



Gestión integrada de los procesos y máquinas para la mejora del mantenimiento y flexibilización de la producción

**Nº Expediente:** IMDECA/2015/82

**PROGRAMA:** PROYECTOS DE I+D EN COLABORACIÓN

**ACTUACIÓN:** IMDECA-Proyectos de I+D en colaboración

**Fecha de concesión:** 23 de octubre de 2015

## Entregable 3.1 (E3.1)

# Informe sobre la sensorización de la fábrica del futuro

Pertenece al paquete de trabajo: PT3

Participante responsable: ITI

Mes estimado de entrega: 24

---

## Resumen

OPTIMAN es un proyecto financiado con el Instituto Valenciano de Competitividad Empresarial (IVACE) y la Unión Europea a través del Fondo Europeo de Desarrollo Regional (FEDER).

El presente documento tiene como objetivo recoger las especificaciones de diseño y resultados de la construcción del Sistema de Sensorización de la Fábrica del Futuro. Para ello, se hará un recorrido desde el diseño de la arquitectura de sensorización, hasta la construcción y validación del sistema.

## Abstract

OPTIMAN is a project funded by the Valencian Institute for Business Competitiveness (IVACE) and the European Union through the European Regional Development Fund (FEDER).

This document aims to collect the design specifications and results of the construction of the Sensing System of the Factory of the Future. It contains the design of the sensor architecture, the construction and validation of the system.

## Tabla de Contenidos

<b>1. Glosario de términos</b>	<b>4</b>
<b>2. Introducción</b>	<b>5</b>
2.1. <i>Objetivos del paquete de trabajo E3</i>	5
2.2. <i>Objetivo del presente documento (E3.1)</i>	5
<b>3. Infraestructura de sensorización</b>	<b>6</b>
3.1. <i>AIMPLAS</i>	6
3.2. <i>INESCOP</i>	18
<b>4. Arquitectura de software de sensorización</b>	<b>30</b>
4.1. <i>Flujo de datos</i>	31
4.2. <i>Componentes del Sistema de Sensorización</i>	32
4.2.1. <i>Data Polling</i>	32
4.2.2. <i>Data Parsers</i>	34
4.2.3. <i>Data Converters</i>	35
4.2.4. <i>Monitoring Tools</i>	36
<b>5. Desarrollo del sistema de sensorización</b>	<b>37</b>
5.1. <i>Estructura de datos del sistema de sensorización</i>	37
5.2. <i>Interfaz REST</i>	41
5.3. <i>Inicio, configuración y arranque de la arquitectura</i>	44
5.3.1. <i>Notas adicionales sobre la instalación</i>	45
<b>6. Herramienta de monitorización</b>	<b>46</b>
6.1. <i>Monitorización del software de sensorización</i>	47
6.2. <i>Monitorización del cluster Cassandra</i>	47
6.3. <i>Monitorización del host</i>	47
6.4. <i>Monitorización de logs del sistema</i>	48
<b>7. Validación del sistema de sensorización</b>	<b>48</b>
7.1. <i>Resultados del plan de pruebas</i>	49
7.1.1. <i>Pruebas funcionales unitarias</i>	49
7.1.2. <i>Pruebas de integración y rendimiento</i>	50

## Glosario de términos

En el presente documento se utilizará algunos conceptos que deben ser definidos con anterioridad a su lectura para aclarar ambigüedades y facilitar el entendimiento.

- **Fabrica del Futuro (FoF):** Fábrica inteligente capaz de adaptar el proceso productivo a las necesidades de producción haciendo uso de las nuevas tecnologías a fin de aumentar la eficiencia, la calidad y reducir el impacto medioambiental.
- **Big data:** Conjunto de tecnologías que permiten el manejo de grandes volúmenes de datos que no pueden ser tratados con las tecnologías convencionales.
- **Machine Learning:** Conjunto de técnicas que permiten crear algoritmos capaces de generalizar comportamientos a partir de información no estructurada.
- **Sensorización:** Método de recepción de datos mediante uno o varios sensores que se instalan en la máquina que se quiere medir. La sensorización puede ser de muy diversa naturaleza basada en métodos de detección físicos, ópticos, electromagnéticos...
- **CRC:** Es un código de detección de errores usado frecuentemente en redes digitales y en dispositivos de almacenamiento para detectar cambios accidentales en los datos.
- **BANDA PASANTE:** Gama o conjunto de frecuencias que un determinado sistema, amplificador, filtro, sensor es capaz de dejar pasar sin atenuación.
- **FFT:** Transformada rápida de Fourier: FFT es la abreviatura usual (del inglés Fast Fourier Transform) de un eficiente algoritmo que permite calcular la transformada de Fourier discreta (DFT) y su inversa. Nos permite un tratamiento matemático, para pasar una señal del dominio temporal al dominio de la frecuencia y obtener las diferentes componentes frecuenciales que configuran su espectro. La FFT es de gran importancia en una amplia variedad de aplicaciones, desde el tratamiento digital de señales y filtrado digital en general a la resolución de ecuaciones en derivadas parciales o los algoritmos de multiplicación rápida de grandes enteros.
- **ACELERÓMETRO:** Dispositivo que nos permite captar la variación de velocidad o vibración de un determinado objeto en movimiento.
- **SENSOR DE INTENSIDAD AISLADO:** Dispositivo que nos permite medir la intensidad de un circuito eléctrico con aislamiento galvánico.
- **SENSOR PIEZOELECTRICO:** Dispositivo basado en material piezoeléctrico que puede transformar un movimiento mecánico en una señal eléctrica

## 2. Introducción

### 2.1. Objetivos del paquete de trabajo E3

El objetivo de este paquete pretende dotar de las infraestructuras de sensorización y gestión de la información adecuada a los sistemas de producción de la FoF. Para ello, será necesario el diseño y desarrollo de una arquitectura software de sensorización, adaptable a los diferentes sistemas de producción.

Por otro lado, se desarrollará las diferentes interfaces de comunicación entre los elementos definidos en la arquitectura de sensorización, permitiendo así su orquestación para la captación de datos. El resultado de estas mediciones se sintetizará visualmente para mostrar la traza y el comportamiento de los elementos hasta un nivel de detalle que permita la auditoría del funcionamiento de los sensores.

### 2.2. Objetivo del presente documento (E3.1)

El principal objetivo del entregable E3.1 es el de recoger las **especificaciones de diseño y resultados de la construcción del Sistema de Sensorización** de la Fábrica del Futuro. Para ello, se hará un recorrido desde el diseño de la arquitectura de sensorización, hasta la construcción y validación del sistema.

El documento recoge los resultados de las tareas del paquete de trabajo:

- **T.3.1. Diseño de la arquitectura software de sensorización:** que soporta la obtención de las medidas de los sensores instalados en los mecanizados para su posterior envío, procesamiento y análisis. Para ello se determinarán los parámetros a medir, señales que se van a monitorizar en cada proceso seleccionado y la forma en la que se realizará la recogida de datos.
- **T.3.2. Desarrollo de la arquitectura de software de sensorización:** implementación del sistema de sensorización software y hardware en ambos casos de uso.
- **T.3.3. Desarrollo de herramienta de supervisión de la sensorización:** implementación de una herramienta web que permita detectar malfuncionamientos del sistema de sensorización.
- **T.3.4. Validación del sistema de sensorización:** pruebas de funcionamiento de la integración del sistema de sensorización en ambos casos de uso.

Se puso especial atención a los requerimientos relacionados con el tipo de variables a medir de los mecanizados y el tipo de sensores que permitirán medir su funcionamiento y que permitan realizar predicciones, detectar anomalías y optimizar los procesos de fabricación.

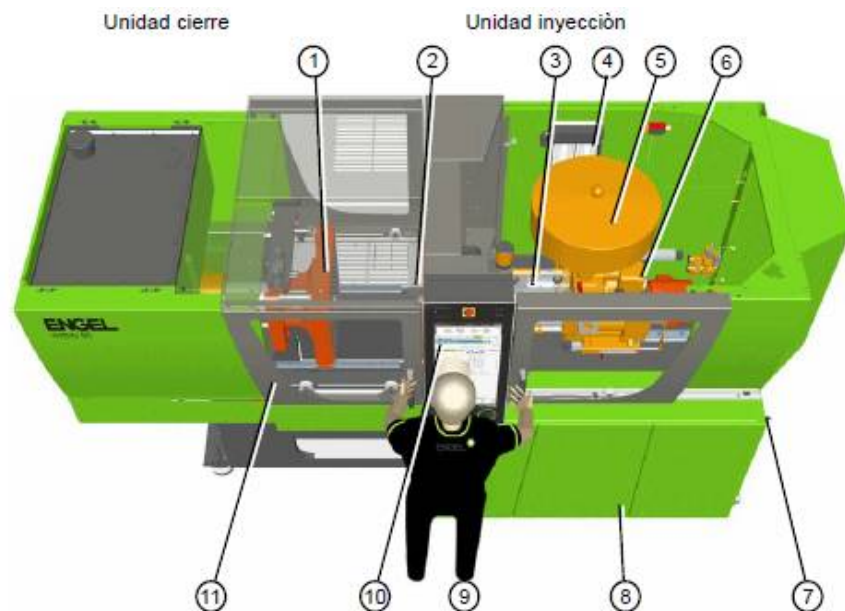
### 3. Infraestructura de sensorización

La infraestructura de sensorización utilizada en el proyecto tiene elementos comunes en los contextos del laboratorio de AIMPLAS e INESCOP, sin embargo, debido a que los mecanizados son diferentes, existen diferencias que se describen a continuación:

#### 3.1. AIMPLAS

A continuación se presenta cual ha sido la infraestructura de sensorización utilizada para la realización del experimental.

##### a) Especificaciones de la inyectora Engel Victory 50 y tipo de señales que captura:



- [1] Placa portamoldes móvil
- [2] Placa portamoldes fija
- [3] Cilindro plastificador con boquilla
- [4] Distribuidor de agua refrigeración
- [5] Contenedor material
- [6] Unidad de inyección
- [7] Interruptor principal
- [8] Armario mandos
- [9] Puesto de trabajo personal de servicio
- [10] Panel de manejo
- [11] Tapa protección móvil unidad cierre

- Imagen 1: Partes de la máquina inyectora

Como se puede observar en la imagen, la máquina inyectora está compuesta por dos unidades; la unidad de inyección y la unidad de cierre. La unidad de inyección está al cargo de dosificar el material plástico al cilindro para que a lo largo de éste se plastifique y pueda ser inyectado en el molde.

La unidad de cierre es donde se posiciona el molde y el sensor de presión en cavidad del molde (ver conexionado en Imagen 2), el cual mide la presión que existe dentro del molde.

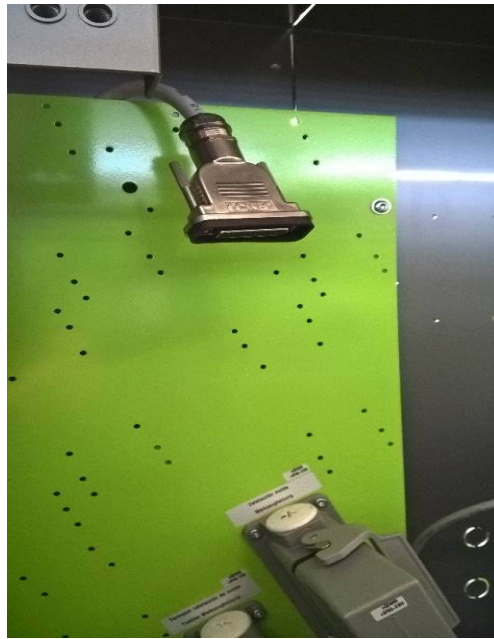


Imagen 2: Sensor de presión en cavidad

Todos los parámetros de entrada tanto de la unidad de inyección como del molde (temperatura, velocidad y presión) que es necesario que se especifiquen durante el proceso, son introducidos a la máquina mediante el panel de manejo (Punto 10 de la Imagen 1).

Entre la unidad de inyección y la unidad de cierre se encuentra el cable de red Ethernet RJ45 para la transmisión de datos de la máquina al servidor. Este cable va desde la tarjeta capturadora, donde recoge los parámetros de salida, a la caja de conexión de red.



Imagen 3: Cable de red Ethernet RJ45 de categoría 6



Imagen 4: Caja de conexión

En la parte frontal de la máquina inyectora se encuentra el armario de mandos (punto 8 de la Imagen 1). Dentro de este panel se localiza la tarjeta capturadora a la que va conectado el cable de red que se observa en la imagen superior y el ordenador de la máquina.

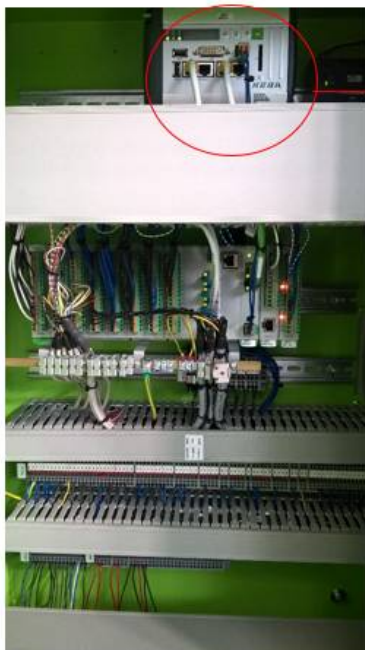


Imagen 5: Ordenador de la máquina inyectora



Imagen 6: Tarjeta capturadora

La energía que necesita la máquina para realizar cada ciclo de inyección es extraída del cuadro de energía.

En este cuadro se encuentra el disyuntor o interruptor general de máquina de donde salen los cables de potencia que alimentan la máquina inyectora, a los que están conectados unas pinzas de medida de voltaje, y las pinzas de medición de amperaje (ver Imágenes 7 a 9).



Imagen 7: Cuadro de energía



Imagen 8: Disyuntor o interruptor general



Imagen 9: Pinzas de medición de amperaje

Como se observa en la imagen 8, a los cables de potencia están conectadas unas pinzas de medida de voltaje.

La medida de voltaje que recogen es enviada al analizador de redes. En la imagen 9 se pueden observar las pinzas de medición de amperaje, éstas están conectadas a la vez al analizador de redes (ver Imagen 10).



Imagen10: Analizador de redes

Los parámetros de entrada del ciclo de inyección son introducidos mediante el panel de mandos que es el sistema de control de la máquina, mediante el cual se puede ordenar y dirigir el proceso de inyección. Este sistema de control está basado en un Linux Ubuntu.

Después de cada ciclo los parámetros de salida son recogidos por la tarjeta capturadora y transmitidos al ordenador de la inyectora para que los interprete y los envíe de nuevo a la tarjeta.

Como se ha comentado anteriormente en el apartado de especificaciones, a esta tarjeta está conectado el cable de red Ethernet RJ45 de categoría 6, mediante el cual se realiza la transmisión de datos a la caja de conexión para que sean enviados al servidor.

Los datos son exportados en archivos de texto plano, separando los datos mediante el símbolo ','. Los archivos de exportación de los datos del experimental son volcados en una carpeta compartida, mediante protocolo SMB en un servidor Microsoft Windows 2008, creado a tal efecto.

Los datos de consumo energético se pueden recoger gracias a las pinzas de medición de voltaje y amperios que están conectadas en el cuadro de energía.

Los datos recogidos son transferidos al analizador de redes, el cual se encarga de interpretarlos y registrarlos. Los resultados que proporciona el analizador son los KW/h de consumo de la máquina inyectora.

Estos resultados son enviados del analizador al ordenador para que, con el programa correspondiente se interpreten y se puedan volcar en la carpeta habilitada en el servidor de AIMPLAS.

Por seguridad e integridad de los datos exportados se ha creado un recurso en el cual se ha habilitado un usuario de acceso a la carpeta compartida y se han restringido los permisos de escritura en ella para que solo ese usuario pueda crear y modificar los archivos de datos exportados, para que de esta manera el Instituto Tecnológico de Informática pueda acceder a ellos y poder realizar la correspondiente interpretación.

Para la realización de todo este proceso de recopilación, registro y transferencia de datos ha sido necesaria la instalación del utillaje (todos los cables y conexiones)

Se realizaron diferentes pasos y tareas para poder exportar la información directamente a la nube:

- La máquina se conectó a la red local de AIMPLAS y se le asignó una dirección IP del rango de direcciones locales.
- Se trató de habilitar el acceso directamente a la máquina desde internet, para que el personal del ITI pudiera extraer y analizar los datos. Para ello se reservó una dirección IP pública de AIMPLAS y se crearon reglas en el sistema de firewall para habilitar la comunicación desde internet a esa dirección IP. Esta dirección IP se configuró en la zona desmilitarizada de la red de AIMPLAS (DMZ), donde están el resto de servicios accesibles desde internet ofrecidos por AIMPLAS (web, correo electrónico...).
- En el firewall se configuró una redirección de las conexiones entrantes a esa dirección pública asignada de la máquina a la dirección interna.
- Se configuró una conexión privada virtual (VPN) mediante Ipsec para que la comunicación con la máquina fuera encriptada. Ipsec es un conjunto de protocolos que actúan a nivel de capa de red utilizados para asegurar la comunicación entre dos direcciones IP.
- La conexión privada virtual mediante Ipsec daba muchos problemas por lo que se reconfiguró para utilizar túneles securizados mediante el protocolo SSL. SSL es un conjunto de protocolos criptográficos que utilizan certificados x.509 para autenticar los extremos de la comunicación y encriptar los datos que se están comunicando.
- Como seguía dando problemas en el establecimiento de la conexión se decidió que la máquina exportara la información directamente a una carpeta compartida mediante protocolo SMB en un servidor Windows.

Se configuró en la máquina la conexión a esa carpeta compartida y el tipo de exportación.



### c) Arquitectura de la máquina para poder aportar datos del proceso


Para poder controlar y dirigir el proceso de inyección y registrar los resultados del mismo, en el panel de control de la máquina aparecen dos apartados a tener en cuenta.

## Paquete datos proceso

**Página de pantalla**

■ Paquete datos proceso

La página de la pantalla se encuentra típicamente abajo de los siguientes componentes y tareas:

Componentes	Cometidos
 Oficina	<ul style="list-style-type: none"> <li> Controlar producción</li> <li> Optimizar producción</li> <li> Primeras muestras</li> <li> Ajustar vigilancias</li> <li> Ajustes producción</li> </ul>

El paquete de datos de proceso (PDP) posibilita un registro numérico y gráfico de todos los datos de proceso medidos por el mando. Sirven como ayuda para analizar, vigilar y documentar el proceso de inyección.

El paquete se comprende de programas:

- Protocolo de datos de proceso
- Gráfica datos de proceso
- Vigilancia datos de proceso
- Histograma
- Grafica correlación

Imagen13. Paquete de datos de proceso

Se puede encontrar un subpartado en el que se pueden elegir todos los parámetros que se quieren controlar



Imagen 14: Selección de parámetros proceso

Los resultados que se van obteniendo en cada ciclo se almacenan y se pueden consultar clasificados y ordenados (ver Imagen 15).

### Protocolo de datos de proceso

El programa sirve para el registro de datos de proceso en la máquina y memorizándolos en un fichero para el procesamiento posterior en PC.

Protocolo

Seleccionar protocolo

Nombre proto- Protocolo 090913

Tiempo Inicio	SCx (S-N)	SCx (S-N)	SPx (S-N)	SPx (S-N)	SPx (S-N)	SPx (S-N)	SPx (S-N)	Adu
10:33:57	1725	1715	103,4	199,9	10,1	23,0	8,0	
10:34:04	1726	1716	103,4	199,9	10,1	23,0	8,0	
10:34:10	1727	1717	103,4	200,0	10,3	23,0	7,8	
10:34:17	1728	1718	103,4	200,0	10,3	23,0	7,8	
10:34:24	1729	1719	103,4	199,9	10,0	23,0	8,8	
10:34:31	1730	1720	103,4	199,9	10,0	23,0	8,8	
10:34:37	1731	1721	103,4	199,9	10,1	23,0	7,7	
10:34:44	1732	1722	103,4	199,9	10,1	23,0	7,7	
10:34:51	1733	1723	103,4	199,9	10,1	23,0	8,0	
10:34:58	1734	1724	103,4	199,9	10,1	23,0	8,0	
10:35:04	1735	1725	103,0	200,0	10,2	23,0	8,0	
10:35:11	1736	1726	103,4	200,0	10,2	23,0	8,0	
10:35:18	1737	1727	103,4	200,0	10,0	23,0	8,2	
10:35:25	1738	1728	103,4	200,0	10,0	23,0	8,2	
10:35:32	1739	1729	103,0	200,0	10,0	23,0	6,6	
10:35:38	1740	1730	103,4	200,0	10,0	23,0	6,8	
10:35:45	1741	1731	103,4	200,0	10,3	23,0	7,9	
10:35:52	1742	1732	103,0	200,0	10,3	23,0	7,9	
10:35:59	1743	1733	103,4	199,9	10,2	23,0	8,0	
10:36:05	1744	1734	103,4	199,9	10,2	23,0	8,0	
10:36:12	1745	1735	103,4	199,9	10,0	23,0	8,7	
10:36:19	1746	1736	103,4	199,9	10,0	23,0	8,7	
10:36:26	1747	1737	103,4	200,0	9,9	23,0	9,1	
10:36:33	1748	1738	103,4	200,0	9,9	23,0	9,1	
10:36:39	1749	1739	103,4	200,0	10,2	23,0	8,2	
10:36:46	1750	1740	103,4	200,0	10,2	23,0	8,2	
10:36:53	1751	1741	103,4	200,0	10,0	23,0	8,9	
10:37:00	1752	1742	103,4	200,0	10,0	23,0	8,9	
10:37:06	1753	1743	103,4	199,9	10,2	23,0	8,3	
10:37:13	1754	1744	103,4	199,9	10,2	23,0	8,3	
10:37:20	1755	1745	103,4	199,9	10,0	23,0	7,9	
10:37:27	1756	1746	103,4	199,9	10,0	23,0	7,9	
10:37:33	1757	1747	103,0	199,9	10,1	23,0	8,1	
10:37:40	1758	1748	103,4	199,9	10,1	23,0	8,1	
10:37:47	1759	1749	103,4	199,9	10,1	23,0	8,7	
Valor Ideal		1700	103,4	199,9	10,1	23,0	8,0	
Mínimo		1650	103,3	199,9	9,9	23,0	8,0	
Máximo		1740	103,4	200,0	10,3	23,0	9,2	
Diferencia		90	0,0	0,1	0,4	0,0	1,1	
valor medio		1700	103,4	199,9	10,1	23,0	8,3	

Inicio    Protocolo    Export    Vista

Imagen 16. Parámetros de salida obtenidos

Estos datos se pueden registrar y exportar, como se muestra en la Imagen 16.



Imagen 16. Exportación de datos.

#### d) Paquete de datos de temperatura.

En esta sección del control de la máquina se puede realizar el seguimiento y monitorización de las temperaturas del proceso.

### Paquete de datos de temperatura

**Página de pantalla**

**Paquete de datos de temperatura**

La página de la pantalla se encuentra típicamente abajo de los siguientes componentes y tareas:

Componentes	Cometidos
 Oficina	<ul style="list-style-type: none"> <li> Controlar producción</li> <li> Optimizar producción</li> <li> Primeras muestras</li> <li> Ajustar vigilancias</li> <li> Ajustes producción</li> </ul>

El monitoreo de calidad para un máximo de 40 parámetros con el cálculo de los valores estadísticos. Representación gráfica seleccionable individualmente de los parámetros. La unidad de control guarda los valores de hasta 1000 ciclos.

Imagen 17. Paquete de datos de temperatura.

Dentro del paquete de datos de temperatura se puede encontrar un subpartado en el que se pueden seleccionar aquellas zonas de calefacción que se desean controlar.

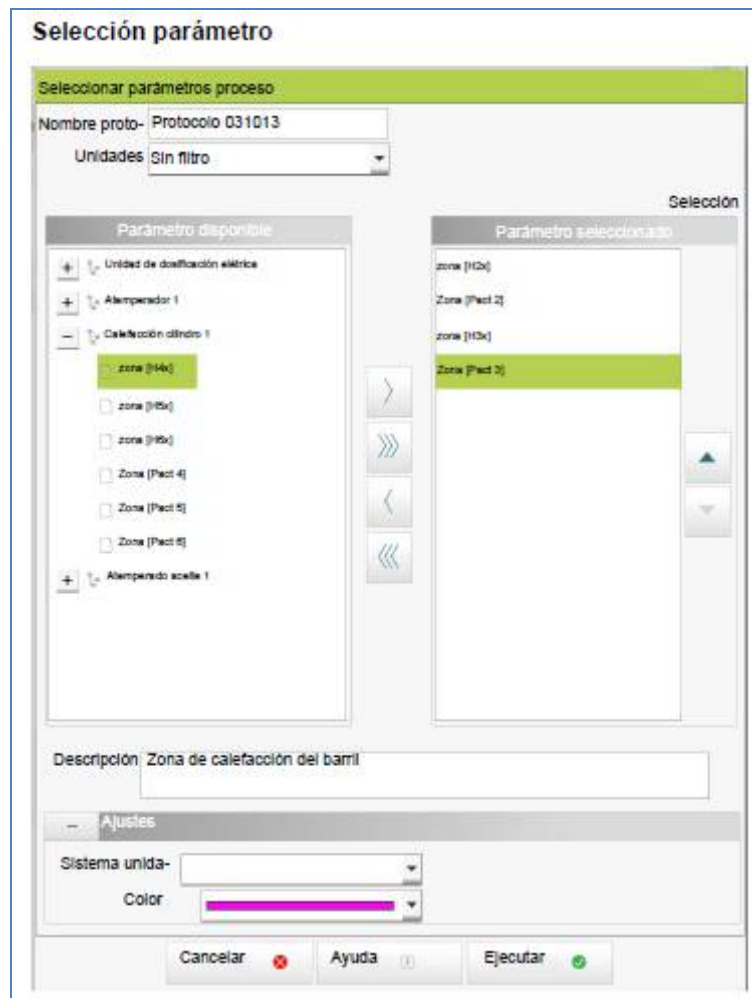


Imagen 18. Selección de las zonas de temperatura a controlar

El tratamiento conjunto de señales comentado anteriormente así como la infraestructura de comunicación, recogida de datos y securización de los mismos ha sido necesaria para el desarrollo del proyecto piloto en las instalaciones de AIMPLAS.

### 3.2. INESCOPI

El primer paso para realizar la sensorización del centro de mecanizado es definir la estrategia y los parámetros característicos de la sensorización. Conceptualmente, el sistema de sensorización se orienta del mismo modo que un operador experto, siendo capaz de "detectar" anomalías en función del nivel y características de las vibraciones que se generan durante el proceso de mecanizado. El operador, de una forma un tanto intuitiva va estableciendo una correlación entre resultados de mecanizado y el patrón de de vibración, de modo que es capaz de "detectar" anomalías de funcionamiento en función del tipo de vibración observada. Existen diferentes estudios y patentes sobre sistemas de mantenimiento predictivo en máquinas rotativas basados en el análisis de la vibración [vibra\_1],[ vibra\_2]

En nuestro caso, una de las principales dificultades consistentes es la diversidad de señales, frecuencias y periodos característicos que se producen durante el mecanizado de una determinada pieza, debido a que el patrón de mecanizado es variable y con trayectorias complejas por lo que únicamente mediante un procesado matemático avanzado será posible obtener conclusiones correctas.

Partiendo del asesoramiento de los expertos, previamente, definimos las variables que por sus características, nos pueden aportar información relevante del mecanizado a analizar, y se plantea un sistema capaz de capturar diferentes variables:

- Vibración en diferentes zonas,
  - Medida de vibración en el eje de mecanizado
  - Medida de vibración sobre soporte de pieza de mecanizado
- Velocidad de giro de herramienta mecanizado
- Velocidad de avance de mecanizado
- Consumo eléctrico de motor de mecanizado

Inicialmente definimos una banda pasante característica comprendida entre 20 y 5Khz, para la medida de vibración, y una banda pasante de 10 hz para la medida de velocidad de giro, avance y consumo eléctrico.

Para la medida de vibración, se estudian diferentes tipos de captura, primeramente se estudia la posibilidad de sistemas basados en interferometría láser, pero se descartan, por la dificultad de aplicación en entorno industrial con partes móviles, siendo la opción de sistemas piezoeléctricos, el que mejores características nos proporciona.

La sensorización llevada a cabo en el centro de mecanizado se ha realizado mediante distintos tipos de sensores. Estos sensores nos han proporcionado toda la información relevante sobre el centro de mecanizado aun siendo esta información de distinta naturaleza y sin una relación aparente entre ella. Para ello se ha partido de cero y se ha construido una infraestructura adaptada a nuestras necesidades y al propio centro de mecanizado donde se realizan las distintas pruebas.

La idea de concepto, consiste en la captura de vibraciones en tiempo real, con un ancho de banda dimensionado conforme a la información que se quiere monitorizar, en nuestro caso, la banda que aporta información relevante es la comprendida entre 20Hz y 5000Hz. De esta forma, se ha optado por la utilización de sensores piezoeléctricos, muy sensibles a las vibraciones, que son proporcionan una información muy valiosa acerca de las vibraciones que produce el centro de mecanizado en la fabricación de una horma.

Para la captación de esta información, se han utilizado sensores piezoeléctricos. El efecto piezoeléctrico es un fenómeno presentado por determinados cristales que al ser sometidos a tensiones mecánicas adquieren una polarización eléctrica en su masa, apareciendo una diferencia de potencial y cargas eléctricas en su superficie. Este fenómeno también se presenta a la inversa, esto es, se deforman bajo la acción de fuerzas internas al ser sometidos a un campo eléctrico. El efecto piezoeléctrico es normalmente reversible: al dejar de someter los cristales a un voltaje exterior o campo eléctrico, recuperan su forma.

Los materiales piezoeléctricos son cristales naturales o sintéticos que no poseen centro de simetría. El efecto de una compresión o de un cizallamiento consiste en disociar los centros de gravedad de las cargas positivas y de las cargas negativas. Aparecen de este modo dipolos elementales en la masa y, por influencia, cargas de signo opuesto en las superficies enfrentadas.

Una de las aplicaciones importantes de un cristal piezoeléctrico es su utilización como sensor de vibración. Cada una de las variaciones de presión producidas por la vibración provoca un pulso de corriente proporcional a la fuerza ejercida. Se ha convertido de una forma fácil una vibración mecánica en una señal eléctrica lista para amplificar.

[piezo\_1]

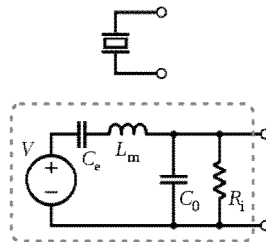


Imagen 19: Símbolo esquemático y modelo electrónico de un sensor piezoeléctrico.

Las propiedades eléctricas que presentan este tipo de sensores equivalen a un transductor eléctrico que tiene una muy alta impedancia de salida de corriente continua y puede ser modelado como una fuente proporcional de voltaje y como una red de filtro. El voltaje  $V$  en la fuente es directamente proporcional a la fuerza, presión o tensión aplicada. La señal producida está relacionada con esta fuerza mecánica como si hubiera pasado a través de un circuito equivalente. Un modelo detallado incluye los efectos de la construcción mecánica del sensor y otras no idealidades. La inductancia  $L_m$  es causada gracias a la masa sísmica y la inercia del propio sensor.  $C_e$  es inversamente proporcional a la elasticidad mecánica del sensor.  $C_0$

representa la capacitancia estática del transductor, la cual es resultado de la inercia de una masa de tamaño infinito.  $R_i$  es la resistencia de la salida del aislamiento del elemento del transductor. Si el sensor está conectado a una resistencia de carga, esto también actúa en paralelo con la resistencia del aislamiento, incrementando la alta frecuencia de corte.

[piezo\_2]

Para la creación de estos sensores piezoeléctricos, se utilizaron sensores comerciales que se pueden encontrar en tiendas especializadas, pero se realizaron sobre ellos una serie de modificaciones para que su funcionamiento mejorara. Una de las modificaciones que se añadieron estuvo la de la colocación de una barra de cobre de 0.5 gramos. Esta barra se ha colocado para que el sensor entre en resonancia y así incrementar la sensibilidad que presenta para captar las vibraciones, obteniendo de este modo un número mayor de datos que se nos escaparían si no se introdujera esta modificación. Del mismo modo, se ha diseñado y fabricado un circuito para la colocación de este sensor, optimizando su posterior colocación y reduciendo las pérdidas que se puedan originar en él, consiguiendo que las señales que nos proporcionan tenga una mayor amplitud y puedan ser captadas posteriormente por el circuito captador de señales.

Estos sensores se han colocado en cajas metálicas especialmente diseñadas según las medidas que presentan los mismos, para que su colocación en el centro de mecanizado sea de una forma sencilla y permita que el sensor este protegido de las especiales condiciones dentro del área de mecanizado, en la que hay presentes virutas y materiales de mecanizado, lubricantes, posibilidad de golpes, etc. De igual forma, estas cajas se han diseñado de tal forma que su composición no interfiera en los datos que van a registrar los sensores en el proceso de fabricación de la horma.

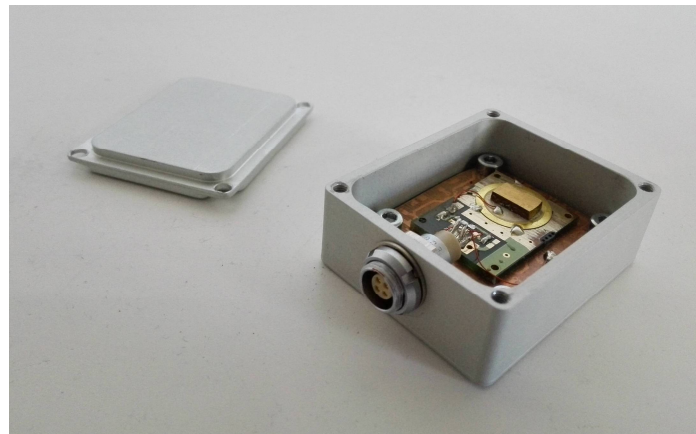


Imagen 20: Detalle del sensor piezoeléctrico en su caja de protección

Además de estos sensores, se ha realizado una circuitería específica para la captación no sólo de las señales que nos proporcionan estos sensores, sino también las potencias consumidas por el sistema, el avance de la fresa y las revoluciones de la fresa del centro de mecanizado. Para captar estas señales, se ha tenido que hacer un análisis pormenorizado del centro de mecanizado para saber de donde provienen las señales que necesitamos y como poder

captarlas para que sean comprensibles. Una vez realizado este análisis, se vio que la mejor manera posible de obtener estos valores era a partir del variador que posee el centro de mecanizado. Con el fin de obtener las señales modificando lo menos posible la estructura del centro de mecanizado, se introdujeron una serie de resistencias para bajar el nivel de tensión proporcionado por el sistema y que fuera compatible con el de nuestra tarjeta de adquisición. Antes de realizar esta modificación, se hicieron distintas pruebas sobre el variador para ver que salidas, de todas las que presenta este aparato eran las que nos interesaban para obtener los datos que necesitamos.

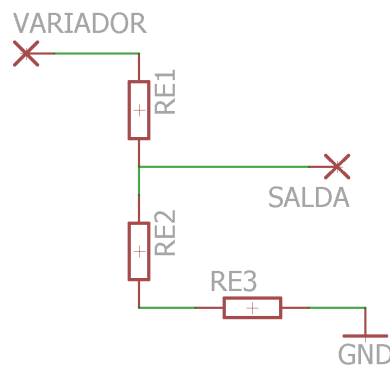


Imagen 21: Detalle del esquema para el acoplamiento de las señales a sistema

Este circuito vino a consecuencia de que las señales que se originan en el variador vienen con un voltaje máximo de 12 V este voltaje no se utiliza por nuestro sistema, puesto que lo más común entre los sensores es tener valores de 0 a 5V. Por esta razón, se diseñó el esquema que aparece en la figura anterior, el cual hace que los valores que nos proporciona el variador que van de 0 a 12V se reduzcan a unos valores de 0-5V sin que se origine pérdida de información. Se puede asegurar que no hay pérdida de información puesto que el circuito relaciona el voltaje original con el que obtenemos a la salida de forma directa y su relación es constante en todo el periodo, asegurándonos al 100% que, aunque en otros rangos de trabajo, los valores relacionados son siempre los mismos.

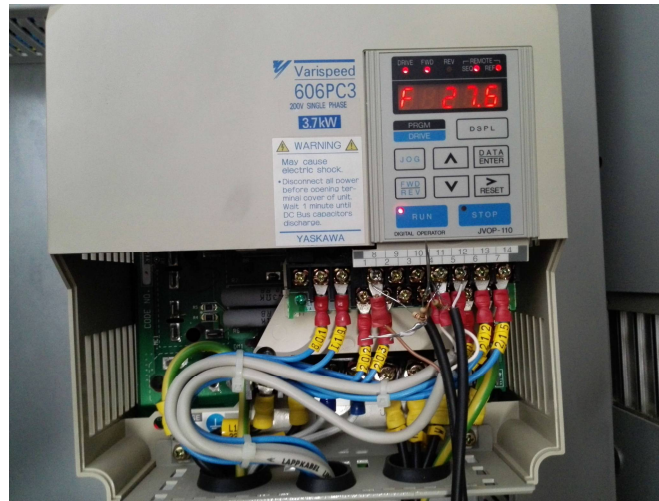


Imagen 22: Detalle de las modificaciones en el variador

Todos estos datos del centro de mecanizado han sido cableados primeramente desde sus puntos de información del centro de mecanizado hasta una caja que alberga la circuitería necesaria para interpretar las señales. Para ello se ha tenido que llevar a cabo un estudio del mismo hasta averiguar dónde se encontraba esta información relevante. Una vez conocida, se ha realizado un pequeño circuito de acoplamiento de señales para que estas se pudieran transmitir a nuestra tarjeta diseñada para captar las señales, tanto la de los sensores piezoeléctricos como las demás señales mencionadas.

El cableado del mismo ha consistido en la unión de todos estos puntos del centro de mecanizado con la tarjeta de adquisición de señales. Esta unión mediante cable se ha llevado a cabo manteniendo la estructura del centro de mecanizado y realizando las mínimas modificaciones posibles sobre él. Además la elección del cable ha sido clave, ya que se ha seleccionado un cable que no posea unas pérdidas elevadas y que no le afecte el ruido externo, consiguiendo que los datos que nos lleguen sean limpios y no estén enmascarados por agentes externos. A cableado se le han colocado terminales de conexión rápida y estándar, para que no se produzcan fallos en su conexión y se realice de una forma eficiente y rápida.

Este cableado termina en una caja en la que se encuentra la tarjeta de adquisición de datos. El propósito de esta tarjeta es recibir todos los datos que se originan en los distintos sensores y en el centro de mecanizado y tratarlos a un lenguaje comprensible y entendible, convirtiendo estas señales a dígitos numéricos.

La tarjeta de captación de datos está preparada para obtener un total de ocho canales distintos provenientes de sensores. Cada uno de estos canales son independientes entre sí. Los canales están conectados a un conversor analógico digital, que permite convertir las señales analógicas que nos llegan de los sensores a un formato digital. Las ventajas de convertir la información a digital son las siguientes:

- Cuando una señal digital es atenuada o experimenta perturbaciones leves, puede ser reconstruida y amplificada mediante sistemas de regeneración de señales.

- Cuenta con sistemas de detección y corrección de errores, que se utilizan cuando la señal llega al receptor; entonces comprueban (uso de redundancia) la señal, primero para detectar algún error, y, algunos sistemas, pueden luego corregir alguno o todos los errores detectados previamente.
- Facilidad para el procesamiento de la señal. Cualquier operación es fácilmente realizable a través de cualquier software de edición o procesamiento de señal.
- La señal digital permite la multigeneración infinita sin pérdidas de calidad.

[digital]

Una vez que la señal está en formato digital, con todas las ventajas que conlleva, un microcontrolador se ocupa de interpretarla. Este microcontrolador es el cerebro de la tarjeta de adquisición de datos y se ocupa de hacer las operaciones más relevantes con los datos de los sensores para su posterior representación en el ordenador. Este microcontrolador no sólo lleva a cabo estas operaciones sino que también se ocupa de su transmisión al ordenador cuando este lo solicita. El microcontrolador utilizado para este proyecto ha sido el P18F46K22 de la empresa Microchip. Se ha seleccionado este microcontrolador por la cantidad de puertos de entrada que tiene y por su conectividad a la hora de captar y enviar información. De igual forma se han tenido en cuenta otros factores como la dificultad a la hora de su programación, el tipo de memoria que tiene, las distintas configuraciones que se le pueden aplicar... factores importantes a la hora de captar señales a una velocidad muy elevada sin que se pierda información relevante por el camino.

Las características más significativas de este microcontrolador son las siguientes:

#### **CPU RISC de alto rendimiento**

- Compilador C optimizado para la arquitectura y conjunto de instrucciones
- Datos EEPROM de 1024 bytes
- Dirección de memoria de programación lineal de 64 Kbytes
- Direccionamiento de memoria de datos lineales de 4 Kbytes
- Hasta 16 MIPS de operación
- Instrucciones de 16 bits, 8-bit de ruta de datos
- Prioridad por niveles de interrupción
- 31 niveles de acceso por pila de hardware
- Multiplicador de hardware de único ciclo de 8 x 8

#### **Gestión extrema de baja potencia con nanoWatt XLP™:**

- Modo Sleep: 100 nA, típico
- Watchdog Timer: 500 nA, típico
- Reloj Timer1: 500 nA @ Estructura de oscilador flexible de 32 kHz típica
- Bloque de oscilador interno de precisión de 16 MHz:
  - Calibrado de fábrica al  $\pm 1\%$
  - Intervalo de frecuencias seleccionables por software de 31 kHz a 16 MHz
  - Rendimiento de 64 MHz disponible mediante PLL

- no se requieren componentes externos
- Cuatro modos de cristal hasta 64 MHz
- Dos modos de reloj externo hasta 64 MHz
- 4X Phase Lock Loop (PLL)
- Segundo reloj cuando se usa Timer1 @ 32 kHz
- Monitor de reloj con falla de seguridad:
  - Permite un apagado de seguridad si el reloj exterior se para
- Arranque del oscilador de dos velocidades

#### **Características especiales del microcontrolador:**

- Operaciones completas a 5.5V
- Opción de baja tensión disponible para operación de 1.8V-3.6V
- Auto programable bajo software de control
- Reinicio de encendido (POR), Encendido por timer (PWRT) y encendido por reloj (OST)
- Reajuste programable de Brown-out (BOR)
- Watchdog extendido (WDT) con reloj en chip y software habilitado
- Protección programable de código
- In-Circuit Serial Programming™ (ICSP™) con dos pines
- In-Circuit Debug con dos pines

#### **Características periféricas:**

- 24/35 pines de entrada/salida y 1 pin de entrada única:
  - Sumidero / fuente de alta corriente 25 mA/25 mA
  - Programación individual de pull-ups
  - Programación individual de interrupciones por cambio de estado de un pin
- Tres pines de interrupciones externas
- Hasta siete modules de Timer:
  - Hasta cuatro timers/contadores de 16-bit con preescalado
  - Hasta tres timers/contadores de 8-bit
  - Reloj Timer1 dedicado y de bajo consumo
- Hasta dos módulos de captura/comparación/PWM (CCP)
- Hasta tres módulos de captura mejorada/comparación/PWM (ECCP) con:
  - Una, dos o cuatro salidas PWM
  - Polaridad seleccionable
  - Tiempo muerto programable
  - Auto apagado y auto reinicio
  - Control de dirección de salida PWM
- Dos módulos Master Synchronous Serial Port (MSSP) con dos modos de operación:
  - 3 SPI (soporta los 4 modos SPI)
  - I2C™ modos maestro y esclavo
- Dos módulos de transmisor receptor síncrono asíncrono universal mejorado (EUSART):
  - Soporta RS-232, RS-485 y LIN 2.0





Imagen 24: Detalle de la caja de adquisición de datos con la tarjeta

Esta tarjeta de adquisición de datos se ha programado con un firmware específico para este proyecto, optimizando los recursos que se disponen en la tarjeta y optimizando la captura de los datos de los sensores. Este firmware ha sido diseñado pensando en cada momento en los tipos de sensores que se podían conectar a ella y los datos que se iban a obtener, teniendo rutinas diferentes para cada sensor, ya que los datos que se originaban tenían una naturaleza diferente, haciendo que el tratamiento de datos que tiene un sensor no se pueda tratar del mismo modo que otro sensor completamente diferente.

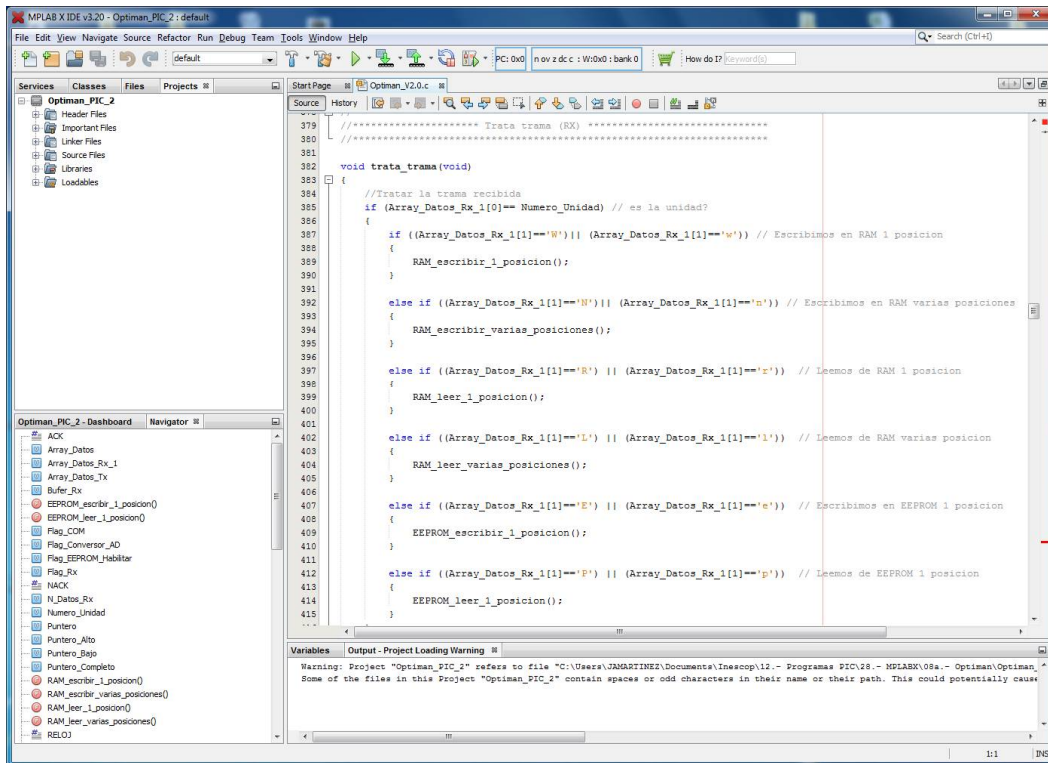


Imagen 25: Detalle del firmware programado

Todos los datos que se han obtenido del sistema, son enviados a un ordenador en el que está instalado un software de monitorización programado específicamente para la recepción de datos provenientes de la tarjeta de adquisición de datos. Los datos enviados al ordenador están en formato decimal y se organizan mediante un patrón de envío, asegurándonos de esta forma que la recepción es correcta si se tiene el mismo patrón y no lo es si ha habido alguna modificación por circunstancias ajenas al sistema.

De igual forma, se le ha añadido un código de comprobación CRC para verificar la integridad de los datos.

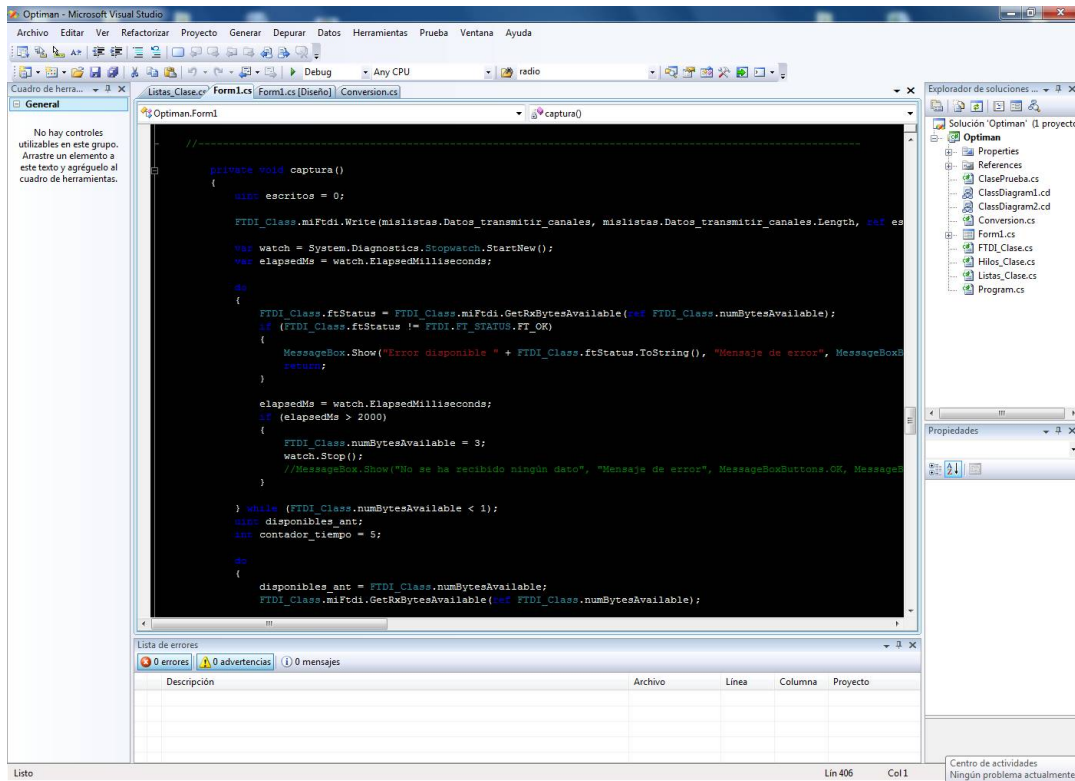


Imagen 26: Detalle del software programado

La secuencia de envío de datos desde la tarjeta al ordenador es la siguiente:

- Número de dispositivo
- Comando de lectura
- Posición del dato en la memoria de la tarjeta
- Número de canales que se han analizado
- Datos de los canales (cada dato es un canal)
- Código CRC

Si los datos recibidos en el ordenador por la tarjeta tienen esta secuencia, el software del ordenador los acepta como buenos y procede a su tratamiento, si no tienen esta secuencia, los datos son descartados como no válidos.

El tratamiento de los datos válidos por parte del ordenador consiste en una ordenación, acomodado y preparación de los mismos para poder ser enviados al sistema que posteriormente decidirá si la horma está bien o mal realizada. Al igual que ocurre con los datos que van de la tarjeta al software del ordenador, los datos que van del software del ordenador a la parte de decisión también llevan una secuencia para que sean comprensibles, esta secuencia es la siguiente:

- Fecha (delimitada por día, mes, año y hora hasta milisegundos)
- Sensor 1 (sensor que está más pegado a la fresa del centro de mecanizado)
- Sensor 2 (sensor que está más pegado al bloque de la horma)
- Potencia consumida
- Revoluciones de la fresa
- Paso de la fresa
- Delimitación si es afinado o desbaste

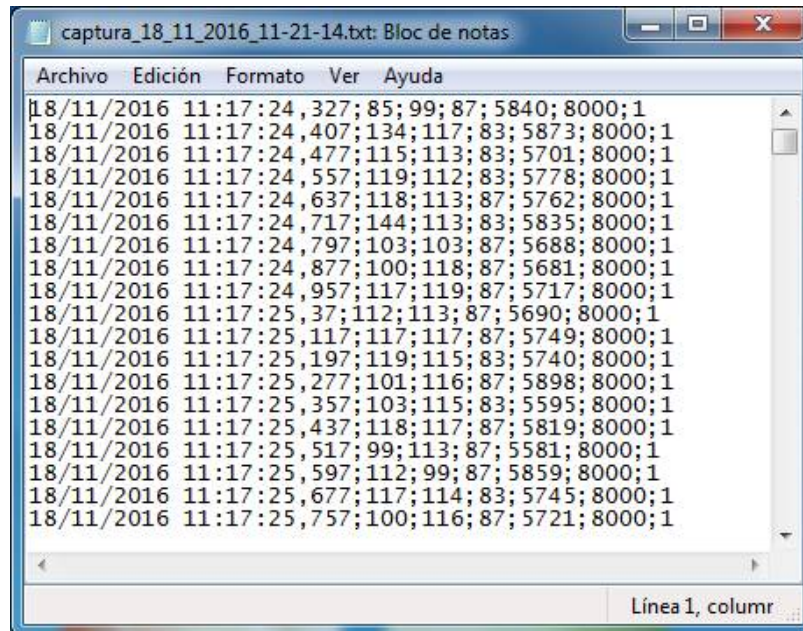


Imagen 27: Ejemplo de archivo de salida de datos

Los datos están separados mediante ";", delimitando así cada uno de los mismos y facilitando al sistema de detección saber en cada momento que es cada cosa. Esta exportación de datos siempre se repite, por lo que el sistema no tiene problemas en saber a qué elemento corresponde cada uno de los datos. De igual manera, esta exportación de datos se realiza sobre ficheros ".txt", ficheros que por su estructura no pesan mucho y son fácilmente manejables para su envío y recepción.

## 4. Arquitectura de software de sensorización

La arquitectura del software de sensorización está compuesta por diferentes módulos o componentes relativamente independientes entre sí que pueden ser encajados para dotar de las funcionalidades requeridas en del propio proyecto como para otros de índole similar.

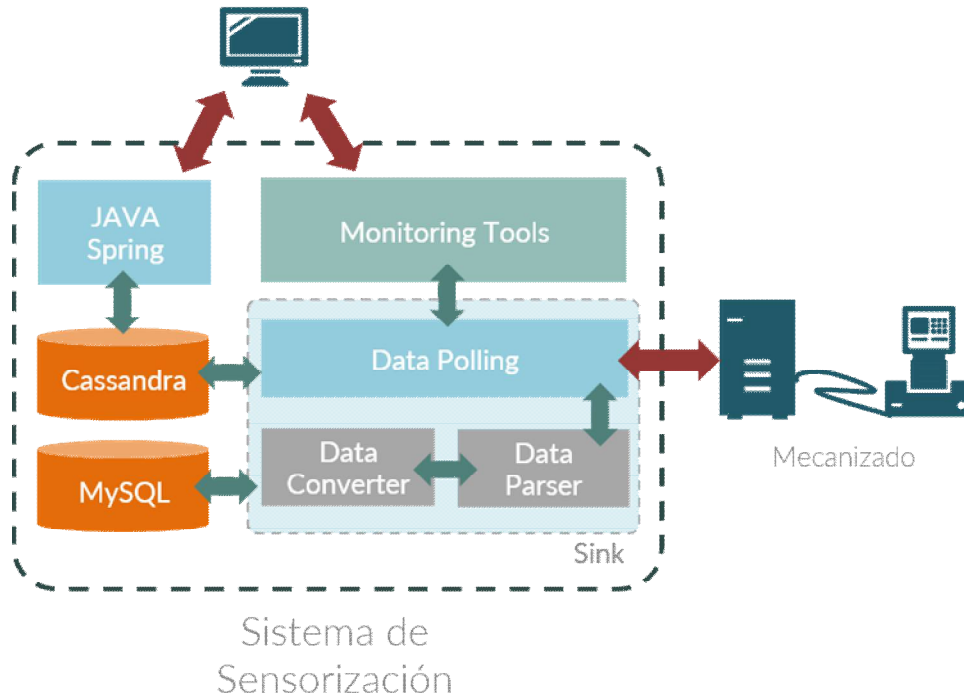


Imagen 28: Arquitectura del sistema

El sistema consta de los siguientes componentes principales:

- **Data Polling:** que es un script que activa y ejecuta cuando han sido programadas las tareas que han sido definidas dentro de la aplicación.
  - **Planificador de tareas:** que es un script que activa y ejecuta cuando han sido programadas las tareas que han sido definidas dentro de la aplicación.
  - **Tareas y "Trabajos":** las tareas son rutinas que se definen para ser ejecutadas periódicamente mediante la definición de un intervalo o cualquier otra condición de ejecución. Las tareas cuando son activadas ejecutan a su vez scripts que son denominados "Trabajos". Los trabajos pueden tener capacidad de ser ejecutados autónomamente y de forma explícita por el usuario sin requerir de una activación periódica.
  - **Modelo-lenguaje general para tratamiento/almacenamiento de información de sensorización:** que resulta de la definición de un modelo de datos que permite contener de la forma más abstracta posible diferentes arquitecturas y

aplicaciones de sensorización que pueden tenerse. El objetivo es agrupar en un único modelo diferentes modelos de arquitectura.

- **Data Parsers:** estos componentes permitirán traducir del modelo original de sensorización hacia el modelo-lenguaje general de tratamiento/almacenamiento. Se deberá crear un Data Parser por cada modelo de sensorización que se tenga.
- **Data Converters:** estos componentes son opcionales y permiten una vez pasada la sensorización al modelo-lenguaje general obtener como salida conversiones hacia otros lenguajes, modelos o aplicaciones.
- **Monitoring Tools:** que son diversas herramientas y utilidades que ayudan a mantener, gestionar y consultar el funcionamiento correcto del sistema.
- **Java Spring:** componente que hace de interfaz con Cassandra para servir los datos a la herramienta de monitorización

#### 4.1. Flujo de datos

Para el desarrollo del proyecto, se ha realizado un diseño de una arquitectura de sensorización denominada **Sink** y que está compuesta por diferentes módulos o componentes relativamente independientes entre sí que pueden ser encajados para dotar de las funcionalidades requeridas en del propio proyecto como para otros de índole similar.

El proceso que implementa Sink es el siguiente:

1. El sistema recoge los log de los sensores del disco duro mediante el componente de Data Polling, estos datos en crudo se redirigen al componente de Data Parsing que se encarga de interpretar los datos.
2. Posteriormente se transfieren los datos parseados al componente de persistencia para su almacenamiento temporal en MySQL.
3. Finalmente el componente Data Converter que adapta los datos a Cassandra para el almacenamiento definitivo de los datos de los sensores.

En el siguiente gráfico puede apreciarse el flujo de información cómo se llevaría a cabo entre los diferentes componentes desde el mecanizado hasta la herramienta de visualización.

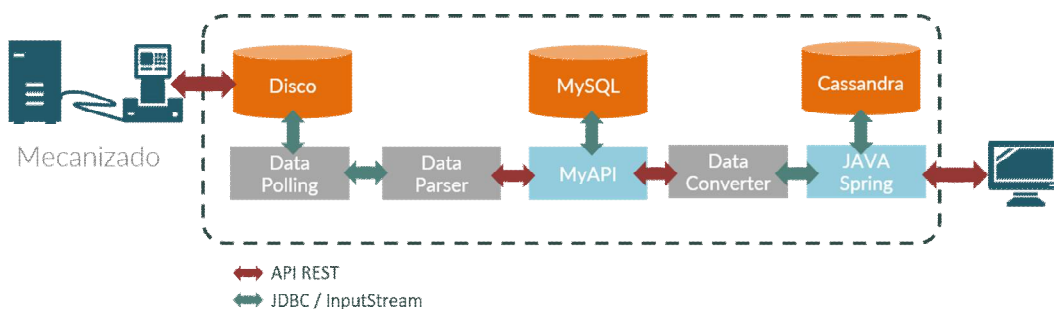


Imagen 29: Diagrama de secuencia

## 4.2. Componentes del Sistema de Sensorización

### 4.2.1 Data Polling

A continuación, se muestra en detalle los elementos más importantes del componente Data Polling.

#### Planificador de tareas

El planificador de tareas es un script compuesto de los siguientes ficheros:

- **OPTIMAN\_Data\_Polling.py**: que es el script principal que establece un planificador de tareas. Se encuentra ejecutándose continuamente comprobando a
- **OPTIMAN\_Data\_Polling\_Constants.py**: que es el script que contiene constantes de configuración para el script principal, entre las que se encuentran el tiempo en el cual el sistema permanece dormido o si el propio script imprime información sobre el avance de ejecución del mismo.

El planificador espera una lista de tareas que han sido creadas y configuradas en el fichero `OPTIMAN_Data_Polling_Tasks.py` (que se explicará más adelante en este documento) denominada como `taskList`. Las tareas tienen establecida una condición de activación periódica que el propio planificador se encarga de comprobar y activar cuando esta se cumple. Se pueden tener tantas tareas como se necesiten dentro del sistema.

Para activar el planificador se realizará con el comando:

```
$ python OPTIMAN_Data_Polling.py loop
```

A partir de este momento el planificador se ejecutará en segundo plano. Cuando el planificador se activa tras permanecer el tiempo establecido en modo "dormido" chequea cada una de las tareas definidas comprobando si deben activarse o no mediante el método de la tarea `checkInterval()`. Si la tarea se activa se invoca al método de la tarea `action()`. Una vez ejecutada la tarea se reprograma el nuevo intervalo de activación mediante la invocación al método `setNextInterval()`. A los anteriores comandos el planificador les facilita el valor de timestamp actual y un valor de temporizador virtual que el propio planificador mantiene por utilidad.

#### Tareas y "Trabajos"

El sistema distingue entre dos tipos de scripts para la activación y ejecución de eventos definidos por el usuario que son:

- **Tareas**: que son objetos que incluyen código que permite programar la activación periódica de la misma.
- **"Trabajos" o Jobs**: son scripts que realizan acciones tales como la invocación de comandos del sistema. Estas acciones son ejecutadas de forma explícita y deben poder funcionar ejecutándolas manualmente.

Las tareas son objetos definidos por el usuario que sirven para definir eventos periódicos en el sistema. De forma obligada derivan de la clase “abstracta” padre `OPTIMAN_Data_Polling_Task`. Cada tarea debe definir los siguientes métodos:

- `checkInterval(self, timestamp = None, timer = None)`: que es el método que es invocado cada vez que el planificador “despierta” y permite determinar si debe activarse o no la acción definida para la tarea.
- `setNextInterval(self, timestamp = None, timer = None)`: que es el método que una vez finalizada la acción definida cuando se activó la tarea. Permite establecer el siguiente intervalo de activación para la tarea.
- `action(self, timestamp = None, timer = None)`: que es en sí el método que contiene el código que debe ejecutarse con la tarea. Usualmente, dentro de la definición de las acciones se invocarán a diferentes ficheros de scripts de tipo “trabajo”.

Las tareas son creadas en el fichero de script `OPTIMAN_Data_Polling_Tasks.py` y son añadidas a la lista de nombre `taskList` que será la lista que espera el planificador.

Por otra parte, los **trabajos** o **jobs** son los scripts que realizan las diferentes actividades y comandos en el sistema como obtener datos de sensorización de diferentes fuentes. Aquí se definirán diferentes funciones que realizarán las acciones pertinentes. Las tareas que invoquen a estos trabajos deben conocer la forma de invocación correcta a estas funciones. Además, es recomendable que los propios trabajos puedan ser invocados desde el símbolo del sistema como scripts individuales.

## Modelo-lenguaje general

Uno de los objetivos principales en el diseño del sistema de sensorización es la definición de un modelo-lenguaje lo suficientemente abstracto que permita caracterizar y almacenar datos provenientes de diferentes aplicaciones y sistemas. Este modelo general permite actuar de puente hacia otros sistemas o modelos destino, de tal forma que desde él puedan portarse o generarse salida a diferentes modelos destino simplemente partiendo de diversas implementaciones de este lenguaje general hacia el modelo destino. Además, este lenguaje general mantiene un repositorio intermedio que puede ser empleado para generar los resultados de salida a posteriori o bien como base de datos intermedia hasta su final procesado posterior. El modelo, lenguaje y esquema que da soporte a este se ha denominado `Sensor Data Store` y está implementado en los ficheros de script `SDSServer_Iface_Python_v_1_3.py` y `ConstantsSDSServerIface.py`. Adicionalmente se tiene una librería de utilidad de uso común denominada `SDS_Profile_Iface_v_1_3.py` que proporciona métodos para la conexión y gestión de las llamadas a los servicios web ofrecidos. Hay que recalcar que este es un componente adicional de la arquitectura que puede ser empleado por otras aplicaciones.

El lenguaje general define las siguientes abstracciones principales:

- **Data Source:** que define a una entidad superior de la cual dependen elementos que son capaces de proveer sensorización, es decir, sensores.
- **Sensor:** que son aquellos elementos que disponen de sondas que permiten capturar valores físicos del entorno.
- **Channel:** que son cada una de las sondas que proveen valores del entorno y que están asociadas a un sensor.

El lenguaje intermedio requiere que previamente se defina el catálogo de recursos de cada Data Source, es decir, la definición de todos los sensores y canales asociados al mismo. Una vez declarados, se podrá enviar información de sensorización de cada sistema basándose en este modelo.

#### 4.2.2 Data Parsers

Los *Data Parsers* son entidades que procesan los datos que han sido obtenidos a través de las diferentes Tareas y Jobs y realizan la traducción de esos datos hacia el formato de lenguaje-modelo común definido en la arquitectura. Por cada uno de los posibles tipos de origen de datos que se tengan en el sistema se deberá crear un *Data Parser* particular para traducir de ese formato origen a ese formato final. Adicionalmente, los *Data Parsers* se les permite asociar una serie de *Data Converters* (que serán explicados más adelante en este documento) que permitirán traducir de este lenguaje intermedio a otros formatos y modelos de salida en función de las necesidades del sistema.

Usualmente (aunque no es obligatorio), cada Data Parser tendrá implementadas las siguientes funciones:

- `init_converters()`: inicializa los Data Converters que serán utilizados con este Data Parser. Es invocado al inicio de del proceso de traducción los ficheros de sensorización.
- `close_converters()`: que finaliza la conexión con los Data Converters empleados con este Data Parser. Es invocado al finalizar el proceso de traducción de los ficheros de sensorización.
- `call_converters(measurementIdList, rawData)`: que es invocado cada vez que se realiza un procesado de datos. Se facilita la lista de identificadores obtenidos de la base de datos en la inserción junto con el conjunto de datos en "crudo" extraídos del fichero de origen por si la rutina de proceso de lenguaje de salida las necesita.
- `register_datasource()`: que es un rutina que permite declarar un Data Source en el sistema.

- `register_catalogue(fileName)`: que es un rutina que permite declara el catálogo de un Data Source en el sistema. Opcionalmente, y si se implementa, puede facilitarse un nombre de fichero desde el cual leer y obtener los datos.
- `get_measurements(duration, fileName, dataSourceName, sensorName)`: que es una rutina que permite leer un fichero de datos de origen y traducirlo al lenguaje-modelo general. Como parámetros opcionales se permite especificar la duración o delay que puede introducirse en el sistema con cada medida que es introducida, el nombre de fichero de donde obtener los datos y los nombres del Data Source y el Sensor que se están introduciendo las medidas.

Los *Data Parsers* por norma general serán llamados cuando una tarea se haya activado en el sistema, usualmente invocando la llamada del mismo `get_measurements()`.

Como convención los *Data Parsers* tendrán como estructura de nombre de fichero la siguiente forma: `DP_nombre.py`. Para el fichero que define las constantes se tendrá `DP_nombre_Constants.py`.

### 4.2.3 Data Converters

Los *Data Converters* son entidades que realizan una traducción o acción como resultado final de todo el proceso. Pueden implementarse tantos *Data Converters* como se necesiten, pudiendo obtener los datos o bien del listado de medidas que han sido introducidas en el repositorio general como del análisis y traducción de cada los elementos en "crudo" obtenidos de por el *Data Parser* en el sistema. Usualmente (aunque no es obligatorio), los *Data Converters* implementarán las siguientes funciones:

- `init_converter()`: que contiene el código de inicialización del Data Converter. Usualmente esta función será llamada desde el método `init_converters()` del Data Parser.
- `end_converter()`: que contiene el código de cierre o finalización del Data Converter. Usualmente esta función será llamada desde el método `end_converters()` del Data Parser.
- `dataSourceRegister()`: función que permite registrar un Data Source en el modelo de datos final, pudiendo ser obtenido a partir del modelo-lenguaje general.
- `catalogueRegister()`: función que permite registrar el catálogo de un Data Source en el modelo de datos final, pudiendo ser obtenido a partir del modelo-lenguaje general.
- `insertMeasurements(measurementIdList, rawData, parameters)`: función que permite registrar las medidas provenientes del Data Parser, recibéndose como argumentos la lista de identificadores de medidas obtenidas por el Data Parser en la base de datos, los datos en crudo facilitados por el Data Parser así como adicionalmente una lista de parámetros de utilidad.

Como convención los *Data Converters* tendrán como estructura de nombre de fichero la siguiente forma: `DC_nombre.py`. Para el fichero que define las constantes se tendrá `DC_nombre_Constants.py`.

#### 4.2.4 Monitoring Tools

Para facilitar la monitorización del funcionamiento del sistema, la arquitectura está diseñada de tal manera que pueda ser modificada para informar sobre el estado, eventos y otras acciones que se van realizando en ella. Las principales vías para activar la monitorización del sistema son las siguientes:

- **Fichero de *log*:** en el cual se almacena en un fichero los distintos eventos que van ocurriendo en el sistema.
- **Ejecución de Tareas periódicas de monitorización :** en las cuales pueden crearse tareas que serán iniciadas por el planificador de Optiman y pueden realizar operaciones como pings, pruebas de conexión, solicitud de información, etcétera a servidores y aplicaciones de tal manera que se monitorice así el estado de los componentes principales del sistema.

De momento, el sistema integra el sistema de *log* o información de eventos que va generando información sobre el estado y avance de la aplicación volcándose sobre un fichero. Este *log* es mantenido a través de la clase `MT_Logger.py` que implementa las facilidades para generar y escribir en el fichero aquellos eventos que desean ser notificados, clasificándose los mismos según la el tipo o severidad del propio mensaje (información, alarma, error, etcétera). Por defecto, el fichero con esta información tiene el nombre de `optiman.log`.

## 5. Desarrollo del sistema de sensorización

### 5.1. Estructura de datos del sistema de sensorización

A continuación se muestra y describe el esquema de la base de datos empleado para dar soporte a este lenguaje-modelo general. La base de datos empleada es MySQL siendo su esquema el siguiente:

```
CREATE TABLE CONFIG_SDSServerInterface (
  SRV_serverId VARCHAR(255) NOT NULL COMMENT 'Unique identifier for this server',
  SRV_address VARCHAR(255) COMMENT 'Network address of the server',
  SRV_version VARCHAR(64) COMMENT 'Indicates the version of the SDS Server Model',
  SRV_description VARCHAR(255) COMMENT 'Description of the system',
  SRV_optional VARCHAR(255) COMMENT 'Used for additional info or other purposes',
  PRIMARY KEY(SRV_serverId)
);

-- SYSTEM REGISTER TABLES

-- Table for source communicator (Bubble, gateway, concentrator/aggregator,...)

-- Old name: DataSource --> New name: CATALOGUE_DataSource
-- Field prefix: DS

CREATE TABLE CATALOGUE_DataSource (
  DS_dataSourceId INT NOT NULL AUTO_INCREMENT,
  DS_dataSourceName VARCHAR(255) NOT NULL UNIQUE COMMENT 'Name identifier for the data
source (Gateway, Coordinator, ...)',
  DS_address VARCHAR(255) COMMENT 'Network address of the data source',
  DS_description VARCHAR(255) COMMENT 'Brief description of the data source',
  DS_active BOOLEAN COMMENT 'Indicates if the data source is currently active',
  PRIMARY KEY (DS_dataSourceId)
);

-- Table for sensor catalogue

-- Old name: SensorInfo --> New name: CATALOGUE_Sensor
-- Field prefix: SEN

CREATE TABLE CATALOGUE_Sensor (
  SEN_sensorId INT NOT NULL AUTO_INCREMENT,
  SEN_dataSourceId INT NOT NULL COMMENT 'Identifier at which the sensor is attached',
  SEN_sensorName VARCHAR(255) NOT NULL UNIQUE COMMENT 'Name identifier for the
sensor',
  SEN_address VARCHAR(255) COMMENT 'Network address of the sensor',
  SEN_description VARCHAR(255) COMMENT 'Brief description of the sensor',
  SEN_sensorType VARCHAR(64) COMMENT 'Indicates the sensor type (e.g. temperature,
humidity, co2, ...)',
  PRIMARY KEY (SEN_sensorId),
  CONSTRAINT fk_CATALOGUE_Sensor_dataSourceId FOREIGN KEY (SEN_dataSourceId)
REFERENCES CATALOGUE_DataSource(DS_dataSourceId)
);

-- Table for SensorChannels

-- Old name: SensorChannels --> New name: CATALOGUE_SensorChannel
-- Field prefix: SENCH

CREATE TABLE CATALOGUE_SensorChannel (
  SENCH_dataSourceId INT NOT NULL COMMENT 'Identifier at which the sensor is
attached',
  SENCH_sensorId INT NOT NULL COMMENT 'Unique identifier for the sensor',
  SENCH_channelId INT NOT NULL COMMENT 'Sensor channel measure identifier',
  SENCH_unit VARCHAR(255) COMMENT 'Standard unit used',
```

```
    SENCH_maximumValue VARCHAR(255) COMMENT 'Maximum value of the measures',
    SENCH_minimumValue VARCHAR(255) COMMENT 'Minimum value of the measures',
    SENCH_granularity VARCHAR(255) COMMENT 'Granularity of the measures',
    SENCH_tolerance VARCHAR(255) COMMENT 'Precision of the measures',
    SENCH_sensorDataType VARCHAR(32) COMMENT 'Indicates the measure data type: INT,
FLOAT, ... ',
    SENCH_location VARCHAR(255) COMMENT 'UTM or WGS84 coordinates',
    SENCH_channelEnabled BOOLEAN COMMENT 'Indicates if the channel is enabled',
    PRIMARY KEY (SENCH_dataSourceId, SENCH_sensorId, SENCH_channelId),
    CONSTRAINT fk_CATALOGUE_sensorChannel_dataSourceId_sensorId FOREIGN KEY
(SENCH_dataSourceId, SENCH_sensorId)
REFERENCES CATALOGUE_Sensor(SEN_dataSourceId, SEN_sensorId)
);
```

```
-- Tables for incoming Data
```

```
-- Old name: IncomingData --> New name: DATA_IncomingData
-- Field prefix: MS
```

```
CREATE TABLE DATA_IncomingData (
    MS_measureId INT NOT NULL AUTO_INCREMENT COMMENT 'Autogenerated field to set a
unique identifier for each measure',
    MS_dataSourceId INT NOT NULL COMMENT 'Data source identifier',
    MS_sensorId INT NOT NULL COMMENT 'Sensor identifier',
    MS_measureEntryTimestamp TIMESTAMP COMMENT 'Measure timestamp set by the server',
    PRIMARY KEY (MS_measureId, MS_dataSourceId, MS_sensorId),
    CONSTRAINT fk_DATA_incomingData_dataSourceId_sensorId FOREIGN KEY (MS_dataSourceId,
MS_sensorId)
REFERENCES CATALOGUE_Sensor(SEN_dataSourceId, SEN_sensorId)
);
```

```
-- Old name: IncomingDataChannels --> New name: DATA_IncomingDataChannel
-- Field prefix: MSCH
```

```
CREATE TABLE DATA_IncomingDataChannel (
    MSCH_measureId INT NOT NULL COMMENT 'Reference to measure Id',
    MSCH_channelId INT NOT NULL COMMENT 'Sensor channel measure identifier',
    MSCH_measureTimestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP COMMENT 'Measure timestamp
generated by the sensor or data source',
    MSCH_measureValue BLOB COMMENT 'Measure value',
    PRIMARY KEY (MSCH_measureId, MSCH_channelId),
    CONSTRAINT fk_DATA_incomingDataChannel_measureId FOREIGN KEY (MSCH_measureId)
REFERENCES DATA_IncomingData(MS_measureId)
);
```

A continuación se definen las tablas y campos del esquema.

## CONFIG\_DewiServerIface

Esta tabla almacena la configuración y descripción del servidor donde se aloja la base de datos. Sus principales campos son:

- **SRV\_serverId:** que establece un nombre identificador único para el servidor donde se alberga la base de datos.
- **SRV\_address:** que almacena la dirección del servidor.
- **SRV\_version:** que almacena la versión del servidor.
- **SRV\_description:** que almacena una descripción adicional sobre el servidor.
- **SRV\_optional:** que es un campo adicional que puede ser empleado para albergar información adicional de configuración o similar.

## CATALOGUE\_DataSource

Esta tabla almacena la información de cada uno de los *Data Sources* que han sido declarados en el sistema. Los principales campos son:

- **DS\_dataSourceId:** que establece un identificador numérico interno para el Data Source. Es generado automáticamente por la base de datos como un campo auto-incremental.
- **DS\_dataSourceName:** que establece un nombre identificativo único para el Data Source definido por el usuario y que permita facilitar la identificación del mismo.
- **DS\_address:** que declara la dirección de internet del propio Data Source.
- **DS\_description:** que proporciona información adicional descriptiva sobre el Data Source.
- **DS\_active:** que sirve para indicar si el Data Source permanece actualmente activo o no en el sistema.

## CATALOGUE\_Sensor

Esta tabla sirve para albergar la información relativa al catálogo de un *Data Source*. Sus campos son:

- **SEN\_sensorId:** que establece un identificador numérico interno para el Sensor. Es generado automáticamente por la base de datos como un campo auto-incremental.
- **SEN\_dataSourceId:** que establece el identificador numérico del *Data Source* al cual está asociado.
- **SEN\_sensorName:** establece un nombre identificativo único descriptivo para el Sensor, facilitando su identificación en el sistema.
- **SEN\_address:** que es utilizado para almacenar la dirección (si se tiene) del Sensor.
- **SEN\_description:** que permite proporcionar información descriptiva adicional sobre el Sensor.
- **SEN\_sensorType:** que indica el tipo de Sensor. Esta información puede ser dependiente del usuario, modelo y/o aplicación (ejemplos: humedad, temperatura, máquina XXYYYYZ, ubicación XXYYYYZ, etcétera).

## CATALOGUE\_SensorChannel

Esta tabla almacena información sobre los canales asociados a los diferentes sensores. Los principales campos son:

- **SENCH\_dataSourceId:** hace referencia al Data Source al cual está asociado el Sensor.
- **SENCH\_sensorId:** hace referencia al identificador de Sensor.
- **SENCH\_channelId:** establece un identificador entero único para el canal.

- **SENCH\_unit**: usado para especificar las unidades que obtiene la sonda (ejemplos: C, meters, V, etcétera).
- **SENCH\_maximumValue**: valor máximo que puede ser proporcionado por el canal.
- **SENCH\_minimumValue**: valor mínimo que puede ser proporcionado por el canal.
- **SENCH\_granularity**: granularidad del canal.
- **SENCH\_tolerance**: error o tolerancia máxima que puede proporcionar el canal.
- **SENCH\_sensorDataType**: permite establecer el tipo de canal. Esta información puede ser dependiente del usuario, modelo y/o aplicación (ejemplos: integer, boolean o medidas de temperatura, conjunto de valores de voltaje, etcétera).
- **SENCH\_location**: permite establecer las coordenadas geográficas de la sonda. Hay que indicar que un sensor puede tener dispuestas sondas (canales) localizadas en diferentes canales.
- **SENCH\_channelEnabled**: indica si el canal está siendo utilizado o no actualmente.

### DATA\_IncomingData

Esta tabla almacena medidas procedentes de cada uno de los *Data Sources* y *Sensors* del sistema. Los campos son:

- **MS\_measureId**: que establece un identificador único para la medida.
- **MS\_dataSourceId**: que establece el identificador para el Data Source para la medida.
- **MS\_sensorId**: que establece el identificador del Sensor para la medida.
- **MS\_measureEntryTimestamp**: establece la marca temporal en el cual la medida ha sido insertada en el sistema.

### DATA\_IncomingDataChannel

Esta tabla almacenada los valores de cada canal asociada a una medida. Los campos son:

- **MSCH\_measureId**: asocia el valor a la medida introducida.
- **MSCH\_channelId**: indica el identificador de canal para el valor.
- **MSCH\_measureTimestamp**: establece la marca temporal para el cual el valor del canal fue obtenido.
- **MSCH\_measureValue**: que indica el valor del canal.

## 5.2. Interfaz REST

Toda operación con este lenguaje intermedio se realiza mediante la invocación de servicios web. Las principales funciones de estos servicios son:

- **Registro de Data Sources:** que permite definir y registrar los Data Sources en el sistema.
- Registro del catálogo: que permite definir el catálogo de un Data Source en el sistema.
- **Envío de datos:** que permite registrar medidas procedentes de Data Sources y Sensores.
- **Consulta de datos:** que permite consultar los datos almacenados en el sistema.

El interfaz de los servicios web es de tipo RESTful y se han implementado en lenguaje Python. Los servicios se clasifican en función de su objetivo:

- **Input services:** que sirve para proporcionar datos al sistema (registro de Data Sources, Sensores y medidas).
- **Output services:** utilizados para solicitar y consultar los datos almacenados.

Los servicios pueden invocarse a través de la siguiente URL:

```
http://server_name:port/sds-server-ivace/rest/[input|output]/service_name
```

La forma de operar es la siguiente:

1. Cada *Data Source* debe registrarse en el sistema. Puede facilitar a su vez todo su catálogo.
2. Si el catálogo no ha sido proporcionado con el registro del *Data Source*, debe facilitarse a continuación.
3. Desde este momento puede enviarse datos de sensorización hacia el servidor.

Los servicios son los siguientes:

### Input Services

Los servicios de entrada permiten registrar información en el sistema, ya sea el registro de las fuentes de datos y sensores como las propias medidas que van a generándose.

### DataSourceRegister

Este servicio permite declarar un *Data Source* en el sistema. Puede ser accedido a través de la siguiente URL:

```
input/datasourceregi ster
```

Para hacer esto, debe generarse un mensaje POST con un contenido application/JSON que incluye la información del *Data Source*. Hay que reseñar que puede incluirse en la llamada a este servicio la propia declaración del registro del catálogo. Si no desea declararse el catálogo puede ponerse el valor *null*. El JSON sigue la siguiente estructura:

```
{
  "dataSourceName": "string",
  "address": "string",
  "description": "string",
  "sensors": [
    {
      "sensorName": "string",
      "address": "string",
      "description": "string",
      "sensorType": "string",
      "sensorChannels": [
        {
          "channelId": "string",
          "unit": "string",
          "maximumValue": "string",
          "minimumValue": "string",
          "granularity": "string",
          "tolerance": "string",
          "sensorDataType": "string",
          "location": "string"
        }
      ]
    }
  ]
}
```

## CatalogueRegister

Este servicio permite registrar los recursos o catálogo asociado a un *Data Source*. Este servicio puede ser accedido a través del siguiente URL:

`input/catalogueregister`

Para hacer esto, debe generarse un mensaje POST con un contenido *application/JSON* que incluye la información del catálogo del *Data Source*. El JSON tendrá la siguiente estructura:

```
{
  "dataSourceName": "string",
  "sensors": [
    {
      "sensorName": "string",
      "address": "string",
      "description": "string",
      "sensorType": "string",
      "sensorChannels": [
        {
          "channelId": "string",
          "unit": "string",
          "maximumValue": "string",
          "minimumValue": "string",
          "granularity": "string",
          "tolerance": "string",
          "sensorDataType": "string",
          "location": "string"
        }
      ]
    }
  ]
}
```

## IncomingData

Este servicio puede ser usado para registrar las medidas que van generando los *Data Sources* en el sistema. Puede ser accedió mediante el siguiente URL:

```
input/incomingdata
```

Para hacer esto, debe generarse un mensaje POST con un contenido *application/JSON* que incluye la información de las medidas provenientes de un *Data Source*. El JSON tiene la siguiente estructura:

```
{
  "dataSourceName": "string",
  "incomingDataSensorPayloadMessage": [
    {
      "sensorName": "string",
      "incomingDataSensorChannelsPayloadMessage": [
        {
          "channelId": "string",
          "measureValue": "string",
          "measureTimestamp": "string"
        }
      ]
    }
  ]
}
```

## DataSourceChangeState

Este servicio es usado para indicar el estado (*enabled/disabled*) de un *Data Source*. Puede ser accedió a través de la siguiente URL:

```
input/datasourcechangestate/{dataSourceName}/{status}
```

Para hacer esto, debe generarse un mensaje indicando como parámetros el nombre del *Data Source* y el estado que desea establecerse: *enabled* or *disabled*.

## Output services

Los servicios permiten a los usuarios y aplicaciones obtener los datos que están almacenados en la base de datos. Los principales servicios son:

### DataSourceCatalogue

Este servicio permite obtener un el catálogo registrado para un *Data Source* particular. Puede ser accedió a través del siguiente URL:

```
output/datasourcecatalogue/{dataSourceName}
```

Para hacer esto, debe enviarse un mensaje de tipo GET. Como resultado se obtendrá un mensaje de tipo *application/JSON* con el contenido del catálogo.

## RegisteredDataSources

Este servicio permite obtener la lista de todos los *Data Sources* registrados en el sistema. Puede ser accedido a través de la siguiente URL:

```
output/registereddatasources
```

Para hacer esto, debe enviarse un mensaje de tipo GET. Como resultado se obtendrá un mensaje de tipo *application/JSON* con todos los *Data Sources* registrados en el sistema.

## Measurements

Este servicio permite obtener las medidas almacenadas en la base datos. Para estos servicios pueden establecer un conjunto de filtros adicionales para obtener la información demandada. Esto puede realizarse a través de la siguiente URL:

```
output/data  
output/data/{dataSourceName}  
output/data/{dataSourceName}/{sensorName}  
output/data/{dataSourceName}/{sensorName}/{channelId}
```

Para hacer esto, debe enviarse un mensaje de tipo GET. Como resultado se obtendrá un mensaje de tipo *application/JSON* con el conjunto de medidas solicitadas.

## Info

Este servicio permite obtener información sobre el servidor. Puede ser accedida a través de la siguiente URL:

```
output/info
```

Para hacer esto, debe enviarse un mensaje de tipo GET. Como resultado se obtendrá un mensaje de tipo *application/JSON* con la información del servidor.

## 5.3. Inicio, configuración y arranque de la arquitectura

En primer lugar, deberán revisarse todos aquellos ficheros de constantes de cada uno de los módulos para permitir la configuración de las diferentes constantes y directivas que afectan a la definición de direcciones y nombres de servidores. Es de particular importancia definir los siguientes:

- **Servidor SDS** : configurando el script `SDS_Profile_Iface_v_1_3.py`.
- **Credenciales de acceso a la base de datos MySQL**: configurando el script `ConstantsSDSServerIface.py`.
- **Servidor Cassandra** : configurando el script `DC_Cassandra_Constants.py`.

Hay que destacar que en el servidor donde se instala toda la arquitectura, y en especial el modelo-lenguaje general deberá crearse una base de datos que contenga el esquema que se facilita en el fichero `sds_server_iface_schema_v_1_3.sql` creando un usuario con permisos suficientes para el manejo y creación de esa misma base de datos.

Es importante recalcar que siempre que se deseen procesar datos de lecturas de sensorización, la base de datos del modelo-lenguaje general deberá haber registrado los *Data Sources* y los *catálogos* pertinentes del modelo del sistema.

Finalmente, dentro de la carpeta que contiene los scripts se tendrán dos sub-carpetas principales:

- **SensorDataStore:** que alberga los scripts relativos al *SensorDataStore*. Para arrancarlo se deberá ejecutar el siguiente comando (el puerto puede establecerse aquel que se considere):

```
$sudo python SDSServerIfacePython_v_1_3.py 8081
```

- **OptimanSensing:** que alberga toda la arquitectura de sensorización de Optiman. Para arrancarlo se deberá ejecutar el siguiente comando:

```
$python OPTIMAN_Data_Polling.py loop
```

### 5.3.1. Notas adicionales sobre la instalación

Para la instalación del sistema es posible que algunos paquetes de librerías de Python no estén instalados en el sistema. Algunos de ellos son:

- Web.py : necesario como framework para la implementación de servicios web.  
`pip install web.py`
- MySQL-python : necesario para el acceso a la base de datos MySQL.  
`apt-get install python-dev libmysqlclient-dev`  
`pip install MySQL-python`
- PyOpenSSL : para el caso en que se habilite el uso de SSL en los servicios web.  
`pip install pyopenssl`

## 6. Herramienta de monitorización

Esta tarea se centra en el diseño y desarrollo de una herramienta de visualización de los resultados que la arquitectura de software implementado, facilitando la interpretación de los mismos por parte de los usuarios.

A tal fin se ha desarrollado un componente de monitorización basado en Python integrado vía HTML en la interfaz web del sistema.

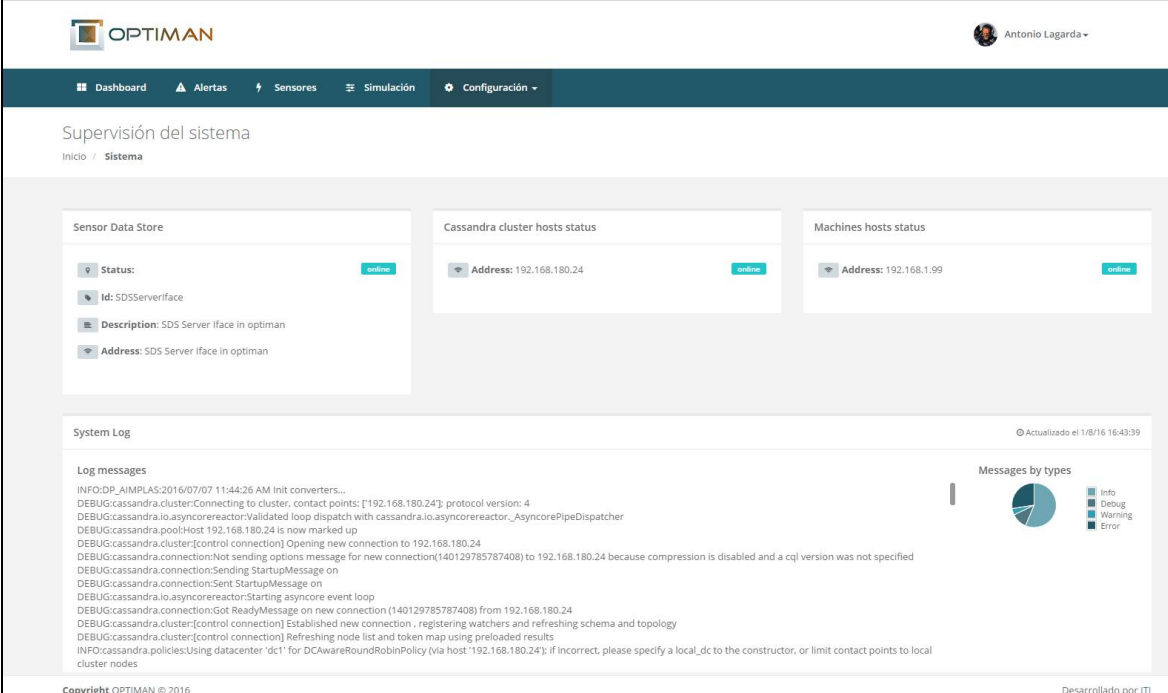


Imagen 30: Herramienta de monitorización

Este script en Python se encarga de monitorizar los siguientes componentes:

- Software de sensorización
- Clúster Cassandra de almacenamiento de datos
- Máquina que hace el hosting del sistema
- Logs del sistema

## 6.1. Monitorización del software de sensorización

La herramienta intenta la conexión con el software de sensorización y en caso de éxito muestra información sobre su identificador, descripción, dirección y su status.



Imagen 31: Herramienta de monitorización del software de monitorización

## 6.2. Monitorización del cluster Cassandra

La herramienta intenta la conexión con el clúster de Cassandra y en caso de éxito muestra información sobre la ip del clúster y su status.



Imagen 32: Herramienta de monitorización del clúster Cassandra

## 6.3. Monitorización del host

La herramienta intenta la conexión con el host del sistema y en caso de éxito muestra información sobre la ip del host y su status.

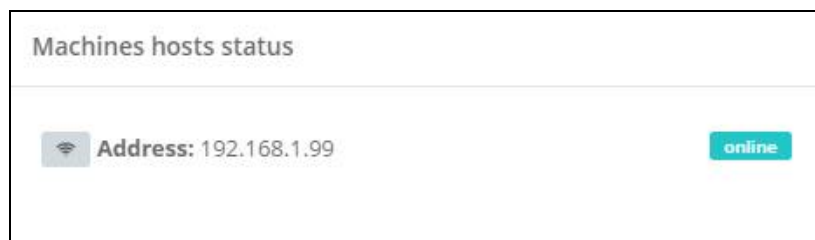
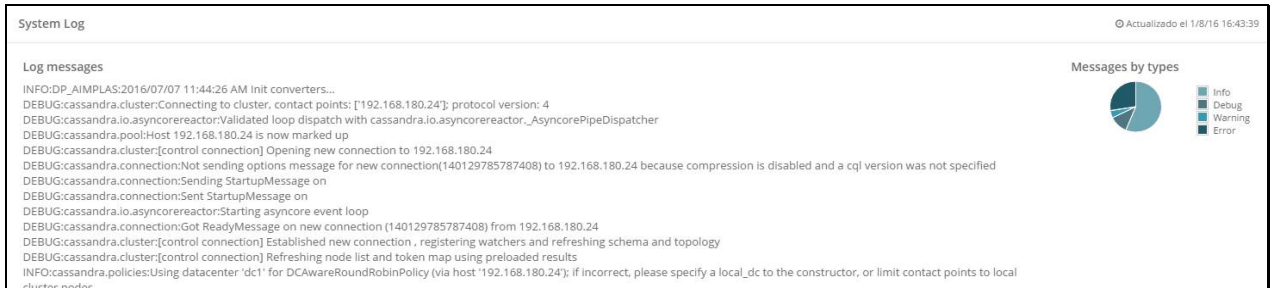


Imagen 33: Herramienta de monitorización del host

## 6.4. Monitorización de logs del sistema

La herramienta muestra los últimos logs del sistema y clasifica los mismos en función de su tipo (INFO, DEBUG, WARNING, ERROR)



System Log Actualizado el 1/8/16 16:43:39

Log messages

```

INFO:DP_AIMPLAS:2016/07/07 11:44:26 AM Init converters...
DEBUG:cassandra.cluster:Connecting to cluster, contact points: ['192.168.180.24']; protocol version: 4
DEBUG:cassandra.io.asyncoreactor:Validated loop dispatch with cassandra.io.asyncoreactor._AsyncorePipeDispatcher
DEBUG:cassandra.pool:Host 192.168.180.24 is now marked up
DEBUG:cassandra.cluster:[control connection] Opening new connection to 192.168.180.24
DEBUG:cassandra.connection:Not sending options message for new connection(140129785787408) to 192.168.180.24 because compression is disabled and a cqj version was not specified
DEBUG:cassandra.connection:Sending StartupMessage on
DEBUG:cassandra.connection:Sent StartupMessage on
DEBUG:cassandra.io.asyncoreactor:Starting asyncore event loop
DEBUG:cassandra.connection:Got ReadyMessage on new connection (140129785787408) from 192.168.180.24
DEBUG:cassandra.cluster:[control connection] Established new connection, registering watchers and refreshing schema and topology
DEBUG:cassandra.cluster:[control connection] Refreshing node list and token map using preloaded results
INFO:cassandra.policies:Using datacenter 'dc1' for DCAwareRoundRobinPolicy (via host '192.168.180.24'); if incorrect, please specify a local_dc to the constructor, or limit contact points to local cluster nodes
    
```

Messages by types

- Info
- Debug
- Warning
- Error

Imagen 34: Herramienta de monitorización de Logs del sistema

## 7. Validación del sistema de sensorización

La tarea de validación del sistema de sensorización estuvo centrada en el diseño de un **plan de pruebas** que permitiera la validación de la infraestructura y la interconexión de los elementos del sistema de sensorización, así como la herramienta de supervisión.

El desarrollo de un plan de pruebas nos permite identificar los chequeos necesarios para validar el sistema en su conjunto y la relación de sus subsistemas, verificando que las especificaciones y requisitos funcionales, no-funcionales y técnicos se cumplen. Por tanto, permitirá detectar fallos o incongruencias del sistema con los resultados esperados, así como una aproximación bastante fiable del comportamiento final del sistema en el entorno de implantación.

Las pruebas especificadas han permitido comprobar:

- Si el sistema de sensorización es capaz de recibir las señales obtenidas con los sensores en el mecanizado.
- Si las medidas son almacenadas de forma digital en un fichero de datos sin ningún tipo de procesamiento.
- Verificar si los datos son ingeridos por el almacén Big Data y servidos a través de los diferentes componentes interfaz.
- Si los tiempos de respuesta están dentro de los intervalos establecidos en las especificaciones del sistema de sensorización.
- Si el funcionamiento del sistema es correcto cuando está trabajando con grandes volúmenes de datos, simulando las cargas de trabajo esperadas.
- Si los diferentes componentes del sistema están operativos a lo largo del todo el proceso de captura de datos.

Las diversas pruebas a las que debe ser sometido el sistema deben ser realizadas tanto por el equipo de desarrolladores, como por los/as usuarios/as. Para ello, se debe establecer un orden de ejecución de estas para tener un procedimiento metódico que nos permita probar todos los puntos, en especial los críticos, del sistema. El orden de realización de pruebas será:

- **Pruebas Funcionales Unitarias:** Se comprobará de forma independiente el funcionamiento de cada subsistema identificado en el sistema mediante pruebas de disponibilidad y funcionalidad. Se debe comprobar que el sistema realiza con corrección todo aquello que realiza, y además, que realiza todo aquello que en la fase de análisis se identificó que debería llevar a cabo dicho subsistema.
- **Pruebas de Integración y Rendimiento:** El objetivo de estas pruebas es verificar el correcto ensamblaje entre los distintos componentes una vez que han sido probados unitariamente. El fin de estas pruebas es la de comprobar que interactúan correctamente a través de sus interfaces cubren la funcionalidad establecida y se ajustan a los requisitos no funcionales especificados en las verificaciones correspondientes. También en este caso se debe comprobar tanto en la regla de negocio como en la integridad de los datos.
- **Pruebas de Aceptación:** El objetivo de estas pruebas es validar que el sistema cumple con el funcionamiento esperado. El resultado final de estas pruebas es la aceptación del sistema durante la ejecución del proyecto piloto. Esta aceptación ha de ser desde el punto de vista de su funcionalidad y su rendimiento. Su realización es simbólica ya que vienen determinadas por el resultado de las pruebas anteriores.

## 7.1. Resultados del plan de pruebas

Específicamente, las actividades de revisión y pruebas del sistema de sensorización conllevaron la realización de dos tipos de pruebas: **pruebas funcionales** y **pruebas de integración**. Estas pruebas sirvieron para realizar una puesta a punto del sistema antes de su implantación para ejecución del proyecto piloto en AIMPLAS e INESCOP, dejando las pruebas de aceptación para la ejecución del mismo, disponibles en el entregable "E5.2 Análisis de los resultados del Proyecto Piloto".

### 7.1.1 Pruebas funcionales unitarias

Las pruebas funcionales se llevaron de forma independiente para cada uno de los componentes desarrollados que agruparan funcionalidades completas.

Este procedimiento tuvo como objetivo la detección de incidencias o errores y la aplicación de las acciones correctivas que fueran necesarias para obtener así una versión más robusta del prototipo, para la realización del piloto.

El resumen de resultados de las pruebas funcionales realizadas se presenta en la siguiente tabla:

Total de incidencias	Número total: 29	Resueltas: 39	Sin resolver: 0
<b><i>Incidencias por tipo</i></b>	<ul style="list-style-type: none"> <li>▪ Generales: 5 (12,8%)</li> <li>▪ Funcionamiento: 10 (25,6%)</li> <li>▪ Rendimiento: 3 (7,8%)</li> <li>▪ Despliegue: 6 (15,4%)</li> <li>▪ Integración: 13 (33,3%)</li> <li>▪ Otros: 2 (5,1%)</li> </ul>		
<b><i>Incidencias por módulo</i></b>	<ul style="list-style-type: none"> <li>▪ API Cassandra: 8 (20,55%)</li> <li>▪ Sink: 18 (46,15%)</li> <li>▪ Herramienta de Sensorización: 13 (33,3%)</li> </ul>		

La mayoría de los problemas estuvieron centrados en la integración y correcto funcionamiento de los componentes del sistema de sensorización. El conjunto de servicios REST utilizados para la coordinación de los elementos tienen la debilidad de la cantidad de datos que deben recorrerse para mapear los objetivos devueltos.

### 7.1.2 Pruebas de integración y rendimiento

Una vez concluida la realización de pruebas funcionales, se llevaron a cabo también pruebas de integración y rendimiento. El objetivo de dichas pruebas fue comprobar la viabilidad del entorno de ejecución del piloto y de la integración de cada uno de los componentes en un contexto de rendimiento óptimo.

Cabe destacar que para la realización de este tipo de pruebas se utilizaron las siguientes herramientas de análisis:

- **Monitorización de recursos utilizados:** *YSLOW*, que dispone de una interfaz en el navegador para conocer los recursos consumidos de los servicios web.
- **Generación de pruebas de estrés:** *HTTPERF*, para crear múltiples accesos al conjunto de servicios webs que emulen las funcionalidades del sistema.

Los resultados de la ejecución de dichas pruebas son los que se muestran en la siguiente tabla:

<b>Pruebas de Integración</b>	<ul style="list-style-type: none"><li>▪ <b>API Cassandra: 2 (14,3%)</b></li><li>▪ <b>Sink: 11 (78,6%)</b></li><li>▪ <b>Herramienta de Sensorización: 1 (7,1%)</b></li></ul>
<b>Pruebas de Rendimiento</b>	<ul style="list-style-type: none"><li>▪ API Cassandra: 3 (60%)</li><li>▪ Sink: 2 (40%)</li><li>▪ Herramienta de Sensorización: 0 (0%)</li></ul>

Como se puede observar, de nuevo los problemas estuvieron centrados en el despliegue de los componentes de forma aislada en la máquina virtual que se utilizará en el proyecto piloto. El ecosistema de servicios de cada componente, incluyendo la ejecución en modo de servicio de cada componente (java spring, python) ha supuesto un reto para el equipo.