



## **GRADO EN INGENIERÍA DE ROBÓTICA SOFTWARE**

Escuela Técnica Superior de Ingeniería de Telecomunicación

Curso académico 2021-2022

### **Trabajo fin de grado**

Sistema de Monitorización de animales de laboratorio  
bajo plataforma de bajo coste

**Tutor:** Julio Vega Pérez

**Autor:** Isabel Cebollada Gracia



Este trabajo se distribuye bajo los términos de la licencia internacional CC BY-NC-SA International License (Creative Commons AttributionNonCommercial-ShareAlike 4.0). Usted es libre de *(a) compartir*: copiar y redistribuir el material en cualquier medio o formato; y *(b) adaptar*: remezclar, transformar y crear a partir del material. El licenciador no puede revocar estas libertades mientras cumpla con los términos de la licencia:

- *Atribución.* Usted debe dar crédito de manera adecuada, brindar un enlace a la licencia, e indicar si se han realizado cambios. Puede hacerlo en cualquier forma razonable, pero no de forma tal que sugiera que usted o su uso tienen el apoyo de la licenciatante.
- *No comercial.* Usted no puede hacer uso del material con propósitos comerciales.
- *Compartir igual.* Si remezcla, transforma o crea a partir del material, debe distribuir su contribución bajo la misma licencia del original.

# Agradecimientos

---

Unas bonitas palabras...

Quizás un segundo párrafo esté bien. No te olvides de nadie.

Un tercero tampoco viene mal para contar alguna anécdota...

¿Alguien más? Aunque sean *actores* secundarios.

Un quinto párrafo como colofón.

*A alguien especial;  
si no, tampoco pasa nada*

Madrid, xx de xxxxxx de 20xx

*Tu nombre*

# Resumen

---

Desde hace siglos, la experimentación con animales se ha llevado a cabo para saber qué les sucede a los humanos y al mundo que les rodea. A día de hoy, aunque de una manera distinta debido tanto al avance de la humanidad como de la tecnología, esta experimentación sigue desarrollándose en diferentes ámbitos; para el estudio de los comportamientos de animales o el uso de éstos como modelo de investigación en diferentes campos, desde el testado de productos hasta la sanidad humana.

Los estudios con animales en la medicina han tenido gran importancia en el desarrollo de vacunas modernas como la tuberculosis o la meningitis ya que los animales, más concretamente los ratones, sufren enfermedades similares a los humanos. De esta manera, los ratones se han convertido en uno de los principales animales para estudiar su comportamiento y el efecto que éste puede tener en la detección de enfermedades neurológicas, como el autismo, el parkinson o el alzheimer.

La observación de estos animales en todo momento es una necesidad para analizar y estudiar su comportamiento, así como tener la información de las condiciones del entorno a las que se encuentran, haciendo necesario que haya trabajadores analizando las condiciones y grabaciones que tienen que hacerse a los ratones diariamente. Para evitar esto, el fin de este trabajo es crear un sistema dotando con los sensores necesarios al entorno de los roedores para obtener la información de forma automatizada y traducirlo en una interfaz comprensible para cualquier usuario, así como el control por vídeo a través de algoritmos de visión artificial para detectar los movimientos de los animales bajo una plataforma económicamente accesible para todo el mundo.

Durante todo el desarrollo del trabajo han surgido problemas en conexiones, seguridad, instalación, en el uso de algunos sensores y el uso de herramientas no usadas hasta el momento. Tras el estudio y la comprensión de los errores encontrados, así como con la ayuda de internet y de las experiencias de otros usuarios, tanto con problemas similares como con posibles soluciones, todos los errores han podido solucionarse.

# Acrónimos

---

**IA** *Inteligencia Artificial*

**VA** *Visión Artificial*

**ML** *Machine Learning*

**IU** *Interfaz de Usuario*

**2FA** *Factor de doble Autenticación*

# Índice general

---

<b>1. Introducción</b>	<b>1</b>
1.1. La robótica . . . . .	2
1.2. Inteligencia Artificial . . . . .	3
1.2.1. Visión Artificial . . . . .	5
1.3. Machine Learning . . . . .	7
1.4. Deep Learning . . . . .	8
1.5. Sistemas multisensoriales . . . . .	10
<b>2. Objetivos</b>	<b>13</b>
2.1. Descripción del problema . . . . .	13
2.2. Requisitos . . . . .	14
2.3. Metodología . . . . .	14
<b>3. Plataforma de desarrollo</b>	<b>16</b>
3.1. Infraestructura hardware . . . . .	16
3.2. Lenguaje Python . . . . .	16
3.3. TensorFlow Lite . . . . .	17
3.4. OpenCV . . . . .	18
3.5. Flask . . . . .	19
3.6. Node-Red . . . . .	20
<b>4. Sistema multisensorial</b>	<b>23</b>
4.1. Hardware . . . . .	24
4.2. Software . . . . .	27
4.2.1. Fichero Python para lectura sensorial . . . . .	29
4.2.2. Creación de la interfaz de usuario . . . . .	30
4.2.3. Integración de las cámaras en la UI . . . . .	32
4.2.4. Seguridad . . . . .	34
4.2.5. Autoarranque . . . . .	36

<b>5. Conclusiones</b>	<b>38</b>
5.1. Conclusiones . . . . .	38
5.2. Corrector ortográfico . . . . .	39
<b>Bibliografía</b>	<b>40</b>

# Índice de figuras

---

1.2.	Detección de fracturas en radiografías mediante IA . . . . .	4
1.6.	Decreto sobre la Acrópolis de Atenas reconstruido con Ithaca. . . . .	9
1.8.	Patrón de movimientos e imagen grabada del grupo de cerdos. . . . .	11
1.9.	Esquema del desarrollo del sistema mediante sensores. . . . .	11
3.1.	Detección de ratones en una imagen usando TensorFlow Lite. . . . .	18
3.2.	Ejemplo de detección de bordes con OpenCV. . . . .	19
3.3.	Muestra de los dos servidores del trabajo usando Flask. . . . .	20
3.4.	Flujo del proyecto en Node-Red. . . . .	21
3.5.	Muestra del uso de Node-Red en el trabajo. . . . .	22
4.1.	Raspberry Pi 4B . . . . .	24
4.2.	Sensor BME680. . . . .	25
4.3.	Sensor AMG8833. . . . .	25
4.4.	Sensor DS18B20. . . . .	26
4.5.	Sensor MQ-135. . . . .	26
4.6.	Sensor de nivel de agua. . . . .	27
4.7.	Cámara térmica. . . . .	27
4.8.	Cámara de Raspberry. . . . .	28
4.9.	Esquema de conexiones. . . . .	28
4.10.	Esquema de clases de cada sensor. . . . .	30
4.11.	IU sin los sensores cámaras. . . . .	32
4.12.	Incorporación de la fecha y hora en la visualización de la PiCam. . . . .	33
4.13.	Diagrama de casos de uso del proyecto. . . . .	34
4.14.	Visualización de la interfaz de usuario. . . . .	35
4.15.	Acceso al flujo de nodos en Node-Red. . . . .	36
4.16.	Acceso a la interfaz de usuario desde distintos dispositivos. . . . .	37

# Listado de códigos

# Listado de ecuaciones

---

# Índice de cuadros

---

---

# **Capítulo 1**

## **Introducción**

---

El desarrollo tecnológico ha provocado el avance en la robótica, un campo de investigación muy amplio en la actualidad que está facilitando la vida a los humanos.

Cualquier robot está formado por tres componentes principales; un sistema de control, sensores y actuadores. Dentro de los numerosos sensores que pueden tener los robots, uno de los que está adquiriendo mayor relevancia en los últimos años es el sensor de visión.

Los sensores de visión requieren un proceso previo a su funcionamiento para obtener los resultados deseados. Este proceso está basado en algoritmos de entrenamiento.

Los algoritmos más utilizados hoy en día se obtienen bajo la técnica del Machine Learning (ML) —o aprendizaje automático—, cuyo objetivo es que las máquinas sean capaces de aprender sin tener que ser programadas explícitamente. Está formado por diferentes subcampos, entre ellos el Deep Learning —o aprendizaje profundo—, que utiliza algoritmos de ML basándose en estructuras similares al modelo del cerebro humano. El Deep Learning es mucho más complejo que el ML, y es uno de los más utilizados para el desarrollo de la visión artificial(VA).

En este capítulo se explican en profundidad los términos presentados en el párrafo anterior. Asimismo se presentan sistemas actuales que existen en el mercado con una idea similar a la de este trabajo.

## 1.1. La robótica

La robótica, aunque no siempre bajo ese nombre, se lleva utilizando desde hace mucho tiempo con el propósito de desarrollar herramientas automatizadas. Hasta mediados del siglo pasado, esta disciplina se centraba en el desarrollo de máquinas que realizaban un trabajo que resultaba tedioso, debido a su repetitividad, para el ser humano. Esta concepción se refleja con Henry Ford y la producción en serie en su fábrica de coches inaugurada en 1901.

Sin embargo, la palabra *robota*, origen de la palabra robot que significa trabajo forzado aparece por primera vez en 1921 en una obra de teatro de un autor checo llamado Karel Capek.

Empezando a concebir esta palabra a principios de siglo pasado, comenzó a nacer el desarrollo de la idea de robótica que existe hoy en día y que no se centra exclusivamente en el desarrollo de herramientas automatizadas, sino que se concibe como una industria interdisciplinaria que surge de la intersección de la ciencia, la ingeniería y la tecnología.

Esta nueva concepción une el conocimiento científico, computacional e informático con diversas ramas de la ingeniería, ya que no solo implica el estudio de robots, sino también de su diseño, programación y aplicación. Así, la robótica incluye disciplinas como la IA, la informática, la VA, la programación o el álgebra entre otras muchas.

En los últimos veinte años, la robótica ha adquirido un gran peso y un desarrollo tal que las películas futuristas que surgían hace unos años se han hecho realidad en el presente, aunque todavía quede mucho camino por continuar en esta amplia disciplina.

Tras la primera aparición de la palabra *robota*, la definición de robot ha ido formándose hasta la que hay en la actualidad: cualquier máquina que opera de forma automática y autónoma, y que sustituye a los seres humanos en determinadas tareas, especialmente las peligrosas, aburridas o pesadas. Está compuesto por el sistema de control, los sensores y los actuadores.

La analogía de un robot respecto a un ser humano es la siguiente: los sensores son sus sentidos, que transmiten la información percibida, bien sea del propio robot o del entorno al sistema de control, que se asemeja al cerebro humano. El sistema de control toma decisiones adaptándose a la información recibida, que modifica bien el entorno o bien la manera de actuar internamente del propio robot. En el caso de la modificación del entorno con sus actuadores, la similitud en el humano son las extremidades, mientras

que si la modificación es interna, equivaldría a las reacciones del cuerpo humano frente a una herida o una infección, por ejemplo.

El objetivo de la robótica es ayudar al ser humano en todos los ámbitos, tanto en la vida cotidiana como en la vida profesional, evitando los trabajos perjudiciales, e incluso haciendo trabajos que el ser humano no sería capaz de hacer por sí mismo en distintos sectores, como —por ejemplo— la medicina. En este ámbito cabe resaltar el robot Da Vinci (Figura 1.1-a), uno de sus robots más famosos que permite a un cirujano comandar órdenes a través de una consola, permitiéndole realizar cirugías de alta precisión eliminando los temblores nerviosos que un humano pueda tener y permitiendo al cirujano una mayor visión.

Otro de los campos más importantes es la robótica espacial, cuyo objetivo primordial es explorar sitios inaccesibles u hostiles para el ser humano. Un ejemplo de estos robots es el Curiosity (Figura 1.1-b), cuya misión era evaluar la habitabilidad en Marte recogiendo información del entorno.

Otro ejemplo de entornos hostiles se encuentra en el accidente nuclear que ocurrió en Fukushima en 2011 causado por un tsunami y un terremoto. Gracias a los robots (Figura 1.1-c), se pudo acceder a la zona para poder limpiarla. La industria del automóvil, con los coches autónomos es otro de los grandes frentes de la robótica. El objetivo es que un coche sea capaz de realizar todas las tareas de conducción de forma autónoma.



(a) Robot Da Vinci



(b) Robot Curiosity



(c) Fukushima

Figura 1.1: Ejemplos de la robótica en diferentes campos.<sup>1</sup>

## 1.2. Inteligencia Artificial

La robótica engloba un conjunto de diferentes disciplinas que permiten abordar todos los campos que esta comprende. Uno de los campos más importantes es la IA,

---

<sup>1</sup>Imágenes obtenidas de <https://images.google.com>

que consiste en replicar los mecanismos del cerebro humano mediante algoritmos que emplean unidades equivalentes a las neuronas. Estos algoritmos van aprendiendo a medida que realizan sus tareas en base a la experiencia que van adquiriendo durante su ejecución.

Numerosos son los estudios existentes en la actualidad sobre la IA. Uno de ellos es la mejora del rendimiento y la eficiencia del reconocimiento de fracturas radiográficas a través de la IA [Guermazi et al., 2021]. Este algoritmo es capaz de detectar las fracturas en las radiografías con un menor rango de fallo y en menor tiempo que si es el humano quien realiza el trabajo. En el caso de la figura 1.2-a, nueve especialistas no detectaron la fractura,

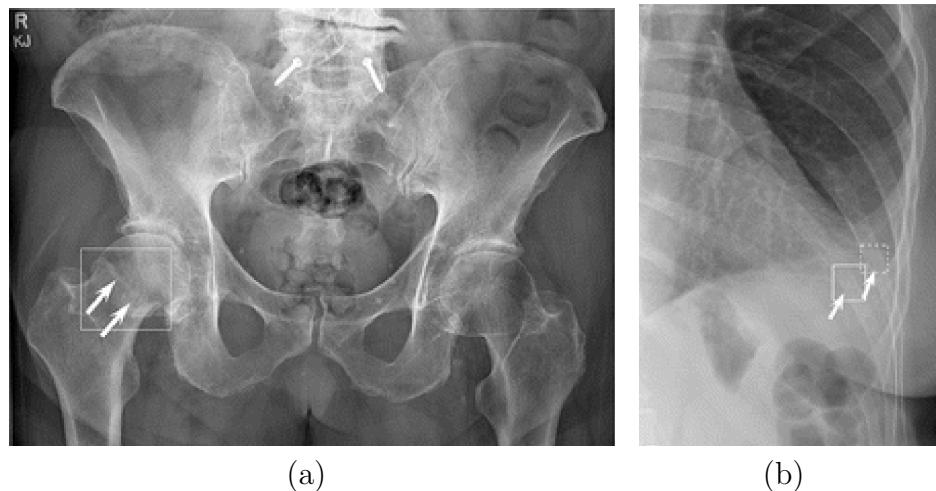


Figura 1.2: Detección de fracturas en radiografías mediante IA

Hay dos tipos de inteligencia artificial:

- IA fuerte, compuesta por la AI general y la superinteligencia artificial. La AI general está inspirada en la teoría de que una máquina tenga inteligencia humana, siendo capaz de resolver cualquier problema por sí misma. Por otro lado, la superinteligencia artificial supera la capacidad del cerebro humano. La IA fuerte es todavía teórica y no hay ejemplos reales de su uso.
- IA débil, entrenada para realizar tareas específicas que operan dentro de un rango previamente definido. Los sistemas dotados con esta inteligencia parecen inteligentes, pero están limitados al contexto en el que trabajan. Un ejemplo es el asistente de voz Siri de Apple.

La IA abarca diferentes áreas, como la comprensión, el reconocimiento o el aprendizaje. Asimismo, una de las líneas de investigación dentro de la IA es la VA, que se detalla a continuación.

### 1.2.1. Visión Artificial

La visión artificial es uno de los campos más importantes de la IA cuyo objetivo es que un sistema inteligente obtenga la información más relevante de las imágenes que percibe, y en base a ello lo aprenda con las acciones más oportunas.

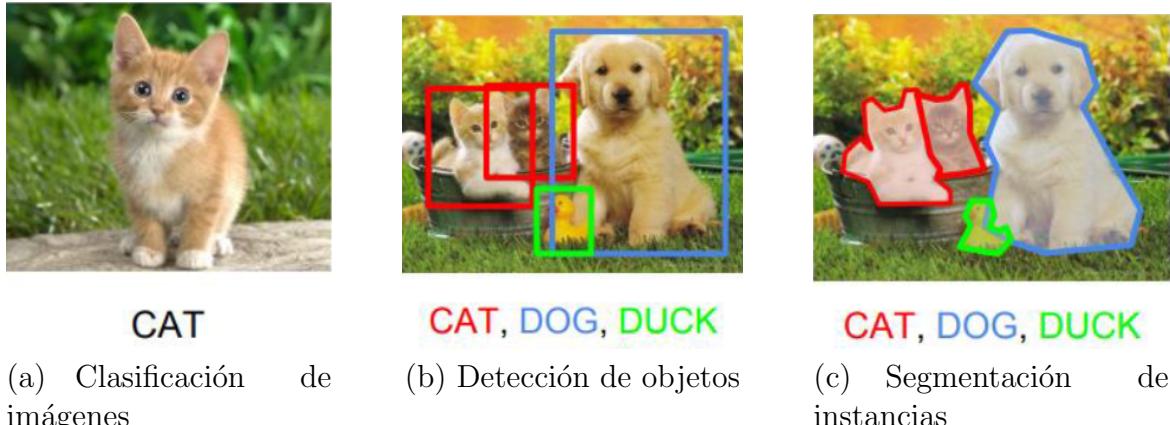
Así como la IA trata de emular un cerebro humano, la VA hace lo mismo con respecto a la visión humana, con la diferencia de que esta segunda cuenta con las experiencias aprendidas para, entre otras cosas, diferenciar los elementos que le rodean, su movimiento, distancia o tamaño. Así pues, la visión artificial trata de asemejar el funcionamiento del ojo humano y su posterior procesamiento con el cerebro mediante una cámara y el posterior procesamiento de los datos que esta vierte.

Tiene diferentes usos, entre ellos los más fundamentales son:

- *Clasificación de imágenes (Figura 1.3-a)*. Consiste en asignar imágenes a una serie de categorías predefinidas a través de un algoritmo. Es necesario contar con una gran cantidad de datos para poder tener suficiente entrenamiento para fallar lo mínimo posible.
- *Detección de objetos (Figura 1.3-b)*. Consiste en analizar partes de la imagen para localizar objetos señalándolos mediante cuadros limitadores. Esto se consigue mediante el entrenamiento de una gran cantidad de imágenes en las que se etiquetan diferentes tipos de objetos, obteniendo así los datos que se usarán para la detección en una imagen.
- *Segmentación de imágenes (Figura 1.3-c)*. Consiste en el enmascaramiento exacto de los píxeles que representan a los objetos en una imagen. Es un paso más en la detección de objetos, siendo capaz de separar el objeto concreto del resto de la imagen.

---

<sup>2</sup>Imágenes obtenidas de <https://images.google.com>

Figura 1.3: Algunos usos en la visión artificial.<sup>2</sup>

Un sistema dotado de visión funciona siguiendo tres niveles de operación. En primer lugar, la obtención de imágenes o vídeos mediante una cámara. Estas imágenes son transferidas al sistema. En segundo lugar, el procesamiento de estas imágenes para poder representar correctamente los datos de interés. Este proceso consiste en un trabajo automático del sistema en el que se elimina el ruido, se reescalan las imágenes o se ajusta el contraste entre otros para adaptar las imágenes. Finalmente, se realiza la comprensión de imágenes, que es el paso clave para que el sistema pueda llamarse inteligente, donde a través de un modelo de aprendizaje, es capaz de realizar el objetivo de interés, como puede ser detectar un determinado objeto utilizando datos aprendidos en el pasado.

Un ejemplo muy extendido de uso de la VA son los coches autónomos (Figura 3.1). Necesitan un funcionamiento de la VA impecable y sin fallos debido a sus graves consecuencias. Con el uso de cámaras y sensores, el ordenador del coche es capaz de detectar los objetos de las imágenes que recibe, como señales o semáforos —entre otros— y actuar en consecuencia del estado de estos. El coche es capaz de tener una vista de 360º y de reconocer todos los elementos de su entorno, para ser capaz de actuar como un humano actuaría.

Para que la VA funcione correctamente, es necesario disponer de algoritmos precisos que sean fiables. Entre los distintos métodos que existen para entrenar estos algoritmos, uno de los más utilizados es el Machine Learning o aprendizaje automático.

---

<sup>3</sup>Imagen obtenida de <https://images.google.com>



Figura 1.4: Vision de las cámaras y sensores de un Tesla autónomo.<sup>3</sup>

### 1.3. Machine Learning

Para que la IA pueda simular al cerebro humano, debe ser capaz también de aprender. Este aprendizaje se lleva a cabo a través de algoritmos. Dentro de la IA se encuentra una categoría denominada Machine Learning o aprendizaje automático, cuyo fin es que los algoritmos descubran patrones en conjuntos de datos que les hacen aprender y mejorar respecto a la experiencia anterior pudiendo así tener un aprendizaje autónomo para poder realizar una tarea sin ayuda externa.

En el sistema de aprendizaje de un algoritmo de ML hay tres partes principales. En primer lugar, un proceso de decisión, en el que una vez recibidos los datos de entrada el algoritmo genera una estimación en los datos partiendo de un patrón. Los datos de entrada pueden estar o no etiquetados. Si lo están, indican al modelo lo que debe identificar. La segunda parte es una función de error que sirve para evaluar la precisión del modelo sobre la decisión tomada. La tercera y última parte es un proceso de optimización de modelos donde las ponderaciones se ajustan en el caso de que el modelo se pueda adaptar mejor a el conjunto de datos de entrenamiento, repitiendo este proceso hasta cumplir un umbral de precisión.

Hay diferentes tipos de *Machine Learning*:

- *Aprendizaje supervisado*, donde los conjuntos de datos están etiquetados. Estos conjuntos de datos sirven para entrenar los datos que servirán como experiencia y base al algoritmo a la hora de hacer una clasificación con un nuevo dato. Por tanto, usa un conjunto de datos entrenados iniciales a la hora de recibir un nuevo dato.
- *Aprendizaje no supervisado*, donde los conjuntos de datos están sin etiquetar, es

decir, no cuenta con un conjunto entrenado previamente.

- *Aprendizaje semisupervisado*, que es el término medio entre los dos anteriores.

En este caso, en el entrenamiento se usa un conjunto de datos más pequeño que en el aprendizaje supervisado y usa un grupo más grande de datos sin etiquetar.

## 1.4. Deep Learning

De la misma forma que el ML está dentro del amplio campo de la IA, el ML también tiene diferentes campos. Entre ellos está el Deep Learning o aprendizaje profundo, que es mucho más complejo del ML. El Deep Learning sigue tratando de simular el cerebro humano —siendo siempre el objetivo— con la diferencia de que lo hace simulando unidades equivalentes a las neuronas.

Mientras el ML necesitaba un humano para definir previamente las características de un conjunto de datos para su clasificación, el Deep Learning no necesita intervención humana. Este aprendizaje profundo se asemeja mucho más al aprendizaje humano por tener un funcionamiento similar al de las neuronas. Este funcionamiento se denomina red neuronal.

Las redes neuronales (Figura 1.5) consisten en distintas capas de nodos interconectados, donde cada nodo se basa en su capa anterior para refinar y optimizar la precisión. Está compuesto por la capa visible, que es la primera capa, donde el modelo recibe los nuevos datos de entrada. Las otras son las capas ocultas o hidden layers, que son las que realizan todo el proceso de optimización hasta llegar al resultado final.

Existen multitud de estudios sobre el Deep Learning. Uno de ellos ha sido la restauración de textos antiguos con el uso de una red neuronal profunda llamada Ithaca [Assael et al., 2022].. Con ella se ha conseguido reparar textos dañados con una precisión del 62 % por sí sola. Está pensada para utilizarla con los historiadores, que han mejorado su precisión del 25 % al 72 % con esta nueva herramienta.

También existen aplicaciones de Deep Learning que se utilizan en la vida cotidiana, como los asistentes virtuales. Más conocidos bajo el nombre de Siri o Alexa —entre otros—, los asistentes virtuales funcionan con algoritmos de Deep Learning ayudando a

---

<sup>4</sup>Imagen obtenida de <https://images.google.com>

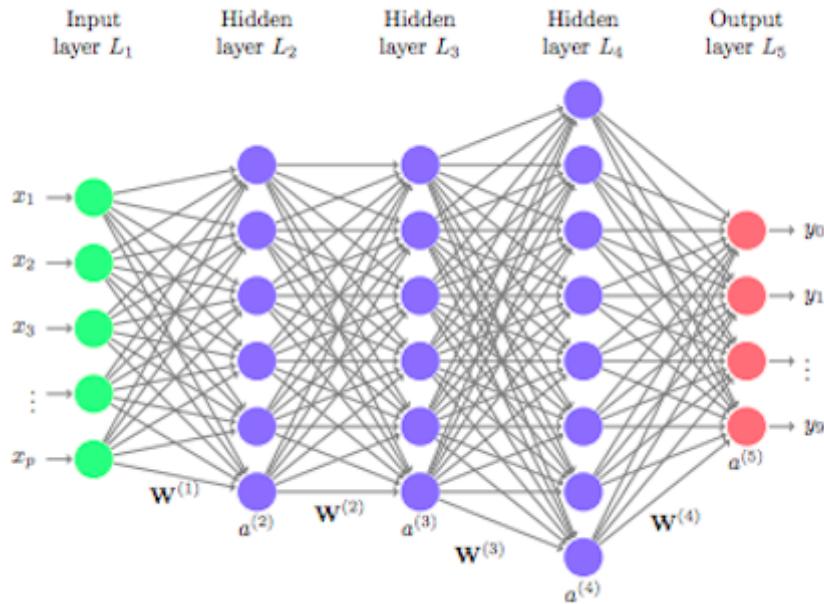


Figura 1.5: Simple ejemplo de una red neuronal en el algoritmo de Deep Learning.<sup>4</sup>



Figura 1.6: Decreto sobre la Acrópolis de Atenas reconstruido con Ithaca.

los usuarios a realizar tareas, aprendiendo continuamente de la información que reciben.

Otro ejemplo es el reconocimiento de imágenes. Entre otros ejemplos, para desbloquear algunos teléfonos móviles es posible hacerlo a través del reconocimiento facial. Este sistema funciona gracias al aprendizaje del patrón de la cara para posteriormente poder detectarla.

<sup>5</sup>Imagen obtenida de <https://images.google.com>

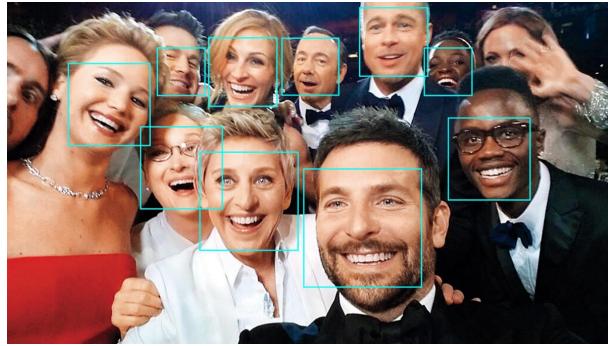


Figura 1.7: Reconocimiento facial con técnica de Deep Learning.<sup>5</sup>

## 1.5. Sistemas multisensoriales

Como se ha explicado en las secciones anteriores, el desarrollo del Deep Learning es muy importante. Lo es especialmente en el campo de la visión artificial, pues permite crear aplicaciones sofisticadas que simulan la visión humana. Si además de estos sensores de visión, a los robots les incorporan otros tipos de sensores, el resultado de la actuación del robot mejora, ya que, además de percibir la imagen del entorno está percibiendo más información que puede ser útil a la hora de tomar decisiones. Retomando la analogía con el ser humano, si un humano, además de ver, es capaz de sentir, oler o tocar, recibe más información por lo que puede conocer mejor la situación del entorno en el que se encuentra.

Algunos ejemplos de sistemas multisensoriales que se han desarrollado son:

- *Sistema de monitorización en tiempo real la salud animal*<sup>6</sup>. La Universidad Complutense de Madrid<sup>7</sup> realizó un sistema para detectar de forma temprana las enfermedades o infecciones que tienen síntomas febriles en animales de granja, más concretamente en cerdos. Este sistema se desarrolló con la instalación de microchips y otro tipo de sensores en los animales para poder controlar las condiciones como su temperatura o el consumo de agua de los bebederos, de forma que un ordenador muestra esta información de manera gráfica. También se incluye una monitorización grupal mediante grabaciones diarias para que el sistema detecte cualquier tipo de actividad no común.
- *Monitorización de rebaños de bovinos a través de redes de sensores inalámbricos* [Arce et al., 2009]. La Facultad de Zootecnia e Ingeniería de la Universidad de São

---

<sup>6</sup>[https://www.ucm.es/data/cont/docs/3-2017-05-22-2017\\_05\\_not5.pdf](https://www.ucm.es/data/cont/docs/3-2017-05-22-2017_05_not5.pdf)

<sup>7</sup><https://www.ucm.es>

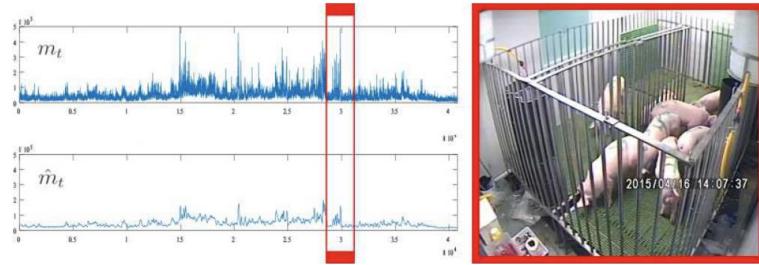


Figura 1.8: Patrón de movimientos e imagen grabada del grupo de cerdos.

Paulo<sup>8</sup> desarrolló un sistema de sensores con el fin de obtener datos fisiológicos de los animales, en este caso vacas, y obtener las condiciones en las que los animales tienen menos perturbaciones en su comportamiento natural, pues las condiciones climatológicas en cada región pueden afectar de diferentes maneras con estrés, pérdidas productivas o incluso la muerte.

Este sistema se creó con la instalación de sensores en cada una de las vacas creando una red de sensores que se comunican con una estación para obtener todos los datos del conjunto de animales.

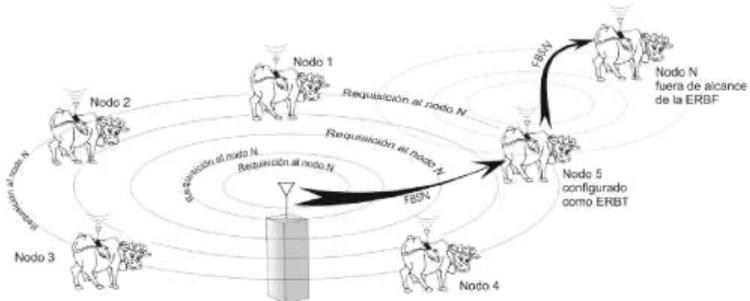


Figura 1.9: Esquema del desarrollo del sistema mediante sensores.

El comportamiento que tienen los animales puede ofrecer mucha información acerca de su salud si se mantiene una observación constante, pero también tienen una gran importancia las condiciones del entorno en la que se encuentran, ya que pueden ser decisivas para detectar el motivo de su comportamiento. Por ello, es importante tener un seguimiento constante y automatizado de estas características permitiendo al humano encargarse únicamente en controlar los datos que el sistema registra.

---

<sup>8</sup><http://www.usp.br>

En esta última sección de los sistemas multisensoriales se enmarca el trabajo realizado, englobado bajo el contexto de los términos nombrados en las diferentes secciones.

En los próximos capítulos se describe el sistema desarrollado. En el Capítulo 2 se comentan los objetivos, requisitos y metodología del trabajo. En el Capítulo 3 se explican las plataformas de desarrollo que se han utilizado para desarrollar el trabajo. En el Capítulo 4 se describe el proceso exhaustivo que ha llevado el trabajo y finalmente, en el Capítulo 5 se escriben las conclusiones.

---

# Capítulo 2

## Objetivos

---

En el capítulo anterior se ha dado el contexto del trabajo, mientras en este se presenta el plan de trabajo, definiendo los objetivos tanto generales como específicos que se han marcado para desarrollarlo y los requisitos que debe respetar el proyecto. Posteriormente se explica la metodología utilizada para cumplir con los objetivos.

### 2.1. Descripción del problema

El objetivo general de este proyecto es crear un sistema accesible económicamente para cualquier usuario que a través del uso de diferentes sensores sea capaz de medir las condiciones del entorno en la que se encuentran los animales en un laboratorio —en concreto, ratones— y mostrarlo en una interfaz gráfica en tiempo real que sea fácilmente manejable por cualquier usuario. Asimismo, que el sistema también sea capaz de controlar el tiempo en el que los animales pasan haciendo las diferentes tareas.

Actualmente el control de estas características es mediante un humano, no estando automatizado por ningún sistema. Es por ello que el objetivo de este trabajo tiene una necesidad real de la que actualmente carece el laboratorio.

Para cumplir este objetivo general, es necesario marcar una serie de objetivos específicos para el correcto desarrollo del trabajo:

- Recoger la lectura de los sensores en un mismo fichero. Cada sensor necesita las librerías pertinentes para obtener una lectura correcta, por lo que se creará una clase por sensor de manera que sea más fácil unificar todas estas lecturas. Se utilizará la librería *Thread* que permitirá ejecutar concurrentemente la lectura de todos los sensores con el uso de la función `threads.append(Thread(target=funcion))`.
- Crear un servidor web para los dos sensores que recogen imágenes. Para la

cámara térmica, la única función del servidor será mostrar la imagen que graba la cámara. Sin embargo, para la cámara normal, además de mostrar las imágenes, deberá incorporar un botón que permita iniciar y parar la grabación cuando el usuario quiera, guardando el vídeo automáticamente en el sistema. Además, la visualización desde el servidor aportará la hora y la fecha. Para llevar esto a cabo se utilizará la librería OpenCV. Con el uso de las funciones `cv2.rectangle()` y `cv2.putText()` se creará un rectángulo y se añadirá el texto deseado respectivamente.

Estos servidores de flask estarán protegidos por un factor de doble autenticación para aportar seguridad.

- Detectar los diferentes ratones mediante un algoritmo de reconocimiento de objetos a través de TensorFlowLite para poder determinar el tiempo que pasan los animales haciendo las diferentes actividades.

## 2.2. Requisitos

El trabajo cumplirá la siguiente serie de requisitos:

- El sistema sobre el que se realizará el trabajo será la Raspberry Pi 4B, haciendo de este un sistema económico.
- El lenguaje de programación será Python, debido a la familiaridad del autor con él además de la amplia variedad de librerías —útiles para este trabajo— que permite usar.
- La interfaz de usuario será creada con Node-Red, y deberá ser fácil e intuitiva.

## 2.3. Metodología

Para la satisfacción de los objetivos y el cumplimiento de los requisitos mencionados anteriormente, se han usado diferentes herramientas para el correcto control y desarrollo del proyecto.

Para el seguimiento del trabajo se han llevado a cabo reuniones semanales con el tutor del trabajo en la plataforma Microsoft Teams<sup>1</sup>, donde se compartían los problemas surgidos durante la semana junto a posibles soluciones que evaluar, además del control

---

<sup>1</sup><https://www.microsoft.com/es-ar/microsoft-teams/log-in>

y evaluación de los objetivos marcados la semana anterior. También se establecían los nuevos objetivos para la próxima semana. Asimismo, se ha utilizado el correo electrónico para problemas que surgían a lo largo de la semana.

Paralelamente se ha contactado con los investigadores del laboratorio de la Universidad de Alcalá de Henares <sup>2</sup>tanto por videoconferencia como por correo electrónico para conocer la situación y la prioridad de los problemas que les gustaría solucionar.

La evolución de todo el trabajo se ha registrado en GitHub<sup>3</sup> donde se ha ido escribiendo el progreso que se ha llevado para realizar el trabajo, junto a los problemas que han ido surgiendo acompañados de imágenes o vídeos, así como las soluciones que han servido para solventarlos. También se han aportado fotos y vídeos mostrando el funcionamiento de las distintas partes del sistema, así como el funcionamiento entero de este.

---

<sup>2</sup><https://www.uah.es/es/>

<sup>3</sup><https://github.com/jmvega/tfg-icebollada>

---

## Capítulo 3

# Plataforma de desarrollo

---

En este capítulo se definen tanto la infraestructura utilizada como los métodos, tanto a nivel software como hardware, que se han utilizado para desarrollar este trabajo.

### 3.1. Infraestructura hardware

Para este trabajo se ha utilizado la Raspberry Pi 4B con su respectivo sistema operativo Raspbian. Junto a ella, se han utilizado una serie de sensores como la PiCam —la cámara de Raspberry—, una cámara térmica, sensores de temperatura, humedad, calidad del aire o medición del nivel de agua. Para conectar de forma cómoda todos estos sensores se ha utilizado una protoboard. La protoboard es una tabla auxiliar con pines que permite conectar los sensores de una forma más visible y cómoda mediante cables, llamados cables dupont.

Para conseguir que todo funcione adecuadamente, se ha utilizado la infraestructura software descrita a continuación.

### 3.2. Lenguaje Python

Python<sup>1</sup> es un lenguaje de alto nivel de programación, interpretado y orientado a objetos cuya filosofía hace hincapié en la legibilidad de su código y por ello su sintaxis es sencilla. Utiliza módulos y librerías para las distintas aplicaciones. Algunas de ellas son: TensorFlow para aplicaciones en ML, Pandas para análisis de datos, Flask para aplicaciones web o SQLAlchemy para comunicación entre bases de datos y programas.

El uso de módulos y librerías facilita la reusabilidad de código y la programación modulada. No requiere del proceso de compilación, lo que lo convierte en un lenguaje

---

<sup>1</sup><https://www.python.org>

muy rápido en el ciclo de edición, prueba y depuración.

Actualmente es el lenguaje de programación más usado en el mundo según la calificación de la empresa TIOBE<sup>2</sup>, por lo que cuenta con una gran comunidad. Además, también es el más popular en el ámbito del Machine Learning. Algunas de las aplicaciones que usan Python son Google, Netflix, Dropbox o Spotify.

La decisión de usar Python para el desarrollo de este TFG ha sido, además de que el autor está familiarizado con el lenguaje, por las numerosas librerías útiles para este proyecto adaptadas a Raspberry que Python posee.

En este trabajo, Python se utiliza para la lectura de los sensores de forma concurrente, para la creación de los dos servidores web que tiene tanto la PiCamera como la cámara térmica con Flask y para el reconocimiento de los ratones a través de la librería TensorFlow Lite.

### 3.3. TensorFlow Lite

TensorFlow Lite<sup>3</sup> es una variación de TensorFlow más ligera adaptada a dispositivos como teléfonos móviles, microcontroladores o dispositivos embebidos como Raspberry. Es una plataforma de código abierto multiplataforma que entrena un modelo para su posterior uso. Algunos de estos usos pueden ser la clasificación o el reconocimiento de imágenes, entre otros.

Las características que hacen que pueda utilizarse en este tipo de dispositivos son las siguientes:

- Ligereza, ya que estos dispositivos tienen límite tanto en el almacenamiento como en la capacidad de cómputo
- Baja latencia, ya que las inferencias se realizan en el dispositivo y no en un servidor externo.
- Seguridad, debido a que el modelo viene implementado en el propio dispositivo.
- Consumo de energía óptimo, dado a que no es necesario que el dispositivo esté conectado a la red, que consume mucha energía.

---

<sup>2</sup><https://www.tiobe.com/tiobe-index/>

<sup>3</sup><https://www.tensorflow.org/lite/>

El uso de TensorFlow Lite ha sido, en primer lugar, para la creación y entrenamiento de un modelo capaz de detectar ratones en una imagen, basándose en un dataset de 60 imágenes. En segundo lugar, para la detección de ratones en cualquier imagen o vídeo, siendo capaz de detectar el número de ratones o el lugar en el que se encuentra cada uno. Para conseguir esta detección a tiempo real del vídeo grabado por la cámara es requerido el uso de otra librería llamada OpenCV.

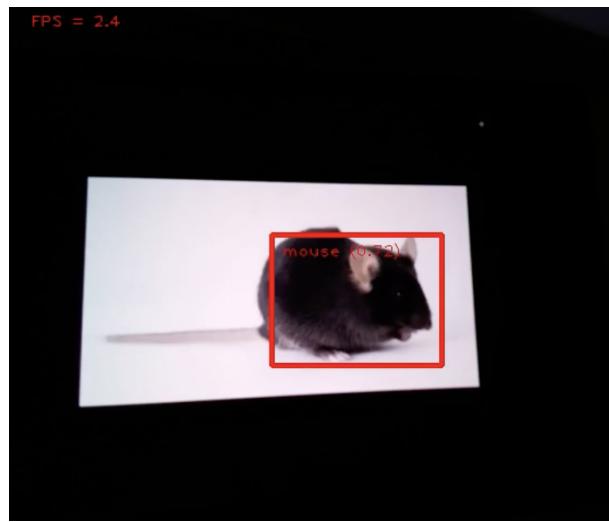


Figura 3.1: Detección de ratones en una imagen usando TensorFlow Lite.

### 3.4. OpenCV

OpenCV<sup>4</sup> (Open Source Computer Vision Library) es una librería de software de visión computacional y machine learning de código abierto. Es ampliamente usada en todo el mundo debido al soporte multiplataforma que ofrece para Windows, Linux, MacOS y Android, además de ofrecer interfaz en diferentes lenguajes como Python, Java, C++ y MATLAB.

Compuesto por más de 2500 algoritmos, OpenCV está especializado en la visión computacional y en algoritmos de aprendizaje, permitiendo reconocer rostros, identificar o clasificar objetos, extraer modelos 3D, mejorar la calidad de las imágenes o detectar bordes entre otros.

El uso de OpenCV en este trabajo ha permitido la manipulación de los vídeos

---

<sup>4</sup><https://opencv.org>

grabados por las cámaras en tiempo real, de manera que se ha podido integrar en él la fecha y hora del momento. También ha permitido guardar los vídeos cuando el usuario lo solicite. Debido a su amplio uso y su fácil adaptación a otras aplicaciones, ha sido posible mostrar los vídeos de ambas cámaras en el servidor web sin problema con el uso de OpenCV junto a Flask.

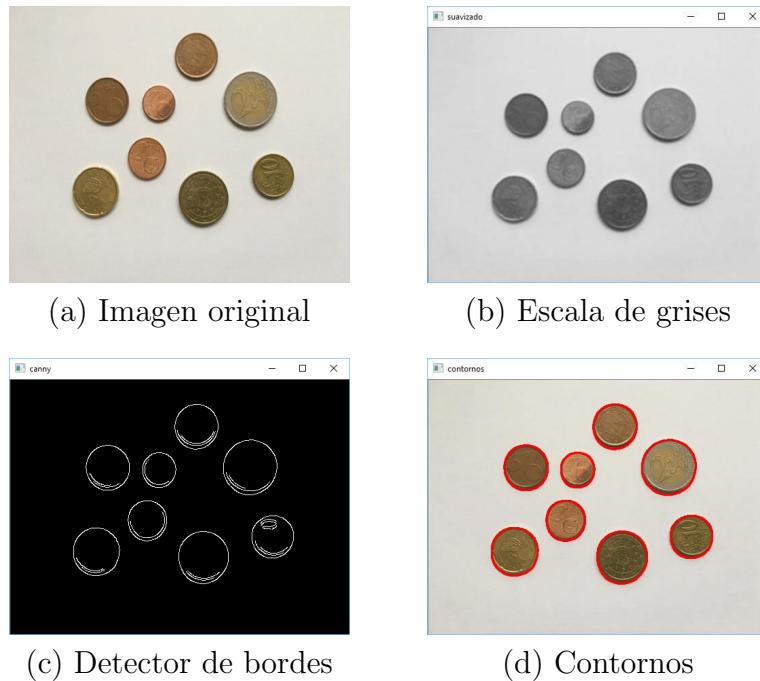


Figura 3.2: Ejemplo de detección de bordes con OpenCV.

### 3.5. Flask

Flask<sup>5</sup> es un marco de aplicación web minimalista —en inglés, microframework— escrito en Python. Esto significa que ofrece al usuario herramientas y librerías para crear una aplicación web, y al ser minimalista, no requiere de otras dependencias para realizar las cosas básicas. Ofrece una serie de extensiones que permiten dotar de más características a la aplicación, como autenticación o manejo a través de comandos, entre otros.

Se compone de dos partes; las plantillas —en inglés, templates—, que están escritas en lenguaje HTML e indican la organización y diseño que tendrá cada página al ser visualizada. La otra parte es código en Python que indica las rutas de cada página y

---

<sup>5</sup><https://flask.palletsprojects.com/en/2.1.x/>

las funciones de cada ruta. Es el fichero que se ejecuta para crear el servidor.

En este proyecto se ha utilizado Flask para la creación de los dos servidores web de las cámaras. Se ha preferido Flask frente a Django<sup>6</sup>, que es el más conocido para el uso con Python, debido a que es más sencillo de utilizar y de aprender. Después de haber realizado el login o el registro en el servidor, se accede a la visualización de cada cámara. Además, en el caso de la PiCamera, con Flask se ha integrado un botón que permite empezar o parar la grabación del vídeo.

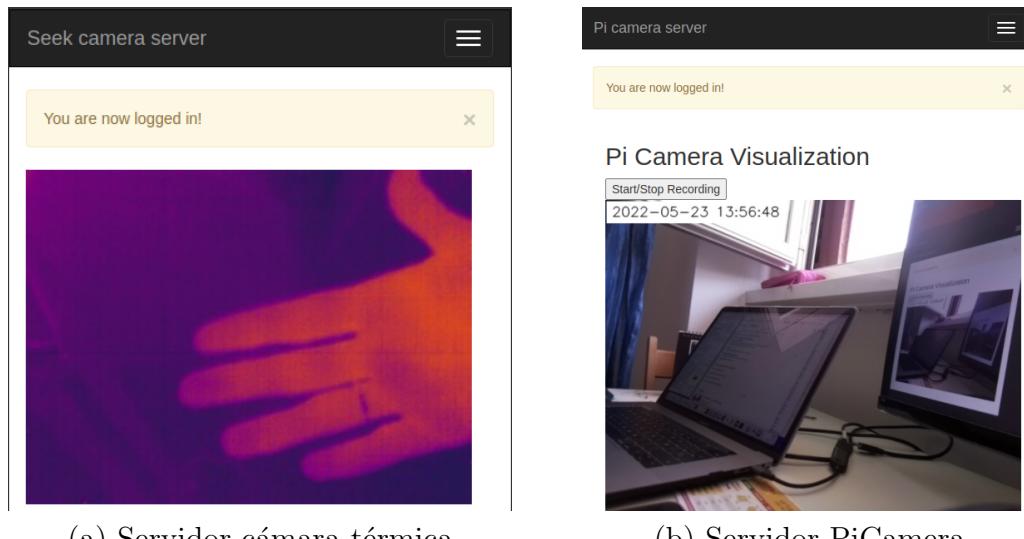


Figura 3.3: Muestra de los dos servidores del trabajo usando Flask.

### 3.6. Node-Red

Node-Red<sup>7</sup> es una herramienta de programación creada por IBM y escrita en JavaScript que permite conectar dispositivos hardware y servicios en línea. Muestra de manera visual las conexiones de los nodos en un panel, mostrando gráficamente el flujo de la información. Además, es un entorno de ejecución ligero, lo que lo hace ideal para ejecutarse en hardware de bajo coste como la Raspberry.

Está compuesto por dos partes principales. Una de ellas es la edición de flujo desde navegador, donde muestra los nodos disponibles en el margen izquierdo así como la

<sup>6</sup><https://www.djangoproject.com>

<sup>7</sup><https://nodered.org>

disposición y conexión del flujo creado por el usuario. La otra parte es el cuadro de mando, más comúnmente dashboard, que muestra la interfaz de usuario en base al flujo de los nodos de la parte previamente comentada.

Node-Red tiene una amplia variedad de nodos. Algunos de ellos permiten distintos tipos de conexiones, la interacción directa con los pines de Raspberry, la conexión con twitter o el correo electrónico. También permite la creación de ficheros de distintos tipos o la ejecución de ficheros ya existentes en la computadora.

Node-Red está escrito en JavaScript, por lo que cualquier función que se escriba directamente ahí deberá estar escrita en este lenguaje. Pueden crearse nuevos nodos para agregar nuevas capacidades gracias a los más de 225000 módulos que hay en el repositorio de paquetes.

Los flujos de datos se pueden importar o exportar con ficheros JSON, permitiendo compartirlos con cualquier persona. La página de Node-Red tiene una librería de flujos que han creado otros usuarios, permitiendo el acceso a ellos para utilizarlos como base en otros proyectos.

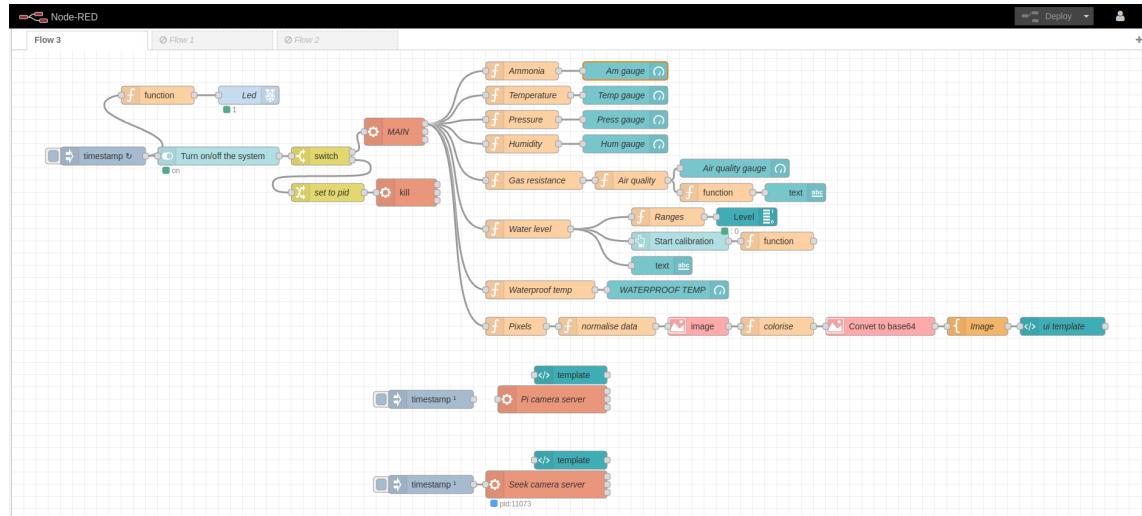


Figura 3.4: Flujo del proyecto en Node-Red.

Para el desarrollo de este TFG, Node-Red ha tenido un papel muy importante. Además de tener una gran comunidad que permite la resolución de la mayoría de dudas, ha permitido la creación de la interfaz de usuario donde se muestran las mediciones de cada sensor, los dos servidores web de las cámaras y los botones e interruptores necesarios que permiten al usuario interactuar con la interfaz de una manera sencilla

para obtener toda la información.

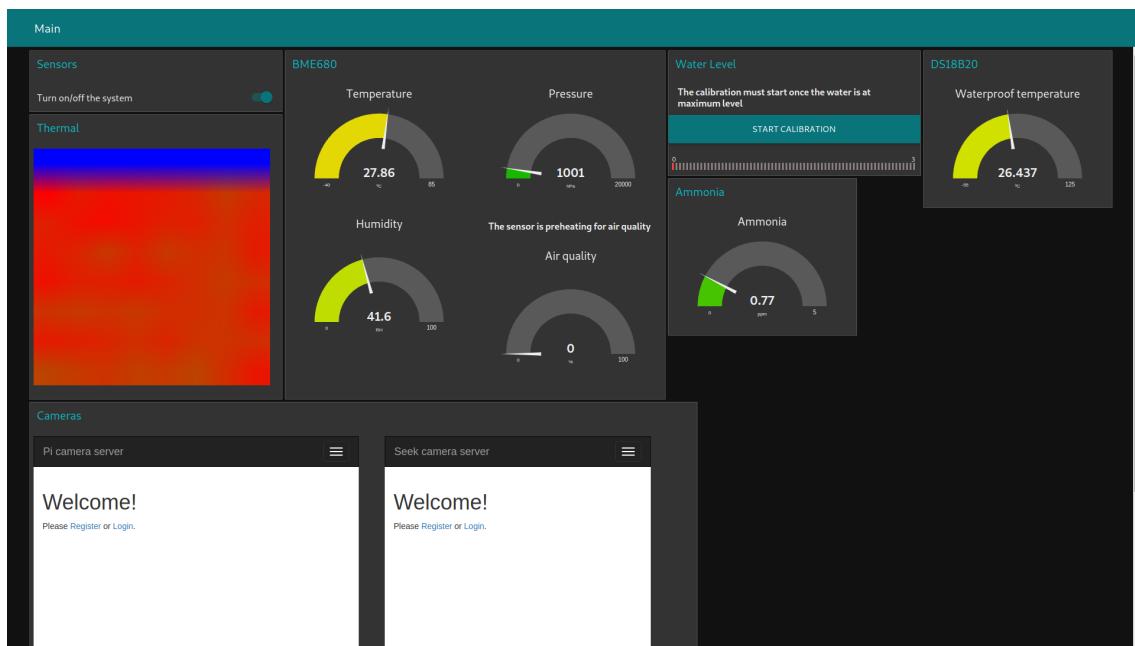


Figura 3.5: Muestra del uso de Node-Red en el trabajo.

---

## Capítulo 4

# Sistema multisensorial

---

Este capítulo consta de la descripción detallada del proceso que se ha llevado a cabo para crear el sistema multisensorial para la monitorización de animales de laboratorio.

La base idea de este trabajo ha sido pensada para un laboratorio de animales, ya que la robótica está presente en este ámbito, así como en otros muchos, a pesar de que en este sector no trabajan ingenieros. Debido a este motivo, puede haber trabajo que se realice en el laboratorio que podría mejorarse hasta incluso automatizarse.

Para obtener más información, se ha contactado con el laboratorio de animales de la Universidad de Alcalá de Henares<sup>1</sup> para conocer la situación en la que trabajan. Se han apreciado bastantes cosas mejorables y se ha enfocado el objetivo en el estudio de los ratones del laboratorio, que requieren una observación constante así como del entorno en el que se encuentran debido a que deben estar bajo unas condiciones determinadas.

Después de conocer esta información la idea del trabajo ha tomado más forma e idea, decidiendo así componer un sistema multisensorial. Este sistema consiste en dotar a un computador de diferentes sensores que le permitan obtener datos del entorno que le rodea para poder actuar en función a ellos.

Asimismo, debido al desconocimiento de los trabajadores del laboratorio del mundo de la robótica y la ingeniería en general, se ofrece una interfaz gráfica donde los valores numéricos obtenidos por los sensores, así como cualquier interacción necesaria se traducen en *widgets* comprensibles para cualquier usuario de una manera intuitiva y sencilla.

Dos son las partes esenciales para el desarrollo: hardware y software. En la primera sección se presenta el desarrollo hardware y en la sección posterior el desarrollo software que se ha realizado sobre la configuración hardware.

---

<sup>1</sup><https://www.uah.es/es/>

## 4.1. Hardware

La placa que se ha utilizado para el proyecto es la Raspberry Pi 4B (Figura 4.1), el último modelo de Raspberry y muy utilizada para el desarrollo de proyectos debido a su bajo coste. Este modelo es el más rápido y potente, ofreciendo de dos a tres veces el rendimiento del procesador de su modelo predecesor. El sistema operativo que utiliza la Raspberry es Raspbian.



Figura 4.1: Raspberry Pi 4B

Está dotada de una serie de pines y puertos que permiten su conexión con distintos sensores o actuadores. Para este trabajo se han utilizado los siguientes sensores:

- Sensor BME680 (Figura 4.2). Este sensor es capaz de medir hasta cuatro parámetros: el gas, la temperatura, la presión y la humedad.
- Sensor AMG8833 (Figura 4.3). Este sensor es una cámara térmica que permite recibir la información del entorno y mostrarla a través de un rango de colores. Los colores más cálidos —como el rojo— indican la presencia de mayor temperatura, mientras que los colores fríos —como el azul— indican una temperatura inferior. Los valores que ofrece este sensor son una lista de 8x8 valores, que debe ser traducida para mostrar una imagen.
- Sensor DS18B20 (Figura 4.4). Este sensor permite medir la temperatura en superficies húmedas, ya que es resistente al agua.



Figura 4.2: Sensor BME680.

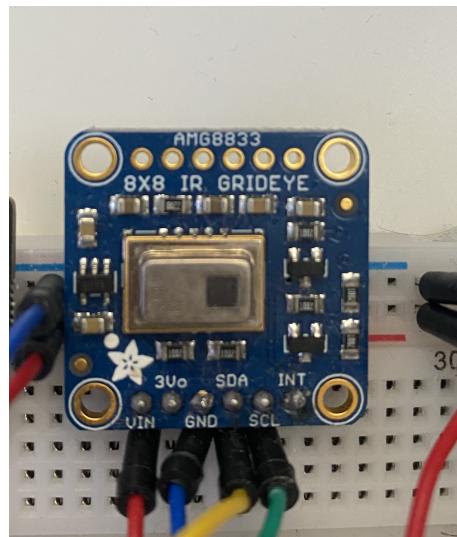


Figura 4.3: Sensor AMG8833.

- MQ-135 (Figura 4.5). Este sensor permite medir la calidad de distintos gases, entre ellos el de amoníaco. Este sensor tiene lecturas analógicas, por lo que es necesario el uso de un conversor analógico a digital.
  
- Sensor nivel de agua (Figura 4.6). Con este sensor se puede medir la presencia de agua, así como la cantidad de agua que hay. Este sensor tiene lecturas analógicas, por lo que es necesario el uso de un conversor analógico a digital.



Figura 4.4: Sensor DS18B20.



Figura 4.5: Sensor MQ-135.

- Seek thermal (Figura 4.7). Este sensor es una cámara térmica. Permite obtener imágenes representando las temperaturas con colores de la misma forma que el sensor 4.3, pero con mucha más calidad y precisión.
- PiCamera (Figura 4.8). Este sensor es una de las cámaras oficiales de Raspberry, que permite obtener imágenes del entorno y mostrarlas en pantalla.

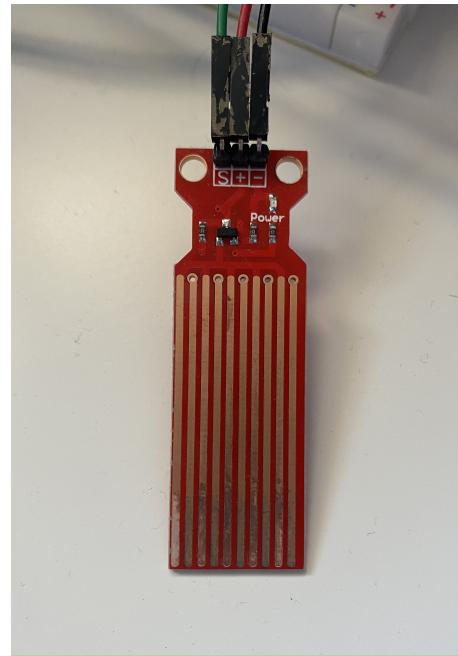


Figura 4.6: Sensor de nivel de agua.



Figura 4.7: Cámara térmica.

Además de los sensores, se han utilizado otros componentes como son resistencias, un led o un conversor analógico a digital para el correcto funcionamiento de los sensores y proporcionar mayor comprensión al funcionamiento del sistema.

En la figura 4.9 se puede ver un esquema de la conexión de los sensores mencionados en la Raspberry.

## 4.2. Software

Tras disponer de la parte hardware, el equipo está preparado. Para que este equipo funcione, es necesario crear las instrucciones y reglas a través de programas



Figura 4.8: Cámara de Raspberry.

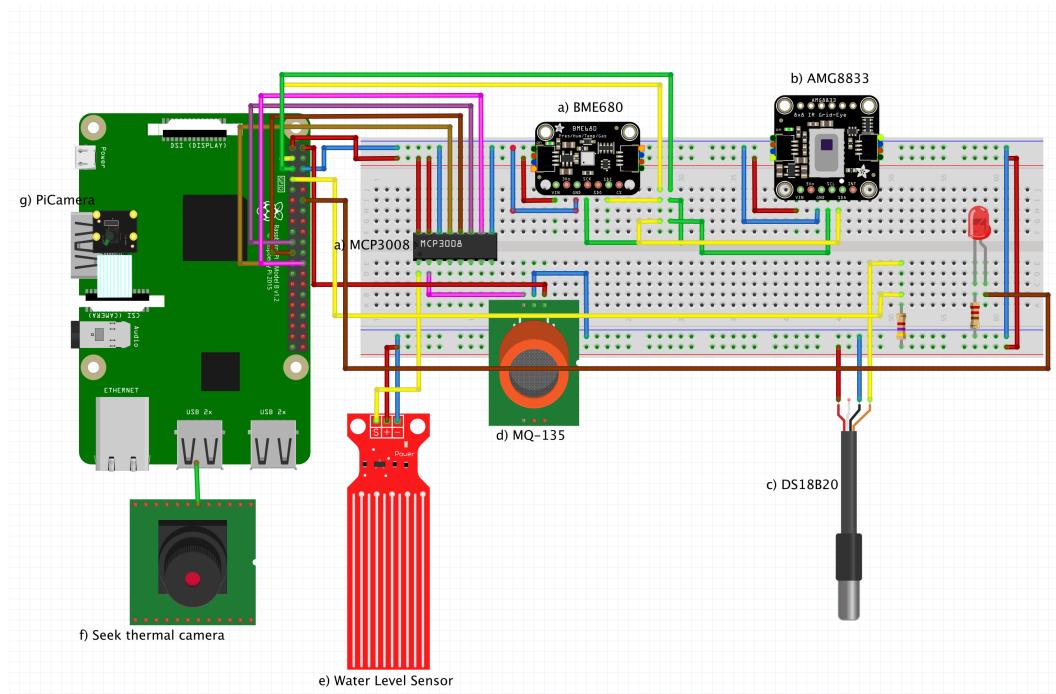


Figura 4.9: Esquema de conexiones.

o aplicaciones, esto es, la parte software.

En primer lugar ha sido necesaria la instalación de las librerías necesarias para el funcionamiento de todos los sensores. Después de la instalación, han surgido varios problemas al intentar leer de los distintos sensores. Uno de los problemas encontrado en dos sensores —los que funcionabas con conexión i2c— ha sido que la Raspberry no

los detectaba.

Se pueden encontrar los errores que han surgido en la instalación y las respectivas soluciones para hacer funcionar todos los sensores en la wiki<sup>2</sup> que se ha creado para realizar el proyecto.

#### 4.2.1. Fichero Python para lectura sensorial

Durante este proceso, ha habido dos sensores que se han intentado utilizar sin éxito debido a que no tienen compatibilidad con Raspberry: CCS811, que mide la calidad del aire y AM2315, que mide la temperatura y humedad. Aun así, esto no ha sido un problema ya que son datos que se han podido obtener con otros sensores disponibles.

Se ha considerado que el usuario no tiene conocimiento para comprender los datos en crudo que ofrece un sensor, por tanto, ha sido necesario realizar diferentes cambios para ofrecer una comprensión fácil al usuario.

Se ha utilizado el programa Node-Red para ofrecer una interfaz al usuario, de forma que este sea capaz de ver la información del estado del sistema representada a través de widgets, que son pequeñas imágenes que proveen información visual. Así ha sido posible crear un interfaz ameno e intuitivo que el usuario puede comprender a través de algo más visual que valores numéricos.

Para poder mostrar todos los sensores en tiempo real en Node-Red, ha sido necesario disponer de un fichero de Python que cada vez que se ejecute lea una sola vez de todos ellos. No debe leer continuamente ya que es la configuración de Node-Red la que se encarga de ello. En este fichero sólo se procesan los sensores que devuelven salidas numéricas, esto es, todos sensores excepto la PiCam 4.8 y la cámara térmica 4.7.

Para crear este fichero de una manera organizada, se ha creado una clase por cada sensor. El esquema de la clase de cada sensor se puede ver en la Figura 4.10.

Este fichero de Python utiliza la librería Threads para poder leer de manera concurrente todos los sensores, ahorrando tiempo y ganando eficacia. El código 4.1 muestra cómo se crea un hilo —o thread— por cada sensor, siendo la lista del primer bucle una función que lee el valor del sensor respectivo.

Además, este programa guarda las lecturas en un fichero csv por si fuese necesaria la comparación o estudio de datos a lo largo del tiempo.

---

<sup>2</sup><https://github.com/jmvega/tfg-icebollada/wiki/2.February-progress>

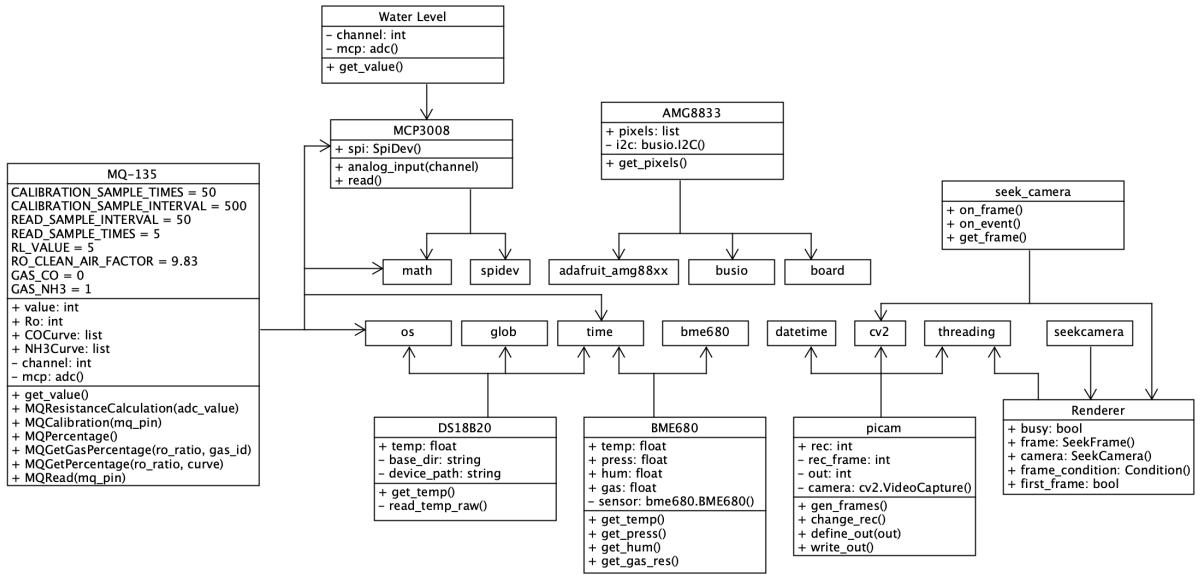


Figura 4.10: Esquema de clases de cada sensor.

---

```

threads = []

for func in [x.ammo, x.bme, x.pix, x.lev, x.water_t]:
    threads.append(Thread(target=func))
    threads[-1].start()

for thread in threads:
    thread.join()
  
```

---

Código 4.1: Función para crear un Thread por sensor y obtener su lectura

El fichero devuelve una cadena con la lectura de cada sensor, para poder procesar la salida en Node-Red.

#### 4.2.2. Creación de la interfaz de usuario

Una vez creado este fichero, se ha procedido a crear la interfaz en Node-Red. Este programa funciona a través de diferentes tipos de nodos, que deben ser combinados para obtener el resultado deseado.

En primer lugar, a través del nodo de ejecución, representado en rojo con un engranaje, se pueden ejecutar ficheros que están en la máquina local. Este nodo es el que se ha utilizado para integrar en la aplicación el fichero que lee una vez todos los sensores.

Para obtener individualmente el valor de cada sensor, se ha utilizado otro nodo

llamado nodo función, que permite incluir código directamente en Node-Red. Este nodo se ha utilizado para extraer de la salida del nodo ejecución el valor que le interesa.

Con estos dos nodos algunos sensores han estado listos para mostrar los valores en la interfaz de usuario (IU), por lo que solo ha sido necesario añadir el widget más adecuado para cada caso a través del nodo pertinente. Este ha sido el caso de los sensores MQ-135 4.5, BME680 4.2 (en el caso de temperatura, presión y humedad) y DS18B20 4.4.

En el caso de los otros sensores han sido necesarios otros cambios para obtener el resultado adecuado:

- El sensor BME680 4.2 mide la resistencia del gas, pero se ha considerado que no es un parámetro intuitivo y no proporciona conocimiento al usuario. Por ello se ha utilizado otro nodo función que primero realiza la media de 50 lecturas de resistencia de gas y posteriormente establece la calidad del aire en función a la humedad y resistencia del aire.
- Durante diferentes pruebas, se ha detectado que el sensor que mide el nivel de agua 4.6 no es preciso a la hora de detectar la cantidad de agua presente, solamente es preciso detectando la presencia de esta.

Por tanto, se ha optado por dividir en 4 rangos el nivel de agua, de tal manera que el usuario tenga una idea sobre la cantidad de agua presente. Aún así, el sensor no es suficientemente preciso a lo largo de diferentes pruebas y esta opción tampoco ha sido válida. Como ajuste que dota de más precisión a este sensor, se ha incorporado un botón en la interfaz para poder calibrar el sensor cuando el recipiente esté lleno de agua. Todo el desarrollo y las pruebas que se han llevado a cabo con este sensor pueden ser encontradas en la wiki <sup>3</sup> del proyecto.

- El sensor AMG8833 4.3 devuelve en su lectura una lista de 8x8 valores. Estos valores han sido convertidos a imagen ya que este sensor es una cámara térmica. Para este proceso ha sido necesario enlazar una serie de nodos hasta poder obtener la imagen más nítida, ya que la imagen obtenida originalmente ha sido una imagen de 8x8 píxeles que no alcanzaba a diferenciar objetos. De nuevo, tanto este progreso como las diferentes soluciones obtenidas se pueden encontrar en la wiki.

Con estos ajustes realizados, se ha obtenido parte de la IU. Para hacerla más completa se ha incorporado un interruptor de forma que el usuario puede decidir si

---

<sup>3</sup><https://github.com/jmvega/tfg-icebollada/wiki/3.March-progress>

apagarla o bien mantenerla encendida, así como un nodo que regula la repetitividad del sistema cada 3 segundos.

La interfaz a este punto se puede ver en la imagen 4.11.

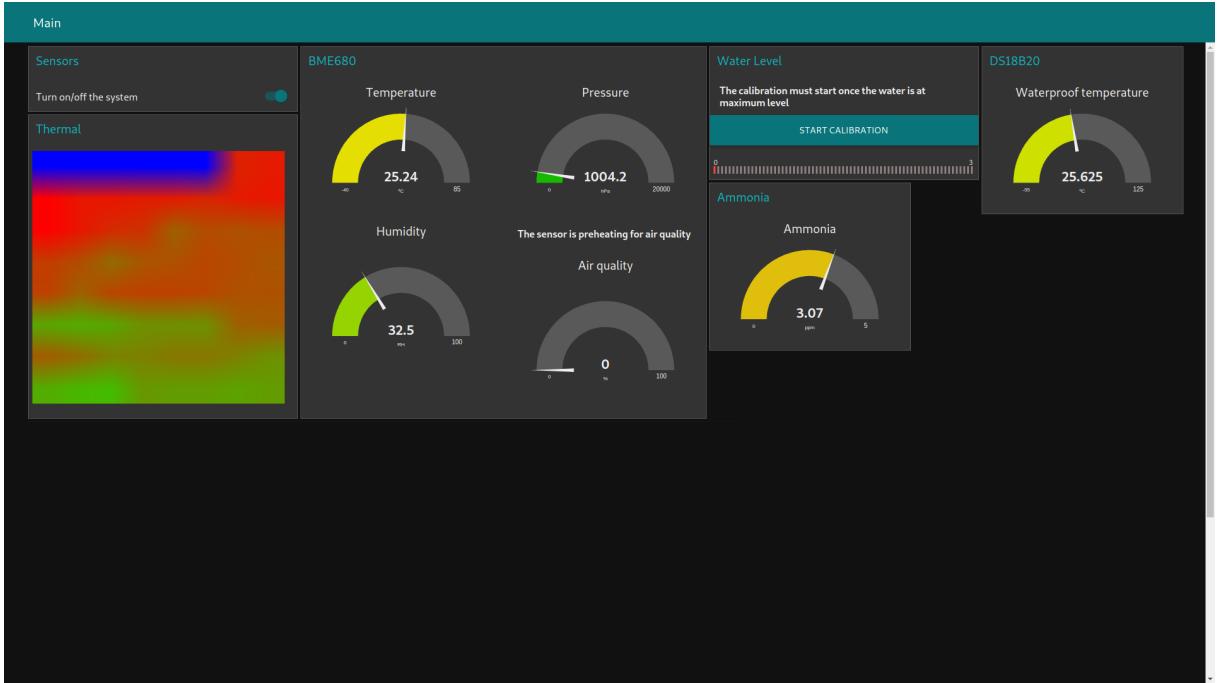


Figura 4.11: IU sin los sensores cámaras.

#### 4.2.3. Integración de las cámaras en la UI

La creación de la interfaz en la subsección anterior no es completa, ya que faltan dos de los sensores que se utilizan en este trabajo: las cámaras.

Para poder visualizar las cámaras desde Node-Red debe ser a través del nodo template (que significa plantilla), que permite visualizar una dirección web. Por tanto ha sido necesario crear un servidor que permita visualizar las imágenes en un servidor y por tanto poder incluirlo en la IU. Para crear este servidor se ha utilizado Flask.

Se ha decidido crear una servidor por cada cámara. De esta manera se puede acceder solamente a una visualización o bien a las dos, según el interés del usuario.

Para el funcionamiento de la cámara térmica Seek Thermal 4.7 se ha utilizado la librería que tiene seek, que permite su visualización a través de OpenCV.

Para el funcionamiento de la PiCam 4.8 se ha utilizado OpenCV tanto para su visualización como para la adición de la fecha y la hora en la visualización (Figura



Figura 4.12: Incorporación de la fecha y hora en la visualización de la PiCam.

4.12). El código 4.2 muestra como se consigue.

---

```
ret, frame = self.__camera.read()
frame = cv2.rectangle(frame, (2,2), (275,35), (255,255,255), -1)
frame = cv2.putText(frame,
    str(datetime.datetime.now().replace(microsecond=0)), (10,25),
    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,0,0), 1, cv2.LINE_AA)
```

---

Código 4.2: Código para crear un rectángulo y escribir la fecha actual sobre el.

Una vez obtenidas las imágenes de las cámaras desde la Raspberry, se ha incorporado al código el uso de Flask para poder crear el servidor.

Es necesario disponer de dos elementos: el código para mostrar la imagen de la cámara y una carpeta donde estás las plantillas. Estas plantillas son ficheros escritos en html que muestran en el servidor la página que interesa en cada momento.

La estructura que sigue un programa con Flask es la representada en el código 4.3.

---

```
from flask import *
app = Flask(__name__)

@app.route('/')
def home():
    return render_template('index.html')

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000, debug=True)
```

---

Código 4.3: Código que crea un servidor web y muestra el contenido de index.html.

Después de incorporar el código del funcionamiento de cada cámara en los respectivos servidores, se ha añadido un botón para iniciar o parar la grabación del vídeo en la cámara normal. Para diferenciar cuando se está grabando de cuando no, se ha añadido un mensaje en pantalla indicando que se está grabando. El vídeo se guarda en la Raspberry de forma local.

El diagrama de casos de uso del proyecto con la interfaz creada aparece representado en la figura 4.13

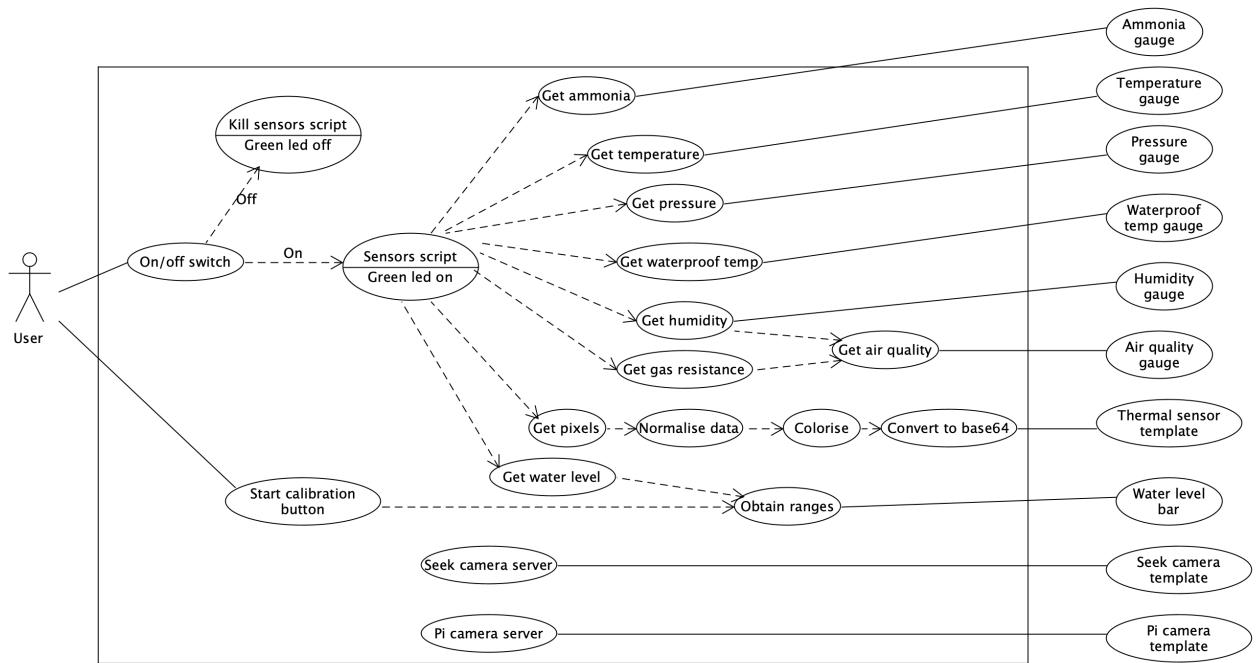


Figura 4.13: Diagrama de casos de uso del proyecto.

#### 4.2.4. Seguridad

Debido a que se está trabajando en un servidor web, es importante dotar de cierta seguridad al servidor para evitar que sólo las personas autorizadas puedan acceder a las imágenes. Para ello, se ha tomado la decisión de hacerlo a través del factor de doble autenticación (del inglés two factor authentication, 2FA). 2FA es el proceso de autenticación donde se añade un segundo paso en el proceso de identificación frente a un servicio. En este caso, además de que un usuario introduzca su contraseña, es necesario un código que se genera cada 30 segundos y pasado este tiempo caduca. La

generación de este código se hace a través de google authenticator<sup>4</sup>. Esta extensión de google genera los códigos de 30 segundos a partir de el escaneo de un código QR o por la inserción manual de un código inicial.

Para que estos servidores funcionen con 2FA, ha sido necesario incorporar otras librerías así como otros módulos de Flask como Flask\_login, Flask\_bootstrap, Flask\_sqlalchemy o Flask\_wtf. Durante el desarrollo del servidor con 2FA, han surgido determinados problemas que pueden encontrarse junto a la solución en la wiki del proyecto<sup>5</sup>.

Una vez los servidores web han funcionado, se han integrado en Node-Red a través del nodo template. Un servidor está en el puerto 5000 y el otro en el 8000 para que así puedan estar ambos disponibles a la vez. Después de añadirlos, la interfaz de usuario (Figura 4.14) se ha completado.

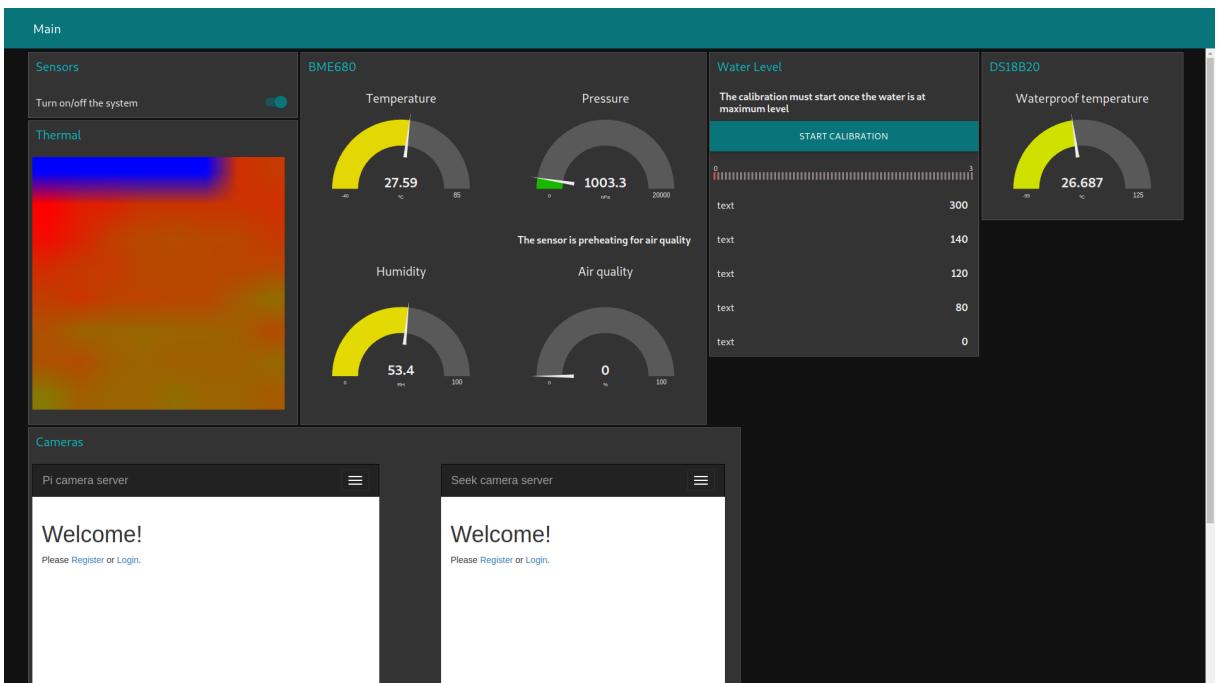


Figura 4.14: Visualización de la interfaz de usuario.

También se ha tenido en cuenta que Node-Red funciona sobre una dirección a la que puede acceder cualquier usuario. Por tanto, es imprescindible que sean solo usuarios

<sup>4</sup><https://chrome.google.com/webstore/detail/authenticator/bhghoaemapcdpbophigoooaddinpkbai?hl=es>

<sup>5</sup><https://github.com/jmvega/tfg-icebollada/wiki/5.May-progress>

autorizados los que puedan acceder a esta plataforma, sobre todo en la parte del flujo de nodos.

Por tanto se ha optado por dividir este acceso en dos partes: Una de ellas la parte de administrador 4.15, donde se encuentra el flujo de nodos y se crea toda la interfaz de usuario. Únicamente un administrador debe poder acceder a esta parte, ya que es importante que una persona que no tiene conocimientos acerca de su funcionamiento no pueda realizar modificaciones. La otra parte corresponde a la interfaz de usuario 4.16 que, aunque ahí no pueden realizarse modificaciones, solo un usuario registrado conocedor de la contraseña sea capaz de acceder para evitar que personas ajenas puedan acceder a la información disponible.

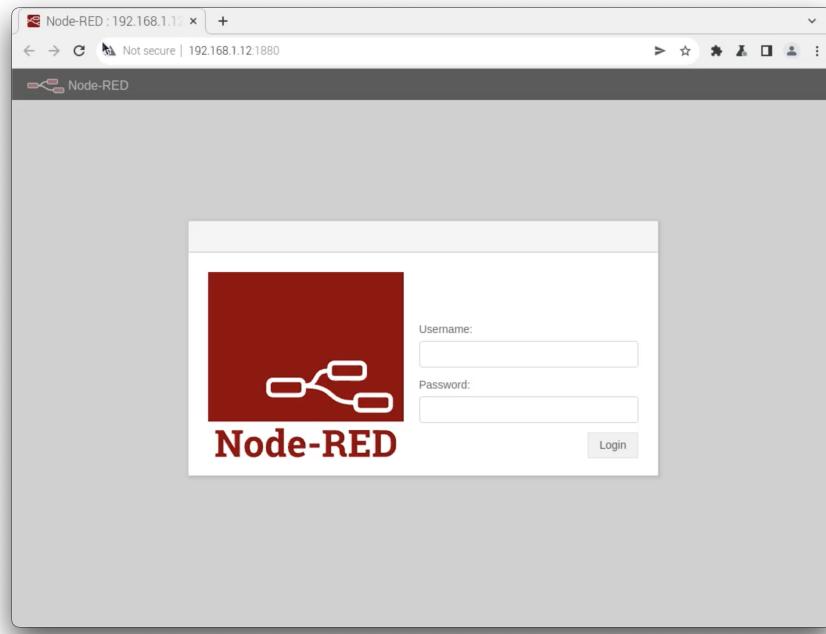


Figura 4.15: Acceso al flujo de nodos en Node-Red.

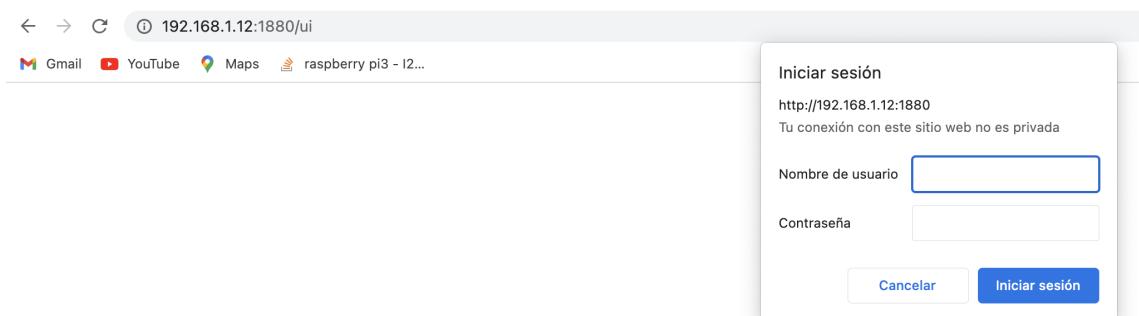
#### 4.2.5. Autoarranque

Recordando que este sistema está pensado para personas que no están familiarizadas con la programación o los ordenadores, se ha considerado añadir una opción que facilite el arranque del sistema.

Con el encendido de la Raspberry, la interfaz de usuario se lanza de forma automática y se enciende solicitando el login del usuario para acceder a la IU. Además, se ha incorporado un led verde que indica cuando el sistema está listo para ser usado.



(a)



(b)

Figura 4.16: Acceso a la interfaz de usuario desde distintos dispositivos.

---

# **Capítulo 5**

# **Conclusiones**

---

*Quizás algún fragmento de libro inspirador...*

Autor, Título

Escribe aquí un párrafo explicando brevemente lo que vas a contar en este capítulo, que básicamente será una recapitulación de los problemas que has abordado, las soluciones que has prouesto, así como los experimentos llevados a cabo para validarlos. Y con esto, cierras la memoria.

## **5.1. Conclusiones**

Enumera los objetivos y cómo los has cumplido.

Enumera también los requisitos implícitos en la consecución de esos objetivos, y cómo se han satisfecho.

No olvides dedicar un par de párrafos para hacer un balance global de qué has conseguido, y por qué es un avance respecto a lo que tenías inicialmente. Haz mención expresa de alguna limitación o peculiaridad de tu sistema y por qué es así. Y también, qué has aprendido desarrollando este trabajo.

Por último, añade otro par de párrafos de líneas futuras; esto es, cómo se puede continuar tu trabajo para abarcar una solución más amplia, o qué otras ramas de la investigación podrían seguirse partiendo de este trabajo, o cómo se podría mejorar para conseguir una aplicación real de este desarrollo (si es que no se ha llegado a conseguir).

## 5.2. Corrector ortográfico

Una vez tengas todo, no olvides pasar el corrector ortográfico de L<sup>A</sup>T<sub>E</sub>Xa todos tus ficheros *.tex*. En Windows, el propio editor TeXworks incluye el corrector. En Linux, usa aspell ejecutando el siguiente comando en tu terminal:

```
aspell --lang=es --mode=tex check capitulo1.tex
```

# Bibliografía

---

- [Arce et al., 2009] Arce, A., Tech, A., Silva, A., and Costa, E. (2009). Wireless sensor networks for bovine herd monitoring. *Archivos de Zootecnia*, 58:253 – 263.
- [Assael et al., 2022] Assael, Y., Sommerschield, T., Shillingford, B., Bordbar, M., Pavlopoulos, J., Chatzipanagiotou, M., Androutsopoulos, I., Prag, J., and de Freitas, N. (2022). Restoring and attributing ancient texts using deep neural networks — nature.
- [Guermazi et al., 2021] Guermazi, A., Tannoury, C., Kompel, A. J., Murakami, A. M., Ducarouge, A., Gillibert, A., Li, X., Tournier, A., Lahoud, Y., Jarraya, M., Lacave, E., Rahimi, H., Pourchot, A., Parisien, R. L., Merritt, A. C., Comeau, D., Regnard, N.-E., and Hayashi, D. (2021). Improving radiographic fracture recognition performance and efficiency using artificial intelligence — radiology. Radiological Society of North America.