UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial, Aeroespacial i Audiovisual de Terrassa

Diseño e implementación de un sistema de procesado de imagen aplicado a un proceso industrial

Documento:
Memoria

Autor/Autora:
Izan Luque Mateos-Aparicio

Director/Directora:
Delgado Prieto, Miguel

Titulación:
Grado en Ingeniería Electrónica Industrial y Automática

Convocatoria:
Primavera 2022

TRABAJO DE FINAL DE ESTUDIOS

Agradecimientos

Quiero agradecer y dedicar este trabajo en primer lugar a mi familia, por ayudarme y acompañarme a lo largo de esta etapa en la universidad. Agradecer el apoyo y el esfuerzo que han realizado para que pudiera llevar a cabo estos estudios de la manera más cómoda posible.

También quiero agradecer a todos los compañeros y profesores que han fomentado mi crecimiento tanto académico como personal.

Por último, agradecer al tutor del trabajo y todo el equipo que trabaja en el laboratorio por ayudarme y guiarme durante todo el proyecto, prestándome toda la ayuda necesaria durante su desarrollo.

Resumen

Este trabajo está enfocado en mostrar una arquitectura IoT en la que intervienen varios componentes como bases de datos, servicios en la nube, interfaces graficas para mostrar los datos y un procesado de imagen que compone una gran parte del proyecto realizado.

Se utilizará visión por computador mediante una cámara conectada a una Raspberry Pi la cual hará fotos a unos engranajes que irán pasando por una cinta. Una vez tomada esta imagen del engranaje se extraen características físicas de la pieza como puede ser el diámetro exterior e interior.

Una vez finalizado el procesado de imagen intervienen varias tecnologías como las bases de datos para almacenar los diámetros que se van adquiriendo a través de las imágenes.

Después de almacenar esta información se muestra mediante un dashboard que enseñará al usuario de una forma más grafica la información que se está generando.

Todo esto está corriendo en un sistema operativo Ubuntu en una máquina virtual instalada en la nube mediante los servicios de máquinas virtuales que ofrece Azure.

Abstract

This work is focused on showing an IoT architecture involving several components such as databases, cloud services, graphical interfaces to display data and image processing that makes up a large part of the project.

Computer vision will be used through a camera connected to a Raspberry Pi which will take pictures of gears that will go through a tape. Once this image of the gear is taken, physical characteristics of the piece are extracted, such as the outer and inner diameter.

Once the image processing is finished, several technologies are used, such as databases to store the diameters that are acquired through the images.

After storing this information, it is displayed through a dashboard that will show the user in a more graphic way the information that is being generated.

All this is running on an Ubuntu operating system in a virtual machine installed in the cloud through the virtual machine services offered by Azure.

Índex

Índice de tablas

Índice de figuras

1. Introducción

1.1 Objecto

Diseño e implementación de un sistema de procesado de imagen en el que se extraerán características físicas de una serie de engranajes para poder hacer una toma de decisiones a tiempo real durante el proceso industrial que se estará llevando a cabo. Lo que hace diferente a este trabajo comparándolo con trabajos similares es que no se busca clasificar piezas, sino que tiene como objetivo extraer características de las piezas para hacer un enrutado del proceso industrial.

Además, se pretende hacer una estructura de comunicación entre todos los dispositivos que intervienen en el proceso, para que la información fluya entre los diferentes dispositivos y programas.

1.2 Alcance

El alcance de mi trabajo vendrá delimitado por los siguientes puntos:

- Diseño del programa que detectará las características de la pieza. Se programará mediante Matlab un script que extraiga rasgos de la pieza y la dimensione.
- Diseño de la comunicación. Para conseguir que todo este conectado se necesita hacer una estructura de comunicación entre dispositivos con una herramienta llamada Node-RED
- Puesta en marcha en el laboratorio situado en el TR2 de la ESEIAAT. Para validar que todo se ha realizado correctamente se hará una prueba en la celda automatizada del laboratorio.
- Implementación de un sistema de visualización. Por pantalla el supuesto operario podría ver a tiempo real las características que se extraen, sus valores o los tiempos entre acciones.

A pesar de que el sistema está pensado para hacer un enrutado y toma de decisiones a partir de las características que se extraen, en este proyecto no se definirán estas decisiones. El trabajo se centrará en esta extracción de los datos y tanto la visualización como el almacenaje de estos datos.

En este trabajo no se implementarán las acciones anteriores y posteriores al tratamiento de imagen. El supuesto entorno industrial en el que se utiliza este procesamiento de imagen es en el de una fresadora que cambia de máquina herramienta según las dimensiones de la pieza.

Tampoco se cambiará el sistema físico donde esta implementada la captura de imagen del laboratorio. Como limite cabría la posibilidad de añadir una segunda cámara.

1.3 Requisitos

Para poder realizar este trabajo se necesitan una serie de requisitos:

- Emplear el programa Matlab con las toolboxes de procesamiento de imagen. Este programa será el que controle las características que se extraen de la pieza y las toolboxes son recursos que proporciona Matlab que facilitaran este proceso.
- Utilizar la herramienta de programación visual Node-RED. La necesitamos para poder controlar el flujo de información entre programas y dispositivos.
- El uso del dashboard Grafana. Para crear la interfaz gráfica donde se verá la información a tiempo real que se extrae en el proceso
- Tratar la información con la base de datos Influxdb. Necesitamos una base de datos potente para almacenar la información extraída del proceso industrial.
- Tanto la cámara como el dispositivo que la controla, en este caso la Raspberry Pi. Todo el material para la implementación física lo subministra el laboratorio de Schneider Electric que hay en la universidad.
- Emplear contenedores de Docker para instalar Node-RED,Grafana e Influxdb
- Utilizar servicios de computación en nube con Azure

1.4 Justificación

Actualmente estamos viviendo un cambio muy notorio en la industria, este fenómeno se identifica como industria 4.0, es decir, la cuarta revolución industrial. Lo que se está consiguiendo es que elementos que existen hace tiempo como la inteligencia artificial, bases de datos, IIoT o la robótica se puedan conectar y comunicar de una manera relativamente sencilla en comparación a hace unos años.

Esto hace que los sistemas que están conectados sean más eficientes y a la larga más económicos. Por eso creo necesario enfocar el tema de mi TFG hacia este sector.

La visión artificial y el uso de cámaras en procesos industriales, entra dentro de este gran grupo que hay en la industria 4.0. Cada vez es más común ver cámaras que están instaladas en un punto de un proceso industrial para supervisión, detección de errores y controles de calidad. Debido al gran potencial que tiene incluir cámaras en alguno de estos procesos he decidido basar mi trabajo en ello.

El proceso de mecanizado de superficies como el fresado es una técnica que se utiliza mucho en la industria para producir piezas con una forma específica. En estos procedimientos se suelen hacer unos diseños en 3D de la pieza y se pasan al programa para realizar operaciones como perforar o rebasar.

Pero qué pasaría si tuviéramos una cámara que ya te dijera las dimensiones de la pieza y a partir de esta información elija la herramienta idónea para la acción que se ha pedido con anterioridad.

Sobre este contexto voy a realizar mi trabajo, ya que, considero que es una solución muy válida que se puede implementar en un proceso industrial real. Además, la implementación de la cámara optimiza el proceso debido a que nos ahorraríamos que un operario fuera e introdujera las dimensiones manualmente, siendo así más seguro y exacto.

En el laboratorio de Schneider Electric que hay en el TR2 de la ESEIAAT trabajan para llevar casos como este a un escenario real, aprovechando lo que ya hay instalado. En mi caso, un alumno anterior a mi instaló una cámara en el laboratorio diseñando un sistema

de visión artificial. Yo voy a utilizar lo que en su momento se instaló allí para darle una utilidad totalmente diferente a la que implementó él.

2 Antecedentes y/o revisión del estado de la cuestión

La visión artificial es una tecnología muy utilizada hoy en día debido a los amplios beneficios que se obtienen con su implementación en una de las etapas productivas de la línea de producción.

Tener un control de tu producto con visión por computador hace que tengas una eficiencia mayor en el proceso productivo. Este aumento de la eficiencia viene dado por diferentes factores.

- Ahorro económico: Una vez instalada esta tecnología se obtiene un ahorro económico con el paso del tiempo. Además, los componentes que se utilizan para desarrollar estas funciones es cada vez más barato y asequible para las empresas
- Objetividad en la inspección: Los entornos controlados por visión artificial eliminan totalmente la subjetividad. Los ciclos de trabajo pasan a ser automáticos y repetitivos, y en todo momento el usuario determina y controla las variables que influyen en el proceso.
- Flexibilidad de los procesos productivos: La constante variabilidad de los productos que desarrollan las industrias hace que la visión por computador sea idónea debido a que esta es capaz de adaptarse a cambios con relativa facilidad.
- Disminución del error: Los humanos empiezan con una eficiencia muy buena, pero con el paso del tiempo la precisión en el trabajo disminuye y aumenta el error. La visión artificial tiene un error notablemente menor al humano ademas de que es constante y en todo momento lo podemos controlar.

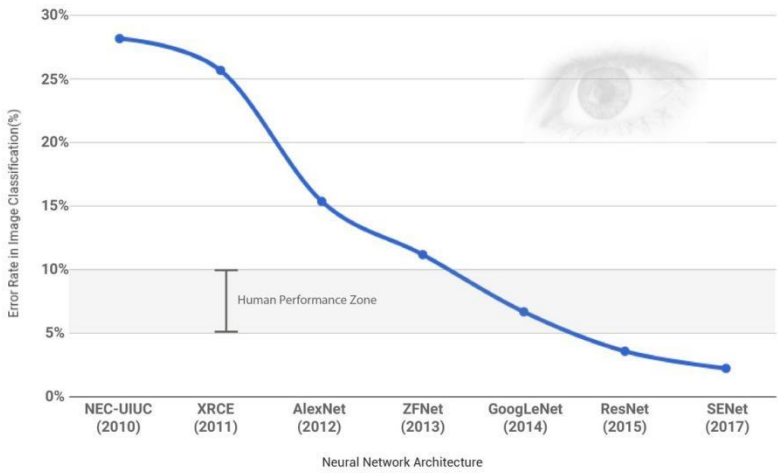


Ilustración 1: Error mediante visión artificial durante el tiempo

Por estos factores que enriquecen el producto final muchas empresas en diferentes sectores de la industria han decidido implementar la visión por computador en una o más de las etapas del proceso productivo.

2.1 Funcionalidades de visión artificial en la actualidad

A continuación, se detallan algunos de los sectores de la industria en los que se utiliza visión artificial.

2.1.1 Industria Farmacéutica

Kern Pharma es un laboratorio farmacéutico que se dedica a producir medicamentos genéricos y que ha implementado en la etapa final de su proceso de producción un control de calidad con visión artificial.

Esta etapa consiste en que una cámara con infrarrojos extraiga información del producto, en este caso el volumen de las proporciones que hay en el interior de las capsulas o comprimidos, y determine si alguna de ellas no tiene la dosis correcta. Con esto podemos encontrar y desechar los productos que no cumplen los los parámetros que deberían.

Ilustración 2: Uso de visión artificial para control de calidad en comprimidos

2.1.2 Robótica

La robótica es una tecnología que evoluciona de manera exponencial y que muchas industrias están empezando a implementar en su cadena de producción. Para que el desempeño de los robots sea más eficiente y flexible se han añadido cámaras en sus entornos de trabajo para múltiples funcionalidades.

Una de ellas es permitirle al robot identificar y definir la posición de cualquier elemento para recolectarlo y trasladarlo a su destino. Gracias a la visión artificial se hace un mapeado de la zona, incluso añadiendo más de una cámara, para saber la posición en 3D. Esta actividad tiene el nombre de “Pick & Place” y es una actividad que representa el 40% de las tareas manuales que se llevan a cabo en una fabrica

Ilustración 3: Visión por computador aplicado a la robótica

2.1.3 Industria Agroalimentaria

La primera etapa de la cadena agroalimentaria es la producción de semillas y granos. Las empresas que realizan esta función requieren de complejos desarrollos que conllevan grandes inversiones en I+D que algunas empresas no son capaces de cubrir.

Para solucionar este problema se aplica visión por computador, en el que una cámara extrae el peso de las semillas que van pasando por una cinta ademas de hacer un conteo de estas. Con ello, logramos una mayor precisión, mejorar los tiempos de ejecución y reducir los costes.

Ilustración 4: Visión por computador aplicado a la industria agroalimentaria

2.2 Visión Artificial + IoT

Todos los sistemas de visión artificial están compuestos por dos bloques. Uno se compone de la tecnología más relacionada con la sensórica, es decir, las cámaras que vamos a utilizar para la aplicación en cuestión y las especificaciones de esta. Y otra es el procesado de datos que consiste en el conjunto de técnicas que se aplican una vez que ya tenemos la imagen para extraer la información que nos interesa dependiendo de la aplicación que le queremos dar.

Estas dos etapas son parte de este trabajo, pero quiero darle más importancia.

La sensórica es una parte fundamental a la hora de cubrir las necesidades que requiere nuestro proceso industrial. Se debe determinar el número de cámaras que se querrá utilizar, la resolución o si la luz que se aplicará puede dañar la cámara.

Otro aspecto para tener en cuenta es el procesado de imagen. Dependiendo del producto y la información que queremos extraer se aplicaran diferentes técnicas de procesado de datos una vez tengamos la imagen.

Para acercar la visión artificial a sectores industriales de manera masiva debemos acercarnos a pymes industriales. Las pymes son pequeñas empresas, que, en este caso en específico, te pueden ofrecer un sistema integral, es decir, no solo compras las funciones que hemos comentado anteriormente (sensórica y procesado de datos, ya que no es algo genérico, depende de tu aplicación), sino que además compras todo el paquete de infraestructura; sensor, comunicación, repositorio, etc. Partiendo de esto creo interesante que el entorno principal del proyecto sea diseñar e implementar una estructura IoT específica para aplicaciones de visión artificial.

El alcance de esta arquitectura vendrá dado por las funcionalidades que me pueda aportar el laboratorio de automatización de Schneider Electric de la ESEIAAT, donde se ha realizado el trabajo.

2.3 Laboratorio

El entorno de trabajo donde se van a realizar las pruebas y se va a desarrollar toda la parte práctica del trabajo es en el laboratorio de Schneider Electric situado en el TR2 de la Universidad Politécnica de Terrassa (ESEIAAT).

El laboratorio dispone, a rasgos generales, de un conjunto de sensores y actuadores controlados por PLC que mueven unas cintas para desplazar una serie de bandejas que contienen un producto a modo de simular un proceso de producción parecido al que hay en una fábrica.

Ilustración 5: Imagen general del laboratorio

Los recorridos que se pueden tomar son diferentes dependiendo de las estaciones en las que se quieran parar las bandejas. Dependiendo de la información que se recoja en cada una de las estaciones se puede redireccionar la bandeja hasta otro destino al esperado en un inicio.

Ilustración 6: Cintas del laboratorio donde se ha realizado el trabajo

Una de estas estaciones en las que se recoge información del producto podría estar enfocada a visión artificial mediante una cámara que realiza fotos a un producto que va pasando por la cinta y eligiendo el camino que debe tomar el producto dependiendo de una característica predefinida del producto.

Ilustración 7: Estación de visión artificial

Esta estación dispone de una Raspberry Pi 4, una placa de microordenador que nos servirá para añadirle la extensión de la cámara y poder tomar las imágenes y almacenarlas. En general necesitaremos varios componentes para que la Raspberry empiece a funcionar. Lo primero es el cargador para enchufarla a la corriente, seguido de un cable HDMI para visualizar la imagen en una pantalla. Por último, necesitamos una microSD para instalar el sistema operativo de la Raspberry.

PROCESADOR	Quad Core Cortex A-72
FRECUENCIA DE RELOJ	1,5 GHz
RAM	2, 4, 8 GB LPDDR4
ALIMENTACIÓN	5V/3A USB tipo C
HDMI	2 puertos

Tabla 1: Características Raspberry Pi 4

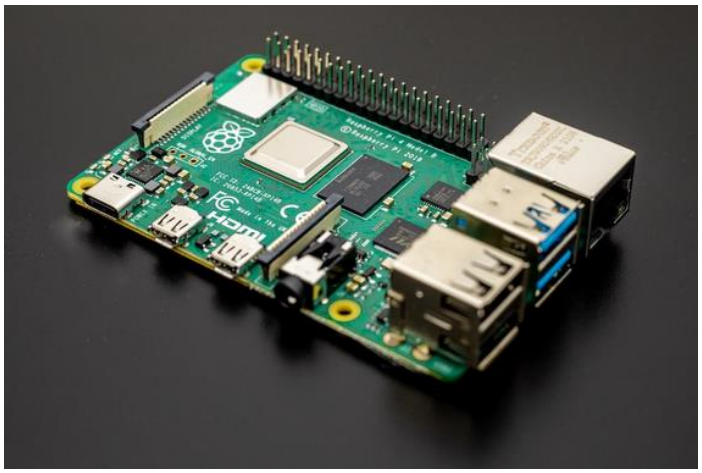


Ilustración 8: Raspberry Pi 4

A esta Raspberry se le añade la cámara de la propia empresa Raspberry Pi llamada “Module Camera V2”. Que sea de la misma compañía nos asegura la compatibilidad entre los dos dispositivos. Esta cámara irá directamente conectada a la Raspberry Pi 4 mediante el puerto CSI. Para que la cámara este situada en un punto fijo estará junto a la Raspberry dentro de una funda.

NUMERO DE CANALES	1
INTERFAZ DE BUS ADMITIDA	CSI
MÁXIMA FRECUENCIA DE IMAGEN CAPTURA	30 fps
DIMENSIONES	23,86 x 25 x 9 mm

Tabla 2: Características de Module Camera V2

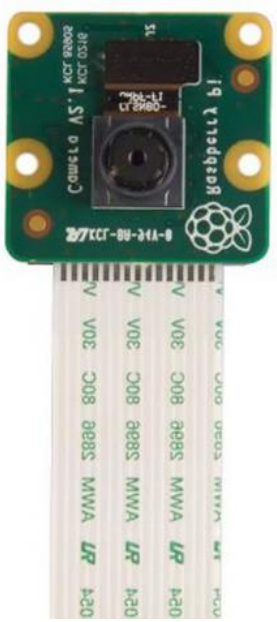


Ilustración 9: Module Camera V2

3 Diseño de la solución

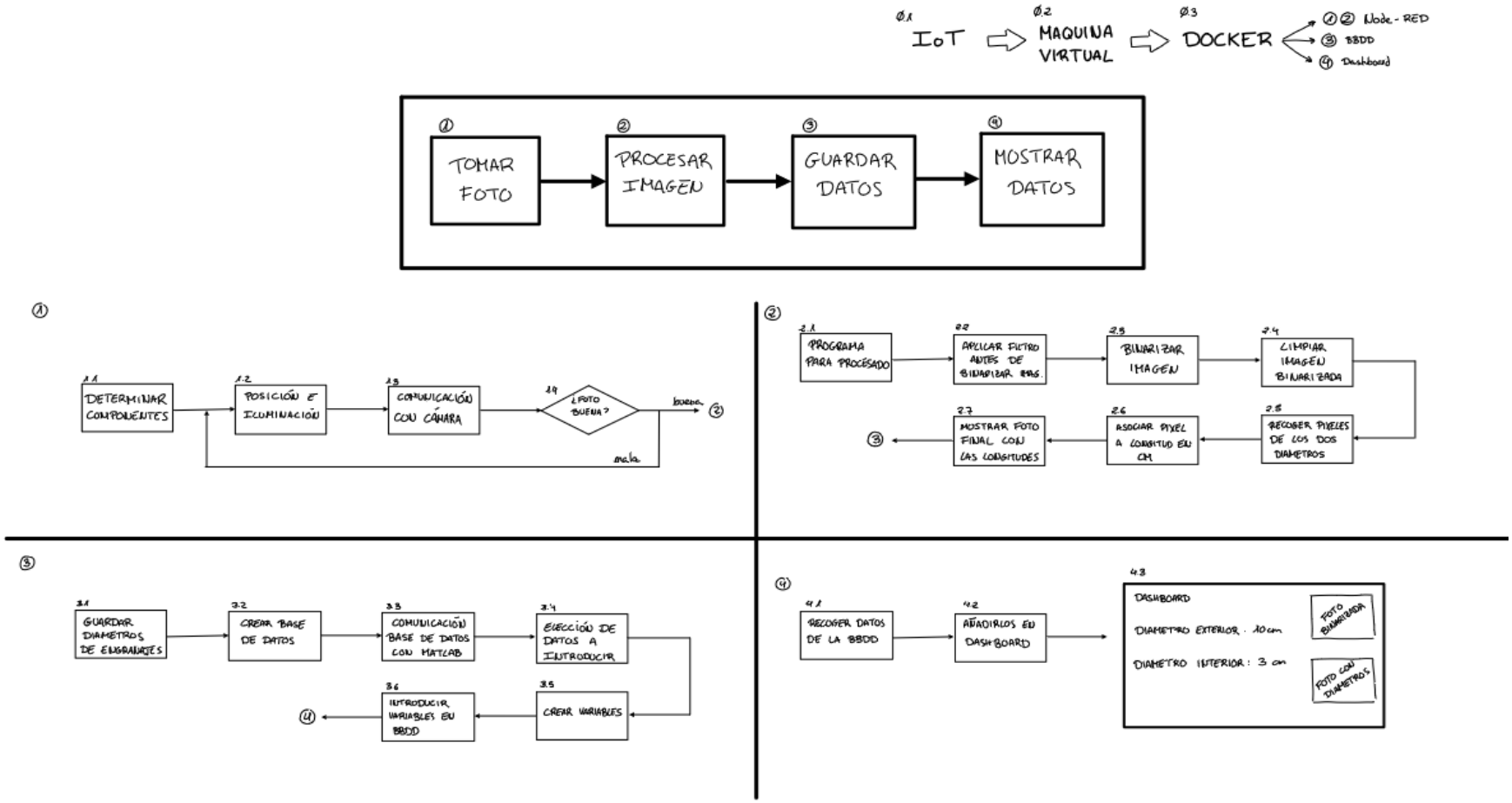


Ilustración 10: Diagrama de bloques de la solución planteada

ARQUITECTURA IoT

0.1 IoT:

En esta etapa se pretende elegir el servicio cloud que sostiene nuestro proyecto. Hay varias posibilidades, pero las dos plataformas más consistentes son Azure y AWS.

0.2 MÁQUINA VIRTUAL:

Se creará una máquina virtual donde se instalarán todos los programas para realizar el proyecto. Se debe elegir las especificaciones de la máquina virtual como la ubicación donde se instalará, RAM, CPU, etc.

0.3 DOCKER:

Se utilizarán también contenedores donde se instalarán en cada uno los diferentes programas usados como Influxdb, Grafana o Node-RED.

BLOQUE 1: TOMAR FOTO

1.1. DETERMINAR COMPONENTES:

Elegir el hardware del proyecto. En este trabajo se utiliza una estación que ya tiene el hardware elegido, pero se tiene que estudiar para ver si es idóneo para la funcionalidad que le queremos dar. Tal vez la cámara no pueda alcanzar una resolución necesaria o la memoria de la raspberry es insuficiente. Se determinará modelo de raspberry, modelo de la cámara, numero de cámaras, cables para las conexiones, etc.

1.2. POSICIÓN E ILUMINACIÓN:

Determinar la iluminación que daremos al entorno en el que se hace la foto. Decidir si incluimos al sistema más de un foco desde diferentes ángulos. También si la posición de la cámara es fija o se puede desplazar.

1.3. COMUNICACIÓN CON CÁMARA:

Establecer la vía de comunicación que tendremos con la cámara, además de especificar donde se guardará la imagen para decidir si es buena en etapas posteriores.

1.4. TOMAR FOTO:

Una vez establecida la comunicación con la cámara en esta etapa tomaremos la foto y decidiremos si es buena o mala. Si la foto es buena podremos empezar con el procesado de esta. Si es mala volveremos a etapas anteriores donde habrá que cambiar parámetros de los entornos de comunicación o la iluminación.

BLOQUE 2: PROCESAR IMAGEN

2.1 PROGRAMA PARA PROCESADO:

Elegir el programa para hacer el procesado de la imagen.

2.2 APLICAR FILTRO ANTES DE BINARIZAR IMAGEN:

Antes de binarizar la imagen se aplicarán unos filtros para que al binarizar la imagen sea todo lo nítida posible. Escala de grises,

2.3 BINARIZAR IMAGEN:

En esta etapa binarizamos la imagen previamente tratada.

2.4 LIMPIAR IMAGEN BINARIZADA:

Posteriormente a la binarización se aplican también técnicas para limpiar la imagen

2.5 RECOGER PIXELES DE LOS DOS DIAMETROS:

Para saber los diámetros de los engranajes es necesario obtener los píxeles y asociarlo a la longitud

2.6 ASOCIAR PIXEL A LONGITUD EN CM:

Se establece la relación entre píxel y cm

2.7 MOSTRAR FOTO FINAL CON LAS LONGITUDES:

Una vez tenemos todos los datos creamos una imagen con la foto original y los diámetros obtenidos durante toda la etapa de procesado.

BLOQUE 3: GUARDAR DATOS

3.1 GUARDAR DIÁMETROS:

Guardar los diámetros y decidir si se quieren recoger otros datos que añadir

3.2 CREAR BASE DE DATOS:

Elegir base de datos; influxdb, mongo, MySQL, etc. Ver especificaciones de cada una y compatibilidad de cada una de ellas con los diferentes programas que se van a utilizar en el proyecto

3.3 COMUNICACIÓN BASE DE DATOS CON MATLAB:

Establecer la comunicación entre la base de datos y el software que procesa la imagen y adquiere los diámetros. Desde node-RED o desde Matlab con una librería.

3.4 ELECCIÓN DE DATOS A INTRODUCIR:

Elegir los datos que se han adquirido al procesar la imagen y elegir cuales se guardaran en la base de datos.

3.5 CREAR VARIABLES:

Preparar las variables que queremos introducir en la base de datos de la forma adecuada para la base de datos

3.6 INTRODUCIR VARIABLES A BBDD:

Ejecutar los comandos para añadir los datos previamente preparados a la base de datos.

BLOQUE 4: MOSTRAR DATOS

4.1 RECOGER DATOS DE LA BBDD:

Enlazar la plataforma donde se mostrará el dashboard con la base de datos donde tenemos toda la información que queremos enseñar.

4.2 AÑADIRLOS EN DASHBOARD:

Crear la query para añadir la información que hemos recopilado en la base de datos. La idea principal es ir mostrando el último diámetro que se ha añadido para mostrar siempre la pieza que acaba de ser procesada.

4.3 DASHBOARD

En este bloque se trata de esquematizar la idea principal de cómo se verá el dashboard con la información adquirida. Los dos diámetros y las dos fotos del engranaje, una con las medidas y otra binarizada

4 Desarrollo de la solución o las soluciones escogidas

4.1 Diagrama de flujo de Node-RED para obtener la imagen

Node-RED es una potente herramienta capaz de comunicar hardware y servicios de manera ágil y muy sencilla. La principal utilidad de Node-RED es que da al usuario la posibilidad de hacer uso de complejas tecnologías de la manera más sencilla.

Este software será útil para controlar el flujo de la información, en el caso de este proyecto, la imagen generada por la cámara integrada en la raspberry Pi.

Lo primero que se realizó en este proyecto es el esquema de Node-RED para verificar que la cámara capturaba bien la imagen y el estado en el que se veía. Para ello se ejecutó el siguiente flujo:

Ilustración 11: Diagrama de flujo en Node-RED

Podemos ver que la distribución del flujo de información está basada en 5 bloques.

Primero tenemos un nodo HTTP IN, en nuestro caso con el método GET, que crea un end-point para crear un servicio web y poder aceptar solicitudes de un cliente bajo la URL de <IP ADRESS>:1880/test2 o <localhost>:1880/test2.

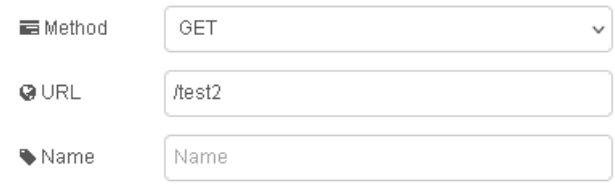


Ilustración 12: Configuración nodo HTTP IN

Una vez configurado este primer nodo lo uní a “camerapi-takephoto node”, que se encarga de hacer la foto con la cámara escogida y guardar la imagen en el directorio /home/pi/Pictures/. Desde este bloque también definimos algunas características de la foto que tomamos, como la resolución que está fijada en 3280x2464 o el formato de la imagen que es JPG.

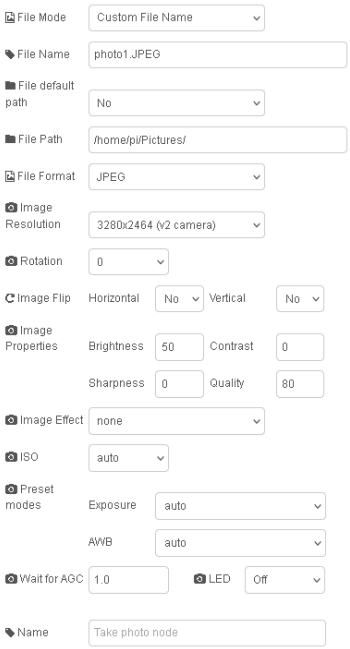


Ilustración 13: Configuración nodo cámara Raspberry Pi

En este punto del flujo ya se habrá guardado la imagen en nuestra ruta indicada. El siguiente paso fue capturar esa imagen y para ello se necesita un nodo que lea un fichero "read node" el cual se le debe especificar la ruta con el nombre del fichero y el formato en que saldrá la imagen del bloque. En este caso se ha elegido "A single Buffer object" en el que se presenta el archivo como una cadena de bytes.

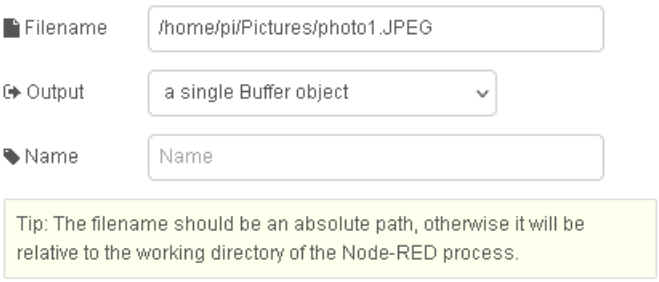


Ilustración 14: Configuración nodo read node

A continuación, tenemos el nodo "change", que se encargará de hacer el encabezado de la respuesta. En este caso va a ser una respuesta string.

Ilustración 15: Configuración nodo change

Por último, el nodo HTTP RESPONSE que se encarga de enviar la respuesta con la información de la imagen.

4.2 Posición e iluminación

Una vez la comunicación de la cámara está operativa y vemos que la imagen llega correctamente se establece la iluminación y la posición en la que estará ubicada la cámara con la Raspberry pi.

El primer día que accedí al laboratorio la Raspberry estaba sin colocar en un sitio fijo, por lo tanto, empecé haciendo pruebas situando la Raspberry con la cámara sujetada por una caja e iluminando con la linterna del móvil. Estas primeras fotografías fueron desechadas debido a que algunas de estas estaban borrosas debido a que el posicionamiento no era fijo y se movía con facilidad. También influyó la iluminación ya que se veían sombras que dificultaban el procesado posterior.

Para arreglar esto situamos la Raspberry con la cámara en una estación que estaba acondicionado para ese uso, con una barra de aluminio y una funda del modelo de la Raspberry pi utilizada. La iluminación se solventó añadiendo un foco circular para evitar cualquier tipo de sombra.

Ilustración 16: Foco utilizado para iluminación

4.3 Script Matlab

4.3.1 Obtención de la imagen

Cuando solventé todos los problemas relacionados con la imagen ya podía empezar a procesar la imagen para extraer la información que interesa, en el caso de este proyecto el diámetro de los engranajes.

Pero antes de empezar el procesado tenemos que disponer, en Matlab, de la imagen que se ha sacado con la cámara de Raspberry. Una vez enviada la petición de hacer una foto mediante la cámara conectada a la Raspberry esta devuelve la imagen en Matlab como un array de bytes codificado en base64[1].

A partir de ahí lo que se hace es deshacer la codificación mediante métodos de java que están integrados en Matlab para obtener "jImg". Como podemos ver en la imagen extraemos el ancho y alto de la imagen que se guardan en "h" y "w". Por último, la variable "p" esta codificada en formato uint8[2] para poder mostrarlo en Matlab. Finalmente la variable "Engranaje" será la foto final donde se concatena con la función cat() las tres capas de la variable "p", traspuestas y redimensionadas con las variables "h" y "w" creadas anteriormente.

Con estas líneas de código obtenemos la imagen en el formato que necesitamos:

```
close all
import matlab.net.*
import matlab.net.http.*
url ='http://10.7.0.25:20000/test1';
jImg = javax.imageio.ImageIO.read(java.io.ByteArrayInputStream ...
       (base64decode(webread(url))));
h = jImg.getHeight;
w = jImg.getWidth;
p = reshape(typecast(jImg.getData.getDataStorage, 'uint8'), [3,w,h]);
Engranaje = cat(3, ...
        transpose(reshape(p(3,:,:), [w,h])), ...
        transpose(reshape(p(2,:,:), [w,h])), ...
        transpose(reshape(p(1,:,:), [w,h])));
```

Ilustración 17: Código para tratar imagen de Node-RED

[1] Base64 es un esquema de codificación de binario a texto. Representa datos binarios en un formato de cadena ASCII.

[2] uint8 es un entero sin signo de 8 bits con un rango de 0 a 255 y es el formato habitual en el que se muestran las imágenes en Matlab.

4.3.2 Procesado de imagen

4.3.2.1 Diagrama de flujo del script

Al ejecutar estas líneas de programación ya tendremos la imagen en Matlab. Una vez la tenemos ahora si que podemos empezar a procesarla. Para ello, he realizado un diagrama de flujo que representa todos los pasos que se seguirán a la hora de programar.

Ilustración 18: Diagrama de flujo para código de tratamiento de imagen

4.3.2.2 Tratamiento de la imagen para obtener foto final

Primero se ha recortado la imagen para obtener la región que nos interesa, ya que, en la original sale parte del suelo del laboratorio. Solo queremos tomar lo que hay en la bandeja que se pone en la cinta.

```
recortada = imcrop(Engranaje,[150 84 2980 1868]);
```

La variable "Engranaje" es la foto original y [150 84] es la posición de la esquina superior izquierda. Por otro lado [2980 1868] es la posición de la esquina inferior derecha menos la posición mencionada anteriormente. Una vez ejecutada esta línea tenemos la imagen recortada.

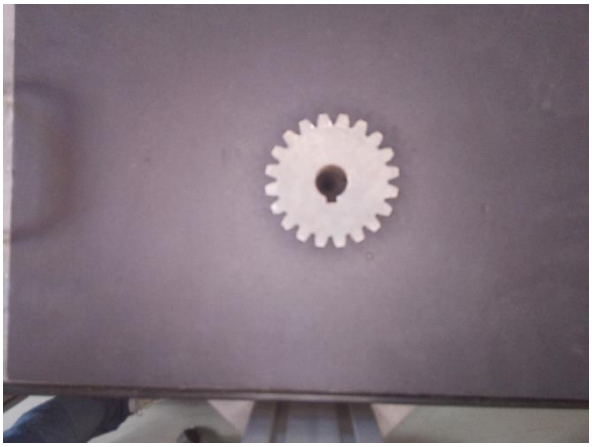


Ilustración 19: Imagen original

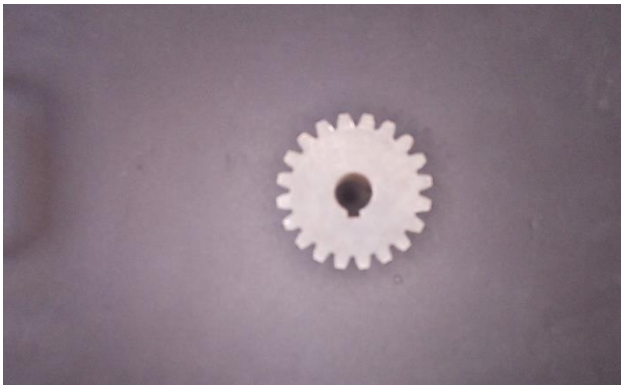


Ilustración 20: Imagen recortada

Luego pasé la imagen a escala de grises. Es decir, pasamos de una matriz de tres variables (RGB) a una de dos variables. El rango de variables es 0 ~ 255.

```
imgrey = rgb2gray(recortada);
```

Realizados estos pasos tenemos la imagen lista para binarizar la imagen. Pasamos del rango 0 ~ 255 a 0 o 255. Para ello tenemos que determinar un threshold que he ido probando hasta determinar el idóneo para la iluminación establecida.

```
imblack = imbinarize(imgrey,0.65);
```

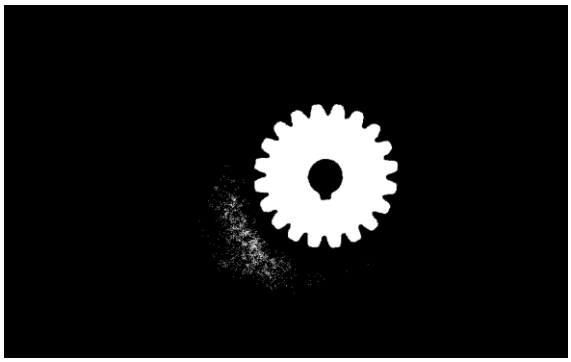


Ilustración 21: Imagen binarizada

Como podemos observar he binarizado la imagen, pero hay pequeños puntos blancos que se ha podido originar por la luz que hay en ese momento o sobras por destellos de luz desde otras posiciones. Hay que especificar que lo idóneo seria tomar la foto del engranaje dentro de una caja negra, para que la luz sea siempre la misma y no interfiera en el proceso. Para prevenir este tipo de problemas se ha utilizado un filtro que elimina objetos pequeños. Se utiliza la función de bwareaopen() que elimina todos los objetos con 10000 píxeles.

```
BW = bwareaopen(imblack,10000);
```

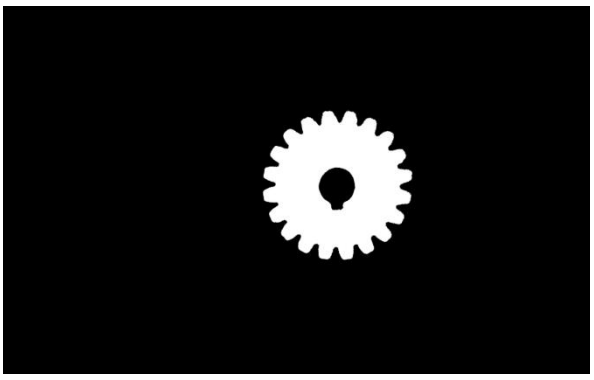


Ilustración 22: Imagen filtrada para eliminar imperfecciones

Una vez llegados a este punto tenemos en la imagen un solo objeto en blanco. Lo siguiente que se hizo fue detectar los objetos que hay en la imagen con la ayuda de la función regionprops.

Lo que hace esta función es recoger la posición del objeto además de recoger la altura y el ancho del objeto. El tratamiento realizado anteriormente es importante, ya que, si no hubiéramos limpiado bien la imagen y aparecieran más objetos esta función recogería los datos de todos ellos

```
bboxes = regionprops (BW, 'BoundingBox')
```

Field	Value
BoundingBox	[1.1465e+03,554.5000,924,928]

Ilustración 23: Posición, ancho y alto del objeto encontrado

En la ilustración 18 podemos ver que los dos primeros valores son la posición del objeto encontrado. El tercer componente del array con el valor 924 es la altura. El cuarto componente con valor 928 es el ancho del engranaje.

Tanto la altura como el ancho que nos devuelve la función regionprops viene dada en pixeles por lo que en este punto hay que hacer la conversión entre pixeles y centímetros.

4.3.2.3 Obtención diámetro exterior

Para poder hacer la conversión de pixeles a centímetros necesitamos obtener los pixeles de un diámetro conocido.

Se comenzó haciendo la prueba con el primer engranaje, ya que, es el más grande y obtendremos un error menor. Lo primero que se realizó fue repetir el proceso explicado hasta ahora para obtener una gran variedad de diámetros asociados a la altura y ancho expresado en pixeles. Cuando disponía de varios diámetros se hizo una media y obtuve que para un diámetro de 10 cm devolvía un valor de 932 píxeles.

Sabiendo la relación entre píxel y centímetros podemos hacer el calculo mediante un factor de conversión.

```
Diametro_ext_cm = (10*CurrBB(3))/932;
```

Y para que se más visual añadimos un cuadro con el valor de la medida recogida pasada a centímetros. Para ello hago uso de las funciones rectangle() a la cual le introduje los valores obtenidos mediante la función regioprops (CurrBB(1), CurrBB(2), CurrBB(3), CurrBB(4)). “EdgeColor” es el color de la línea del rectángulo, en este caso azul y en ancho de la linea del rectángulo que le he dado el valor de 2

```
rectangle('Position' , [CurrBB(1),CurrBB(2),CurrBB(3),CurrBB(4)],
                'EdgeColor','b','LineWidth', 2);
```

Después de añadir el rectángulo decidí añadir el valor del diámetro calculado usando la función text(). Le introduje la posición en la que quería que saliera el texto, el texto que quería enseña, el color y el tamaño de letra.

```
text(CurrBB(1),CurrBB(2)-70,strcat('Diametro: ', ...
num2str(Diametro_ext_cm)),'Color','blue','FontSize',18);
```

La posición viene dada por las variables CurrBB(1) y CurrBB(2). A CurrBB(2) se le resta 2 para que no aparezca en medio de la línea del rectángulo. La función strcat() se utiliza para juntar dos strings. Uno de esos strings es el diámetro que se utiliza num2str() para pasar de una variable numérica a string.

Realizado todo esto nos queda la imagen del engranaje con el diámetro exterior.

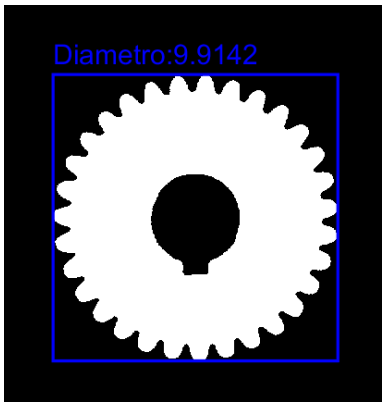


Ilustración 24: Engranaje con diámetro exterior

4.3.2.4 Obtención diámetro interior

Ahora realizaremos el mismo proceso para el interior. Pero para ello vamos a invertir la imagen binarizada. Los pixeles en negro pasaran a blanco y viceversa. Cuando ejecutamos la función regionprops veremos que aparecen las medidas de dos objetos, el fondo blanco y la esfera interior del engranaje.

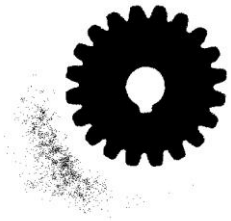


Ilustración 25: Imagen binarizada invertida

Fields	BoundingBox
1	[0.5000,0.5000,2981,1869]
2	[1.4645e+03,877.5000,289,327]

Ilustración 26: Posición y medidas de los dos objetos encontrados

A nosotros nos interesa el objeto más pequeño, por lo tanto, eliminamos el objeto más grande y nos quedamos las medidas del más pequeño. Sabiendo los dos diámetros volvemos a repetir todo el procesado que se realizó para obtener el diámetro exterior.

Cuando disponemos de los dos diámetros hacemos una imagen final donde se vean los dos y la región que pertenece a ese valor.

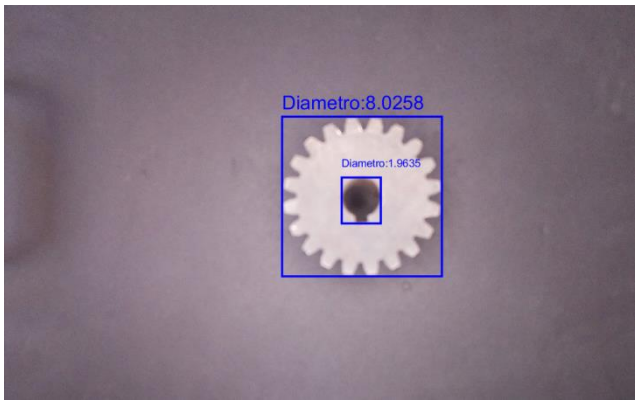


Ilustración 27: Imagen con medidas calculadas

Una vez adquiridos los diámetros se clasificará el tipo de engranaje del que he tomado esta información. Los engranajes que se están utilizando tienen diferentes diámetros interiores y exteriores, por lo que, me parece útil e interesante mostrar al usuario el tipo de engranaje que tiene delante y realizar un conteo.

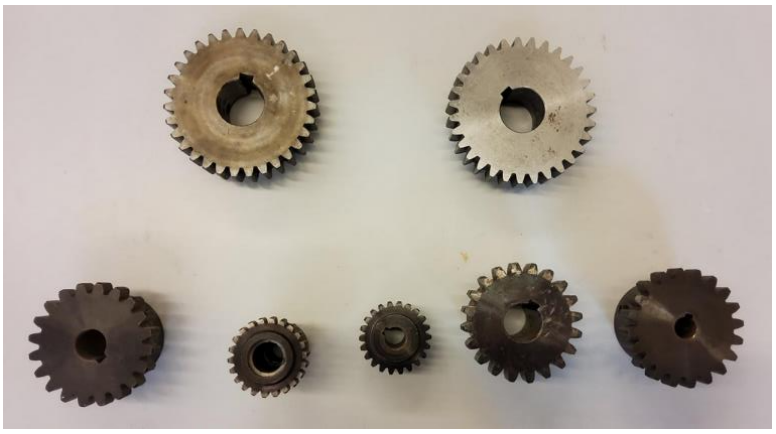


Ilustración 28: Engranajes utilizados en procesado de imagen

Para ello se ha establecido unos rangos que delimitan a que tipo de engranaje pertenece. Estos rangos se han reflejado en la siguiente tabla:

Diámetro Exterior	Diámetro Interior	Tipo
10.2> *Diametro_ext_cm* >9.7	3.2> *Diametro_int_cm* >2.7	1
8.1> *Diametro_ext_cm* >7.4	2.4> *Diametro_int_cm* >1.8	2
5.3> *Diametro_ext_cm* >4.5	2> *Diametro_int_cm* >1.5	3
5.3> *Diametro_ext_cm* >4.5	1.8> *Diametro_int_cm* >1.3	4
8.1> *Diametro_ext_cm* >7.4	3> *Diametro_int_cm* >2.4	5
8.1> *Diametro_ext_cm* >7.4	1.9> *Diametro_int_cm* >1.3	6

Tabla 3: Rangos de medidas de los engranajes con su tipo asociado

Durante el desarrollo de la parte de procesado de imagen se han tomado gran multitud de fotografías de todos los tipos de engranajes que disponemos. Esto ha hecho que comparando las medidas reales de los engranajes con los diámetros que se han ido obteniendo podemos llegar a un error aproximado.

Los errores que se reflejan en el dashboard son los que obtenemos haciendo las fotos con una iluminación y un threshold predefinido. Una variación de estos componentes hace que la situación que tengo en el proyecto cambie.

El error se ha calculado aplicando la siguiente formula y haciendo una media de los diferentes errores obtenidos.

$$Error = \frac{Valor\ medido - Valor\ exacto}{Valor\ exacto}$$

Según el tipo de engranaje se obtienen los siguientes errores:

Tipo de engranaje	*Error [%]*
1	0,9
2	3,7
3	8,3
4	8,3
5	3,7
6	3,7

Tabla 4: Error asociado a los tipos de engranajes

Como podemos observar los engranajes más grandes obtienen un error más pequeño comparado con los engranajes con diámetros más pequeños. Esto puede venir provocado por la resolución de la cámara. Si obtuviéramos una cámara con mejor resolución la toma de las medidas sería más precisa.

4.4 Azure

4.4.1 Máquina Virtual

Llegados a este punto se ha extraído la información que queremos del engranaje y pasamos al entorno cloud. En esta etapa queremos contratar un servicio cloud con Azure. Estos servicios tienen un coste monetario dependiendo de las funciones que utilices o de las características que selecciones. La UPC proporciona a los estudiantes 100€ de saldo para probar varios de sus servicios a los que les daré uso.

Con la cuenta enlazada de la UPC empezamos por crear lo que va a ser nuestra máquina virtual. En esta máquina virtual instalaremos todos los programas que se utilizan en este trabajo de final de grado. Para comenzar primero debemos seleccionar las características de esta, que se ajustan a las necesidades de este proyecto.

Como el uso que se le va a dar no requiere de grandes prestaciones se va a escoger una máquina virtual con un precio bajo en comparación al catálogo que ofrece Azure. Como al principio vamos a instalar una cantidad considerable de archivos se ha elegido una máquina virtual con un coste un poco más elevado a la máquina que utilizaremos una vez terminado el proceso.

La primera máquina virtual tiene las siguientes características:

Sistema Operativo	***vCPU***	***RAM***	***Ubicación***
Linux(Ubuntu 18.04)	2	4 GB	East Asia

Tabla 5: Características máquina virtual inicial

Para ahorrar gastos y ya que no es necesario, la máquina virtual pasa a tener unas especificaciones más simples cuando se acabe con las instalaciones. Estas son sus características:

Sistema Operativo	***vCPU***	***RAM***	***Ubicación***
Linux(Ubuntu 18.04)	1	1 GB	East Asia

Tabla 6: Características máquina virtual una vez instalado el software necesario

4.4.2 Configuración de puertos

Para garantizar una seguridad y un control de los accesos de los usuarios el siguiente paso es abrir los puertos. Debemos seleccionar los puertos correspondientes para crear puntos de conexión que servirán para interactuar con los programas que utilicemos.

En la siguiente tabla se muestra el número de puertos con el protocolo utilizado:

Nombre del puerto	***Puerto***	***Protocolo***
SSH	22	TCP
Node-RED	1880	TCP
Grafana	3000	TCP
Influxdb	8086	TCP

Tabla 7: Puertos habilitados en Azure

4.5 Terminal

4.5.1 Conexión con maquina virtual

Una vez realizada esta configuración pasamos a trabajar con la terminal donde realizaremos la conexión con la maquina virtual mediante SSH.

SSH es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas.

Cuando se ha creado la máquina virtual en Azure hemos obtenido también una clave PEM que nos servirá para acceder e iniciar sesión en nuestra máquina virtual.

Un archivo PEM es un archivo de texto que contiene uno o más elementos en la codificación ASCII Base64, cada uno con encabezados y pies de página de texto sin formato. Un solo archivo PEM podría contener un certificado de entidad final, una clave privada o varios certificados que forman una cadena de confianza completa.

En la siguiente imagen podemos ver el comando que pasamos por la terminal para realizar la conexión con el servidor. C:\Users\izanl\.ssh\ es la ruta donde esta guardada la clave PEM con el nombre TestClave_key. Lo que le sigue es el nombre del usuario y la ip publica que se ha generado al iniciar la máquina virtual.

```
PS C:\Users\izanl> ssh -i C:\Users\izanl\.ssh\TestClave_key.pem LuqueIzan@20.24.192.227
```

Ilustración 29: Comando conexión SSH

4.5.2 Docker

Cuando la conexión sea satisfactoria pasaremos a instalar todo el software necesario para realizar este proyecto. Primero instalaremos docker y docker-compose.

Docker es un sistema operativo para contenedores. Los contenedores son unidades estandarizadas que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución

```
LuqueIzan@TFG:~$ sudo apt-get install docker-ce docker-ce-cli containerd.io
```

Ilustración 30: Comando instalación docker

4.5.3 Creación de imagenes de Grafana y Node-RED

Para disponer de las imágenes con el software adecuado con las versiones idóneas para no generar errores entre programas se instala un directorio que me ha proporcionado el laboratorio. Para transferir archivos entre mi ordenador personal y la máquina virtual se hace uso del protocolo SCP.

El protocolo SCP es un protocolo para sistemas informáticos, que garantiza la transferencia segura de datos entre un equipo local y un equipo remoto. SCP ofrece un método de autenticación entre los dos equipos, así como un cifrado de la transmisión.

Una vez transferida la carpeta con los archivos que se necesitan ya se puede crear las imágenes de Node-RED y Grafana. Para ello utilizamos el siguiente comando que cambiará dependiendo si creamos la de Node-RED o la de Grafana.

```
LuqueIzan@TFG:~$ sudo docker build -t grafana:<tag> .
```

Ilustración 31: Comando para configurar las imágenes

Después se edita el archivo docker-compose.ylm cambiando el <tag> por un valor cualquiera.

```
version: "3"

services:
  node-red:
    image: nodered:1.3.0
    environment:
      - TZ=Europe/Amsterdam
    ports:
      - "1880:1880"
    networks:
      built_in_container_network:
        ipv4_address: 172.16.238.6
    volumes:
      - nodered-storage:/data
  web:
    image: grafana:1.3.0
    ports:
      - "3000:3000"
    environment:
      - GF_SECURITY_ADMIN_USER=admin
      - GF_SECURITY_ADMIN_PASSWORD=supersecretpassword
    volumes:
      - grafana-storage:/var/lib/grafana
      - ./grafana-provisioning/:/etc/grafana/provisioning
```

Ilustración 32: Tag para cambiado

Finalmente, para inicializar los contenedores y cada vez que queramos trabajar con los programas que hemos instalados, debemos escribir en la terminal el siguiente comando:

```
LuqueIzan@TFG:~/IIACPS_base_folder$ sudo docker-compose up
```

Ilustración 33: Comando para inicializar los contenedores

4.5.4 Influxdb

Una vez instalados Grafana y Node-RED necesitamos crear la base de datos en Influxdb en la que guardaremos los diámetros de los engranajes. Para ello, primero debemos entrar en el terminal del contenedor asociado a influxdb. Dentro del contenedor ejecutaremos el comando para crear la base da datos con su respectivo nombre. En este caso se ha elegido llamar a la base de datos "Engranajes". Dentro de esta base de datos se crea la tabla "Medidas"

```
LuqueIzan@TFG:~$ sudo docker exec -it iiacps_base_folder_database_1 bash
bash-5.0# influx
Connected to http://localhost:8086 version 1.8.4
InfluxDB shell version: 1.8.4
> create database Engranajes
```

Ilustración 34: Comando para acceder a la base de datos influxdb

4.5.4.1 Script Matlab

Hay varias maneras de añadir la información a la base de datos:

- Mediante la terminal
- Utilizando bloques de Node-RED
- Con librerías de Matlab

Para esta solución se ha escogido Matlab, ya que, tenemos todos los datos necesarios porque ha sido el programa con el que se ha hecho el procesado de imagen. Para añadir los datos se ha dividido el código en tres partes. Se muestra el siguiente diagrama de flujo para ver el proceso del código.

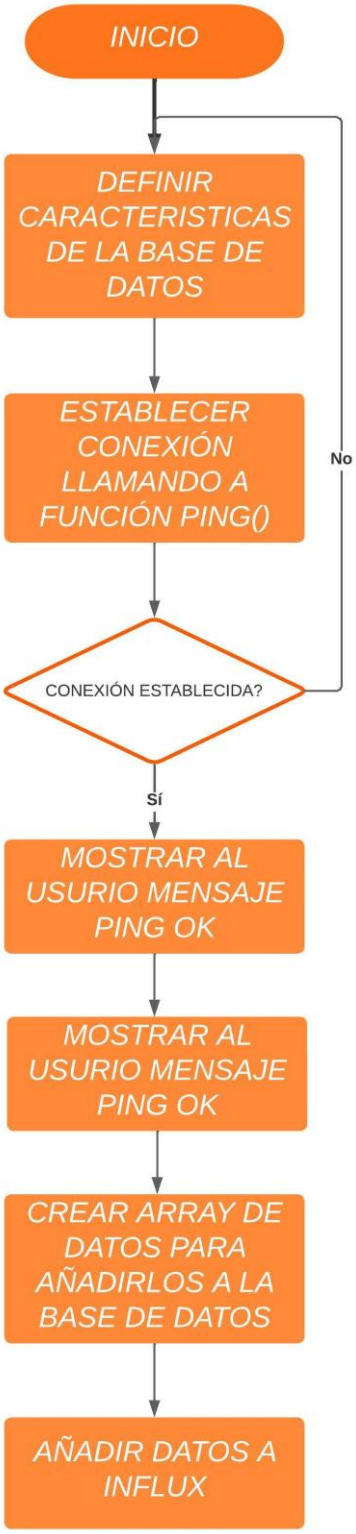


Ilustración 35: Diagrama de flujo de la conexión con Influx a través de Matlab

La primera es la conexión con la base de datos. Para poder realizar la conexión se añade la librería que tendrá las funciones para que el script funcione. Añadimos la URL con la IP publica de la máquina virtual y el puerto asociado a influxdb, el 8086.

En nuestro caso no hemos puesto usuario y contraseña, los campos están vacíos, pero podríamos añadirla para proteger nuestros datos. Como último dato de entrada se añade el nombre de la base de datos. Como hemos comentado en apartados anteriores es "Engranajes". Finalmente añadimos un mensaje para comprobar si la conexión se ha realizado con éxito.

```
addpath(genpath(['C:\Users\izanl\Downloads\influxdb-matlab-master(1)' ...
                 '\influxdb-matlab-master\influxdb-client']));

URL = 'http://20.24.192.227:8086';
USER = '';
PASS = '';
DATABASE = 'Engranajes';
Influxdb = InfluxDB(URL, USER, PASS, DATABASE);

% Test the connection
% [ok, millis] = influxdb.ping();
[ok, millis] = ping(Influxdb);
assert(ok, 'InfluxDB is DOWN!');
fprintf('InfluxDB is OK (%.2fms)\n\n', millis);
```

Ilustración 36: Parte del código donde se crea la base de datos

En la segunda parte preparamos los datos que queremos añadir a la base de datos. En la linea del código que vemos en la imagen los parámetros entre comillas son los measurements, es decir, las columnas que hay en la base de datos. En nuestro caso es el diámetro exterior y interior que se han llamado "diametroExt" y "diametroInt" respectivamente.

```
DatosExtraidos = Series('Medidas') ...
    .fields('diametroExt', Diametro_ext_cm, 'diametroInt', ...
                           Diametro_int_cm, 'Type', type);
```

Ilustración 37: Parte del código donde se acomodan los datos que se quieren introducir en la base de datos

Finalmente añadimos a la base de datos los datos preparados con la siguiente linea de código:

```
Influxdb.writer().append(DatosExtraidos).execute();
```

Ilustración 38: Parte del código donde se añaden los datos

4.6 Grafana

Grafana es una solución de código abierto para ejecutar análisis de datos, extraer métricas y monitorear nuestras aplicaciones con la ayuda de paneles personalizables.

En el caso de este proyecto he creado un dashboard[3] donde poder ver las medidas del engranaje a tiempo real, además de un contaje de los tipos de engranajes que van pasando por la cámara y el tipo de pieza que hay en ese momento.

Lo primero que hice fue inicializar Grafana, añadiendo la dirección IP publica más el puerto en el que esta asignado Grafana, el 3000. Una vez dentro hay que crear los diferentes paneles que mostrarán los datos. Para que estos paneles generen una utilidad necesitan recoger la información de una base de datos.

Como hemos visto en apartados anteriores, desde Matlab, se han ido añadiendo diferentes datos en Influxdb. Es por eso por lo que hay que activar un plugin[4] de influxdb. Cuando el plug-in este descargado hay que introducir unos parámetros para garantizar una conexión exitosa entre Grafana y la base de datos.

Dentro de la configuración del plug-in añadí la dirección ip junto al puerto asociado a influxdb que como ya he explicado en el apartado de los puertos de Azure es el 8086. A través de este puerto Grafana se conectará a la base de datos.

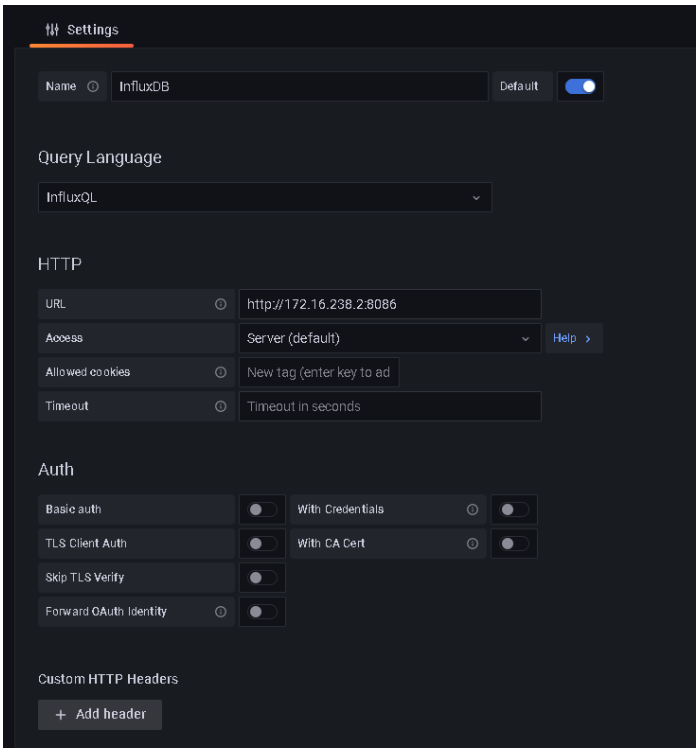


Ilustración 39: Configuración plug-in Influxdb

Una vez añadida esta funcionalidad empecé a añadir los paneles que informan al usuario del proceso de producción.

[3] Es el panel o la pantalla donde se refleja, mediante una representación gráfica, las diferentes métricas

[4] Un plugin es aquella aplicación que, en un programa informático, añade una funcionalidad adicional o una nueva característica al software.

Hay diferentes paneles según la manera en la que se quiere visualizar la información se eligen unos u otros. En mi caso comencé añadiendo la información que había guardado en la base de datos influxdb para comprobar su correcto funcionamiento. Estos datos son el diámetro exterior e interior y el tipo de engranaje. Para visualizar valores estáticos he elegido el panel "stat". Dentro de la configuración de este panel se debe crear la query que manda a la base de datos para recoger la información que se quiera visualizar en el panel.

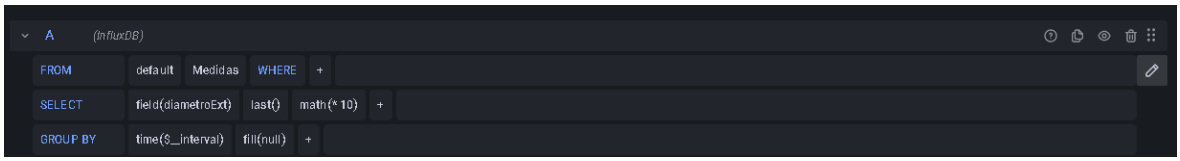


Ilustración 40: Interfaz donde se seleccionan los datos que se quieren visualizar

Como podemos ver en la imagen, una query se compone de una combinación de palabras que determinan la información que se quiere extraer de la base de datos. En esta query tenemos diferentes palabras clave que tienen diferentes funciones. A nosotros en este caso nos interesa la petición SELECT y FROM, que serán las únicas que se modificarán.

SELECT: Es el comando más utilizado a la hora de mandar peticiones de información. En este comando se especifica el campo del que se quiere extraer la información.

FROM: Es el comando para especificar de que tabla se quiere extraer la información

Este proyecto tiene tres campos en la base de datos; diámetro exterior, diámetro interior y tipo. Además de seleccionar el campo con el que se quiere trabajar también se puede utilizar funciones a estos campos. Una de estas funciones es last() que será la que utilizaremos, ya que, elige el último valor que haya en ese campo, que es precisamente lo que queremos escoger para visualizarlo. También hice uso de la función math(), ya que, Grafana solo da la opción de poner los datos en metros o milímetros y para ser fiel a las unidades se ha multiplicado por diez para pasar de centímetros a milímetros.

Finalmente se elige la tabla de dónde queremos extraer la información, que es "Medidas". Repetimos esta acción solo cambiando field.

A parte de la selección de la información mediante querys a la base de datos de influxdb también tuve que cambiar una serie de parámetros dentro del panel. Uno de estos es la elección de las unidades del valor mostrado de la base de datos. Elegí mostrarlo en milímetros.

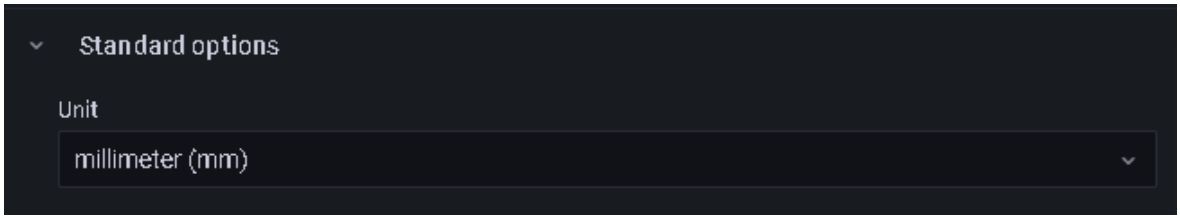


Ilustración 41: Opción de unidades en Grafana

Otra opción que cambié fue el tamaño del texto, que se accede desde la pestaña “Text Size” al cual se le dio un valor de 60.

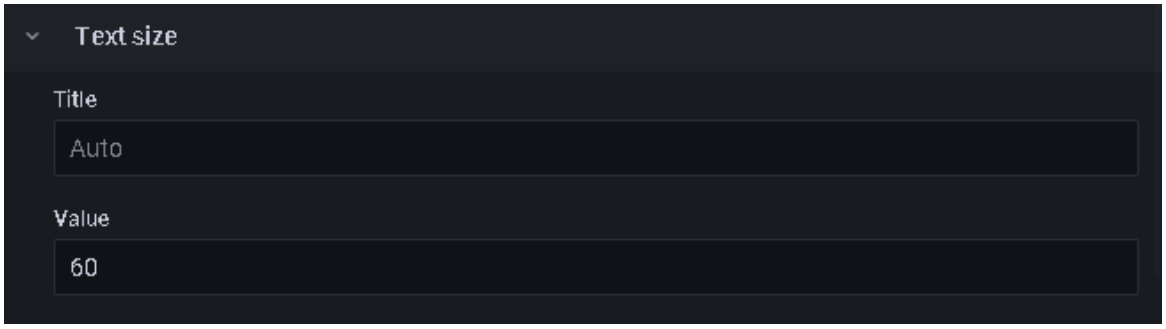


Ilustración 42: Opción tamaño de texto en Grafana

Luego he querido hacer un conteo de los tipos de engranajes que van pasando por la cámara. Para ello volví a seleccionar el panel stat, para mostrar un único valor, pero esta vez el campo es “Type” y se añaden dos modificaciones comparado a las peticiones anteriores. Ahora ya no utilizo last(), ya que no quiero mostrar el último valor. Paso a utilizar cont() que cuenta el número de valores o filas que hay en este campo.

Pero si se hiciese un SELECT y un FROM variando lo mencionado nos devolvería el conteo de todos los valores que hay en “type”. Cosa que no nos interesa debido a que queremos hacer un conteo de cada tipo. Es por eso que hay que introducir el comando WHERE.

WHERE: Es un comando que se usa para especificar una condición al obtener los datos de una tabla. Tiene la función de filtrar la información que nos conviene y desechar la que no se requiere.

Para determinar cada uno de los tipos de engranaje se introduce desde la query original de grafana [WHERE Type = <número del engranaje>]. Esto se ha repetido para los 6 tipos de engranaje que hay.

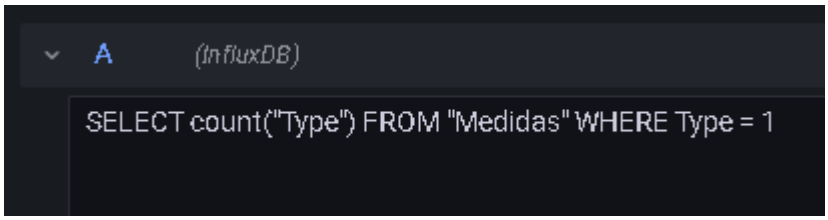


Ilustración 43: Query de la cantidad de engranajes de un mismo tipo en Grafana

Por último, en lo que concierne a la información que hay en la base de datos, se ha añadido una gráfica de barras para mostrar el histórico de todos los tipos de engranajes que van pasando durante el tiempo por la cámara.

Para ello, he utilizado el panel “Bar Char” en el que se muestra información mediante barras. Para hacer una selección de todos los tipos de engranajes que hay en la base de datos se ha realizado la siguiente query:

SELECT "Type" FROM "Medidas"

Finalmente se añade texto para mejorar la presentación del dashboard y que quede claro para el operario la información que se está mostrando.

Lo que he hecho es crear tres paneles tipo "text" y dentro de estos selecciono la opción de HTML para que las propiedades del texto las pueda modificar a mi gusto. He utilizado <b> para poner el texto en negrita, <i> para escribir en cursiva y <u> para subrayar el texto. Además se añaden estilos como text-align para indicar la posición del texto y font-size para indicar el tamaño de la letra.

Texto del titulo en HTML:

 <i><b>Diseño e implementación de un sistema de procesado de imagen aplicado a un proceso industrial<b/><i/>

Texto de los apartados:

 <u>ENGRANAJE ACTUAL</u>

 <u>ENGRANAJES TOTALES </u>

También se añade una imagen del logo de la UPC para mejorar la presentación. Para añadir imágenes en Grafana debemos obtener una URL publica y añadirlo mediante un panel de tipo "text" en HTML. He buscado una imagen por google del logo de la UPC, he obtenido su URL y la he añadido al panel de la siguiente manera:

<img src="https://upload.wikimedia.org/wikipedia/ca/thumb/b/b5/Logo_upc.svg/1200px-Logo_upc.svg.png" width=300 length=300>

Como podemos ver lo que sigue a src es el link de la imagen en google y width y length son el ancho y el largo para dimesnionar las medidas de la imagen.

El dashboard final tiene el siguiente aspecto:

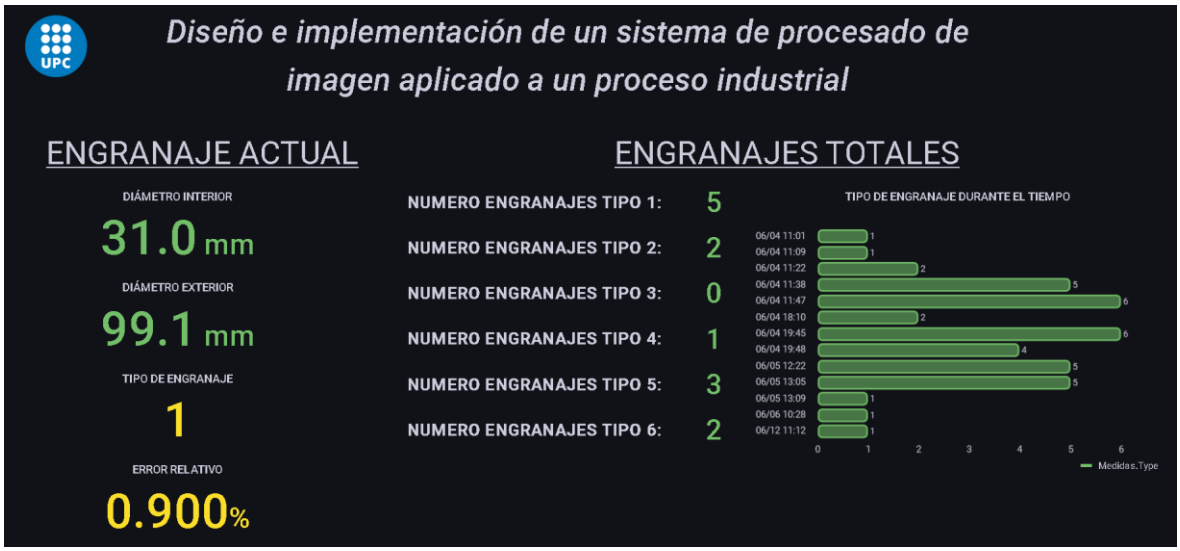


Ilustración 44: Dashboard final de Grafana

5 Validación

5.1 Validación operacional

5.1.1 Comunicación Terminal - Máquina Virtual

DESCRIPCIÓN DEL ENSAYO	Se pretende conectar a la máquina virtual que se está ejecutando en Azure mediante la terminal con el protocolo SSH. Para ello necesitamos escribir un comando en la terminal con la ruta donde está la clave con extensión .pem y la IP publica de nuestra máquina virtual. ssh -i <ruta de la clave> <nombre máquina virtual>@<ip pública>
RESULTADO ESPERADO	Se espera que la conexión sea aceptada y esto realice un cambio en la interfaz gráfica de la línea donde se vea

RESULTADO DE LA PRUEBA	El resultado de la prueba no ha sido el esperado. Se ha denegado la conexión con la máquina virtual añadiendo el siguiente mensaje.
	`Load key "C:\\Users\\izanl\\.ssh\\": Operation not supported on socket`
	`LuqueIzan@20.24.214.210: Permission denied (publickey).`
	Ilustración 45: Resultado de la prueba con SSH

SOLUCIÓN EN CASO DE FALLO	El fallo en este ensayo fue provocado porque se ha seleccionado la ruta donde está la clave con extensión .pem pero no se ha especificado el nombre del archivo. Para solucionarlo se escribe la misma linea de comando, pero añadiendo el nombre del archivo que es “TestClave_key.pem”.
	`Last login: Mon Jun  6 08:25:17 2022 from 46.24.244.115`
	`LuqueIzan@TFG:~$`
	Ilustración 46: Solución al fallo de conexión SSH

Tabla 8: Validación operacional donde apreciamos el funcionamiento de la comunicación Terminal – Máquina Virtual

5.1.2 Comunicación Matlab - InfluxDB

DESCRIPCIÓN DEL ENSAYO	Cuando se ha acabado el procesado de imagen y ya disponemos de los diámetros y el tipo de engranaje se tiene que establecer una comunicación entre Matlab y la base de datos donde vamos a introducir estos datos. Para ello desde Matlab, se ejecuta un script que muestra por pantalla si se ha conectado bien a Influxdb.
RESULTADO ESPERADO	Si la conexión se ha realizado con éxito se verá por pantalla un mensaje que dice lo siguiente: "InfluxDB is OK". Si, por otro lado, no se ha realizado con éxito obtendremos el siguiente mensaje: "InfluxDB is DOWN"
RESULTADO DE LA PRUEBA	No se ha realizado la conexión con éxito. Sale el mensaje de "InfluxDB is DOWN" `InfluxDB is DOWN!` *Ilustración 47: Resultado comunicación con base de datos desde Matlab*
SOLUCIÓN EN CASO DE FALLO	La solución a este fallo es cambiar la dirección ip pública del script de Matlab, ya que, cada vez que se inicia la máquina virtual esta dirección cambia. `InfluxDB is OK (539.84ms)` *Ilustración 48: Solución al fallo de conexión con Influxdb*

Tabla 9: Validación operacional donde apreciamos el funcionamiento de la comunicación Matlab – Influxdb

5.1.3 Conexión Influxdb - Grafana

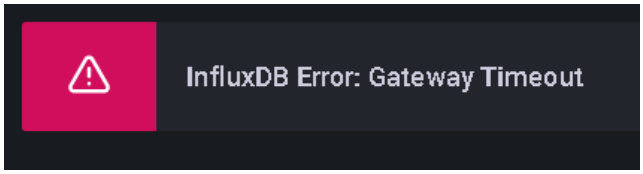
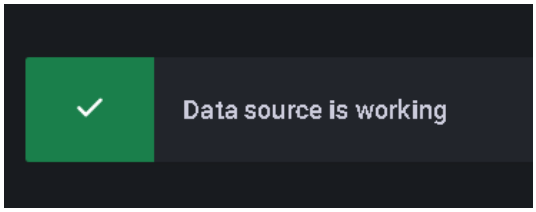
DESCRIPCIÓN DEL ENSAYO	Para poder agregar en el dashboard los datos que hemos almacenado en Influxdb a través de Matlab tenemos que instalar el plug-in de Influxdb en Grafana. Una vez instalado se añaden los datos para poder establecer la conexión.
RESULTADO ESPERADO	Si la conexión se ha realizado con éxito Grafana mostrará un mensaje para informar al usuario que la conexión está establecida.
RESULTADO DE LA PRUEBA	No se ha realizado la conexión con éxito. Sale el mensaje de "InfluxDB Error: Gateway Timeout".  *Ilustración 49: Mensaje error Influxdb-Grafana*
SOLUCIÓN EN CASO DE FALLO	Este fallo viene dado porque la ip no se ha introducido correctamente, y por lo tanto, pasa el tiempo de conexionado. Una vez colocada la dirección correcta sale el mensaje.  *Ilustración 50: Mensaje conexión exitosa Influxdb-Grafana*

Tabla 10: Validación operacional donde apreciamos el funcionamiento de la comunicación Influxdb - Grafana

5.2 Validación funcional

5.2.1 Imagen con medidas

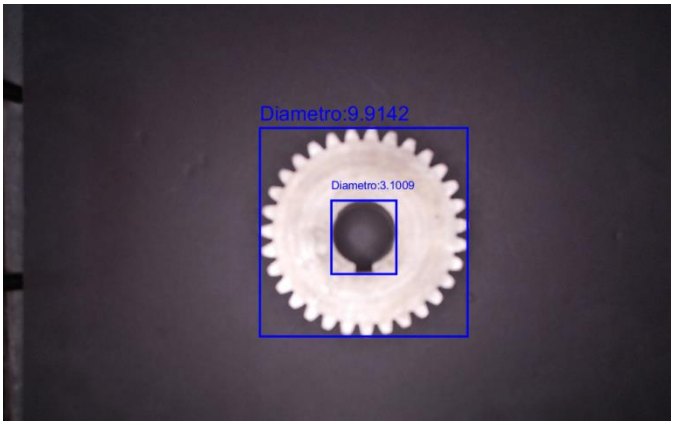
DESCRIPCIÓN DEL ENSAYO	En este ensayo se pretende visualizar las medidas del engranaje poniendo en marcha todo el procesado de imagen desde que llega la foto inicial de la cámara de la Raspberry hasta que se muestra la imagen con los diámetros. Para ello se va a ejecutar toda la sección del script de Matlab donde hace este procesado.
RESULTADO ESPERADO	Se muestra la imagen con dos rectángulos que recubren el circulo que ha analizado y el valor de cada diámetro.
RESULTADO DEL ENSAYO	La prueba se ha realizado con éxito.  *Ilustración 51: Resultado Engranaje con diámetros*
SOLUCIÓN EN CASO DE FALLO	-

Tabla 11: Validación funcional donde apreciamos el resultado final del engranaje con sus medidas

5.2.2 Sistema completo

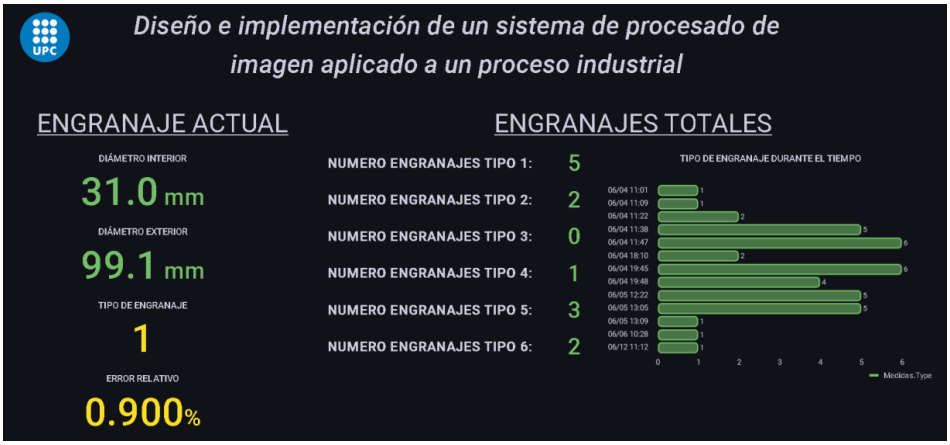
DESCRIPCIÓN DEL ENSAYO	Validación final de todo el proceso donde compruebo la llegada de la imagen a Matlab, el procesado de imagen para obtener las medidas, la escritura de estos datos en InfluxDB y visualización de los datos con Grafana.
RESULTADO ESPERADO	Obtención de la imagen con los diámetros del engranaje y visualizar los datos en Grafana.
RESULTADO DEL ENSAYO	El ensayo muestra correctamente la imagen y también los datos por Grafana.  *Ilustración 52: Resultado Dashboard Grafana*
SOLUCIÓN EN CASO DE FALLO	-

Tabla 12: Validación funcional donde apreciamos el dashboard final de Grafana

6 Presupuesto

En este proyecto no se han adquirido componentes, es por eso que la adquisición de hardware no se ve reflejada en el presupuesto. El coste se basará en las horas que se han invertido en diseñar la solución al problema planteado. Estas horas se diferenciarán en dos tipos.

Unas son las horas de estudio y diseño de la solución, es decir, todo el tiempo que conlleva a encontrar la solución más optima al problema inicial. Esto implica las horas que se invierten, por ejemplo, en elegir los programas más convenientes o que mejor funcionan para el uso que le queremos dar. Elección de base de datos, estructura IoT, características de máquina virtual o el software utilizado para procesar la imagen.

Por otro lado, tenemos las horas de desarrollo y validación, que son la gran mayoría. Estas conformar el tiempo que se ha pasado desarrollando la solución, haciendo pruebas y validando que el funcionamiento es el correcto.

Finalmente, al total se le añade un porcentaje de costes indirectos, en los que se tomaran en cuenta costes relacionados con desplazamientos o licencias de software. Se cree oportuno que dado el tipo de solución desarrollada se aplica un 15% de costes indirectos.

	Horas (h)	**Precio (€/h)**	**Total (€)**
Estudio y diseño de la solución	40	52	2080
Desarrollo	160	52	8320
Validación	40	52	2080
PRECIO SIN COSTES INDIRECTOS E IVA			12480
Coste indirecto - 15%			1872
PRECIO SIN IVA			14352
IVA - 21%			3013,92
PRECIO TOTAL			**17365,92**

Tabla 13: Presupuesto final del proyecto

7 Conclusiones

En este proyecto se ha diseñado un sistema de procesado de imagen enfocado a aplicarse en un proceso industrial. Consiste en un conjunto de engranajes con diferentes medidas que van pasando por una cámara conectada a una Raspberry Pi. Esta cámara toma una imagen que se procesa para extraer características del objeto, en este caso los diámetros interiores y exteriores. Una vez obtenidos estos datos se almacenan en una base de datos y se muestran por un dashboard para que, a partir de esta información, se tomen decisiones en procesos posteriores.

Todo este proceso esta soportado por una arquitectura IoT. La información obtenida en el procesado de imagen se almacena y muestra a través de una máquina virtual ejecutándose en la nube por vía de la tecnología de Azure.

Como se puede comprobar se ha logrado cumplir con éxito todos los objetivos planteados al inicio del proyecto. Se ha diseñado un procesado de imagen que presenta una arquitectura flexible y fácilmente adaptable a un proceso industrial.

Durante el desarrollo de la solución, una vez cumplidos los objetivos propuestos, se planteó añadir la imagen al dashboard para mostrar de una manera más gráfica la información obtenida. Desafortunadamente no fue posible debido a que para añadir una imagen al dashboard debíamos adjuntarla mediante una URL, la cual se intentó crear desde Node-RED y python. Además de que la imagen debería actualizarse por cada engranaje y esto no es compatible, ya que, el sistema de refresco del dashboard se intuye que no se aplica para una URL porque está más enfocada a bases de datos.

7.1 Futuro del proyecto

Dada la flexibilidad que ofrece este proyecto hay gran variedad de oportunidades de mejora en los que poder situar la aplicación diseñada.

Como trabajo futuro se podría integrar la aplicación creada en este proyecto con sistemas SCADA que existan en la planta de producción. Esto nos proporcionaría más control en la planta de producción, por ejemplo, interactuando directamente con dispositivos como sensores o motores. También podríamos desarrollar paneles de alarma para avisar de cualquier fallo que surja durante la ejecución del proceso.

También podríamos integrarlo a un sistema ERP o MES del sistema productivo donde a través de la información que va generando la aplicación se pueda determinar qué tipo de engranaje se va a fabricar y cuando. Integrando estos sistemas el proyecto puede adoptar una posición más enfocada a la planificación y a la ejecución de la producción que harán que a largo plazo supongan un ahorro económico y de tiempo.

8 Referencias

1. INDUSTRIA 4.0. *deloitte.* [En línea] 13 de 06 de 2022. [Citado el: 13 de 06 de 2022.] https://www2.deloitte.com/es/es/pages/manufacturing/articles/que-es-la-industria-4.0.html.

2. LINKEDIN. [En línea] 13 de 06 de 2022. [Citado el: 13 de 06 de 2022.] https://www.linkedin.com/pulse/machine-vision-camera-types-parth-desai/.

3. Mathworks. [En línea] 13 de 06 de 2022. https://es.mathworks.com/products/image.html.

4. VISIÓN ARTIFICIAL. [En línea] 13 de 06 de 2022. https://www.contaval.es/que-es-la-vision-artificial-y-para-que-sirve/.

5. VISIÓN ARTIFICIAL FRESADO. [En línea] 13 de 06 de 2022. https://repository.javeriana.edu.co/handle/10554/38555.

6. Github. [En línea] 13 de 06 de 2022. https://github.com/EnricSala/influxdb-matlab.

7. GRAFANA. [En línea] 13 de 06 de 2022. https://grafana.com/.

8. Azure Microsoft. [En línea] 13 de 06 de 2022. https://azure.microsoft.com/es-es/features/azure-portal/.

9. InfluxData. [En línea] 13 de 06 de 2022. https://docs.influxdata.com/influxdb/v1.8/concepts/glossary/#field-set.

10. InfluxData Manage Database. [En línea] 13 de 06 de 2022. https://docs.influxdata.com/influxdb/v1.8/query_language/manage-database/.

11. Mathworks RegionProps. [En línea] 13 de 06 de 2022. https://es.mathworks.com/help/images/ref/regionprops.html#d123e243268.

12. Mathworks Insert Text. [En línea] 13 de 06 de 2022. https://es.mathworks.com/help/vision/ref/inserttext.html#d123e214107.

13. Mathworks bwareaopen. [En línea] 13 de 06 de 2022. https://es.mathworks.com/help/images/ref/bwareaopen.html#buwet8w-1-P.

14. Mathworks Image Erode. [En línea] 13 de 06 de 2022. https://es.mathworks.com/help/images/ref/imerode.html#d123e139818.

15. Mathworks Measure Image. [En línea] 13 de 06 de 2022. https://es.mathworks.com/help/images/measure-distances-in-images.html?searchHighlight=measure%20distance%20&s_tid=srchtitle_measure%20distance%20_1.

16. Mathwork Measure Circularity. [En línea] 13 de 06 de 2022. https://es.mathworks.com/matlabcentral/answers/167366-how-to-measure-circularity?s_tid=ta_ans_results.

17. profesionalreview. [En línea] 13 de 06 de 2022. https://www.profesionalreview.com/2018/11/30/ssh-windows-10/.

18. hostinger. [En línea] 13 de 06 de 2022. https://www.hostinger.es/tutoriales/comando-scp.

19. Grafana. [En línea] 13 de 06 de 2022. https://grafana.com/grafana/plugins/marcusolsson-dynamictext-panel/.

20. volkovlabs. [En línea] 13 de 06 de 2022. https://demo.volkovlabs.io/d/DtsbVE3Mk/analyzing-camera-feed-with-ai-work-in-progress?orgId=1).

21. cookbook nodered. [En línea] 13 de 06 de 2022. https://cookbook.nodered.org/http/create-an-http-endpoint.

22. nodered. [En línea] 13 de 06 de 2022. https://nodered.org/docs/tutorials/first-flow.

23. hostinger docker. [En línea] 13 de 06 de 2022. https://www.hostinger.es/tutoriales/como-instalar-y-usar-docker-en-ubuntu.

24. diarioinforme. [En línea] 13 de 06 de 2022. https://diarioinforme.com/como-usar-docker-cp-para-copiar-archivos-entre-hosts-y-contenedores/.

25. docs influxdata. [En línea] 13 de 06 de 2022. https://docs.influxdata.com/influxdb/v2.2/.

26. paralleldots. [En línea] 13 de 06 de 2022. https://blog.paralleldots.com/data-science/research-papers-image-classification/.