Universidad Zaragoza
1542

Trabajo Fin de Grado

# Análisis e implementación de una arquitectura IoT sobre controladoras de soldadura

# Analysis and implementation of an IoT architecture on welding controllers


Autor

Priscila Gómez Lizaga

Director

Antonio Novo Guerrero

Ponente

Dr. Antonio Miguel Artiaga

ESCUELA DE INGENIERÍA Y ARQUITECTURA



2019

# AGRADECIMIENTOS

A Izarbe,
mi pequeña compañera.

# Análisis e implementación de una arquitectura IoT sobre controladoras de soldadura

## RESUMEN

En este trabajo de final de carrera se van a explotar las diferentes opciones que hay disponibles para establecer conexión con cuatro controladoras de soldadura en un entorno empresarial real. Bien utilizando software propio de cada empresa fabricante de los controladores o con aplicaciones de software libre, se analizará la viabilidad de ser capaces de comunicarnos con cada controlador con el fin recoger los datos que se generan en cada proceso de soldadura. En esta fase también se propone una arquitectura, basada en la ya propuesta para otros proyectos de Opel-Vauxhall que se engloban dentro del proyecto Welding Wizard, para cada tipo de controladora de soldadura teniendo en cuenta las necesidades de cada una. A continuación, se examinan las base de datos generadas por cada dispositivo y, dentro de cada una, se intenta detectar las que pueden resultar de más interés. Finalmente, se aplican algoritmo de aprendizaje supervisado como primera aproximación al objetivo de poder predecir fallos en un futuro.

# Índice

# Capítulo 1

# Introducción y objetivos

Hace un tiempo, solo se encontraban conectados a la red los ordenadores, pero con el paso de los años esto ha ido cambiando. El número de dispositivos que se conectan crece de forma exponencial y esto ha abierto la puerta al concepto denominado Internet of Things (IoT).

Normalmente, cuando se habla de IoT pensamos en relojes inteligentes o en controlar la aspiradora desde el teléfono, pero ¿y si aprovechamos esta idea para conectar dispositivos de una fábrica, de la tal forma, que nos permita conocer su estado actual o monitorizar sus parámetros en tiempo real? A esta nueva revolución industrial se le conoce como Industria 4.0.

La pregunta anterior motivó la colaboración entre el Grupo PSA y la Asociación Investigación, Desarrollo e Innovación en Aragón (IDia)[1].

## 1.1. Motivación

IDia es un clúster de empresas que tiene como objetivo impulsar la ejecución de proyectos de carácter innovador en el sector científico o tecnológico, por lo que la necesidad del Grupo PSA de encontrar la forma óptima de conectarse a los dispositivos de soldadura, conocer su estado y aprovechar dicho conocimiento para optimizar los parámetros y que mejorase su rendimiento, supuso un reto que decidió afrontar IDia.

El punto de partida en un escenario industrial donde existe una gran diversidad de dispositivos de soldadura que forman parte de una cadena de producción que no solo se ubican en Zaragoza, sino también en Ellesmere Port o Kaiserslautern, por ejemplo. Para el momento en el que yo pasé a formar parte del proyecto, ya se había establecido conexiones con algunos dispositivos de soldadura y debido a que el proyecto resultaba de interés para la compañía, se decidió estudiar la viabilidad de conexión a otros tipos de soldadura.

## 1.2. Propósito y alcance del proyecto

La posibilidad de ser capaces analizar los datos de cada dispositivo supondrá un beneficio, como ya se ha visto en las fases anteriores del proyecto, en términos de ser capaces de reducir el tiempo de paro de los equipos mediante la predicción de ciertas averías, mejorar el rendimiento de los equipos reducir potenciales problemas de calidad.

Al ser una prueba de concepto, este trabajo de final de grado se puede dividir en dos tareas principales. En la primera tarea será necesario conocer los diferentes caminos a través de los cuales nos podremos conectar a los dispositivos para obtener todos los datos que generan cada controlador de soldadura. Esta fase terminará cuando se ha decidido la forma mediante la cual se realiza la conexión, se haya comprobado que funciona y se hayan conseguido extraer los parámetros que genera cada dispositivo. Mientras que la segunda consistirá en el análisis de los datos para ser capaces de detectar los que resultan más interesantes de almacenar para su uso futuro. Esta fase se puede realimentar la primera, debido a que en algunos dispositivos existe la opción de seleccionar que información se desea extraer y de esta forma se podrá reducir el ancho de banda a transmitir, característica que es de gran importancia cuando se conecten todos los dispositivos existentes en planta. Además, se implementarán técnicas de procesado sobre los datos con el objetivo de enriquecer la información que ha generado cada controlador.

Debido a que la prueba de concepto se engloba dentro de un proyecto ya existente, el trabajo desarrollado en este trabajo de final de carrera ha de tener en cuenta los siguientes puntos:

(i) Se busca depender en la menor medida de software de terceros.

(ii) Se va a intentar utilizar tecnologías de librería abierta.

(iii) Se implementará de tal forma de que sea posible escalar el proyecto.

## 1.3. Estructura del documento

El documento se organiza en cinco capítulos donde se explica en qué consiste el proyecto incluyendo al final del mismo la bibliografía utilizada, la lista de figuras, el glosario y los anexos correspondientes.

En el primer capítulo se realiza una introducción a la prueba de concepto se va a desarrollar y la motivación que implicó llevarlo a acabo. Se habla de los objetivos que se plantean cumplir una vez finalizado el proyecto.

El segundo capítulo consiste principalmente en una explicación detallada de los conceptos más relevantes dentro del proyecto, de las aplicaciones que se van a utilizar, se describen los *softwares* que se pretenden emplear. También se da una pequeña introducción de los algoritmos de aprendizaje automático, como son: árboles de regresión, bosque aleatorio y XGBoost.

En el capítulo tres es donde se explica de forma detallada los dispositivos de soldadura en los que recae nuestro interés. El peso del trabajo de fin de grado recae en este apartado, ya que es donde se describen los diferentes caminos que ofrece cada dispositivos para establecer conexión con él; así como los problemas a los que nos tuvimos que enfrentar.

En el cuarto capítulo se realizan pruebas experimentales sobre los datos que se consiguieron recoger de las controladores de soldadura. Se espera que las conclusiones obtenidas tras este experimento permitan conocer un poco más el conjunto de datos almacenado.

Finalmente, en el quinto capítulo, se habla de las conclusiones a las que se ha llegado una vez finalizado el proyecto.

# Capítulo 2

# Conceptos subyacentes

Durante el proceso de investigar las diferentes formas de establecer conexión con cada uno de los dispositivos, nos dimos cuenta de que pese a que uno de los objetivos principales es utilizar software libre en ocasiones no iba a ser posible. Esto se debe a que en varias empresas ya se habían planteado la posibilidad de ofrecer los servicios de recolección y análisis de los datos.

Por este motivo la única forma de establecer una conexión era adquiriendo el software de las empresas. Esto cambio la dirección en la que se había planteado hasta ahora el problema puesto que en la mayoría de los dispositivos donde ya se ha solventado esta problemática, permitían el uso de programas o aplicaciones de software libre.

## 2.1. Internet of Things

Tal y como se define el libro *The Internet of Things*[2], IoT es un paradigma novedoso que está ganando terreno rápidamente en el escenario de las telecomunicaciones inalámbricas. La idea básica es la capacidad de interactuar con los diferentes dispositivos físicos, mediante esquemas de direccionamiento, con el objetivo de convertirlo en un dispositivo inteligente.

En el caso de este proyecto, los dispositivos inteligentes son los controladores de soldadura a través de los cuales se obtendrán lo diferentes datos que permitirán conocer los parámetros que se generan en los procesos de soldadura y analizarlos con el objetivo de realizar un mantenimiento predictivo.

### 2.1.1. Protocolos IoT

Un protocolo de comunicación es un sistema con reglas que permite que dos o más entidades se comuniquen entre ellas para transmitir información. Se pueden implementar por hardware, software, o por una combinación de ambos.

Teniendo en cuenta que los controladores de soldadura, dispositivos con los que se quiere establecer conexión, son dispositivos pasivos es necesario que el envío de mensajes sea asíncrono. Entre las diferentes opciones existentes, cabe destacar los protocolos Advanced Message Queuing Protocol (AMQP) y Message Queue Telemetry Transport (MQTT).

Las principales diferencias entre los dos protocolos anteriores es que mientras MQTT se basa principalmente en la publicación/suscripción, lo cual es rápido y fácil de implementar, AMQP es más complejo debido a que se creó para que destaque por su fiabilidad e interoperabilidad, características que se refleja, por ejemplo, en el tamaño de su cabecera (AMQP con un tamaño mínimo de 60 bytes frente a los 2 bytes de MQTT en el mejor de los casos).

Por lo tanto, se decide utilizar el protocolo de comunicación MQTT.

### 2.1.2. Protocolo MQTT

Se trata de un protocolo desarrollado por IBM, implementado sobre la capa TCP/IP y basado en una comunicación asíncrona de mensajes. Entre las principales características se encuentran:

- Los mensajes que envía son de tamaño reducido debido al pequeño tamaño de las cabeceras.
- Es posible utilizarlo cuando el ancho de banda es mínimo.
- Requiere pocos recursos en cuanto al procesado y a la memoria.
- Soporta diferentes lenguajes de programación a través de implementaciones de código abierto.

La arquitectura MQTT (figura 2.1) sigue una topología en estrella, donde existe un servidor, o *broker*, que se encarga de recibir los mensajes de los dispositivos emisores y distribuirlos a los receptores.

MQTT utiliza un modelo de publicación y suscripción basado en *topics* (temas). Un *topic* es un grupo virtual de mensajes. Cuando un cliente publica un mensaje en un topic especifico, los mensajes se entregan a todos los clientes que se han suscrito a dicho topic. Además, tiene una estructura jerárquica gracias a la cual se permite establecer relaciones padre-hijo, es decir, al suscribirnos a un topic padre, también recibimos la información de sus hijos.

Otra característica importante de MQTT es la fiabilidad, entiéndase como la garantía de entrega de los mensajes. Este protocolo permite que el cliente sea quien

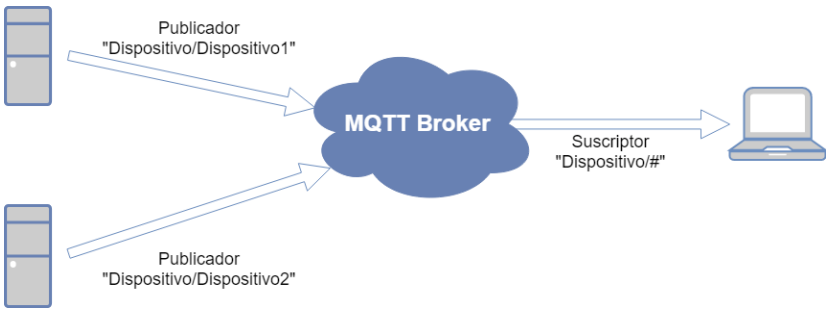



Figura 2.1: Ejemplo de arquitectura para el protocolo MQTT. El publicador envía mensajes al *topic* "Dispositivo/[Nombre del dispositivo]". El suscriptor, mediante el símbolo #, recibe los mensajes que publiquen ambos dispositivos

elija la calidad del servicio a través del parámetro Quality of Service (QoS) que puede tomar diferentes valores:

- 0: se entrega el mensaje como mucho una vez y sin garantía de recepción.
- 1: se entrega el mensaje al menos una vez.
- 2: se garantiza la entrega y recepción del mensaje.

### 2.1.3. RabbitMQ

Teniendo en cuenta que se necesita un servidor que se encargue de la gestión de colas y de la comunicación entre los diferentes clientes, RabbitMQ se postula como el gestor más ligero y fácil de implementar. Entre sus principales funcionalidades encontramos:

- Gestión asíncrona de mensajes.
- Está preparado para su implementación en sistemas distribuidos, una característica que permitirá la escalabilidad del proyecto en un futuro.
- Monitorización mediante una interfaz gráfica.

Una importante limitación de este *broker* se debe a la calidad de servicio, QoS. A pesar de que el protocolo MQTT permite una calidad de servicio de nivel 2, RabbitMQ solo permite configurar como máximo una calidad de servicio de nivel 1.


Figura 2.2: RabbitMQ como *broker*.

### 2.1.4. Mosquitto

Se trata de un *broker* ampliamente utilizado debido a la ligereza que brinda, característica que permite emplearlo en un gran número de ambientes independientemente de los recursos que haya disponibles.

Para comprobar su correcto funcionamiento y realizar las pruebas se ejecutaron directamente sobre la línea de comandos las siguientes ordenes:

- *Mosquitto_pub*: permite publicar un mensaje en un determinado topic.

- *Mosquitto_sub*: permite monitorizar desde la línea de comandos el estado indicando el topic al que se desea acceder.

Se eligió este *broker* debido a que una de sus características por defecto es que no necesita autenticación en el cliente.


Figura 2.3: Mosquitto como *broker*.

### 2.1.5. Node-RED

Node-RED es una herramienta de programación visual de código abierto creada por IBM que sirve para comunicar hardware y servicios de una forma rápida y sencilla.

Se creó a partir de NodeJS[3], quien proporciona la potencia suficiente para que Node-RED sera fiable y escalable. Además, D3.js[4] se encarga de la interfaz web donde se pueden crear flujos de eventos e interconectar los nodos para crear un flujo de datos. Cabe destacar la amplia gama de nodos que hay disponibles y que se gestionan mediante el ecosistema Node Package Manager (NPM).

De entre todos los nodos que tenemos disponibles, se puede destacar la utilidad de un nodo que permite suscribirse a un topic mediante el protocolo MQTT. Mediante este nodo nos podemos conectar de forma remota a un dispositivo que se encuentra en alguna de las plantas.

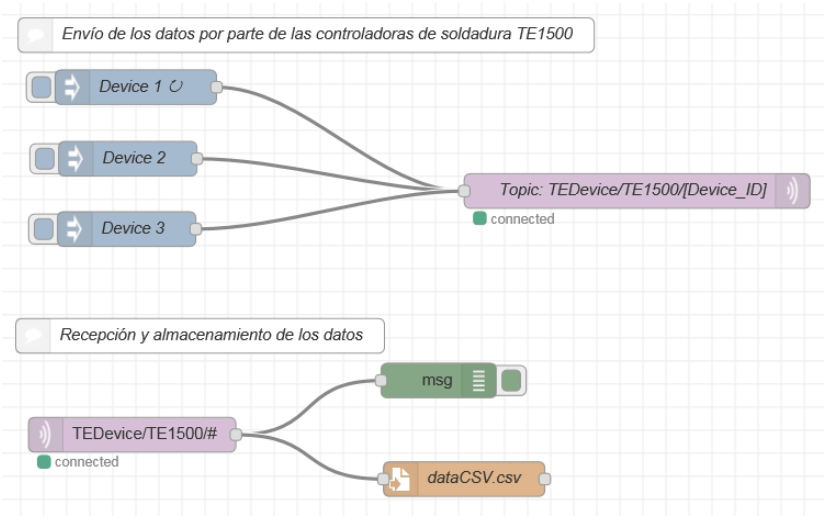



Figura 2.4: Ejemplo de un flujo de captura de datos en Node-Red. Se representan tres nodos con nombre [DeviceID] que simulan los controlaodres de soldadura publicando a un topic. Los clientes se suscriben al topic y almacen los datos en un fichero CSV.

### 2.1.6. Node-RED

## 2.2. Almacenamiento de los datos

Una base de datos es una aplicación que se encarga de almacenar una colección de datos que se organiza de tal forma, que se pueda acceder rápidamente a los fragmentos de datos que se necesiten.

Actualmente casi todas las aplicaciones tecnológicas producen una gran cantidad de volumen de datos, lo que hace que las bases de datos sean un pilar fundamental en los sistemas. Las más utilizadas son las bases de datos relacionales.

Debido a que se trata de una prueba de concepto, se decidió no modificar el lugar por defecto en el que se almacenan los datos que generan los dispositivos. Esto implica que, a pesar de que uno de los objetivos es utilizar software de librería abierta, en esta fase habrá ocasiones que no será posible cumplirlo.

### 2.2.1. Base de datos relacional

Una base de datos relacional [5] es una colección de elementos de datos organizados en un conjunto de tablas totalmente descritas desde las que se puede acceder a los datos o volver a montarlos de muchas maneras diferentes sin tener que reorganizar las tablas de la base. La mayoría de las bases de datos relacionales utilizan el lenguaje de consulta estructurado (por sus siglas en inglés, Structured Query Language (SQL)) que permite un enfoque declarativo de la consulta.

Por lo tanto, para el proyecto, se a utilizar el sistema de gestión de base de datos

relacional *Microsoft SQL Server*[6], desarrollado por la empresa Microsoft.

## 2.3. Análisis de los datos

Una vez concluida la primera fase del proyecto se procede a realizar un análisis de los datos que se han recogido. Para esta fase se opta por realizar un primer estudio de los ficheros que se han recibido, puesto que es la primera vez que se capturan. Y, posteriormente, se realiza la primera aproximación a lo que en un futuro se conocerá como mantenimiento predictivo.

### 2.3.1. Algoritmos empleados

Se van a utilizar para el estudio métodos de aprendizaje supervisado no paramétricos, enfocados principalmente a la clasificación de una clase binaria la cual se divide entre la presencia o no de fallos en el dataset. El trabajo a realzar se enfoca desde un punto de vista de conocer las características más importantes de entre todas las que se recogen de las controladoras con el objetivo de predecir fallos.

También se aplicarán los modelos de regresión para los mismos métodos ya que se vio posteriormente que existe fallos que en realidad son avisos.

La elección de utilizar modelos basados en los árboles de decisiones [7] se debe a que en muchas ocasiones se utiliza como modelo selector de características importantes permitiendo reducir el número de columnas con las que se trabaja.

#### Arboles de decisión

Los árboles de decisiones [8] son uno de los modelos más sencillos debido a que solo construye un árbol en el cual se aplican diferentes reglas que permiten realizar predicciones. En cada paso, se toma una decisión basada en un atributo y, como resultado, se genera una nueva rama del árbol.

De una forma más detallada, un árbol de decisiones esta formado por un conjunto de nodos que se conoce con el nombre de nodos de decisión y otro conjunto que hace referencia a los nodos terminales.

- En cada nodo de decisión (nodo raíz o nodo interno) se asocia una característica del conjunto de datos de entrada. Para cada nodo se generan dos ramas, por las cuales se toma una dirección u otra en función de un límite que se establece en el nodo de decisión.

- En los nodos terminales se devuelve la decisión que ha tomado el árbol en función de los datos de entrada.

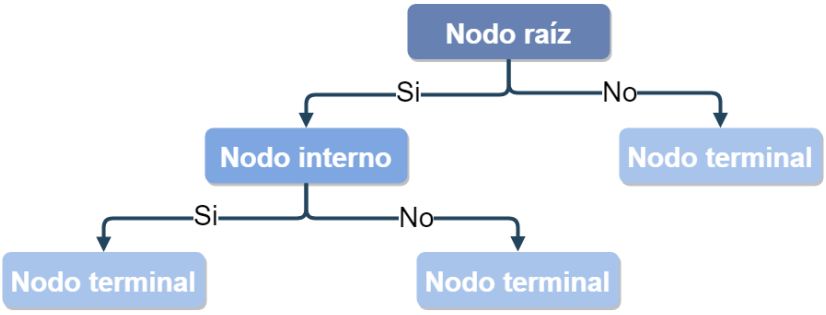



Figura 2.5: Diagrama ejemplo de un árbol de decisiones.

### Bosques aleatorios

El modelo [9] se genera mediante una combinación de un gran número de árboles de decisión. Mejora la precisión con la que clasifica gracias a la incorporación de aleatoriedad en la construcción de cada clasificador individual. La principal diferencia con los árboles de decisiones radica en que, en lugar de buscar la característica más importante al dividir un nodo de decisión, busca la mejor característica entre un subconjunto aleatorio de características. En cuanto a la precisión de acertar, cada árbol calculará de forma individual un resultado el cual se promediará y en función del resultado promedio se obtendrá la predicción del problema.

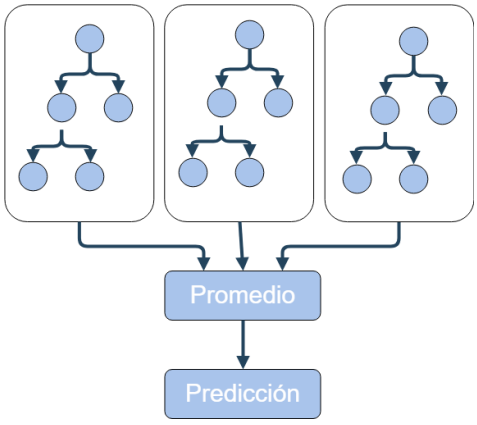



Figura 2.6: Diagrama ejemplo de un bosque aleatorio.

### XGBoost

Es necesario definir previamente el término *Boosting* el cual hace referencia a crear una regla de predicción altamente precisa combinando muchas reglas relativamente débiles e imprecisas. La idea principal es que realizando estas combinaciones se obliga a inferir algo nuevo sobre los datos cada vez que se llama al algoritmo.

El algoritmo de aprendizaje supervisado XGBoost[10], es una de las implementaciones más populares y eficientes de los algoritmos de árboles de decisión

con Gradient Boosting. La principal diferencia con respecto a los modelos anteriores (arboles de decisión y bosque aleatorio) es que los árboles se construyen de maneara secuencial, lo que conlleva a que cada árbol nuevo aprenda de sus predecesores y sea capaz de reducir los errores cometidos por los árboles anteriores.

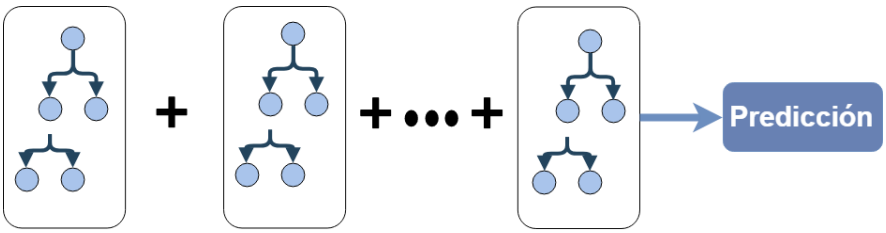



Figura 2.7: Diagrama ejemplo de XGBoost.

# Capítulo 3

# Establecimiento de conexión con las controladoras de soldadura

En esta primera fase, el objetivo que se pretende abordar consiste en averiguar la forma de establecer una conexión entre un PC y el controlador. Se ha dividido el capítulo en tantas secciones como controladores de soldadura existe debido a que cada uno se abordará de una forma diferente.

Debido a que fue necesario realizar un informe para Opel-Vauxhall ciertos detalles, como las configuraciones que fue necesario hacer, se explican en el Anexo D.

## 3.1. Antecedentes

Cuando se realizó la primera conexión entre los controladores de soldadura y la central donde se almacenarían los datos, se dieron cuenta de que con el paso del tiempo resultaría muy complicado almacenar toda la cantidad de datos que se recogían de los controladores en una base de datos, ya que no se realizaba ningún filtrado y se enviaban los datos incluso aunque no resultasen de interés. Esto, a su vez, suponía un problema en la representación de los datos pues resultaba imposible generar informes a tiempo real. Y, además, enviar todos los datos por la red al centro de almacenamiento central, suponía ocupar grande parte del ancho de banda disponible. Así, surgió la necesidad de reducir la cantidad de datos a enviar y a procesar, lo que implico rediseñar la arquitectura.

Posteriormente, la capacidad de generar informes a tiempo real generó la posibilidad de realizar un mantenimiento predictivo. Añadir esta opción resulto ser de gran interés ya que implicaba reducir el tiempo de paso de los equipos mediante la predicción de determinadas averías, mejorar el rendimiento de los equipos de soldadura, reducir problemas de calidad y, de forma implícita, se iba a obtener un repositorio con los mejores parámetros de soldadura.

## 3.2. Dispositivos de interés

Como ya se ha comentado con anterioridad, el proyecto Welding Wizard engloba la conexión y el análisis a diferentes controladores de soldadura. Hasta este momento se ha cubierto la soldadura por puntos, pero, en la fase 3, se pretende abarcar nuevos dispositivos. A continuación se enumeran los dispositivos de interés y además de da una breve explicación de en qué consiste cada proceso de soldadura.

### Soldadura por haz láser

El proceso de soldadura láser utiliza un metal de aportación para la unión sin fundir realmente el material base. Además, permite lograr una alta precisión y una mínima deformación de la pieza de trabajo con una excelente relación calidad-precio.

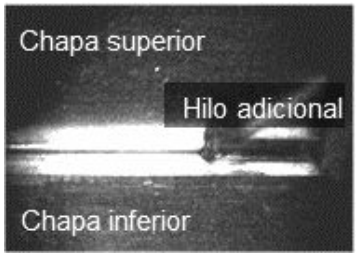



Figura 3.1: Soldadura por haz láser.

### Soldadura de pernos

Se pretende unir diferentes pernos a una chapa aplicando presión tras haber calentado lo suficiente ambos elementos con un arco eléctrico.

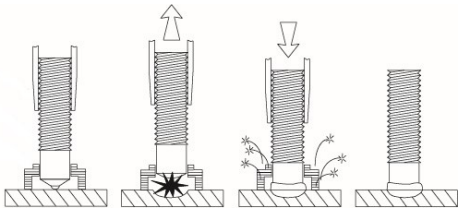


Figura 3.2: Soldadura de pernos.

### Soldadura con gas

La soldadura con gas inerte metálico (MIG) o con gas activo metálico (MAG) es un proceso de soldadura en el que se forma un arco eléctrico entre un electrodo de alambre MIG consumible y el metal o los metales de la pieza de trabajo, que calientan el metal o los metales de la pieza de trabajo, haciendo que se fundan y se unan. Junto con el

electrodo de alambre, un gas protector se alimenta a través de la pistola de soldadura, que protege el proceso de los contaminantes en el aire.

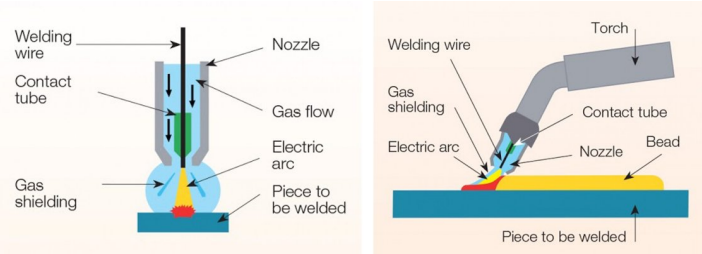



Figura 3.3: Soldadura con gas.

### Unión por adhesivo

Se pueden diferenciar dos tipos: sellado y adhesivo. El sellado es un método que permite el cierre de periféricos mediante una sustancia mástica impidiendo que ocurran fugas. Mientras que el adhesivo, se encarga de unir dos o más materiales que pueden tener, o no, las mismas características a través de una sustancia adhesiva.

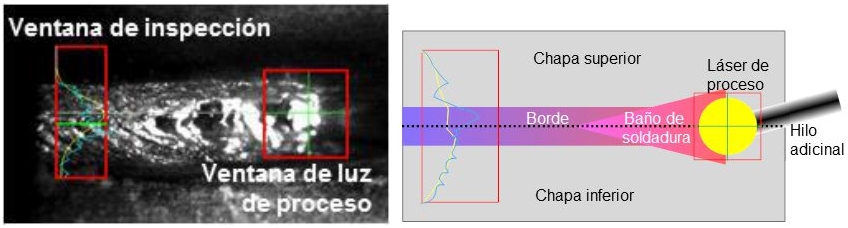
Figura 3.4: Unión por adhesivo.

## 3.3. Soldadura por haz láser

Se parte de que las instalaciones de los controladores de estos dispositivos ya incluyen un servidor llamada Weldeye donde se realizan las siguientes funciones:

- Comunicación con la interfaz del bus de campo.
- Comunicación/entrada de la imagen de la cámara.
- Análisis y registro de los datos de imagen.

Hay que tener en cuenta de que el sistema es capaz de detectar desajustes comparando ópticamente las curvas que se obtienen del proceso de soldadura actual con unas curvas de referencia, tal y como se puede apreciar en la figura 3.5a. Esto




(a) Fotografía real (b) Representación gráfica

Figura 3.5: Ventana de inspección y ventana del proceso actual

significa que además de recibir los parámetros de soldadura también se reciben tanto los errores como los avisos, en caso de que los haya.

Para poder extraer estos datos es necesario el software la empresa Lessmüller, fabricante de los dispositivos de soldadura por haz láser, que permite realizar este proceso y tanto la instalación como la configuración la realizado un ingeniero especialista. Una vez se ha instalado, estos datos se envían a una base de datos SQL que se encuentra localizada en un ordenador en planta. En la figura 3.6 se muestra un esquemático de la arquitectura que compone el sistema descrito anteriormente.

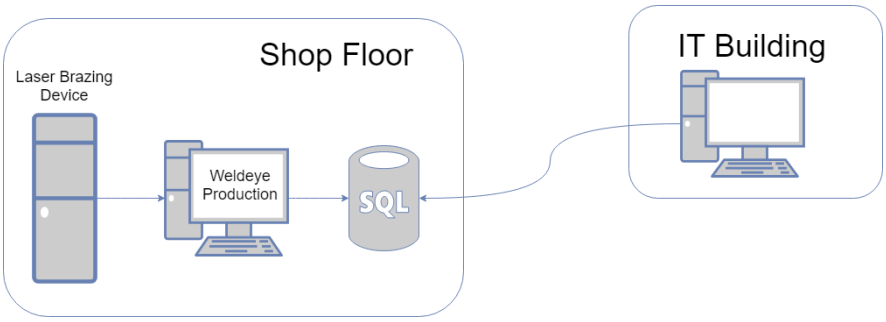



Figura 3.6: Arquitectura del sistema de soldadura láser. En planta se encuentra tanto el dispositivo como el controlador de soldadura, el Process-PC donde se ejecuta el software Weldeye y el ordenador donde se encuentra la base de datos. El acceso a la base de datos se realizará desde un ordenador que se encuentra fuera de planta.

Es importante tener en cuenta de que existe una limitación temporal a la hora de transmitir los datos desde el Process-PC a la base de datos, ya que solo se puede realizar en los momentos en los que no se produce la soldadura, es decir, desde que una carrocería sale tras ser soldada hasta que entra la siguiente. El tiempo del que se dispone es aproximadamente de 11 segundo. Además, esta limitación se traduce en que será necesario elegir de forma correcta los valores que resulten de más interés, ya que en el primer intento de conexión se detectó que se transmitía un fichero que contiene la configuración que hay en el controlador en cada momento. Se propone para futuras capturas intentar quitar el envío de este fichero y utilizar su tiempo de transmisión para otros parámetros que pueden ser de más interés.

## 3.4. Soldadura de pernos: Tucker

Existen dos familias diferentes de controladores de la marca Tucker, se tratan de los controladores DCE y TE de la empresa Stanley[11]. El método de establecer la conexión con cada uno de los controladores se abordará de forma diferente.

### 3.4.1. Controlador TE

Se trata del modelo más moderno, TE1500, en cuanto a soldadura de pernos de los existentes en planta y, gracias a eso, la forma de conectarse al controlador se realiza mediante una interfaz MQTT. Es decir, tras realizar la debida configuración, se conectará automáticamente al *broker* MQTT y enviará los datos a un determinado *topic*.

La arquitectura que se propone para esta prueba de concepto es la que se muestra en la figura 3.7. El dispositivo, que se encuentra en una planta de Ellesmere Port, se configura para que envíe los datos que recoge del controlador al *broker* y lo publique en un *topic* que asigna siguiendo el manual [12]. La instalación del *broker* se llevará a cabo en un jumbox situado en Ellesmere Port, donde se encontrará el cliente, quien se suscribirá a los *topic* y, haciendo uso de Node-RED, almacenará los datos capturados.

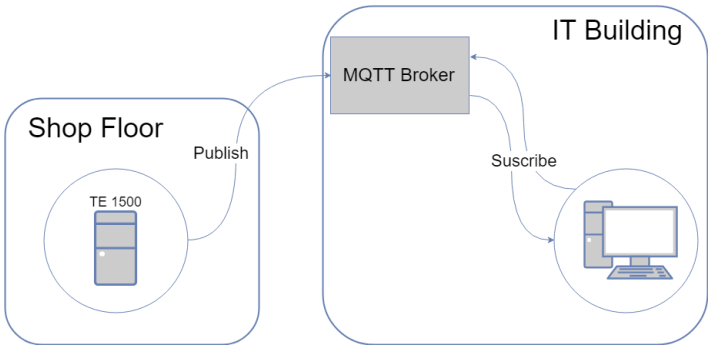



Figura 3.7: Arquitectura del sistema de soldadura TE 1500. En planta se encuentra tanto el dispositivo como el controlador de soldadura. Fuera de planta, en un ordenador se encuentra el broker y el suscriptor quien se encarga de almacenar los datos recogidos.

Las dos aplicaciones disponibles que pueden ser utilizadas como *broker* son RabbitMQ y Mosquitto. Aunque en ambas existe la posibilidad de que el publicador TE no necesite autenticación, se optó por utilizar finalmente Mosquitto debido a que ya se había comprobado con anterioridad su correcto funcionamiento.

Para el momento de finalizar el trabajo de final de grado no ha sido posible conseguir recibir los datos que envía el controlador de soldadura debido a que no se sabe con certeza si el dispositivo no es capaz de publicar en el *broker* por algún impedimento ocasionado por el *firewall*.

### 3.4.2. DCE Controller

Cuando se realizó un inventario para conocer la variedad de soldadores de pernos existentes en las plantas, el resultado que se obtuvo es que para este tipo de controlador existen dos modelos: DCE 1500 y DCE 1800. Al tratarse de los modelos más antiguos, no cabe la posibilidad de acceder a los datos mediante algún protocolo de software libre por lo que es la empresa quien tiene que facilitar el software para poder establecer la conexión.

Hay que tener en cuenta que, pese a que software es compatible con ambos modelos, no fue posible establecer conexión con los dispositivos DCE 1500 ya que la versión de firmware tiene una capa que impedía acabar de completar el protocolo de control de transmisión (Three-Way Handshake). Por lo que finalmente solo fue posible establecer la conexión con los dispositivos DCE 1500.

El software que ofrece la empresa se conoce como DCELink[13]. Da la posibilidad de establecer una conexión por la red entre los dispositivos de soldadura y una aplicación cliente-servidor donde el intercambio de los datos se realiza mediante el protocolo TCP/IP. Se compone por los siguientes módulos:

- DCELink-Server: gestiona la comunicación entre los controladores y la base de datos.
- DCELink-Manager: aplicación que se ejecuta en segundo plano y se encarga de controlar y monitorizar el estado del DCELink-Server.
- DCELink-Client: interfaz gráfica del programa DCELink-Server

Pese a que se recomendaba la instalación por separado de cada uno de los módulos, para la prueba de concepto se optó por instalar todos en el mismo ordenador. Para poder establecer la conexión con cada uno de los controladores, es necesario que tengan una IP única que permita identificarlos una vez se hayan añadido en la aplicación DCELink-Client. De una forma más detallada se muestran en el manual de instalación del DCELink[13] y en el Anexo D, los pasos necesario para realizar tanto la instalación como la configuración. A continuación, se muestra una figura donde se puede apreciar la arquitectura utilizada para la prueba de concepto necesaria para conseguir recoger los datos generados.

Durante la configuración fue necesario hacer frente a dos problemas principalmente los cuales dejaron las siguientes conclusiones.

- En los equipos DCE 1800 no es posible realizar la conexión ya que el firmware instalado debe de tener una capa que impide que el servidor, en este caso el controlador, responda al envío del paquete de sincronización por parte del cliente.

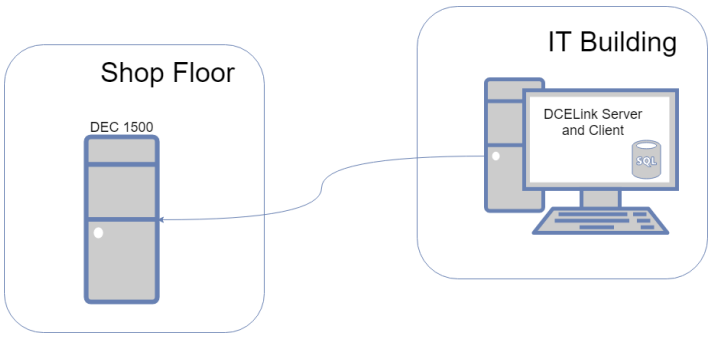



Figura 3.8: Arquitectura del sistema de soldadura DCE 1500. En planta se encuentra tanto el dispositivo como el controlador de soldadura. Fuera de planta se instala la base de datos, el servidor y el cliente.

- Es necesario una segunda tarjeta de red y otro cable de fibra óptica, en caso de que no esté ya instalada, porque el segundo cable que se conecte será el encargado del envío de datos.

## 3.5. Unión por adhesivo: SCA

Los modelos de los dispositivos con los que se va a trabajar son los SYS6000, que se presenta con y sin el sistema de visión llamado Quiss. Independientemente de la presencia del sistema Quiss, existen dos formas diferentes de poder acceder a los datos que generan los controladores:

- Mediante un software que proporciona la empresa Atlas Copco (SCA) llamado PDM.

- Realizando copias de seguridad.

Debido a que no se sabía con certeza el momento en el que sería posible realizar la instalación del software, se optó por recoger los datos mediante copias de seguridad y, cuando fuese posible, se procedería a establecer la conexión con el controlador de soldadura mediante el software.

### 3.5.1. Copias de seguridad

En un primer planteamiento, se intentó averiguar cómo se viable era realizar una conexión mediante el escritorio remoto al ordenador que tienen integrados los dispositivos que disponen de un sistema de visión. Se plantearon tres formas de realizarlas. La primera fue intentar acceder de forma remota al ordenador que tiene incorporado el dispositivo, pero se descartó debido a que el personal que se encuentra en planta temía que existiese la posibilidad de que se produjese algún conflicto entre

nuestra conexión y la que se encuentra controlando a la máquina. La segunda opción fue comprobar si el software ofrecía la posibilidad de automatizar las copias se seguridad. Ambas quedaron descartadas debido a que no eran viables por lo que se optó por la tercera, que consiste en realizar copias de seguridad de forma manual.

Se recogieron datos de 16 equipos diferentes y se vio que existía una limitación en cuanto a cantidad de datos ya que en la tabla solo era posible almacenar un máximo de 5000 filas. Además, se consultó con los técnicos de SCA sobre los datos recogidos y confirmaron que se podían utilizar como muestra para el proyecto ya que los parámetros que se obtiene tanto por copia de seguridad como mediante el software son prácticamente los mismos.

### 3.5.2. Software PDM

La forma escalable de obtener los datos es utilizar el software desarrollado por la empresa Atlas Copco (actual propietaria de SCA Schucker), que permite conectarse a controladores SCA y extraer datos de proceso.

Para su instalación fue necesario que un técnico de la empresa realizara la instalación del PDM, ya que se trata de un software que han desarrollado recientemente. Esto último implicó que la instalación no fue concluida con éxito a día de terminar el proyecto de final de carrera, ya que el técnico encargado de realizar la instalación se enfrentó ante problemas por lo que se pospuso para más adelante.

Pese al problema descrito anteriormente, se obtuvieron características relevantes para plantear la posible arquitectura que se implementaría, la cual se presenta en la figura 3.9. Aunque los requisitos se encuentran descritos en el manual [14] se comprobó que, pese a las recomendaciones dicen lo contrario, es posible instalar la aplicación y la base de datos SQL en máquinas separadas.

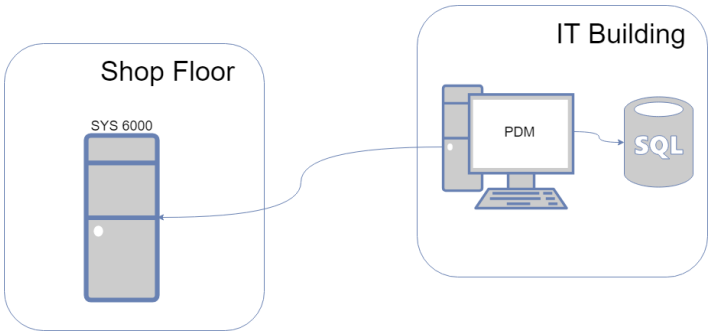



Figura 3.9: Arquitectura del sistema de soldadura SYS 6000. En planta se encuentra tanto el dispositivo como el controlador de soldadura. Fuera de planta se instala la base de datos y el software PDM.

# Capítulo 4

# Tratamiento de los datos

El punto de partir de esta fase es tras haber detectado las tablas que contienen los parámetros más relevantes habiendo realizado un análisis previo, como se puede ver de forma resumida en el capítulo anterior y, de una forma más extensa, en el anexo D.

La principal intención de este capítulo es detectar los parámetros más, o los menos, influyentes que conllevan a producir fallos y, como consecuencia, podrían parar durante un instante el proceso de soldadura. Para este estudio, se aplicarán diferentes algoritmos de predicción sobre cada conjunto de datos.

No fue posible realizar un análisis completo en los casos de soldadura por haz láser ni en soldadura de pernos TE. Esto se debe a que en el primer la cantidad de valores nulos es muy grande pero se adjunta en el anexo A el estudio que se realizó; en el segundo caso se debe a que para la fecha de realización del trabajo de final de grado no se había conseguido completar aún la primera fase lo que supone la imposibilidad de obtener tablas con los datos que genera el controlador para poder realizar su análisis a posteriori.

La tabla 4.1 se muestra un resumen de los objetivos planteados y si fue posible o no cumplirlos. Además, se muestra un resumen de las características del dataset con el que se trabaja en este capítulo.

| **Tecnología** | **Establecimiento de conexión** | **Adquisición de datos** | **Número de columnas** | **Longitud del dataset** | **Porcentaje de valores nulos** |
|---|---|---|---|---|---|
| Soldadura por haz láser | Si | Si | 8 | 8088288 | 0,0389 % |
| Soldadura con pernos (DCE1500) | Si | Si | 172 | 4587 | 67,53 % |
| Soldadura con pernos (TE1500) | No | No | - | - | - |
| Unión por adhesivo | No | Si | 31 | 138667 | 44,47 % |

Tabla 4.1: Resumen del cumplimiento de los objetivos. Contiene, además, información relevante para este capítulo.

A los valores reales de los datos se les ha añadido ruido para que no se pueda inferir la situación real de los procesos de soldadura. A pesar de esto, se trabajará con los datos como se fuesen los datos reales.

## 4.1. Soldadura de pernos

### 4.1.1. DCE Controller

Se comienza fusionando la tabla de parámetros con la de fallos. La cantidad de valores nulos que se generan es nuestro punto de partida en este estudio.

Realizando un estudio visual de la figura B.1c, se puede ver la existencia de grandes espacios en blanco. Estos espacios se deben a dos posibles causas: la existencia de un fallo durante un largo periodo o la posibilidad de que se haya desconectado el dispositivo debido a que los datos se recogen de una línea que se ha implantado recientemente.

El objetivo del experimento es ver si es posible detectar los parámetros más relevantes de cara a la predicción de la existencia de un error. Tras realizar el experimento se vio que en función del algoritmo que se ejecute, el resultado de las características más importantes variaba, por lo que se decidió que definir las menos relevantes sería un buen punto de partida.

Para realizarlo de una forma más realista, se optó por dividir el dataset en función del número de filas resultantes tras realizar el llenado de los valores nulos. Como ya se comentó anteriormente, en el dataset resultante de unir los fallos con los parámetros se vio que las filas no se complementaban pero que, tras producirse un fallo, los valores de las columnas de la tabla de parámetros eran todos nulos; y de forma análoga ocurría cuando en la tabla de parámetros se visualizan valores en la tabla de fallos solo se obtienen valores nulos. El método de llenado que se utilizó fue copiar los últimos valores de la tabla de parámetros en el momento que se produce un fallo y borrar las repeticiones de fallos que está almacenadas. Finalmente se eliminaron todos los valores que no tenían varianza ya que se correspondían con los máximos y mínimos de ciertas características del proceso de soldadura.

La cantidad de datos disponibles tras realizar esto es de 52 columnas y 2889 filas. A continuación, haciendo uso de las funciones que ofrece *Python*, se realizan matrices de correlación entre las columnas con el objetivo de ver si tras combinarlas, su relación es capaz de mostrar una clara división en el momento en el que se produce un error frente a cuando no ocurre. Tras realizar este experimento, se obtuvo un total de 17 columnas candidatas a ser estudiadas y el número de filas se mantuvo igual.

Con el objetivo de simular un entorno realista, se dividió el dataset en:

- 60 % dedicado a entrenamiento (1637 filas asociadas a una soldadura correcta frente a 96 filas que indican error).

- 20 % dedicado a validación (549 filas asociadas a una soldadura correcta frente a 29 filas que indican error).

– 20 % dedicado a la prueba final (551 filas asociadas a una soldadura correcta frente a 21 filas que indican error).

Los algoritmos de aprendizaje supervisado utilizados fueron: arboles de decisión, bosque aleatorio y XGBoost. En todos los casos se utilizan para clasificar, es decir, la predicción que realizan es la existencia o no de un error. A continuación, se adjunta una tabla que resume los resultados obtenidos tras su ejecución.

| | Corrección desbalanceo de clases | Árbol de decisiones | Bosque aleatorio | XGBoost |
|---|---|---|---|---|
| Porcentaje de acierto (%) | No | 95.32 | 94.46 | 94.98 |
| | Si | 72.31 | 75.43 | 76.81 |

Tabla 4.2: Porcentaje de acierto en función de diferentes modelos de predicción y haciendo uso de la corrección por desbalanceo de clases.

Con los resultados obtenidos después de haber ejecutado los tres algoritmos de aprendizaje automático se puede dar las siguientes conclusiones que se presentan en los siguientes párrafos y hacen referencia al anexo B.

En primer lugar, se ve que es necesario que el tiempo de captura sea más largo ya que un 5 % de datos de fallos no es suficiente para poder predecir de una forma correcta. Esto se refleja en la tabla 4.2, donde el porcentaje es muy elevado pero eso no implica que se haya realizado una buena predicción ya que con el dataset analizado no resulta muy complicado. Se puede ver que, al balancear las clases, el porcentaje de aciertos se reduce, pero esto a su vez significa que es más fiel a una realidad en la que se dispusiese de más muestras.

A través de las gráficas B.2, donde se muestran varios diagramas de barras con los porcentajes que asigna cada método a las columnas que componen el dataset, se pueden descartar parámetros como *isWip*, *ActualPilotCurrent* o *PilotVoltajePlusToleracce*.

## 4.2. Unión por adhesivo

Al igual que en el caso anterior se procede estudiar los valores nulos existentes en el dataset. En este caso, se puede ver en la figura C.1 que los datos que son valores nulos se producen mayoritariamente cuando existe un error aunque en alguna ocasión si se ha generado un fila completa que contiene tanto valores de los parámetros propios de la unión por adhesivo como del error que se ha generado.

Se realiza un recuento y se contabiliza que el 44,49 % de las filas se corresponden con valores nulos sobre un total de 138667 filas existentes.

Como se puede ver en al anexo D la mayor parte de los errores se producen en el sistema de visión los cuales no producen un paro en el proceso de soldadura y se

podrían tratar como falsos positivos. Asumiendo esta última afirmación, el método que se va a utilizar para completar los datos es copiar al valor previo. Teniendo en cuenta de que parte de los errores, no son realmente errores, sino que más bien son avisos, se utilizan modelos de regresión los cuales permiten marcar un umbral a partir del cual se detecta como fallo.

Para simular un entorno real en el que los datos nuevos se van acumulando y el modelo de clasificación y regresión se entrena con valores pasado, se va a dividir el dataset para utilizar una parte como entrenamiento, otra como validación y finalmente, tras entrenar el modelo, una parte para testear el modelo. Es importante recalcar que este último trozo no se ha utilizado para ajustar los parámetros del modelo. A continuación, se presenta una tabla en la que se muestra la cantidad de valores existentes en casa división del dataset.

| | Entrenamiento | Validación | Prueba final | Total | Porcentaje sobre el total ( %) |
|---|---|---|---|---|---|
| Valor 0 | 48099 | 13964 | 14906 | 76968 | 78.55 |
| Valor 1 | 10689 | 5633 | 4690 | 21012 | 21.44 |

Tabla 4.3: Subdivisión del dataset

Con estos datos disponibles se procede a entrenar a los modelos de clasificación en primer lugar. En la tabla 4.4 se pueden ver los resultados obtenidos tras ajustar los parámetros utilizando los datos de entrenamiento y validación y, posteriormente, prediciendo la existencia de fallos en el conjunto de datos de testeo final.

| | Corrección desbalanceo de clases | Árbol de decisiones | Bosque aleatorio | XGBoost |
|---|---|---|---|---|
| Porcentaje de acierto | No | 76.02 | 76.045 | 76.02 |
| ( %) | Si | 70.61 | 77.50 | 77.03 |

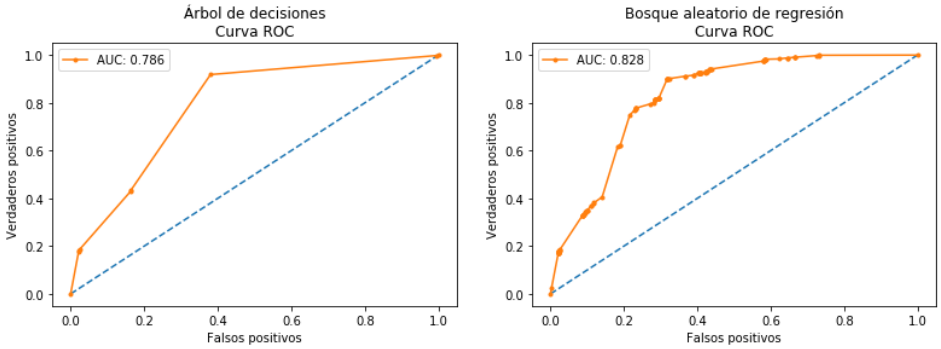
Tabla 4.4: Porcentaje de acierto en función de diferentes modelos de predicción y haciendo uso de la corrección por desbalanceo de clases.

Se puede ver que pese a que los porcentajes de acierto no varían mucho, cada modelo hace un uso diferente de los parámetros a la hora de realizas la predicción, como se puede ver en la figura C.2. Aunque no se puede dar un nombre exacto de los parámetros más influyentes, si se puede afirmar que a la variable *Bomba_activa* no se le da mucho uso; le sigue el parámetro *Desviacion_volumen*. Prácticamente coinciden con los valores que tiene menos interés para los predictores en los modelos de regresión que son *Bomba_activa*, *Par_giro_medio* y *Desviacion_volumen*.

Al haber utilizado modelos de regresión es posible generar la gráfica de la curva

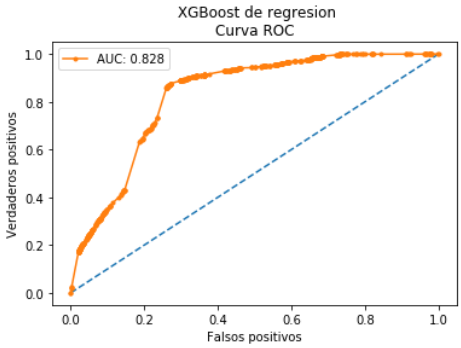
ROC. Esta curva permite elegir el punto a partir del cual se considera la existencia de un fallo. Es un valor queda a elección.

Tras ver las figuras de abajo, un modelo de elección podría ser perder uno de cada cinco fallos a cambio de tener prácticamente un 80 % de aciertos. Esta pérdida se podría asumir ya que, como se ha comentado previamente, gran parte de los fallos en realizad son avisos.




(a) Curva ROC del algoritmo árbol de decisión.

(b) Curva ROC del algoritmo bosque aleatorio.




(c) Curva ROC del algoritmo XGBoost.

Figura 4.1: Curvas ROC de cada uno de los algoritmos probados

# Capítulo 5

# Conclusiones

Teniendo en cuenta que llevar a cabo el proyecto resultó ser más complicado de lo esperado, en cuando a tiempos se refiere, se consiguió dar por finalizada la fase de establecimiento de conexión de dos de los cuatro tipos de dispositivos que entraban dentro del plazo de estudio del proyecto, se trata de los controladores de soldadura por haz láser y los soldadores de pernos modelo DCE 1500. Sobre los otros dos tipos de controladoras, soldadura de pernos modelo TE 1500 y unión por adhesivo, será necesario seguir investigando ya que no se ha conseguido completar la fase de establecimiento. En el caso de soldadura de pernos modelo TE1500 no se ha conseguido recibir datos del controlador y será necesario buscar ayuda por parte de los técnicos de Tucker para encontrar el motivo por el cual, pese a que tanto el controlador como el broker y el suscriptor están configurados, no se consigue o bien que el controlador publique los datos en el tópico acordado o que el cliente, es decir nosotros, nos suscribimos de forma correcta al topic. En principio se descarta que el broker elegido no sea compatible con los requerimientos del controlador ya que hace un año se realizó esta misma prueba en otra planta y se consiguió conectar sin problemas. Existe un quinto dispositivo que se pretendía abordar paralelamente a los otros cuatro, pero resultó más complicado adquirir una versión de prueba por lo que quedó fuera del objetivo.

En la segunda fase del proyecto se intenta explorar las tres tablas recogidas partiendo de que contiene parámetros que pueden resultar desconocidos. Se consiguió conocer tanto los parámetros como las tablas donde se almacenan y se realizó un resumen de ellas. Tras estar un poco más familiarizado con los datos de cada controlador, el siguiente paso fue sumergirnos un poco más en los dataset con el objetivo de detectar los parámetros más relevantes desde el punto de vista de predecir errores. Aplicando estos algoritmos se consiguió detectar los parámetros menos importantes y algunos que influyen siempre en la decisión.

# Capítulo 6

# Bibliografía


[1] https://www.idia.es/.

[2] G. Morabito L. Atzori D. Giusto, A. Iera. *The Internet of Things*. Springer, 2010.

[3] https://nodejs.org/es/.

[4] https://d3js.org/.

[5] Definición: Base de datos relacional. https://searchdatacenter.techtarget.com/es/definicion/Base-de-datos-relacional. [Internet; visitado 14/05/2019].

[6] https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-2017. [Internet; visitado 13/05/2019].

[7] Surabhi Chouhan, Divakar Singh, and Anju Singh. An improved feature selection and classification using decision tree for crop datasets. *International Journal of Computer Applications*, 142(13), 2016.

[8] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[9] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[10] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[11] http://www.emhart.eu.

[12] Lessmüller Lasertechnik. *WELDEYE PRODUCTION - Manual del sistema*, Abril 2013. Versión 2.8.

[13] Stanley Engineered Fastening. *Operating instructions - DCELink*, Marzo 2017. Version 01.82.

[14] Atlas Copco (SCA). *PDM SpecSheet*, Marzo 2017. Version 2.0.

[15] `https://www.rabbitmq.com/`.

[16] `https://flows.nodered.org/node/node-red-contrib-python3-function`.

[17] Luis H. Menéndez. Soldadura 'inteligente' en las plantas de opel y psa desarrollada en zaragoza. *Heraldo de Aragón*, 2018.

[18] `http://www.lessmueller.de/`.

[19] `http://www.emhart.eu/eu-en/products-services/products-by-category/tucker-stud-welding/stud-welding-systems/energy-units.php`.

[20] Proceso de soldadura - gmaw. `https://www.esab.com.ar/ar/sp/education/blog/proceso_soldadura_gmaw.cfm`. [Internet; visitado 26-abril-2019].

# Lista de Figuras

# Lista de Tablas

# Glosario

**AMQP** Advanced Message Queuing Protocol. 6

**Broker** Elemento central encargado de gestionar la red y transmitir los mensajes entre los clientes. 6–8, 17

**Firewall** Sistema que permite o bloquea el tráfico saliente o entrarte en función de un conjunto de restricciones de seguridad. 17

**IDia** Investigación, Desarrollo e Innovación en Aragón. 1

**IoT** Internet of Things. 1, 5

**Jumbox** Ordenador que se encuentra en red y se utiliza para acceder y controlar dispositivos. 17

**MQTT** Message Queue Telemetry Transport. 6, 7, 17

**NPM** Node Package Manager. 8

**QoS** Quality of Service. 7

**SQL** Structured Query Language. 9

**Topic** Caracteristica del protocolo MQTT que permite que el suscriptor diferencie al cliente que envía el mensaje. 17