza la bobina se pierde tiempo del proceso productivo y además ocasionan que la máquina se sobre caliente al intentar producir sin recibir materia prima dentro de su proceso productivo. Gracias al escenario de adaptabilidad definido, el cual es capaz de a través de los datos del sensor de control de grosor de la bobinas de cada una de las máquinas, detectar en primer caso que el grosor baja de manera considerable lo que aunque no bloquea el productivo, es un indicador de un posible fallo en un corto periodo de tiempo, por lo que el bucle de control planificara una acción preventiva notificando a través de la arquitectura de comunicación propuesta y la herramienta de reporting el estado de bobina cerca de acabarse para que el operario de fabrica planifique un reemplazo de la bobina previo al fallo total de la misma. Si el operario, no reemplaza la bobina a tiempo, llegara un momento que la bobina de materia prima se acabe y empiece a ocasionar la situación inesperada definida anteriormente. Para evitar esta circunstancia, el bucle de control planificara una acción correctiva, mandando a la maquina con la bobina acabada detener el proceso productivo inmediatamente y notificando al usuario del inesperado evento. La siguiente imagen muestra el diagrama adaptativo del este escenario.

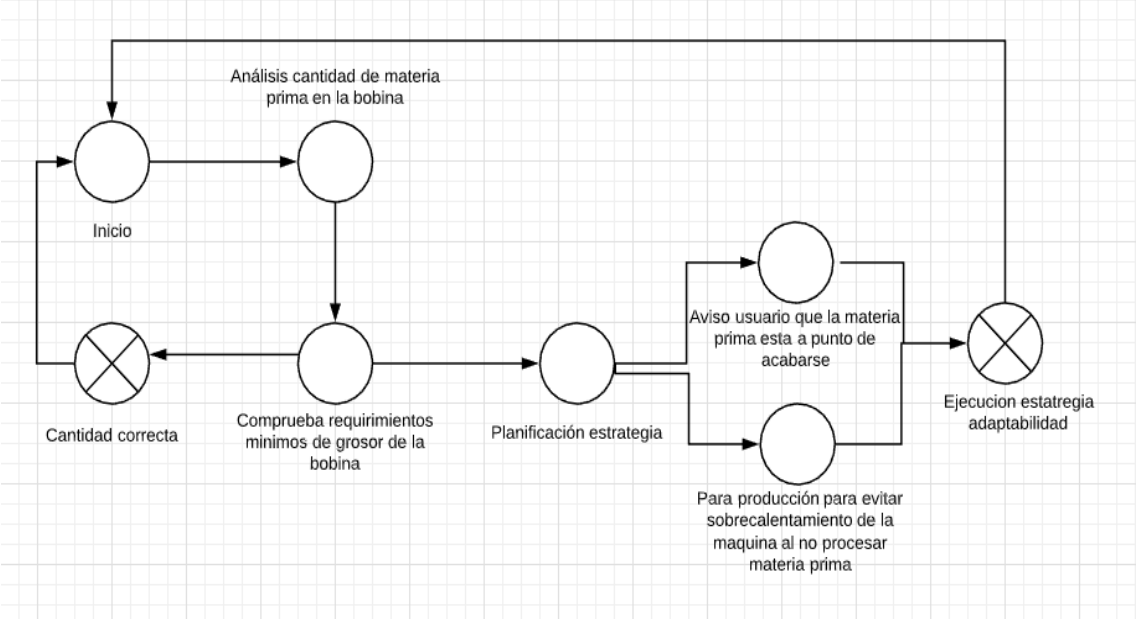


Ilustración 41 Adaptación bobina de materia prima cercana a su finalización

La introducción de sistemas independientes de tipo adaptativo, en la arquitectura que controla en proceso productivo de manera independiente y no invasiva, ofrece como principales beneficios la reducción de los costes productivos, minimizando perdidas de materia prima y reduciendo los tiempos de parada del propio proceso productivo. Por otro lado, los fallos serán menos frecuentes y, por tanto, los costes de mantenimiento sobre todo los de tipo correctivo se reducirán drásticamente. Desarrollar estrategias de control efectivas a través de sistemas autónomos para el rendimiento de activos junto con estrategias de gestión para el rendimiento de estos es la combinación correcta para optimizar el valor generado por todos los conjuntos de activos industriales. Utilizar esta estrategia dual de gestión y control es la esencia de la arquitectura diseñada.

El siguiente fragmento de texto, presenta una trama de datos generada por el bucle de control con una situación de cuchilla desafilada, una vez se detecta el estado indeseado, se ejecuta el escenario 2 propuesto. Tras estudiar el estado del afilado y detectar el valor por debajo del limite aceptable, planifica y ejecuta la acción, en este caso mandando detener la velocidad de

```
****************************************
MessageReceived
{"Line":{"LineVelocity":31,"LineIdentifier":"a","LineTemperature":90.0,"PaperCoilPercentage":-61,"SharpBladePercentage":-61,"CurrentWorkOrder":{"WorkOrderId":1,"Status":0,"Anotations":{},"CurrentProducedUnit":125,"TotalUnits":100},"LineWorking":true},"WorkOrder":{"WorkOrderId":1,"Status":0,"Anotations":{},"CurrentProducedUnit":125,"TotalUnits":100}}
Monitor phase, data received for line a is
Coil status percentage is 89
Blade status percentage is 10
Temperature is 90
----------------------------------------
Analyze phase
Analyzing temperature
Normal temperature detected
Analyzing Coil status
Coil quantity is correct
Analyzing blade sharp status
Blade sharp is not correct
----------------------------------------
Planning phase
No action for temperature planned
No action for Coil status planned
Action for blade status is planned
----------------------------------------
Decreasing the speed of the machine as a preventive action
****************************************
```

El mensaje que recibe la controladora de la máquina de producción desde el bucle de control a través de broker de comunicaciones, de reducción de velocidad es el siguiente:

```
MessageReceived is
{"Action":"Decrease","LineIdentification":"a","Velocity":5,"FunctionName":"LineChangeVelocityOrder"}
```

6.6 **Reportes de Producción**

Tener una visión general a golpe de vista del estado de las diferentes máquinas de producción y de las ordenes de trabajo que se están ejecutando en ellas junto a notificaciones sonoras cuando algo no funciona correctamente en alguna de las máquinas, es el principal objetivo de esta herramienta de reporting introducida en la arquitectura general (concretamente en la subsección 5 de la arquitectura general del sistema).

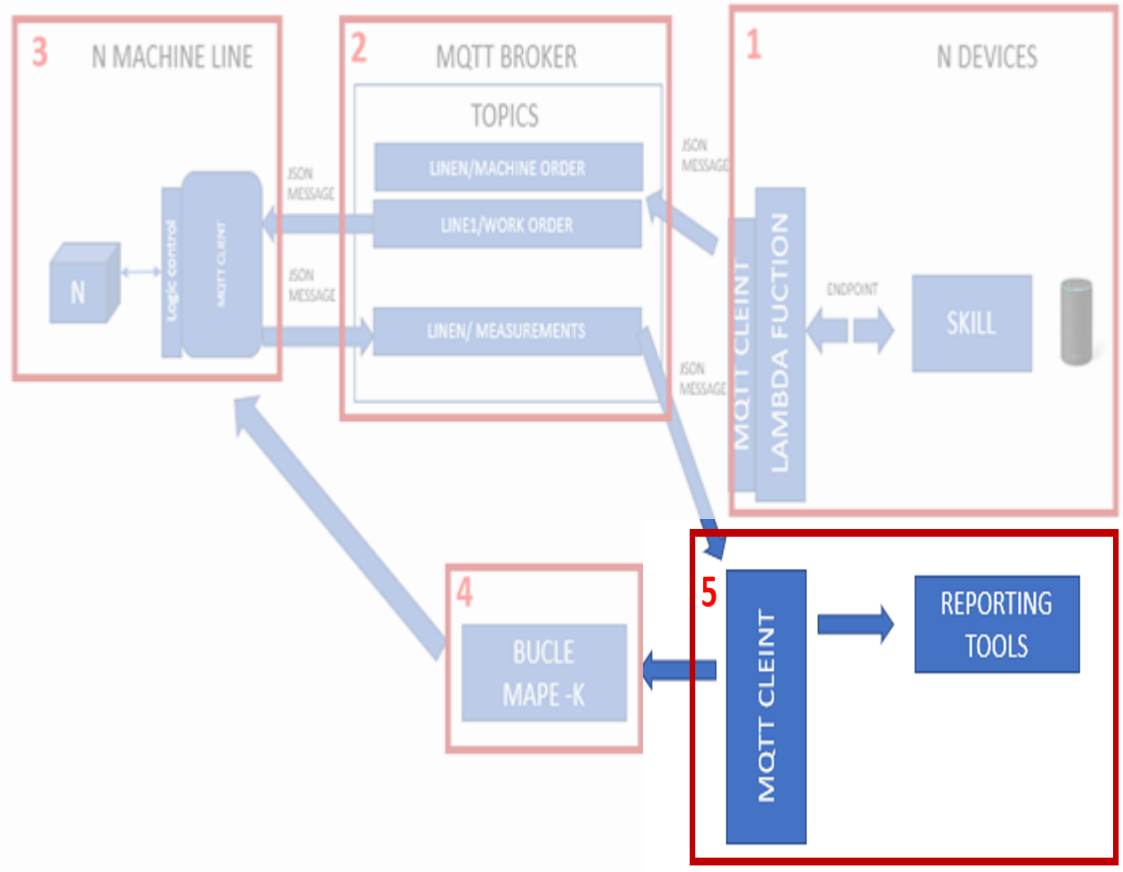


Ilustración 42 Arquitectura general. Sección reporting tool, interfaces multimodales

Para proporcionar una herramienta de reporte que permita a los diferentes operarios monitorizar el estado del proceso dentro de la Smart Factory propuesta, se ha desarrollado una aplicación cuya interfaz es multimodal abordando dos modalidades de comunicación con sus usuarios, el modo visual mediante una interfaz gráfica y el modo oral mediante una interfaz de voz. La información que se muestra de modo visual o a través de notificaciones sonoras se obtiene a través del broker mqtt concretamente del mensaje de “ReportStatus” publicado periódicamente por las diferentes controladoras de las máquinas de producción.

La interfaz actual de monitorización de cada máquina es visual a través del display de cada una de ellas, lo que hace que el operario necesite estar físicamente al lado de la maquina y además la única forma de acceder a la información es través de la interfaz exclusivamente visual. Partiendo de uno requisitos iniciales de la solución propuesta que se centra en sacarle el máximo rendimiento al personal y a las líneas de producción, surge por un lado la necesidad de centralización de la información para

un control más integral del sistema de producción y por otro consiguiendo la eliminación de la necesidad de la presencia física de los operarios en las líneas de producción para la monitorización de estas. Por ello, se propone una herramienta de reporting que presenta el acceso a la información de forma más natural e intuitiva posibilitando que la información sea accesible desde nuevos entornos que se adapten a las necesidades de los usuarios. Se busca eliminar barreras de acceso de forma que los usuarios que por diversos motivos no puedan acceder a una interfaz visual dispongan de más alternativas. Actualmente, las interfaces de voz son un componente muy importante de las interfaces multimodales, dado que una interfaz de voz sofisticada puede ser de gran utilidad para proporcionar acceso a la información a distintos tipos de usuario de una forma sencilla y natural. La incorporación del dialogo en un sistema informático, permite la interacción persona-máquina usando el habla para la solución de diversos problemas cotidianos. De este modo, los sistemas de diálogo tienen como objetivo facilitar la interacción natural mediante el habla entre una persona y un sistema informático. Un sistema de diálogo ideal reconoce el habla sin restricciones y proporciona respuestas bien formadas, con sentido y voz completamente natural.

El prototipo ha sido desarrollado como una aplicación de escritorio que puede correr en cualquier sistema Windows. Para la parte grafica se ha utilizado la tecnología WPF implementando el patrón MVVM. Dicho patrón de implementación busca dividir los programas en 3 partes para un mejor mantenimiento: el modelo (mantiene los orígenes de datos), la vista (mantiene la información de la UI) y la vista-modelo (actúa de conector entre las otras dos capas). Por otro lado, para la parte de notificaciones sonoras, se ha utilizado el software de sinterización de voz de Microsoft Speech. Microsoft Speech Platform es una tecnología de Microsoft para síntesis y reconocimiento de voz de alta calidad para los equipos con sistema operativo Windows.

Las siguientes imágenes muestras una captura de la interfaz gráfica implementada del prototipo de esta herramienta de reporting.

Linea	Estado	Velocidad	Temperatura	Cuchilla	Bobina	Trabajo actual
A	Parada	0	0	0	0	0
B	Parada	0	0	0	0	0
C	Parada	0	0	0	0	0
D	Parada	0	0	0	0	0

Ilustración 43 UI estado líneas de trabajo

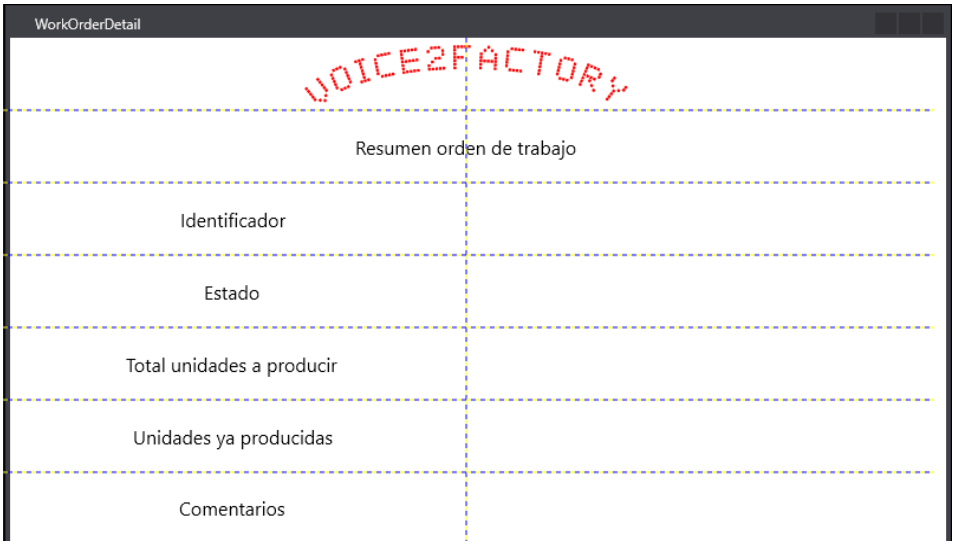


Ilustración 44 IU orden de trabajo

7. **Validación del prototipo implementado**

Este capítulo se centrará en la realización de las pruebas de validación tanto a nivel de componentes individuales como de integración de los componentes pertinentes para la comprobación del correcto funcionamiento de la solución implementada. La importancia de la realización de pruebas en el desarrollo de software reside en que en cualquier etapa del ciclo de vida es posible que aparezcan errores, los cuales pueden permanecer sin ser descubiertos hasta incluso en las fases de implantación del sistema. Dependiendo de los objetivos a los que se enfoquen las pruebas, hay cuatro tipos de pruebas de software:

- Pruebas funcionales: Las pruebas funcionales verifican que cada función de la aplicación de software funciona de acuerdo con la especificación de requisitos. Pruebas funcionales muestra “Lo que hace el sistema”. El objetivo de esta prueba es verificar si el sistema es funcionalmente perfecto. Entre la categoría de pruebas funcionales destacan:
    - Pruebas Unitarias: Este tipo de pruebas tienen como objetivo asegurar que cada unidad funcione de forma correcta y eficiente por separado. Además de que el código se comporta respecto a las expectativas que se esperan de él, verificando que sea correcto el nombre, los nombres y tipos de los parámetros, el tipo de lo que se devuelve, además si el estado inicial es válido, entonces el estado final es válido.
    - Pruebas de Integración: Consiste en la realización de pruebas específicas, concretas y exhaustivas con el fin probar y validar que el comportamiento del software es el que se ha especificado.
    - Pruebas del Regresión: Son un tipo de pruebas de software que tienen como objetivo descubrir bugs (errores), carencias de funcionalidad, o divergencias funcionales con respecto al comportamiento esperado del software. Esto se produce debido a la realización de un cambio en el programa. Donde se evalúa el correcto funcionamiento del software desarrollado respecto a evoluciones o cambios funcionales.
    - Prueba de Componentes: Tienen por objetivo localizar defectos y comprobar el funcionamiento de módulos software, programas, objetos, clases, etc. Que puedan probarse por separado; es decir, se pueden realizar de manera independiente al resto del sistema en función del contexto.

- Pruebas no funcionales: Se ocupan de los requisitos no funcionales. Las pruebas no funcionales ayudan a estimar la preparación de un sistema de acuerdo con los diferentes criterios que no están cubiertos por las pruebas funcionales. A diferencia de las pruebas funcionales, muestra “Qué bueno funciona el sistema”. Entre la categoría de pruebas no funcionales destacan:
    - Pruebas de interfaz de usuario (UI): tiene como objetivo garantizar que la interfaz gráfica de usuario de la aplicación cumpla con las especificaciones. Esto ayuda a evaluar elementos de diseño como el diseño, colores, fuentes, tamaños de fuente, etiquetas, cuadros de texto, formato de texto, títulos, botones, listas, iconos, enlaces y contenido.

- Pruebas de almacenamiento: Verifica la aplicación bajo prueba, almacena los datos relevantes en los directorios correctos y tiene suficiente espacio para evitar la terminación inesperada debido a un espacio en disco insuficiente. Ayuda a determinar si la aplicación utiliza o no más memoria que la que se estima que ocupa el espacio en disco puede causar un tiempo de inactividad significativo.
- Pruebas operativas: Tienen como objetivo evaluar un sistema o componente en su entorno operativo. Al usarlo, podemos garantizar el cumplimiento del sistema y los componentes en el entorno operativo estándar de la aplicación.
- Pruebas de vulnerabilidad y seguridad: Tienen como objetivo evaluar la cantidad de riesgos involucrados en el sistema para reducir la probabilidad del evento. Ayuda a prevenir problemas que pueden afectar la integridad y estabilidad de la aplicación.

- Pruebas estructurales: verifica la implementación del programa o código a través de pruebas de la estructura del sistema de software o sus componentes. El probador se concentra en el trabajo del software durante las pruebas estructurales. Puede ser utilizado en todos los niveles de prueba.

- Pruebas relacionadas con el cambio: Pruebas relacionadas con el cambio se proporciona para garantizar que los errores erradicados previamente se hayan solucionado y para detectar errores que pueden haber aparecido accidentalmente en una nueva versión. Entre la categoría de pruebas relacionadas con el cambio destacan:
  - Pruebas de confirmación: Sirven para asegurarse de que el error ha sido eliminado con éxito. En pocas palabras, el caso de prueba que detectó originalmente el error se ejecuta de nuevo y esta vez debería pasar sin problemas.
  - Pruebas de regresión: Este tipo de prueba Consisten no solo en los casos de prueba de errores detectados. sino también para garantizar que no se hayan detectado o descubierto nuevos defectos después de los cambios.

Existen diferentes técnicas de evaluación dinámica proporcionando diferentes criterios para diseñar los casos de prueba que se ejecutaran durante la fase de validación del proyecto. La agrupación más general y conocida a nivel de técnicas de prueba de software son las siguientes:

- Técnicas de caja blanca o estructurales, estas se basan en un minucioso examen de los detalles procedimentales del código a evaluar, por lo que es necesario conocer la lógica del programa.

- Técnicas de caja negra o funcionales realizan pruebas sobre la interfaz del programa a probar, entendiendo por interfaz las entradas y salidas de dicho programa. No es necesario conocer la lógica del programa, únicamente la funcionalidad que debe realizar.

El objetivo de las pruebas no es asegurar la ausencia total de defectos en un software, únicamente intentar encontrar defectos en el software desarrollado. El objetivo es diseñar pruebas que sistemáticamente saquen a la luz diferentes clases de errores, haciéndolo con la menor cantidad de tiempo y esfuerzo. En este proyecto se ha optado por la utilización de pruebas funcionales concretamente unitarias por las ventajas que ofrecen la posibilidad del análisis de cada módulo del código de forma individual, ayudando a detectar errores concretos y de comunicación, con el fin probar y validar que el comportamiento del software es el que se ha especificado dentro de la arquitectura final del sistema.

7.1 Validación de la skill VoiceToFactory. Alexa developer console

Para poder realizar pruebas unitarias a la skill desarrollada se ha utilizado en entorno de testeo de “Alexa Developer Console”. Este entorno es el site principal a la hora de desarrollar una skill para Alexa. Permite cubrir gran parte del ciclo de desarrollo, pruebas, distribución, certificación y analíticas. La siguiente imagen muestra captura tomada en la fase de testeo de la skill de VoiceToFactory.

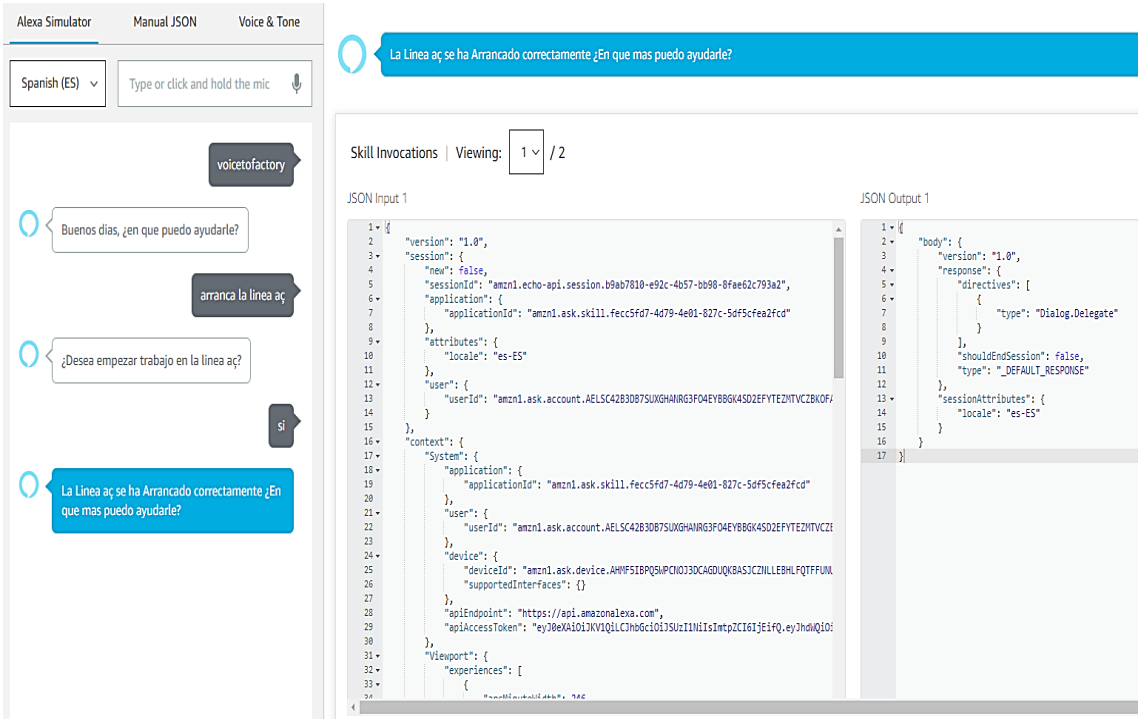


Ilustración 45 Testeo de la skill VoiceToFactory

Para validar la gramática definida, se han definido diferentes casos de test para validar las diferentes interacciones vocales entre la skill y el operario de maquina definidas en el dialog de la skill. Estos casos de test intentan cubrir el proceso de testing de los diferentes Intents y los Confirmacion y Elicitation prompts asociados con ellos. Las siguientes tablas muestran unos ejemplos del proceso de testing llevado a cabo durante el proceso de validación de la skill:

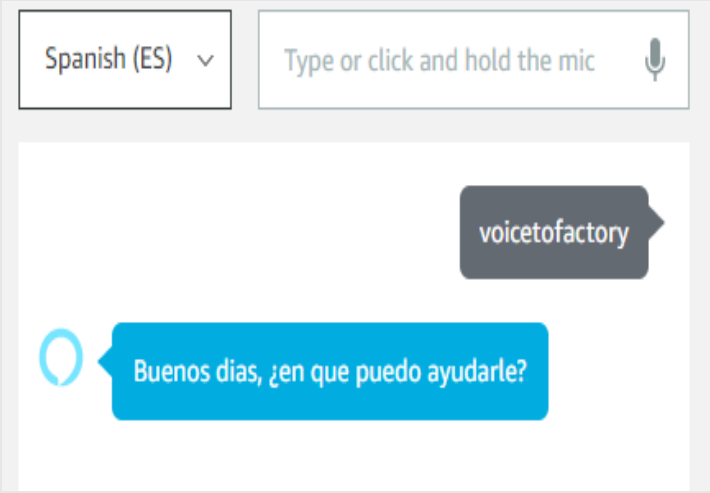
Validación invocation name skill	Código del CP	CP-1
Descripción: Validación invocation name skill		
Prerrequisitos 1. Skill VoiceToFactory cargada en el simulador de Alexa		
Pasos: 1. Invocar la skill con el nombre VoiceToFactory para que Alexa abra la skill de VoiceToFactory		
Resultado esperado: Alexa abre la skill desarrollada y nos contesta con el mensaje de LaunchMessage "Buenos dias, ¿en qué puedo ayudarle?"		
Resultado obtenido: Se recibe el mensaje de LaunchMessage correctamente, la imagen siguiente muestra la evidencia 		

Tabla 1 Caso de prueba Validación invocation name skill

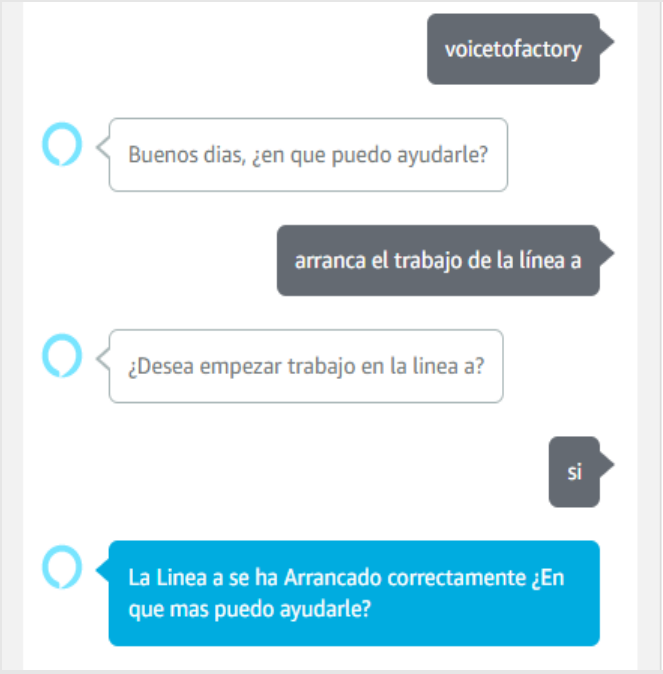
Validación Confirmacion prompt Intent Arranca	Código del CP	CP-2
Descripción: Validación Confirmation prompt del Intent Arranca		
Prerrequisitos 1. Skill VoiceToFactory cargada en el simulador de Alexa 2. Skill invocada a través del InvocationName		
Pasos: 1. Invocar el intent Arranca con el nombre de línea A, a través de la frase “Arranca el trabajo de la línea A” 2. La skill deberá preguntar por confirmación sobre la orden emitida		
Resultado esperado: Al tener un confirmation prompt activo para el slot del Intent, al reconocer que se invoca el intent nos pedirá una confirmación del Intent		
Resultado obtenido: Se recibe el mensaje de confirmación de la línea A, tras confirmar que la orden es correcta la skill manda el mensaje a la función lambda y recibe el mensaje de confirmación de ejecución de la orden desde la función lambda. 		

Tabla 2 Caso de prueba Validación Confirmacion prompt Intent Arranca

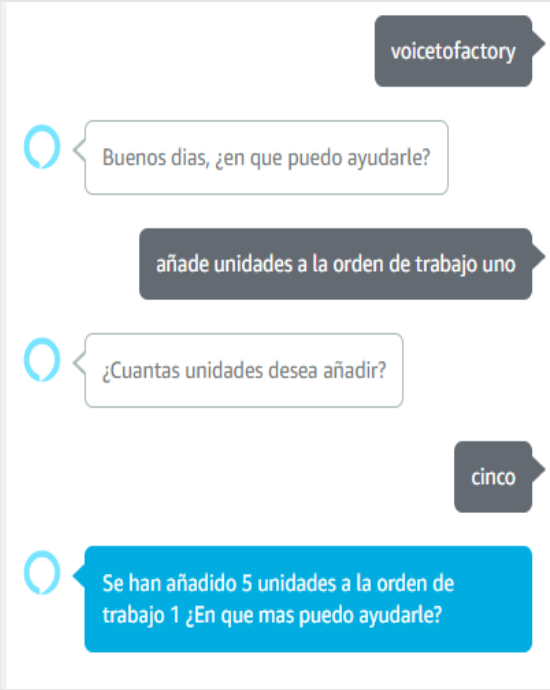
Validación Elicitation prompt Intent Añade Unidades	Código del CP	CP-3
Descripción: Validación elicitation prompt del Intent AñadeUnidades		
Prerrequisitos 1. Skill VoiceToFactory cargada en el simulador de Alexa 2. Skill invocada a través del InvocationName		
Pasos: 1. Invocar el intent AñadeUnidades con el nombre de la orden de trabajo 1, a través de la frase “Añade unidades a la orden de trabajo uno” 2. La skill deberá preguntar cuántas unidades se quieren añadir a la orden de trabajo		
Resultado esperado: Al tener un elicitation prompt activo para el slot del Intent, al reconocer que se invoca el intent nos pedirá una el número de unidades a añadir a la orden de trabajo.		
Resultado obtenido: Se recibe el mensaje de petición de unidades de la orden 1, tras indicarle la cantidad de unidades a añadir la skill manda el mensaje a la función lambda y recibe el mensaje de confirmación de ejecución de la orden desde la función lambda.  		

Tabla 3 Caso de prueba Validación Elicitation prompt Intent Añade Unidades

Validación Elicitation & Confirmation prompt Intent Acción sobre orden de trabajo	Código del CP	CP-4

Descripción:
Validación combinación elicitation prompt para slot del Intent y confirmation prompt para Intent Acción sobre orden de trabajo

Prerrequisitos

1. Skill VoiceToFactory cargada en el simulador de Alexa
2. Skill invocada a través del InvocationName

Pasos:

1. Invocar el intent Acción sobre orden de trabajo con el nombre de la orden de trabajo 1, a través de la frase “Pausa la orden de trabajo 1”
2. La skill deberá preguntar por la validación sobre la Acción Pausa
3. Tras confirmar, la skill deberá hacer una petición acerca de sobre qué línea de trabajo queremos realizar esta acción.
4. Tras indicarle la orden de trabajo, la skill deberá pedir una confirmación de la orden completa construida

Resultado esperado:
Al tener un elicitation prompt activo para el slot del Intent, al reconocer que se invoca el intent nos pedirá una el número de unidades a añadir a la orden de trabajo.

Resultado obtenido:
Se reciben los diferentes mensajes de confirmación y petición a través de la interacción con la skill, tras todas las validaciones la skill manda el mensaje a la función lambda y recibe el mensaje de confirmación de ejecución de la orden desde la función lambda.

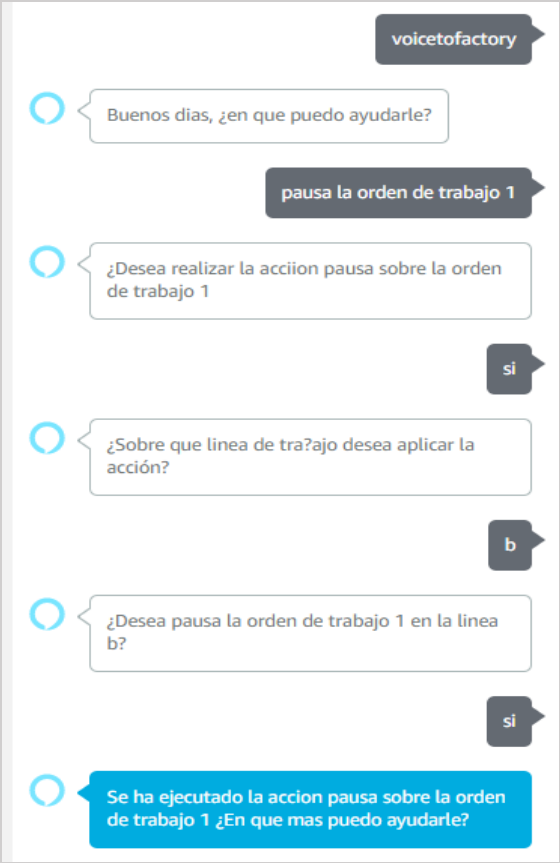


Tabla 4 Caso de prueba Validación Elicitation & Confirmation prompt

Validación Stop Intent	Código del CP	CP-5
Descripción: Validación del mensaje de Stop Intent que provoca que Alexa deje de responder a peticiones sobre la skill VoiceToFactory		
Prerrequisitos 1. Skill VoiceToFactory cargada en el simulador de Alexa 2. Skill invocada a través del InvocationName		
Pasos: 1. Invocar el Stop intent con la palabra termina.		
Resultado esperado: Alexa cierra la skill desarrollada y nos contesta con el mensaje de EndMessage		
Resultado obtenido: La skill tras recibir la orden de desactivar la skill a través de la orden "termina", cierra la skill y notifica al usuario a través del mensaje predeterminado "Adiós"		

Tabla 5 Caso de prueba Validación Stop Intent

7.2 Validación de función lambda

Es importante probar el código de la función de Lambda de forma independiente, para asegurarse de que completa la tarea prevista. Los errores pueden ser una respuesta no válida que devuelve la función de Lambda, errores de ejecución cuando se dispara la función o errores debido a limitaciones de ejecución por parte del servicio AWS. Para la validación de la función lambda implementada, se ha utilizado la herramienta de Amazon AWS llamada Testing Lambda functions. A través de la consola de desarrollador de la herramienta, se ha invocado manualmente la función Lambda utilizando datos de eventos de ejemplo. AWS Lambda ejecuta la función Lambda, realiza las funciones específicas y devuelve resultados obtenidos. Se simulan las diferentes invocaciones que llegarían desde la skill y además se monitoriza el broker mqtt para validar que la función lambda publica diferentes mensajes en el topic especifico.

Se han definido e implementado diferentes casos de prueba en el entorno de test para la validación de la función lambda, En esta herramienta de test se pueden ver diferentes métricas tras la ejecución del escenario de test:

- La sección Resultado de la ejecución muestra el estado de ejecución como exitoso y también muestra los resultados de ejecución de la función, devueltos por la instrucción return.
- La sección Resumen muestra la información clave informada en la sección Salida del registro (la línea INFORME en el registro de ejecución).
- La sección Salida del registro muestra el registro que AWS Lambda genera para cada ejecución. Estos son los registros escritos en CloudWatch por la función Lambda. La consola AWS Lambda muestra estos registros para su conveniencia.

Algunos de los casos de validación definidos se definen en las siguientes tablas, en ellos, junto con la descripción del test y el resultado obtenido se añaden varias capturas de la sección de resultado de ejecución y resumen además de una imagen del mensaje serializado en JSON publicado en el broker de comunicaciones tras la ejecución correcta de la función lambda:

Validación Lambda Arranca línea trabajo	Código del CP	CP-6

Descripción:
Validación de la función lambda que interpreta la Intent de la skill de Alexa Arranca.

Prerrequisitos
1. Función lambda desplegada en el nodo especifico de Amazon AWS

Pasos:
1. Invocar el evento de prueba de invocación de la skill
2. Invocar el evento de prueba de ArrancaLineaA

Resultado esperado:
La función lambda interpreta la orden enviada a través de la skill, publica el mensaje especifico en el broker de comunicaciones y construye el mensaje textual para enviárselo de vuelta a la skill para informar al usuario de la ejecución correcta de la función.

Resultado obtenido:
El test se ejecuta correctamente, el mensaje es publicado en el broker de comunicaciones y el mensaje de respuesta a la skill se construye correctamente, las imágenes siguientes muestran las evidencias del resultado de los test

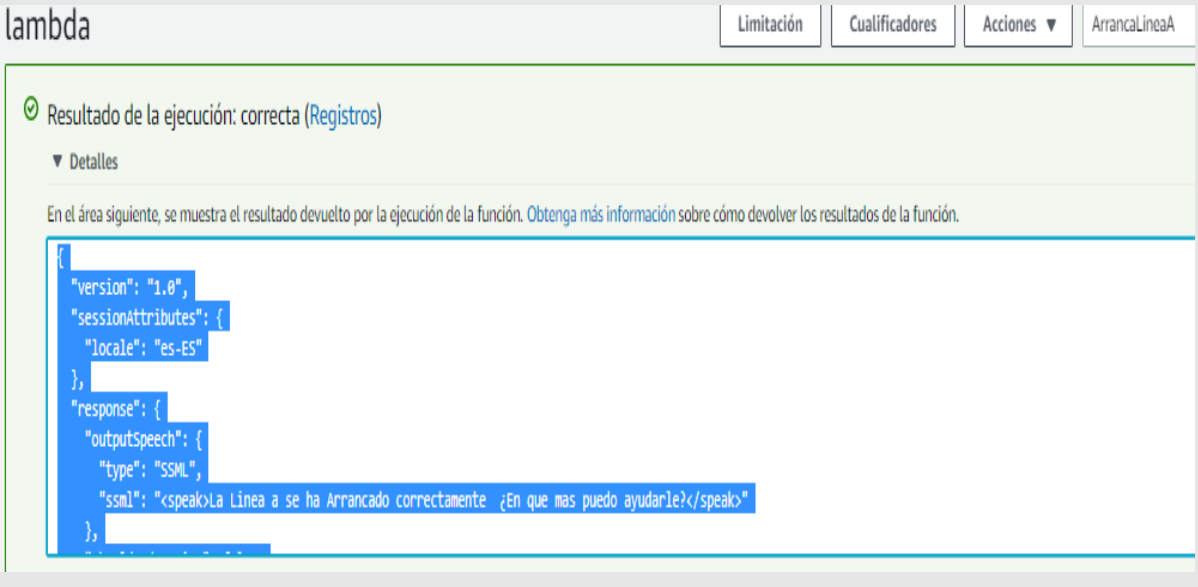


Tabla 6 Validación Lambda Arranca línea trabajo

.

Validación Lambda Aumenta la velocidad	Código del CP	CP-7

Descripción:

Validación de la función lambda que interpreta la Intent de la skill de Alexa AumentaVelocidad.

Prerrequisitos

2. Función lambda desplegada en el nodo especifico de Amazon AWS

Pasos:

3. Invocar el evento de prueba de invocación de la skill
4. Invocar el evento de prueba de AumentaVelocidad

Resultado esperado:

La función lambda interpreta la orden enviada a través de la skill, publica el mensaje especifico en el broker de comunicaciones y construye el mensaje textual para enviárselo de vuelta a la skill para informar al usuario de la ejecución correcta de la función.

Resultado obtenido:

El test se ejecuta correctamente, el mensaje es publicado en el broker de comunicaciones y el mensaje de respuesta a la skill se construye correctamente, las imágenes siguientes muestran las evidencias del resultado de los test

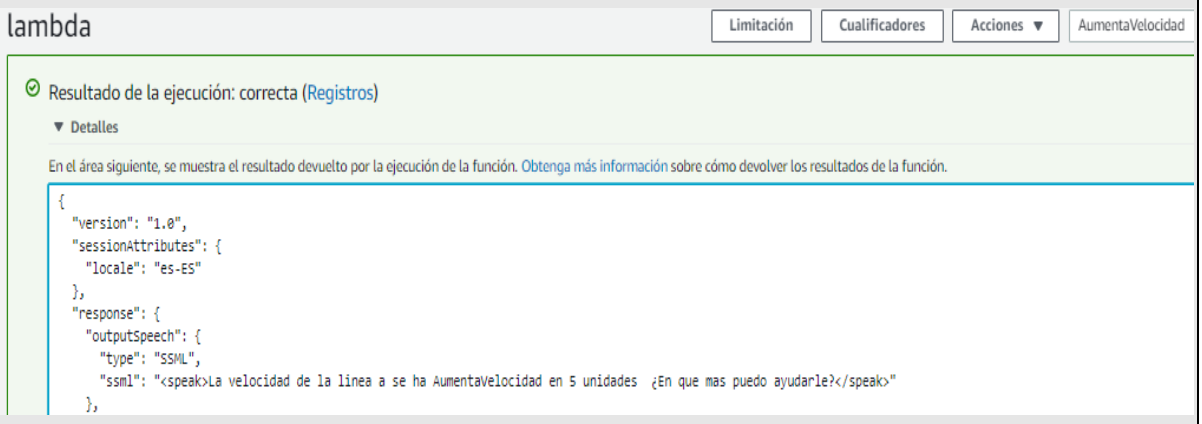


Tabla 7 Validación Lambda Aumenta velocidad línea trabajo

Validación Lambda Termina Orden de trabajo	Código del CP	CP-8

Descripción:

Validación de la función lambda que interpreta la Intent de la skill de Alexa AcciónSobreOrdendeTrabajo.

Prerrequisitos

3. Función lambda desplegada en el nodo especifico de Amazon AWS
4. La máquina de la Línea A está ejecutando la orden de trabajo 2

Pasos:

5. Invocar el evento de prueba de invocación de la skill
6. Invocar el evento de prueba de AcciónSobreOrdendeTrabajo

Resultado esperado:

La función lambda interpreta la orden enviada a través de la skill, publica el mensaje especifico en el broker de comunicaciones y construye el mensaje textual para enviárselo de vuelta a la skill para informar al usuario de la ejecución correcta de la función.

Resultado obtenido:

El test se ejecuta correctamente, el mensaje es publicado en el broker de comunicaciones y el mensaje de respuesta a la skill se construye correctamente, las imágenes siguientes muestran las evidencias del resultado de los test

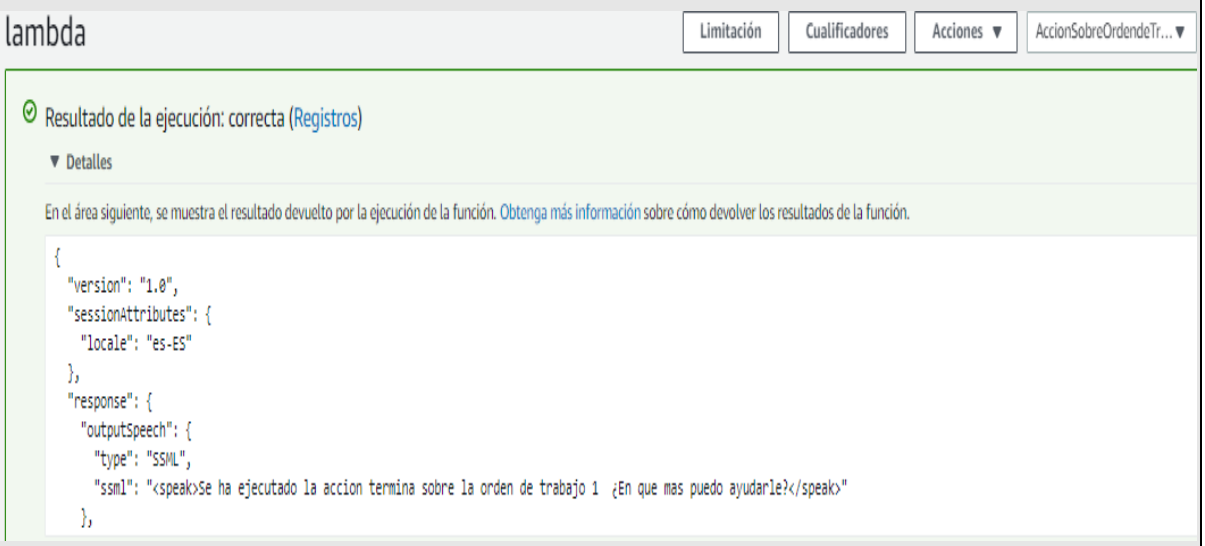


Tabla 8 Validación Lambda Termina Orden de trabajo

Validación Lambda Añade unidades a orden de trabajo	Código del CP	CP-9

Descripción:

Validación de la función lambda que interpreta la Intent de la skill de Alexa AñadeUnidades.

Prerrequisitos

5. Función lambda desplegada en el nodo especifico de Amazon AWS

Pasos:

7. Invocar el evento de prueba de invocación de la skill
8. Invocar el evento de prueba de AñadeUnidades

Resultado esperado:

La función lambda interpreta la orden enviada a través de la skill, publica el mensaje especifico en el broker de comunicaciones y construye el mensaje textual para enviárselo de vuelta a la skill para informar al usuario de la ejecución correcta de la función.

Resultado obtenido:

El test se ejecuta correctamente, el mensaje es publicado en el broker de comunicaciones y el mensaje de respuesta a la skill se construye correctamente, las imágenes siguientes muestran las evidencias del resultado de los test

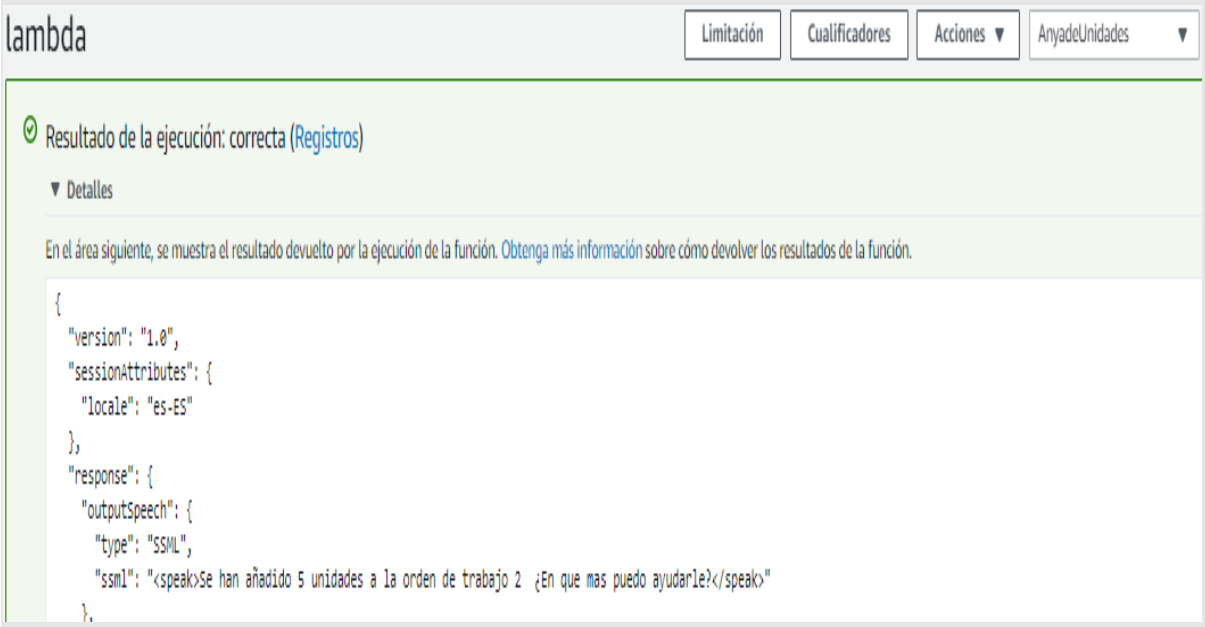


Tabla 9 Validación Lambda Añade unidades a orden de trabajo

8. **Conclusión**

Durante el siguiente capítulo se citan las diferentes conclusiones derivadas del desarrollo del proyecto. Además de la citación de las diferentes conclusiones, también se propone una visión de futuro presentando líneas de investigación futura y proyectos de mejora y/o complementarios en los que se puedan derivar del actual.

8.1 **Resumen del trabajo realizado**

En el entorno actual de competencia global, desarrollo tecnológico e innovación, las empresas, sobre todo de manufactura, se ven forzadas a reconfigurar sus procesos. La industria 4.0 y la manufactura inteligente son parte de una transformación, en la que las tecnologías de fabricación y de la información se han integrado para crear innovadores sistemas de manufactura, gestión y formas de hacer negocios, que permiten optimizar los procesos de fabricación, alcanzar una mayor flexibilidad, eficiencia y generar una propuesta de valor para sus clientes, así como responder de forma oportuna a las necesidades de su mercado (El Entorno de la Industria 4.0: Implicaciones y Perspectivas Futuras, 2017). El objetivo inicial del proyecto se centraba en aprovechar, dentro de una arquitectura modular, el potencial de una nueva generación de redes, plataformas de comunicación, disciplinas tecnológicas como la inteligencia ambiental y la computación autónoma para generar una solución que pudiera generar un primer avance o punto inicial para conseguir evolucionar desde “el proceso de automatización industrial” a la “industria IoT autónoma”.

Para conseguir esta evolución, las empresas han de transitar hacia arquitecturas tecnológicas que les permitan alcanzar mayores niveles de integración, como es la arquitectura diseñada a lo largo de este proyecto. Utilizando como base esta arquitectura, que se espera que las empresas que la apliquen dentro de su proceso productivo incrementen la autonomía de su manufactura, permitiendo una mayor y dinámica interacción de humanos y máquinas, un mayor intercambio y análisis de la información que permitirá por un lado gracias a la implantación de sistemas de computación autónoma y adaptativos la reconfiguración automática del proceso productivo detectando y corrigiendo o mitigando el impacto de situaciones indeseadas y consiguiendo un proceso productivo más flexible y eficiente. Gracias a la introducción de este tipo de arquitecturas tecnológicas, se espera que las empresas alcancen una evolución hacia sistemas productivos más autónomos, reportando un en corto periodo de tiempo beneficios que van desde el aumento de la productividad laboral y la colaboración, hasta una mayor eficiencia general del equipo, una mayor agilidad en el mercado y experiencias positivas para los clientes, aumento de seguridad en la empresa pivotando al personal más a oficina y reduciendo el tiempo de exposición de los empleados dentro de los principales lugares de riesgo dentro de la empresa.

Una vez diseñada la solución basada en la arquitectura tecnológica y con el propósito de validarla antes de involucrarse en ningún desarrollo importante, ha sido conveniente la validación inicial de la idea de una forma funcional. Pudiendo estar lejos de lo que puede ser un producto final en tamaño, prestaciones, consumo y precio, el prototipo permite tener una primera aproximación de la solución. Esto puede resultar de gran valor tecnológico e incluso financiero, ya que su existencia argumenta las razones por las que invertir en su desarrollo. El prototipo se ha implementado

utilizando soluciones tecnológicas comerciales actuales centrado el desarrollo dentro de la suite IoT que ofrece Amazon AWS, el asistente de voz Alexa junto a su funcionalidad de skill. Para la comunicación con las máquinas de producción y ante la imposibilidad actual de tener un controlador inteligente que comunique con la solución propuesta, se ha desarrollado un simulador de este para poder validar la solución completa.

Para el desarrollo del prototipo, se ha tomado como empresa objetivo la empresa Embalpack. Se han estudiado y modelado los diferentes componentes que interfieren en el proceso productivo y se ha diseñado una solución para introducir la arquitectura propuesta dentro del ámbito de esta empresa. Para ello, se ha definido un modelo de dialogo que permite a los diferentes operarios de la fábrica interactuar de manera natural con las diferentes máquinas de la factoría y emitir las diferentes órdenes para la gestión del proceso productivo de manera sencilla y rápida. Una vez definido el dialogo, se ha implementado un dialog model dentro de una skill de Alexa para que el asistente de voz virtualizado sepa controlar los diferentes pasos del dialogo y convertir las ordenes vocales, a órdenes directas a las controladoras de la maquinas a través del broker de comunicaciones. Para poder adaptar la solución propuesta a la empresa seleccionada, como se ha presentado en el capítulo de implementación, se han emulado las controladoras de las máquinas de producción puesto que estas, todavía no están preparadas para la comunicación con la arquitectura de comunicación propuesta en el proyecto. Para una mayor monitorización del estado del proceso productivo, y dotando al sistema de una mayor interacción con el usuario final a través de diferentes interfaces multimodales, en la herramienta de reporting se ha introducido un sintetizador de voz que permite ofrecer a nivel vocal el reporte periódico de las diferentes controladoras de voz si este lo desea. El prototipo desarrollado para la empresa Embalpack, sirve para validar y demostrar que la solución arquitectónica propuesta se puede adaptar al proceso productivo, dotando a la empresa de una arquitectura de comunicación y monitorización que permitirá sacarle el máximo rendimiento al personal y a las líneas de producción y minimizar las pérdidas de materia prima, reduciendo al máximo las paradas de línea debido a mantenimientos correctivos.

La solución propuesta, junto al prototipo presentado, quedan en el ámbito de una solución a nivel conceptual, sentando las bases de un sistema de gestión semi autónomo, incorporando un sistema de comunicación bidireccional entre humanos y dispositivos IOT aplicados a la gestión de máquinas de producción dentro del ámbito de la industria 4.0. Para una total industrialización del sistema propuesto, diferentes ámbitos han de ser estudiados y tomados en cuenta para una implantación correcta y segura de la solución en cualquier industria del tejido productivo, tanto a nivel nacional como internacional. La ciberseguridad es un ámbito importante para la total industrialización de esta solución, para ello es necesario conocer las vulnerabilidades de las diferentes redes de producción de la empresa objetivo y las herramientas utilizadas por los posibles atacantes, lo que permitirá analizar, prevenir y detectar los ataques y reparar los daños que se hayan producido de una manera rápida. Dicha ciberseguridad se constata como un elemento habilitador para que la Industria 4.0 pueda seguir creciendo. Tiene que proteger tanto los componentes tecnológicos, como los procesos que forman parte de su negocio. La ciberseguridad debe verse dentro de la Industria 4.0 como un mecanismo de protección, pero también como un elemento clave para la continuidad de la implantación de este tipo de solución en el tejido industrial.

Además de tener en cuenta la ciberseguridad, partiendo de esta solución propuesta, y en línea con las ideas introducidas, el proceso productivo del futuro requerirá la implantación de nuevos desarrollos tecnológicos orientados cada vez más orientados a la producción y gestión autónoma, por ejemplo, modelos de integración más interoperables, inteligentes, adaptables, y distribuidos bajo una arquitectura orientada al servicio y el cómputo en la nube.

8.2 **Trabajo futuro**

A lo largo de este documenta se expone una arquitectura de control que puede aplicarse con una capa de personalización extra a cualquier proceso productivo de una empresa en un futuro cercano, dentro del paradigma creciente de Industria 4.0. Existen algunas mejoras que podrían producirse en trabajos futuros como es la introducción de mayor capacidad de adaptabilidad y autonomía, introduciendo otro tipo de técnicas emergentes como son la inteligencia artificial o el big data, permitiendo que el sistema vaya aprendiendo poco a poco del proceso productivo diario incrementando la eficiencia de este de manera autónoma.

Como áreas de investigación futura se podría profundizar en el conocimiento de cada una de las tecnologías de la industria 4.0, sus implicaciones y su relación con la manufactura. De igual manera, explorar otras iniciativas de impulsar la Industria 4.0 y revisar casos de éxito que permitan continuar mejorando y expandiendo las ventajas de la adopción de este sistema de gestión del proceso productivo.

Por otro lado, para futuras investigaciones, se debe tener en cuenta la seguridad de la información por medio de encriptaciones que permitan la confidencialidad de los datos y la protección de estos frente a terceros. Se deben establecer protocolos de seguridad para el acceso a la información. Se deben estandarizar las tecnologías para permitir la integración de todos los elementos por medios de diferentes plataformas, independiente del software o hardware con los que cuenten, estudiando los enfoques y mecanismos para apoyar la interoperabilidad entre los componentes. Adicionalmente, el desarrollo de diversos algoritmos para el procesamiento de datos procedentes de la cadena de suministro para proporcionar una mejor comprensión del funcionamiento de la cadena de suministro global. La transparencia de las acciones informáticas empleando soluciones de diseño para hacer visible esta información, como indicadores claros de que acciones se llevan a cabo y por qué.

8.3 **Valoración personal**

En el diseño de la arquitectura propuesta, se ha conseguido fusionar dentro de una arquitectura de comunicación de tipo broker diferentes disciplinas aprendidas a lo largo del master concretamente en las asignaturas de DIM (Diseño de interfaces multimodales), IoT (Internet of Things) y SUA (Sistema ubicuos y adaptativos). A través de la utilización de estas disciplinas se ha diseñado un marco de gestión descentralizado y autónomo para las futuras industrias inteligentes. Estas fábricas modulares inteligentes podrán habilitar autónomamente la descentralización y automatización de las decisiones relativas a la producción, así como ser capaces de comunicarse y cooperar a través de IoT con operadores humanos y otras fábricas inteligentes para completar un cambio de suministro vertical u horizontal completo.

Se debe tener en cuenta la repercusión de la implantación de un sistema de estas características si se considera el hecho de que se conseguiría incrementar el proceso productivo a un mayor nivel de eficiencia y a intervenciones más predictivas que minimizarían los procesos de reparación correctivos, un problema principal en el que las industrias de todo el mundo están luchando hoy en día.

Este proyecto me ha dado una base que complementa mi formación universitaria, tanto a nivel profesional como a nivel personal, dado que ha sido un gran reto poder enfrentarme a un proyecto de esta envergadura y que implique diferentes conceptos. De esta forma, he dedicado todo mi esfuerzo y dedicación para intentar realizar de la mejor manera este trabajo intentando plasmar algunos de los aspectos más importantes que he aprendido a lo largo el master.

Por último, quiero agradecer muy especialmente a, Joan y Vicente quienes me apoyaron y me dieron confianza en el trayecto y apoyo en los días más difíciles con la redacción del trabajo. También agradecerle que me aceptara como alumno y compartiera mi visión del proyecto para realizar mi trabajo final de master supervisado por ellos.

9. **Bibliografía**

A survey of formal methods in self-adaptive systems. **Weyns, Danny & Iftikhar, M. & Gil de la Iglesia, Didac & Ahmad, Tanvir. 2012.** s.l. : ACM International Conference Proceeding Series, 2012.

Altimir, Jordi Ribera. Inteligencia Ambiental. *https://www.cs.upc.edu/.* [En línea] [Citado el: 02 de 06 de 2020.] https://www.cs.upc.edu/~bejar/ia/material/trabajos/Inteligencia%20Ambiental.pdf.

Álvaro Hernández, Rubén Fernández, Luis A. Hernández. *Arquitectura para la evaluación de la usabilidad de interfaces multimodales.* Madrid : s.n.

Amazon. Alexa Skills Kit. [En línea] [Citado el: 21 de 11 de 2019.] https://developer.amazon.com/es-ES/alexa/alexa-skills-kit.

Amazon AWS. Políticas administradas y políticas insertadas. [En línea] https://docs.aws.amazon.com/es_es/IAM/latest/UserGuide/access_policies_managed-vs-inline.html.

Amazon Inc. What Is AWS Lambda? [En línea] https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

AWS Amazon. Amazon Virtual Private Cloud. [En línea] [Citado el: 18 de 11 de 2019.] https://aws.amazon.com/es/vpc/.

Barr, Alistair. Amazon buys text-to-speech software company Ivona. [En línea] [Citado el: 25 de 11 de 2019.] https://www.reuters.com/article/us-amazon-ivona/amazon-buys-text-to-speech-software-company-ivona-idUSBRE90N0T020130124.

Boehm, Barry. 1986. *ACM SIGSOFT Software Engineering Notes.* s.l. : ACM, 1986.

CATANIA, SIMONE. IoT e Industria 4.0 ¿cual es la diferencia? [En línea] [Citado el: 02 de 01 de 2020.] https://www.noticias.ltda/sociedad-digital/iot-e-industria-40-diferencia/.

Chowdhury, Gobinda G. 2003. Natural Language Processing. *Annual Review of Infomation Science and Technology.* Glasgow : s.n., 2003.

Dr. Dragan Stokic, Dr. Uwe Kirchoff, Mr. Harald Sundmaeker. Ambient intelligence in manufacturing industry: Control system point of view. [En línea] http://www.ami4sme.org/files/AMI4SME_IASTED_Paper.pdf.

El Entorno de la Industria 4.0: Implicaciones y Perspectivas Futuras. **Carmen Berenice Ynzunza Cortés, Juan Manuel Izar Landeta y otros. 2017.** 54, AguasCalientes : s.n., 2017, Vol. 1.

infoplc. infoPLC. [En línea] [Citado el: 18 de 11 de 2019.] https://www.infoplc.net/plus-plus/eventos-ferias/item/106890-isa-spain-convoca-a-una-reunion-tecnica-sobre-mqtthttps://www.infoplc.net/plus-plus/eventos-ferias/item/106890-isa-spain-convoca-a-una-reunion-tecnica-sobre-mqtt.

ISO/IEC. 2016-06. Message Queuing Telemetry Transport (MQTT). 2016-06.

Juraj Kačur, Gregor Rozinaj, Renata Rybárová. Primera edición, 2017. *MMI (interfaces multimodales) para la comunicación y navegación interactiva.* Praga, Czech Republic :

European Virtual Learning Platform for Electrical and Information Engineering, Primera edición, 2017.

López, José Manuel Molina. 2017. *Modelo de Privacidad Digital en Inteligencia Ambiental basado en Sistema Multiagente.* Leganes : s.n., 2017.

María Gómez, Joan Fons, and Vicente Pelechano. Evolución de Sistemas Auto-Adaptables mediante Modelos en Tiempo de Ejecución. [En línea] [Citado el: 05 de 02 de 2020.] https://biblioteca.sistedes.es/wp-content/uploads/2016/05/JISBD-2012-537-Evoluci%C3%B3n-de-Sistemas-Auto-Adaptables-mediante-Modelos-en-Tiempo-de-Ejecuci%C3%B3n.pdf.

Mvelegon. Overview of the AVS Device SDK. [En línea] https://github.com/alexa/avs-device-sdk.

Sidorov, Grigori, Ibarra Romero, Martín, Markov, Ilia, Guzman-Cabrera, Rafael, Chanona-Hernández, Liliana, & Velásquez, Francisco. 2016. Detección automática de similitud entre programas del lenguaje de programación Karel basada en técnicas de procesamiento de lenguaje natural. *Computación y Sistemas.* 279-288 : s.n., 2016.

Soni, Dipa & Makwana, Ashwin. 2017. *A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT).* 2017.

The computer for the 21st Century. **Weiser, Mark. 2002.** 2002, PERVASIVE Computing, págs. 18-25.

Wikipedia. .Net Core. [En línea] [Citado el: 08 de 11 de 2019.] https://es.wikipedia.org/wiki/.NET_Core.

—. Microsoft Speech API. [En línea] https://es.wikipedia.org/wiki/Microsoft_Speech_API.

10. **Anexos**

10.1 **Estructura general Skill Alexa**

Alexa es el servicio de voz ubicado en la nube de Amazon disponible en los dispositivos de Amazon y dispositivos tercios con Alexa integrada. Además, cuenta con funcionalidades, o lo que Amazon llama Skill, que permiten a los consumidores crear una experiencia más personalizada. El Alexa Skills Kit (ASK) es un conjunto de herramientas, documentación, muestras de código y API en self-service con el que puedes añadir Skills a Alexa de forma rápida y sencilla. El ASK permite a diseñadores, desarrolladores y marcas crear Skills atractivas y llegar a los consumidores. Con este kit, puedes aprovechar el conocimiento y la innovación de Amazon en el sector del diseño de voz (Amazon).

Antes de comenzar el diseño de nuestra skill debemos determinar su propuesta de valor y cómo contribuirá a la solución eficaz y satisfacción de la necesidad del cliente y el negocio, además de definir las pautas en las que la skill ayudará a la mejora de la experiencia del usuario. Al diseñar y desarrollar una habilidad personalizada, se deben definir varios componentes dentro de ella para poder aprovechar toda la capacidad que nos ofrece el ASK de Alexa. Los componentes mínimos que crear son los siguientes:

- Un conjunto de “Intens” que representan acciones que los usuarios pueden hacer con su habilidad. Estas intenciones representan las funcionalidades principales definidas en la skill.
- Un conjunto de “utterances” que especifican las palabras y frases que los usuarios pueden decir para invocar los intents anteriores. Para cada uno de los intents se pueden definir varias palabras o frases similares que la skill mapeara con el intent especifico. Este mapeo crea el llamado interaction model para la skill
- Un nombre de invocación que identifica la skill. El usuario incluye este nombre cuando inicia una conversación con su habilidad.
- Si corresponde, un conjunto de imágenes, archivos de audio y archivos de video que desea incluir en la habilidad. Estos deben almacenarse en un sitio de acceso público para que cada elemento sea accesible mediante una URL única.
- Un servicio basado en la nube (función lambda) que acepta los mensajes desde la skill interpretando los intents como solicitudes estructuradas y luego actúa sobre ellas. Este servicio debe ser accesible y mapeado a través de un una URL denominada endpoint.

Una vez definida la skill, debemos elegir con qué tipo de servidor trabajará la skill, es decir, si lo hará mediante un servidor propio o desde Amazon, por ejemplo, AWS Lambda. En el caso de seleccionar la segunda opción será necesario acceder con la cuenta de Amazon a la que están vinculados los asistentes de voz virtuales y skill y actualizar con el endpoint.

Ilustración 46 Arquitectura de interacción entre usuario Alexa skill y función externa

10.2 **Toolkit for Visual Studio**

AWS Toolkit for Visual Studio es una extensión para Microsoft Visual Studio que se ejecuta en Microsoft Windows y que facilita a los desarrolladores el desarrollo, la implementación y la depuración de aplicaciones .NET mediante Amazon Web Services. Con AWS Toolkit for Visual Studio, podrá comenzar a trabajar más rápido y aumentar sus niveles de productividad a la hora de crear aplicaciones AWS.

A través de esta extensión de Microsoft Visual studio se puede desarrollar y desplegar de manera más sencilla una función lambda dentro del entorno de Amazon AWS. La siguiente imagen muestra como generar un proyecto de AWS Lambda.

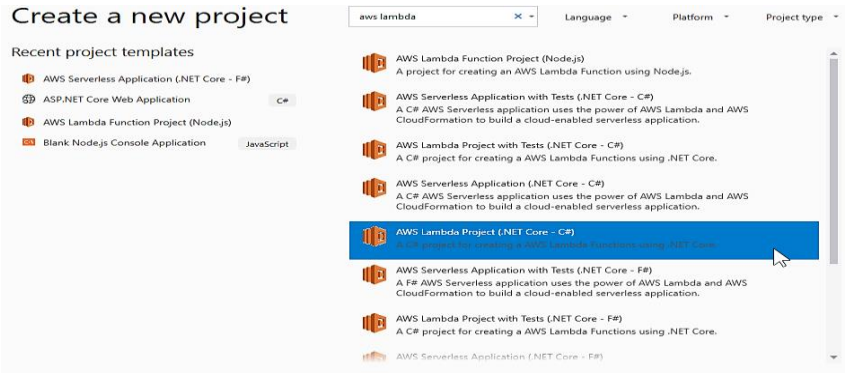


Ilustración 47 Plantilla de desarrollo función AWS Lambda

Una vez creado el proyecto, como parte de este, se genera un fichero llamado “aws-lambda-tools-defaults.json”. A través de este fichero, se puede configurar las opciones que las herramientas Lambda leen de forma predeterminada para llevar a cabo su ejecución y despliegue. El plugin genera una plantilla de proyecto creada en Visual Studio donde se establecen muchos de estos campos con valores predeterminados. En este se especifica el manejador de funciones, razón por la cual no tiene que configurarlo en el asistente. Pero si cambia el nombre de la función, clase o ensamblaje, deberá actualizar el campo en el archivo “aws-lambda-tools-defaults.json”.

El fichero modificado para el proyecto SmartIoT4Industry tiene el siguiente aspecto

```
{
  "Information" : [
    "This file provides default values for the deployment wizard inside Visual Studio and the AWS Lambda commands added to the .NET Core CLI.",
    "To learn more about the Lambda commands with the .NET Core CLI execute the following command at the command line in the project root directory.",
    "dotnet lambda help",
    "All the command line options for the Lambda command can be specified in this file."
  ],
  "profile"     : "default",
  "region"      : "us-east-1",
  "configuration" : "Release",
  "framework"     : "netcoreapp2.1",
  "function-runtime" : "dotnetcore2.1",
  "function-memory-size" : 256,
  "function-timeout"     : 10,
  "function-handler"     : "lambda::lambda.Function::FunctionHandler",
  "function-name"        : "lambda",
  "function-role"        : "arn:aws:iam::603417540615:role/lambda_exec_lambda",
  "environment-variables" : "",
  "tracing-mode"          : "PassThrough"
}
```

Ilustración 48 Ejemplo de fichero aws-lambda-tools-defaults.json

Una vez programada la función lambda, a través de este kit de desarrollo, se puede desplegar automáticamente la función dentro de la capa de AWS lambda. En la página Cargar función Lambda, en Nombre de la función, escriba un nombre para la función o seleccione una función publicada anteriormente para volver a publicar. Luego elija Siguiente. La siguiente pantalla de configuración avanzada se deben indicar los siguientes campos:

- Role Name: El obligatorio definir un nombre de un rol asociado con su cuenta. Se tiene que elegir entre un rol existente o uno nuevo basado en una política administrada de AWS o su propia política administrada. Para más información acerca políticas administradas dirigirse a (Amazon AWS). El rol se utiliza para proporcionar credenciales para cualquier llamada de servicio de AWS realizada por el código en la función. La cuenta definida debe tener permiso para ejecutar la acción de “IAM ListPolicies”.
- VPC : si la función Lambda accede a recursos en una VPC de Amazon, sirve para seleccionar las subredes y los grupos de seguridad. Para más información acerca de VPC de Amazon diríjase a (AWS Amazon).
- KSM Key: Parámetro opcional en el cual se definen las claves de cifrado KSM. KMS es un servicio administrado que se puede usar para crear y controlar las claves de cifrado utilizadas para cifrar sus datos. Si se tiene en propiedad una clave de tipo AWS KMS, puede seleccionarla de la lista.

Una vez definidos los campos descritos anteriormente y pulsando sobre el botón de “Upload” y si la validación de información es correcta, la función lambda se despliega automáticamente y estará accesible inmediatamente para su uso como muestra la siguiente imagen.

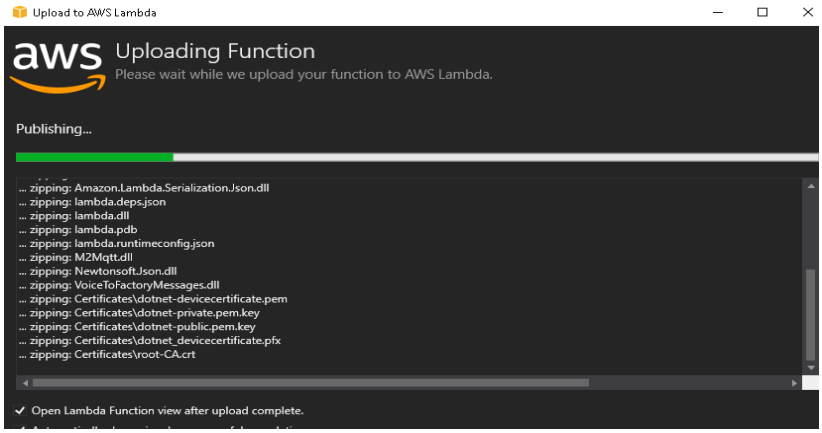


Ilustración 49 Plugin Visual Studio para despliegue de función lambda