Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Instrumentación inteligente no invasiva para procesos en ganadería y agricultura

Autor: Miguel Ángel Prieto Ferrer
Tutor: Ramón González Carvajal

Dpto. Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2021

Proyecto Fin de Carrera
Ingeniería Electrónica, Robótica y Mecatrónica

Instrumentación inteligente no invasiva para procesos en ganadería y agricultura

Autor:
Miguel Ángel Prieto Ferrer

Tutor:
Ramón González Carvajal
Profesor Catedrático

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2021

Proyecto Fin de Carrera: Instrumentación inteligente no invasiva para procesos en ganadería y agricultura

Autor: Miguel Ángel Prieto Ferrer

Tutor: Ramón González Carvajal

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2021

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

En primer lugar, agradecer a Ramón, mi tutor, por depositar tanta confianza en mí y brindarme la oportunidad de participar en este proyecto. Gracias por el contínuo apoyo, haciendome ver que, en todo momento, lo mas importante es ser feliz con lo que uno hace. Desde el primer día hasta el último he estado aprendiendo y, tú, nunca has dejado de enseñarme con una sonrisa de oreja a oreja, valorando mi esfuerzo y cada meta que he cumplido.

También me gustaría agradecer a mis padres, por el apoyo incondicional que me han dado, no solo durante estos cuatro años, si no desde que tengo uso de razón. Gracias por estar ahí, ante todo, ayudandome a ver el lado racional de las cosas y motivandome a seguir lo que me dice el corazón. Desde chico, me habéis enseñado a luchar por mis sueños y, gracias a eso, estoy aquí. No podría haber pedido tener mejores padres.

Gracias a mis hermanos, Ignacio y Javier, por haber estado siempre a mi lado, ayudandome en lo que habéis podido y escuchándome cuando lo necesitaba. Gracias a vosotros he aprendido muchas cosas, pero, lo que mas me habéis enseñado es que, para encontrar a tus mejores amigos, no hace falta ni asomarse a la puerta.

Finalmente, me gustaría agradecerme a mí. Porque nadie ha estado en tantos peores momentos como yo mismo y, gracias a mi testarudez, he conseguido las metas que me he propuesto. Llegar hasta aquí no ha sido fácil, pero, sin duda, no hubiese sido posible si yo no hubiera sido capaz de sacar fuerzas para seguir luchando

Miguel Ángel Prieto Ferrer

Sevilla, 2021

Resumen

La fiebre en los cerdos y el estrés hídrico que sufren las plantas son algunos ejemplos de las preocupaciones de la industria ganadera y agricultora, donde la tecnología de sensores e IoT pueden aportar soluciones a un coste razonable.

La problemática de síntomas febriles en cerdos reside en la obligación de diezmar la población infectada con objeto de evitar la propagación del virus. Por tanto, una detección temprana podría evitar grandes pérdidas económicas. Se planteará una solución mediante la toma de la temperatura de los sujetos. Cabe destacar que esta toma de temperatura usará métodos no invasivos, evitando así obtener datos irreales causados por el estrés del animal.

Por otra parte, el estrés hídrico en plantas puede originar problemas de producción al no ser capaz de generar frutos de manera eficiente. Aunque no están muy definidos los métodos para determinar la falta de agua, se desarrollará una infraestructura que nos aporte los datos necesarios para orientar el problema.

Abstract

Fever in pigs and water stress suffered by plants are some examples of concerns in the livestock and agriculture industry, where sensor technology and IoT can provide solutions at a reasonable cost.

The problem of febrile symptoms in pigs lies in the obligation to decimate the infected population in order to avoid the spread of the virus. Therefore, an early detection could avoid great economic losses. A solution will be proposed by taking the temperature of the subjects.

On the other hand, water stress in plants can cause production problems by not being able to produce fruit efficiently. Although the methods to determine the lack of water are not well defined, an infrastructure will be developed to provide us with the necessary data to address the problem.

Índice

Agradecimientos ix

Resumen xi

Abstract xiii

Índice xiv

Índice de Figuras xvi

Notación xviii

1 Introducción 1

1.1. Motivación 3

1.2 Objetivos, alcance y requisitos 3

2 Estado del arte en medidas de bajo coste 5

2.1. Toma de temperatura en animales 5

2.2. Estrés hídrico 7

2.3 Conclusiones comunes 8

3 Arquitectura 9

4 Hardware 11

4.1. Raspberry Pi 3 11

4.2. Cámara FLIR Lepton y módulo Purethermal mini 12

4.3. PT100 y MAX31865 13

4.4. Manta térmica y MOSFET Driver IRF520 14

4.5. Cámara NoIR 15

4.6. DHT11 16

4.7. Conexiones y montaje final 17

4.7.1 Montaje Común 17

4.7.2 Montaje para la toma de temperatura 17

4.7.3 Montaje para determinar el estrés hídrico 18

5 Firmware 19

5.1. Obtencion de datos 19

5.1.1 Captura de imagenes 19

5.1.1.1 Cámara FLIR 20

5.1.1.2 Cámara NoIR 21

5.1.2 Obtención de datos de los sensores de humedad y temperatura 22

5.2. Detección de ojos 23

5.3 Control de temperatura 24

5.4 Correción de error y almacenaje de resultados 25

6 Resultados y conclusiones 29

6.1. Resultados finales 29

6.2. Análisis de resultados *30*
6.2.1 Análisis del Control de temperatura *30*
6.2.2 Análisis de la Detección de ojos *31*
6.2.3 Análisis del sistema completo *32*
6.3 Consideraciones *33*
6.4 Aplicaciones futuras *34*

Referencias **35**

Glosario **36**

Anexo: Código 37

ÍNDICE DE FIGURAS

Figura 1-1. Desarollo de la grasa del cerdo durante su cría. 1
Figura 2-1. Escáner de temperatura 6
Figura 2-2. Clasificadores de Haar 6
Figura 2-3. Foto de un satélite tras aplicar el NDVI 7
Figura 2-4. Ensayo Bolómetro 8
Figura 3-1. Arquitectura 10
Figura 4-1. Raspberry Pi 3 Modelo B 11
Figura 4-2. Cámara FLIR Lepton 12
Figura 4-3. Módulo Purethermal mini 12
Figura 4-4. Montaje PT100 y MAX31865 14
Figura 4-5. Driver IRF520 15
Figura 4-6. Cámara NoIR v2 16
Figura 4-7. Sensor DHT11 16
Figura 4-8. Montaje común 17
Figura 5-1. Configuración Cámara FLIR 20
Figura 5-2. Uso de la función minMaxLoc 21
Figura 5-3. Configuración Cámara NoIR 21
Figura 5-4. Código PT100 22
Figura 5-5. Rutina DHT11 23
Figura 5-6. Preprocesado de la imagen 23
Figura 5-7. Detección de ojos 24
Figura 5-8. Control de temperatura 25
Figura 5-9. Corrección de la temperatura de la imagen 26
Figura 5-10. Temperatura media del ojo 26
Figura 5-11. Almacenaje de gráficas 27
Figura 5-12. Almacenaje datos cultivo 27
Figura 6-1. Ejemplo de resultado 30
Figura 6-2. Control de temperatura 31
Figura 6-3. Detección de ojos en personas 32
Figura 6-4. Prueba final 32

Notación

ºC	Grados centígrados
V	Voltios
mW	milivatios
GHz	Gigahercios
mm	milímetros
s	segundos
Ω	Ohmio
A	Amperios
GB	Gigabyte

1 INTRODUCCIÓN

La toma de temperatura en animales y la determinación del estrés hídrico son problemas que, hoy en día, no tienen solución de bajo coste que sea adoptable por todo tipo de explotaciones ganaderas y agrícolas. Mediante el desarrollo de este proyecto se pretende determinar una solución efectiva y de bajo coste que permita determinar si un animal presenta fiebre o si una planta presenta estres hídrico.

Para comenzar, al determinar la fiebre en un animal es preciso tomarle la temperatura, pero esto no es una tarea fácil. Si lo hacemos de manera invasiva, termómetros rectales, produciremos estrés sobre el sujeto, lo que alterará la medida real. Además, requiere un alto coste de mano de obra que lo hace poco realista, puesto que en España hay más de 55 Millones de cerdos, 20 Millones de Ovejas, 8 de vacas y 6 de cabras. Es preciso, por tanto, encontrar un método no invasivo y desatendido para la toma de temperatura, de forma que no haya que activarlo, esto es, que sea automático. Otro problema que surge al tomar la temperatura de un animal es que la gran mayoría de su cuerpo se encuentra rodeado de grasa, en el caso porcino, de lana en el caso de la oveja... para aislarlos de la temperatura exterior. Así, necesitamos un lugar del cuerpo que no esté recubierto de grasa, lana, piel aislante o mecanismos de control de temperatura corporal. Este punto, debe ser fácilmente identificable por técnicas de inteligencia artificial y común a todos los animales. El mejor candidato es el ojo, en concreto, la zona del lagrimal, pues no presenta ningun mecanismo que lo recubra y existen técnicas avanzadas de detección de dicho elemento.

Si nos centramos en el cerdo ibérico, podemos ver que algunos estudios [1], definen que la grasa es algo necesario para la cría del cerdo, lo que hace evidente la dificultad específica de esta industria a la hora de tomar la temperatura. Es por ello que nos centraremos en la toma de medidas sobre estos sujetos.

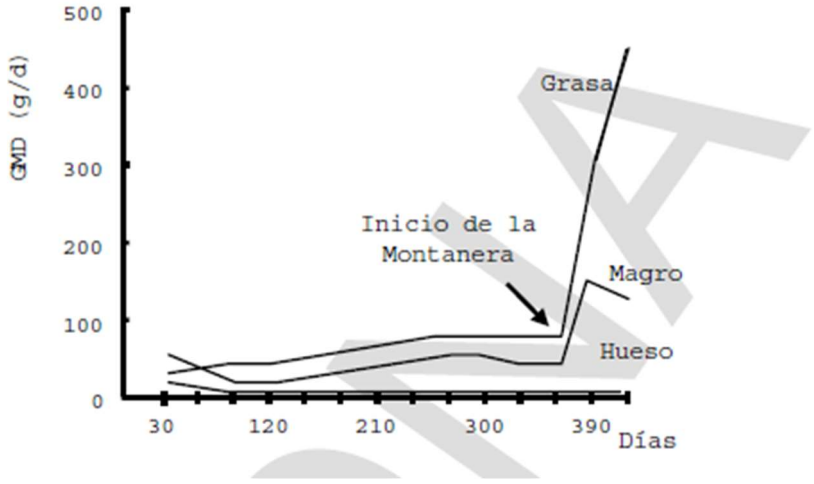

Figura 1-1. Desarrollo de la grasa del cerdo durante su cría

En la actualidad existen productos que permiten obtener la temperatura de algunos seres vivos, sin embargo, no se han llegado a implementar en los animales. En este último año, con la pandemia del COVID-19, el desarrollo de productos similares ha aumentado, aunque se han centrado únicamente en humanos. No obstante, es un buen punto de partida pues el conjunto de sensores que se usan son similares. Cabe destacar el alto precio de estos productos, lo que nos lleva a buscar elementos de bajo coste para el desarrollo de este producto.

Por otra parte, por lo que respecta al estrés hídrico, sabemos que las plantas que sufren estrés hídrico tienen una temperatura mayor a su temperatura normal, debido a que no son capaces de evapotranspirar[1] al no disponer de agua.
Algunos estudios [3], resaltan la necesidad de hacer varias tomas de temperatura tanto de la tierra donde se encuentran las raíces de la planta como del aire que rodea a la misma, lo que convertirá en un requisito indispensable el uso de sensores de temperatura y humedad para la determinación de este problema.
Además, existen otros métodos para determinar si una planta sufre estrés hídrico o no, por ejemplo, los basados en métodos infrarrojos. Destaca el Índice de Estrés Hídrico o Crop Water Stress Index (CWSI), el cual nos permite obtener un parámetro entre 0 y 1, siendo el primero indicativo de un correcto estado de la planta y siendo el segundo indicativo de que la planta sufre estrés hídrico o biótico y, por tanto, deberemos comprobar su estado. Este parámetro (1-1) depende de la temperatura de un dosel, la temperatura del aire y el deficit de presión de vapor. De esta manera, si lográsemos incorporar una expresión como la mostrada, podemos empezar a estimar el estado en el que se encuentra el cultivo. Por lo que, la obtención de un resultado, se encuentra en el desarrollo de un sistema que nos permita obtener los parámetros necesarios.

$$CWSI = \frac{[(T_C - T_a) - (T_{nws} - T_a)]}{[(T_{dry} - T_a) - (T_{nws} - T_a)]} \qquad (1-1)$$

Donde:

- **Tc**: canopy Temperature (ºC)
- **Ta**: air Temperature (ºC)
- **Tnws**: non-water stressed canopy Temperature (ºC)
- **Tdry**: waterstressed canopy Temperature (ºC).

Finalmente, para poder llegar a un producto a través de estre proyecto, será necesario automatizar mediante el uso de redes neuronales para poder determinar correctamente el ojo de un animal mediante la detección de este dentro de la foto tomada por el bolómetro .
Tambíen será necesario una mayor investigación de la problemática al determinar el estrés hídrico, pues no es un problema que esté bien definido y no se ha llegado a investigar a nivel profesional, por lo que, realmente, no hay índices que determinen de manera exacta que una planta presenta estos problemas, solo hay indicios de que al cumplirse una serie de características, la planta puede estar presentando algún tipo de problema como son la falta de calcio, estrés hídrico o estrés salino.

El objetivo es, por tanto, la integración de sensores de temperatura, sensores de humedad y cámaras capaces de detectar plantas y animales, tomar su temperatura y tomar la temperatura del ambiente que les rodea. Los datos serán almacenados en un microcontrolador de manera que sean accesibles para el usuario final del producto y pueda determinar una solución para el problema que se le presente.
En resumen, buscamos dotarnos de herramientas que permitan medir la temperatura corporal de los animales y de las plantas y realizar algoritmos que nos permitan mejorar la precisión a la hora de determinar dicha temperatura.

[1] Evapotranspirar: Agua que vuelve a la atmosfera como consecuencia de la evaporación y transpiración de las plantas

1.1 Motivación

La principal motivación del proyecto es el diseño de un sistema que permita mejorar el control de enfermedades en explotaciones ganaderas y el bienestar de los cultivos en explotaciones agrícolas. Estas enfermedades causan perdidas y, si las plantas no son tratadas adecuadamente, no son productivas.

Para detectar las enfermedades es importante tener controlada la temperatura de los animales, un ejemplo es la peste porcina, en la cual el cerdo presenta temperatutras superiores a 40º C. Por lo que respecta a los cultivos, también es importante la temperatura, ya que, si una planta presenta estrés hídrico, aumentará su temperatura, dando lugar a productos finales con una calidad por debajo de lo esperado. Por tanto, nos vamos a centrar en proponer una solución a estos problemas que generan grandes perdidas económicas a la industria ganadera y agricultora.

El correcto diseño del sistema que se propone, constituiría un empuje a un sector que, generalmente, obtiene poco beneficio del producto final debido a aranceles, impuestos y otros tipos de gastos. Además, gracias a la detección temprana de los problemas mentados, estaremos otorgando a los ganaderos y agricultores la posibilidad de evitar diezmar la población de animales y evitar obtener un producto cuya calidad no es la esperada, respectivamente.

Por otra parte, ya que estos sectores disponen actualmente de métodos instrusivos y manuales para la detección de estos problemas, disponer de medios tecnológicos que les ayuden a desempeñar estas labores de forma eficiente, autónoma y económica, convierte a este proyecto en una necesidad.

Partiendo de estas ideas, se trata de generar un producto final competente que llame la atención de las grandes industrias, permitiendo una reducción de costes al fabricar en masa este producto. Abaratando costes, con una gran escalabilidad y permitiendo a ambos sectores (ganadero y agricultor) la adquisición de dispositivos económicos, poniendo fin a los problemas expuestos.

1.2 Objetivos, alcance y requisitos

A la hora de desarrollar este proyecto, se seguirán unos objetivos que describen el punto de partida de cada una de las partes de las que consta el proyecto y tienen el propósito de orientar el desarrollo dejando claro el punto de partida y el punto de finalización.

El objetivo principal es diseñar sistemas embebidos IoT dotados de inteligencia capaces de medir la temperatura de animales y plantas con las siguientes características:

- Bajo coste, para que pueda ser adquirido por cualquier explotación agrícola o ganadera sin importar el tamaño de esta.
- Desatendido, para que no requiera ningún tipo de conocimiento avanzado por parte del agricultor o ganadero.
- Fácil escalabilidad, para poder llegar al máximo número de explotaciones agrícolas o ganaderas
- Precisión y fiabilidad suficiente en el rango de temperaturas objetivo, para que realmente sea útil el dispositivo y algoritmos diseñados.

Para cumplir este objetivo, será necesario realizar un desarrollo técnico que incluya:

- Uso de sensores de bajo coste.
- Uso de algoritmos que permitan fusionar la información de los sensores de bajo coste, aumentando la precisión de la medida que otorga el sistema.
- Uso de cámaras que permitan conocer donde estamos tomando la medida.
- Uso de cámara térmica para tomar la temperatura de un punto del espacio.
- Conectividad IoT para que el sistema avise cuando se requiera intervención humana.

Así, los Objetivos finales del proyecto quedan de la siguiente forma:

En relación a la fiebre en animales:

- Obj. 1: Captura de imagenes térmicas y almacenamiento de las mismas en una unidad de procesamiento.
- Obj. 2: Toma de temperatura de un punto del espacio mediante un sensor de temperatura.
- Obj. 3: Control de la temperatura del punto del espacio anterior.
- Obj. 4: Establecimiento de un punto de referencia para la calibración de la imagen térmica gracias a lo logrado en Obj. 3.
- Obj. 5: Detección de los ojos del animale mediante el uso de cámaras.
- Obj. 6: Integración de los datos de los objetivos 1 y 5 para la obtención de la temperatura del ojo en la imagen térmica corregida.
- Obj. 7: Envío de datos a un servidor remoto.

En relación al estrés hídrico

- Obj. 8: Toma de datos y almacenamiento de sensores de temperatura y humedad.
- Obj. 9: Captura y almacenamiento de imágenes térmicas.
- Obj. 10: Interpretación de datos y determinación del estado de la planta.
- Obj. 11: Envío de datos a un servidor remoto.

Respecto al alcance, se considerará exitoso el proyecto al completar los objetivos 6 y 10, lo que permitirá en un futuro la ampliación del proyecto con una base sólida.

Los requisitos se consideran alcanzados siempre y cuando se consigua que:

- El sistema sea fiable, es decir, funcione sin necesidad de ser reiniciado durante largos periodos de tiempo (varias semanas).
- Las tomas de temperaturas tengan un error inferior a 1ºC.
- El sistema sea seguro: protección de elementos hardware para evitar daños sobre personas que puedan entrar en contacto con el sistema.
- Los datos obtenidos sean accesibles y fiables para el usuario final.

2 ESTADO DEL ARTE EN MEDIDAS DE BAJO COSTE

La toma de temperatura en animales y la determinación del estrés hídrico en plantas, son uno de los grandes temas que suscitan interés hoy día en la industria ganadera y agricultora. En sendos casos, no existe ninguna solución automatizada que permita a los interesados mantener un control de estos factores.

Si bien es verdad, estos problemas no se encuentran en igualdad de desarrollo, por ello, vamos a dedicar un apartado a cada uno.

2.1 Toma de temperatura en animales

Por lo que respecta a la toma de temperatura de los animales, encontramos una casuística muy desarrollada y más orientada a la solución concreta de nuestro problema.

Tradicionalmente, la toma de temperatura en animales, se ha hecho de forma intrusiva, esto es, mediante el uso de termómetros rectales. Esa solución tiene varios inconvenientes: Es intrusivo y requiere de un veterinario. Es decir, la medida puede no ser del todo fiable debido a que estresamos al animal y, además, supone una dependencia en el horario de trabajo del veterinario, así como el elevado coste que puede suponer la visita del mismo a la explotación.
En adición a esto, con el COVID-19, surgió interés por dispositivos capaces de detectar la temperatura de individuos. Los más utilizados fueron los escáneres de temperatura, como el mostrado en la figura 2-2. Sin embargo, la precisión de la medida obtenida es de muy poca calidad, debido a que, estos elementos suelen apuntarse a la frente para obtener dicha medida. En esa zona, acorde al profesor Tipton de la Universidad de Portsmouth [6], "los escáneres no dan una medida precisa", además, "la temperatura de la piel puede cambiar independientemente de la temperatura corporal profunda". Esto es obvio, pues la mayor parte del cuerpo presenta grasa que nos aisla de la temperatura exterior, el cuerpo suda para intentar regular la temperatura....

Figura 2-1. Escáner de temperatura.

Por tanto, el método que se propone como más efectivo es la toma de temperatura en función de una media entre la temperatura del ojo del sujeto y la temperatura del dedo del sujeto. Así, nos podremos hacer una idea del estado real del sujeto sin entrar en contacto con el mismo.
En el caso de los animales, las zonas idóneas para realizar estas medidas, son los ojos. De esta manera, tomando una media de la temperatura del ojo, podremos determinar la temperatura real del sujeto.

Por otra parte, sabemos que, hoy día, existen algoritmos capaces de detectar caras y ojos tanto de animales como de humanos. Estos tipos de algoritmos, están basados en redes neuronales que son capaces de discernir unas características en una serie de datos que se usan para entrenar a la red neuronal. No obstante, estos métodos son laboriosos de hacer y requieren de amplios conocimientos matemáticos, por lo que surgieron algunas formas adicionales de detectar características, por ejemplo, en fotos. Un ejemplo de estos nuevos métodos son las cascadas de Haar.

Los algoritmos basados en cascadas de Haar, extraen las características que queremos detectar a partir de imagenes positivas y negativas, esto es, que presentan en la imagen el objeto que buscamos o que no lo presentan. Así, cada característica es un valor individual obtenido restando la suma de pixeles bajo rectángulo blanco de la suma de píxeles bajo rectangulo negro. Estos rectángulos se establecen alrededor de una serie de núcleos y reciben el nombre de clasificadores de Haar, figura 2-3. Así, por cada núcleo de la imagen, se aplicará ese procedimiento, lo que da lugar a un gran número de características y un gran coste computacional. El alto coste computacional, se consiguió reducir gracias al uso de imagenes integrales, que simplifican la suma de rectangulos.

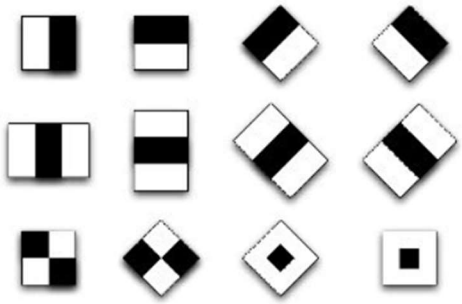

Figura 2-2. Clasificadores de Haar

Sin embargo, seguimos teniendo un gran número de características, bien, este problema, se soluciona usando

el clasificador Adaboost [7]. Este clasificador consiste en aplicar todas y cada una de las características de haar en todas las imagenes de entrenamiento y, para cada característica, encuentra el mejor umbral que separa a las imagenes positivas de las negativas. Este proceso se hace iterativamente hasta conseguir reducir el error a tasas que se consideren aceptables.

2.2 Estrés Hídrico

El problema de toma de temperaturas en plantas y determinación del estrés hídrico es un problema poco definido, esto es, hay muchas investigaciones que apuntan a posibles soluciones que pueden determinar si una planta padece o no este déficit.

Convencionalmente, se han tomado medidas de humedad de la tierra del cultivo, asumiendo la retención uniforme del agua en la zona del cultivo., por lo que se añadieron medidas de variables meteorológicas. Más adelante, surgieron modelos de evapotranspiración (ET), usados para predecir como afectan, los cambios en parámetros del tiempo, al estado del agua de la planta. Destacan el modelo de Penman-Monteith y el modelo de Hangreaves [4].
Sin embargo, estos metodos quedan antiquados en una época donde disponemos de grandes variedades de sensores a bajo coste. Además, requieren el cálculo de muchas variables, obteniendo solo una estimación del estado de la planta.

De esta manera, gracias a la sensorización, surgen métodos o índices capaces de determinar a partir de pocos parámetros si una planta sufre estrés hídrico. Entre estos, destacan los métodos infrarrojos, donde, basandonos en medidas de temperatura proporcionadas por bolómetros y algunas medidas de temperatura, somos capaces de determinar el estado de la planta. El método infrarrojo más destacado, debido a su sencillez, es el CWSI, que ya vimos en la introducción.

Hay algunos estudios que apuntan hacia el índice de vegetación de diferencia normalizada (NDVI). Este índice estima la cantidad, calidad y desarrollo de la vegetación en base a la radiación emitida por un cultivo o conjunto de cultivos [5]. Sin embargo, no se considera relevante ya que la mayoría de formas de obtención del NDVI se hacen mediante fotos proporcionadas por satélites, lo cual excede los límites del proyecto. En la Figura 2-1 podemos ver un ejemplo de una foto tomada por satelite a la que se le ha aplicado el cálculo NVDI. Las zonas con colores mas claros (verde y amarillo), representan zonas donde hay vegetación frondosa. Las zonas representadas por colores mas oscuros (rojo y naranja), destacan por la ausencia de vegetación.

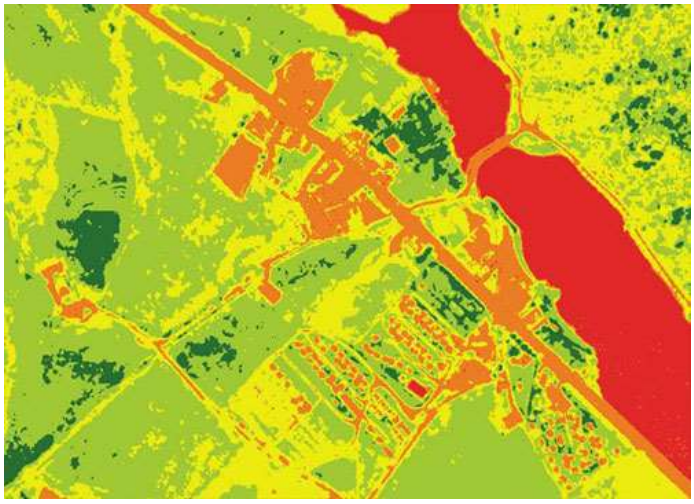

Figura 2-3. Foto de un satélite tras aplicar el NDVI.

Como conclusión, pocos estudios han llevado a la practica métodos basados en sensores. La mayoría de implementaciones se basan en el NDVI, puesto que, la mayoría de agricultores, tienen terrenos amplios,

haciendo más comodo una implementacion de ese estilo. Por tanto, el realizar un sistema que utilice sensores para determinar el estrés hídrico es algo necesario para agricultores que no tengan acceso a imagenes satelitales y busquen una solución de coste inferior.

2.3 Conclusiones comunes

Debido a los problemass mentado, hoy día, se proponen soluciones basadas en cámaras térmicas, también conocidas como bolómetros, obteniendo una solución no intrusiva y de bajo coste, para la toma de temperatura tanto de plantas como animales. Estos dispositivos captan infrarrojos, lo que nos ayudará a determinar la temperatura de zonas específicas de la imagen que toman. Sin embargo, el principal inconveniente es que la temperatura captada por la cámara, de un punto concreto del espacio, puede oscilar hasta 10º C respecto a su valor real. Adicionalmente, este error no es lineal, es decir, para cada temperatura, el bolómetro tendrá un error.
Con objeto de mostrar al lector este problema, se realizó un ensayo usando un bolómetro, modelo FLIR Lepton. Este ensayo consiste en, calentar agua hasta unos 35ºC y dejarla enfriar. Más tarde, se añaden hielos para ver el comportamiento a bajas temperaturas. También se tomó la temperatura del recipiente de agua con un sensor de temperatura PT100. Se obtuvo, la figura 2-4, donde se observa que a temperaturas cercanas a los 30º C el error puede ser inferior a 1 º, mientras que, para bajas temperaturas, el error aumenta.

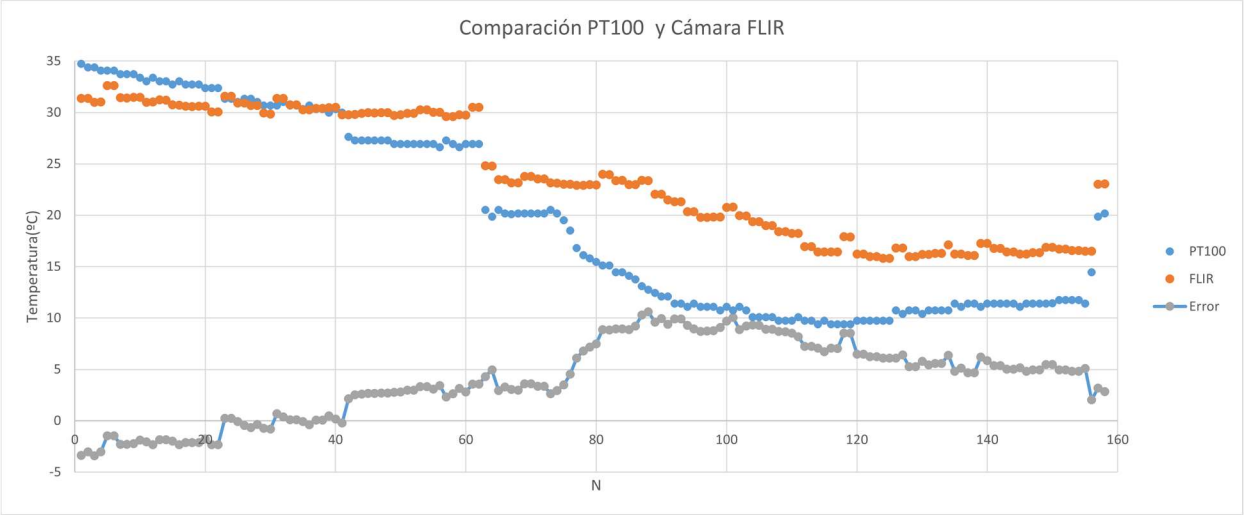

Figura 2-4. Ensayo Bolómetro

Esto, hace necesario el uso de sensores adicionales que nos permitan saber una medida de temperatura más cercana a la real y, así, poder calibrar la imagen que proporciona el bolómetro. De esta manera, con el uso de estos elementos y algunos algoritmos, podremos detectar la temperatura de un punto de interés.

3 ARQUITECTURA

En este apartado, vamos a proponer una arquitectura para el sistema que queremos diseñar. Para ello, nos vamos a basar en la búsqueda de elementos que nos permitan obtener medidas precisas, pero que, además, sean de bajo coste. Adicionalmente, buscaremos que dichos elementos, nos guíen a un sistema final, que pueda trabajar de forma autónoma y, por tanto, sea capaz de enviar y recibir datos de una red.

Para comenzar, necesitaremos sensores que nos proporcionen los datos mencionados en apartados anteriores, como son: la temperatura y humedad. Adicionalmente, vamos a necesitar una cámara térmica y una cámara normal, para captar imágenes térmicas y, además, tener una imagen de partida para detección de objetos en la imagen, como el ojo del animal.

Para ser capaces de almacenar los datos que se recojan de los sensores, vamos a necesitar un sistema procesador capaz de procesar imágenes procedentes de cámaras y bolómetros. Este sistema, debe ser sencillo de programar y barato, para permitir implementar funciones avanzadas como lo es la detección del punto de interés de la imagen tomada, así como realizar cálculos matemáticos básicos necesarios para procesar la información de los sensores. También debe poder almacenar ficheros de texto, para poder guardar la información que nos sea relevante de los sensores que se implementen en el sistema completo. Como uno de los objetivos del sistema es el envío de datos a un servidor remoto, esta unidad procesadora debe ser capaz de conectarse a redes moviles o WiFi.
El sistema procesador, debe tener unas especificaciones que permitan la implementación de técnicas de visión computacional, puesto que serán necesarias para la detección de objetos.

Para captar la radiación emitida por los objetos cuya temperatura queremos determinar, usaremos un bolómetro. Como estos elementos, sin calibración, presentan grandes errores, debemos incluir algún elemento que nos permita fijar la temperatura de un punto del espacio, de manera que, sabiendo dicha temperatura, podamos calibrar la imagen térmica y, así, obtener medidas fiables. Para dicha calibración necesitaremos un sistema de control de temperatura que consista en calentar un punto del espacio visible para el bolómetro. Ese sistema de control debe ser capaz de, en un tiempo inferior a 30 minutos, aproximadamente, establecer la referencia necesaria, con objeto de permitir la toma rápida de la temperatura de un sujeto.

Adicionalmente, es de utilidad una cámara que capture el mismo entorno que el bolómetro, permitiendo la detección de objetos en dicho entorno. Es importante que este elemento, sea de configuración rápida, para que, el tiempo que tarda en tomar una foto no comprometa la velocidad del sistema completo. Además, la imagen tomada, debe tener una calidad que permita la distinción entre los diferentes elementos que aparezcan y, así, permita una fácil identificación de la zona del ojo.

Como ya hemos mencionado, necesitaremos un sensor de humedad. Esto se debe a que, para la determinación del estrés hídrico, ya que nos ayudará a determinar las condiciones que rodean al cultivo y, así, tomar decisiones de una forma mas precisa.

Finalmente, será necesario desarrollar un código capaz de implementar todos estos elementos, así como el cableado necesario para la interconexión de elementos. Es importante que, el código que almacene los datos en la unidad procesadora sea lo mas eficiente posible, maximizando el número de toma de datos posible antes de tener que recurrir a eliminar datos de mayor antigüedad.

Así, el sistema final, debe tener una arquitectura, similar a la mostrada en la figura 3-1.

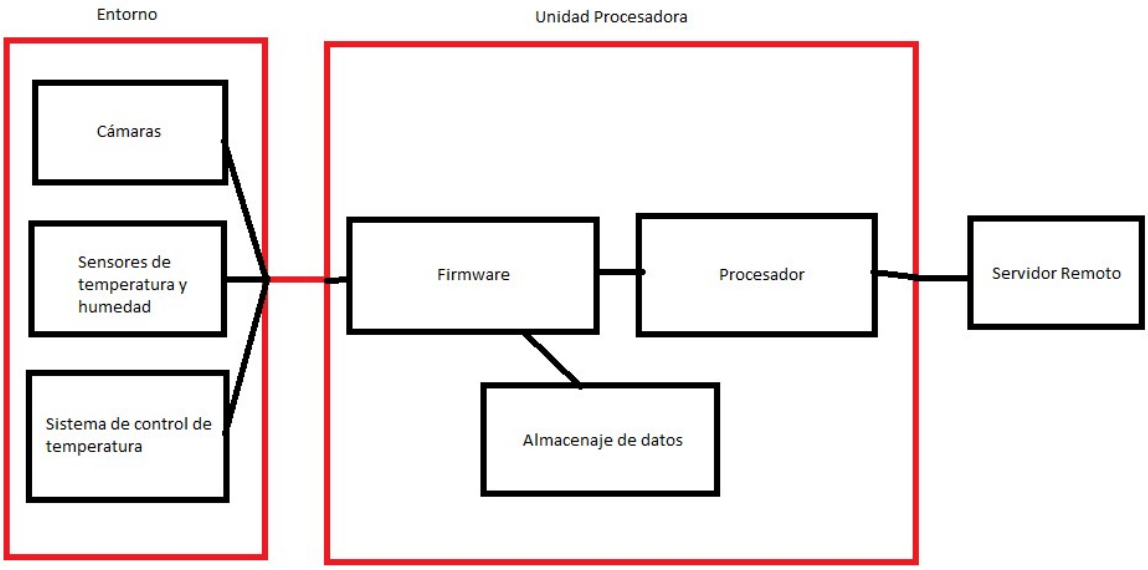

Figura 3-1. Arquitectura

4 HARDWARE

En este apartado, vamos a describir los dispositivos que se usan para proponer una solución a los problemas descritos en apartados anteriores. Además, hablaremos sobre como se conectan entre sí para lograr el montaje final. Como tenemos dos casuísticas, vamos a describir primero los elementos generales que se usan y, mas tarde, aclararemos qué se usa en cada problema. Así, vamos a hablar sobre: Cámara FLIR Lepton, módulo Purethermal mini, PT100, MAX31865, Raspberry Pi 3, una manta térmica, MOSFET driver IRF520, Cámara NoIR y DHT11.

4.1 Raspberry Pi 3

Como elemento central para procesar la información procedente de todos los sensores/dispositivos que se usen en el proyecto, vamos a usar la Raspberry Pi 3, figura 3-1.

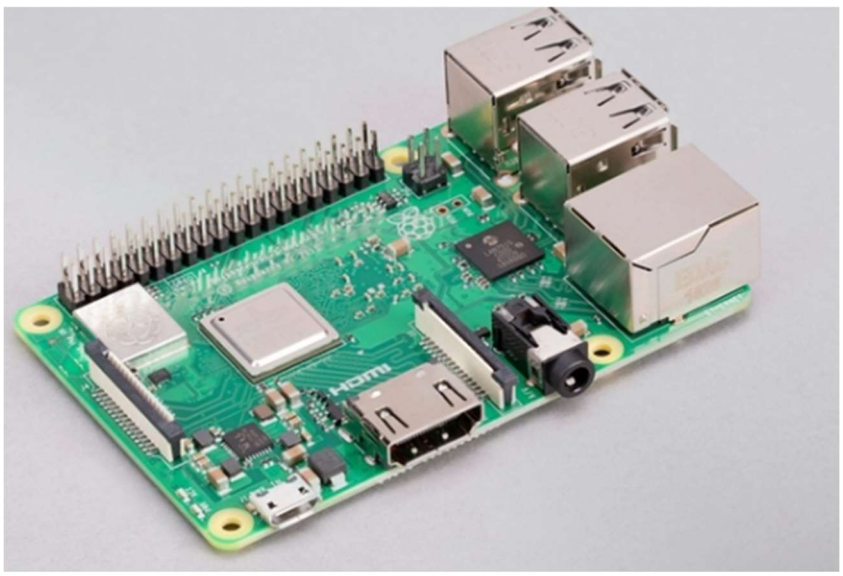

Figura 4-1. Raspberry Pi 3 Modelo B

Este dispositivo, es un computador en una sola placa que presenta conectividad LAN y Bluetooth 4.2, lo que facilitará el envío de información propuesto en el Objetivo 7. Además, como se ha observado en la imagen, la placa dispone de:

- 40 pins GPIO
- 4 puertos USB 2.0
- 1 puerto HDMI
- 1 puerto cámara CSI
- 1 puerto LAN
- 1 conexión tipo Jack

Estas características convierten al dispositivo en una solución ideal para cualquier proyecto que requiera el conexionado de un gran número de sensores. Además, destaca su bajo coste, en torno a 40€, el modelo más barato. Adicionalmente, presenta una gran capacidad de procesamiento gracias a su procesador y memoria RAM:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM

Por otra parte, cabe destacar la facilidad de programación que tiene el dispositivo, pues presenta un entorno propio basado en Linux, denominado Raspbian. Este sistema operativo nos permite conectar la raspberry a una pantalla y usarla como un ordenador. Aquí, instalaremos la última versión de python teniendo fácil acceso a todas las funcionalidades del dispositivo.

4.2 Cámara FLIR Lepton y Módulo Purethermal mini

Para captar las imágenes térmicas, hemos optado por la FLIR Lepton, figura 3-2, que es capaz de tomar la radiación que emiten los cuerpos a los que está enfocando. Para conectarnos a ella, necesitamos usar el módulo Purethermal mini, figura 3-3, que nos permite conectarnos a la cámara a través de un puerto USB.

Figura 4-2. Cámara FLIR Lepton

Figura 4-3. Módulo Purethermal mini

La comunicación con el dispositivo se hace a traves de una variante de I2C, el CCI (Common Communication Interface). El CCI, nos ayuda a establecer una comunicación, y controlarla, con la cámara Lepton, cuando acabamos de comunicarnos, también nos permite cerrar la comunicación. Este protocolo, nos permite recibir paquetes de 16 bits, por lo que, el módulo Purethermal, se encarga de dividir en paquetes la imagen que proporciona la cámara.

Por otra parte, podemos establecer una configuración de la cámara. Por defecto, tiene la radiometría activada, pero se puede desactivar accediendo a registros internos. Tener activa o no esta función, es decir, tener un modo radiométrico u otro, afectan a la función de transferencia entre el flujo incidente y el píxel de salida. Escoger activar o desactivar la radiometría no afecta a la calidad de la imagen, asegura el

fabricante [2]. Sin embargo, para aplicaciones donde pretendemos convertir la salida de la cámara en algo proporcional a la escena de la temperatura, se debe activar la radiometría. Por tanto, para nuestra aplicación, se dejará por defecto la configuración.

El fabricante hace referencia a un mecanismo de regulación de la temperatura que capta la imagen. Esto se hace cada cierto tiempo y se conoce por el nombre de FFC (Flat Field Correction). Este mecanismo es necesario para ajustar la deriva que sufre la cámara pues se calienta con el tiempo debido a su funcionamiento.

Respecto al error de la temperatura detectada por la cámara, el fabricante no hace referencia a la misma para la version 3 del producto, que es el que se va usar para este proyecto. Por lo que se realizó el ensayo mencionado en el apartado 2.3.

Ahora ya tenemos caracterizado el error de la cámara, que sabemos que va a ser reducido para temperaturas altas y aumentará conforme disminuye la temperatura.

4.3 PT100 y MAX31865

Para obtener medidas adicionales a la cámara FLIR Lepton, vamos a usar un sensor de temperatura que nos ayude a fijar la temperatura de un punto del espacio. Para ello, se escoge el sensor PT100 conectado a una placa amplificadora MAX31865. Esto se hace así, debido al bajo coste y gran fiabilidad de ambos componentes.

Respecto a la PT100, es necesario explicar que es un sensor RTD, es decir, un sensor en el que la resistencia interna varía en función de la temperatura. Este tipo de sensores tienen una gran precisión comparados con el resto de sensores de temperatura, como los termistores NTC y PTC, pues estos últimos tienen mayor utilidad a la hora de medir incrementos de temperatura. La PT100 elegida presenta las siguientes características [8]:

- Conexión de 3 cables
- 100 Ω a 30 ºC
- Variación de la resistencia con respecto a la temperatura: 0.385 Ω / ºC nominal

Una vez vistas las características del sensor, tenemos que ver como comunicarnos con él. Para ello, el fabricante, recomienda usar la placa MAX31865, que actúa como driver, permitiéndonos la comunicación mediante el uso del protocolo SPI. Para la conexión de estos dos elementos, seguimos las recomendaciones del fabricante [9], obteniendo el montaje mostrado en la figura 3-5, proporcionada por el propio fabricante. En nuestro caso, el dispositivo de la izquierda, sería la raspberry pi 3.

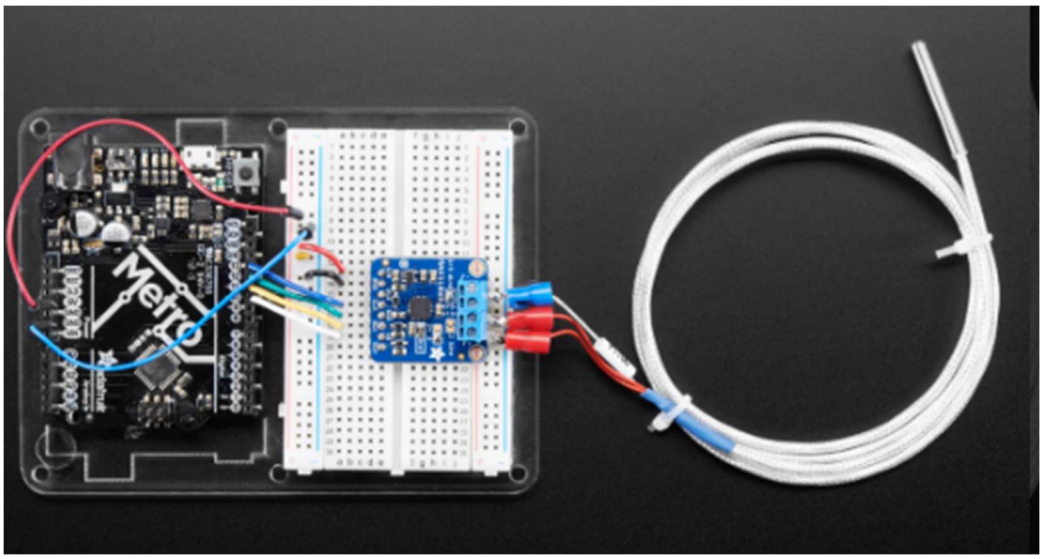

Figura 4-4. Montaje PT100 y MAX31865

4.4 Manta térmica y MOSFET Driver IRF520

Para fijar una referencia de temperatura y poder corregir la imagen que proporciona la FLIR Lepton, vamos a calentar una PT100 con una manta térmica.

Una manta térmica es, básicamente, una resistencia que calentamos mediante el paso de la corriente por la misma, es decir, aprovechamos el efecto Joule.
Si directamente conectáramos la manta térmica a la conexión de 5V de la raspberry, no podríamos controlar la temperatura, pues esa salida siempre da un voltaje constante. Si optáramos por conectar un PWM a uno de los GPIO de la Raspberry, podríamos controlar la temperatura, el inconveniente es que estos pines no son capaces de propocionar la potencia necesaria para calentar la manta térmica. Como posible solución, podemos controlar un transistor mosfet conectado en fuente común con la manta térmica conectada en el drenador y, así, usar la salida del PWM como una señal lógica que controla la puerta. Esta idea la propone el Driver IRF520, de Wingoneer [10].

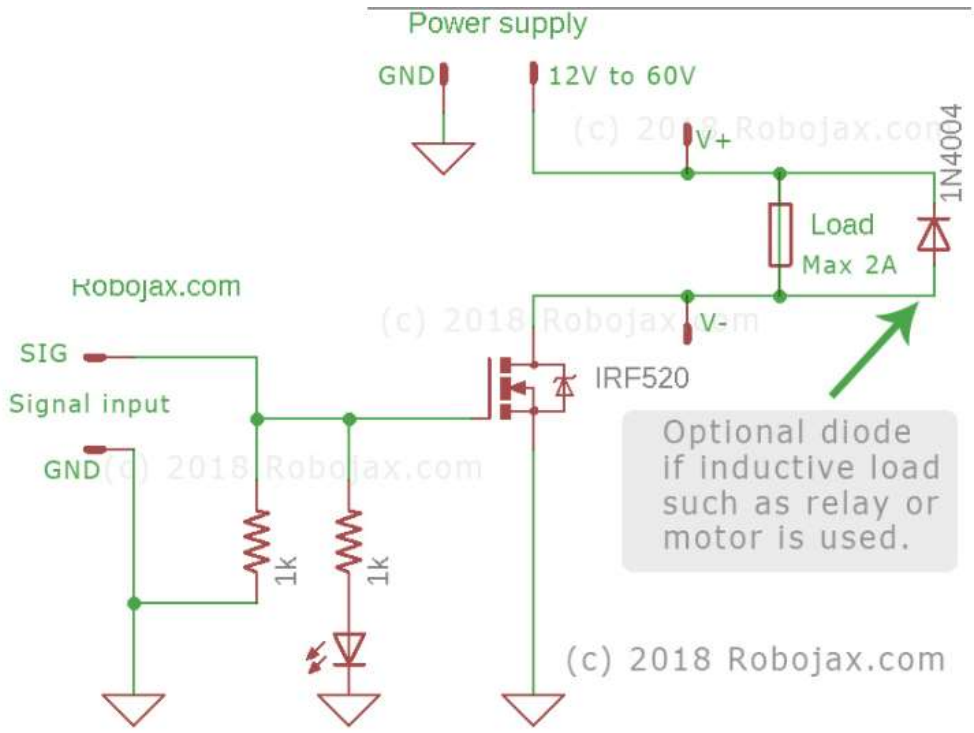

Figura 4-5. Driver IRF520

En la figura 3-6, podemos observar las conexiones del driver mencionado. Cuando no ponemos ninguna señal en "Signal input", conecta directamente la entrada a tierra para evitar comportamientos indeseados. Por otra parte, las tierras están conectadas físicamente entre sí, lo que nos facilita referenciar los voltajes al trabajar con este driver. En el caso de conectar la manta térmica, el terminal V+ irá conectado a 5V. Ya que la manta térmica usada tiene una resistencia de unos 2 Ohmios, debemos conectar una resistencia adicional para limitar la corriente que circula por la rama, disminuyendo el consumo de la Raspberry Pi 3.

Por tanto, la raspberry controlará una señal PWM en función de la temperatura que devuelva una PT100 que estará colocada encima de la manta térmica.

4.5 Cámara NoIR

Para la detección de objetos vamos a usar una Cámara NoIR, figura 3-7. Esta cámara se conecta por un puerto específico que tienen las Raspberry Pi, estas cámaras no disponen de un filtro infrarrojo, lo que las hace especialmente útiles para la observación de plantas pues, si colocamos una pequeña lámina de gel azul que nos da el fabricante, podremos ver la cantidad de luz que refleja una planta. Esto, en algunos tipos de planta, en concreto las que tienen hojas verdes, es un indicativo de la salud de la misma. Por tanto, esta cámara nos permite tanto detectar ojos para la problemática ganadera como añadir una medida en el caso de la problemática agricultora.

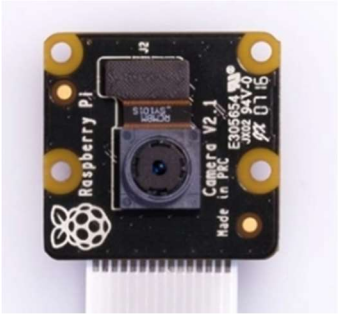

Figura 4-6. Cámara NoIR v2

En lo concerniente a la configuración de la cámara, casi todo viene configurado de fábrica por lo que no tendremos que preocuparnos. Sin embargo, podemos elegir la resolución de la imagen y, así, reducir el tamaño del archivo que genera al guardarla, a máxima resolución puede llegar a ocupar 4 MB, mientras que a mínima resolucion se queda en unos 20 kB. El fabricante destaca que, una vez establecida la configuración, debemos dejar reposar la cámara 2 segundos para que autoajuste la configuración y, tras esto, ya podremos tomar la foto.

4.6 DHT11

Para tomar medidas de Humedad Relativa (RH), usamos un DHT11, un sensor que destaca por su bajo coste, mostrado en la figura 3-8. La medida que nos otorgue, será orientativa y nos permitirá hacernos una idea del tiempo actual que hace. Además, también nos puede proporcionar una medida de temperatura por si en algún momento dudáramos de las medidas dadas por las PT100.

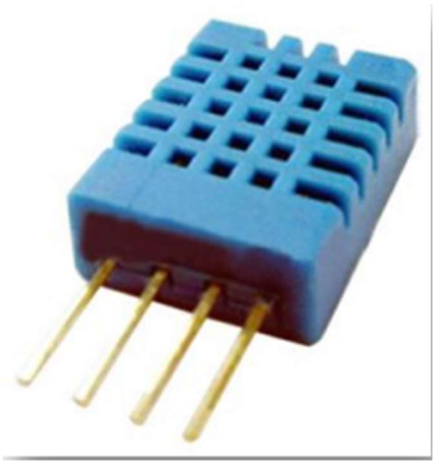

Figura 4-7. Sensor DHT11

Como hemos mencionado, este sensor presenta grandes características en comparación con su coste, algunas de estas características son:

- Resolución de 1% RH (8bits)

- Repitibilidad de la medida: +1% RH, -1%RH
- Rango de medida (25 ºC): Desde 20% hasta el 90% RH
- Tiempo de respuesta típico: 10 s

4.7 Conexiones y montaje final

Antes de comenzar, debemos resaltar que se usarán dos montajes: uno para la toma de temperatura en animales y otro para la determinación del estrés hídrico. Ambos usarán la Raspberry Pi, la manta térmica, la cámara FLIR Lepton y, al menos, una PT100.

4.7.1 Montaje común

Para la obtención del montaje común, figura 3-8, vamos a conectar a un puerto USB la FLIR Lepton. Además, añadiremos la conexion y configuración de una salida PWM para el driver IRF520, para el control de la temperatura, para el que también conectaremos una PT100 con su respectiva MAX31865.

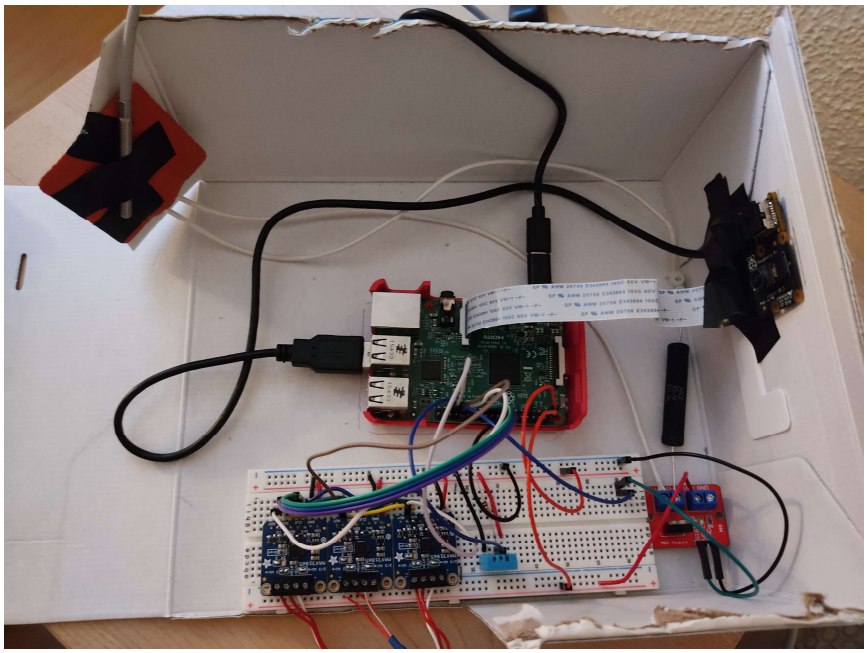

Figura 4-8. Montaje común

4.7.2 Montaje para la toma de temperatura

Respecto a la toma de temperatura, añadiremos el uso de la cámara NoIR, con un software que desarrollaremos para detección de ojos, basado en cascadas de HAAR. Se supondrá conocida la posición del animal para simplificar la ejecución del resto del proyecto.

4.7.3 Montaje para determinar el estrés hídrico

Para determinar el estrés hídrico, añadiremos al montaje común un sensor DHT11 y 2 PT100 adicionales, una para tomar la temperatura de la tierra y otra para tomar la temperatura del aire. Adicionalmente, usaremos la cámara NoIR por si en algún futuro se quisiera desarrollar alguna utilidad relacionada con la cantidad de luz que refleja la planta o por si se añadiera un algoritmo de detección de hojas. En nuestro caso, pondremos la cámara lo suficientemente cerca de la hoja como para que todo lo recogido por la cámara sean hojas.

5 FIRMWARE

Ahora, vamos a describir las partes mas importantes del firmware que se implementará en el hardware descrito anteriormente para conseguir un correcto funcionamiento del sistema.

Antes de describir el código desarrollado, es menester detallar las librerías usadas por el mismo:

- Libuvc: Usado para el envío de datos mediante el protocolo UVC (USB Video Class)
- Numpy: Permite el uso de funciones matemáticas generales, como el uso de matrices, vectores...
- Logging: se usa para el almacenaje de información, errores y advertencias generadas durante la ejecución del programa, en un documento de texto.
- Matplotlib: usado para representaciones gráficas, como la muestra de imágenes.
- PiCamera: permite la comunicación con la cámara NoIR de forma simplificada.
- OpenCV (Open Source Computer Vision Library): implementa los algoritmos de visión computacional que nos permiten la detección de objetos, seguimiento de los mismos y el tratamiento de las imagenes tomadas por las cámaras.
- Adafruit MAX31865: simplifica la comunicación con la placa MAX31865.
- Adafruit DHT: implementa las funciones necesarias para comunicarnos con el sensor DHT11.

5.1 Obtención de datos

En la obtención de datos, podemos destacar dos grupos principales: las imágenes tomadas por las cámaras y los datos procedentes del resto de sensores que constituyen el sistema.

5.1.1 Captura de imágenes

Por lo que respecta a la captura de información procedente de la cámara FLIR Lepton y la cámara NoIR, debemos describir dos escenarios diferentes. La primera, nos permite capturar imágenes mediante una conexión USB, como ya mencionamos en el apartado anterior. La segunda, captura las imágenes de forma "autónoma" gracias a librerías python proporcionadas por el fabricante.

5.1.1.1 Cámara FLIR

Para la comunicación con la cámra térmica, usaremos la libreria libuvc. Como se ve en la figura 5-1, podemos configurar la cámara antes de pedir que nos envíe las fotos que tome mediante el USB. Si la configuración se ha hecho de forma correcta, empezará la toma de imágenes. No vamos a detallar las funciones usadas para la configuración ni las variables que intervienen ya que alargaría la memoria y nos alejaríamos del tema de interés de estos apartados.

```
#Thermal camera Stream control
libuvc.uvc_get_stream_ctrl_format_size(
      devh, byref(ctrl), UVC_FRAME_FORMAT_Y16,
      frame_formats[0].wWidth, frame_formats[0].wHeight,
      int(1e7 / frame_formats[0].dwDefaultFrameInterval)
      )
#Save stream in result
res = libuvc.uvc_start_streaming(devh, byref(ctrl),
                                 PTR_PY_FRAME_CALLBACK, None, 0)
if res < 0:
  logging.error("uvc_start_streaming failed: {0}".format(res))
  exit(1)
else:
  logging.info("UVC started")
```

Figura 5-1. Configuración Cámara FLIR

Los datos que proporcionan las funciones anteriores han de ser almacenados en un vector, para, posteriormente comprobar que no presentan errores. Esto se hace mediante la funcion minMaxLoc, figura 5-2, que comprueba que todos los valores obtenidos estén dentro de unos márgenes, donde se considerarán correctos. Cabe destacar que los datos obtenidos están en grados Kelvin y, para mayor comodidad al interpretar los datos, trabajaremos con grados Celsius.
Finalmente, los datos se almacenan en una matriz, para su posterior corrección con los datos que obtengamos del sensor de temperatura usado con la manta térmica.

```
if data is None:
  logging.error("Error al leer la cámara térmica")
  exit(1)
else:
  logging.info("datos leídos")

minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(data)
while minVal == 0 and maxVal == 0:
  logging.warning("no values from thermal camera, new attempt:stop and start UVC")
  libuvc.uvc_stop_streaming(devh)
  time.sleep(3) # wait 3 seconds to retry
  res = libuvc.uvc_start_streaming(devh,
                                   byref(ctrl),
                                   PTR_PY_FRAME_CALLBACK,
                                   None, 0)
  if res < 0:
    logging.error("uvc_start_streaming failed: {0}".format(res))
    exit(1)
  data = q.get(True, 500)
  if data is None:
    logging.error("Error reading thermal camera data (no data)")
    exit(1)
  minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(data)
  logging.warning("new attempt done")
logging.info("Value loaded")
```

Figura 5-2. Uso de la función minMaxLoc

5.1.1.2 Cámara NoIR

Primero, como ya explicamos en la parte de Hardware, debemos configurar la cámara. La configuración elegida para el proyecto es tal que no perdemos detalle en la imagen, pero tampoco ocupa demasiado tamaño en memoria, permitiendo así, el almacenaje de un número eleveado de imagenes.

Tras esto, procedemos a almacenar la imagen, usando para ello, la librería openCV. El código implementado para esta rutina de captura de imágenes se observa en la figura 5-3.

```
camera = PiCamera()
camera.rotation = 180

print("Inicializando...")

#ajuste de resolucion max: 2592,1994; min: 64,64
camera.resolution = (720, 480)
sleep(2)
print("Tomando foto...")
camera.capture('/home/pi/Desktop/image.jpg')
```

Figura 5-3. Configuración cámara NoIR

5.1.2 Obtención de datos de los sensores de humedad y temperatura

Tras capturar las imágenes, necesitamos recoger los datos que tienen todos los sensores que hemos implementado en el sistema.

Empezando por los sensores de temperatura PT100, vamos a detallar la configuración mostrada en la figura 5-4. Para poder comunicarnos, debemos definir una comunicación SPI. Para ello hay que seleccionar el pin GPIO de la Raspberry que necesitamos usar, denominado CS (Chip Select). Además, necesita que configuremos un protocolo SPI, al que le otorgamos un reloj interno de la Raspberry, SCK. Finalmente, necesita saber que pin vamos a usar como MISO (Master Input Slave Output) y cual como MOSI (Master Output Slave Input).

Una vez definido nuestro protocolo SPI, pasamos dicha información a la función proporcionada por el fabricante para leer los datos del sensor, esto es: adafruit_max31865.MAX31865.
Adicionalmente, debemos especificar que resistencia de referencia usa la placa MAX31865 y que resistencia nominal tiene nuestro sensor, en este caso 430 y 100 Ω, respectivamente.

```
spi = busio.SPI(board.SCK, MOSI=board.MOSI, MISO=board.MISO)
cs1 = digitalio.DigitalInOut(board.D5) #Chip select of the MAX31865 board
sensor1 = adafruit_max31865.MAX31865(spi, cs1, wires=3, rtd_nominal=100, ref_resistor=430)

while True:
    #Obtain temperature
    print('Temperature 1: {0:0.3f}C'.format(sensor1.temperature))

    time.sleep(5.0)
```

Figura 5-4. Código PT100

Tras configurar los protocolos, simplemente, debemos llamar a la función adafruit_max31865.MAX31865.temperature para obtener los datos y almacenarlos.

Una vez obtenidos los datos de los sensores de temperatura, necesitamos la información del sensor de humedad. Para ello, debemos usar la función DHT11 de la librería proporcionada por el fabricante para elegir el pin que vamos a usar para el protocolo de comunicación.
Respecto a la comunicación con el dispositivo, el fabricante usa un protocolo propio, descrito en la hoja de datos del dispositivo [11]. La Raspberry Pi debe iniciar la comunicación mandando una señal de inicio, despertando al DHT11, esté responderá mandando 40 bits de información, donde contiene: 8 bits para el número entero de Humedad Relativa, 8 bits para los decimales de la Humedad relativa, 8 bits para el numero entero de la temperatura, 8 bits para los decimales de la temperatura y una suma de comprobación para determinar que haya sido correcta la transmisión.
Se distinguen los 1 y 0 lógicos en función de la duración de los pulsos.

Cabe destacar, que este protocolo es propenso a fallar, por lo que se ha incorporado un algoritmo que evita que deje de ejecutarse el programa si hemos recibido una lectura incorrecta. Todo esto, se observa en la figura 5-5.

```
dhtDevice=adafruit_dht.DHT11(board.D19)
while True:
    try:
        # Print the values to the serial port
        temperature_c = dhtDevice.temperature
        temperature_f = temperature_c * (9 / 5) + 32
        humidity = dhtDevice.humidity
        print(
            "Temp: {:.1f} C    Humidity: {}% ".format(
                temperature_c, humidity
            )
        )

    except RuntimeError as error:
        # In case of errors, continue executing
        print(error.args[0])
        time.sleep(2.0)
        continue
    except Exception as error:
        dhtDevice.exit()
        raise error

    time.sleep(2.0)
```

Figura 5-5. Rutina DHT11

5.2 Detección de ojos

Como ya hemos mentado, el ojo es una zona muy fiable para obtener la temperatura de un animal, es por ello, que debemos ser capaces de saber donde está dicho ojo al examinar una imagen. Esto es posible hacerlo gracias a las cascadas de haar. Por evitar una memoria demasiado extensa, no vamos a profundizar mucho en esta parte y vamos a explicar lo básico para que el lector entienda el funcionamiento del código.

Las cascadas de haar, tratan de encontrar una serie de características en la imagen, para ello, se basan en encontrar los filos de dicha imagen. Por tanto, el color de la imagen no es importante y podemos reducir el tamaño de la foto capturada si la guardamos en blanco y negro sin afectar al resultado de la detección. Tras esto, aplicamos un filtro que añade blur a la imagen, evitando así la detección de ojos "falsos" fuera de nuestra zona de interés, el sujeto frente a la cámara. Esto queda realizado según el código de la imagen 5-6.

```
#Convert img to gray scale
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY)
#apply blur
blur = cv.GaussianBlur(gray, (3,3), cv.BORDER_DEFAULT)
#display image
cv.imshow('blurred image', blur)
```

Figura 5-6. Preprocesado de la imagen

Una vez tenemos preparada la imagen, pasamos a leer el fichero xml de las cascadas de haar, donde tenemos una inteligencia artificial ya preentrenada. Ahora, con el uso de la función haar_cascade.detectMultiScale, detectamos los ojos de la imagen, previamente procesada. Las variables scaleFactor y minNeighbours son relevantes en lo respectivo a la sensibilidad del algoritmo al detectar ojos. Finalmente, como vemos en la figura 5-7, guardamos las variables donde hemos encontrado el ojo para, posteriormente pasar las coordenadas a la cámara térmica y que obtenga la media de temperatura de esa zona.

```
#Access xml file
haar_cascade = cv.CascadeClassifier('eye.xml')
#Search for eyes
eyes_found= haar_cascade.detectMultiScale(img, scaleFactor = 1.5, minNeighbors=3)
print(f'Numero de ojos encontrados = {len(eyes_found)}')

#Draw a rectangle where the eyes are
for (x,y,w,h) in eyes_found:
    x_i=x
    y_i=y
    x_f=x+w
    y_f=y+h

    cv.rectangle(img, (x_i,y_i), (x_f, y_f), (0,255,0), thickness=2)
```

Figura 5-7. Detección de ojos

5.3 Control de temperatura

Para fijar la temperatura de un punto del espacio, vamos a calentar una de las PT100 con una manta térmica, como ya hemos explicado anteriormente. El control de esta temperatura, se hace gestionando una salida PWM de la Raspberry, tomando como señal de control el Duty Cycle (DC) de esta salida. La salida del sistema, será la temperatura que reporte la PT100 mencionada, en grados Celsius. Y, el error, será lo lejos que estemos de la temperatura de referencia.

Como bien sabemos de experimentos realizados para determinar el error de la cámara térmica, cuanta mayor temperatura tengamos, menor será el error en ese punto. Por tanto, buscamos corregir la imagen, sabiendo la temperatura de el punto en el que se encuentra la PT100. Debido a que el error no es lineal, no quiere decir que el resto de puntos de la imagen estén a la temperatura real una vez que se corrija dicha imagen, pero si podremos suponer que los puntos cuya tempertura sea similar a la marcada por la PT100 estarán cercanos al valor de referencia fijado en el código.

Analizando el sistema de la manta térmica y la PT100, nos damos cuenta de que realizar un control dinámico no es rentable puesto que solo nos interesa que lleguemos al punto de referencia, no nos importa el proceso intermedio, pues una vez estemos en torno a un valor de temperatura, solo habrá que mantenerlo. Es por ello, que se considera que implementar un controlador tipo PI aumentaría el coste computacional del sistema sin añadir ningún tipo de valor al funcionamiento del mismo. Por tanto, se ha decidido implementar un control por tramos, de manera que si nos encontramos lejos del valor de referencia actuaremos con una señal de control cercana al 100% de DC. Si estamos llegando al valor o nos encontramos con un margen de 1º C respecto a la referencia, intentaremos mantener la temperatura, con un 50 ó 60% de DC. Este control, se observa en la figura 5-8.

```
#Config PWM for thermal sheet
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(12,GPIO.OUT)

signal = GPIO.PWM(12,50)
duty = 90
signal.start(duty)

while True:
    #Save temperature
    print('Temperature 1: {0:0.3f}C'.format(ptl00.temperature))
    temp = open("temp.txt", "a")
    dc = open("duty.txt", "a")
    temp.write("{0:0.3f}\n ".format(ptl00.temperature))
    dc.write("DC = %d \n" %(duty))

    temp.close()
    dc.close()
    if ptl00.temperature > 41:
        duty = 20
    elif (ptl00.temperature <= 41) & (ptl00.temperature >=40):
        duty = 60
    else:
        duty = 100
    signal.ChangeDutyCycle(duty)
    time.sleep(30.0)
```

Figura 5-8. Control de temperatura

Destacar que, debido a la dinámica del sistema, podemos actualizar el valor cada 30 segundos sin perder información relevante.

5.4 Corrección del error y almacenaje de resultados

Por último, una vez tenemos todos los datos de las cámaras y sensores, tenemos que proceder con la corrección de la imagen del bolómetro y el almacenaje de dicha imagen, la temperatura de los distintos sensores, la humedad y las variables usadas para el control de la manta térmica.

Para corregir el error de la camara térmica, debemos ver primero que valor está tomando el bolómetro en el punto donde tenemos colocado el control de temperatura. Tras esto, sabiendo el valor real de temperatura de ese punto, calculamos el error cometido en dicha zona. Así, aplicamos la corrección como vemos en la figura 5-9.

```
print("\nMedia foto:",np.mean(fever),"\nMáx foto:",np.max(fever))
x1,x2,y1,y2=71,74,58,68 #Coordenadas del cuerpo negro
bbox=np.zeros((x2-x1,y2-y1))
bbox=fever[y1:y2,x1:x2] #Matriz contenida en el cuerpo negro
print("Tipo bbox:",type(bbox),"Dimension:",bbox.shape,"\n")
print(bbox)
error=np.mean(bbox)-tempBB
print("\n############",np.mean(bbox),"##################\n")
fever_OK = fever -error
```

Figura 5-9. Corrección de la temperatura de la imagen

Así, podemos ahora obtener la media de temperatura en las coordenadas donde hemos detectado el ojo, figura 5-10. Esta media se guarda en una variable para, más tarde, poder decidir el estado del animal. Hay que destacar que, en el caso de analizar cultivos, este paso se omite, ya que no tenemos que detectar ningún ojo, por lo que obtendremos la media de la temperatura de la hoja de la planta.

```
# Calculamos la temperatura en los pixeles interesantes
eye=np.zeros(pixel_w,pixel_h)
eye=fever[pixel_y:(pixel_y+pixel_h),pixel_x:(pixel_x+pixel_w)]
body_temp = np.mean(eye)-error
```

Figura 5-10. Temperatura media del ojo

Tras haber corregido el error del bolómetro, tenemos que almacenar todos los datos. Para ello, comenzamos con guardar las imágenes. En este paso, nos ayudamos de la librería matplotlib, donde dibujaremos dos gráficas: una con la temperatura sin corregir y otra con la temperatura corregida, este código se muestra en la figura 5-10. Finalmente, en el título de las gráficas, encontraremos información relevante como puede ser: error cometido por el bolómetro, media de temperatura de la región de interés y media de temperatura de la zona del cuerpo negro, es decir, la PT100 del control de temperatura.

```
fig, (ax,ax1) = plt.subplots(1,2, sharey=True,
                             constrained_layout=True,
                             figsize=(10,5))
ax.invert_yaxis()  # como están igualados afecta a ambos ejes verticales
logging.info("Plot ready")

fig.suptitle("Analysis of:" + data_name
             ,fontsize=12)
pcm = ax.pcolormesh(fever, # Imprime la matriz con los datos en bruto
                    rasterized=True,
                    vmin=np.min(fever_OK),
                    vmax=np.max(fever_OK)+error,
                    cmap='jet')
pcm2 = ax1.pcolormesh(fever_OK,  #  matriz fiebre menos error.
                    rasterized=True,
                    vmin=np.min(fever_OK),
                    vmax=np.max(fever_OK),

                    cmap='jet')
fig.colorbar(pcm2, shrink=0.95, extend='both',
             label='Temperature ($^oC$)')
```

Figura 5-11. Almacenaje de gráficas

Ahora solo nos queda guardar la información relevante del resto de sensores. La información procedente del control de temperatura se guardará en dos archivos de texto separados. En uno, se almacena el dutycycle usado en cada momento y, en el otro, se almacenará la temperatura que marcaba el sensor.

En otro fichero de texto a parte, guardamos la información relevante para el estrés hídrico como es: la humedad, la temperatura del aire y la temperatura de la tierra. Estos códigos se observan en la figura 5-11.

```
humidity = dhtDevice.humidity
temp_dht11 = dhtDevice.temperature
print("Humidity: {}% Temp DHT11: {:.1f} C".format(humidity, temp_dht11))
print('Air Temperature: {0:0.3f}C'.format(pt100_air.temperature))
print('Ground Temperature: {0:0.3f}C'.format(pt100_ground.temperature))
data = open("data.txt", "a")
data.write("Air temp: {0:0.3f} C, Ground temp: {0:0.3f} C, Humidity: {}% \n".format(pt100_air.temperature, pt100_ground.temperature, humidity) )
```

Figura 5-12. Almacenaje datos cultivo

6 RESULTADOS Y CONCLUSIONES

Con el estado actual del sistema, somos capaces de generar resultados tomando medidas de temperatura tanto en plantas, animales como personas. Debido al COVID-19, no se han podido realizar ensayos sobre animales, pues esto requeriría de desplazamientos que se podían ver afectados por las restricciones de movilidad marcadas por la situación de cada municipio.

Por ende, las pruebas del sistema completo se han realizado sobre personas. Por otra parte, esto nos facilita la toma de datos y un diseño de experimentos con parámetros más controlados. Respecto al framework montado para la detección de enfermedades en cultivos, se probará que el almacenaje de datos se realiza correctamente, permitiendo así la futura interpretación de estos.

6.1 Resultados finales

Antes analizar resultados, debemos primero mencionar como ha quedado el almacenaje de datos, para que el lector sea capaz de interpretar resultados. Se dispone de los siguientes documentos:

- Un archivo de texto donde almacenamos la información relevante de los cultivos, esto es: temperatura y medidas de humedad.
- Dos archivos de texto para guardar la información relevante del control de temperatura y así poder juzgar la bondad de dicho control.
- Un archivo pdf donde mostramos la imagen tomada por la cámara térmica, la imagen corregida e información concerniente a la temperatura del cuerpo negro y de la región de interés.

Los archivos correspondientes al cultivo y control de temperatura, se mostrarán mas adelante, cuando se analizen dichos datos puesto que, debido a la simplicidad de su almacenaje, no necesita explicación.

Por lo que respecta a las imágenes tomadas, tenemos que destacar que los nombres de los archivos vienen dados por la fecha en la que se tome la foto, siguiendo el formato:

$$data/\%día/\%mes/\%año/\%hora/\%minutos/\%segundos$$

En la figura 6-1, podemos ver uno de los ejemplos de fotos tomadas sobre personas. En la parte superior de la imagen, observamos la ruta completa donde se encuentra el archivo en cuestión.

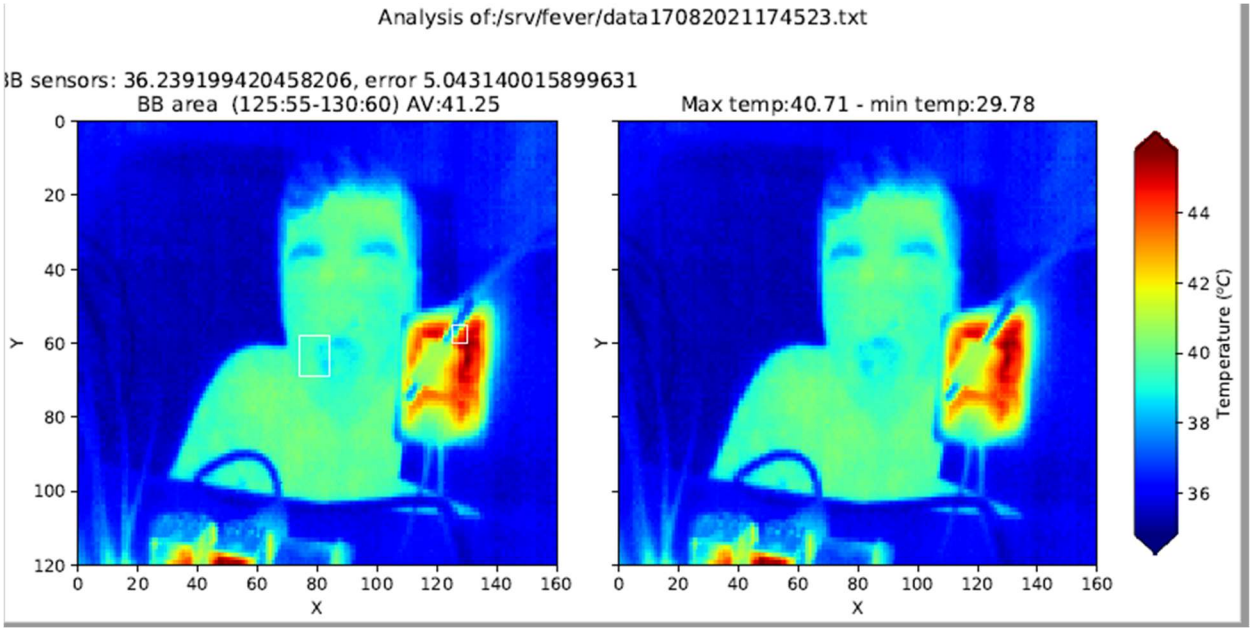

Figura 6-1. Ejemplo resultado

La información que obtenemos de los gráficos es la siguiente:

- Error cometido por el bolómetro en grados Celsius (error)
- Temperatura que marca la PT100 del control de temperatura (BBsensors)
- Temperatura máxima en la imagen (Max temp)
- Temperatura mínima en la imagen (Min temp)
- Media de temperatura en el cuadrado donde se debe encontrar la PT100 (AV)

Así con estos datos, podremos observar el estado real del sujeto y, adicionalmente, comprobar si los datos tomados tienen sentido. Por otra parte, por si fuera de interés se observa el error que cometía inicialmente la cámara térmica, esto puede servir para observar si el error se mantiene si cambiasemos de cámara, por ejemplo.

6.2 Análisis de resultados

Debido a que el sistema consta de partes bien diferenciadas cuyos rendimientos no estan relacionados de forma directa, vamos a dedicar un apartado a cada una de las partes y, finalmente, veremos como queda el sistema completo.

6.2.1 Análisis del control de temperatura

Por lo que respecta al control de temperatura implementado, tal y como se comentó en el apartado de Firmware, se ha buscado alcanzar la referencia en régimen permanente, obviando la dinámica del sistema. Sabemos que, aumentando la acción de control cuando nos encontramos por debajo de la referencia, obtenemos una dinámica más rápida. Sin embargo, esto último genera un aumento considerable del consumo del sistema cuando estamos en régimen transitorio. Un consumo elevado no es deseable, pues este sistema se pretende que sea lo más económico posible. Adicionalmente, si se llegara a implementar con baterías, también

buscaríamos un consumo que maximize la vida de dicha batería. Así, se ha buscado un punto de compromiso, donde el consumo es bajo, pero mantiene una dinámica que permite llegar a los 35º C en un tiempo inferior a los 20 minutos. De esta manera, terminamos obteniendo el sistema mostrado en la figura 6-2.

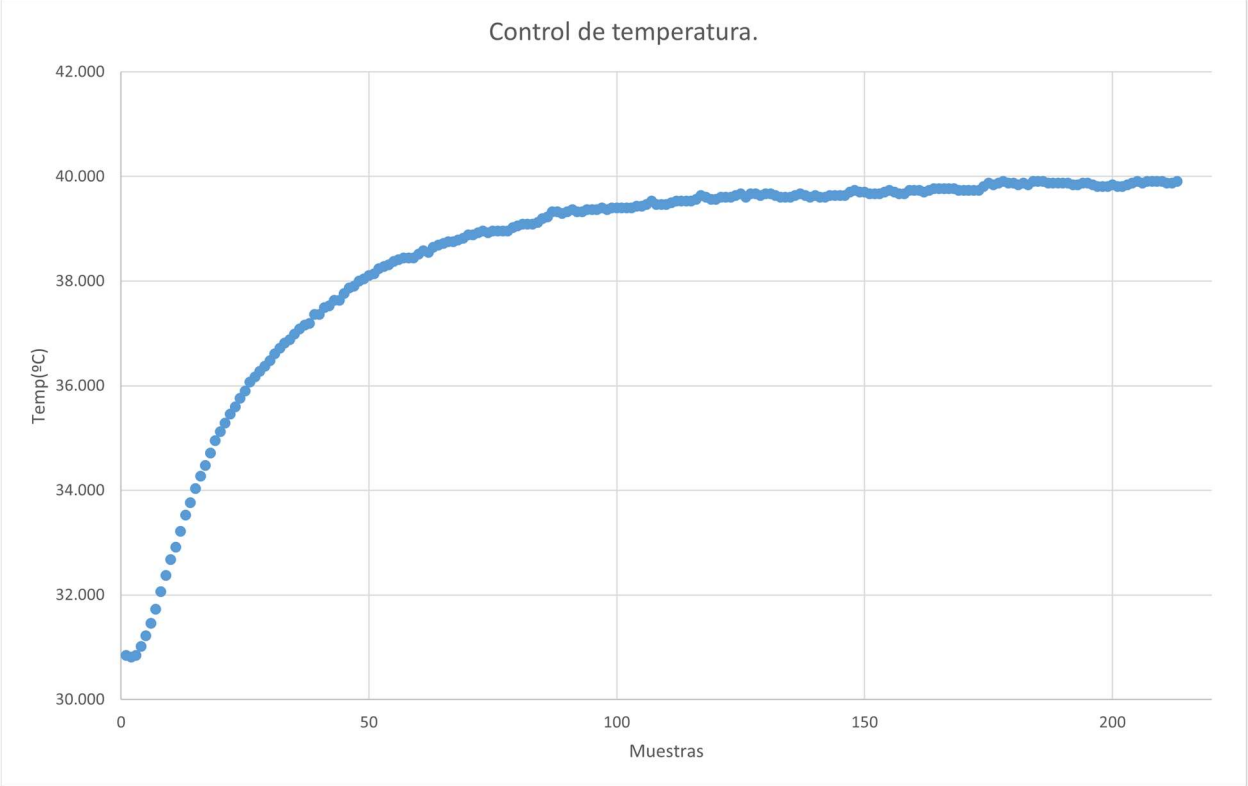

Figura 6-2. Control de temperatura

En este caso, se tomó una referencia de 40ºC, llegando a los límites del sistema. Hay que destacar, que se tomaron muestras cada 10 segundos. Este tiempo se considera aceptable, pues solo nos enfrentamos a la dinámica cuando ponemos en marcha el sistema. Además, teniendo en cuenta que a partir de los 35º C el bolómetro comete errores inferiores a 1º C, se considera aceptable empezar a tomar fotos a los 38º C, temperatura que tardamos unos minutos en alcanzar y, mantener dicho valor, no supone un gran consumo.

Por tanto, una posible modificación para este control sería usar una señál de control mas conservadora, de manera que se fije una referencia en torno a 38º C y tardemos en alcanzar dicho valor unos 15 minutos, tiempo considerado aceptable comparado con el tiempo que tardamos en alcanzar 40º C.

En conclusión, el controlador que se ha realizado, aunque no haya sido implementado mediante el uso de un PID, permite alcanzar la referencia que marquemos en régimen permanente y es capaz de mantenerla correctamente. Como ya hemos mentado, la dinámica final del sistema nos permite el uso de acciones de control suaves, evitando un consumo elevado.

6.2.2 Análisis de la detección de ojos

Respecto al algoritmo de detección ocular implementado, basado en cascadas de haar, se ha probado tanto en personas como animales. El resultado obtenido, tiene un mayor porcetanje de acierto en personas que en animales. Esto se debe a que, debido al fondo de la imagen, el algoritmo encuentra ojos en zonas donde hay sombras. A estos ojos los nombramos como “ojos falsos”.

En las personas, podemos evitar la detección de ojos falsos, es decir, la detección de ojos en lugares donde no hay ojos. La solución es sencilla, primero detectamos la cara, añadiendo la cascada de haar correspondiente y, dentro de esa zona de la imagen, sabemos que tienen que estar los ojos. Ajustando el algoritmo de detección, buscamos un ojo en esa zona.

Por contra, en animales, no disponemos de cascadas de haar que identifiquen donde se encuentra un animal concreto o donde está su cara. Durante el desarrollo del proyecto, se barajó crear una cascada con ese objetivo. Sim embargo, debido a la complejidad de crear una cascada de haar o una inteligencia artificial con estos fines, se considerará que, cuando tenemos un animal, sabemos sus coordenadas, evitando así añadir complejidad al proyecto.

En la figura 6-3, podemos ver como se detecta correctamente el ojo en una persona gracias al algoritmo mentado.

Figura 6-3. Detección de ojos en personas

Cabe destacar que, en función de la iluminación que presente el rostro de la persona, encontraremos el ojo o no. Esto es, si la cara está muy oscura, la cascada de haar puede detectar ojos donde no los hay o, incluso, no detectar ningún ojo. Adicionalemente, según el ángulo de inclinación de la persona respecto a la cámara, puede llegar a detectar los orificios nasales como ojos si la configuración del algoritmo no se ha hecho correctamente.

6.2.3 Análisis del sistema completo

Una vez hemos analizado por separado los sistemas, tenemos que analizar el resultado global. Como hemos mencionado anteriormente, se ha probado en humanos, obteniendo la figura 6-4.

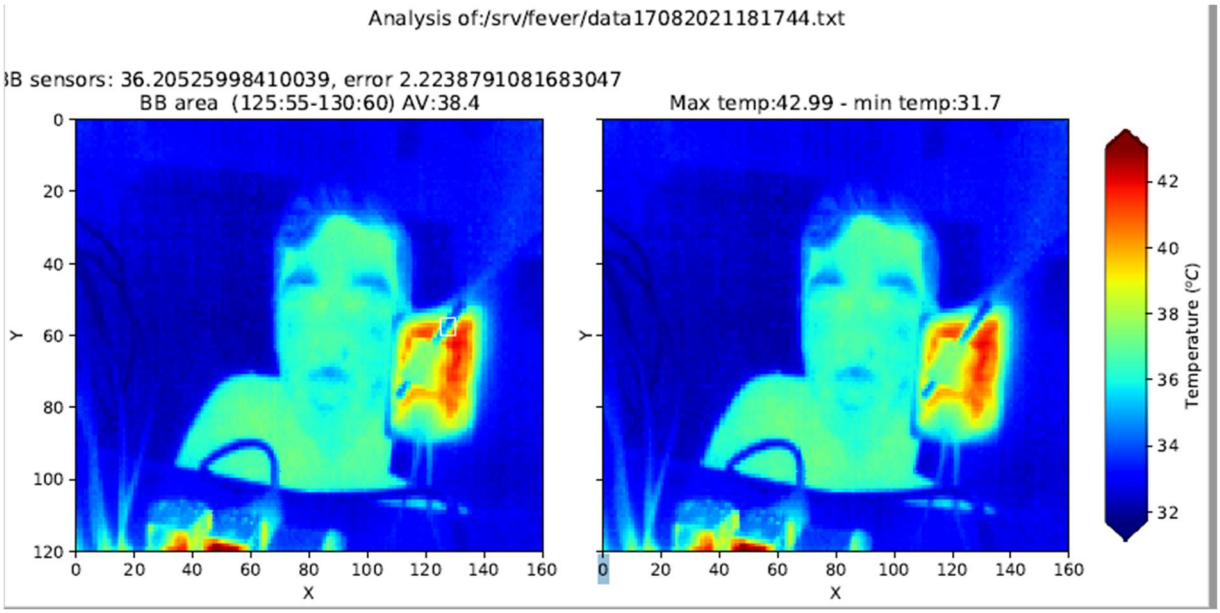

Figura 6-4. Prueba final

En este caso, se fijó una referencia de temperatura de 36 ºC, con objeto de realizar los ensayos con mayor rapidez. Se observa que, en la zona donde está la PT100 y la manta térmica, la temperatura está cercana a la referencia. En concreto, el sensor PT100 marcaba 36,02º C, lo cual se considera correcto. Debido a que, inevitablemente, la media de temperatura de esa zona tomada por el bolómetro, incluye parte de la temperatura de la manta térmica (mayor que la de referencia puesto que el sensor de temperatura no esta en completo contacto), tenemos una media en esa zona de 38ºC. Sin embargo, fijandonos en los colores de referencia, podemos determinar que, la temperatura en la PT100 está cercana a la referencia que buscamos.

Gracias al algoritmo de detección de ojos, obtenemos las coordenadas (100, 50), donde tenemos una media de temperatura cercana a los 37º C, lo cual se considera una temperatura aceptable para una persona. En el momento del ensayo, la persona que participó no presentaba síntomas febriles, por lo que se toma como correcta esta medida.

Adicionalmente, para probar la parte del sistema relativa al análisis del estado de cultivos, se tomaron las medidas de temperatura y humedad en el mismo recinto. Se obtuvieron los siguientes resultados:

- Temperatura del aire: 31,12º C
- Temperatura del suelo: 30,43º C
- Humedad relativa: 41%

Dado que el ensayo se realizón Sevilla durante verano, en un recinto cerrado, se toman estos datos como cercanos a los correctos.

6.3 Consideraciones

Tras analizar los resultados, debemos tener en cuenta una serie de limitaciones, que impiden la obtención de un resultado 100% exitoso.

En primer lugar, encontramos las limitaciones de los sensores que se han usado durante el proyecto. Como se han buscado sensores de bajo coste, la precisión y fiabilidad de los mismos, nos induce a errores a la hora de tomar datos. Por ejemplo, una medida simultánea de temperatura con dos PT100 diferentes, pueden diferir entre sí. Esta diferencia no es notable como para causar resultados finales erróneos, puesto que difieren a partir del segundo decimal. Sin emabrgo, el fabricante nos indica que estos sensores pueden tener hasta 1º C, en el peor de los casos, de desviación respecto a la temperatura real, por tanto, al usar como referencia la medida de una de ellas, podemos estar corrigiendo la imagen térmica con una medida errónea. No es algo que afecte de forma grave al proyecto, pero se debe tener en cuenta si tenemos dudas sobre los resultados finales.

Por otra parte, en la detección ocular, encontramos que, en función de la iluminación, podemos encontrar ojos falsos. Para una iluminación constante, se puede ajustar el algoritmo para que el numero de aciertos aumente, sin embargo, esto no es accesible para el usuario final. Además, una iluminación constante sabemos que no es posible obtener y menos en entornos rurales que es donde prentedemos instalar el sistema. Por otra parte, existen otras opciones para la detección de objetos que tardan menos tiempo en procesar la información y son mas exactos. Sin embargo, requieren de conocimientos mas avanzados sobre visión computacional, inteligencias artificiales y, principalmente, conocimientos matemáticos. Debido al limitado tiempo de desarrollo, implementar algo similar a lo mentado se sale del alcance del proyecto.

6.4 Aplicaciones futuras

A pesar de finalizar el proyecto y considerar que se han obtenido buenos resultados, el sistema diseñado es susceptible de mejora.

La aplicación actual de este sistema es en humanos, donde hemos visto que los resultados son fiables a pesar del bajo coste del sistema final. Sin embargo, como vimos anteriormente, el sistema final no es robusto y se puede ver afectado por las condiciones metereológicas, por lo que se recomienda su uso solo en interiores y en un sitio fijo. Mediante el uso de piezas obtenidas, por ejemplo, con impresoras 3D, se podría obtener un montaje robusto, que permita situar el dispositivo en exteriores, aumentando considerablemente las posibles aplicaciones del producto.

Por otra parte, para que el proyecto sea aplicable en animales, se requiere el diseño de un algoritmo, basado en inteligencias artificales, capaz de identificar la posición del animal y, posteriormente, buscar el ojo para hacer la media de temperatura de esta zona. Esto debe ser un proyecto a parte, donde se baraje cuales son las mejores formas de detectar un objeto mediante el diseño de varios algoritmos basados en diferentes técnicas de detección como pueden ser: diseño de una cascada de haar o diseño de una inteligencia artificial. Para este caso, tambien se podría mejorar el montaje donde, además de añadir robustez, permita aislar el sistema de condiciones desfavorables como la lluvia, viento y nieve, permitiendo así que funcione completamente desatendido.

Las imágenes térmicas que se obtienen gracias al bolómetro se han corregido para que presenten un error, a priori, cercano a 1º C. Sin embargo, se ha aplicado una corrección lineal de la imagen, es decir, en función del error cometido en el punto de referencia, corregimos la temperatura de todos los pixeles. Sería de conveniencia, probar el sistema con dos puntos de referencia, para ver si realmente, el error que comete el bolómetro es lineal o no. Si fuese lineal, el resultado de este proyecto es válido. Si no lo fuera, habría que buscar otra forma de corregir dicho error.

Paralelamente, se necesita un proyecto adicional para estudiar en profundidad la problemática del estrés hídrico. Con los datos recogidos por el sistema diseñado, sabemos las condiciones meteorológicas que rodean al cultivo. Sin embargo, no somos capaces de determinar el estado de la planta pues no es un resultado que se obtenga de forma trivial a raíz de los datos tomados, si no, que existen varias formas posibles de determinarlo y cada una tiene sus ventajas y desventajas, como ya explicamos en el apartado 2.

Sería interesante implementar un sistema de generación y envío de alarmas, de manera que, cuando se detecte que un sujeto presenta síntomas febriles o que una planta no está en buen estado, envíe un mensaje al agricultor o ganadero, informando de la necesidad de intervenir.
Para el caso concreto de animales, se podría también añadir un sensor de proximidad, dejando al sistema en reposo hasta que se active dicho sensor. Entonces, se procede a tomar la temperatura. Para ello, se podría dejar funcionando solo el sistema de control de temperatura y, así, cuando se active el sensor, se toma la foto y se almacenan los datos.
Para los cultivos, simplemente podemos mantener el reposo hasta que, cada varias horas, queramos ver el estado de dicho cultivo.
Esto podría suponer una gran reducción del consumo del dispositivo, así como una mayor tranquilidad para el agricultor o ganadero, puesto que no tendrá necesidad de asistir a la explotación para verificar las fotos que se hayan tomado a no ser que se le haya informado de algún problema mediante las alertas comentadas.

REFERENCIAS

[1] C. Lopez Botez, et al. Sistemas de producción procina y calidad de la carne. El cerdo ibérico, 2000

[2] Lepton Engineering Datasheet, Document Number: 500-0659-00-09 Rev: 203.

[3] RAY D. JACKSON, PAUL J. PINTER, JR., ROBERT J. REGINATO, AND SHERWOOD B. Detection and Evaluation of Plant Stresses for Crop Management Decisions

[4] Samuel O. Ihuoma, Chandra A. Madramootoo. Recent advances in crop water stress detection, 2017

[5] Instituto Cartográfico y Geológico de Cataluña. NDVI v1.0 Especificacions tècniques 19.04.2021

[6] Mike Tipton, Experimental Physiology, 2017

[7] Freund, Yoav; Schapire, Robert E (1997). "A decision-theoretic generalization of on-line learning and an application to boosting". *Journal of Computer and System Sciences*. **55**: 119–139.

[8] Adafruit, PT100 Technical Specifications

[9] MAX31865 Datasheet 19-6478; Rev 3; 7/15

[10] WINGONEER IRF520 MOSFET Driver Module Datasheet.

[11] DHT11 Technical DataSheet Translated Version, 1143054

GLOSARIO

IoT	Internet of Things
NDVI	Índice de Vegetación de Diferencia Normalizada
GPIO	General Input Output Port
GMD	Grasa media diaria
SPI	Serial Peripheral Interface
UVC	USB Video Class
USB	Universal Serial Buss
CS	Chip Select
PDF	Portable Document Format
CSI	Camera Serial Interface
LAN	Local Area Network
PWM	Pulse Width Modulation
MOSFET	Metal-Oxide-Semiconductor Field-Effect-Transistor

ANEXO: CÓDIGO

```
from uvctypes import *
import schedule #para temporizado de fotos
import time
import cv2
import sys
import numpy as np
import picamera
import Adafruit_DHT #sensores utilizados para hacer el cuerpo negro
import logging
import RPi.GPIO as GPIO
import requests
import json
import base64
import uuid
# Para el tratamiento de imagen
import numpy as np
import matplotlib
matplotlib.use('agg') #para impedir que de error al tratar de abrir "$DISPLAY"
que no existe en Raspberry
import matplotlib.pyplot as plt
import matplotlib.cbook as cbook
import matplotlib.colors as colors
import matplotlib.patches as patches
try:
  from queue import Queue
except ImportError:
  from Queue import Queue
import platform
import datetime
from w1thermsensor import W1ThermSensor #Sensor ambiental en la caja de control
import board
import busio
import digitalio
import adafruit_max31865
import adafruit_dht
```

```
BUF_SIZE = 2
DHT_SENSOR = Adafruit_DHT.AM2302
DHT_PIN = 15 #pin físico número 10, GPIO15
URL = "http://gebrais.ddns.net/api/json_agromotica"
URL_LOGIN = "http://gebrais.ddns.net/api/login"
URL_UPLOADS = "http://gebrais.ddns.net/api/uploads"
LOG_LEVEL = logging.INFO
LOG_FILE = "/var/log/radiopi"
LOG_FORMAT = "%(asctime)s %(levelname)s %(message)s"

#Comunicacion con las PT100
spi = busio.SPI(board.SCK, MOSI = board.MOSI, MISO = board.MISO)
cst = digitalio.DigitalInOut(board.D5)
csa = digitalio.DigitalInOut(board.D6)
csg = digitalio.DigitalInOut(board.D13)
pt100_thermal = adafruit_max31865.MAX31865(spi, cst, wires =3, rtd_nominal =100,
ref_resistor=430)
pt100_air = adafruit_max31865.MAX31865(spi, csa, wires =3, rtd_nominal =100,
ref_resistor=430)
pt100_ground = adafruit_max31865.MAX31865(spi, csg, wires =3, rtd_nominal =100,
ref_resistor=430)

dhtDevice = adafruit_dht.DHT11(board.D19)

#PWM for thermal sheet
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(12,GPIO.OUT)

signal = GPIO.PWM(12,50)
duty = 90
signal.start(duty)
#Sistema Control de temperatura
def thermal():
  print('Temperature 1: {0:0.3f}C'.format(pt100_thermal.temperature))
  temp = open("temp.txt", "a")
  dc = open("duty.txt", "a")
  temp.write("{0:0.3f}\n ".format(pt100_thermal.temperature))
#  dc.write("DC = %d \n" %(duty))
  temp.close()
  dc.close()
```

```
  if pt100_thermal.temperature > 39:
      duty = 20
  elif (pt100_thermal.temperature <= 39) & (pt100_thermal.temperature >=38):
      duty = 60
  else:
      duty = 100
  signal.ChangeDutyCycle(duty)
  time.sleep(10)


#Rutina principal de captura de la información
def get_all_data():
  """ función que captura los datos. Se llama por temporizador o al pulsar el botón"""
  logging.info("##Starting routine 'get_all_data'##")
  now = datetime.datetime.now()
  suffix = now.strftime("%d%m%Y%H%M%S")  #se usará para guardar los ficheros de datos
  img_name_rgb = "/srv/fever/rgb%s.jpeg" % suffix
  img_name = "/srv/fever/img%s.jpeg" % suffix
  data_name = "/srv/fever/data%s.txt" % suffix
  dht_file_name = "/srv/fever/dht%s.txt" % suffix

  #Environmental parameters
  try:
    humidity = dhtDevice.humidity
    temp_dht11 = dhtDevice.temperature
    print("Humidity: {}% Temp DHT11: {:.1f} C".format(humidity, temp_dht11))
    print('Air Temperature: {0:0.3f}C'.format(pt100_air.temperature))
    print('Ground Temperature: {0:0.3f}C'.format(pt100_ground.temperature))
    data = open("data.txt", "a")
    data.write("Air temp: {0:0.3f} C, Ground temp: {0:0.3f} C, Humidity: {}% \n".format(pt100_air.temperature, pt100_ground.temperature, humidity) )

  except RuntimeError as error:
    print(error.args[0])
    time.sleep(2.0)
    #continue
  except Exception as error:
    dhtDevice.exit()
#    raise error
```

```
  print("*Stream")

  #Thermal camera Stream control
  libuvc.uvc_get_stream_ctrl_format_size(
        devh, byref(ctrl), UVC_FRAME_FORMAT_Y16,
        frame_formats[0].wWidth, frame_formats[0].wHeight,
        int(1e7 / frame_formats[0].dwDefaultFrameInterval)
        )
  #Save stream in result
  res = libuvc.uvc_start_streaming(devh, byref(ctrl),
                                   PTR_PY_FRAME_CALLBACK, None, 0)
  if res < 0:
    logging.error("uvc_start_streaming failed: {0}".format(res))
    exit(1)
  else:
    logging.info("UVC started")

  # Take picture

  camera =picamera.PiCamera()
# camera.rotation = 180
  camera.resolution = (160, 120)

  # Camera warm-up time
  time.sleep(2)
  camera.capture(img_name)
  img = cv2.imread(img_name)
  #Acceder al fichero xml
  haar_cascade = cv2.CascadeClassifier('eye.xml')
  #buscar ojos en la imagen
  eyes_found=    haar_cascade.detectMultiScale(img,    scaleFactor    =    1.1,
minNeighbors=5)
  print(f'Numero de ojos encontrados = {len(eyes_found)}')

  #Dibujamos un rectangulo en los ojos y guardamos la imagen
  for (x,y,w,h) in eyes_found:
    x_i=x
    y_i=y
    x_f=x+w
    y_f=y+h
```

```
    cv2.rectangle(img, (x_i,y_i), (x_f, y_f), (0,255,0), thickness=2)
  cv2.imwrite("ojos.jpg", img)


  #queue data to variable
  data = q.get(True, 500)  #La cola q está creada como variable global.¿Mejorar?
  if data is None:
    logging.error("Error al leer la cámara térmica")
    exit(1)
  else:
    logging.info("datos leídos")

  minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(data)
  while minVal == 0 and maxVal == 0:
    logging.warning("no values from thermal camera, new attempt:stop and start
UVC")
    libuvc.uvc_stop_streaming(devh)
    time.sleep(3) # wait 3 seconds to retry
    res = libuvc.uvc_start_streaming(devh,
                                     byref(ctrl),
                                     PTR_PY_FRAME_CALLBACK,
                                     None, 0)
    if res < 0:
      logging.error("uvc_start_streaming failed: {0}".format(res))
      exit(1)
    data = q.get(True, 500)
    if data is None:
      logging.error("Error reading thermal camera data (no data)")
      exit(1)
    minVal, maxVal, minLoc, maxLoc = cv2.minMaxLoc(data)
    logging.warning("new attempt done")
  logging.info("Value loaded")

  #Salvamos el fichero con las temperaturas en bruto. Ojo! no corregidas!!
  np.savetxt(data_name, data, delimiter=',', fmt='%d')
  logging.info("Temperature saved without corrections")

  #Paramos el streaming, ya no lo vamos a necesitar por ahora más
  libuvc.uvc_stop_streaming(devh)
  logging.info("streaming stopped")
  # Calculamos el error con el blackbody y los valores medios
```

```
fever = np.loadtxt(data_name, delimiter=',',ndmin=2)
fever = fever/100 -273.15 # Pasamos a °C

print("\nMedia foto:",np.mean(fever),"\nMáx foto:",np.max(fever))
#x1,x2,y1,y2=28,62,70,100 #Coordenadas del agua caliente
#Coordenadas de los ojos
heat=np.zeros((x_f-x_i,y_f-y_i))
heat=fever[y_i:y_f,x_i:x_f] #Matriz del ojo

x3,x4,y3,y4=100,135,75,100 #Coordenadas de la pt100
bbox=np.zeros((x4-x3,y4-y3))
bbox=fever[y3:y4,x3:x4]
error = np.mean(bbox)- pt100_thermal.temperature
#dt_bbox = np.std(bbox)
print("\n############",np.mean(bbox),"##################\n")
fever_OK = fever - error

# Rutina para generar la imagen tratada con escala de temperaturas
# Se salva en un pdf
# Preparamos el canvas para los dos gráficos
fig, (ax,ax1) = plt.subplots(1,2, sharey=True,
                             constrained_layout=True,
                             figsize=(10,5))
ax.invert_yaxis()  # como están igualados afecta a ambos ejes verticales
logging.info("Plot ready")

fig.suptitle("Analysis of:" + data_name
             ,fontsize=12)
pcm = ax.pcolormesh(fever, # Imprime la matriz con los datos en bruto
                    rasterized=True,
                    vmin=np.min(fever),
                    vmax=np.max(fever),
                    cmap='jet')
pcm2 = ax1.pcolormesh(fever_OK,  #  matriz fiebre menos error.
                    rasterized=True,
                    vmin=np.min(fever_OK),
                    vmax=np.max(fever_OK),

                    cmap='jet')
fig.colorbar(pcm2, shrink=0.95, extend='both',
             label='Temperature ($^oC$)')
```

```
  logging.info("Matrices y barra colores")
  #Ejes de la primera imágen
  ax.set_xlabel('X')
  ax.set_ylabel('Y')
  ax.set_title(
               "\nBB sensors: " + str(pt100_thermal.temperature) +
               ", error " + str(error) +
               "\nBB area  (" + str(x3)+":" +str(y3)+"-"+str(x4)+
               ":"+str(y4) +
               ") AV:" + str(round(np.mean(bbox),2))

)
  #Ejes de la segunda imagen, con temperatura corregida
  ax1.set_xlabel('X')
  ax1.set_ylabel('Y')
  ax1.set_title(
               "Max temp:"+str(round(np.max(fever_OK),2))+
               " - min temp:"+str(round(np.min(fever_OK),2))
               )
  rect = patches.Rectangle((x_i,y_i),x_f-x_i,y_f-y_i,
                           linewidth=1,
                           edgecolor='w',
                           facecolor='none')
  ax.add_patch(rect)
  rect2 = patches.Rectangle((x3,y3),x4-x3,y4-y3,

linewidth=1,

edgecolor='w',

facecolor='none')
  ax.add_patch(rect2)
  logging.info("Ejes ready")
  #plt.show()  # En la raspberry no tenemos donde presentar la imagen
  plt.savefig(data_name+".pdf") #Salvamos los datos en formato pdf
  logging.info("Pdf generated")

  # Valores de la cámara al log
  logging.info("Thermal image Min:%f en %d,%d" %(ktoc(minVal),
```

```
                                                    minLoc[0], minLoc[1]))
  logging.info("Thermal image Max:%f en %d,%d" %(ktoc(maxVal),
                                                    maxLoc[0], maxLoc[1]))
  print("Thermal image Min:%f en %d,%d" %(ktoc(minVal), minLoc[0], minLoc[1]))
  print("Thermal image Max:%f en %d,%d" %(ktoc(maxVal), maxLoc[0], maxLoc[1]))

#Activamos la cámara por primera vez
q = Queue(BUF_SIZE) # Queue to store the data -Global!-
ctx  = POINTER(uvc_context)()
dev  = POINTER(uvc_device)()
devh = POINTER(uvc_device_handle)()
ctrl = uvc_stream_ctrl()
res = libuvc.uvc_init(byref(ctx), 0)
if res < 0:
  logging.error("uvc_init error")
  exit(1)
res = libuvc.uvc_find_device(ctx, byref(dev), PT_USB_VID, PT_USB_PID, 0)
if res < 0:
  logging.error("uvc_find_device error")
  exit(1)
res = libuvc.uvc_open(dev, byref(devh))
if res < 0:
  logging.error("uvc_open error")
  exit(1)
logging.info("device opened!")
print_device_info(devh)
print_device_formats(devh)
frame_formats = uvc_get_frame_formats_by_guid(devh, VS_FMT_GUID_Y16)
if len(frame_formats) == 0:
  logging.error("device does not support Y16")
  exit(1)
#Si hemos llegado hasta aquí, todo está listo para empezar

#Wait 10 seconds to take first photo

time.sleep(10)
#Calentamos la manta al iniciar el sistema
while pt100_thermal.temperature <= 36:
  print("Calentando manta")
  thermal()
```

```
get_all_data()
#Automatic schedule to take a picture every 20 seconds
schedule.every(10).seconds.do(thermal)
schedule.every(20).seconds.do(get_all_data)

while True:
    schedule.run_pending()
    time.sleep(1)

signal.stop()
GPIO.cleanup()
```