

**UNIVERSIDAD EUROPEA DE MADRID**

**ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO**

**GRADO EN INGENIERÍA INFORMÁTICA**

PROYECTO FIN DE GRADO

**PROYECTO INDUSTRIA 4.0: DETECCIÓN DE GALLETAS ROTAS MEDIANTE VISIÓN ARTIFICIAL EN LÍNEA DE PRODUCCIÓN**

**Adrián Gabriel García Ortiz**

**Dirigido por**

**Profesor Héctor Barrio Crespo**

**CURSO 2021-2022**

**TÍTULO**: PROYECTO INDUSTRIA 4.0: DETECCIÓN DE GALLETAS ROTAS MEDIANTE VISIÓN ARTIFICIAL EN LÍNEA DE PRODUCCIÓN

**AUTOR**: ADRIÁN GABRIEL GARCÍA ORTIZ

**TITULACIÓN**: GRADO EN INGENIERÍA INFORMÁTICA

**DIRECTOR/ES DEL PROYECTO**: PROFESOR HÉCTOR BARRIO CRESPO

**FECHA**: JUNIO DE 2022

# RESUMEN

Actualmente se utiliza la inteligencia artificial en múltiples ámbitos, uno de estos es la detección de errores. La detección de errores mediante inteligencia artificial es un campo en expansión con una gran cantidad de empresas que ofrecen soluciones, pero la mayoría son personalizadas para el cliente. Este proyecto se ha desarrollado con el objetivo de realizar un sistema que permita la detección de defectos en una línea de producción mediante visión artificial de forma rápida y sencilla para un usuario avanzado. El objetivo es crear un sistema que permita una integración simple mediante la realización de un conjunto de imágenes del objeto a detectar con sus posibles fallos, el entrenamiento de un modelo de detección mediante inteligencia artificial que distinga entre productos correctos y defectos y la implantación de un equipo con una webcam para captar las imágenes que procesa el modelo entrenado. Este sistema contará con un servidor que dará avisos al usuario de los fallos detectados un e-mail.

**Palabras clave: machine learning, visión artificial, transformación digital, industria 4.0**

# ABSTRACT

Artificial intelligence is currently used in multiple fields, one of these is error detection. Artificial intelligence error detection is a growing field with a large number of companies offering solutions, but most are customized to the customer. This project has been developed with the aim of creating a system that allows the detection of defects in a production line by means of artificial vision quickly and easily for an advanced user. The objective is to create a system that allows simple integration by creating a set of images of the object to be detected with its possible failures, training a detection model using artificial intelligence that distinguishes between correct products and defects, and implementing a computer with a webcam to capture the images that the trained model processes. This system will have a server that will notify the user of the failures detected by e-mail.

**Keywords: machine learning, artificial vision, digital transformation, industry 4.0**

**AGRADECIMIENTOS**

**Gracias a mis padres, a mis amigos y a toda la gente que me ha apoyado para terminar este proyecto.**

*Solo podemos ver poco del futuro, pero lo suficiente para darnos cuenta de que hay mucho que hacer.* Alan Turing

# TABLA RESUMEN

|  |  |
| --- | --- |
|  | **DATOS** |
| **Nombre y apellidos:** | Adrián Gabriel García Ortiz |
| **Título del proyecto:** | PROYECTO INDUSTRIA 4.0: DETECCIÓN DE GALLETAS ROTAS MEDIANTE VISIÓN ARTIFICIAL EN LÍNEA DE PRODUCCIÓN |
| **Directores del proyecto:** | Profesor Héctor Barrio Crespo |
| **El proyecto se ha realizado en colaboración de una empresa o a petición de una empresa:** | NO |
| **El proyecto ha implementado un producto:**  (esta entrada se puede marcar junto a la siguiente) | NO |
| **El proyecto ha consistido en el desarrollo de una investigación o innovación:**  (esta entrada se puede marcar junto a la anterior) | SI |
| **Objetivo general del proyecto:** | DETECCION DE GALLETAS ROTAS CON VISION ARTIFICIAL |

**Índice**

[RESUMEN 3](#_Toc109090090)

[ABSTRACT 4](#_Toc109090091)

[TABLA RESUMEN 7](#_Toc109090092)

[Capítulo 1. RESUMEN DEL PROYECTO 12](#_Toc109090093)

[1.1 Contexto y justificación 12](#_Toc109090094)

[1.2 Planteamiento del problema 12](#_Toc109090095)

[1.3 Objetivos del proyecto 12](#_Toc109090096)

[1.4 Resultados obtenidos 12](#_Toc109090097)

[1.5 Estructura de la memoria 13](#_Toc109090098)

[Capítulo 2. ANTECEDENTES / ESTADO DEL ARTE 15](#_Toc109090099)

[2.1 Estado del arte 15](#_Toc109090100)

[2.2 Contexto y justificación 15](#_Toc109090101)

[2.3 Planteamiento del problema 16](#_Toc109090102)

[Capítulo 3. OBJETIVOS 17](#_Toc109090103)

[3.1 Objetivos generales 17](#_Toc109090104)

[3.2 Objetivos específicos 17](#_Toc109090105)

[3.3 Beneficios del proyecto 17](#_Toc109090106)

[Capítulo 4. DESARROLLO DEL PROYECTO 18](#_Toc109090107)

[4.1 Planificación del proyecto 18](#_Toc109090108)

[4.2 Descripción de la solución, metodologías y herramientas empleadas 19](#_Toc109090109)

[4.3 Recursos requeridos 20](#_Toc109090110)

[4.4 Presupuesto 20](#_Toc109090111)

[4.5 Viabilidad 21](#_Toc109090112)

[4.6 Resultados del proyecto 22](#_Toc109090113)

[Capítulo 5. DISCUSIÓN 24](#_Toc109090114)

[Capítulo 6. CONCLUSIONES 25](#_Toc109090115)

[6.1 Conclusiones del trabajo 25](#_Toc109090116)

[6.2 Conclusiones personales 25](#_Toc109090117)

[Capítulo 7. FUTURAS LÍNEAS DE TRABAJO 26](#_Toc109090118)

[Capítulo 8. Bibliografía y Referencias 27](#_Toc109090119)

[Capítulo 9. ANEXOS 28](#_Toc109090120)

**Índice de Figuras**

[Ilustración 1. Etiquetado de galletas con LabelImg 13](#_Toc109090081)

[Ilustración 2. Entrenamiento del modelo de detección 13](#_Toc109090082)

[Ilustración 3. Sistema de detección de defectos de E2M 15](#_Toc109090083)

[Ilustración 4. Cronograma de desarrollo del proyecto 18](#_Toc109090084)

[Ilustración 5. Línea de tiempo de desarrollo del proyecto 19](#_Toc109090085)

[Ilustración 6. Proceso de detección de defectos 1 19](#_Toc109090086)

[Ilustración 7. Proceso de detección de defectos 2 20](#_Toc109090087)

[Ilustración 8. Galletas de ejemplo utilizadas en el proyecto 20](#_Toc109090088)

[Ilustración 9. Tarifas del servicio Amazon Lookout 22](#_Toc109090089)

**Índice de Tablas**

Tabla 1. Presupuesto del proyecto...............................................................................................20

Tabla 2. Licencias del proyecto....................................................................................................21

# RESUMEN DEL PROYECTO

Este proyecto se ha desarrollado con el objetivo de realizar un sistema que permita la detección de defectos en productos en una línea de producción. El sistema utiliza la inteligencia artificial y la visión artificial para generar un modelo predictivo que permite la detección de fallos. El sistema cuenta con una implementación que avisa al usuario de los fallos en la línea de producción.

## Contexto y justificación

Actualmente existen multitud de empresas que ofrecen sistemas que se pueden desplegar en una línea de producción para facilitar a las empresas la detección de fallos en los productos que elabora. Estos sistemas son costosos y requieren de mucho tiempo de estudio y desarrollo para ofrecer una solución personalizada al usuario. En el mercado podemos encontrar productos como Amazon Lookout que proporcionan soluciones para estos sistemas.

El sistema que se plantea desarrollar en este proyecto intenta ser una solución sencilla de implementar para que un usuario pueda elaborar un conjunto de imágenes, entrenar un modelo y desplegar un servidor que en conjunto con una webcam pueda colocarse en una línea de producción.

## Planteamiento del problema

El problema que va a tratar este proyecto concretamente es un ejemplo de una línea de producción en una fábrica de galletas. Las galletas pueden tener defectos en su elaboración, como grietas, roturas, etc. que un operario humano a simple vista puede distinguir. De esta forma un sistema con inteligencia artificial puede distinguir los fallos que se producen en una galleta mediante visión artificial.

## Objetivos del proyecto

Los objetivos que busca el proyecto son:

* Realizar un dataset de galletas con fallos para el modelo predictivo
* Crear y entrenar un modelo predictivo para la detección de errores
* Implementar un sistema de visión artificial que haga uso de un modelo predictivo para detectar errores
* Desplegar el sistema de detección, que avise al usuario de los errores

## Resultados obtenidos

Se ha generado un dataset de galletas con galletas con defectos y galletas en buen estado:

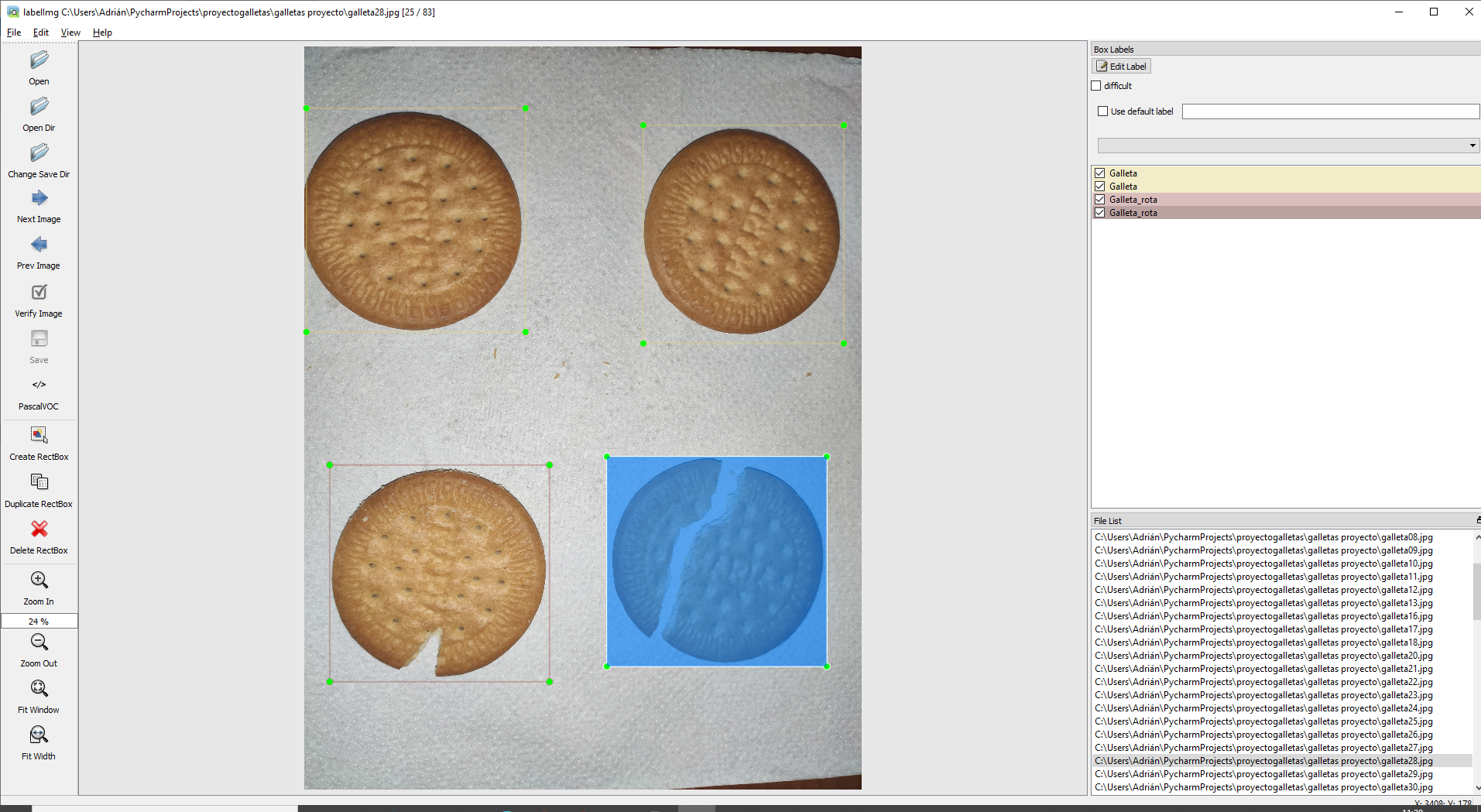


Ilustración . Etiquetado de galletas con LabelImg

Se ha generado un sistema que permite entrenar un modelo de detección a partir de una colección de imágenes etiquetadas:

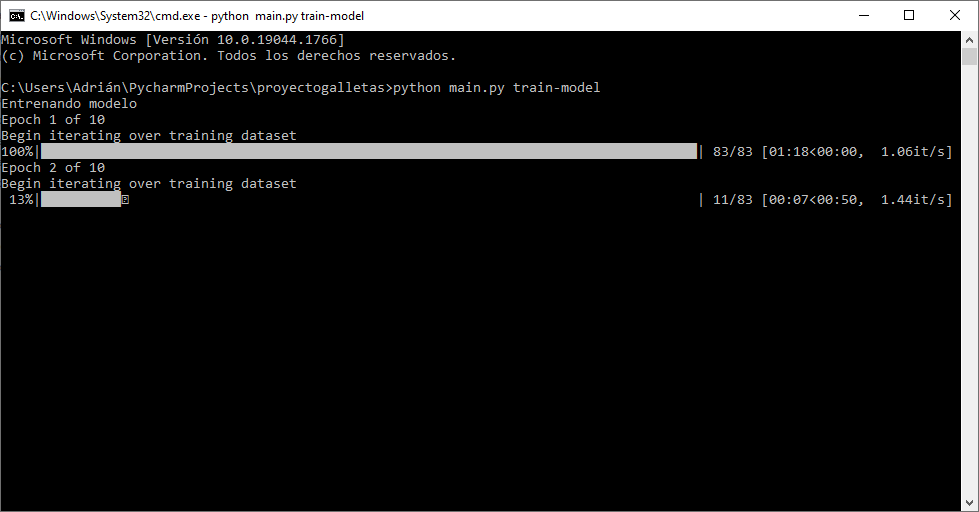


Ilustración . Entrenamiento del modelo de detección

Se ha realizado un sistema que permite detectar defectos mediante un modelo de detección y un servidor web que procesa las imágenes para utilizar el modelo.

## Estructura de la memoria

En el capítulo 2 se tratará el estado del arte actual de la detección de errores mediante inteligencia artificial, de las empresas que los desarrollan y sus soluciones.

En el capítulo 3 se hablará de los objetivos que se han planteado para el proyecto, de su alcance y de las mejoras que se obtienen.

En el capítulo 4 se explicará el desarrollo del proyecto, de como se han planificado las distintas etapas, los problemas que han surgido y el resultado final.

En el capítulo 5 se discutirá de las aproximaciones existentes al problema y de las diferencias respecto de la solución planteada.

En el capítulo 6 se expondrán las conclusiones alcanzadas con el desarrollo del proyecto.

# ANTECEDENTES / ESTADO DEL ARTE

En este capítulo se tratarán todos los detalles y cuestiones que justifican el desarrollo del proyecto. Aspectos tales como el estado actual de la detección de errores en la elaboración de productos, los sistemas que existen actualmente y las soluciones que ofrecen actualmente las empresas.

## Estado del arte

Actualmente existen multitud de algoritmos y sistemas que permiten la detección de objetos a partir de colecciones de objetos o datasets. Estos datasets tienen que elaborarse de forma específica para cada caso a tratar, como defectos en la producción de una pieza.

Dependiendo del sistema, la solución comercial y la precisión necesaria pueden utilizarse distintos algoritmos, como el algoritmo YOLO o el algoritmo Faster R-CNN ((Region Based – Convolutional Neural Network).

En el mercado ya existen soluciones comerciales, como la recientemente lanzada Amazon Lookout, que permite a los clientes contratar una solución personalizada para su negocio e integrarla con AWS para obtener información en tiempo real de los problemas que puedan surgir. Otras soluciones comerciales que se pueden encontrar en el mercado son servicios personalizados que ofrecen empresas como Micro.ai o E2M, que tienen un coste más elevado.

## Contexto y justificación

El problema de las soluciones existentes para la detección de defectos es su elevado coste y lo difícil que suele ser su despliegue, ya que requiere un estudio de las líneas de producción, el análisis de los productos y sus posibles defectos, la colocación de cámaras en la línea de producción y la instalación de terminales para comprobar el estado del sistema.

En el caso de Amazon Lookout, el pago se realiza por horas de uso con lo que para una pequeña empresa puede no ser una opción asequible.

En el caso de E2M se trata de un sistema cerrado que requiere un despliegue e instalación completa por parte de la empresa que ofrece la solución comercial.



Ilustración . Sistema de detección de defectos de E2M

## Planteamiento del problema

El sistema que plantea el desarrollo de este proyecto se concibe como una solución sencilla, económica, ya que estará basada en una SBC, y fácil de desplegar en una línea de producción. Además, se aplicará el algoritmo de detección Faster R-CNN, por lo que será más rápido preparar un modelo de detección y tendrá una mayor precisión.

# OBJETIVOS

## Objetivos generales

* El objetivo principal del proyecto es crear un sistema de detección mediante visión artificial que permita detectar errores en líneas de producción.

## Objetivos específicos

* Creación de un dataset de galletas sin defectos y con defectos para su detección mediante un modelo predictivo
* Creación y entrenamiento de un modelo predictivo con redes neuronales convolucionales mediante el uso de inteligencia artificial para distinguir entre galletas sin defectos y con defectos.
* Desarrollo de un sistema que haga uso de visión artificial y que, mediante un modelo predictivo entrenado con IA, detecte galletas con defectos.

## Beneficios del proyecto

Los beneficios del proyecto que aporta respecto de las soluciones actuales en el mercado es su sencillez, rapidez y facilidad de implementación en un sistema pequeño y tolerante a fallos.

# DESARROLLO DEL PROYECTO

## Planificación del proyecto

Para el desarrollo de este proyecto se ha optado por una metodología de desarrollo ágil, en concreto la metodología SCRUM. Se ha escogido esta metodología debido a que permite el desarrollo incremental de las distintas iteraciones en las que se planea dividir el proyecto, con la supervisión del tutor en cada una de las iteraciones y la posibilidad de hacer modificaciones en caso de que sea necesario.

Estas iteraciones o “sprints” del proyecto constarán de 4 semanas, tras lo cual se realizará una reunión con el tutor para informarle de los avances, revisar lo que se ha realizado y realizar posibles cambios si fuese necesario.

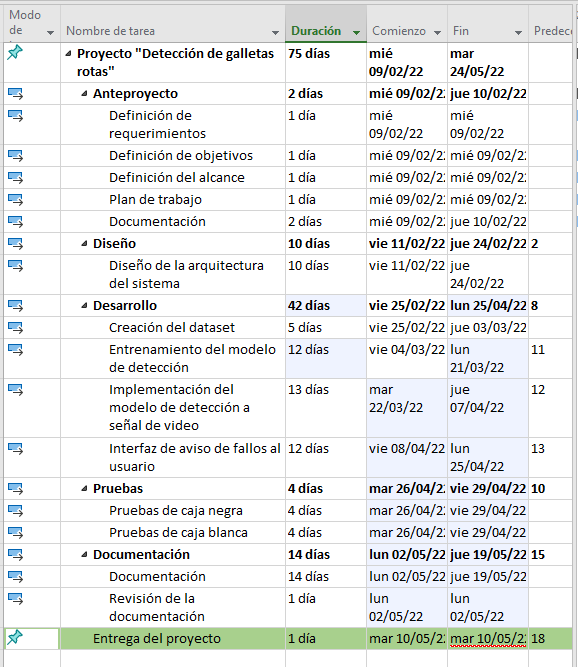


Ilustración . Cronograma de desarrollo del proyecto

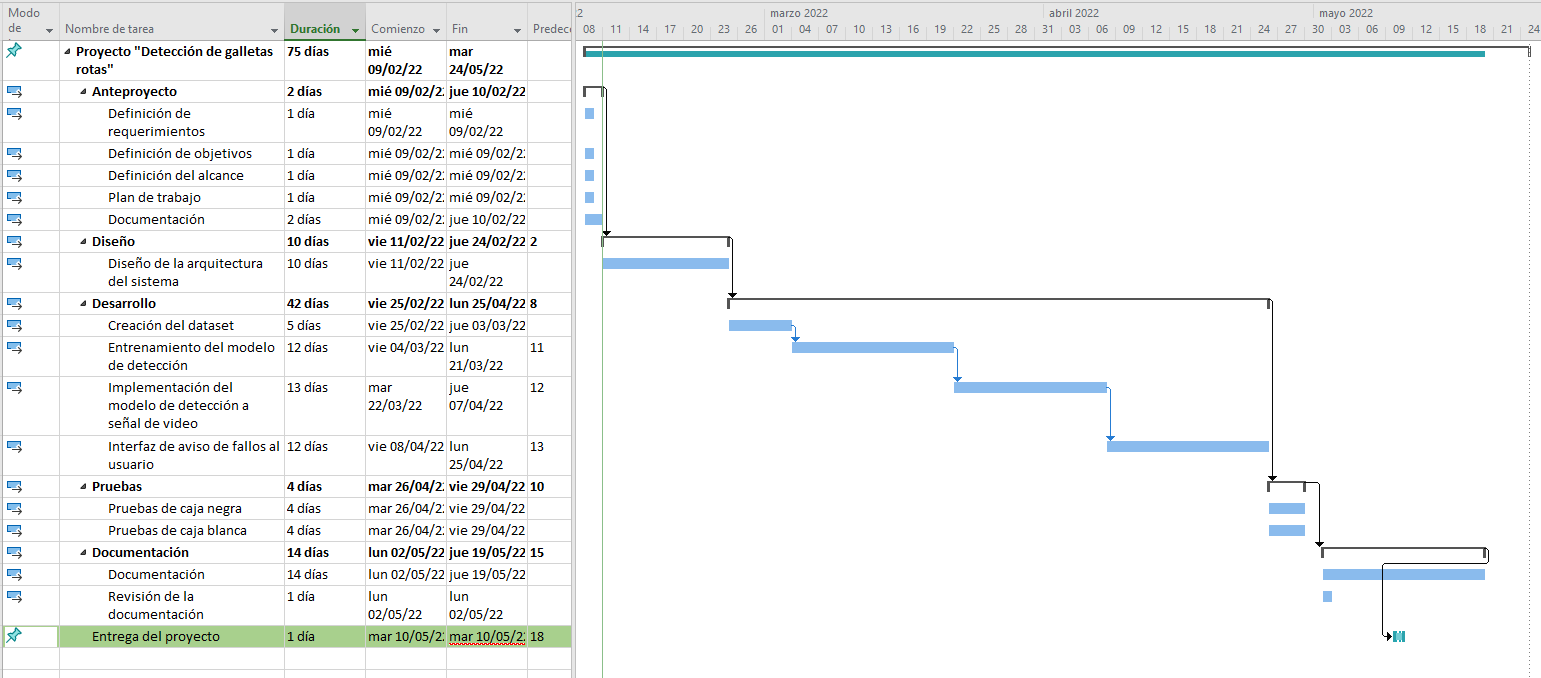


Ilustración . Línea de tiempo de desarrollo del proyecto

## Descripción de la solución, metodologías y herramientas empleadas

Para el desarrollo del proyecto se ha utilizado el lenguaje de programación Python, que cuenta con múltiples herramientas y librerías disponibles para inteligencia artificial. Este lenguaje se ha utilizado con el IDE de programación PyCharm.

Para la primera fase del proyecto, la elaboración del dataset de galletas, se ha usado el software de etiquetado “LabelImg”, que hace uso de Python para delimitar las imágenes con sus cajas de referencia y etiquetar las imágenes en el caso de que se trate de un defecto o no.

También se ha hecho uso de la librería de Python “Detecto”, que es una librería de detección de objetos implementada sobre PyTorch que se utiliza para entrenar un modelo de detección con el algoritmo Faster R-CNN (Region Based – Convolutional Neuronal Network)

Se utiliza la librería OpenCV para obtener las imágenes y procesarlas para que el sistema pueda hacer uso del modelo de detección.

Por último, se usa la librería Flask de Python para montar un servidor local que permita el uso de las distintas librerías implementadas y ayudándose de la webcam, capture las imágenes que se tratan para detectar los defectos.

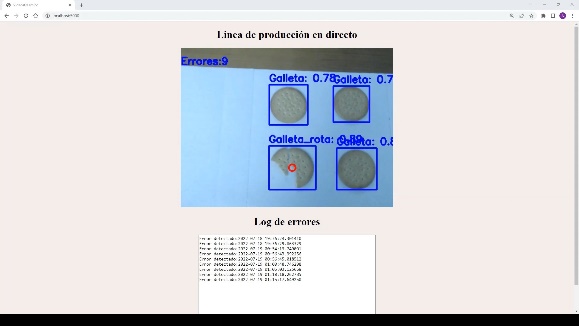


Ilustración . Proceso de detección de defectos 1

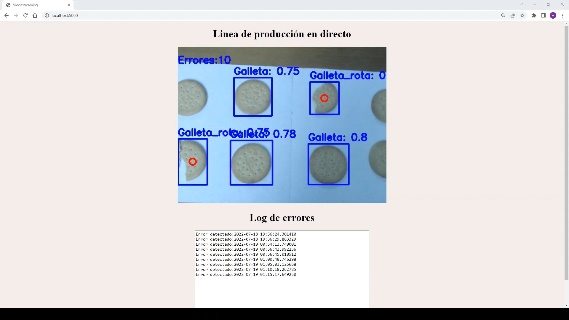


Ilustración . Proceso de detección de defectos 2

## Recursos requeridos

Para la realización de este proyecto ha sido necesario contar con un ordenador portátil, que ya había sido previamente adquirido, pero se estima su coste en el presupuesto. Posteriormente se adquirió una webcam para obtener imágenes de mayor calidad para su procesamiento.

Para el desarrollo del código se utilizó el IDE PyCharm, así como un conjunto de librerías libres, como Detecto, Flask y OpenCV.

Los materiales para las pruebas que se utilizan son galletas y una base para tener contraste con las galletas.



Ilustración . Galletas de ejemplo utilizadas en el proyecto

## Presupuesto

|  |  |  |  |
| --- | --- | --- | --- |
| **Tipo de coste** | | **Valor** | **Comentarios** |
| Horas de trabajo en el proyecto | | 450 horas |  |
| Equipo técnico utilizado | Ordenador portátil | 1.100 € | Ya contaba con el ordenador. |
| Webcam | 60 € | Se compró una webcam para que la calidad de imagen fuera mejor y la detección fuera más precisa |
| Software utilizado | IDE PyCharm | 0 € | IDE de desarrollo utilizado |
| LabelImg | 0 € | Software de etiquedado para la elaboración del dataset |
| Detecto | 0 € | Librería de detección de imágenes para generar el modelo |
| Flask | 0 € | Librería para montar un servidor local en Python con el que probar el sistema |
| OpenCV | 0 € | Librería que se utiliza para el procesamiento de imágenes |
| Materiales empleados | Paquete de galletas | 1 € | Galletas que se utilizan de ejemplo para la detección de errores |
| Cinta transportadora | 5 € | Maqueta casera de una cinta transportadora para probar el sistema con una representación de una línea de producción |

Tabla 1. Presupuesto del proyecto

Las distintas licencias que se utilizan son:

|  |  |  |  |
| --- | --- | --- | --- |
| **Software utilizado** | **Licencia** | **Permisos** | **Limitaciones** |
| PyCharm | Apache License 2.0 | Commercial Use | Trademark Use |
| Modification | Liability |
| Distribution | Warranty |
| Patent Use |  |
| Private Use |  |
| LabelImg | MIT License | Commercial Use | Liability |
| Modification | Warranty |
| Distribution |  |
| Private Use |  |
| Detecto | MIT License | Commercial Use | Liability |
| Modification | Warranty |
| Distribution |  |
| Private Use |  |
| Flask | BSD 3-Clause "New" or "Revised" License | Commercial Use | Liability |
| Modification | Warranty |
| Distribution |  |
| Private Use |  |
| OpenCV | Apache License 2.0 | Commercial Use | Trademark Use |
| Modification | Liability |
| Distribution | Warranty |
| Patent Use |  |
| Private Use |  |

Tabla 2. Licencias del proyecto

## Viabilidad

El análisis de viabilidad de este proyecto se realiza en relación a las soluciones que se comercializan actualmente. Se va a realizar una comparación respecto del servicio Amazon Lookout. Amazon Lookout cobra por horas de uso, tanto para el entrenamiento del modelo de detección como para la inferencia de las imágenes.

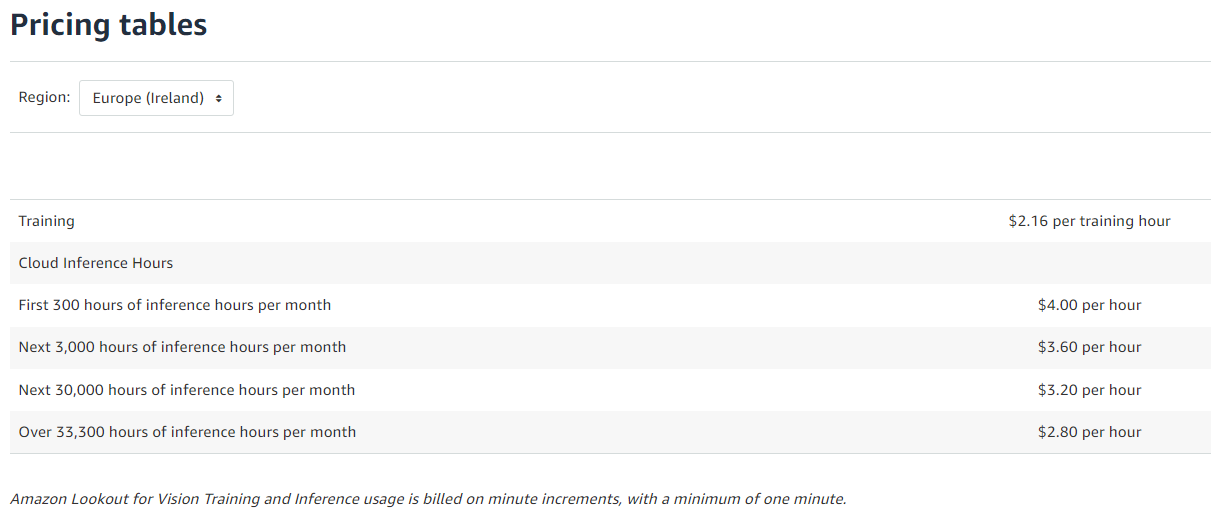


Ilustración . Tarifas del servicio Amazon Lookout

Esto hace que en un sistema de producción continua el coste se dispare rápidamente.

En consecuencia, el sistema que se plantea tendría un coste inicial más alto, pero más sostenible a lo largo del tiempo. Un usuario podría realizar una implementación rápida invirtiendo en una SBC o miniPC y una webcam para sus necesidades iniciales, una implementación Edge computing. Posteriormente si quisiera escalar la solución podría optar por una aproximación mediante cloud computing en la que el procesamiento se realizaría en un servidor que recibiría las imágenes por unas webcams conectadas a la red. Por tanto, se considera que esta implementación es potencialmente viable dado que su coste es menor a lo largo del tiempo que otras soluciones comerciales.

## Resultados del proyecto

Los resultados del proyecto obtenidos se consideran satisfactorios en relación a los objetivos planteados. Se detallan los siguientes objetivos y los resultados obtenidos:

* Creación de un dataset de galletas sin defectos y con defectos para su detección mediante un modelo predictivo

Se ha realizado un dataset de galletas sin defectos y con defectos con la herramienta de etiquetado LabelImg.

* Creación y entrenamiento de un modelo predictivo con redes neuronales convolucionales mediante el uso de inteligencia artificial para distinguir entre galletas sin defectos y con defectos.

Se ha entrenado un modelo de detección con la librería Detecto que permite detectar defectos en las galletas.

* Desarrollo de un sistema que haga uso de visión artificial y que, mediante un modelo predictivo entrenado con IA, detecte galletas con defectos.

Se ha desarrollado un sistema que permite el despliegue de un servidor que, conectado a una webcam, hace uso del modelo de detección entrenado para distinguir los defectos de las galletas en la línea de producción.

# DISCUSIÓN

En este capítulo se van a discutir aspectos del proyecto como la metodología utilizada, el código utilizado y los problemas que han surgido.

Respecto a la metodología utilizada, se planteó el uso de la metodología SCRUM para realizar avances en el proyecto mediante etapas y consultar los avances con el tutor. El desarrollo no se ha ajustado al calendario planteado, pero posteriormente se ha ido adaptando a las necesidades del proyecto. Se han realizado reuniones semanales con el tutor para comprobar el avance del proyecto, plantear dudas y realizar cambios y mejoras.

El código desarrollado es el siguiente:

Fichero main.py

En este fichero se encuentra el código que contiene las funciones para entrenar el modelo de detección haciendo uso del dataset generado. También hay unas funciones accesorias para hacer uso del modelo en pruebas con videos, así como una reimplementación de una función de detecto para adaptarla a las necesidades del proyecto. Puede consultarse en el anexo.

Fichero server.py

En este fichero se incluyen las funciones que permiten levantar un servidor local para captar las imágenes de la webcam y poder acceder de forma remota al sistema. También se encuentra la función que permite dibujar las cajas de detección de las galletas sobre las imágenes obtenidas y comprobar si hay un defecto o no. Puede consultarse en el anexo

# CONCLUSIONES

## Conclusiones del trabajo

Mediante el desarrollo de este proyecto se han conseguido alcanzar los objetivos planteados. Se ha conseguido realizar un dataset de galletas con y sin defectos, se ha realizado un sistema que permite el entrenamiento de un modelo de detección imágenes y por último se ha realizado la implementación de un sistema mediante un servidor web que permite la detección de imágenes de galletas con defectos mediante el uso del modelo de detección generado.

## Conclusiones personales

Tras la realización de este proyecto puedo concluir que he aprendido un poco más de aspectos de la inteligencia artificial, como es la visión artificial. Creo que es un aprendizaje valioso para mi futuro por la importancia que tiene la inteligencia artificial y la demanda que tiene este tipo de tecnología actualmente en el mercado laboral.

También he aprendido a gestionar mejor los tiempos en un proyecto de desarrollo, ver los problemas que surgen y como solucionarlos y como adaptarme a fallos en la planificación que yo mismo realicé.

Creo que es de gran importancia la realización de esta clase de proyectos para prepararse a la entrada del mercado laboral y ver cómo se gestiona un proyecto de desarrollo.

# FUTURAS LÍNEAS DE TRABAJO

Debido a algunas limitaciones que se plantearon al inicio del proyecto, se han considerado algunas futuras líneas de trabajo, son las siguientes:

* Desplegar y comprobar el funcionamiento real del sistema en una línea de producción. Dado que no se contó con tiempo suficiente para contactar con una empresa que se prestase a permitir pruebas en producción, sería interesante ver como se comporta el sistema en una línea de producción real, con la velocidad que se le exigiría.
* Realizar una implementación propia de PyTorch. Durante el desarrollo del proyecto se optó por utilizar la librería de detección Detecto, que era a su vez una implementación de la librería PyTorch, la cual es más potente. Sería interesante de cara al futuro del sistema implementar una librería propia con funciones adaptadas al sistema que permitan una mejor configuración de los requisitos, como la precisión, el entrenamiento, etc.
* Plantear el despliegue del sistema mediante una solución cloud. Realizar un plan de despliegue e implementación mediante webcams conectadas a la red que envíen las imágenes a un servidor y que este procese todas las imágenes para reducir costes al no tener que adquirir una SBC para desplegar el sistema.

# Bibliografía y Referencias

**Amazon Lookout for vision.** [En línea] https://aws.amazon.com/es/lookout-for-vision/.

**Gandhi, Rohith. 2018.** Towards Data Science. [En línea] 9 de Julio de 2018. https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

**Micro.Ai. 2020.** [En línea] 28 de Julio de 2020. https://www.micro.ai/resources/case-studies/ai-enabled-defect-detection-solutions-in-manufacturing.

# ANEXOS

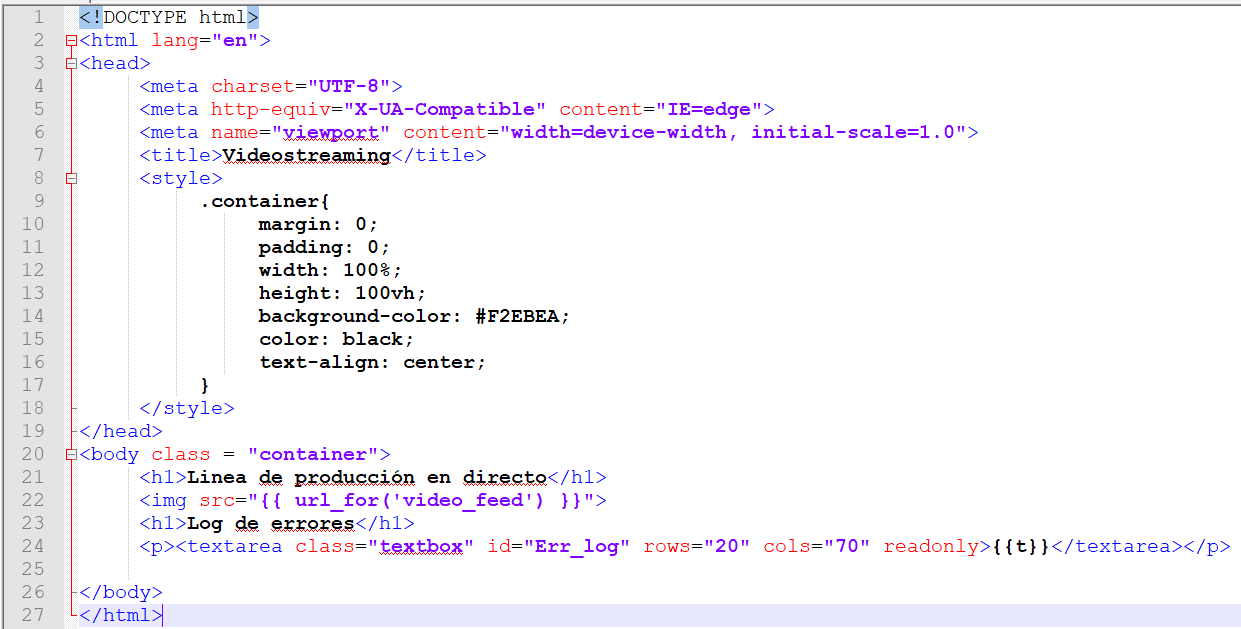
Fichero main.py

from detecto.core import Dataset  
import cv2  
from detecto import utils  
from detecto.core import Model  
from torchvision import transforms  
from detecto.visualize import detect\_video  
import matplotlib.pyplot as plt  
import click  
  
@click.group()  
def cli():  
 pass  
  
#Funcion sobreescrita para probar sobre detecto  
def detect\_live2(model,score\_filter=0.6):  
 errores = 0  
 cv2.namedWindow('Detecto')  
 try:  
 video = cv2.VideoCapture(0)  
 width = video.get(cv2.CAP\_PROP\_FRAME\_WIDTH) # float `width`  
 height = video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT) # float `height`  
 print(width)  
 print(height)  
 except:  
 print('No webcam available.')  
 return  
  
 while True:  
 ret, frame = video.read()  
 if not ret:  
 break  
 cv2.putText(frame, 'Errores:{}'.format(errores),(0, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 3)  
 labels, boxes, scores = model.predict(frame)  
  
 # Plot each box with its label and score  
 for i in range(boxes.shape[0]):  
 if scores[i] < score\_filter:  
 continue  
  
 box = boxes[i]  
 cv2.rectangle(frame, (int(box[0]), int(box[1])), (int(box[2]), int(box[3])), (255, 0, 0), 3)  
 if labels:  
 fail\_detected=False  
 if 'Galleta\_rota' in labels:  
 fail\_box\_min=int(box[0])  
 fail\_box\_max=int(box[2])  
 fail\_box\_center= int((fail\_box\_min + fail\_box\_max)/2)  
 print(fail\_box\_center)  
 if fail\_box\_center <325 and fail\_box\_center>315 and not fail\_detected:  
 errores=errores+1  
 fail\_detected=True  
 else:  
 fail\_detected=False  
 cv2.putText(frame, '{}: {}'.format(labels[i], round(scores[i].item(), 2)), (int(box[0]), int(box[1] - 10)),  
 cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 3)  
  
 cv2.imshow('Detecto', frame)  
  
 # If the 'q' or ESC key is pressed, break from the loop  
 key = cv2.waitKey(1) & 0xFF  
 if key == ord('q') or key == 27:  
 break  
  
 cv2.destroyWindow('Detecto')  
 video.release()  
  
detect\_live = detect\_live2  
  
@cli.command(help="Funcion que entrena y genera un modelo de deteccion de galletas")  
def train\_model():  
  
 print("Entrenando modelo")  
  
 custom\_transforms = transforms.Compose([  
 transforms.ToPILImage(),  
 transforms.Resize(800),  
 transforms.ColorJitter(saturation=0.3),  
 transforms.ToTensor(),  
 utils.normalize\_transform(),  
 ])  
  
 dataset = Dataset('galletas proyecto',transform=custom\_transforms)  
 labels = ['Galleta', 'Galleta\_rota']  
 model = Model(labels)  
 model.fit(dataset, verbose=True)  
 #val\_dataset = Dataset('val\_dataset')  
 #losses = model.fit(dataset, val\_dataset, epochs=15, learning\_rate=0.01,  
 # gamma=0.2, lr\_step\_size=5, verbose=True)  
 #plt.plot(losses)  
 #plt.show()  
 model.save("modelo.pth")  
 print('Modelo creado y guardado')  
  
@cli.command(help='Funcion que utiliza un modelo generado para probar con un video')  
def use\_model():  
 print('Usando modelo para probar')  
 model = Model.load('modelo.pth', ['Galleta', 'Galleta\_rota'])  
 print("Analizando video")  
 detect\_video(model, 'galletas2.mp4', 'galletas\_analizadas2.avi')  
 print("Final del analisis")  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 cli()

Fichero server.py

from flask import Flask  
from flask import render\_template  
from flask import Response  
from detecto.core import Model  
from datetime import datetime  
import os.path  
  
import cv2  
app = Flask(\_\_name\_\_)  
cap = cv2.VideoCapture(0, cv2.CAP\_DSHOW)  
  
#Función que se encarga de representar las cajas de detección en una imagen  
#que se envía al servidor web para mostrarse en tiempo real  
def visualizar():  
 #Se fija un límite de puntuación para la detección  
 score\_filter=0.6  
 #Se carga el modelo  
 model = Model.load('modelo.pth', ['Galleta', 'Galleta\_rota'])  
 #Se carga el fichero de errores para poner el contador de errores  
 if os.path.exists("errors.txt") == True:  
 with open("errors.txt","r") as f:  
 errores = len(f.readlines())  
 else:  
 errores=0  
  
 while True:  
 ret, frame = cap.read()  
 if ret:  
 #Se procesa la imagen para obtener las coordenadas de las cajas de detección de los objetos  
 labels, boxes, scores = model.predict(frame)  
 cv2.putText(frame, 'Errores:{}'.format(errores), (0, 50), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 3)  
 for i in range(boxes.shape[0]):  
 if scores[i] < score\_filter:  
 continue  
 box = boxes[i]  
 #Se dibuja un rectángulo que contiene al objeto que se ha detectado  
 cv2.rectangle(frame, (int(box[0]), int(box[1])), (int(box[2]), int(box[3])), (255, 0, 0), 3)  
 label = labels[i]  
 #Se comprueba si se ha detectado un objeto  
 if label:  
 fail\_detected = False  
 #Si el objeto detectado es una galleta rota se calcula su centro para seguirla  
 if label == "Galleta\_rota":  
 fail\_box\_x\_min = int(box[0])  
 fail\_box\_x\_max = int(box[2])  
 fail\_box\_center\_x = int((fail\_box\_x\_min + fail\_box\_x\_max) / 2)  
 fail\_box\_y\_min = int(box[1])  
 fail\_box\_y\_max = int(box[3])  
 fail\_box\_center\_y = int((fail\_box\_y\_min + fail\_box\_y\_max) / 2)  
 radius = 10  
 cv2.circle(frame, (fail\_box\_center\_x, fail\_box\_center\_y), radius, (0, 0, 255), 3)  
  
 #Se coloca una línea imaginaria en el centro de la pantalla para comprobar que galletas la cruzan  
 #y en el caso de que sea un agalleta con defectos, aumentar el contador de errores  
 if fail\_box\_center\_x < 325 and fail\_box\_center\_x > 315 and not fail\_detected:  
 fail\_detected = True  
  
 #Se obtiene la fecha y hora actuales y se guardan en el fichero de errores  
 now = datetime.now()  
 error\_text="Error detectado:{}\n".format(now)  
 with open('errors.txt','a') as f:  
 f.write(error\_text)  
 errores=errores+1  
 else:  
 fail\_detected = False  
 #Se coloca la etiqueta del objeto detectado y la precisión con la que se ha detectado  
 cv2.putText(frame, '{}: {}'.format(label, round(scores[i].item(), 2)), (int(box[0]), int(box[1] - 10)),  
 cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 3)  
 (flag, encodedImage) = cv2.imencode(".jpg", frame)  
 if not flag:  
 continue  
 yield(b'--frame\r\n' b'Content-Type: image/jpeg\r\n\r\n' +bytearray(encodedImage) + b'\r\n')  
  
#Funcion que carga el index.html además de poblar el area de texto con los errores del fichero errors  
@app.route("/")  
def index():  
 with open('errors.txt','r') as f:  
 t = f.read()  
 return render\_template("index.html",t=t)  
  
#Función que se encarga de mostrar el cuadro con las imágenes en la plantilla del index.html  
@app.route("/video\_feed")  
def video\_feed():  
 return Response(visualizar(),  
 mimetype = "multipart/x-mixed-replace; boundary=frame")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app.run(debug=False)  
  
cap.release()

Fichero index.html



[PÁGINA INTENCIONADAMENTE EN BLANCO]