Universidad de Valladolid

E.T.S Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

SISTEMA DE SEGURIDAD Y MONITORIZACIÓN BASADO EN RASPBERRY PI

Autor:
D. Javier García Prieto
Tutor:
D. Benjamín Sahelices Fernández

Agradecimientos

A mis padres y hermano, por hacer que siempre salga el sol sin importar las nubes que aparezcan a lo largo del camino.

A Marta, por ser "Mi Constante".

A mi tutor D. Benjamín Sahelices Fernández, por haberme dado la oportunidad de trabajar en algo con lo que he disfrutado, sin lugar a dudas un lujo.

Resumen

Este Trabajo de Fin de Grado consiste en emplear una Raspberry Pi, un mini ordenador de bajo coste, para dotar de cierta inteligencia a un habitáculo que precise de monitorización y algunas medidas de seguridad. De forma que el responsable del lugar mantenga una comunicación con el sistema proporcionando un mayor grado de confort a la hora de realizar las inspecciones pertinentes. Seguridad, confort y comunicación son características propias de la domótica/inmótica.

La placa Raspberry Pi es el eje de este TFG, con ella se establece la comunicación con los sensores que se encuentran dentro de la sala que precise de los servicios de este proyecto, pero también mediante la Raspberry y el servidor que en ella se instale, se logrará comunicación con el usuario final mediante una interfaz web. En cualquier lugar con acceso a Internet, el responsable de la seguridad de la sala podrá administrarla a través de una página web.

El escenario sobre el cual se trabajará a lo largo del proyecto es la sala donde se encuentran los servidores de una empresa y donde es conveniente desarrollar sistemas de control de temperatura, detección de humos, alertas en tiempo real mediante una plataforma de mensajería instantánea, sistema de videovigilancia… Sin embargo, el contexto del proyecto puede ser ambiguo en el sentido de que las características de seguridad que se persiguen en el escenario de la sala de servidores pueden ser muy similares a las deseables en otros ámbitos, incluso el doméstico: monitorizando un determinado espacio de la vivienda.

La idiosincrasia del TFG hace que se trabaje con una cantidad de datos elevada (por ejemplo, las continuas mediciones de temperatura), es por ello que permite una ligera aproximación al concepto de Big Data, donde se abordarán procedimientos para simplificar, esclarecer y obtener resultados estadísticos a partir de enormes colecciones de datos. De esta manera se podrá entregar al usuario final informes periódicos, legibles y significativos.

Conceptos clave: Raspberry Pi, inmótica, GPIO, tratamiento de datos, monitorización.

Abstract

This Final Project Grade consists of employing a Raspberry Pi, a low-cost mini computer, to provide intelligence to a cabin that needs monitoring and some security measures. So that the person in charge of the place maintains a communication with the system providing a greater degree of comfort. Security, comfort and communication are characteristic of home automation.

The Raspberry Pi is the axis of this project, with it establishes the communication with the sensors that are in the room that needs the services of this project, but also through the Raspberry and the server that it is installed, will be communication with the end user through a web interface. In any place with Internet access, the person in charge of the security of the room can administer it through a web page.

We work assuming the project is used in the room where are the servers of a company and where it is convenient to develop systems of temperature control, smoke detection, real-time alerts through an instant messaging platform… However, the project context may be ambiguous in that the security features that are pursued in the server room may be very similar in other domains, including the domestic one: monitoring a place of the house.

The project works with a high amount of data (for example, continuous temperature measurements), that's why it allows a small approximation to the concept of Big Data, where we will use procedures to simplify, clarify and obtain statistics results from huge collections of data. In this way, periodic, legible and meaningful reports can be delivered to the end user.

Key concepts: Raspberry Pi, home automation, GPIO, data processing, monitoring.

Tabla de contenidos

Lista de figuras

Lista de tablas

Bloque I: Introducción

Capítulo 1: Introducción

La Raspberry Pi es un pequeño ordenador de bajo coste pero completamente funcional y de gran versatilidad. Este dispositivo, del tamaño de una tarjeta de crédito, soporta el desarrollo de gran cantidad de proyectos informáticos, desde la optimización de un centro multimedia hasta el control automático de la puerta de un garaje. Gracias a la incorporación de una serie de pines de propósito general que posee la placa se permite la comunicación con infinidad de sensores, dando lugar múltiples aplicaciones.

Este Trabajo de Fin de Grado pretende hacer uso de una placa Raspberry Pi y su generosa conectividad para simplificar las labores de monitorización y seguridad que precisa un habitáculo que requiere tales cuidados, como puede ser una sala de servidores de una empresa. Los servidores para su correcto funcionamiento precisan trabajar dentro de un rango de temperaturas y sería deseable que el lugar donde se encuentran estuviese dotado de algunas medidas de seguridad como detectores de humo, cámaras de vigilancia y un sistema de alertas en tiempo real que avise de los fallos al responsable de la sala. Todas estas medidas de seguridad se pretenden implementar en este proyecto donde además se pone a disposición del usuario una interfaz web para la gestión de esta seguridad. Se instalará en la Raspberry un servidor web que permitirá el acceso a la interfaz desde cualquier lugar con acceso a Internet.

Por lo que se ha explicado anteriormente podemos ver a la Raspberry como el eje de este TFG. Trabajará como recolector recogiendo información de los distintos sensores, esta información será procesada por la propia placa y en función de los datos obtenidos, la Raspberry tendrá también el papel de actuador, encargándose de ejecutar las respuestas adecuadas.

Como es lógico, para conocer en tiempo real el estado de una magnitud, deberemos realizar muchas mediciones periódicas, mediciones que en muchas ocasiones pueden ser aparentemente irrelevantes. Por ejemplo, conocer el valor de la temperatura a una hora concreta de forma aislada puede ser un dato poco relevante, sin embargo, la medición periódica de la temperatura puede alumbrar resultados más significativos como una media diaria, un valor máximo mensual, anual, etc. Dicho de otra forma, en este TFG se implementarán mecanismos de selección y simplificación de grandes volúmenes de datos además de aumentar la seguridad de una habitación obteniendo información de la misma en tiempo real.

Hoy en día, con la inclusión de la tecnología en tareas cotidianas que hasta hace poco prescindían de la misma, es habitual toparse con términos como domótica e inmótica. A continuación veremos cómo los define la Asociación Española de Domótica en Inmótica.

Mientras que la domótica es definida como: "*el conjunto de tecnologías aplicadas al control y la automatización inteligente de la vivienda, que permite una gestión eficiente del uso de la energía, que aporta seguridad y confort, además de comunicación entre el usuario y el sistema*".

La inmótica se define como: "*el conjunto de tecnologías aplicadas al control y la automatización inteligente de edificios no destinados a vivienda, como hoteles, centros comerciales, escuelas, universidades, hospitales y todos los edificios terciarios, permitiendo una gestión eficiente del uso de la energía, además de aportar seguridad, confort, y comunicación entre el usuario y el sistema*".

Por tanto, la diferencia radica en el ámbito donde se aplican las tecnologías y este proyecto sería perteneciente a la rama de la inmótica al situarse, según la idea inicial, en un entorno empresarial. No obstante, se trata de un proyecto que ofrece una seguridad, confort y comunicación con el usuario también válidos para aplicarse en una vivienda. Es por ello que en ciertos momentos de la memoria se pueden emplear las palabras domótica e inmótica indistintamente.

El sistema de seguridad y monitorización creado y del que versa esta memoria, se ha bautizado con el nombre de RaSpy (cuyo logo podemos ver en la Figura 1.1).

Figura 1.1: logo del sistema creado, RaSpy

La elección de este nombre ha surgido de la forma en la que de modo un abreviado nos referimos a la Raspberry Pi, “Rasp”, pero añadiendo una “y” al final del nombre para crear la palabra inglesa “Spy”, entendiendo la monitorización como una metáfora del espionaje que podemos realizar sobre el lugar que deseamos controlar. Los colores del logo están acordes con los del logo de Raspberry Pi (una frambuesa) y la forma sugiere un ojo, relacionado con la parte del espionaje, o mejor dicho, monitorización.

1.1. Motivación

La idea de que un Trabajo de Fin de Grado y el desarrollo del mismo permitan la posibilidad de implementar un sistema domótico/inmótico completamente funcional orientado a la monitorización y seguridad de cualquier instalación es un tema que considero de interés, especialmente cuando todo el sistema gira en torno a un mini ordenador y una serie de sensores cuyo coste total es inferior a cincuenta euros.

La utilización de una placa Raspberry Pi ya es una motivación en sí misma debido a la gran versatilidad que ofrece el dispositivo. Concretamente, resulta muy interesante experimentar cómo la informática trabaja conjuntamente con una disciplina complementaria como es la electrónica para integrar la tecnología en un espacio que, al incluirla, mejora en algún aspecto.

Son infinitos los ámbitos donde la tecnología puede hacer su aparición para ofrecer soluciones que mejoren una determinada situación actual. La elección de que esa tecnología se oriente hacia la seguridad y la monitorización surge motivada por una experiencia laboral personal en la que uno de los cometidos del puesto de trabajo consiste en controlar cada dos horas la temperatura de la habitación de servidores de una empresa, en detrimento de las ventas que se pierden en tal comprobación. En términos informáticos, dicha actividad puede ser considerada altamente ineficiente ya que el consumo de recursos es elevado (dedicación exclusiva del auxiliar de venta durante un 8% de su jornada) y afecta a la productividad del negocio (pérdida de parte de las ventas).

El sistema de seguridad propuesto presenta una elevada escalabilidad, propiedad deseable en cualquier sistema. Aunque el sistema inicialmente se idease para evitar tener que comprobar la temperatura de una sala físicamente, es posible añadir innumerables sensores que aumenten la funcionalidad del sistema y lo hagan más seguro. Teniendo en cuenta que no se puede garantizar la total seguridad de un sistema, considero que la inclusión de la tecnología en este tipo de habitáculos con el fin de mejorar la seguridad de los mismos sería una

práctica adecuada ya que son lugares donde se almacena información sensible de las empresas y la manifestación de un riesgo podría ocasionar graves pérdidas.

En cuanto a la implementación del proyecto, se hace uso de diferentes lenguajes de programación (todos ellos vistos a lo largo de los cursos de nuestra titulación), a fin de adecuar las características que ofrece cada lenguaje a las necesidades del momento, emplearemos Python para hacer uso de los pines GPIO de la Raspberry, PHP para programar el servidor web que se instalará en la placa, SQL para la base de datos MySQL, y HTML junto con CSS para la estructura, el contenido y el diseño de la página web que se ofrece al usuario del sistema. Recordar estos lenguajes y aunarlos para crear un sistema es una experiencia enriquecedora que acerca, aunque sea a pequeña escala, a la confección de grandes sistemas.

La toma de decisiones de diseño e implementación que a lo largo del proyecto se afrontan considero que constituyen un paso adelante en la formación académica, al tener que tratar con situaciones y elecciones que no tenían lugar en muchos enunciados de las prácticas de las asignaturas, al estar estas bien delimitadas buscando competencias concretas.

1.2. Objetivos

El propósito principal de este Trabajo Fin de Grado consiste en diseñar e implementar un sistema de seguridad y monitorización basado en una Raspberry Pi 2 modelo B y aplicarlo a una sala de servidores, de modo que el responsable de dicha sala no deba presenciarse en la misma para llevar un control de las magnitudes que en ella se miden. De este amplio objetivo general subyacen otros objetivos también importantes que se persiguen con la implementación de este proyecto:

- Conocer con cierto nivel de detalle las características del dispositivo eje del sistema, la Raspberry Pi, asimilando así su gran potencial y versatilidad en el desarrollo de proyectos.
- Configurar la Raspberry e implementar el montaje eléctrico mediante el cual la placa se comunicará con los distintos sensores y actuadores.
- Mediante la instalación de un servidor en la Raspberry Pi, se debe permitir la comunicación con el exterior a través de Internet.
- El usuario responsable de la sala podrá gestionar la seguridad de la misma mediante una página web que le ofrecerá los datos relativos a los sensores.
- La Raspberry automatizará tareas: recibirá la información captada por los sensores existentes en la sala, la procesará y realizará la toma de decisiones pertinente, accionando el correspondiente actuador en caso necesario.
- Se desarrollará un sistema de alertas que comunicará cualquier incidencia al usuario en tiempo real a través de una aplicación de mensajería.
- Debido a la gran cantidad de datos, se marca como objetivo el tratamiento de estos datos masivos con el propósito de ofrecer al usuario informes que sólo contengan datos significativos.

Por último, escribir programas de calidad y bien documentados es un objetivo siempre presente a la hora de codificar la solución de cualquier problema, no es esta una excepción y cada uno de los programas y su documentación se han codificado siempre pensando en que un tercero ajeno al proyecto pudiese comprender las líneas de código y el producto sea más fácil de mantener.

1.3. Metodología

La metodología empleada para llevar a cabo el desarrollo del software de este proyecto se ha basado en el desarrollo iterativo propio del Proceso Unificado (PU), que nos ha permitido adaptarnos a las características del proyecto.

Se ha decidido trabajar en este proyecto acogiéndonos a las directrices marcadas por la metodología basada en el modelo iterativo e incremental para el desarrollo evolutivo de la ingeniería del software. La elección de este modelo se basa en que estamos abordando un proyecto amplio del cual se conoce una serie de requisitos centrales pero a medida que se avanza en ellos se requiere una especialización de cada una de las partes que constituyen el sistema final. Además, este modelo permite que desde fases tempranas podamos desarrollar versiones funcionales del sistema, un sistema por el momento incompleto pero que cuya siguiente fase partirá del final de su antecesora, y así sucesivamente hasta haber desarrollado un sistema completo que satisfaga las necesidades del cliente.

La necesidad del desarrollo iterativo y evolutivo (propio del PU) impera en este proyecto debido a que en ocasiones se debe partir de cero al añadir una funcionalidad al sistema ya y una batería de pruebas de la funcionalidad a añadir. En las primeras iteraciones, la elección de que el subsistema a añadir puede requerir de forma aislada un estudio de la viabilidad, un análisis, un diseño, una implementación los requisitos y el diseño pueden no ser lo que se espera del sistema final, sin embargo, el hecho de poseer material funcional en las primeras fases del diseño, nos permite obtener una rápida retroalimentación extraída de los usuarios, desarrolladores y de la correspondiente batería de pruebas. Tener retroalimentación en una etapa temprana permite la posibilidad de modificar o adaptar la comprensión de los requisitos o el diseño debido a que los usuarios en fases tempranas ya tiene un producto al que probar y esto permite un grado de exactitud superior al que ofrecen las especulaciones sobre requisitos y diseños correctos.

En la Figura 1.2 podemos ver de forma gráfica el concepto del Proceso Unificado en el que nos basamos: la funcionalidad se va ampliando dividiendo la propia funcionalidad del sistema en subsistemas que poseen sus correspondientes fases de análisis, diseño, implementación y pruebas. Algunos de los beneficios de este desarrollo iterativo son:

- Correcta gestión de la complejidad al no abordar tareas excesivamente largas y complejas ni existir parones generados por un análisis exhaustivo.
- Progreso viable en las primeras etapas.
- El conocimiento adquirido en una iteración se puede mejorar para mejorar el propio proceso de desarrollo iteración a iteración.
- Los altos riesgos (técnicos, requisitos, objetivos, usabilidad, etc.) pueden ser mitigados en fases tempranas.
- Con el Proceso Unificado y sus principales características de retroalimentación, compromiso de los usuarios y adaptación, llevan a un sistema más refinado que se ajusta a las necesidades reales del cliente.

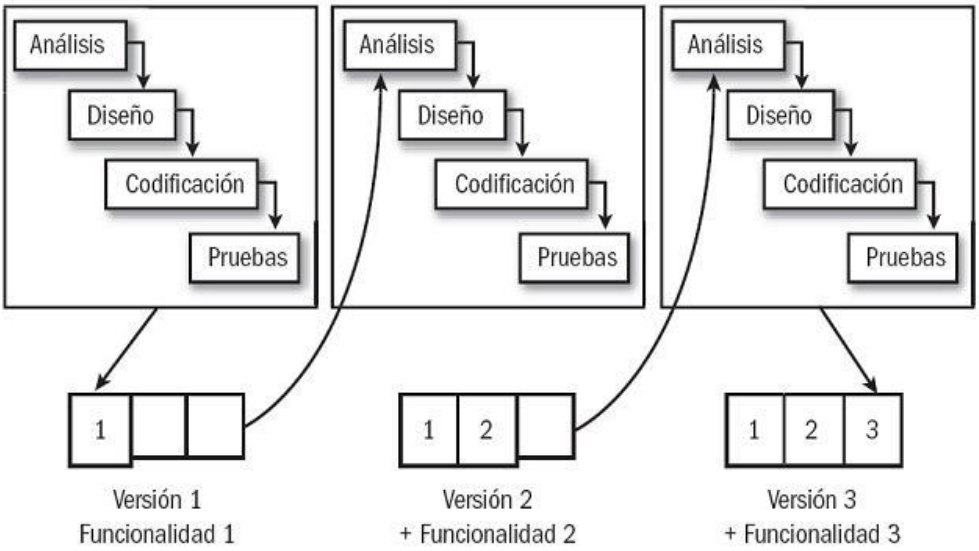


Figura 1.2: desarrollo iterativo e incremental

Este modelo implica un frecuente trato con el cliente, como ya se explicó en la motivación del proyecto, este se planteó como solución a una dificultad laboral propia, con lo cual, yo mismo junto con las entrevistas con mi tutor de proyecto hacemos las funciones de cliente. Habitualmente, es en las entrevistas con el cliente cuando se entrega una versión con una funcionalidad determinada y se inicia una siguiente fase que parte de dos artefactos: la versión (o incremento) del sistema recientemente entregado y los requisitos extraídos de la entrevista con el cliente de cara a añadir una nueva funcionalidad al sistema. La frecuencia de las entrevistas con el cliente disminuyen notablemente los fallos empleando este modelo iterativo e incremental. Las primeras iteraciones se alejan más de lo que se espera del sistema final, pero a través de la retroalimentación y la adaptación el sistema converge hacia los requisitos y el diseño más apropiados. Por tanto, llegando a las últimas iteraciones es más difícil encontrar un cambio significativo en los requisitos.

1.4. Resumen de la memoria

La presente memoria se ha dividido en seis bloques de contenido que documentarán todas las fases de desarrollo del sistema y aportarán información sobre el mismo.

En el Bloque I se pretende introducir el TFG explicando qué se persigue con su desarrollo, justificando la necesidad de ser desarrollado bajo una motivación del alumno. Se explica la metodología de desarrollo del trabajo y se fijan unos primeros objetivos como punto de partida.

Por su parte, el Bloque II abordará el contexto tecnológico del sistema. El propósito de este bloque es dar a conocer al lector la información necesaria para que conozca con qué está trabajando: se estudiará con cierto nivel de detalle la Raspberry Pi y el resto de los componentes que completan el sistema final. En este capítulo también se presta especial atención a la configuración que requieren todos estos elementos una vez que ya han sido presentados.

El Bloque III llevará a cabo el análisis del sistema. Se iniciará el bloque detallando el plan de trabajo y la metodología y recursos empleados. La planificación explicará las tareas llevadas a cabo y tendrá en cuenta los tiempos empleados, recogiendo todo esto en un diagrama de Gantt. El estudio económico también se desarrollará en este bloque, así como el estudio y la gestión de los riesgos. Por supuesto, la especificación de los requisitos del sistema y el modelado de los casos de uso también tienen sus correspondientes apartados en este bloque.

En el Bloque IV hablaremos del diseño, tanto de la Base de Datos como de la aplicación. En ambos casos aportaremos diagramas que simplifiquen los conceptos que se pretenden mostrar y se justificarán las decisiones de diseño llevadas a cabo.

El Bloque V versa sobre la implementación del sistema. Comienza con un capítulo que guía paso a paso al lector en la instalación del SO en la Raspberry. Obviamente en esta memoria no se ofrecerá el código de todos los programas que forman el sistema final, para esto se incluye un CD donde sí se recogen todos los archivos que intervienen. Sin embargo, en este bloque sí se recogerán ciertas porciones de código interesante y se justificará a nivel de implementación cómo se llevan a cabo ciertas de las funcionalidades de las que se habló en secciones anteriores. El bloque finaliza con un capítulo que recoge las principales pruebas que se han llevado a cabo en el sistema.

Finalmente en el Bloque VI se exponen las conclusiones obtenidas una vez finalizado el desarrollo del trabajo, se proponen líneas de trabajo futuras tomando como base la realización de este proyecto y se incluirán anexos que aportan información adicional, como bien puede ser un manual de usuario o un índice del contenido a entregar.

Bloque II: Contexto tecnológico

Capítulo 2: Raspberry Pi

Buena parte de la totalidad de este Trabajo Fin de Grado gira en torno a un ordenador de placa reducida (*Single Board Computer*) de bajo coste llamado Raspberry Pi, concretamente se empleará la versión 2 de este SBC.

La Fundación Raspberry Pi (cuyo logotipo se muestra a continuación en la Figura 2.1), de origen británico y fundada en 2009, perseguía el objetivo de hacer llegar las ciencias de la computación a las escuelas con la intención de animar a los niños a aprender informática como ya en 1981 lo hizo el ordenador Acorn BBC Micro.

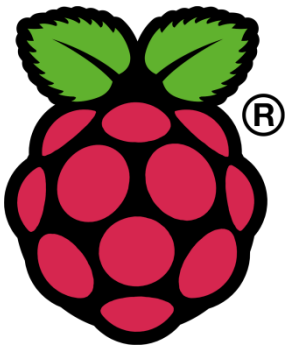


Figura 2.1: logotipo de la fundación Raspberry Pi

Tras superar las fases Alpha y Beta, lanzando para su testeo 50 y 25 placas respectivamente, es en febrero del año 2012 cuando finalmente se comienzan a comercializar estos ordenadores de bajo coste con un primer lote de 10000 placas fabricadas en China y Taiwan y no en Reino Unido debido a que la industria asiática ofrecía unos mejores plazos además de un interés económico relacionado con los impuestos.

La acogida de las primeras placas fue de lo más exitosa ya desde sus inicios, cuando los servidores de la fundación se cayeron y la carga de los servidores de las dos tiendas que las comercializaban aumentó drásticamente. La incesante demanda hizo que el 6 de septiembre del año de su lanzamiento se informase del traslado de la producción a Reino Unido, concretamente a una fábrica de Sony en Gales que aseguraba 30000 unidades mensuales y la creación de 30 puestos de trabajo.

2.1. Hardware

En esta sección se van a presentar las principales características de los seis modelos de placas Raspberry Pi existentes en la actualidad (Tabla 2.1), podremos ver su hardware junto con otros datos de interés como pueden ser el consumo, el tamaño y el precio oficial de cada dispositivo. De esta manera además de comparar las distintas placas podemos hacernos una idea de su evolución en sus primeros tres años de vida. Una vez descritas las diferentes opciones, se justificará la elección de la Raspberry Pi 2 Modelo B como eje de este proyecto y a continuación se describirá con mayor detalle sus principales componentes hardware.

	Modelo A	**Modelo A+**	**Modelo B**	**Modelo B+**	**RPI 2 Modelo B**	**RPI Zero**
SoC	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2835	Broadcom BCM2836	Broadcom BCM2835
CPU	700MHz ARM 11	700MHz ARM 11	700MHz ARM 11	700MHz ARM 11	900MHz Cortex-A7	1GHz ARM 11
GPU	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV	VideoCore IV
RAM	256Mb	256Mb	512Mb	512Mb	1Gb	512Mb
USB	1	1	2	4	4	1
Video	RCA, HDMI	RCA, HDMI	RCA, HDMI	RCA, HDMI	RCA, HDMI	RCA, miniHDMI
Audio	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	Jack, HDMI	miniHDMI
Boot	SD	MicroSD	SD	MicroSD	MicroSD	MicroSD
Red	no	no	Ethernet 10/100	Ethernet 10/100	Ethernet 10/100	no
Consumo	300mA / 1,5w / 5v	400mA / 2w / 5v	700mA / 3,5w / 5v	500mA / 2,5w / 5v	800mA / 4w / 5v	200 mA/ 0,5w / 5v
GPIO	26 GPIO	40 GPIO	26 GPIO	40 GPIO	40 GPIO	40 GPIO
Tamaño	85,6 x 53,98 mm	65 x 56 mm	85,6 x 53,98 mm	85 x 56 mm	85 x 56 mm	65 x 30 mm
Precio	25$	20$	35$	35$	35$	5$

Tabla 2.1: comparativa de los modelos existentes de Raspberry Pi

Como se puede leer en la Tabla 2.1, los "modelos +" básicamente introducen mejoras en el apartado de la conectividad (cambio de tarjeta SD a microSD, más puertos USB y más pines General Purpose Input/Output o Entrada/Salida de Propósito General), manteniendo la misma cantidad de memoria RAM (Random Access Memory) e idénticos SoC (System on a Chip), CPU (Central Processing Unit) y GPU (Graphics Processor Unit).

En lo referente a las diferencias entre los modelos A y B podemos apreciar un aumento de la memoria RAM, que pasa de los 256Mb del modelo A a los 512Mb del modelo B, la aparición de la conexión Ethernet 10/100 que permite al modelo B conectarse a Internet y la aparición de un segundo puerto USB 2.0 en el modelo B.

Como se aprecia en sus características, la segunda versión de Raspberry Pi o la Raspberry Pi 2 Modelo B supone una importante evolución en cuanto a la potencia del dispositivo, siendo esta hasta seis veces superior que el modelo B+, el referente hasta que en febrero de 2015 salió a la luz la RPI 2. Esta afirmación es el resultado de la ejecución del benchmark SysBench (mide el rendimiento en un solo hilo) en los Broadcom BCM2835 y BCM2836.

La Raspberry Pi Zero, es una excelente opción para cualquier sistema empotrado donde el tamaño de la placa deba ser lo más reducido posible. Sin embargo, el 26 de noviembre de 2015 se anunció este modelo con un primer lote de 10000 unidades destinados a la revista MagPi y otras 10000 unidades que se agotaron en cuestión de minutos. Durante los siguientes meses a su lanzamiento, su adquisición es muy complicada debido a la enorme demanda.

La similitud de precios junto con el aumento de potencia y la correcta conectividad manteniendo los 40 pines GPIO, muy importantes en este proyecto, hicieron que la placa escogida para la realización del mismo haya sido la Raspberry Pi 2 Modelo B. La memoria RAM de 1Gb, que se ha visto duplicada respecto al modelo B+, hace que la fluidez del sistema sea adecuada tanto en su modo gráfico como, por supuesto, en el modo consola. En la Figura 2.2 podemos observar la Raspberry con la que se desarrollará el proyecto indicando sus componentes:

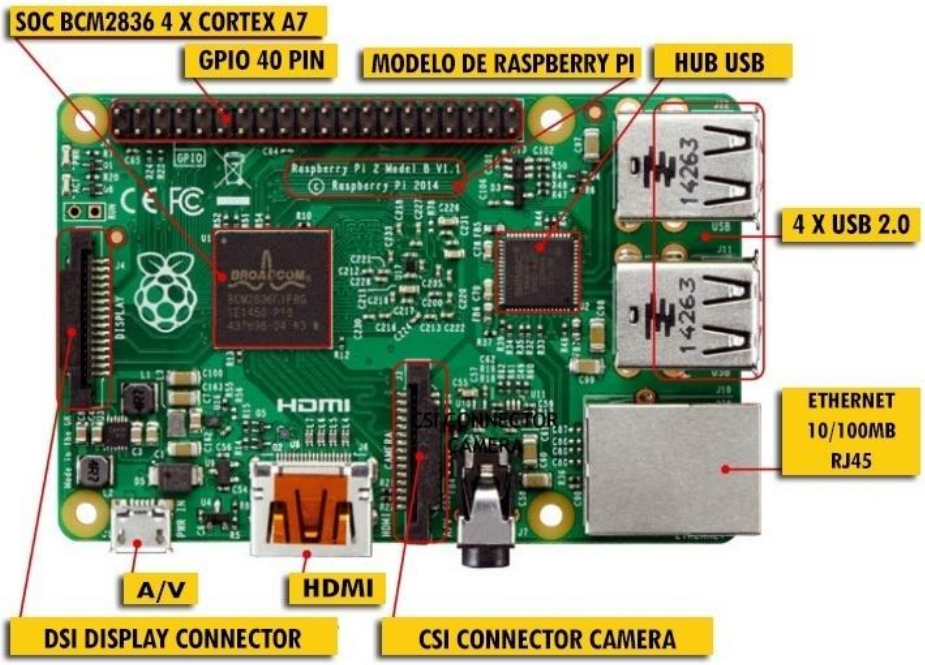


Figura 2.2: Raspberry Pi 2 Modelo B y sus componentes

2.1.1. Broadcom BCM2836 4 X Cortex A7

El diseño de esta Raspberry Pi 2 Modelo B presenta un nuevo chip, el Broadcom BCM2836 que incluye en un mismo integrado un procesador gráfico VideoCore IV, 1GB de memoria RAM LPDDR2, un procesador de cuatro núcleos (el ARMv7 Cortex A7 con un ciclo de reloj de 900MHz) y una tarjeta de sonido estéreo.

La GPU VideoCore IV es un procesador multimedia de tipo DSP (Procesador Digital de Señales, basado en un microprocesador que posee un conjunto de instrucciones, un hardware y un software optimizados para aplicaciones que requieran operaciones numéricas a muy alta velocidad) de dos dimensiones con capacidad para codificar y decodificar archivos multimedia manteniendo el bajo consumo. Soporta codificación y decodificación de videos a 1080p .

El ARMv7 Cortex A7 es un procesador de 32 bits fabricado con tecnología de 28nm lo cual le permite conseguir un tamaño muy reducido, ideal para controlar sistemas embebidos donde un tamaño reducido es necesario. Además, este procesador es tremendamente eficiente hablando en términos energéticos, de hecho consume una quinta parte de la energía que consume su antecesor, el Cortex A8. Este procesador cuenta con una memoria interna de tipo caché de nivel 1 de 16 a 64KB para instrucciones y datos, pudiendo contar con hasta 1MB de caché de nivel 2. Tiene un rendimiento teórico de 1.9 DMIPS/MHz por núcleo.

DMIPS significa Millones de Instrucciones de tipo Dhrystone Por Segundo, siendo Dhrystone un test de rendimiento computacional sintético, desarrollado en 1984 por Reinhold P.Weicker, que tenía la finalidad de comparar la capacidad de cálculo con números enteros. Hoy en día, las versiones mejoradas de este benchmark son un referente representativo de toda la potencia de una CPU.

2.1.2. Pines GPIO

Los pines GPIO, forman un sistema de E/S (Entrada/Salida) de propósito general, lo cual permite infinidad de usos empleando esta interfaz entre la Raspberry Pi y el exterior. Uno de esos posibles usos será el de comunicar nuestra Raspberry con el resto de elementos actuadores y recolectores de información de nuestro sistema global.

En la página web de element14, encontramos la Figura 2.3, la cual muestra el mapa de pines GPIO que podemos encontrar en la Raspberry Pi 2.

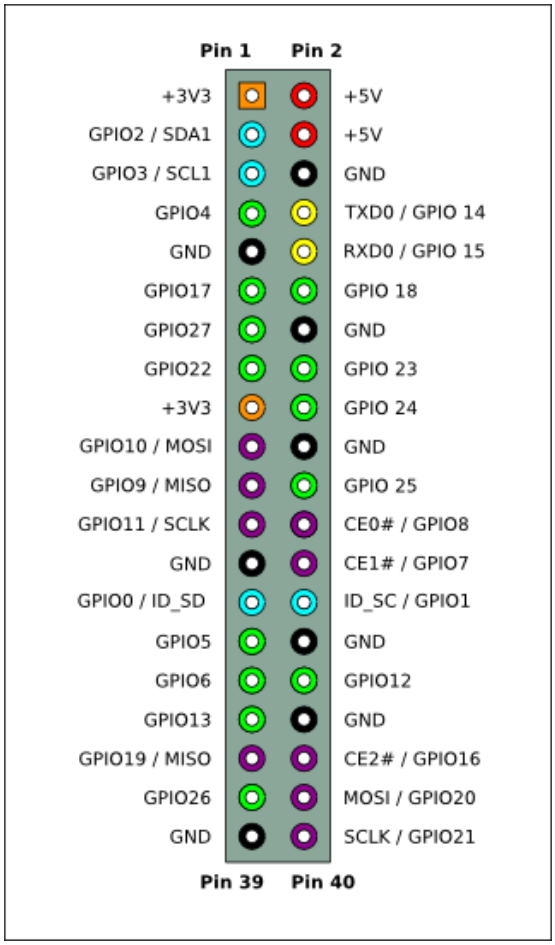


Figura 2.3: mapa de pines GPIO de la Raspberry Pi 2

Como podemos ver en la Figura 2.3, no todos los pines realizan la misma función. El abanico de posibilidades que ofrecen estos pines GPIO respecto a su contacto con el mundo exterior es tan amplio que hace que las funciones de los mismos sean de lo más variadas.

Existen pines que simplemente se limitan a dar corriente al circuito como lo haría cualquier batería, así pues los pines número 2 y 4 alimentarán el circuito con 5V mientras que los pines 1 y 17 lo harán a 3,3V (limitados a 50mA). Es importante señalar que todos los pines de la Raspberry carecen de buffers de protección y un voltaje o intensidad inadecuada dañarán la placa con facilidad. Por supuesto, para completar cualquier circuito eléctrico necesitaremos unos pines que actúen como tierra (Ground o GND), estos pines están distribuidos a lo largos de los 40 disponibles en los números: 6, 9, 14, 20, 25, 30, 34 y 39.

Los pines de la Raspberry Pi 2 también permiten la comunicación empleando el protocolo I2C, cuya principal característica es que emplea dos líneas (suponiendo una masa común) para transmitir la información: una línea de datos (pin número 3) y otra línea para la señal del reloj (pin 5). Esta comunicación posee una velocidad genérica de 100Kbits/s y se utiliza esencialmente entre dispositivos que pertenecen al mismo circuito o para comunicar microcontroladores con sus periféricos.

La interfaz UART (Universal Asynchronous Receiver-Transmitter o Transmisor-Receptor Asíncrono Universal) se encuentra en los pines 8 y 10, actuando estos de transmisor (TXD0) y receptor (RXD0) respectivamente. La UART se encarga del control de puertos y dispositivos serie.

El protocolo de comunicación serial SPI (Serial Peripherical Interface) también está disponible a través de los pines GPIO de nuestra Raspberry. El SPI es un protocolo síncrono empleado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos. Con este estándar se puede controlar un gran número de dispositivos electrónicos digitales siempre y cuando acepte un flujo de bits serie regulado por un reloj. La sincronización y la transmisión en este protocolo requieren cuatro líneas: el reloj (pines 24 y 40), una salida de datos del Maestro y entrada de datos del Esclavo (MOSI en los pines 19 y 38), una salida de datos del Esclavo y entrada al Maestro (MISO en los pines 21 y 35) y por último una cuarta línea que seleccione un Esclavo (pines 24, 26 y 36).

La placa de la Raspberry Pi permite la anexión de diferentes módulos hardware a fin de expandir su funcionalidad, a estos módulos se los conoce como HAT (Hardware Attached on Top) y serán automáticamente detectados gracias a que los pines 27 y 28 (ID_SD e ID_SC respectivamente) se encargan de la configuración y ofrecen la posibilidad de conectar un PROM serial, lo cual permite a la tarjeta de expansión identificarse con la Raspberry para cargar los drivers que sean necesarios y almacenarse en la memoria EEPROM (Electrically Erasable Programmable Read-Only Memory) del HAT. Las memorias EEPROM son memorias no volátiles que permiten su programación, borrado y reprogramación eléctricamente, en otras palabras, las memorias EEPROM son las versiones primitivas de las actuales memorias flash.

El resto de pines (7, 11, 12, 13, 15, 16, 18, 22, 29, 31, 32, 33, 35, 36, 37, 38 y 40), son pines de propósito general que desempeñarán funciones configurables en función de cómo sean programados. Estos pines serán los empleados en el manejo de elementos que no requieran una interfaz de comunicación especial. Para el desarrollo del proyecto estos pines son los más importantes, ya que codificaremos programas (scripts escritos en Python) para hacer uso de estos pines desempeñando tareas como: leer temperatura y humedad, activar actuadores, manipular un servomotor…

En esta sección los pines han sido referenciados según la numeración física o modo BOARD. Existe otra forma de referenciar los pines llamada BCM, que es la numeración del chip SoC Broadcom que los controla, de ahí que, por ejemplo, el nombre del pin físico número 11 en la Figura 2.3 sea nombrado como GPIO17. Sin embargo, a la hora de codificar los scripts en Python se ha empleado el sistema de numeración BCM.

Respecto a la nomenclatura de los pines, en ocasiones se puede apreciar una referencia doble a un mismo pin. Un ejemplo de ello es el pin físico número 3, que además de ser el pin GPIO2 (según la numeración del chip) de uso genérico, también es la línea de datos (SDA1) del interfaz I2C.

2.1.3. Conectividad

En este apartado se van a describir los demás componentes hardware que aún no han sido detallados. Básicamente son elementos referentes a la conectividad y necesarios para anexionar a nuestra placa los periféricos requeridos para comunicar esta con el mundo exterior: teclado, ratón, monitor, cámara…

Empezando por las conexiones USB, en esta segunda versión de la Raspberry Pi encontramos cuatro puertos USB 2.0 gestionados por un microchip LAN9514-JZX, que también gestiona el puerto Ethernet (proporciona una conectividad de 10/100Mbps) pudiendo encontrar un conector RJ-45 junto a los 4 puertos USB.

La alimentación de la placa, al carecer esta de botones físicos que la enciendan o apaguen, se sirve de un conector microUSB que proporciona 5V de tensión, concretamente y para evitar problemas de alimentación, se recomienda el uso del cargador oficial de Raspberry Pi, el cual proporciona una salida de 5V y 2A. Además de botones, la Raspberry también carece de un disco duro, es por ello que en su parte inferior alberga una ranura para insertar una tarjeta microSD que además de almacenar los datos del usuario también contiene el SO.

A la Raspberry se le puede añadir una pequeña pantalla LCD directamente a la CPU de la placa a través del conector DSI (Display Serial Interface). Esta no es la única manera de conectar la Raspberry a una pantalla, ya que a través del conector HDMI se permite la conexión de un dispositivo compatible con la interfaz HDMI 1.3 y 1.4 para la extracción de video y audio. Además de la salida de audio mediante el conector HDMI, la placa posee un conector de audio Jack de 3.5mm.

Por último, la placa dispone de un conector CSI (Camera Serial Interface), mediante este conector tipo bus de 15 pines conectaremos la cámara oficial de la Raspberry Pi, que tendrá un papel importante en la elaboración de este TFG.

2.2. Software

La mayor parte de Sistemas Operativos que podemos encontrar para Raspberry son del tipo GNU/Linux, no obstante y debido al rotundo éxito de Raspberry y el Internet de las cosas, el propio Microsoft ha lanzado una edición de Windows 10 destinado a estos miniPC: Windows 10 IoT Core. En esta sección se darán unas pequeñas pinceladas de las principales distribuciones existentes y se justificará la elección de Raspbian para este TFG.

2.2.1. Raspbian OS

Raspbian OS es una de las distribuciones más veteranas y sin lugar a dudas la más popular en la comunidad Raspberry Pi, lo que hace que probablemente sea la distribución más completa, optimizada y estable de todas las existentes. Raspbian OS se basa en Debian 7.0 pero con las optimizaciones necesarias para adaptar el código a la Raspberry. Una clara ventaja de esta distribución frente a otras, y que en parte ha sido motivo de elección para el desarrollo de este TFG es su intuitivo menú “raspi-config” que permite configurar ciertos parámetros del SO sin la necesidad de modificar sus archivos de configuración manualmente. Además en la distribución podemos encontrar herramientas de desarrollo como el IDLE para Python, lenguaje que se empleará para la mayor parte de las comunicaciones con los pines GPIO.

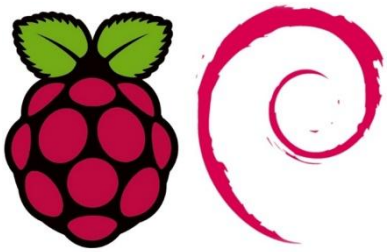


Figura 2.4: Raspbian = Raspberry Pi + Debian

Es por las características indicadas en el párrafo anterior que esta distribución ha sido la elegida, concretamente se ha instalado Raspbian Jessie que presenta algunas mejoras respecto a Raspbian Wheezy, su anterior versión:

- Arranque gráfico por defecto.

- Mejoras de velocidad en el entorno gráfico.
- Acceso a los pines GPIO sin necesidad de ejecutar las órdenes como administrador, lo cual resulta más cómodo a la hora de trabajar con ellos. Sin embargo, en busca de una mayor portabilidad se ha obviado esta característica y al hacer uso de los pines GPIO se han ejecutado las órdenes como administrador.

2.2.2. OpenELEC

OpenELEC (Open Embedded Linux Entertainment Center) es una ligera distribución que tiene la particularidad de satisfacer uno de los mayores usos que la comunidad de usuarios da a la Raspberry Pi: implementar un centro multimedia de bajo coste aprovechando su conexión HDMI. Esta distribución está optimizada para este tipo de tareas audiovisuales, con un rápido arranque, una interfaz sencilla y la preinstalación de una aplicación llamada Kodi o XBMC (Xbox Media Center), una aplicación que inicialmente se creó como centro multimedia para la videoconsola Xbox y que su desarrollo lo ha convertido en una aplicación multiplataforma muy extendida que posee extensos catálogos de películas, series, canales de televisión en directo, informes meteorológicos… Todo ello se consigue mediante paquetes que se instalan en la aplicación y la complementan.

2.2.3. Windows 10 IoT Core

Con Windows 10 IoT Core Microsoft apuesta por la plataforma ARM y por ello ha adaptado su Sistema Operativo Windows 10 a este conjunto de instrucciones reconociendo así el prometedor futuro de Raspberry Pi. Este SO sólo es compatible (por el momento) con la Raspberry Pi 2, debido al salto de Raspberry al juego de instrucciones ARMv7 que es el que Microsoft utiliza para desarrollar Windows RT (SO desarrollado para tablets y móviles). Por último, indicar que la versión de Windows 10 que podemos instalar en una Raspberry Pi 2, no es la misma versión que instalamos en las computadoras actuales, más bien es una versión destinada a los desarrolladores, con abundantes recursos de documentación pero que carece de un menú de inicio y no permite la ejecución de las aplicaciones tradicionales que ejecutaríamos en el Windows 10 de nuestra computadora.

2.2.4. RISC OS

RISC OS es otro ejemplo de SO no basado en Linux, concretamente fue diseñado por la compañía británica Acorn Computers (creadores de ARM) específicamente para ser usado en dispositivos con arquitectura ARM. Este Sistema Operativo mantiene su código fuente gracias a RISC OS Ltd. con una licencia de código abierto.

2.2.5. Retropie

Debido al resurgir de lo retro implantado en la sociedad, nos encontramos ante un Sistema Operativo completamente en auge y apoyado por una enorme comunidad: Retropie. Retropie es un Sistema Operativo basado en Raspbian que tiene una capa de personalización con una interfaz que nos permite ejecutar diferentes emuladores de un sinfín de videoconsolas antiguas: Atari 800, Atari 2600, Game boy, NeoGeo, NES, SNES, Nintendo 64, Sega Master System, Sega Mega Drive, Play Station…

Una característica importante de Retropie es que permite utilizar gran cantidad de mandos de otras videoconsolas gracias a un menú en el que se configuran los botones del mismo. También destacamos que tanto Retropie como sus emuladores son de código abierto, con lo que cualquier usuario puede contribuir al desarrollo de la plataforma.

2.3. Componentes del sistema

En la sección anterior se ha profundizado con cierto nivel de detalle en la Raspberry Pi, eje de este TFG, no obstante existen otros componentes también necesarios para el funcionamiento del sistema, como son los sensores que captarán información del exterior, la cámara de vigilancia, el servidor que dará soporte al sistema, la base de datos… El propósito de esta sección es dar a conocer el resto de componentes que giran en torno a nuestra Raspberry Pi y completan el sistema RaSpy.

2.3.1. Módulo de vídeo Raspberry Pi Camera

En este proyecto de seguridad y monitorización basado en Raspberry Pi, se pretende dotar al sistema de una videocámara con el fin de ofrecer videovigilancia en tiempo real a la que se pueda acceder desde el sitio web del proyecto. A la placa Raspberry se le puede añadir cualquier cámara web de modo similar al que lo haríamos en otra computadora, mediante la conexión USB. Sin embargo he optado por que la cámara de este proyecto sea la que ofrece la organización Raspberry Pi de forma oficial, la Raspberry Pi Camera (Figura 2.5). Esta elección se debe a las siguientes razones: esta cámara se conecta a través de la conexión CSI disponible en la placa para la incorporación de esta cámara, es susceptible de ser manejada mediante programación en Python y además permite trabajar de un modo nativo con los productos de la organización Raspberry Pi.

Figura 2.5: Raspberry Pi Camera

Las principales especificaciones de la Raspberry Pi Camera son:

- Sensor con una resolución de 5 megapíxeles.
- Resolución de imágenes fijas: 2592 x 1944.
- Resolución de vídeo máxima: 1080p.
- Frecuencia de imagen máxima: 30 fps.
- Tamaño: 20 x 25 x 10mm.
- Peso: 3g.
- No dispone de micrófono.

Existe otra cámara oficial de la organización Raspberry Pi, se trata de la Raspberry Pi Camera NoIR. Este modelo se caracteriza por ser sensible a la radiación por infrarrojos de onda corta, con lo cual nos permite grabar con ausencia de luz. Sin embargo, y a pesar de esta característica, se descartó esta opción debido a que en condiciones normales de luminosidad las imágenes capturadas son sensiblemente inferiores a las que se toman con la cámara seleccionada. Además, se puede emplear la domótica para hacer que el habitáculo donde se instale la cámara de videovigilancia siempre goce de buenas condiciones lumínicas mediante un sensor de luminosidad que a partir de cierto valor mantenga encendida una lámpara.

La Raspberry Pi Camera será introducida en una carcasa (ver Figura 2.6) diseñada para ella por dos motivos: colocarla con facilidad sobre un servomotor para permitir que la cámara enfoque en distintas posiciones y evitar daños en el componente.

Figura 2.6: carcasa Raspberry Pi Camera

2.3.2. Sensor de temperatura y humedad (DHT-11)

Se ha elegido el sensor DHT-11 debido a que presenta una excelente relación calidad/precio y además de medir la temperatura, es capaz de medir la humedad relativa del ambiente, lo cual puede resultar interesante cuando se realizan las mediciones en una sala de servidores para controlar que el funcionamiento de los equipos se encuentre dentro de unos rangos adecuados de temperatura y humedad. El sensor DHT-11 se caracteriza por tener la señal digital bien calibrada (la calibración se realiza en laboratorios) que proporciona resultados rápidos y precisos.

El sensor DHT-11 posee un microcontrolador de 8 bits y dos sensores (el de temperatura y el de humedad) resistivos encapsulados en una característica caja azul. Existen varios modelos de este sensor en el mercado, algunos vienen sueltos y otros están soldados en una placa que monta una resistencia que actuará como *pull-up* (he optado por uno de estos últimos ya que simplifica el montaje) y un diodo led que indica que si sensor está encendido.

Otro punto a favor de este sensor es que se puede conectar directamente a los pines GPIO de la Raspberry Pi ya que el protocolo de comunicación se realiza a través de un único hilo, siendo sus conexiones las que vemos en la Figura 2.7:

Figura 2.7: sensor de temperatura y humedad DHT-11

Estas son sus principales características:

- Rango de funcionamiento de la temperatura: 0ºC - 50º.
- Rango de funcionamiento de la humedad relativa: 20% - 90%.
- Tensión de alimentación: 3.3V o 5V.

- Bajo consumo.
- Bajo precio.
- Transmite la señal a una distancia de hasta 20m.
- Permite una tasa de refresco en las lecturas de 2 segundos.

Con los datos recogidos gracias a este sensor se poblará una tabla de la base de datos periódicamente con los valores correspondientes a la temperatura y a la humedad relativa del habitáculo y en caso de que alguno de estos se encuentre fuera del rango establecido se habilitará una respuesta que típicamente será la activación de algún actuador y el envío de una alerta a través del sistema de mensajería instantánea.

2.3.3. Detector de gases (MQ-135)

Se ha seleccionado este detector de gases debido fundamentalmente a su correcta relación calidad/precio (no olvidemos que estamos implementando una instalación de bajo coste, ya que en el mercado existen soluciones ya elaboradas, pero de coste muy superior). Este detector de gases MQ-135 (Figura 2.8) es capaz de detectar benceno, sulfuros, amoníaco, alcohol, butano y metano además de humo.

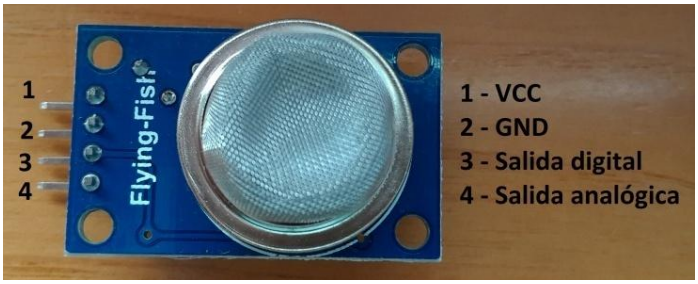


Figura 2.8: detector de gases MQ-135

Además de su amplia gama de gases tóxicos detectados, una característica que hace del MQ-135 un detector muy interesante es que dispone tanto de salida analógica como digital. Se alimenta con 5V y posee unas dimensiones de 32 x 22 x 30mm.

En nuestro caso emplearemos la salida digital del MQ-135 y su funcionamiento básicamente consistirá en que periódicamente "lea" la situación del habitáculo y sólo en caso de detectar la presencia de humo active los correspondientes actuadores y alertas a través del sistema de mensajería en tiempo real.

2.3.4. Adaptador USB wifi (Edimax EW-7811UN)

La Raspberry Pi 2 que hemos utilizado en este TFG, como hemos visto anteriormente, no dispone de un módulo wifi integrado, con lo que hay dos opciones para conectarla a red: emplear un adaptador USB wifi o conectarla mediante un cable de red Ethernet.

Debido a la disposición de la habitación donde se ha desarrollado el proyecto se ha optado por emplear un adaptador USB wifi ya que la habitación no dispone de una toma de red, concretamente se ha empleado el pequeño adaptador Edimax EW-7811EW (Figura 2.9), que a pesar de sus 27g y su reducido tamaño (7.1 x 14.9 x 18.5mm) consigue una tasa de datos de hasta 150Mbps como se puede ver en sus especificaciones técnicas. Respecto a la cobertura es aceptable, recibe en torno al 85% a una distancia de 8m del router y superando una pared maestra de la casa y algún aparato eléctrico como el televisor o un equipo de música. Este adaptador es compatible con los estándares 802.11b/g/n.

Figura 2.9: Edimax EW-7811EW

Una de las ventajas de este adaptador es que con una red correctamente configurada con una IP fija (será visto en el siguiente capítulo), no necesita de una configuración especialmente compleja.

2.3.5. Tarjeta de memoria microSD (SanDisk Ultra microSDHC 32GB Clase 10)

Nuestra Raspberry Pi soporta tarjetas de memoria de hasta 64GB, sin embargo en este componente claramente hay que prestar más atención a la velocidad que a la capacidad. Se ha seleccionado la "SanDisk Ultra microSDHC 32GB Clase 10" (Figura 2.10) debido a que gracias a la Clase 10 garantiza una velocidad de transferencia mínima de 10 MB/s, lo cual es altamente recomendable para favorecer la fluidez del sistema.

Figura 2.10: Sandisk Ultra microSDHC 32GB Clase10

En el apartado de la capacidad, con 16GB (o incluso con 8GB, sólo que con más limitaciones de almacenamiento) puede funcionar perfectamente un Sistema Operativo actual como el Raspbian OS instalado en nuestra placa, la elección de los 32GB de capacidad que presenta nuestra tarjeta se debe principalmente a una buena oferta en el momento de adquirirla, aunque bien es cierto que su utilidad está justificada cuando se destina la placa a un uso multimedia.

Nuestra distribución, Raspbian OS Jessie, se descarga en un fichero .zip que ocupa 1.44GB que al descomprimirse contiene una imagen .img de 4GB. Una vez instalada la imagen en nuestra Raspberry ocupa 3.4GB, es por ello que a partir de 8GB el sistema funcionaría sin complicaciones. No obstante, desde mi experiencia recomendaría el uso de tarjetas de 16GB si se pretende almacenar contenido. Por otra parte, cabe destacar que se puede trabajar con unidades de almacenamiento externo como lo haríamos trabajando desde nuestras computadoras personales.

2.3.6. Micro servo 9g SG90 Tower Pro

Nuevamente se ha optado por este micro servo buscando una adecuada relación calidad/precio del componente. Además, es un componente recomendado para el aprendizaje o para la realización de pequeños proyectos ya que su modo de operar es similar al de otros servos profesionales sólo que con unas dimensiones mucho menores que propician unos requerimientos de energía bastante reducidos, permitiendo ser alimentado desde la propia placa.

La función de este pequeño servomotor (ver Figura 2.11) en el sistema consiste en permitir que la cámara de vigilancia no sólo enfoque en una dirección. La Raspberry Pi Camera será introducida en su soporte y este se colocará sobre el pequeño servomotor (sus reducidas dimensiones no impiden que funcione correctamente con el peso de la cámara y su soporte), permitiendo así que el ángulo de visión de la cámara coincida con el ángulo de rotación del servo (180º) y no esté en una posición fija. Concretamente la cámara podrá enfocar en tres posiciones diferentes acordes con las tres posiciones que podremos fijar en el micro servo: hacia la izquierda, centrado y hacia la derecha.

Sus principales características son:

- Dimensiones: 22.0 x 11.5 x 27mm.
- Peso: 9 gramos.
- Torque a 5V: 1.2kg/cm.
- Velocidad de giro a 5V: 0.12s / 60º.
- Voltaje de funcionamiento: 3.0-7.2V.
- Temperatura de funcionamiento: -30℃ - 60℃.
- Ángulo de rotación: 180°.
- Ancho de pulso: 500-2400µs.
- Longitud de cable de conector: 24.5cm.

Figura 2.11: Micro servo 9g SG90 Tower Pro

2.3.7. Buzzer (YL-44)

Es uno de los elementos más sencillos de la instalación, se empleará cuando se detecte alguna anomalía en el interior de la habitación, como por ejemplo si el detector de gases percibe humo. Es decir, es un elemento actuador que se dispara cuando un elemento recolector recoge un dato anómalo.

Este buzzer (Figura 2.12) se alimenta con una tensión de 3.3V y dispone de una señal entrada/salida para el control del sonido. En ocasiones se ha sustituido (por comodidad, debido al ruido) este zumbador por otro actuador como puede ser una señal luminosa (diodo led rojo) que realiza la misma función de alertar cuando algo no va bien en el sistema.

Figura 2.12: buzzer YL-44

2.3.8. Sistema de refrigeración

Ante ciertas circunstancias la Raspberry Pi debe ser capaz de accionar un sistema de refrigeración con el fin de reducir la temperatura del habitáculo. Este sistema de refrigeración será representado mediante un pequeño ventilador que funciona con 5V (el voltaje de salida de los pines GPIO estándar es menor, en torno a los 3.3V, con lo cual la velocidad de giro del ventilador se ve reducida, no obstante es perfectamente válido como elemento de simulación del sistema de refrigeración). Como vemos en la Figura 2.13, el pequeño ventilador sólo presenta dos conexiones: corriente (cable rojo) y tierra (cable negro).

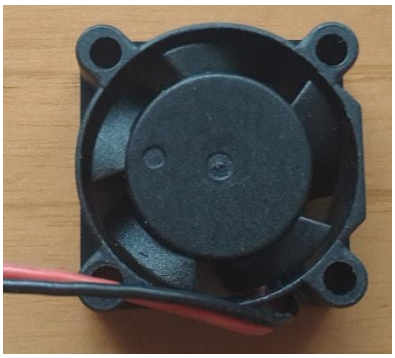


Figura 2.13: ventilador

Debido a la construcción del ventilador y a la velocidad de movimiento de sus aspas a veces resulta complicado ver con claridad si está activado o no. Como ya hicimos con el buzzer, en ocasiones sustituido por un diodo led rojo, este ventilador también será sustituido en ocasiones por un diodo led verde que pasaría a representar el sistema de refrigeración.

2.3.9. Placa protoboard, resistencias, diodos, cables y carcasa

Es habitual emplear placas protoboard cuando se implementan circuitos eléctricos debido a que ofrecen una gran comodidad a la hora de "pinchar" sobre ellas los elementos que forman el circuito. Esto es así ya que las placas protoboard conectan sus entradas en serie de forma horizontal o vertical dependiendo de la placa. Aplicando este proyecto en producción, no sería adecuado agrupar todos los sensores en la misma placa ya que estos debieran situarse estratégicamente en los lugares más adecuados, no obstante, para comprobar el correcto funcionamiento de los circuitos es perfectamente válido el uso de estas placas.

A pesar de que la Raspberry tiene una salida de 5V y otra salida de voltaje inferior (3,3V) puede resultar un voltaje excesivo para tratar con algunos elementos (diodos led, por ejemplo), en estos casos conectaremos resistencias en serie antes de que el voltaje le llegue al elemento en cuestión. Existen integrados, como el que emplearemos con el sensor DHT-11 que básicamente contienen el sensor con su correspondiente resistencia en una misma placa y de este modo se simplifica el esquema de conexiones.

A lo largo de la implementación del proyecto, ha habido numerosas ocasiones donde debido a la falta de un material, para realizar una simple prueba o por comodidad, se han simulado acciones con el encendido y

apagado de un diodo led. Como hemos visto anteriormente, los sistemas de refrigeración y la alarma acústica en ocasiones han sido reemplazados por diodos led verdes y rojos respectivamente.

Resulta recomendable la utilización de varios tipos de cables, de diferentes colores para que las conexiones sean fáciles de reconocer. Del mismo modo, resulta interesante poseer cables con diferentes conexiones (macho-macho, macho-hembra y hembra-hembra), así por ejemplo, si queremos conectar la salida de 5V de nuestra Raspberry con la línea de potencia de la placa protoboard, se recurre al un cable macho-hembra que emplea la conexión macho "pinchada" en la placa y al otro extremo, la conexión hembra conectada con el correspondiente pin GPIO de la Raspberry.

Finalmente, resulta muy importante extremar las precauciones a la hora de manipular las placas Raspberry Pi. Estas placas presentan sus circuitos integrados sin ningún tipo de aislamiento, con lo que un manejo indebido en el que podemos poseer electricidad electroestática podría deteriorar la placa e incluso hacerla inservible. Para proteger las placas, además de descargarnos (tocando alguna superficie metálica) o emplear guantes que aíslen, existen infinidad de carcasas que protegen los integrados dejando accesibles las conexiones de la Raspberry.

2.3.10. Servidor Apache

El proyecto podrá ser gestionado mediante su correspondiente sitio web, es por ello que se requiere de un servidor web para enviar páginas web en la World Wide Web. Concretamente se va a trabajar con Apache, un servidor web HTTP de código abierto multiplataforma y que es mantenido y desarrollado por sus propios usuarios bajo la supervisión de Apache Software Foundation dentro del proyecto HTTP Server (httpd).

Además de por la familiarización con este servidor debida a otros proyectos, la elección de Apache también radica en su amplia difusión (es el servidor HTTP más usado). Concretamente es el componente de servidor web en extendidas plataformas como LAMP o WAMP, junto con MySQL para la parte de base de datos y los lenguajes de programación Perl, Python y PHP, estos dos últimos serán los principales lenguajes de programación con los que se trabajará en este TFG.

Se empleará PHP (versión 5.6.17) para la programación relacionada con la parte del servidor, mientras que Python (versión 2.7.9) será el lenguaje de programación para implementar programas que realicen la comunicación entre los diferentes sensores y los GPIO de la Raspberry debido a que existen muchas librerías en Python que facilitan el trabajo con los GPIO y los sensores.

Para la realización de este proyecto se ha trabajado con la versión 2.4.10 del servidor Apache.

2.3.11. Base de Datos MySQL

Se ha seleccionado MySQL como Sistema Gestor de Bases de Datos (SGBD) debido a que está considerado como el SGBD de código abierto más popular y por la familiarización con el lenguaje SQL. MySQL presenta un desarrollo bajo licencia GPL y también con licencia comercial de Oracle Corporation, quien posee los derechos de la mayor parte del código. Es por ello que existen dos versiones de distribución: las versiones *Enterprise* que incluyen herramientas y servicios privados y las versiones *Community*, que se distribuyen bajo la licencia pública general de GNU.

La gestión de la base de datos se ha llevado a cabo mediante consultas naturales en el lenguaje SQL y también mediante la herramienta phpMyAdmin (versión 4.2.12), haciendo uso de su interfaz gráfica accesible mediante páginas web.

Se ha empleado la versión 5.5.44 de MySQL *Community* en la realización de este TFG.

2.3.12. Servicio de mensajería instantánea Telegram

Se ha seleccionado Telegram como plataforma para llevar a cabo la comunicación en tiempo real con el sistema RaSpy. Esta comunicación básicamente consiste en hacer llegar al usuario alertas: bien sean generadas automáticamente (cuando por ejemplo, la temperatura de la habitación es excesiva y se notifica enviando un mensaje a través de Telegram) o manualmente (cuando se hace uso de la aplicación web para enviar un mensaje a través de Telegram, por ejemplo la persona responsable del mantenimiento de la sala).

Telegram es una plataforma de código abierto que he considerado que presenta algunas características que la diferencian de su mediatizado rival Whatsapp y justifican su elección:

- La nube de Telegram permite el uso multidispositivo, sin limitaciones. Con lo cual, podremos manejar Telegram (mediante comandos) desde la Raspberry con la misma funcionalidad que presenta la aplicación para dispositivos móviles.

- Soporte para bots. Crearemos usuarios virtuales que harán los roles de diferentes personas receptoras de las alertas del sistema: un jefe de fábrica, un encargado de mantenimiento… Sin necesidad de utilizar números de teléfono reales.

- Grupos más grandes (de hasta 200 miembros). Dependiendo de las necesidades, pudiera ser que las alertas tengan que ser recibidas por un número elevado de destinatarios, ante esta situación sería conveniente crear un grupo con el que se comunicaría RaSpy, siendo conveniente que el grupo pueda ser numeroso atendiendo a las necesidades de la organización.

- Posibilidad de enviar archivos más grandes (de hasta 1.5GB), pensando en que bajo algunas circunstancias fuese necesario el envío de videos.

En la realización del TFG se empleará la versión 3.17.1 de Telegram para Android y en la Raspberry Pi se ha instalado la versión 1.0.

2.4. Configuración

Ya presentados los componentes que forman el sistema junto con sus características principales, en esta sección se pretende informar sobre los aspectos de configuración necesarios de cada elemento para su correcto funcionamiento y su correcta interacción con los demás componentes.

También se detallarán los montajes eléctricos en esta sección. Para que las conexiones entre los elementos resulten más fáciles de identificar, en lugar de mostrar imágenes de la instalación real, se hará uso de la herramienta Fritzing.

Fritzing es una utilidad de código abierto que sirve para realizar diseños electrónicos. Este programa posee una amplia librería de elementos que guardan gran similitud con los reales, permitiendo la realización de montajes eléctricos virtuales y sus correspondientes simulaciones con gran nivel de detalle.

2.4.1. Configuración Raspberry Pi

La Raspberry Pi requiere ser configurada para hacer un uso pleno de ella, ya que por defecto puede haber desajustes en horario, idioma del dispositivo, idioma del teclado, puede precisar de actualizaciones…

Es recomendable tener siempre actualizada la Raspberry ya que estas actualizaciones introducen mejoras y corrigen los fallos existentes en el sistema. La forma de actualizar una Raspberry con SO Raspbian se basa en repositorios, son los repositorios lo primero que deberemos actualizar (antes de actualizar los programas instalados) mediante el siguiente comando:

```
sudo apt-get update
```

Tras actualizar los repositorios, ya podemos actualizar los programas:

```
sudo apt-get upgrade
```

Localización

Lo habitual a la hora de instalar un Sistema Operativo como Raspbian es encontrar el dispositivo con una configuración anglosajona. Por ello, en esta sección se va a mostrar la configuración del dispositivo para su uso en España.

En primer lugar cambiaremos la codificación del idioma, que se encuentra en inglés de Reino Unido. Para ello ejecutaremos el siguiente comando:

```
sudo dpkg-reconfigure locales
```

Aparecerá una lista con las codificaciones disponibles y elegiremos el español de España: es_ES.UTF-8. Para que nuestro teclado tome la configuración que acabamos de seleccionar hay que ejecutar el comando:

```
sudo dpkg-reconfigure keyboard-configuration
```

Para seleccionar la zona horaria, en mi caso Europa – Madrid, ejecutaremos:

```
sudo dpkg-reconfigure tzdata
```

Hay que tener en cuenta que la Raspberry Pi, a diferencia de los ordenadores, no dispone de pila, con lo cual se requiere de una conexión a Internet para que la hora se actualice. Sin conexión a Internet, cuando encendamos el dispositivo obtendremos la hora que figuraba antes de apagarlo.

Por último, vamos a indicar al sistema que las actualizaciones se descarguen desde España en lugar de desde Reino Unido, para ello hay que editar el fichero /etc/apt/sources.list y sustituir ftp.uk.debian.org por ftp.es.debian.org. Una vez cambiada la lista de repositorios, la actualizaremos con el comando visto en la sección anterior y reiniciamos el sistema para que los cambios se apliquen (práctica recomendable cuando se producen cambios en el sistema):

```
sudo reboot
```

IP fija

La IP privada de nuestra Raspberry, al igual que la de cualquier otro equipo, puede variar cada vez que se conecta a la red. Convertir nuestra IP privada en estática (o fija) nos va a proporcionar las siguientes ventajas:

- Guardaremos la IP estática en la herramienta PuTTY para conectar nuestro ordenador a la Raspberry sin necesidad de conectarla un monitor o teclado adicional: lo haremos a través de su IP.
- Los servidores virtuales se configuran accediendo al router, en dicha configuración además de indicar el protocolo y el puerto destinado a tal servicio, debemos indicar la dirección IP de nuestra red a través de la cual acceder al servidor. Si esta IP no está fijada y varía, cuando la introduzcamos en la barra de direcciones de nuestro navegador es posible que haya cambiado y no nos dirija al servidor, consecuencia de ello será que no visualizaremos el contenido alojado en él. Fijando la IP nos aseguramos de que esta no varía y siempre accederemos al servidor con la misma dirección.

A continuación se va a explicar la configuración necesaria para fijar la IP de la Raspberry Pi, conectada a la red vía wifi.

Para realizar la configuración y fijar la IP debemos editar el fichero /etc/network/interfaces y añadir lo que encontramos en la Figura 2.14:

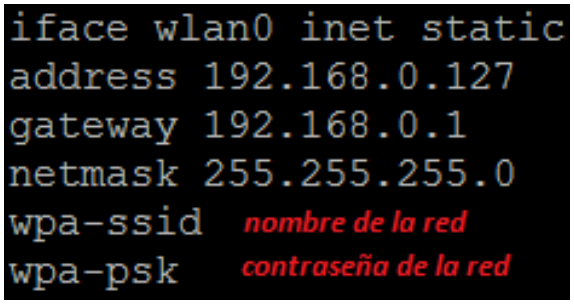


Figura 2.14: fijar IP

De este modo estamos indicando que queremos una red wifi estática o fija (*static*), esta red será la wlan0 en nuestro caso, también hay que indicar dirección de la puerta de acceso al router (192.168.0.1), la máscara de red (255.255.255.0) y la dirección IP que deseamos que se mantenga fija (en este caso la hemos fijado en el 192.168.0.127). Por último, para que conecte de forma automática a la red se añade el nombre y la contraseña de la misma.

Librería RPi.GPIO

Para controlar los pines GPIO de la Raspberry, es necesario hacer uso de una librería que nos permita realizar ese control en un lenguaje de programación: Python. Esta será la librería RPi.GPIO, que viene preinstalada por defecto en las versiones más recientes de Raspbian en su versión 0.5.11. Tras actualizar, en este TFG se ha trabajado con la versión 0.6.2.

Para poder usar la librería RPi.GPIO en nuestros programas, deberemos incluir una línea de código en los mismos para importarla:

```
import RPi.GPIO as GPIO
```

Además de importarla, con esta línea estamos indicando que referenciaremos la librería con GPIO en lugar de RPi.GPIO.

2.4.2. Configuración de No-IP y router

En este punto, nuestra Raspberry siempre va a iniciarse con la misma dirección IP privada (192.168.0.127), con lo cual es fácil reconocerla en nuestra red doméstica para acceder al servidor Apache que hemos instalado en la misma dirección IP. Sin embargo, la realización del proyecto se lleva a cabo en mi vivienda, donde no tengo contratada una IP pública estática con la compañía telefónica, con lo cual esta varía y es imposible redirigir el tráfico al servidor que hemos instalado en la Raspberry. Necesitaremos recurrir al servicio proporcionado por No-IP (https://www.noip.com).

¿Para qué sirve No-IP?

No-IP presenta un servicio especialmente útil cuando se desea hacer visible al exterior el contenido de un servidor local casero: permite registrar gratuitamente un dominio fijo de Internet para hacer más fácil el acceso a nuestra página web. No hay necesidad de introducir la dirección IP pública para acceder al servidor, concretamente, accederemos a él a través de la siguiente dirección escogida para la realización de este TFG: http://tfgjgp.ddns.net/.

Para disfrutar gratuitamente de los servicios de No-IP, lo primero que tenemos que hacer es registrarnos en su página web (https://www.noip.com). Una vez registrados podremos añadir hasta 3 hosts tan sólo rellenando los datos que aparecen en la Figura 2.15:

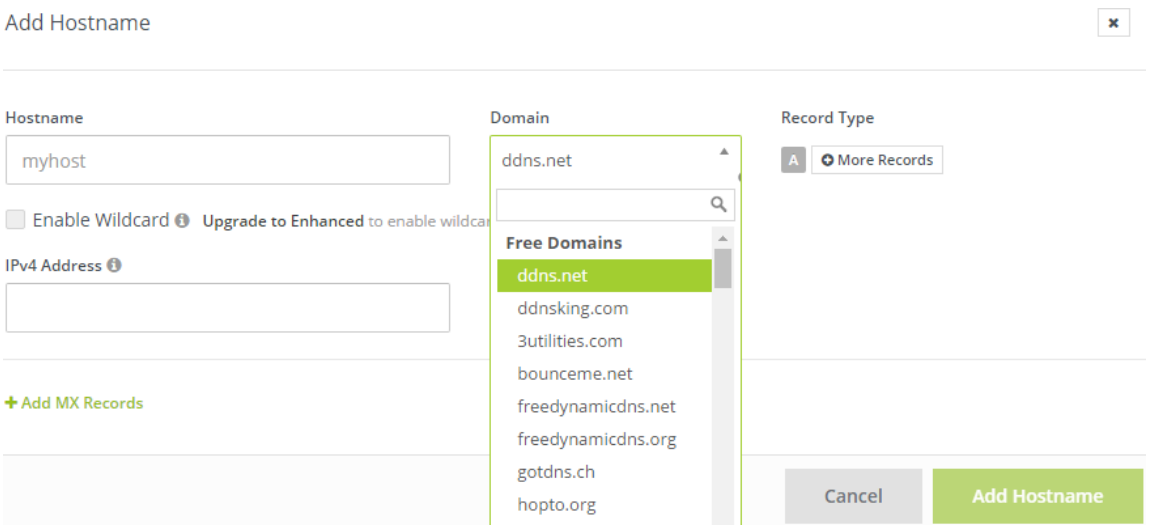


Figura 2.15: añadir host en No-IP

Como vemos, sólo debemos indicar una combinación *Hostname* + Dominio que no se encuentre en uso e indicar la IP pública que tengamos en ese momento. Sin embargo, como hemos mencionado, nuestra IP pública al reiniciar el router o con el paso del tiempo cambiará y el tráfico no se redirigirá a nuestro servidor cuando se introduzca la dirección recientemente creada, por suerte No-IP también presenta la solución a este problema ya sea instalando software adicional ("duc") que se encargue de redirigir el tráfico automáticamente a nuestro ordenador aunque cambie la IP pública, o bien mediante la correcta configuración de No-IP en nuestro router (se ha empleado este método debido a que el router permitía esta configuración).

Configurar No-IP en nuestro router

Está claro que sincronizar la dirección IP dinámica asociada a nuestro dominio fijo creado en No-IP cada vez que esta cambie es ineficiente, por ello en esta sección explicamos cómo configurar nuestro router para que haga el trabajo por nosotros.

Siempre que el router permita agregar una conexión con el servicio No-IP (nuestro caso), lo configuraremos sin necesidad de añadir software adicional. Para llevar a cabo esta configuración deberemos dirigirnos a la sección de Dynamic DNS (he trabajado con un router TP-Link TL-WR1043ND), seleccionar el proveedor del servicio (No-IP en nuestro caso, aunque el router también permite trabajar con otros servicios similares: dyndns y comexe) y completar los campos con nuestros datos ya creados en No-IP: nombre de usuario, contraseña y nombre del dominio. En la Figura 2.16 vemos la pantalla de configuración:

Figura 2.16: configuración del servicio No-IP en el router

Como vemos, es necesario tener activada la casilla DDNS (Dynamic DNS) y el propio router nos informa del correcto estado de la conexión.

2.4.3. Configuración de la Raspberry Pi Camera

Lo primero que tenemos que hacer para usar de la Raspberry Pi Camera, una vez conectada al puerto CSI, es habilitarla para que sea detectada por la Raspberry. Para ello, hay que entrar en el menú de configuración de la Raspberry, al cual se accede mediante el comando:

```
sudo raspi-config
```

Una vez dentro del menú, navegaremos hasta la sección de la cámara (*camera*) y la activaremos (*enable*). Aunque pueda resultar obvio, es indispensable que habilitemos la cámara de esta forma ya que por defecto viene desactivada (*disable*) en la Raspberry Pi.

Activada la cámara, nos aseguramos de que al reiniciar el equipo se va a ejecutar el firmware de la GPU adecuado (driver de la cámara) y que la partición de la memoria de la GPU sea suficiente para que la cámara funcione correctamente.

MJPG-Streamer

Nuestro sistema debe ser capaz de mostrar imágenes en tiempo real, es decir, necesitamos un servicio de streaming al cual accedamos desde nuestro sitio web. Para ello vamos a emplear la herramienta MJPG-Streamer, una utilidad con licencia GNU/GPL que se ejecuta por línea de comandos y toma frames JPG (imágenes) desde cámaras web compatibles con Linux para transmitirlas como M-JPG (secuencia de vídeo) mediante el protocolo HTTP. Estas secuencias de video se pueden visualizar en navegadores web o incluso en ciertos reproductores de vídeo como VLC.

MJPG-Streamer basa su funcionamiento en plugins de entrada y salida. Mientras que el plugin de entrada copia las imágenes JPEG a un directorio fijado, el otro plugin (el de salida) procesa las imágenes y las sirve como ficheros de imágenes o también como secuencia de imágenes que forman un video.

Para que la aplicación funcione correctamente lo primero que se debe hacer es instalar una serie de librerías (es recomendable que antes de instalar cualquier programa o librería actualicemos nuestra distribución de Raspbian):

- **libjpeg8-dev**, es una librería indispensable para el manejo de imágenes JPEG. Este paquete es un estándar de la industria que incluye los programas cjpeg y djpeg que sirven para comprimir ficheros JPEG en los formatos PPM, PGM y BMP. En libjpeg8-dev se incluye la utilidad jpegtran, que sirve para realizar transformaciones sin pérdidas en los ficheros JPEG. Otras utilidades de interés que se incluyen en el paquete son rdjpgcom y wripgcom, que permiten visualizar comentarios y etiquetas de texto insertados en los ficheros JPEG. Instalaremos la librería haciendo uso del comando:

```
sudo apt-get install libjpeg8-dev
```

- **imagemagick**, conjunto de utilidades de código abierto que sirven para mostrar, manipular y convertir imágenes. Es capaz de leer y escribir más de 100 formatos. Este paquete es publicado bajo la Licencia Apache. Instalaremos la librería haciendo uso del comando:

```
sudo apt-get install imagemagick
```

- **libv41-dev**, es un conjunto de bibliotecas que añade una capa de abstracción delgada por encima de los dispositivos Video4Linux (o V4L, una API de captura de video para Linux). El propósito de esta capa es simplificar la tarea a los programadores del desarrollo de aplicaciones soportadas por varios

dispositivos, sin tener que escribir código separado para cada uno de ellos (siendo del mismo tipo). Instalaremos la librería haciendo uso del comando:

```
sudo apt-get install libv4l-dev
```

Una vez instaladas las librerías necesarias, hay que descargar el código fuente (desde https://sourceforge.net/projects/mjpg-streamer/files/latest/download) de la aplicación MJPG-Streamer, descomprimirlo y compilarlo. En el momento de la compilación se generan los plugins anteriormente citados, estos plugins son:

- **mjpg_streamer:** herramienta de línea de comandos que copia las imágenes JPG del plugin de entrada a los plugins de salida.
- **input_file.so:** se encarga de capturar los frames JPG de la webcam. Este plugin es capaz de capturar imágenes a una resolución máxima de 960 x 720 píxeles con una tasa de refresco cercana a los 15fps, todo ello con poca carga sobre la CPU.
- **output_http.so:** es un servidor web HTTP 1.0 que se encarga de servir una única imagen JPEG o bien las emite según el estándar M-JPEG.

Después de compilar e instalar la herramienta en nuestro directorio de trabajo (/home/pi/mjpg), ya se puede comenzar a trabajar con la cámara. Sin embargo, antes de ello debemos subsanar un error de compilación referente a la librería videodev.h: este archivo (necesario para MJPG-Streamer) se ha sustituido por videodev2.h, con lo cual, hay que crear un enlace simbólico para que funcione correctamente la aplicación:

```
sudo ln -s /usr/include/linux/videodev2.h /usr/include/linux/videodev.h
```

Uso de la Raspberry Pi Camera en MJPG-Streamer

Para el manejo de la cámara voy a emplear una aplicación llamada raspistill, esta aplicación es la encargada de capturar las imágenes de la Raspberry Pi Camera. Se basa en línea de comandos y utiliza la API MMAL, esta API es exclusiva de Broadcom y sólo se emplea en sistemas VideoCore IV.

Para la captura de imágenes emplearemos el siguiente comando:

```
raspistill --nopreview -w 640 -h 480 -q 5 -o /tmp/stream/pic.jpg -tl 10 -t 600000 -th 0:0:0 &
```

A continuación se muestran las opciones utilizadas en el comando:

- **--nopreview:** se desactiva la ventana de la vista previa.
- **-w 640:** definición de la anchura de la imagen. 640 píxeles.
- **-h 480:** definición de la altura de la imagen. 480 píxeles.
- **-q 5:** calidad de la imagen, este valor varía de 0 a 100 donde 100 es la imagen sin comprimir. Para la realización de pruebas se toma una calidad de sólo el 5% ya que cuanto mayor es la calidad de la captura, mayor es el ancho de banda consumido y trabajamos en una red doméstica.
- **-o /tmp/stream/pic.jpg:** nombre del archivo de salida, previamente debe ser creado el directorio /tmp/stream.
- **-tl 10:** tiempo entre capturas medido en milisegundos. Cada diez milisegundos se tomará una imagen.

- **-t 600000:** tiempo durante el que se ejecutará el programa. Va ejecutarse durante 600000 milisegundos (10 minutos). Teniendo en cuenta la opción anterior (-tl 10), durante esos 10 minutos se tomarán capturas cada 10 milisegundos, transcurrido ese tiempo, la cámara dejará de capturar imágenes y será necesario que el usuario vuelva a iniciar un nuevo intervalo de 10 minutos de grabación. Esto se ha decidido así para evitar consumir recursos innecesarios si el usuario deja su sesión abierta sin detener el streaming.

- **-th:** 0:0:0: definición de los parámetros de las imágenes en miniatura (x:y:calidad).

En este punto ya tenemos a la Raspberry Pi Camera sacando imágenes cada 10 milisegundos y almacenándolas en el fichero indicado anteriormente, prueba de ello es que en la placa de la cámara se enciende un indicador led de color rojo. Es ahora turno de hacer que esas constantes capturas de imágenes puedan ser accesibles desde otro dispositivo vía Internet, para ello nos serviremos de la aplicación MJPG-Streamer ejecutando el siguiente comando:

```
LD_LIBRARY_PATH=/usr/local/lib mjpg_streamer –i "input_file.so –f /tmp/stream –n pic.jpg" –o "output_http.so –w /usr/local/www –p 8080" &
```

Estas son las opciones del commando:

- **-i:** plugin que realiza las capturas (input_file.so).

- **-f:** directorio donde se guarda la captura (/tmp/stream).

- **-n:** nombre de la captura (pic.jpg).

- **-o:** plugin que actúa como servidor web que se encarga de servir la captura (output_http.so).

- **-w:** directorio que contiene las páginas web.

- **-p:** puerto (8080).

Ahora ya nos podemos conectar a través de un navegador web y ver el streaming en directo. Lógicamente, para acceder a la página web de la aplicación tendremos que hacerlo a través del puerto 8080 (http://localhost:8080). De la página web de la aplicación (Figura 2.17), hemos extraído los fundamentos para emitir en streaming desde el sitio web propio del proyecto.

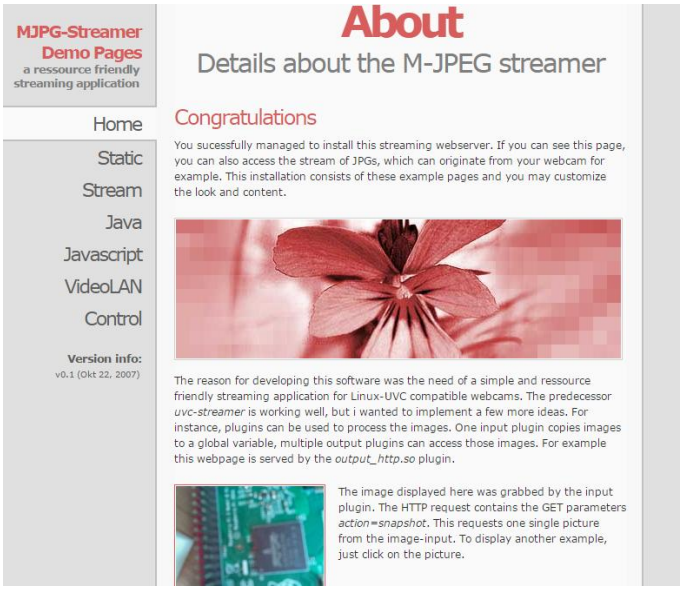
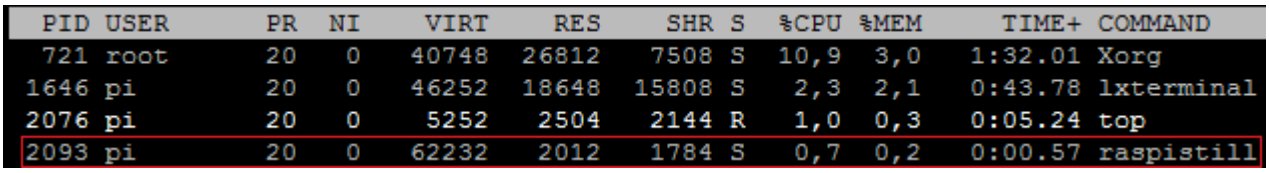


Figura 2.17: web de la aplicación MJPG-Streamer

A continuación se muestran los bajos resultados de carga que se producen en el microprocesador debido al uso de la Raspberry Pi Camera con el software empleado, concretamente veremos en la Figura 2.18 la carga producida al realizar capturas de imágenes según las condiciones mencionadas anteriormente:



PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
721	root	20	0	40748	26812	7508	S	10,9	3,0	1:32.01	Xorg
1646	pi	20	0	46252	18648	15808	S	2,3	2,1	0:43.78	lxterminal
2076	pi	20	0	5252	2504	2144	R	1,0	0,3	0:05.24	top
2093	pi	20	0	62232	2012	1784	S	0,7	0,2	0:00.57	raspistill

Figura 2.18: carga producida por la captura de imágenes

Como vemos, después de dar casi un minuto al programa raspistill capturando imágenes, se aprecia que en dicha tarea el consumo de CPU tan sólo supone un 0.7% mientras que el consumo de memoria con un 0.2% es todavía menor.

2.4.4. Configuración del sensor de temperatura y humedad

Como ya hemos mencionado, Python es el lenguaje de programación empleado en el desarrollo de un buen número de librerías que nos permiten trabajar con determinados componentes, un ejemplo de ello es el sensor de temperatura y humedad DHT-11, para el que Adafruit ha desarrollado una librería que facilita su uso. Descargamos su código:

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git
```

Para el manejo de este sensor hay que tener instaladas las librerías: build-essential (contiene una lista informativa de los paquetes que se consideran esenciales para la creación de paquetes Debian), python-dev (contiene los archivos de encabezado necesarios para crear extensiones de Python) y python-openssl (proporciona una interfaz de alto nivel a las funciones de la biblioteca OpenSSL):

```
sudo apt-get install build-essential python-dev python-openssl
```

Una vez instaladas las librerías ya podemos compilar el código descargado de la librería de Adafruit, para ello ejecutamos:

```
sudo python setup.py install
```

En teoría, ya está instalada la librería y podemos hacer un uso de prueba del sensor si nos dirigimos hacia el directorio de ejemplos:

```
cd examples

sudo python AdafruitDHT.py 11 17
```

Si la librería del sensor y el sensor DHT-11 están correctamente instalados, debería devolvernos la temperatura y la humedad leídas.

Como vemos, se están pasando dos números como argumentos en la ejecución del comando: en primer lugar debemos indicar con qué sensor estamos trabajando (11 para el DHT-11, 22 para el DHT-22 y 2302 para el AM-2302, todos ellos sensores de temperatura y humedad que pueden hacer uso de esta librería de Adafruit) y

en segundo lugar hay que indicar a qué pin GPIO está conectado nuestro sensor (en nuestro caso en el pin 17 corresponde con la conexión de datos del sensor, siguiendo la numeración BCM).

Error de acceso a los GPIO sin root

Desde Raspbian indican que su última versión (Raspbian Jessie, la que utilizamos en la realización de este proyecto), no requiere acceder a los pines GPIO como administrador. Sin embargo, aprovechamos esta puesta en marcha del sensor DHT-11 para rebatir tal afirmación, ya que si no accedemos como administrador nos encontraremos ante el siguiente error (Figura 2.19):

```
RuntimeError: Error accessing GPIO. Make sure program is run as root with sudo!
```

Figura 2.19: error sudo GPIO

Este error precisamente nos indica que nos aseguremos de ejecutar el programa como administradores ya que se produce un error en tiempo de ejecución relacionado con el acceso a los pines GPIO.

2.4.5. Configuración del micro servo

Un servo es un pequeño dispositivo que tiene un eje de salida. Este eje puede posicionarse en diferentes posiciones enviando al servo una señal que mientras exista, mantendrá la posición del servo. A medida que cambia la señal, cambia la posición angular del eje del servomotor.

Los servos se controlan enviando un impulso eléctrico de anchura variable o modulación de ancho de pulso (PWM), a través del cable de control. Existe un pulso mínimo, un pulso máximo y una tasa de repetición. Concretamente este servo sólo puede girar 90º en cualquier dirección describiendo un ángulo máximo de 180º. Definiremos la posición neutra de este servo como aquella que tiene la misma cantidad de rotación potencial (90º) tanto en sentido horario como antihorario.

El PWM enviado al servo determina la posición del eje, y en base a la duración del impulso enviado a través del cable de control el rotor girará a la posición deseada. Este servomotor en concreto espera ver un pulso cada 20 milisegundos (ms) y la longitud del pulso determinará hasta qué punto el motor gira.

En el *datasheet* de este servo encontramos que un pulso de 1,5 ms hará que el motor gire a la posición de 90º, menos de 1,5ms lo moverá hacia la posición de 0º, y más de 1,5ms girará el servo hacia la posición de 180º, como se muestra en la Figura 2.20:

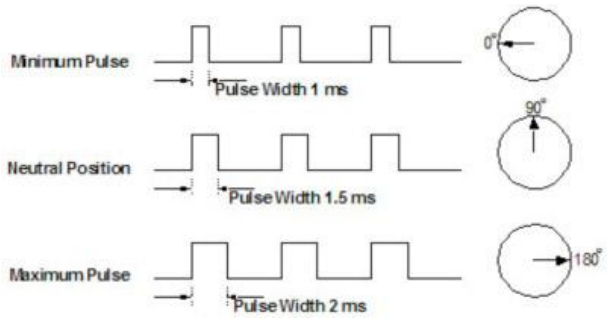


Figura 2.20: duración del impulso y ángulo de rotación

Para controlar el servomotor, vamos a utilizar el módulo PWM de la libraría RPi.GPIO. El primer paso es crear la instancia PWM asociada con el pin GPIO:

```
p = GPIO.PWM(19,50)
```

En nuestro caso se ha instanciado el módulo PWM para el pin GPIO número 19 con una frecuencia de 50 Hz. Esta frecuencia surge porque el servo espera un pulso cada 20ms (período), lo que significa 50 pulsos por segundo (hercios). Una vez instanciado el módulo PWM, para empezar a enviar un pulso y colocar el servomotor en la posición deseada empleamos la siguiente instrucción:

```
p.start(dc)
```

Donde "dc" hace referencia a las siglas inglesas del ciclo de trabajo (*duty cycle*) multiplicado por 100 (para obtener porcentajes). El ciclo de trabajo es la relación existente entre el tiempo en el que la señal se encuentra en estado activo (longitud del pulso) y el periodo de la misma. Dado que el servo utiliza ciclos de 20 ms y teniendo en cuenta los datos de la Figura 2.20 que recoge el *datasheet* del servo, podemos calcular el ciclo de trabajo de las 3 posiciones (derecha, centrado e izquierda):

$$\mathrm{dc} = \frac{0.5}{20} \times 100 = 2.5\%$$

$$\mathrm{dc} = \frac{1.5}{20} \times 100 = 7.5\%$$

$$\mathrm{dc} = \frac{2.5}{20} \times 100 = 12.5\%$$

Para detener la emisión del pulso empleamos la instrucción:

```
p.stop()
```

2.4.6. Configuración de Telegram

Lo primero que tenemos que hacer es instalar una serie de librerías que son necesarias para el correcto funcionamiento de la aplicación:

```
sudo apt-get install libreadline-dev libconfig-dev libssl-dev lua5.2 liblua5.2-dev
```

- **libreadline-dev**: ayuda en la consistencia de la interfaz de usuario a través de programas que necesitan proporcionar una interfaz de línea de comandos.
- **libconfig-dev**: esta biblioteca cuenta con un analizador totalmente reentrante e incluye enlaces para los lenguajes de programación C y C ++.
- **libssl-dev**: es la parte de la implementación del proyecto OpenSSL de los protocolos criptográficos SSL y TLS para la comunicación segura a través de Internet.
- **lua5.2** y **liblua5.2-dev**: estas librería son necesarias para hacer uso del lenguaje de programación Lua utilizado por la aplicación de Telegram para la Raspberry. Este lenguaje potente y ligero fue diseñado para extender aplicaciones. EL motor del lenguaje es accesible como una biblioteca con un API en C que permite intercambiar datos con programas Lua y extender Lua con funciones C. Lua también se puede utilizar como lenguaje autónomo a través del intérprete de línea de comandos.

Una vez instaladas, nos disponemos a clonar el repositorio GitHub de la aplicación de Telegram para la Raspberry Pi:

```
git clone https://github.com/vysheng/tg.git && cd tg
```

Situados en el directorio descargado (/home/pi/tg) ejecutamos el archivo de configuración y compilamos el programa:

```
./configure

make
```

Cuando la compilación haya terminado ya podremos iniciar Telegram desde nuestra Raspberry. Telegram requiere una contraseña pública para iniciarse, esta contraseña se encuentra en el fichero /home/pi/tg/tg-server.pub. Por lo tanto para iniciar Telegram ejecutaremos:

```
/home/pi/tg/bin/telegram-cli –k /home/pi/tg/tg-server.pub
```

Donde al ejecutable telegram-cli se le pasa con la opción –k la contraseña pública citada anteriormente. Al ejecutar por vez primera Telegram en la Raspberry nos pedirá que le asociemos un número de móvil: deberemos introducir el código del país (+34 en España) seguido de las nueve cifras del número de teléfono. Tras introducir el número, Telegram envía un código de cinco dígitos a ese número mediante un SMS para que lo introduzcamos en la consola, terminando así la configuración de Telegram en nuestra Raspberry Pi.

Configuración del destinatario

En el móvil que posee el mismo número asociado al Telegram de la Raspberry, también se ha descargado e instalado la aplicación desde el Play Store. Recordemos que Telegram es una aplicación multidispositivo, con lo cual podemos hacer las mismas acciones desde el teléfono móvil que desde la Raspberry Pi. Por ejemplo, si mandamos un mensaje a un contacto, este nunca sabrá si ha sido enviado desde la Raspberry o desde el teléfono: son idénticos.

Las alertas que harán aparición en nuestro sistema necesitan un destinatario que típicamente sería un responsable de la organización a la que se presta servicio: jefe de fábrica, encargado de mantenimiento… Lo que podríamos hacer sería instalar Telegram en otro dispositivo para que realice las funciones del destinatario, como lo podría ser cualquiera de los contactos que tenemos en nuestra agenda de la aplicación. Sin embargo, con Telegram no es necesario adquirir un nuevo número de teléfono ni tampoco sacrificar un dispositivo: la aplicación nos permite crear bots que harán las funciones del destinatario de nuestras alertas.

Por estar más familiarizado con el entorno, la creación de los bots se ha llevado a cabo desde el Telegram instalado en el dispositivo móvil. En el podemos ver que uno de los contactos es BotFather, este contacto especial está incluido por Telegram para implementar diversas funcionalidades que van desde la creación de juegos hasta la creación de bots: BotFather es un bot que nos permitirá crear bots (como sugiere su nombre).

En la Figura 2.21 se muestra el proceso de creación de un bot, que básicamente tiene tres pasos. En primer lugar indicaremos a BotFather que queremos crear un nuevo bot, para ello emplearemos el comando:

```
/newbot
```

A continuación, BotFather nos preguntará el nombre que queremos poner al bot (este nombre puede ser cualquiera) y después de indicar el nombre del bot, BotFather solicitará un nombre de usuario para el nuevo bot (en esta ocasión el nombre de usuario no puede repetirse). Una vez introducido comprobará que el nombre de usuario no esté ocupado y creará el nuevo bot.

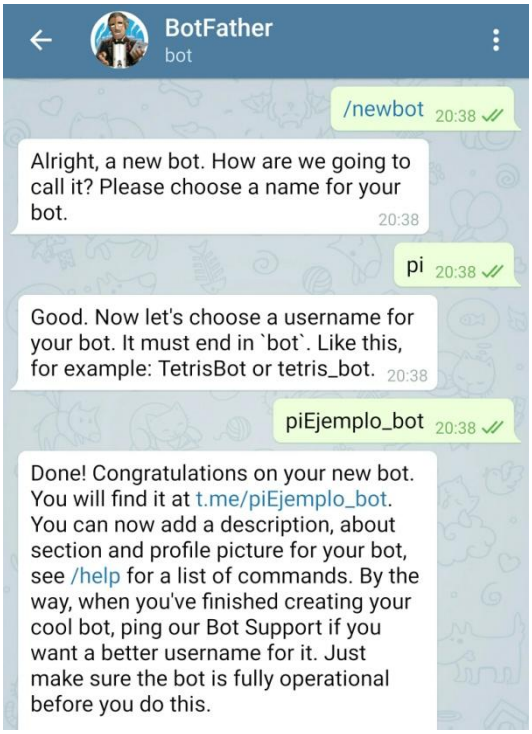


Figura 2.21: proceso de creación de un bot

Al final de la Figura 2.21 podemos ver que BotFather nos da la enhorabuena por haber creado el bot satisfactoriamente y nos facilita un enlace que al acceder inicializa el bot creado con el comando /start y a partir de ahí podemos tratar el bot como si fuese un contacto más, como se aprecia en la Figura 2.22:

Figura 2.22: apariencia del bot creado como si fuese un contacto normal

2.4.7. Configuración de FPDF

En este proyecto existe la necesidad de proporcionar al usuario los mismos informes a los que accede de forma online, pero en formato PDF. Para ello haremos uso de FPDF, una clase escrita en PHP que permite generar documentos PDF dinámicamente a través de nuestros scripts en PHP (lenguaje que hemos empleado en la parte del servidor), siempre y cuando nos encontremos, al menos, en la versión 5 de PHP. FPDF es una clase gratuita para cualquier uso, ya sea comercial o personal (su primera letra, “f”, significa *free* o libre).

Con FPDF no se puede conseguir un estilo visual grandilocuente, la clase FPDF posee unas limitaciones que hacen que la puesta en escena del documento PDF no sea tan atractiva como lo es su correspondiente informe online, sin embargo cumple con la función de presentar una gráfica con los datos y una tabla asociada con información de los mismos.

Lo primero que hay que hacer para poder empezar a usar esta clase es descargarla de su sitio web oficial: http://www.fpdf.org/. Una vez descargada, en nuestro caso la versión 1.6, obtenemos un archivo .zip que hay que descomprimir y subir al servidor, ya sea en la raíz del mismo o en otro lugar (he optado por incluir el directorio en la raíz del servidor, /var/www/html/fpdf16), es importante recordar dónde se ha colocado la clase ya que siempre que queramos utilizarla en nuestros scripts deberemos emplear la sentencia:

```
require('/var/www/html/fpdf16/fpdf.php');
```

UTF-8 en FPDF

Por defecto, FPDF no está configurado para aceptar el formato de codificación de caracteres UTF-8, por lo tanto, cuando utilizamos tildes o nuestra castiza “ñ” el PDF resultante no lo interpreta y se originan símbolos en su lugar. Para poder hacer uso de las tildes y otros caracteres especiales debemos dirigirnos al archivo fpdf.php (que se encuentra en el directorio /var/www/html/fpdf16), y añadir al principio de su función Cell la siguiente línea:

```
$txt = utf8_decode($txt);
```

2.4.8. Diagrama eléctrico

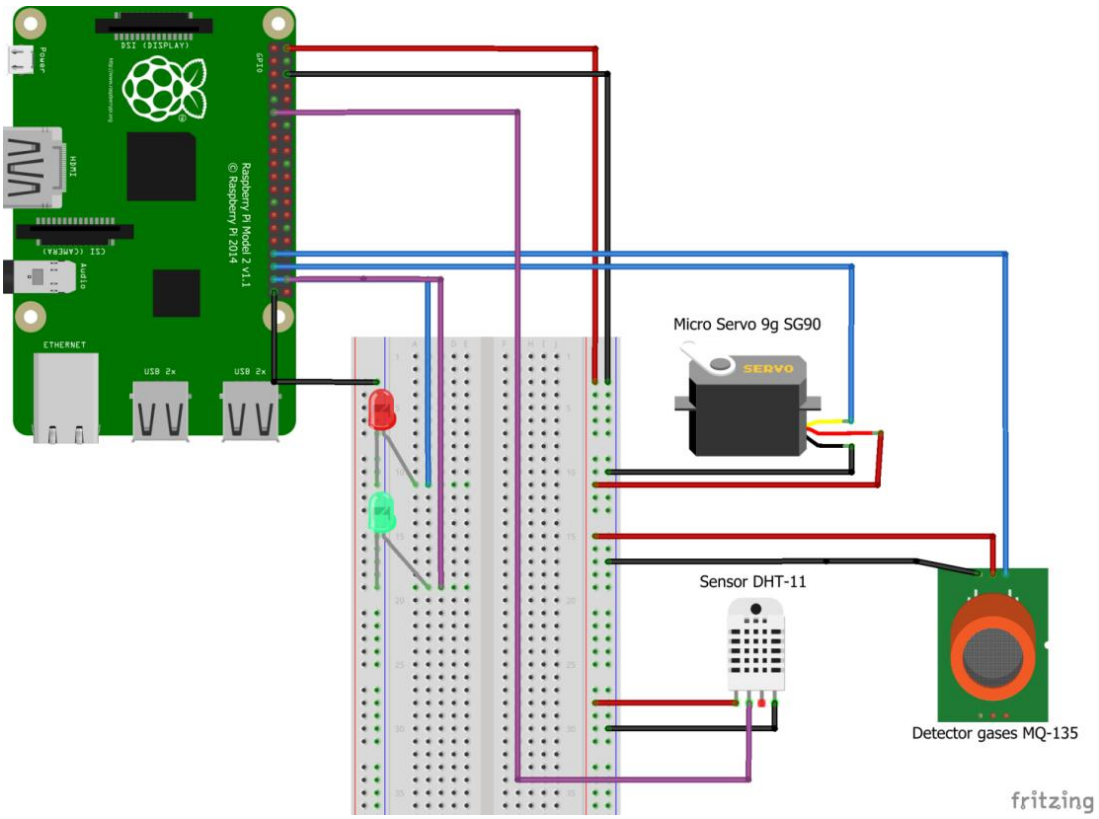


Figura 2.23: diagrama eléctrico

En la Figura 2.23 podemos ver el diagrama eléctrico que muestra el montaje de todos los elementos de nuestro sistema. Ha sido realizado con la herramienta Fritzig, esta herramienta además de permitir crear los convencionales esquemas eléctricos, da la posibilidad de implementar este tipo de diagramas que llama

“protoboard” y su principal ventaja es su similitud con la realidad: podemos ver los componentes y sus conexiones de un modo casi idéntico a como están dispuestas en el montaje real.

A continuación se explicarán una a una las conexiones de los elementos que figuran en el diagrama:

- **Diodos led**: emplearemos el diodo led rojo para hacer las funciones de la alerta (bocina) y el led verde para representar el sistema de refrigeración (ventilador). Se han conectado las conexiones del cátodo (pata más corta) de ambos diodos a tierra (GND) mientras que sus conexiones de ánodo serán controladas con los pines GPIO 26 (rojo) y 21 (verde). Estos GPIO trabajan a un voltaje de 3.3V, este voltaje aunque es elevado para los leds no los funde y permite hacer pruebas con ellos sin problemas, no obstante para alargar la vida útil de los diodos sería conveniente añadir al circuito una resistencia en serie de 220 Ω a cada uno de ellos (para no emborronar el circuito se ha omitido esta resistencia).

- **Sensor DHT-11**: la conexión del cable rojo va a corriente (5V), el cable negro a tierra (GND) y el cable morado es el cable de datos que va al GPIO 17. Recordemos que nuestro sensor DHT-11 real tiene instalada una resistencia en su placa (por eso no figura en el diagrama), de no ser así deberíamos añadir una resistencia (4,7-10 kΩ) que va desde la conexión de corriente hasta la conexión de datos.

- **Detector de gases MQ-135**: la conexión del cable rojo va a corriente (5V), el cable negro a tierra (GND) y el cable azul es el cable de datos que va al GPIO 4.

- **Micro Servo 9g SG90**: la conexión del cable rojo va a corriente (5V), el cable negro a tierra (GND) y el cable azul es el cable de datos que va al GPIO 19 (PWM).

Bloque III: Análisis

Capítulo 3: Metodología, plan de trabajo y planificación

Ya en el capítulo 1 de la presente memoria se indicó que la metodología de desarrollo de software para este proyecto será el Proceso Unificado, siendo una de las principales razones para acogernos a esta doctrina su característico desarrollo iterativo e incremental que nos permitirá desarrollar versiones cada vez más completas y complejas hasta llegar al producto final.

Las versiones parciales del producto final serán llamadas incrementos y cada nuevo incremento se construirá sobre aquellos que ya fueron entregados, de este modo evitaremos llegar al final del proyecto con tareas arriesgadas aún por desarrollar.

Cada incremento, de forma aislada presenta su propio plan de desarrollo de software con sus correspondientes fases de análisis, diseño, implementación y pruebas. Al finalizar cada incremento ya se cumplía parte de la funcionalidad total del sistema, dotando al mismo de más y más funcionalidad a medida que los incrementos iban siendo entregados hasta llegar al sistema final completo.

A continuación se muestran las tareas asociadas a los incrementos que han tenido lugar en el desarrollo de este proyecto:

Incremento 1: en el primer incremento se llevó a cabo un estudio de la viabilidad del sistema a desarrollar. Se obtuvo un boceto en el que se recogían las medidas de seguridad y la monitorización del sistema.

- **Incremento 2**: tras presentar el proyecto al tutor, se comienza a desarrollar el sitio web desde el que se manejará el proyecto y se realizan pruebas con los pines GPIO genéricos de la Raspberry Pi para familiarizarnos con su funcionamiento.

- **Incremento 3**: es en este incremento cuando se empieza a expandir el sistema al adherirle algunos de sus componentes. En este caso se añade la Raspberry Pi Camera, se trabaja con ella y con las herramientas y librerías que permiten su uso (captura de imágenes y streaming).

- **Incremento 4**: continuando con la ampliación del sistema añadiendo otros componentes, se añaden los sensores de detección de gases y de temperatura y humedad. Se trabaja con las librerías necesarias para su manejo y se desarrolla el código. También se realizan pruebas con los sensores.

- **Incremento 5**: en esta ocasión se trabaja con el apartado de las alertas. Se instala Telegram en la Raspberry, se desarrolla el código necesario para que se envíen mensajes en tiempo real al lanzar determinados scripts alojados en la Raspberry y se llevan a cabo las pruebas correspondientes. También se modifican códigos anteriores (el código del sensor de temperatura, por ejemplo) para permitir el lanzamiento automático de alertas si se cumplen determinadas condiciones.

- **Incremento 6**: la funcionalidad parcial del sistema a desarrollar en este incremento es la de permitir al usuario que genere informes de temperatura y humedad teniendo en cuenta una fecha de inicio y otra de fin. Se desarrollan los códigos necesarios y se trabajan con las librerías Highcharts para su visualización online y FPDF para que los informes puedan ser descargados en formato PDF. Se prueba su correcto funcionamiento.

- **Incremento 7**: se amplía el anterior apartado de los informes para que el sistema los genere de forma periódica y automática. Se estudia qué tareas deben ser automatizadas y cuándo deben serlo en función de la elección del usuario. En este incremento se trabaja con el programa cron del Sistema Operativo para automatizar las tareas.

- **Incremento 8**: una vez lograda la funcionalidad del sistema, en este último incremento se refinan detalles de la aplicación web con el objetivo de proporcionar una mejor experiencia a los usuarios y se recopilan todas las anotaciones surgidas a lo largo del desarrollo del proyecto para cumplimentar la memoria del mismo.

Los incrementos descritos recogen a grandes rasgos el proceso que se ha llevado a cabo para desarrollar el sistema. Aunque no se detallan las iteraciones que han tenido lugar en los incrementos, sí permite obtener una perspectiva del plan de desarrollo llevado a cabo y de las partes diferenciadas que componen el sistema de seguridad y monitorización final.

3.1. Recursos

En esta sección se realizará una división a la hora de enumerar los recursos que han intervenido en el desarrollo del proyecto. En función de la naturaleza de los mismos nos encontraremos ante recursos humanos, recursos hardware y recursos software. Se describirá brevemente la función de cada recurso.

3.1.1. Humanos

Existen dos recursos humanos relacionados con la ejecución del proyecto, estos son:

- **Alumno:** adquiere los roles de jefe de proyecto, analista, diseñador, programador y evaluador del sistema.

- **Tutor del proyecto:** se encarga de guiar al alumno en el transcurso de la elaboración del proyecto. Asesora, propone ideas y resuelve las dudas que pudieran surgir.

3.1.2. Hardware

A lo largo del desarrollo del proyecto han intervenido los siguientes elementos hardware:

- **Ordenador portátil del alumno:** se ha empleado un Asus X550JX y un Asus Eee PC 1215b para elaborar los diagramas pertinentes en las fases de análisis y diseño, para implementar el código de los diferentes programas que componen el sistema y para la realización de pruebas. También han sido utilizados para redactar la memoria y elaborar la presentación.

- **Teléfono móvil del alumno:** se ha empleado un Meizu MX4 Pro para realizar pruebas y comprobar el correcto funcionamiento del sistema de mensajería en tiempo real de las alertas a través de Telegram.

- **Raspberry Pi 2:** soporta el funcionamiento de todo el sistema.

- **Componentes del circuito eléctrico:** se han descrito previamente en la memoria, se trata de todos los elementos (sensores, cámara, actuadores…) que son necesarios para interaccionar con la Raspberry Pi y lograr la funcionalidad deseada.

3.1.3. Software

Los sistemas operativos que trabajan bajo el hardware anteriormente citado son:

- **Windows 7 Home Premium** en el ordenador Asus Eee PC 1215b.
- **Windows 10 Pro** en el ordenador Asus X550JX.
- **Android 5.0.1** en el Meizu MX4 Pro.
- **Raspbian OS Jessie** en la Raspberry Pi 2.

Las principales herramientas software que han intervenido en la realización de este proyecto son las siguientes:

- **Microsoft Office Word 2007** para redactar la presente memoria.
- **Microsoft Office Excel 2007 para** generar gráficos y tablas que se incluirán en esta memoria.
- **Microsoft Office PowerPoint 2007** para la presentación del proyecto.
- **Microsoft Office Project 2007** para elaborar la planificación del proyecto.
- **Adobe Dreamweaver CS6** para implementar y probar el sitio web.
- **Adobe Acrobat Reader DC 2015** para visualizar los informes generados.
- **FPDF 1.6** para crear documentos PDF desde PHP.
- **Highcharts 5.0.2** para presentar gráficas desde el sitio web.
- **Google Charts** para mostrar el termómetro y el higrómetro en el sitio web.
- **REM 1.2.2** para la ingeniería de requisitos.
- **Astah Community 6.9.0** para realizar los diagramas oportunos en las fases de análisis y diseño.
- **Editores de texto (nano y vim)** para implementar el código de los diferentes programas Python y scripts para Linux.
- **Navegador web Chrome** para probar el sitio web.
- **Apache 2.4.10** como servidor web.
- **MySQL 5.5.44** como sistema gestor de base de datos.
- **phpMyAdmin 4.2.12** para la gestión de la base de datos.
- **MJPG-Streamer** para el streaming de la Raspberry Pi Camera.
- **raspistill 1.3.2** para tomar imágenes de la Raspberry Pi Camera.
- **Aplicación de mensajería instantánea Telegram** para el envío de mensajes de alerta en tiempo real.

- PuTTY 0.67: este cliente SSH se ha empleado a lo largo del proyecto para acceder a la Raspberry sin necesidad de conectar a la misma el teclado, ratón y monitor. Desde mi propia computadora, a través de esta herramienta se puede acceder a la Raspberry en modo consola mediante SSH.

3.2. Planificación

Al principio de este bloque se habló de la metodología empleada y del plan de trabajo que se llevó a cabo para el desarrollo de este proyecto. Allí se recogieron los incrementos que iban componiendo el sistema y se comentó en qué consistían tales incrementos.

En este capítulo se identifican con mayor detalle las tareas que forman los incrementos y además se lleva a cabo una estimación temporal de dichas tareas, incluyendo sus fechas de inicio, de final y también las relaciones de precedencia que las vinculan y determinan el inicio de la siguiente tarea o incremento (Figura 3.1). Todo esto se representará de forma gráfica en un diagrama de Gantt que se incluye en la siguiente página (Figura 3.2).

Esta estimación se basa en la experiencia adquirida en la asignatura de la titulación Planificación y Gestión de Plataformas Informáticas y en la realización de otros proyectos previos.

Id	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**Incremento 1: fase de inicio**	**13 días**	**lun 15/02/16**	**mié 02/03/16**	
2	Viabilidad del proyecto	8 días	lun 15/02/16	mié 24/02/16	
3	Requisitos del sistema final	3 días	jue 25/02/16	lun 29/02/16	2
4	Boceto del sistema	2 días	mar 01/03/16	mié 02/03/16	3
5	**Incremento 2: GPIO y estructura del sitio web**	**23 días**	**jue 03/03/16**	**lun 04/04/16**	**1**
6	Estructura del sitio web	16 días	jue 03/03/16	jue 24/03/16	
7	Estudio de los GPIO	3 días	vie 25/03/16	mar 29/03/16	6
8	Pruebas con los GPIO	4 días	mié 30/03/16	lun 04/04/16	7
9	**Incremento 3: cámara**	**19 días**	**mar 05/04/16**	**vie 29/04/16**	**5**
10	Estudio de la Raspberry Pi Camera	2 días	mar 05/04/16	mié 06/04/16	
11	Captura de imágenes con raspistill	6 días	jue 07/04/16	jue 14/04/16	10
12	Streaming con MJPG-Streamer	7 días	vie 15/04/16	lun 25/04/16	11
13	Implementación de la galería de la web	4 días	mar 26/04/16	vie 29/04/16	12
14	**Incremento 4: sensores DHT-11 y MQ-135**	**50 días**	**lun 02/05/16**	**vie 08/07/16**	**9**
15	Estudio del sensor de temperatura (DHT-11)	3 días	lun 02/05/16	mié 04/05/16	
16	Implementación para el manejo del DHT-11	7 días	jue 05/05/16	vie 13/05/16	15
17	Estudio del sensor detector de gases (MQ-135)	2 días	lun 16/05/16	mar 17/05/16	16
18	Implementación para el manejo del MQ-135	5 días	mié 18/05/16	mar 24/05/16	17
19	Manejo del DHT-11 y del MQ-135 desde el sitio web	10 días	lun 27/06/16	vie 08/07/16	18
20	**Incremento 5: alertas Telegram**	**22 días**	**jue 15/09/16**	**vie 14/10/16**	**14**
21	Estudio de la viabilidad de Telegram en Raspberry Pi	4 días	jue 15/09/16	mar 20/09/16	
22	Instalación de Telegram en Raspberry Pi	1 día	lun 19/09/16	lun 19/09/16	
23	Modificación de códigos para enviar alertas Telegram	7 días	mié 21/09/16	jue 29/09/16	21
24	Envío de alertas por Telegram desde el sitio web	11 días	vie 30/09/16	vie 14/10/16	23
25	**Incremento 6: informes manuales**	**31 días**	**lun 17/10/16**	**lun 28/11/16**	**20**
26	Implementación de los informes online	15 días	lun 17/10/16	vie 04/11/16	
27	Estudio de la librería Highcharts	3 días	jue 20/10/16	lun 24/10/16	
28	Implementación de los informes PDF	10 días	lun 07/11/16	vie 18/11/16	26
29	Estudio de FPDF	5 días	lun 07/11/16	vie 11/11/16	
30	Implementación de un contenedor de informes	6 días	lun 21/11/16	lun 28/11/16	28
31	**Incremento 7: informes automáticos**	**17 días**	**lun 16/01/17**	**mar 07/02/17**	**25**
32	Estudio de métodos de automatización de tareas	2 días	lun 16/01/17	mar 17/01/17	
33	Selección de tareas a automatizar	2 días	mié 18/01/17	jue 19/01/17	32
34	Implementación de generación automática de informes PDF	10 días	vie 20/01/17	jue 02/02/17	33
35	Pruebas de generación automática de informes	3 días	vie 03/02/17	mar 07/02/17	34
36	**Incremeto 8: refinamiento del sitio web y memoria**	**50 días**	**lun 13/02/17**	**vie 21/04/17**	**31**
37	Refinamiento del sitio web	33 días	lun 13/02/17	mié 29/03/17	
38	Completar memoria del proyecto	40 días	lun 27/02/17	vie 21/04/17	

Figura 3.1: duración y precedencia de las actividades

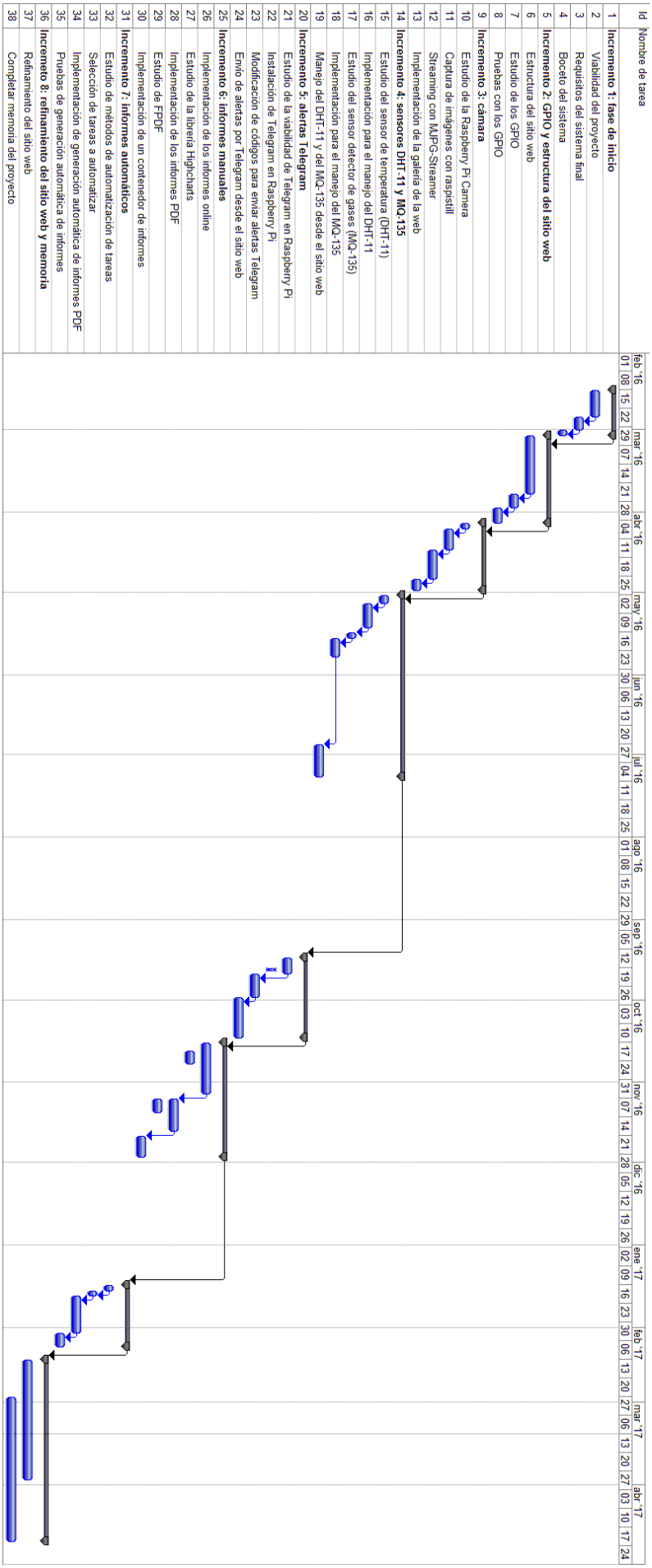


Figura 3.2: diagrama de Gantt

En las figuras 3.1 y 3.2 se recoge la planificación real resultante durante el desarrollo del proyecto, por supuesto esta planificación dista en algunos aspectos con la planificación inicial propuesta. Principalmente este proyecto sufre tres grandes variaciones o retrasos que son perfectamente reconocibles en el anterior diagrama de Gantt:

- **Convocatoria ordinaria del segundo cuatrimestre de la Universidad**: como puede verse en el diagrama de Gantt, este TFG se inició paralelamente al inicio del segundo cuatrimestre, en 2016, sin embargo a medida que avanzaba el cuatrimestre, el desarrollo del TFG se ralentizaba hasta que a finales del mes de mayo se hizo el primer parón en el proyecto para tener una dedicación exclusiva a los exámenes finales.

- **Motivos laborales referentes a la campaña de verano**: el segundo y mayor de los retrasos se sucede al finalizar el incremento 4. Se produce un parón de dos meses que se retoma a mediados de septiembre de 2016 con el inicio del incremento 5. Este parón se debe a un aumento de horas laborales que impiden la compaginación con el TFG que se venía dando en los meses anteriores donde la jornada laboral era menor.

- **Motivos laborales referentes a la campaña de Navidad**: de nuevo, un aumento de la jornada laboral imposibilita la compaginación del trabajo con el desarrollo del TFG y coincidiendo con la campaña de Navidad se produce el último parón que abarca el mes de diciembre de 2016 y los primeros días de enero de 2017.

Estos retrasos en la planificación no pudieron ser previstos de antemano, ya que los aumentos de mi jornada laboral no estaban planeados e inicialmente la preparación de los exámenes estaba diseñada para ir en paralelo a una menor carga de trabajo relacionada con el TFG pero sin realizar parones superiores a cuatro días.

Las jornadas de trabajo a lo largo del desarrollo de este proyecto han sido desarrolladas de lunes a viernes y siendo más o menos extensas (en número de horas) dependiendo de una situación laboral. Finalmente, en la dedicación de este TFG se estima haber invertido más de 400 horas que se distribuyen a lo largo de los incrementos de la siguiente forma que podemos ver en la Tabla 3.1:

Incremento 1	16 horas
Incremento 2	34 horas
Incremento 3	45 horas
Incremento 4	40 horas
Incremento 5	60 horas
Incremento 6	88 horas
Incremento 7	30 horas
Incremento 8	115 horas
Total	**428 horas**

Tabla 3.1: duración de los incrementos en horas

Para finalizar este capítulo se ha considerado oportuno hacer una aclaración referente a la elaboración de la presente memoria. A lo largo del desarrollo del proyecto se han ido documentando las principales acciones en un cuaderno de bitácora sin orden ni formato. En este cuaderno se recogían las fechas y aquellas porciones de

código importantes, secuencias de comandos, librerías necesarias, requisitos del sistema, partes del diseño, pruebas realizadas y aclaraciones en general que iban surgiendo en el día a día. Este cuaderno de bitácora ha servido de guía para la implementación de la memoria.

El orden, formato y primeros capítulos de la actual memoria se desarrollaron en paralelo a la implementación de los incrementos 1 y 2.

Los siguientes incrementos a menudo requerían una mayor interactividad: se estudiaba la viabilidad de cada incremento, se desarrollaba y se realizaban constantes pruebas. Esta interactividad del proyecto hizo que el cuaderno de bitácora y las anotaciones "sin formato" tomaran mayor protagonismo en detrimento de la formateada memoria, por tanto, continué sirviéndome del cuaderno de bitácora y retrasando la finalización de la memoria hasta el incremento 8.

3.3. Estudio económico

El principal objetivo de este capítulo es demostrar que se ha conseguido desarrollar un sistema de seguridad y de monitorización casero y de bajo coste. Para ello se incidirá más en el coste de los bienes materiales que en el de los recursos humanos ya que en todo momento se pretende que el sistema pudiera ser implementado por cualquier persona con una necesidad similar y ganas de aprender.

En la Tabla 3.2 se recoge el precio de todos los componentes que han sido adquiridos y que juntos forman el sistema final:

Componente	Precio
Raspberry Pi 2 Modelo B	32.87 €
Carcasa oficial para Raspberry Pi 2	7.79 €
Fuente de alimentación para Raspberry Pi 2	6.21 €
Edimax EW-7811UN	7.79 €
Raspberry Pi Camera	18.99 €
Carcasa Raspberry Pi Camera	7.14 €
SanDisk Ultra microSDHC 32GB Clase 10	9.99 €
Sensor temperatura y humedad (DHT-11)	0.91 €
Detector de humo (MQ-135)	1.30 €
Micro Servo 9g SG90	1.38 €
Placa protoboard	2.56 €
Cables variados (120 unidades, 20cm)	2.12 €

Diodos led y resistencias variadas	3.05 €
Buzzer (YL-44)	0.71 €
Total	**102,81 €**

Tabla 3.2: precio de cada componente

Como vemos, por unos cien euros podemos tener un sistema de seguridad y monitorización casero y de bajo coste, que además, es fácilmente escalable ya que el sistema puede crecer añadiéndole los sensores que pudieran ser requeridos dependiendo de las necesidades.

Este sistema inicialmente no está pensado para ser puesto en producción y obtener beneficios económicos, más bien surge como respuesta casera a una necesidad de simplificar una tarea y tener información sobre un lugar remoto en tiempo real. Pudiera servir esta memoria como una detallada guía de cómo llevar a cabo un proyecto de características similares.

En cuanto a los recursos software empleados en el desarrollo del proyecto, no se ha invertido dinero en ellos ya que o bien se ha optado por aplicaciones de libre distribución, o bien se han aprovechado licencias que ya se poseían anteriormente, con lo que en este apartado no añadimos esos costes.

De cara a los costes energéticos de la Raspberry Pi 2 B, diremos que este dispositivo también es de bajo coste energéticamente hablando ya que presenta un consumo medio de unos 2.5W conectada a través del wifi y con una carga media de trabajo. Por lo tanto, resulta ser un dispositivo muy adecuado para hacer las funciones de un servidor que requiere estar encendido constantemente a lo largo del tiempo (para hacernos una idea, un ordenador sin monitor gastaría más de 50W).

Teniendo en cuenta el anterior consumo medio de nuestra Raspberry de 2.5W, no calcularemos el coste energético durante la realización del proyecto, pero sí el coste energético diario, mensual y anual ocasionado por nuestra Raspberry cuando ya esté desempeñando su labor. De este modo probaremos el mínimo consumo de este dispositivo. Para desempeñar los cálculos recogemos los datos de partida en la Tabla 3.3:

Descripción	Valor
Consumo medio Raspberry Pi 2 Modelo B	2.5 W (0.0025 kW)
Precio kWh medio en marzo de 2017	0.1187 €/kWh

Tabla 3.3: consumo de la Raspberry y precio del kWh

$$(2.5\ W) \times (24\ h/día) = 60\ Wh/día\ = 0.06\ kWh/día$$

$$(0.1187\ €/kWh)\ \times\ (0.06\ kWh/día)\ = 0.007122\ €/día$$

$$(0.1187\ €/kWh)\ \times\ (0.06\ kWh/día)\ \times\ (30\ días/mes)\ = 0.21366\ €/mes$$

$$(0.1187\ €/kWh)\ \times\ (0.06\ kWh/día)\ \times\ (365\ días/año)\ = 2.59953\ €/año$$

Tras realizar las correspondientes operaciones, queda demostrado el bajo consumo energético de la Raspberry Pi, ya que estando encendida las 24 horas del día sólo tendrá un coste de 20 céntimos de euro al mes. Si en lugar de la Raspberry se hubiese empleado una torre de sobremesa en la que alojar el servidor web (con un consumo aproximado de 50W), los 20 céntimos mensuales ascenderían a 4.27 €: el consumo se multiplicaría por 20.

3.4. Plan de gestión de riesgos

El plan de gestión de riesgos se ha desarrollado para ayudarnos a identificar, analizar y gestionar los riesgos que se puedan producir durante el desarrollo de este proyecto. Se va a emplear una estrategia proactiva, integrada y sistemática, de tal forma que se identifiquen los riesgos potenciales teniendo en cuenta tanto su probabilidad de ocurrencia, como su posible impacto. Se establecerán políticas sistemáticas y disciplinadas de prioridad para los diferentes riesgos. Junto con los planes de acción correspondientes, se controlará cualquier situación posible en caso de ocurrencia.

3.4.1. Identificación de los riesgos

Para la identificación de los riesgos se ha empleado la técnica del *brainstorming* o lluvia de ideas en la cual, teniendo en cuenta los riesgos potenciales ocurridos en anteriores proyectos, se han ido apuntando uno a uno en una lista de riesgos.

Una vez que construida una lista inicial, se han estudiado los riesgos apuntados en ella y se han ido seleccionando aquellos que sí son aplicables y se han buscado nuevos riesgos potenciales que pudieran surgir en este proyecto. El resultado de esta segunda etapa en la que hemos concretado y profundizado en los riesgos potenciales es la Tabla 3.4, donde quedan identificados:

Identificador	Descripción	Tipo de riesgo
R01	Interpretación errónea de los requisitos	De proyecto
R02	Calidad inadecuada	De producto
R03	Planificación optimista	De proyecto
R04	Diseño inadecuado	De producto
R05	Falta de experiencia del alumno	De proyecto
R06	Deterioro de la placa Raspberry Pi	Técnico
R07	Problemas de comunicación	De proyecto
R08	Baja productividad	De proyecto
R09	Realización de pruebas inadecuada	De producto
R10	Falta de seguimiento	De proceso

Tabla 3.4: riesgos identificados

3.4.2. Análisis de riesgos

Teniendo en cuenta la lista de riesgos identificados de la Tabla 3.4, definiremos un nivel de probabilidad de aparición de los riesgos, también el impacto que supondrá en el alcance del objetivo y el coste ocasionado (en nuestro caso, al tratarse de un proyecto universitario más que de costes económicos nos referiremos a costes temporales).

En la Tabla 3.5 podemos observar para cada riesgo, su coste, probabilidad de ocurrencia y el impacto que tendría en caso de que sucediese.

Identificador	Descripción	Probabilidad	Impacto	Coste
R01	Interpretación errónea de los requisitos	Muy alta	Muy alto	Elevado
R02	Calidad inadecuada	Alta	Muy alto	Elevado
R03	Planificación optimista	Media	Alto	Medio
R04	Diseño inadecuado	Baja	Alto	Medio
R05	Falta de experiencia del alumno	Baja	Bajo	Corto
R06	Deterioro de la placa Raspberry Pi	Media	Muy Alto	Medio
R07	Problemas de comunicación	Baja	Medio	Corto
R08	Baja productividad	Baja	Bajo	Corto
R09	Realización de pruebas inadecuada	Media	Alto	Corto
R10	Falta de seguimiento	Alta	Alto	Medio

Tabla 3.5: detalles de los riesgos identificados

Antes de continuar, en la Tabla 3.6 se estima y se clasifica la probabilidad de aparición de cada riesgo:

Probabilidad	Porcentaje	Descripción
Muy Alta	> 80%	Es muy probable que ocurra el evento
Alta	60% - 80%	Es probable que ocurra el evento
Media	30% - 60%	El evento puede ocurrir
Baja	< 30%	Es improbable que ocurra el evento, pero puede ocurrir

Tabla 3.6: estimación de la probabilidad

Estimamos también los posibles impactos de cada riesgo en la clasificación de la Tabla 3.7:

Impacto	Descripción
Muy Alto	Objetivos fundamentales del proyecto con alto grado de impacto o no cumplimiento
Alto	Los objetivos críticos del proyecto se ven amenazados
Medio	Algunos objetivos del proyecto pueden verse afectados
Bajo	Los objetivos del proyecto no se comprometen. Fácilmente remediable

Tabla 3.7: estimación del impacto

Una vez estimada la probabilidad de aparición y el impacto que ocasionarían los riesgos, se elabora la matriz probabilidad-impacto de la Tabla 3.8 que ayuda a identificar y a priorizar los riesgos en función del impacto que generarían en caso de manifestarse y también atendiendo a cuánto es de probable que se manifiesten.

		Probabilidad			
		Muy Alta	**Alta**	**Media**	**Baja**
Impacto	**Muy Alto**	R01	R02	R06	
	Alto		R10	R03, R09	R04
	Medio				R07
	Bajo				R05, R08

Tabla 3.8: matriz probabilidad-impacto

Teniendo en cuenta la matriz de probabilidad-impacto, se realiza la clasificación de los riesgos atendiendo a su prioridad, lo vemos en la Tabla 3.9:

Identificador	Descripción	Prioridad
R01	Interpretación errónea de los requisitos	Muy Alta
R02	Calidad inadecuada	Muy Alta
R03	Planificación optimista	Media
R04	Diseño inadecuado	Media
R05	Falta de experiencia del alumno	Baja

R06	Deterioro de la placa Raspberry Pi	Alta
R07	Problemas de comunicación	Baja
R08	Baja productividad	Baja
R09	Realización de pruebas inadecuada	Media
R10	Falta de seguimiento	Alta

Tabla 3.9: priorización de riesgos

Según el nivel de prioridad del riesgo, seguiremos un protocolo de actuación resumido en la Tabla 3.10:

Prioridad	Criterio	Acción
Muy Alta	Riesgo crítico	Acción inmediata
Alta	Riesgo significativo y probable	Iniciar procedimientos
Media	Riesgo significativo y menos probable	Realizar una revisión
Baja	Riesgo poco probable	Volver a valorar

Tabla 3.10: acciones a realizar en función del riesgo

3.4.3. Planificación de respuesta a los riesgos

Para los riesgos que tienen una prioridad “Muy Alta” o “Alta”, se han desarrollado planes de respuesta específicos que están recogidos en la Tabla 3.11:

Identificador	Plan de mitigación	Plan de contingencia
R01	Reuniones con el tutor para confirmar la aceptación de los requisitos	Revisar el código y la documentación de los requisitos afectados
R02	Establecer un plan de calidad que se acoja al estándar ISO 9000 y 9003	Revisiones frecuentes con el tutor del proyecto
R06	Manipular la Raspberry Pi habiéndonos aislado eléctricamente con anterioridad o haciendo uso de guantes	Es altamente probable que si el riesgo aparece la Raspberry Pi se haya cortocircuitado y sea inservible
R10	Revisar continuamente y de forma periódica el estado del proyecto para detectar sus posibles anomalías	Reuniones con el tutor y realización de nuevas planificaciones

Tabla 3.11: planes de respuesta

Aclaramos que el plan de mitigación trata de reducir la probabilidad de ocurrencia del riesgo o el impacto que puede causar: el objetivo de la mitigación de riesgos es reducir la exposición al riesgo. Sin embargo, el plan de contingencia se pone en práctica sólo si ya hay indicios de ocurrencia del riesgo.

3.4.4. Seguimiento y control de riesgos

El seguimiento adecuado y el control efectivo de los riegos permiten detectarlos de forma temprana y ayudan a la toma de decisiones efectivas, por lo que adoptaremos las siguientes medidas:

- Revisar el estado de los riesgos entre el tutor y el alumno en reuniones periódicas.
- Revisar el estado de los riesgos en los hitos principales del proyecto.

Tras cada hito se revisará y actualizará el documento, manteniéndose activo a lo largo de todo el proyecto.

Capítulo 4: Requisitos del sistema

En este capítulo se describe qué debe hacer el sistema y cómo debe hacerlo. Para ello emplearemos los Requisitos Funcionales y los Requisitos No Funcionales.

4.1. Requisitos Funcionales

A continuación se muestran los Requisitos Funcionales (RF) que recogen la funcionalidad del sistema, es decir, lo que este deberá hacer:

Identificador RF01

Descripción	El sistema permitirá identificarse al usuario.

Identificador RF02

Descripción	El sistema permitirá al usuario monitorizar la temperatura del lugar.

Identificador RF03

Descripción	El sistema permitirá al usuario monitorizar la humedad del lugar.

Identificador RF04

Descripción	El sistema permitirá al usuario comprobar el estado de los componentes que lo forman.

Identificador RF05

Descripción	El sistema permitirá al usuario monitorizar el lugar ofreciendo imágenes en tiempo real.

Identificador RF06

Descripción	El sistema permitirá al usuario tomar fotografías del lugar.

Identificador RF07

Descripción	El sistema permitirá al usuario ajustar la calidad de las fotografías tomadas.

Identificador RF08

Descripción	El sistema permitirá al usuario grabar un vídeo del lugar.

Identificador RF09

Descripción	El sistema permitirá al usuario ajustar la duración del vídeo grabado.

Identificador RF10

Descripción	El sistema permitirá al usuario mover la cámara.

Identificador RF11

Descripción	El sistema permitirá al usuario descargarse el contenido multimedia (imágenes y vídeos) creado.

Identificador RF12

Descripción	El sistema permitirá al usuario generar informes instantáneos de temperatura y humedad.

Identificador RF13

Descripción	El sistema permitirá al usuario configurar la creación de informes periódicos automáticamente.

Identificador RF14

Descripción	El sistema permitirá al usuario descargarse los informes creados.

Identificador RF15

Descripción	El sistema permitirá al usuario enviar alertas configurables.

Identificador RF16

Descripción	El sistema permitirá al usuario controlar los actuadores de forma manual.

Identificador RF17

Descripción	El sistema tendrá un mecanismo de actuación en caso de que la temperatura de la sala aumente por encima de un valor fijado.

Identificador RF18

Descripción	El sistema tendrá un mecanismo de actuación si se detecta humo en la sala.

4.2. Requisitos No Funcionales

Los Requisitos No Funcionales (RNF) describen aquellos aspectos del comportamiento de un sistema, capturando las propiedades y restricciones bajo las que este debe operar. Se han identificado los siguientes y están ordenados según su tipología.

4.2.1. Arquitectura

Identificador RNF01

Descripción	El sistema deberá ser accedido por el usuario vía web.

Identificador RNF02

Descripción	El sistema contará con un sitio web que deberá ser soportado por los principales navegadores.

Identificador RNF03

Descripción	El sistema contendrá datos que deberán estar almacenados en un Sistema Gestor de Bases de Datos relacional con un lenguaje de consultas SQL.

4.2.2. Seguridad

Identificador RNF04

Descripción	El sistema sólo permitirá el acceso a la aplicación web a los usuarios registrados que se identifiquen correctamente ante él (usuario + contraseña).

4.2.3. Persistencia

Identificador RNF05

Descripción	El sistema deberá almacenar los datos de manera persistente en la base de datos.

4.2.4. Estándares

Identificador RNF06

Descripción	El sistema será desarrollado siempre que sea posible empleando software cuya licencia de uso sea lo menos restrictiva posible. Preferentemente de código abierto.

Identificador RNF07

Descripción	El sistema proporcionará un sitio web que deberá cumplir con los estándares marcados por el WWW Consortuim (HTML 4.0 o superior, CSS 2.0 o superior).

Identificador RNF08

Descripción	El sistema empleará un lenguaje de consulta SQL para la comunicación con la base de datos.

Identificador RNF09

Descripción	El sistema empleará el lenguaje PHP 5 para la programación del servidor web.

Identificador RNF10

Descripción	El sistema empleará el lenguaje de programación Python 2.7 para la comunicación con sensores y actuadores.

Identificador RNF11

Descripción	El sistema deberá garantizar que su módulo de alertas a través de una aplicación de mensajería instantánea funcione correctamente tanto en dispositivos móviles Android como iOS.

4.2.5. Escalabilidad

Identificador RNF12

Descripción	El sistema deberá ser extensible por si se precisan nuevas funcionalidades en un futuro.

4.2.6. Usabilidad

Identificador RNF13

Descripción	El sistema proporcionará un sitio web que deberá mostrar una estructura clara y ordenada, empleando menús y submenús para organizar el contenido.

Identificador RNF14

Descripción	El sistema deberá ser fácil de usar para una persona con conocimientos elementales en el manejo de dispositivos informáticos.

Identificador RNF15

Descripción	El sistema deberá proporcionar una interfaz gráfica austera que no se muestre sobrecargada para una rápida localización de las funcionalidades del sitio web, sin distracciones.

Capítulo 5: Modelado de Casos de Uso

Las tareas referentes a esta sección consisten en identificar y describir los actores que interactuarán con el sistema, proporcionar un diagrama de Casos de Uso y especificar cada uno de los Casos de Uso reflejados en el anterior diagrama, tanto de forma escrita como haciendo uso de los Diagramas de Secuencia del Sistema (DSS).

5.1. Descripción general de los actores

Se identifican un único actor que interactúa con el sistema. A continuación se describe:

- **Usuario:** es el actor principal. Interactuando con el sistema debe cumplir con las mismas tareas de monitorización que llevaba a cabo antes de informatizarse la actividad, es decir, en nuestro caso deberá controlar la temperatura de la sala de servidores y actuar si se detecta alguna incidencia. Además, con la implementación del sistema, el usuario podrá desempeñar nuevas funciones como generar informes, enviar alertas desde la aplicación, acceso a streaming de la sala de servidores, tomar fotografías, control manual de los actuadores…

5.2. Diagrama de Casos de Uso

En la Figura 5.1 se encuentra el Diagrama de Casos de Uso de nuestro sistema:

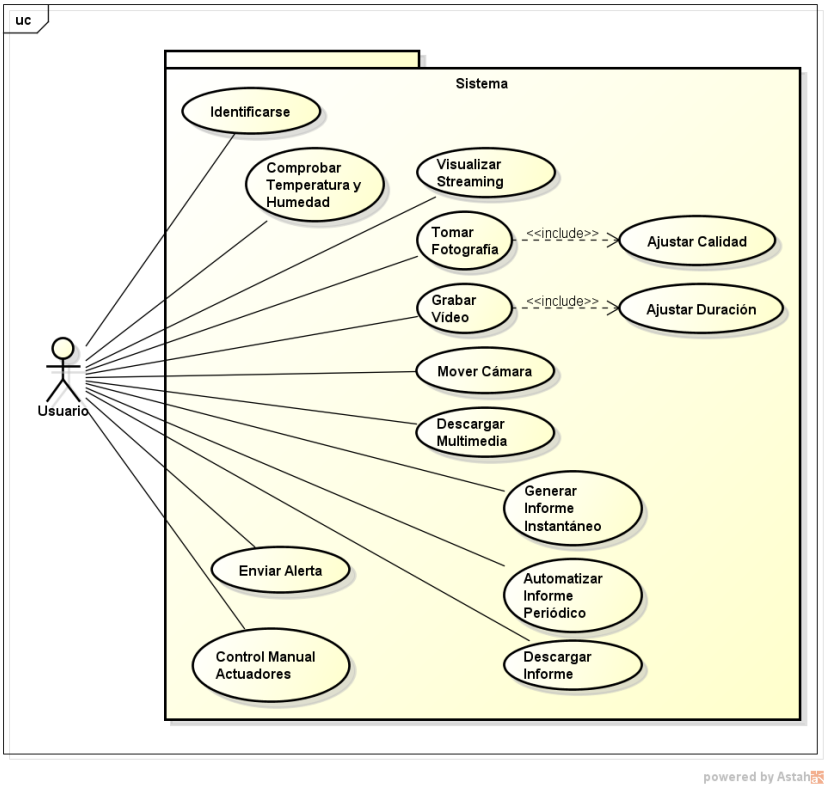


Figura 5.1: Diagrama de Casos de Uso

5.3. Especificación de Casos de Uso

En esta sección se detallará cada uno de los Casos de Uso del sistema.

UC-01	Identificarse	
Descripción	El actor Usuario desea identificarse frente al sistema para realizar alguna acción	
Actores implicados	Usuario	
Precondición	Ninguna	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede al área destinada a la identificación de usuarios
	2	El sistema solicita al actor que introduzca su nombre de usuario y la contraseña
	3	El actor introduce sus credenciales y solicita el acceso
	4	El sistema verifica los credenciales, da acceso al usuario a la aplicación web y el caso de uso finaliza
Postcondición	El actor Usuario se ha identificado frente al sistema	
Excepciones	**Paso**	**Acción**
	4	Si los datos introducidos no son correctos, el sistema se lo comunica al actor y el caso de uso queda sin efecto.

Tabla 5.1: Caso de Uso "Identificarse"

UC-02	Comprobar Temperatura y Humedad	
Descripción	El actor Usuario desea comprobar la temperatura y la humedad de la sala de servidores	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor solicita ver los valores de temperatura y humedad
	2	El sistema muestra los valores de temperatura y humedad y el caso de uso finaliza

Postcondición	El actor Usuario ha comprobado los valores de temperatura y humedad actuales referentes a la sala de servidores
Excepciones	Ninguna

Tabla 5.2: Caso de Uso "Comprobar Temperatura y Humedad"

UC-03	Visualizar Streaming	
Descripción	El actor Usuario desea visualizar el interior de la sala de servidores desde la aplicación web en tiempo real	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la sección de la cámara
	2	El sistema muestra las diferentes opciones
	3	El usuario selecciona el inicio del streaming para visualizar el interior del habitáculo
	4	El sistema muestra la imagen en tiempo real y el caso de uso finaliza
Postcondición	El actor Usuario mediante streaming visualiza el interior de la sala de servidores desde la aplicación web en tiempo real	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección relativa a la cámara, a continuación el caso de uso queda sin efecto

Tabla 5.3: Caso de Uso "Visualizar Streaming"

UC-04	Tomar Fotografía	
Descripción	El actor Usuario desea sacar una fotografía del interior de la sala	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**

	1	El caso de uso comienza cuando el actor accede a la sección de la cámara
	2	El sistema muestra las diferentes opciones
	<<include>>	Caso de Uso “Ajustar Calidad”
	3	El usuario selecciona la opción de tomar fotografía
	4	El sistema saca una fotografía, la guarda y el caso de uso finaliza
Postcondición	El actor Usuario saca una fotografía (de una calidad ajustada) del interior de la sala a la cual podrá acceder	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección relativa a la cámara, a continuación el caso de uso queda sin efecto

Tabla 5.4: Caso de Uso “Tomar Fotografía”

UC-05	Ajustar Calidad	
Descripción	El actor Usuario debe seleccionar la calidad con la que se sacará la fotografía	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema y se encuentra en la sección de la cámara	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor selecciona el porcentaje de calidad con el que le va a sacar la fotografía
	2	El sistema ajusta la calidad de la imagen
Postcondición	El actor Usuario ajusta la calidad con la que se sacará la fotografía	
Excepciones	Ninguna	

Tabla 5.5: Caso de Uso “Ajustar Calidad”

UC-06	Grabar Vídeo
Descripción	El actor Usuario desea grabar un vídeo del interior de la sala
Actores implicados	Usuario

Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la sección de la cámara
	2	El sistema muestra las diferentes opciones
	<<include>>	Caso de Uso “Ajustar Duración”
	3	El usuario selecciona la opción de grabar un vídeo
	4	El sistema graba un vídeo, lo guarda y el caso de uso finaliza
Postcondición	El actor Usuario graba un vídeo (de una duración ajustada) del interior de la sala al cual podrá acceder	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección relativa a la cámara, a continuación el caso de uso queda sin efecto

Tabla 5.6: Caso de Uso “Capturar Vídeo”

UC-07	Ajustar Duración	
Descripción	El actor Usuario debe seleccionar la duración del vídeo a grabar	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema y se encuentra en la sección de la cámara	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor selecciona la duración del vídeo que se va a grabar
	2	El sistema ajusta la duración del vídeo
Postcondición	El actor Usuario ajusta el la duración del vídeo que se va a grabar	
Excepciones	Ninguna	

Tabla 5.7: Caso de Uso “Ajustar Duración”

UC-08	Mover Cámara

Descripción	El actor Usuario desea mover la cámara	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la sección de la cámara
	2	El sistema muestra las diferentes opciones
	3	El usuario selecciona el movimiento de la cámara para que esta cambie de posición y enfoque en otra dirección
	4	El sistema mueve la cámara y el caso de uso finaliza
Postcondición	El actor Usuario cambia la posición de la cámara desde la aplicación web	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección relativa a la cámara, a continuación el caso de uso queda sin efecto

Tabla 5.8: Caso de Uso "Mover Cámara"

UC-09	Descargar Multimedia	
Descripción	El actor Usuario desea descargarse contenido multimedia (imágenes o vídeos) al dispositivo desde el cual accede al sistema	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la sección de la galería de contenido multimedia
	2	El sistema muestra las imágenes y vídeos anteriormente creados
	3	El usuario selecciona el archivo que desea descargar
	4	El sistema descarga el archivo y el caso de uso finaliza
Postcondición	El actor Usuario se descarga en su dispositivo el archivo (imagen o vídeo) deseado	
Excepciones	**Paso**	**Acción**

	2	Si el actor se sale de la sección relativa a la galería de contenido multimedia, a continuación el caso de uso queda sin efecto

Tabla 5.9: Caso de Uso "Descargar Multimedia"

UC-10	Generar Informe Instantáneo	
Descripción	El actor Usuario desea obtener al instante un informe referente a la temperatura y humedad de la sala de servidores, acorde a unas fechas establecidas	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la sección destinada a generar informes relativos a la temperatura y humedad de la sala
	2	El sistema solicita al usuario que configure las fechas que se tendrán en cuenta para generar el informe
	3	El actor introduce las fechas y solicita que se generen el informe
	4	El sistema muestra el informe que ha generado de acuerdo a las fechas introducidas por el usuario y el caso de uso finaliza
Postcondición	El actor Usuario accede en ese mismo momento a un informe sobre la temperatura y humedad acorde a la configuración de las fechas que él mismo ha configurado	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección relativa a generar informes, a continuación el caso de uso queda sin efecto

Tabla 5.10: Caso de Uso "Generar Informe Instantáneo"

UC-11	Automatizar Informe Periódico	
Descripción	El actor Usuario desea automatizar la generación de informes periódicos de temperatura y humedad	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**

	1	El caso de uso comienza cuando el actor accede a la sección destinada a generar informes relativos a la temperatura y humedad de la sala
	2	El sistema solicita al usuario que configure la periodicidad con la que se generarán los informes
	3	El actor introduce la periodicidad de generación de informes deseada
	4	El sistema automatiza la generación de informes de forma periódica de acuerdo a los datos introducidos por el usuario, muestra la periodicidad escogida y el caso de uso finaliza
Postcondición	El actor Usuario configura la generación automática de informes de temperatura y humedad	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección relativa a generar informes, a continuación el caso de uso queda sin efecto

Tabla 5.11: Caso de Uso "Automatizar Informe Periódico"

UC-12	Descargar Informe	
Descripción	El actor Usuario desea descargarse un informe al dispositivo desde el cual accede al sistema	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede al repositorio de informes
	2	El sistema muestra los informes anteriormente creados
	3	El usuario selecciona el informe que desea descargar
	4	El sistema descarga el informe y el caso de uso finaliza
Postcondición	El actor Usuario se descarga en su dispositivo el informe deseado	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección relativa a la galería de contenido multimedia, a continuación el caso de uso queda sin efecto

Tabla 5.12: Caso de Uso "Descargar Informe"

UC-13	Enviar Alerta	
Descripción	El actor Usuario desea enviar una alerta	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la sección destinada a las alertas
	2	El sistema muestra un formulario donde configurar la alerta
	3	El actor configura la alerta
	4	El sistema envía la alerta que ha generado y el caso de uso finaliza
Postcondición	El actor Usuario envía una alerta que él mismo ha configurado a un receptor	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección de alertas, a continuación el caso de uso queda sin efecto

Tabla 5.13: Caso de Uso "Enviar Alerta"

UC-14	Control Manual Actuadores	
Descripción	El actor Usuario desea realizar una operación con algún actuador manualmente	
Actores implicados	Usuario	
Precondición	El actor Usuario ya está identificado en el sistema	
Secuencia normal	**Paso**	**Acción**
	1	El caso de uso comienza cuando el actor accede a la sección destinada al manejo manual de los actuadores
	2	El sistema muestra los actuadores disponibles y las operaciones que puede realizar cada uno de ellos
	3	El actor selecciona una operación
	4	El sistema ejecuta la operación del actuador y el caso de uso finaliza

Postcondición	El actor Usuario controla manualmente algún actuador realizando la operación deseada	
Excepciones	**Paso**	**Acción**
	2	Si el actor se sale de la sección de control de los actuadores, a continuación el caso de uso queda sin efecto

Tabla 5.14: Caso de Uso “Control Manual Actuadores”

5.4. Diagramas de Secuencia del Sistema

Los Diagramas de Secuencia del Sistema (DSS), según Craig Larman “*son artefactos creados de manera rápida que muestran los eventos de entrada y salida relacionados con el sistema que se está estudiando*”. La introducción de estos DSS en la fase de análisis convierten al sistema en una caja negra de la cual no sabemos cómo implementa las operaciones pero sí sabemos qué hace, lo cual nos permite conocer el comportamiento del sistema y entender mejor su funcionalidad.

En esta sección se va a desarrollar un DSS para cada una de las especificaciones de Casos de Uso descritas en la sección anterior.

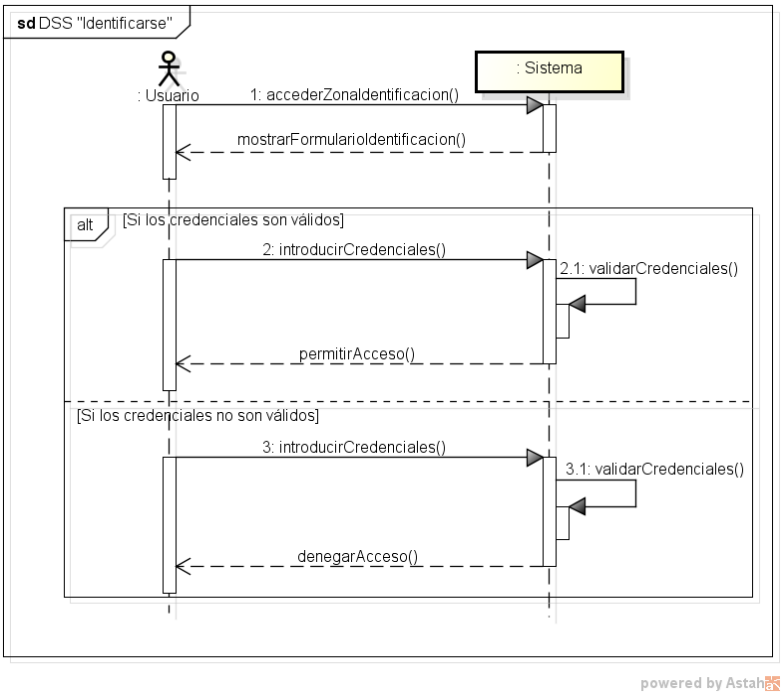


Figura 5.2: DSS "Identificarse"

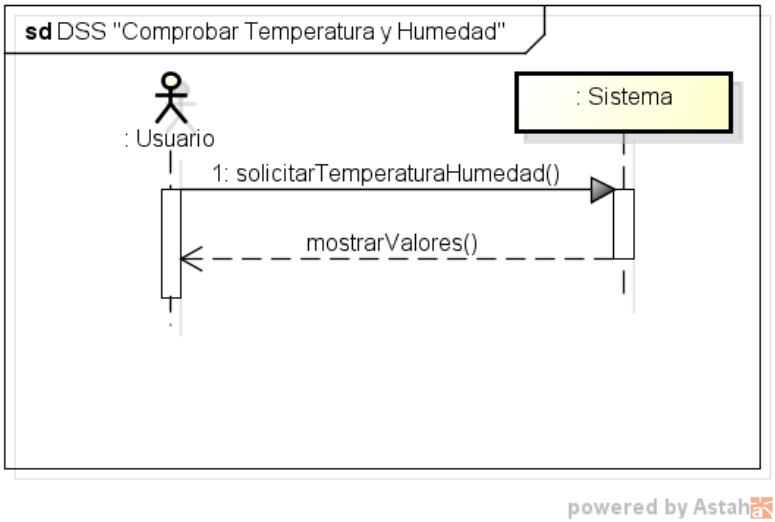


Figura 5.3: DSS "Comprobar Temperatura y Humedad"

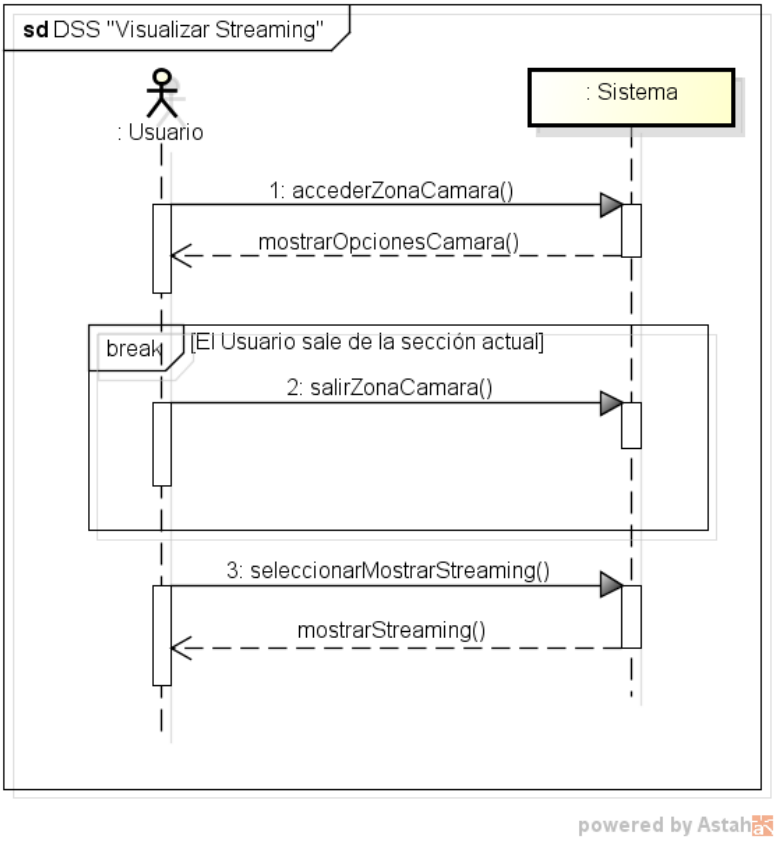


Figura 5.4: DSS "Visualizar Streaming"

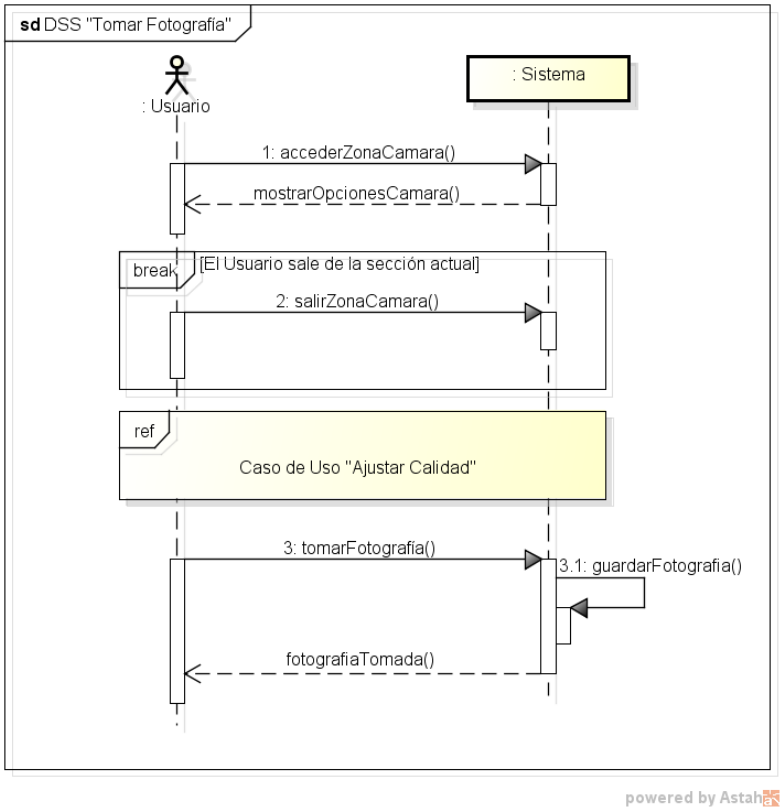


Figura 5.5: DSS "Tomar Fotografía"

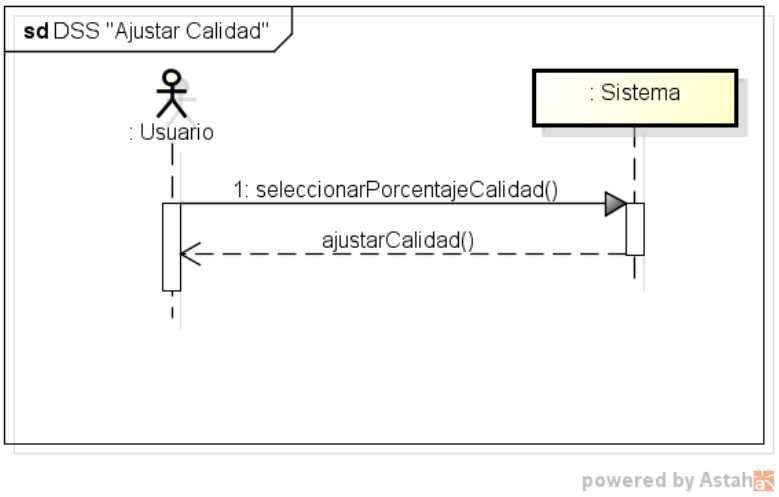


Figura 5.6: DSS "Ajustar Calidad"

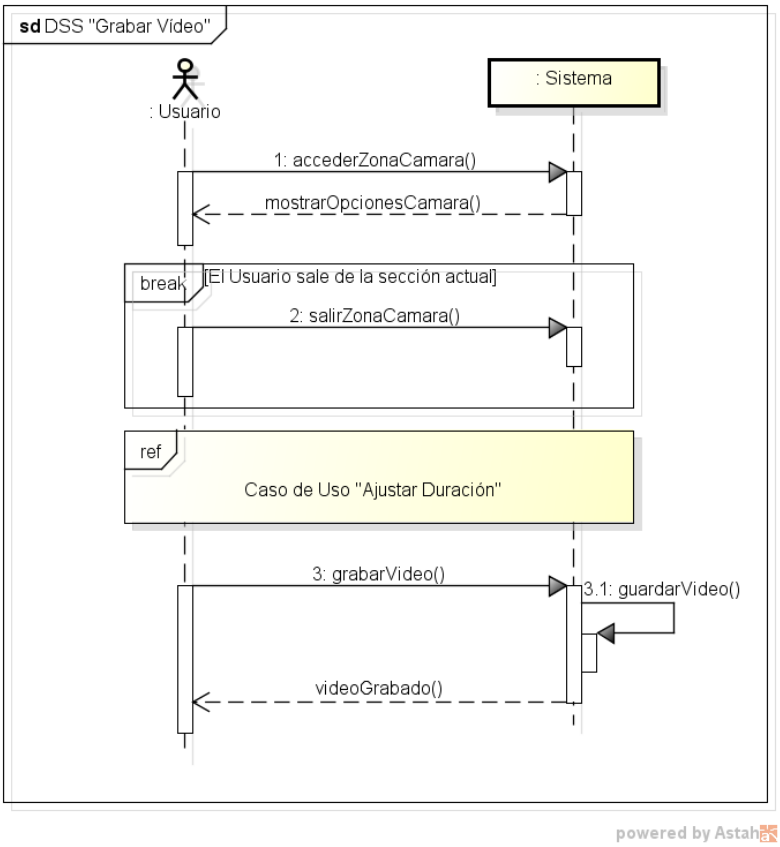


Figura 5.7: DSS "Grabar Vídeo"

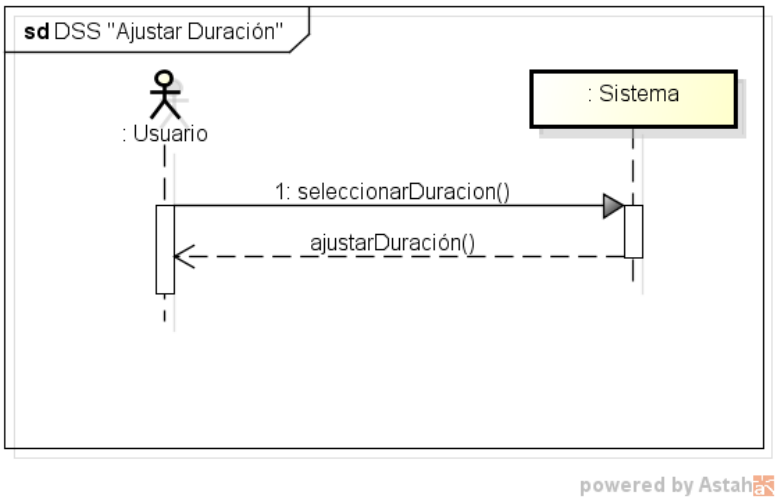


Figura 5.8: DSS "Ajustar Duración"

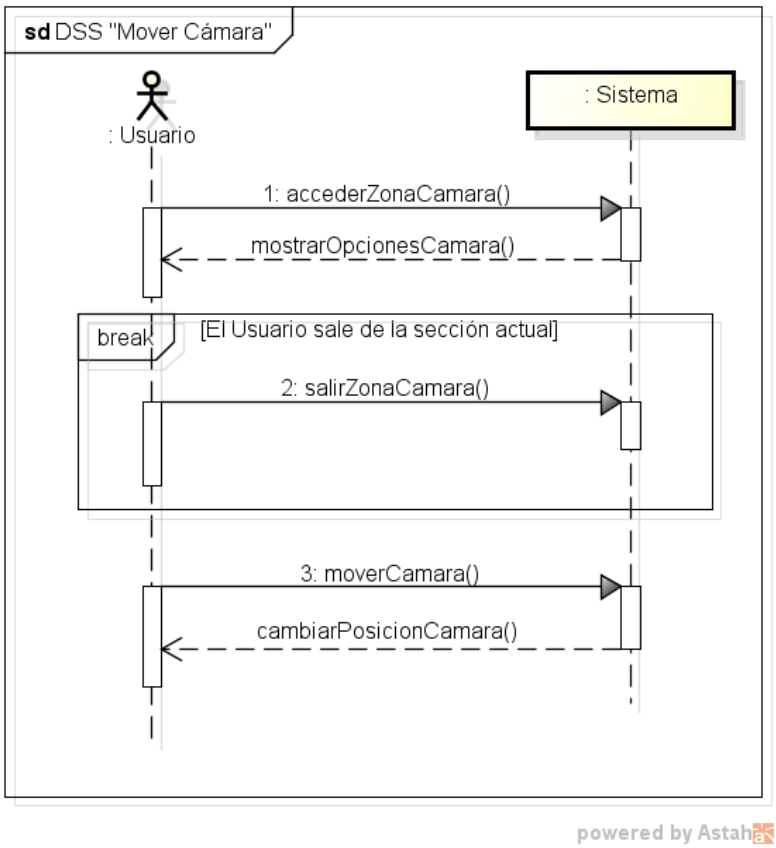


Figura 5.9: DSS "Mover Cámara"

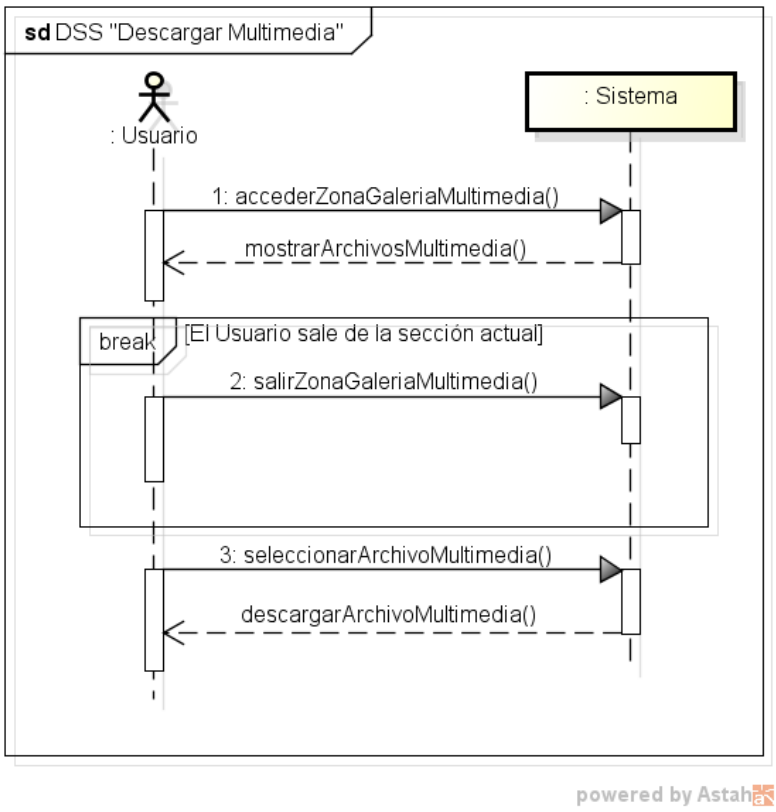


Figura 5.10: DSS "Descargar Multimedia"

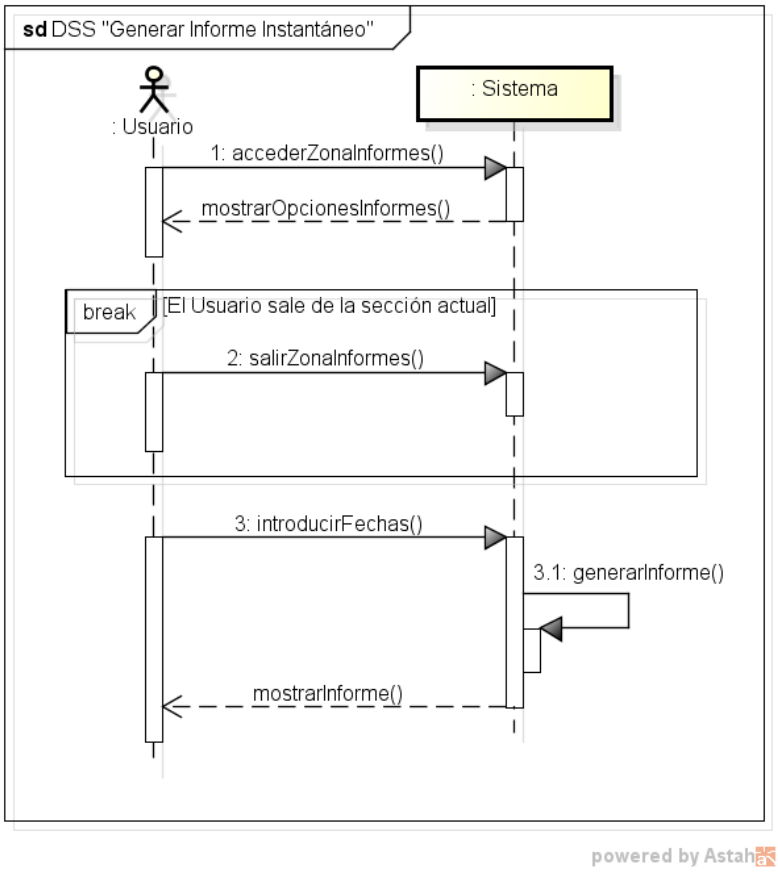


Figura 5.11: DSS "Generar Informe Instantáneo"

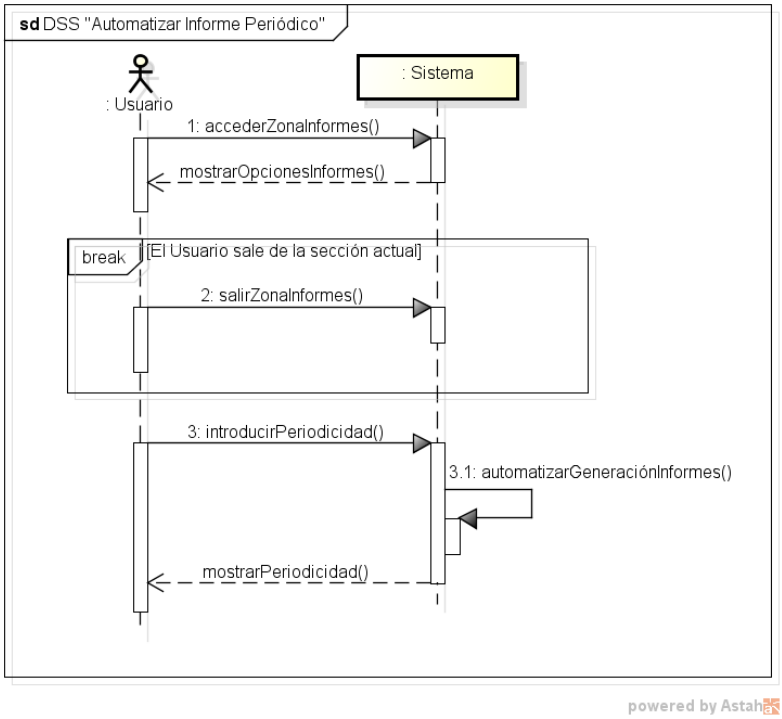


Figura 5.12: DSS "Automatizar Informe Periódico"

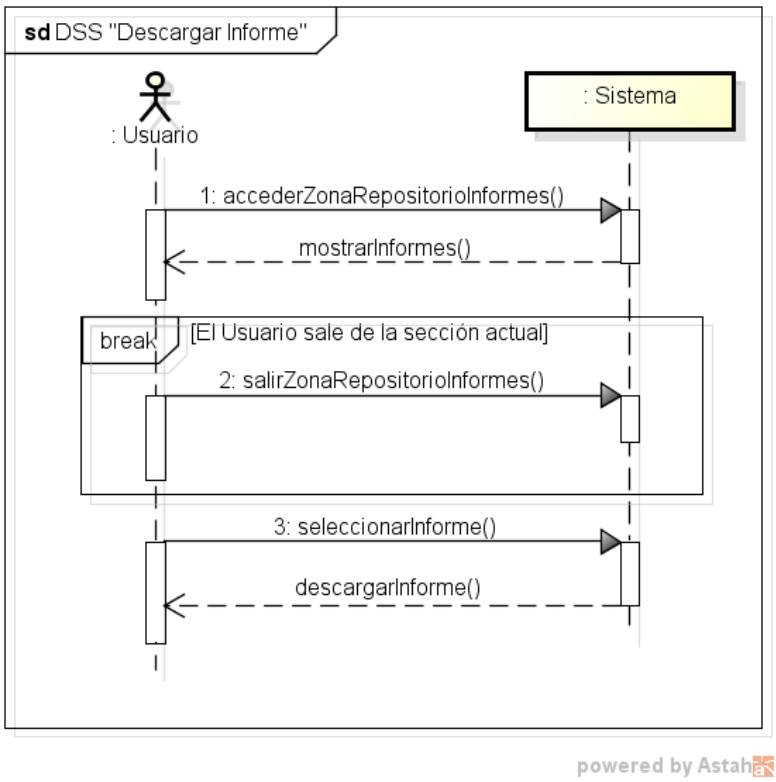


Figura 5.13: DSS "Descargar Informe"

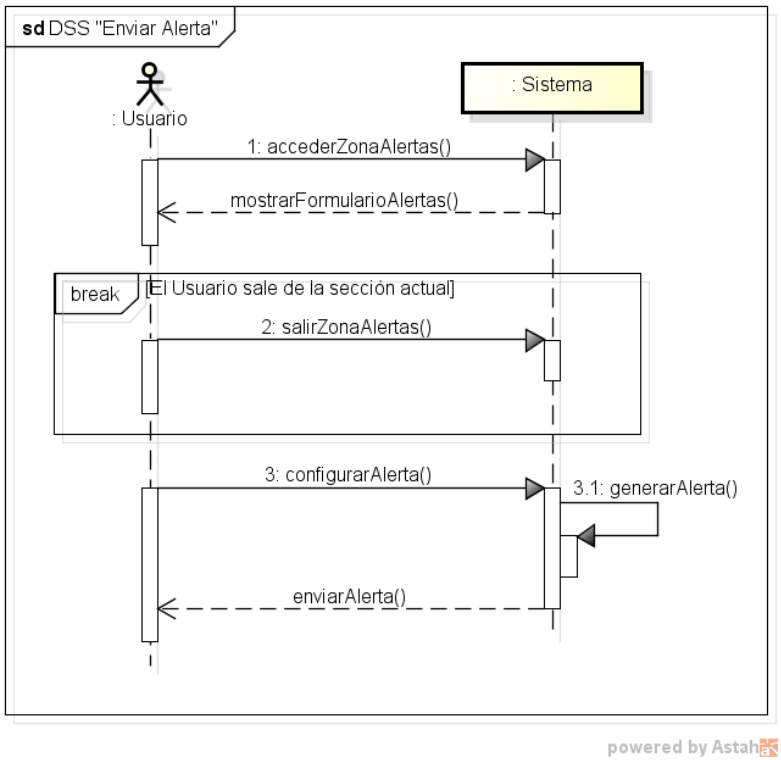


Figura 5.14: DSS "Enviar Alerta"

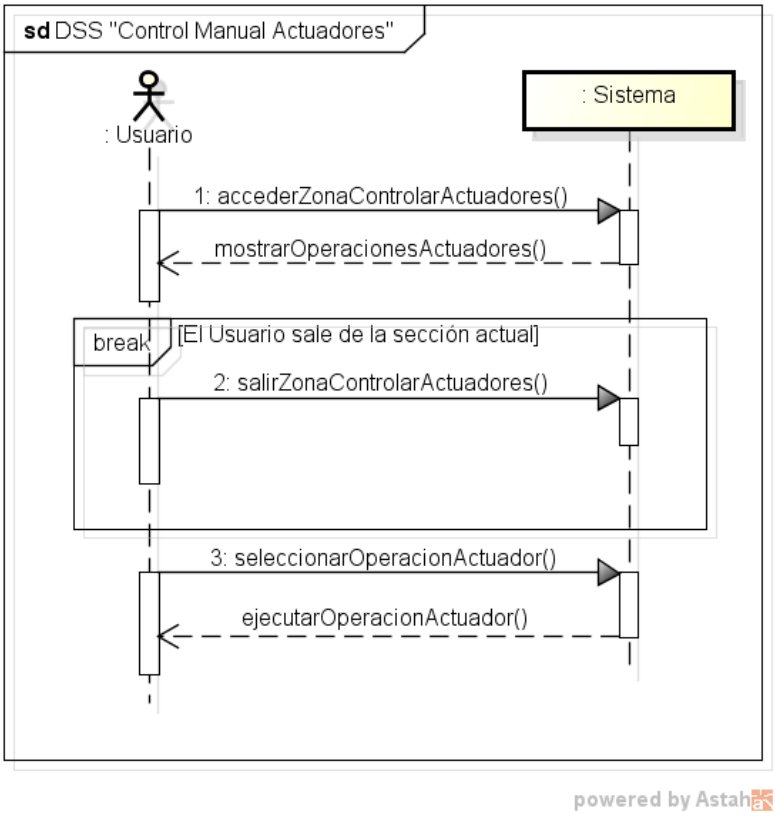


Figura 5.15: DSS "Control Manual Actuadores"

Bloque IV: Diseño

Capítulo 6: Diseño de la aplicación

En este capítulo se llevará a cabo el diseño de la arquitectura del sistema propuesto.

Se comenzará justificando las condiciones que deberán ser garantizadas con el diseño arquitectónico, incluyendo los aspectos organizacionales. Continuaremos dividiendo el sistema completo en subsistemas lógicos que nos ayudarán a comprender qué elementos físicos serán necesarios para componer el sistema. En este punto se prestará atención a las relaciones entre los diferentes módulos. A lo largo del capítulo, se hablará de los modelos y patrones que se han tenido en cuenta a la hora de diseñar el sistema.

6.1. Diseño arquitectónico

Esta sección toma como punto de partida todo el trabajo de análisis desarrollado en el anterior bloque de la presente memoria. Poseemos una serie de requisitos que describen nuestro sistema, los cuales deben satisfacerse para un correcto funcionamiento del mismo. El diseño que se irá implementando en las siguientes secciones debe hacerse de tal forma que garantice las siguientes condiciones:

- **Rendimiento**: ya hemos mencionado que nuestra aplicación surge de la necesidad de monitorizar algún lugar en tiempo real, teniendo en cuenta esta característica y que en numerosos casos el lugar a monitorizar estará relacionado con el mundo empresarial (sala de servidores, por ejemplo), la herramienta debe ofrecer tiempos de respuesta razonables ya que no sería admisible mostrar datos desfasados.

- **Integridad**: el sistema debe garantizar la integridad de los datos. El sistema posee datos sensibles de los cuales depende que se pongan en marcha mecanismos de actuación o no, por lo tanto, los datos deben mantener su integridad siempre.

- **Disponibilidad**: la disponibilidad es otra condición indispensable en este proyecto ya que si el sistema no realiza sus funciones, la monitorización del lugar desaparece.

- **Confidencialidad**: la confidencialidad asegura que el acceso al contenido de la aplicación sea sólo para aquellos usuarios que tengan permiso.

6.2. Organización del sistema

El hecho de que los datos y la funcionalidad general de nuestro sistema pueda ser accedida desde varias localizaciones, inmediatamente nos lleva a pensar en un diseño arquitectónico basado en el modelo cliente-servidor. Este modelo hace un uso efectivo de sistemas en red, puesto que las funcionalidades generales no han de ser implementadas en cada cliente. Con un simple acceso a la red, los usuarios de nuestro sistema podrán acceder a él y a su funcionalidad sin importar su localización. En la Figura 6.1 podemos observar el funcionamiento básico del modelo con sus dos grandes subsistemas: cliente y servidor.

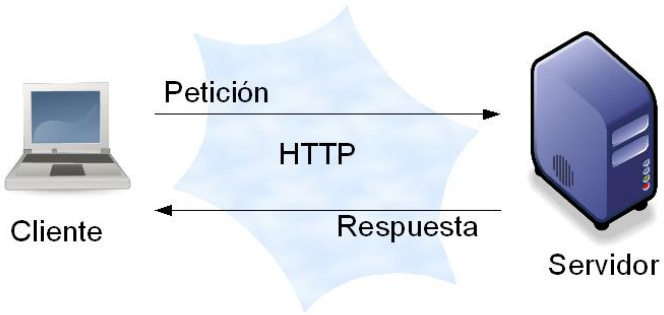


Figura 6.1: modelo cliente-servidor

En este modelo cliente-servidor se pretende integrar el modelo de repositorio de datos ya que los diferentes componentes del sistema compartirán los mismos datos: valor de temperatura, valor de humedad, estados… Este modelo de repositorio se recomienda en sistemas que como el nuestro, poseen grandes volúmenes de información persistentes a lo largo del tiempo y que se dirigen por datos: un cambio en el estado del repositorio dispara acciones, que serán llevadas a cabo por los actuadores de nuestro sistema.

Por último, se aplicará a todo el sistema el modelo de capas, a través de ellas se gestionará adecuadamente la complejidad del sistema proporcionando una descomposición jerárquica. Se aplicarán 3 capas: la capa relacionada con la lógica de la presentación, la relacionada con la lógica de la aplicación y la capa de persistencia. Cada una de estas tres capas sólo podrá relacionarse con sus capas adyacentes.

A continuación se explican las capas que intervienen en el modelo:

- **Lógica de Presentación**: la funcionalidad de esta capa consiste en mostrar al usuario la interfaz gráfica de la aplicación. Interactúa directamente con el usuario de forma que este realizará solicitudes a la capa y esta, para satisfacerlas, dependerá de la capa que contiene la lógica de la aplicación a quien realizará las correspondientes peticiones.

- **Lógica de Aplicación**: la funcionalidad de esta capa consiste en dar soluciones a todos los Casos de Uso que se han identificado anteriormente. Depende de la capa relacionada con la persistencia de los datos pues necesita recuperarlos para realizar sus funciones.

- **Persistencia de Datos**: la funcionalidad de esta capa consiste en almacenar los datos de forma persistente. La capa de la lógica de la aplicación hace uso de esta capa para proporcionar respuesta a las peticiones que la soliciten.

En la Figura 6.2 mostramos modelo de tres capas propuesto:

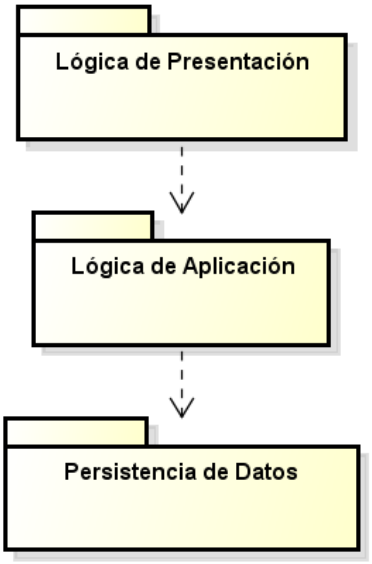


Figura 6.2: modelo de capas

Recapitulando, organizaremos la arquitectura del sistema siguiendo un modelo cliente-servidor de tres capas con un repositorio de datos. De modo que nuestro servidor quede dividido en dos subsistemas: uno se encargará de la lógica de la aplicación y el otro de los datos.

6.3. Descomposición modular

En este punto ya hemos identificado los subsistemas básicos conceptuales de nuestro sistema, a continuación se mostrará una descomposición en componentes. Mediante el diagrama de componentes (Figura 6.3) representaremos cómo se divide sistema en componentes mostrando las relaciones de dependencia entre ellos.

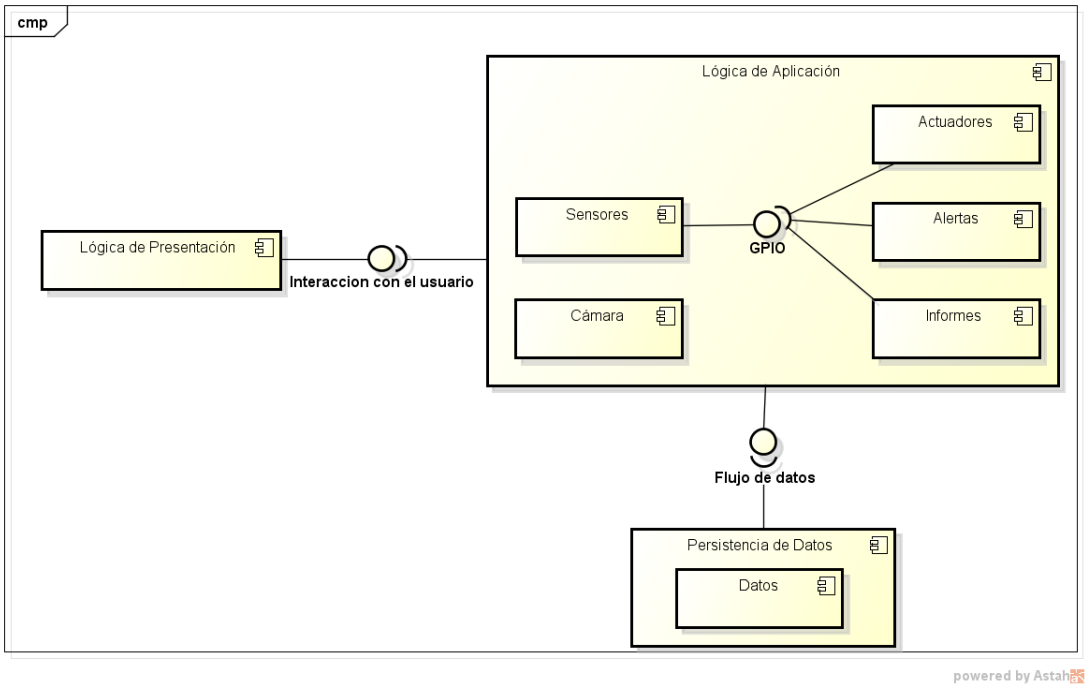


Figura 6.3: diagrama de componentes

A continuación se detallará la funcionalidad de los módulos que componen el sistema.

6.3.1. Funcionalidad módulo "Lógica de Presentación"

Este módulo tiene la funcionalidad de interactuar con el usuario de la aplicación. Se comunica con la lógica de la aplicación informando de la petición que el cliente lanza e informando al usuario una vez se haya procesado dicha petición.

6.3.2. Funcionalidad módulo "Sensores"

La funcionalidad básica de este módulo consiste en obtener información del exterior. Dentro de este módulo encontraremos los sensores DHT11 y al detector de gases. Además de poder encender y apagar los sensores se va a implementar una toma de decisiones. En función de los valores obtenidos, el módulo de los sensores se comunicará con los módulos de alertas y actuadores proporcionando una interfaz a la que nos referiremos como GPIO (ya que la obtención de los datos se consigue a través de los pines GPIO de la Raspberry Pi).

6.3.3. Funcionalidad módulo "Actuadores"

Este modulo tiene la funcionalidad de proporcionar una respuesta cuando sea necesario. La decisión de activar un actuador o no se lleva a cabo en el módulo de sensores. Por ello el módulo de actuadores requiere de la interfaz GPIO a través de la cual se comunican.

Bien es cierto que esté modo también implementa una funcionalidad manual gestionada por el usuario de forma independiente al módulo de sensores. En este caso, el módulo se comunicaría directamente con el módulo de la lógica de presentación donde el usuario interactuaría con el sistema activando o desactivando los actuadores.

6.3.4. Funcionalidad módulo "Alertas"

La funcionalidad de este módulo consiste en enviar alertas a través de un servicio de mensajería instantánea. La decisión de cuándo enviar una alerta se lleva a cabo en el módulo de sensores que proporciona la interfaz GPIO al módulo de alertas para llevar a cabo la comunicación.

De forma análoga al modulo de los actuadores, este módulo también puede funcionar de forma independiente al módulo de sensores, en este caso nos encontraríamos ante el envío manual de alertas que generaría el usuario por su cuenta. Es decir, este módulo se comunicaría directamente con el módulo de la lógica de presentación donde el usuario interactuaría con el sistema configurando la alerta a enviar.

6.3.5. Funcionalidad módulo "Informes"

La funcionalidad básica de este módulo es la de generar informes de temperatura y humedad. Este módulo se relaciona con el módulo de la lógica de presentación ya que el usuario deberá interaccionar con el sistema configurando los informes que desea generar. A su vez, este módulo de informes también se relaciona con el módulo de persistencia de datos ya que necesita acceder a ellos para generar los informes.

6.3.6. Funcionalidad módulo "Cámara"

Este módulo encierra toda la funcionalidad que podemos llevar a cabo con la Raspberry Pi Camera: encender/apagar la cámara, iniciar el *streaming*, mover la cámara, sacar una fotografía con una calidad determinada y grabar un vídeo de una duración determinada. Este módulo se relaciona con el módulo de la lógica de presentación ya que el usuario deberá interaccionar con el sistema seleccionando las operaciones que desea llevar a cabo con la cámara.

6.3.7. Funcionalidad módulo “Datos”

La funcionalidad de este módulo consiste en almacenar los datos que deben mantenerse de forma persistente en el sistema. Se encuentra dentro del componente de persistencia de datos.

6.4. Flujo de control

Los modelos de control a nivel de la arquitectura se encargan del flujo de control entre subsistemas, en nuestro caso hemos optado por un control centralizado (un subsistema controla al resto). Concretamente se ha optado por el Modelo Gestor, donde un componente se diseña como un gestor del sistema que controla el inicio, la parada y la coordinación del resto de subsistemas o módulos que pueden ejecutarse en paralelo, precisamente por ello este modelo es muy apropiado para sistemas de control en tiempo real.

El controlador central del sistema continuamente realizará ciclos de consulta sobre el estado de sensores y otros procesos para detectar eventos o cambios de estado. En la Figura 6.4 vemos un esquema de su funcionamiento.

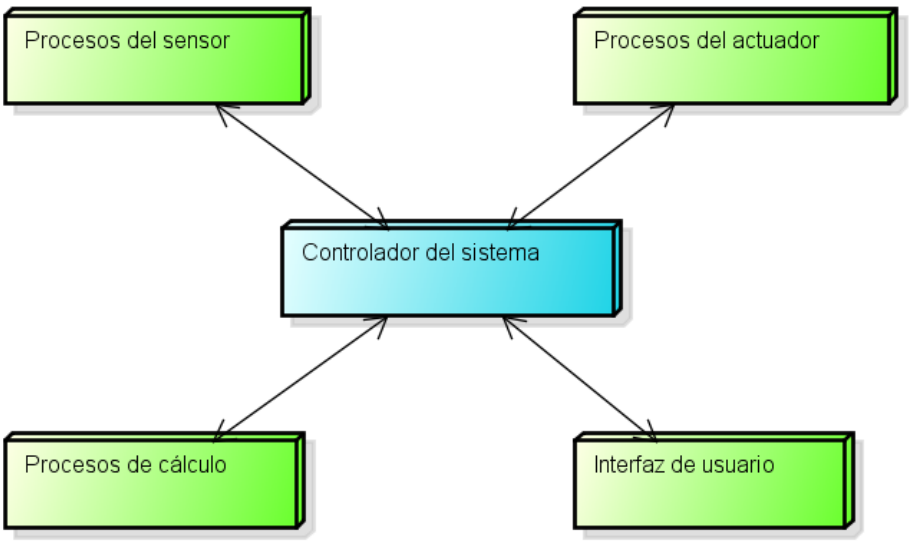


Figura 6.4: modelo gestor

6.5. Diseño físico de la arquitectura

Con el análisis conceptual que se ha llevado a cabo en las secciones anteriores ya nos encontramos en condiciones de ofrecer una aproximación más cercana a la solución del problema con una serie de elementos físicos y sus interrelaciones (incluyendo los protocolos de comunicación empleados) que recogeremos en el diagrama de despliegue de la Figura 6.5.

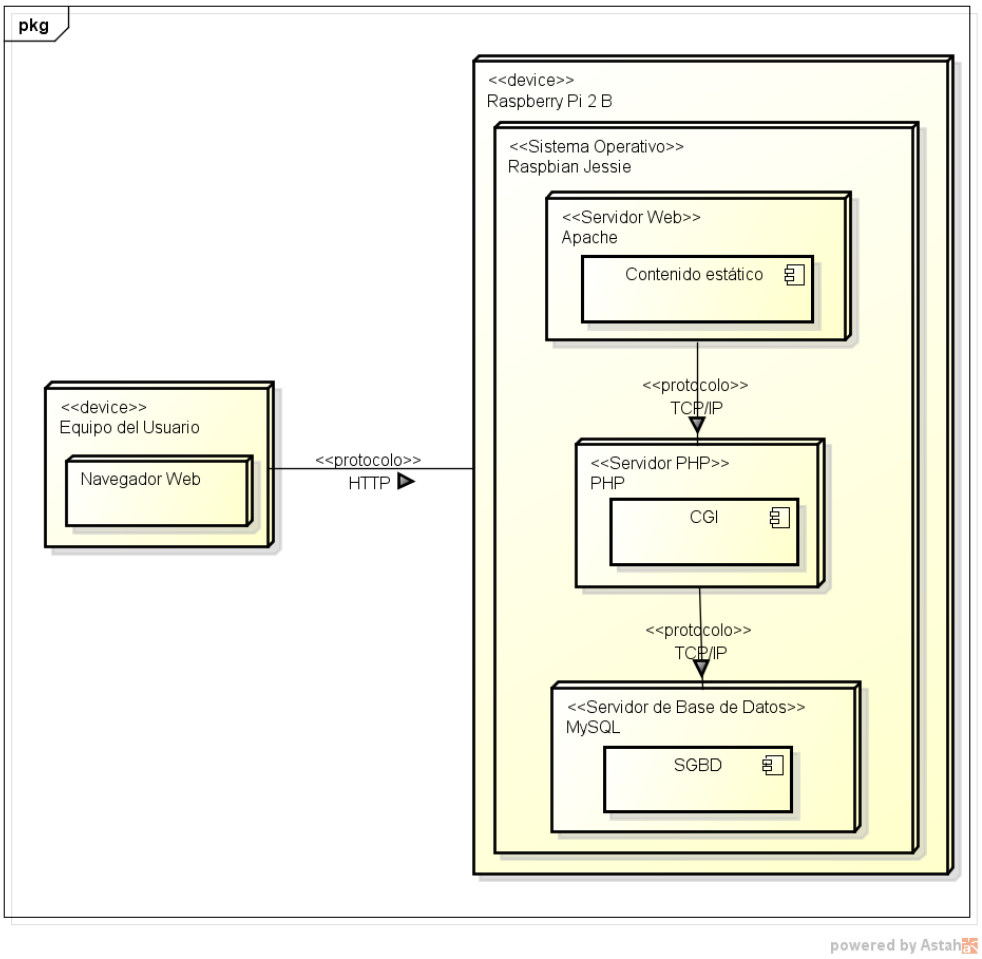


Figura 6.5: diagrama de despliegue

Como vemos en el diagrama, toda la parte del servidor se aloja en la Raspberry, incluyendo el servidor web, el servidor PHP y el servidor de Base de Datos. Estos elementos podrían encontrarse en máquinas diferentes, sin embargo hemos decidido centralizarlo en una única máquina a fin de abaratar costes. A continuación se describirán los elementos que han intervenido en este diagrama de despliegue:

- **Equipo del usuario**: hace las funciones del cliente del modelo cliente-servidor visto con anterioridad. Realiza peticiones al sistema.
- **Servidor Web**: recibe las peticiones del usuario y envía al servidor PHP los ficheros con los que debe trabajar.
- **Contenido estático**: es aquel contenido en el que no se producen cambios. Ejemplos de contenido estático sería el logo de la aplicación, títulos o imágenes del sitio web.
- **Servidor PHP**: interpreta dinámicamente el código PHP ofreciendo respuestas al usuario.
- **CGI**: este módulo contiene el código PHP que generará código HTML de forma dinámica.
- **Servidor de Base de Datos**: se encarga del almacenaje y la gestión de los datos de forma persistente. Un ejemplo de ello serían los valores de temperatura y humedad.
- **SGBD**: es el Sistema Gestor de Bases de Datos por el que se ha optado, en nuestro caso MySQL.

6.6. Patrones utilizados

En esta sección hablaremos de los patrones que hemos empleado para llevar a cabo la funcionalidad del sistema durante la etapa de diseño, justificando la elección elegida.

Como en cualquier otro proyecto, el uso de patrones de diseño siempre es una buena práctica ya que es una forma de aportar soluciones generales a problemas concretos. Estas soluciones además de ser reconocidas por la comunidad de desarrolladores de software tienen la ventaja de que son soluciones muy testadas por expertos en la materia.

6.6.1. Modelo-Vista-Controlador (MVC)

El MVC es un patrón arquitectónico de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el modelo, la vista y el controlador, es decir, por un lado define componentes para la representación de la información, y por otro lado para la interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

En la Figura 6.6 podemos ver cómo se relacionan el modelo, la vista y el controlador.

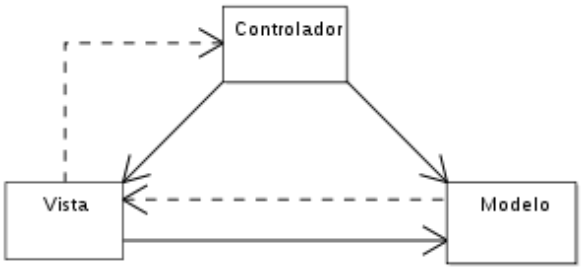


Figura 6.6: patrón arquitectónico MVC

En un caso típico la secuencia de llamadas entre estos elementos es la siguiente, vamos a ilustrarlo con un ejemplo aplicado a nuestro sistema: el usuario solicita encender el sensor de temperatura y humedad, el controlador recibe esta petición y llama a la correspondiente función del modelo para después pasar a la vista los datos obtenidos desde el modelo.

La justificación de este patrón radica en que es uno de los más utilizados para la creación de páginas web y además nos permite diferenciar perfectamente entre el modelo de dominio, la persistencia de los datos y la interfaz gráfica de usuario.

Capítulo 7: Diseño de la Base de Datos

En este capítulo se va a profundizar sobre la Base de Datos que hemos creado para mantener la persistencia de los datos en este proyecto, a esta Base de Datos la hemos llamado "BD_TFG" y contiene todas las tablas necesarias para llevar a cabo la funcionalidad del sistema.

Primeramente se explicará al inicio del capítulo todas las decisiones de diseño llevadas a cabo, justificando la propia estructura de la Base de Datos así como la razón por la que se ha empleado el motor de almacenamiento MyISAM o InnoDB. El capítulo continuará mostrando las relaciones de las tablas en un diagrama Entidad-Relación y finalizará realizando un estudio exhaustivo de todas las tablas, explicando su cometido y detallando los campos que las forman.

7.1. Decisiones de diseño

Para el diseño de la Base de Datos nos hemos basado en garantizar dos principios básicos necesarios para el correcto funcionamiento del sistema:

- **Seguridad**: debemos tener en cuenta que la aplicación RaSpy está diseñada para monitorizar ciertos parámetros de lugares privados, ya se esté empleando en empresas o domicilios. No sería deseable que cualquier intruso sin permisos accediese al sistema, puesto que con malas intenciones podría espiar el habitáculo, enviar alertas falsas, manipular el sistema para que no actúe si ocurre un incendio…

- **Rapidez**: nuestra aplicación muestra datos (temperatura, humedad, estados…) en tiempo real, con lo cual esta característica es primordial para el buen funcionamiento de la aplicación. Mostrar datos con retardo ocasionaría confusiones y podría generar fallos en la aplicación.

MySQL nos permite utilizar diferentes motores de almacenamiento, entre los que se encuentran dos de los más utilizados: innoDB y MyISAM.

El motor de almacenamiento o *storage-engine* se encarga de almacenar, manejar y recuperar información de una tabla. La elección de uno u otro dependerá del escenario en el que nos encontremos. Si necesitamos transacciones, claves foráneas y bloqueos, la mejor opción es escoger InnoDB. Por el contrario, en aquellos casos en los que predominen las consultas SELECT a la base de datos escogeremos MyISAM.

La mayor parte de las consultas que se ejecutan en nuestra aplicación son de tipo SELECT, las cuales típicamente se emplean para mostrar algún tipo de información en tiempo real al usuario, ya sea el valor de la temperatura, de la humedad o es estado de algún actuador. Debido esa necesidad de rapidez de la que hablábamos unas líneas más arriba, emplearemos el motor de almacenamiento MyISAM para que maneje aquellas tablas que muestran información en tiempo real al usuario de la aplicación. Sin embargo, aquellas tablas que no se utilizan para mostrar información en tiempo real serán manejadas con InnoDB.

7.2. Diagrama Entidad-Relación

En la Figura 7.1 presentamos el diagrama de Entidad-Relación donde podemos ver las relaciones existentes entre las tablas de la Base de Datos. Cuatro tablas ("sensor_temperatura", "configuracion_DHT11", "estado_sensores" y "estado_periodicidad_informes") serán manejadas con el motor de almacenamiento MyISAM y las otras cuatro ("control_acceso", "muestreo", "muestreodiario" y "muestreomensual") con InnoDB. Esta decisión se justifica en la frecuencia de acceso a los datos, de modo que las tablas MyISAM

almacenan información a la cual se accede con una frecuencia muy superior a la información almacenada en las tablas InnoDB.

Evidentemente accederemos con una frecuencia mucho mayor a las tablas que almacenan los valores de la temperatura y de la humedad cada pocos minutos (intervalo configurable) o del estado de los componentes en tiempo real que la frecuencia de acceso a las tablas que almacenan los datos de acceso a la aplicación o las que almacenan datos destinados a la generar informes, en cuyo caso la frecuencia de acceso es de un único acceso a la hora.

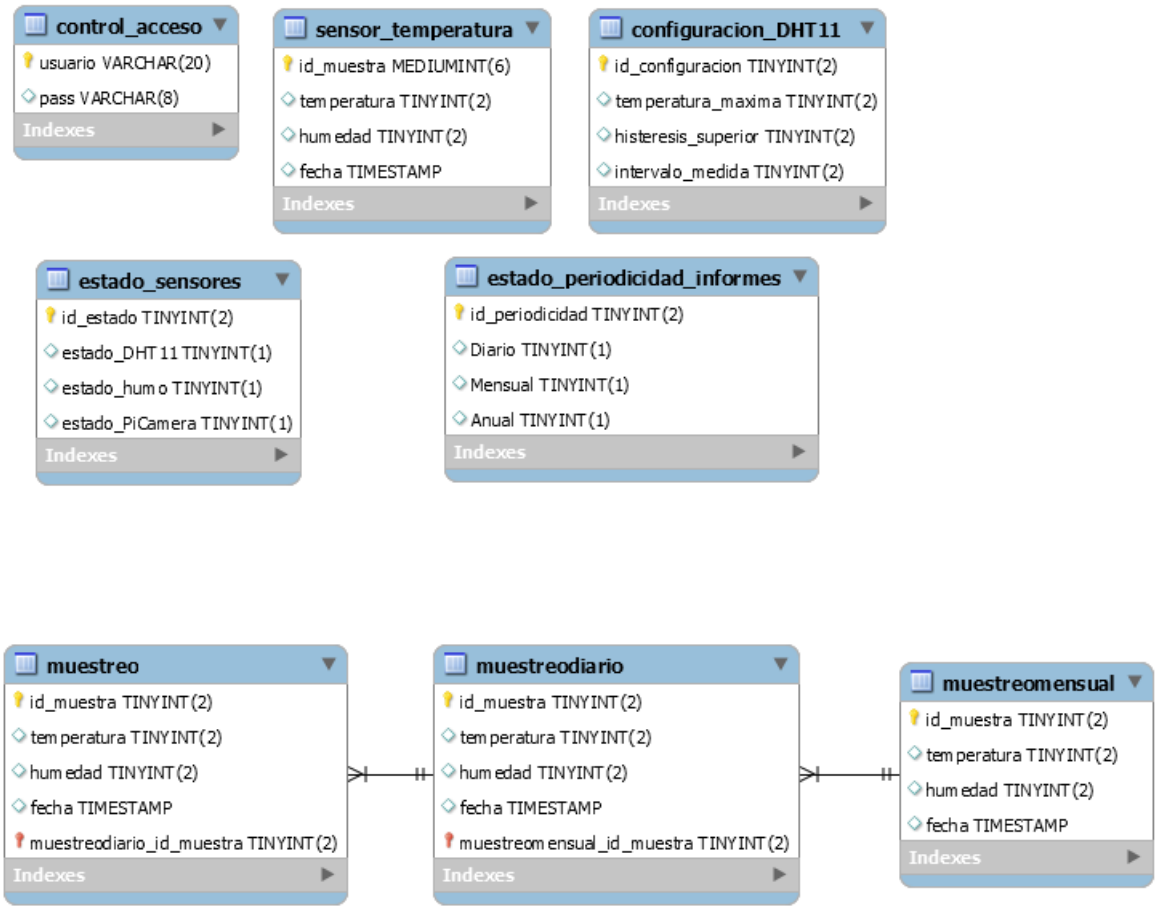


Figura 7.1: diagrama Entidad-Relación

Como vemos, las tablas que almacenan información de frecuente acceso no tienen relación con otras, son tablas MyISAM.

De las tablas InnoDB destacamos las relaciones 1..* entre las tablas “muestreo”, “muestreodiario” y “muestreomensual”. La cardinalidad múltiple de la relación entre “muestreo” y “muestreodiario” será un valor comprendido entre 0 y 24 ya que los valores que se inserten en “muestreodiario” dependerán del número de mediciones que se hayan recogido cada hora en la tabla “muestreo” siendo el máximo 24, las horas que tiene un día, ya que al finalizar el día se vaciará la tabla. Análogamente, la cardinalidad múltiple de la relación entre “muestreodiario” y “muestreomensual” será un valor comprendido entre 0 y 31, en caso de ser 31 estaríamos ante un mes de 31 días donde diariamente se ha insertado en “muestreodiario” una media de las mediciones de temperatura y humedad recogidas a lo largo del día.

7.3. Tabla “control_acceso”

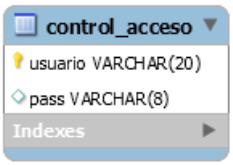


Figura 7.2: tabla BD “control_acceso”

Esta tabla almacena los credenciales necesarios para que un usuario se identifique ante el sistema. Sus campos son:

- **usuario**: define el nombre de usuario que se asigna al usuario del sistema. El administrador encargado de asignarlos debe tener en cuenta que el nombre ha de ser un identificador único compuesto por una cadena de hasta 20 caracteres.
- **pass**: define la contraseña de cada usuario, una cadena de 8 caracteres.

7.4. Tabla “sensor_temperatura”

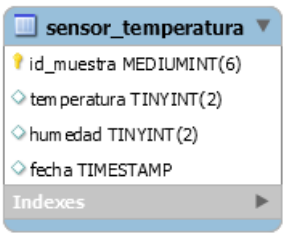


Figura 7.3: tabla BD “sensor_temperatura”

Esta tabla almacena los resultados de las mediciones de temperatura y humedad que se mostrarán en la aplicación web. Sus campos son:

- **id_muestra**: define un identificador único para cada muestra. Con el objetivo de no usar tipos de datos de tamaño superior al necesario empleamos el tipo de datos MEDIUMINT sin signo limitado a 6 caracteres por el siguiente razón: el rango de valores de este tipo de datos sin signo va de 0 a 16777215, si el usuario configurase la aplicación para que muestre la temperatura y la humedad en tiempo real con un refresco por minuto y el sistema estuviese encendido sin detenerse durante todo un año obtendríamos poco más de medio millón de registros, con lo cual no necesitamos los 8 dígitos del MEDIUMINT sin signo, basta con 6. Para que el número de registros no se dispare se implementará un mecanismo de contención al finalizar el año. El campo id_muestra es autoincremental.
- **temperatura**: almacena el valor de temperatura recogido por el sensor DHT11. Este sensor mide un rango de temperatura que va desde los 0º hasta los 50º, con lo cual bastará con un tipo de datos TINYINT sin signo (poseería un rango de 0 a 255), pero limitado a 2 dígitos.
- **humedad**: almacena el valor de humedad recogido por el sensor DHT11. El porcentaje correspondiente a la humedad relativa será un número positivo de hasta dos cifras con lo que volveremos a emplear un tipo de datos TINYINT(2) sin signo.

- **fecha**: es el momento en el que se realiza la medición: año, mes, día y hora.

7.5. Tabla “configuracion_DHT11”

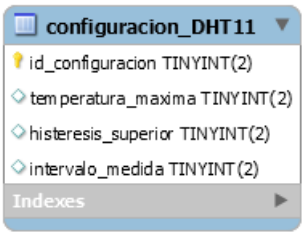


Figura 7.4: tabla BD “configuracion_DHT11”

Esta tabla recoge la configuración del sensor DHT11 que el usuario introduce desde la aplicación web, de este modo cuando el sistema se apaga y se vuelve a encender, el sensor trabajará con la última configuración introducida. Sus campos son:

- **id_configuracion**: define un identificador único para la configuración.
- **temperatura_maxima**: almacena la temperatura máxima que considera el usuario, a partir de la cual el sistema pondrá en funcionamiento sus mecanismos de actuación.
- **histereis_superior**: almacena el valor de histéresis, a efectos prácticos es el número de grados que el usuario considera que hay que esperar para que se detengan los mecanismos de actuación una vez que se han activado. Este valor siempre será menor al valor de temperatura con lo cual un tipo de datos TINYINT(2) sin signo será suficiente.
- **intervalo_medida**: almacena la tasa de refresco con la que el sistema realizará las mediciones de temperatura y humedad y con la que mostrará los valores actualizados. Este intervalo no debería ser muy elevado para un mayor control de la monitorización y por ello se ha limitado a dos dígitos.

7.6. Tabla “estado_sensores”

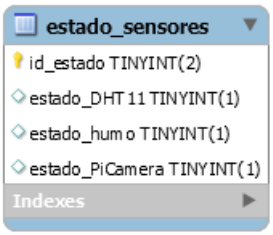


Figura 7.5: tabla BD “estado_sensores”

Esta tabla recoge el estado de los siguientes componentes del sistema: sensor DHT11, detector de gases y Raspberry Pi Camera. La razón de existencia de esta tabla radica en mantener el estado de los componentes ante un posible apagón del sistema, de modo que si un componente estaba encendido, cuando se reinicie el sistema vuelva a estar encendido. Sus campos son:

- **id_estado**: define un identificador único para el estado de los sensores.
- **estado_DHT11**: este campo almacena un 1 si el sensor DHT11 está activado y un 0 si se encuentra apagado.
- **estado_humo**: este campo almacena un 1 si el detector de gases está activado y un 0 si se encuentra apagado.
- **estado_PiCamera**: este campo almacena un 1 si la Raspberry Pi Camera está activada y un 0 si se encuentra apagada.

7.7. Tabla "estado_periodicidad_informes"

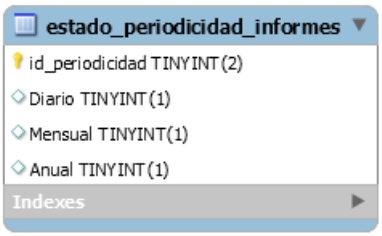


Figura 7.6: tabla BD "estado_periodicidad_informes"

Esta tabla recoge la periodicidad con la que se generan los informes automáticos. Al almacenar la periodicidad en la Base de Datos garantizamos que esta se mantenga igual tras reiniciar el sistema y además se le muestra al usuario.

- **id_periodicidad**: define un identificador único para las periodicidades de creación de informes.
- **Diario**: este campo almacena un 1 si se generan informes automáticos diariamente y un 0 si no.
- **Mensual**: este campo almacena un 1 si se generan informes automáticos mensualmente y un 0 si no.
- **Anual**: este campo almacena un 1 si se generan informes automáticos anualmente y un 0 si no.

7.8. Tabla "muestreo"

Figura 7.7: tabla BD "muestreo"

Esta tabla almacena los resultados de las mediciones de temperatura y humedad que se recogen cada hora y se emplearán para la generación de informes automáticos. Sus campos son:

- **id_muestra**: define un identificador único para cada muestra. Debido a que al final del día se resetea la tabla, el valor máximo que puede alcanzar este campo es de 24, basta con emplear un tipo de dato TINYINT(2) sin signo. El campo id_muestra es autoincremental.
- **temperatura**: almacena el valor de temperatura recogido por el sensor DHT11.
- **humedad**: almacena el valor de humedad recogido por el sensor DHT11.
- **fecha**: es el momento en el que se realiza la medición: año, mes, día y hora.

7.9. Tabla “muestreodiario”

Figura 7.8: tabla BD “muestreodiario”

Esta tabla almacena los resultados de temperatura y humedad generados diariamente a partir de los registros de la tabla “muestreo”. Se emplearán para la generación de informes automáticos y sus campos son:

- **id_muestra**: define un identificador único para cada muestra. Debido a que al final del mes se resetea la tabla, el valor máximo que puede alcanzar este campo es de 31, basta con emplear un tipo de dato TINYINT(2) sin signo. El campo id_muestra es autoincremental.
- **temperatura**: almacena el valor de temperatura recogido por el sensor DHT11.
- **humedad**: almacena el valor de humedad recogido por el sensor DHT11.
- **fecha**: es el momento en el que se realiza la medición: año, mes, día y hora.

7.10. Tabla “muestreomensual”

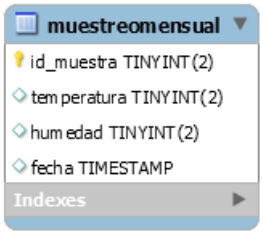


Figura 7.9: tabla BD “muestreomensual”

Esta tabla almacena los resultados de temperatura y humedad generados mensualmente a partir de los registros de la tabla “muestreodiario”. Se emplearán para la generación de informes automáticos y sus campos son:

- **id_muestra**: define un identificador único para cada muestra. Debido a que al final del año se resetea la tabla, el valor máximo que puede alcanzar este campo es de 12, basta con emplear un tipo de dato TINYINT(2) sin signo. El campo id_muestra es autoincremental.
- **temperatura**: almacena el valor de temperatura recogido por el sensor DHT11.
- **humedad**: almacena el valor de humedad recogido por el sensor DHT11.
- **fecha**: es el momento en el que se realiza la medición: año, mes, día y hora.

Bloque V: Implementación

Capítulo 8: Implementación

8.1. Instalación del Sistema Operativo

Típicamente, lo que se hace es introducir una imagen del Sistema Operativo preinstalado en una tarjeta de memoria microSD. En este caso se va a introducir la distribución Raspbian OS (versión Jessie) desde un ordenador con un ordenador portátil con ranura para tarjetas SD.

En primer lugar, se descarga el Sistema Operativo desde la propia página web de Raspberry (https://www.raspberrypi.org/downloads), al ser Raspbian uno de los SO oficiales. Ver Figura 8.1.

Figura 8.1: descarga de Raspbian

Una vez descargada y descomprimida la imagen se graba el archivo .img en la tarjeta microSD, para ello se ha empleado un software libre llamado Win32 Disk Imager. Esta aplicación tiene en su parte superior los campos “Image File” y “Device”, el primero hace alusión a la imagen que queremos grabar y el segundo se refiere al dispositivo (tarjeta microSD) donde queremos grabar la imagen. Una vez completados estos campos sólo debemos pulsar el botón “Write” para realizar el proceso de grabación.

También utilizaremos la herramienta Win32 Disk Imager para realizar copias de seguridad de nuestro sistema. Es recomendable seguir esta práctica periódicamente ya que un fallo en el dispositivo podría destruir nuestro progreso en el proyecto. Cada vez que en el sistema se han desarrollado programas de interés, se han implementado funcionalidades del sitio web o se ha instalado/configurado alguna herramienta, hemos creado copias de seguridad con esta misma aplicación. El procedimiento es el análogo al proceso de grabación, tan solo debemos introducir la tarjeta microSD con nuestra imagen personalizada en el ordenador, seleccionarla en el campo “Device” y posteriormente en el campo “Image File” indicar el directorio de nuestro ordenador donde

queremos alojar el resultante fichero .img, una vez configurados estos campos sólo debemos pulsar el botón "Read" y esperar a que finalice el proceso de copia. En la figura 8.2 vemos la interfaz de Win32 Disk Imager.

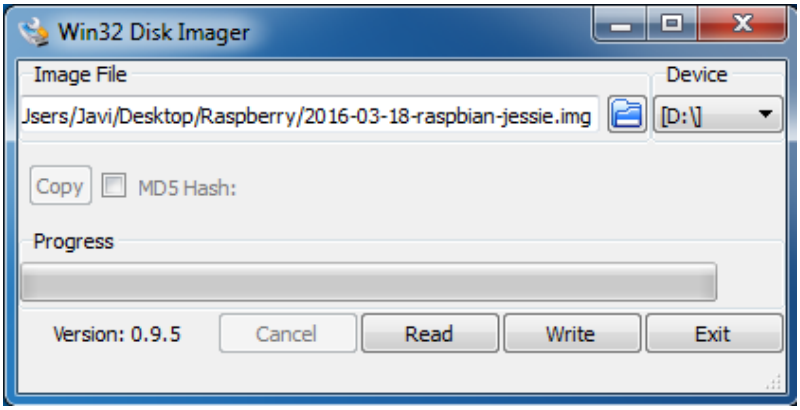


Figura 8.2: interfaz de Win32 Disk Imager

Una vez terminado el proceso de grabación se inserta la tarjeta microSD en la ranura correspondiente de nuestra Raspberry y ya tenemos la Raspberry Pi 2 operativa con un SO instalado. Ver Figura 8.3.

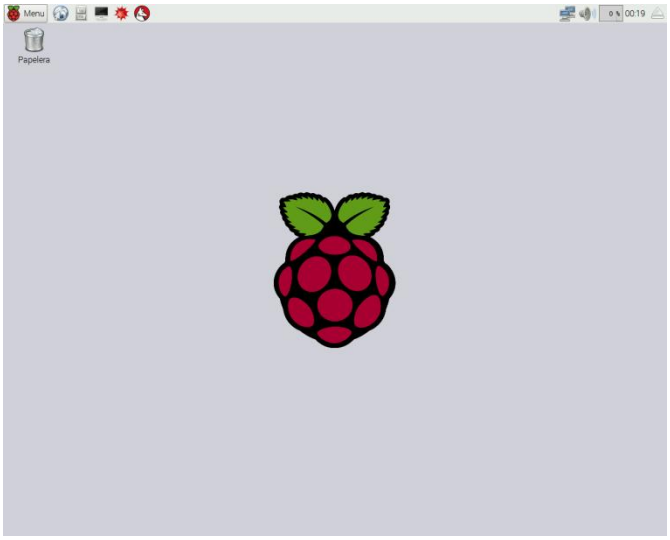


Figura 8.3: escritorio de Raspbian

Obviamente también se puede instalar el Sistema Operativo de la Raspberry desde Linux, para ello se repetirá el proceso anterior sólo que una vez descargada y descomprimida la imagen, esta se vuelca en la tarjeta microSD mediante la herramienta dd, que creará una copia exacta del fichero sin necesidad de emplear otro software adicional.

Para hacer todavía más simple y accesible la instalación del Sistema Operativo en el mini ordenador la organización Raspberry Pi ha desarrollado la aplicación NOOBS (New Out Of Box Software). NOOBS es un instalador de Sistemas Operativos para la Raspberry, que al arrancarse por primera vez crea las particiones necesarias y permite al usuario instalar los Sistemas Operativos oficiales de Raspberry Pi: Raspbian OS, OpenELEC, Windows 10 IoT Core y RISC OS, junto con PiNET (para dar clases con Raspberry Pi en red y centralizadas), OSMC (distribución enfocada a centro multimedia), Ubuntu MATE (versión totalmente funcional de Ubuntu con escritorio MATE) y Snnapy Ubuntu Core (versión minimalista de Ubuntu basada en la nube y con un nuevo instalador similar al empleado en los smartphones). Una vez instalado el SO se puede trabajar con la Raspberry con normalidad, pero NOOBS queda residente en memoria y siempre se podrá acceder a esta aplicación (pulsando la tecla Shift en el arranque) permitiendo cambiar a otro SO con facilidad.

8.2. Estructura de directorios

En este capítulo explicaremos qué hace cada uno de los programas que se han implementado. Organizaremos el capítulo dividiéndolo en dos secciones principales que hacen referencia a los dos directorios en los que se ha trabajado.

8.2.1. Directorio /home/pi/

Se ha seleccionado este directorio como el directorio de trabajo desde el cual se implementarán los archivos necesarios para construir el sistema (siempre que no sean susceptibles de aparecer en el directorio del servidor web, /var/www/html/, que veremos en la siguiente sección). Los archivos que implementaremos bajo este directorio tienen una función directa sobre los elementos del sistema: mover el servo, recoger muestras de temperatura y humedad, encender/apagar un actuador… Dicho de otra forma, cuando el usuario de la aplicación web realiza una petición, el controlador encargado de gestionarla típicamente llamará a alguno de estos programas.

En este directorio se han instalado algunas de las librerías necesarias para el correcto funcionamiento del sistema, como la librería GPIO (directorio "RPi.GPIO-0.5.11"), para el manejo de los puertos GPIO y la librería Highcharts (directorio "Highcharts-5.0.2"), para mostrar la gráfica correspondiente al informe online.

También se han instalado aquí algunas de las aplicaciones que el sistema usa, como la aplicación Telegram (en el directorio "tg"), para el envío de mensajes de alerta en tiempo real y la aplicación MJPG-Streamer (en el directorio "mjpg"), para el manejo de la Raspberry Pi Camera.

Se ha decidido hacer especial mención en los siguientes directorios, ya que es en ellos donde se han implementado los programas que componen el sistema.

Directorio "Sensor_DHT11"

En este directorio se almacena el archivo que empleamos para hacer uso del sensor de temperatura y humedad DHT-11, teniendo como objetivo la monitorización de estas magnitudes. Otros archivos también harán uso de este sensor, por ejemplo el archivo DHT11periodico.py, sin embargo su uso está claramente enfocado hacia la creación automática de informes periódicos, con lo cual, se ha decidido no incluirlo en este directorio separando así las funcionalidades.

- **AdafruitDHT.py**: este programa escrito en python tiene una triple función: leer la temperatura y humedad empleando el sensor DHT-11, grabar los datos recogidos en la Base de Datos y activar los mecanismos de actuación (activar la alarma, activar el regulador de temperatura y enviar un mensaje de alerta a través de Telegram) en caso de que la temperatura leída supere la temperatura máxima configurada en el sistema. Al programa deberemos pasarle cuatro argumentos: el tipo de sensor que estamos usando (valor "11" referente al sensor DHT-11), el número del pin GPIO al que se encuentra conectado el sensor ("17" en nuestro caso), la temperatura máxima permitida (a partir de este valor se pondrán en marcha los mecanismos respuesta) y por último el valor de histéresis (para evitar que el sistema active y desactive los mecanismos de actuación de forma constante y repetitiva). Una vez leídos los valores de temperatura y humedad se almacenan en la Base de Datos. Por último el programa se pregunta si el valor de temperatura leído es superior al máximo permitido, en caso de ser así se activa la respuesta correspondiente, de no ser así, la respuesta se desactivará, pero sólo si la temperatura medida es inferior a la resta entre la temperatura máxima permitida y el valor de histéresis. En este momento vemos la importancia del valor de histéresis que el usuario ha configurado: sin este valor, los mecanismos de respuesta se activarían y desactivarían constantemente ya que al sobrepasar la temperatura máxima permitida el regulador de temperatura enfriaría inmediatamente el lugar, la temperatura bajaría y el regulador de temperatura se desactivaría, pero al momento la temperatura volvería a subir. Para evitar este bucle se emplea el valor de histéresis.

Directorio "Sensor_Humo"

En este directorio se encuentra el archivo que hará uso del detector de gases MQ-135.

- **humo.py**: este programa escrito en python emplea el detector de gases MQ-135 para comprobar si hay presencia de gases en el habitáculo o no. En caso de no detectarse gases el programa no hace nada, pero en caso de detectarse se ponen en marcha las respuestas del sistema ante la aparición de gases, que son: la activación de la alarma y el envio de una alerta en tiempo real a través de Telegram informando de la presencia de gases.

Directorio "piCamera"

En este directorio se almacenan los distintos programas destinados al manejo de la Raspberry Pi Camera. Además, se encuentra en este directorio la carpeta "videosGrabados", que es el lugar donde se almacenan los videos realizados por el usuario desde la aplicación web haciendo uso de la Raspberry Pi Camera.

- **encenderCamara.sh**: este programa lo emplearemos para inicializar el streaming (será visto desde la sección "Cámara" de la aplicación web). Lo primero que hace es crear (si no está creado) el directorio /tmp/stream, donde se depositarán las imágenes que componen el streaming. A continuación se ejecuta la herramienta raspistill que es la encargada de capturar los fotogramas y ajustar sus características (anchura, altura, calidad del fotograma, nombre del archivo, tiempo entre capturas, tiempo durante el cual se llevan a cabo las capturas). Con la aplicación raspistill sobrescribiendo fotogramas en el mismo archivo, entra en juego la aplicación mjpg streamer, la cual copia el fotograma (indicando al plugin de entrada la ubicación del archivo) y lo sirve a través de un pequeño servidor (indicando al plugin de salida la ubicación donde se encuentra instalada la aplicación y también el puerto, en nuestro caso el 8080).

- **apagarCamara.sh**: este script se encarga de apagar la Raspberry Pi Camera, para ello simplemente mata los procesos de raspistill y mjpg streamer que pudiesen estar corriendo en nuestro sistema.

- **capturar_imagen.sh**: como su propio nombre indica, este script es usado cuando el usuario quiere capturar un fotograma. Para ello, lo primero que hace el script es crear un nombre único para la imagen a capturar con el siguiente formato: <día-mes-año_hora:minuto:segundo>. A continuación llamaremos a la herramienta raspistill para que realice tome la fotografía, a raspistill le pasaremos el nombre único del archivo y también la calidad de la imagen que ha seleccionado el usuario (este valor se pasará al script como un argumento). Tal y como se indica a raspisitill, la imagen resultante se almacenará en el directorio /var/www/html/img/imagenesCapturadas.

- **grabar_video.sh**: este script se encarga de grabar vídeos. Para ello lo primero que hace es crear un nombre único para el vídeo resultante con el mismo formato empleado en el fichero capturar_imagen.sh (no habría problema en que una imagen y un vídeo tuviesen el mismo nombre ya que se almacenan en directorios diferentes). A continuación, se recoge el valor del tiempo (introducido por el usuario, en segundos) que se ha pasado al programa como único argumento y se multiplica por 1000 para pasarlo a milisegundos, que es la unidad con la que trabaja raspivid. Por último, se hace uso de la propia herramienta raspivid para grabar el vídeo, a la cual la pasaremos el nombre y el valor en milisegundos. Tal y como se indica a raspivid, el vídeo resultante se almacenará en el directorio /home/pi/piCamera/videosGrabados. El formato nativo del vídeo resultante es .h264.

Directorio "Scripts_Alertas"

En este directorio incluimos el script en bash con el cual haremos uso de Telegram.

- **send-telegram.sh**: emplearemos este script para enviar mensajes en tiempo real a través de la aplicación Telegram. Para realizar el envío deberemos pasarle al programa dos argumentos: el primero de ellos es el nombre de usuario en Telegram del destinatario del mensaje, el segundo de los argumentos corresponde con el texto del mensaje a enviar. Típicamente los mensajes serán o bien

alertas creadas por el usuario desde la aplicación web, o bien alertas automáticas generadas cuando se sobrepase la temperatura máxima permitida o se detecten gases.

Directorio "Actuadores"

En este directorio encontraremos los programas que actúan con los distintos actuadores.

- **alarmaOFF.py**: la alarma estará conectada al GPIO26 (sistema de numeración BCM), este programa la apaga.
- **alarmaON.py**: la alarma estará conectada al GPIO26 (sistema de numeración BCM), este programa la enciende.
- **consultaEstadoGPIO.py**: este script escrito en python sirve para conocer el estado de cualquier pin GPIO, devuelve un "1" si el pin está en alta o un "0" si está en baja. El pin GPIO sobre el cual queremos realizar la consulta se pasa al programa como un argumento.
- **reguladorOFF.py**: el regulador de temperatura estará conectado al GPIO21 (sistema de numeración BCM), este programa lo apaga.
- **reguladorON.py**: el regulador de temperatura estará conectado al GPIO21 (sistema de numeración BCM), este programa lo enciende.

Dentro de este directorio también encontramos la carpeta "servo" que contiene los programas que harán mover el servomotor sobre el que se instalará la cámara. Estos programas son:

- **servo_centrado.py**: este programa sirve para mover el servomotor hacia la posición de centrado (90º), para ello envía un pulso del 7.5% al GPIO19, que es donde tenemos conectado el servomotor.
- **servo_derecha.py**: este programa sirve para mover el servomotor hacia la derecha (0º), para ello envía un pulso del 2.5% al GPIO19, que es donde tenemos conectado el servomotor.
- **servo_izquierda.py**: este programa sirve para mover el servomotor hacia izquierda (180º), para ello envía un pulso del 12.5% al GPIO19, que es donde tenemos conectado el servomotor.

Directorio "informes_periodicos"

En este directorio se almacenan todos los archivos que intervienen en el proceso de automatización de informes periódicos en formato PDF. Antes de detallar en qué consisten los archivos pertenecientes a este directorio se ha considerado oportuno repasar el contenido de las siguientes tablas de la Base de Datos debido a su estrecha relación con los propios archivos.

- **Tabla "muestreo":** almacena las mediciones de temperatura y humedad recogidas cada hora a lo largo de un día.
- **Tabla "muestreodiario":** almacena las medias diarias de temperatura y humedad. Estas medias se calculan diariamente accediendo a los datos de la tabla "muestreo".
- **Tabla "muestreomensual":** almacena las medias mensuales de temperatura y humedad. Estas medias se calculan mensualmente accediendo a los datos de la tabla "muestreodiario".

Ahora pasamos a explicar los archivos.

- **DHT11periodico.py**: este script en python guarda gran similitud con el fichero AdafruitDHT.py, en este caso, el script se limita a medir la temperatura y la humedad (todas las horas a y veinte) para almacenar los resultados en la tabla "muestreo".

- **graba_muestreodiario.php**: lo primero que hace este programa es calcular las medias de temperatura y humedad de la tabla "muestreo", una vez obtenidas, se almacenan diariamente (a las 00:05) en la tabla "muestreodiario".

- **graba_muestreomensual.php**: lo primero que hace este programa es calcular las medias de temperatura y humedad de la tabla "muestreodiario", una vez obtenidas, se almacenan el primer día del mes (a las 00:06) en la tabla "muestreomensual".

- **borra_muestreo.php**: se encarga de vaciar la tabla "muestreo", la tabla seguirá con la misma estructura pero con 0 filas. El vaciado de esta tabla tendría lugar todos los días a las 00:15, después de que se hayan generado los posibles informes.

- **borra_muestreo_diario.php**: se encarga de vaciar la tabla "muestreodiario", la tabla seguirá con la misma estructura pero con 0 filas. El vaciado de esta tabla tendría lugar el primer día del mes a las 00:16, después de que se hayan generado los posibles informes.

- **borra_muestreo_mensual.php**: se encarga de vaciar la tabla "muestreomensual", la tabla seguirá con la misma estructura pero con 0 filas. El vaciado de esta tabla tendría lugar el primer día del año a las 00:17, después de que se hayan generado los posibles informes.

- **pdf_diario.php**: este programa tiene la función de generar un informe diario en formato PDF, para ello hará uso de la clase FPDF16 generando así la gráfica y la tabla correspondiente. Los datos con los que trabajará este programa se encuentran en la tabla "muestreo". Si el usuario efectivamente ha configurado el sistema para que se generen informes diarios automáticamente, este programa generará el informe a las 00:10 todos los días y empleando las muestras del día anterior, es decir, el día 2 de abril a las 00:10 se generará el informe diario del día 1 de abril.

- **pdf_mensual.php**: este programa tiene la función de generar un informe mensual en formato PDF, para ello hará uso de la clase FPDF16 generando así la gráfica y la tabla correspondiente. Los datos con los que trabajará este programa se encuentran en la tabla "muestreodiario". Si el usuario efectivamente ha configurado el sistema para que se generen informes mensuales automáticamente, este programa generará el informe a las 00:11 el primer día de cada mes empleando las muestras del mes anterior, es decir, el día 1 de abril a las 00:11 se generará el informe mensual del mes de marzo.

- **pdf_anual.php**: este programa tiene la función de generar un informe anual en formato PDF, para ello hará uso de la clase FPDF16 generando así la gráfica y la tabla correspondiente. Los datos con los que trabajará este programa se encuentran en la tabla "muestreomensual". Si el usuario efectivamente ha configurado el sistema para que se generen informes anuales automáticamente, este programa generará el informe a las 00:12 el primer día del año empleando las muestras del año anterior, es decir, el día 1 de enero de 2017 a las 00:12 se generará el informe anual del año 2016.

Directorio "contenedorPDF"

En este directorio se almacenan todos los informes periódicos que se generan automáticamente, ya sean diarios, mensuales o anuales. El archivo contenedorPDF.php accede a esta carpeta para mostrar los informes que se han generado y están disponibles para su descarga.

8.2.2. Directorio /var/www/html/

Es la ubicación predeterminada de nuestro servidor Apache, el lugar donde se ha ubicado nuestro árbol web: de este directorio cuelgan el resto de ficheros y directorios que componen nuestro sitio web. A continuación pasaremos a explicar cada uno de ellos, y por su importancia se han creado tres subsecciones específicas para detallar los directorios "contenido", "controladores" y "acceso".

- **Directorio "css"**: en este directorio se recogen todos los archivos CSS que se han empleado para definir la presentación y dar formato al contenido HTML que encontramos en las diferentes vistas.

- **Directorio "fpdf16"**: en este directorio se incluye FPDF en su versión 16. Es una clase escrita en PHP que permite generar documentos PDF desde el propio PHP. Se empleará esta clase siempre que queramos generar un informe PDF, ya sean los informes periódicos que se generan automáticamente, o los informes online que el usuario quiera guardar en formato PDF.

- **Directorio: "img"**: en este directorio se incluyen todas las imágenes que aparecen a lo largo de nuestro sitio web. Además incluye la carpeta "imagenesCapturadas" que es donde se almacenan las imágenes capturadas por el usuario con la Raspberry piCamera, esas imágenes (con su previsualización) serán ofrecidas para su posible descarga en galeria.php.

- **index.php**: es la página que nos encontramos al introducir la dirección de nuestro sitio web. Es una página de autentificación que solicita al usuario que cumplimente un formulario introduciendo su nombre y su contraseña. Este formulario será enviado al fichero control_acceso.php para la toma de decisiones.

Directorio "contenido"

En este directorio encontramos principalmente las vistas de nuestra aplicación web:

- **inicio.php**: esta vista es a la cual accedemos tras iniciar sesión, de alguna manera es la página principal de la aplicación ya que desde ella podemos conocer el estado de todos los elementos del sistema y manipular los sensores DHT-11 y MQ-135. En la parte superior nos encontramos con lo relativo al sensor de temperatura y humedad, en la parte izquierda nos encontraremos con el termómetro y el higrómetro y en la parte derecha con las opciones de configuración del sensor DHT-11 junto a la configuración vigente. En la parte central encontraremos los botones que nos permitirán encender/apagar el detector de gases. Por último, en la parte inferior de la vista se informa al usuario de aquellos elementos a los que no se accede desde la vista actual: alarma, refrigerador de temperatura y cámara.

- **camara.php**: esta vista ofrece al usuario todo lo relacionado con la cámara. Además del streaming de la imagen en directo, encontraremos botones que nos permitirán encender/apagar la cámara, moverla, tomar una fotografía o grabar un vídeo (habiendo configurado previamente la calidad de la fotografía y la duración del vídeo).

- **galeria.php**: esta vista está dividida en dos secciones. En primer lugar se muestran y se permite la descarga de todas las imágenes que el usuario ha capturado mediante la cámara y se encuentran almacenadas en el directorio /var/www/html/img/imagenesCapturadas. De forma análoga se muestran y se permite la descarga de todos los vídeos grabados por el usuario con la Pi Camera, estos vídeos se encuentran almacenados en el directorio /home/pi/piCamera/videosGrabados.

- **forzar_descarga.php**: este archivo realmente es un controlador que hemos incluido en este directorio y que será usado por galería.php y contenedorPDF.php para descargar contenido a través de la aplicación web. Básicamente se encarga de gestionar las descargas de contenido, ya sean informes, imágenes o vídeos.

- **informes.php**: esta vista está dividida en dos secciones. Por un lado, nos encontramos ante una sección donde el usuario seleccionará las fechas de inicio y final sobre las que se generará un informe instantáneo. Por otro lado, el usuario seleccionará la periodicidad con la que quiere que se generen informes automáticos.

- **contenedorPDF.php**: esta vista muestra y permite la descarga de todos los elementos del directorio /home/pi/contenedorPDF, que es el lugar donde se almacenan los informes periódicos que han sido generados automáticamente.

- **estadisticas.php**: teniendo en cuenta las fechas y las horas introducidas por el usuario en informes.php, esta vista muestra un informe online que contiene una gráfica y una tabla con los datos más significativos de la misma. Los datos generados se almacenan en variables de sesión por si el

usuario decide generar un archivo PDF del informe, estos datos serían empleados en guardarEstadisticas.php.

- **guardarEstadisticas.php**: presenta el mismo informe que podíamos encontrar en estadisticas.php pero en PDF, disponible para descargar y con un estilo más austero. Utiliza los datos generados en estadisticas.php, de esta forma nos evitamos un buen número de consultas innecesarias, ya que la información a mostrar es exacta. En la vista de la previsualización del informe PDF a descargar encontramos una cabecera, una gráfica y una tabla.

- **alertas.php**: esta vista ofrece al usuario la posibilidad de enviar un mensaje de alerta en tiempo real a través de Telegram cuyo contenido el propio usuario puede configurar rellenando los campos de un formulario: destinatario de la alerta, tipo de incidencia, descripción de la incidencia e identificación del emisor de la alerta.

- **control.php**: esta vista ofrece al usuario la posibilidad de encender/apagar los actuadores (alarma y regulador de temperatura) manualmente a través de los diferentes botones dispuestos en la vista.

- **salir.php**: es la vista que nos encontramos al cerrar sesión, nos ofrece un enlace para volver a la página de autentificación del sitio web.

Directorio "controladores"

En este directorio se encuentran los controladores encargados de gestionar los diferentes tipos de peticiones por parte de los usuarios:

- **controladorAccesoBD.php**: este programa primeramente nos conecta con la Base de Datos y en segundo lugar selecciona la Base de Datos concreta con la que vamos a trabajar: "BD_TFG".

- **controladorConfigDHT11Histeresis.php**: este script selecciona el valor de histéresis según la configuración del sensor DHT-11 introducida por el usuario.

- **controladorConfigDHT11Intervalo.php**: este script selecciona el intervalo de medición según la configuración del sensor DHT-11 introducida por el usuario.

- **controladorConfigDHT11Tmax.php**: este script selecciona la temperatura máxima permitida según la configuración del sensor DHT-11 introducida por el usuario.

- **controladorElementosOrdenados.php**: este script se encarga de listar todos los elementos de un directorio y mostrarlos ordenados atendiendo al nombre del fichero. Dependiendo del directorio donde se encuentran los elementos, los mostrará de una forma determinada, pero siempre aptos para descargar.

- **controladorEncendidoApagadoAlarma.php**: gestiona las peticiones de encendido y apagado de la alarma, llamando a los archivos alarmaON.py y alarmaOFF.py respectivamente.

- **controladorEncendidoApagadoCamara.php**: gestiona las peticiones de encendido y apagado de la cámara, llamando a los archivos encenderCamara.sh y apagarCamara.sh respectivamente. También actualiza el estado de la cámara: 0 (apagada) ó 1 (encendida).

- **controlador EncendidoApagadoDHT11.php**: teniendo en cuenta la configuración del sensor introducida por el usuario, gestiona la petición de encendido llamando al archivo AdafruitDHT.py y automatizando la tarea en el crontab. Si la petición es la de apagado basta con detener la automatización de la tarea en el crontab y desactivar los mecanismos de actuación que pudieran estar activados. Este fichero también actualiza el estado del sensor: 0 (apagado) ó 1 (encendido).

- **controlador EncendidoApagadoHumo.php**: este fichero gestiona las peticiones de encendido y apagado del detector de gases MQ-135. En caso de recibir una petición de encendido automatizará la ejecución del programa humo.py cada minuto en el crontab. Si la petición es la de apagado basta con

detener la automatización de la tarea en el crontab y desactivar los mecanismos de actuación que pudieran estar activados. También actualiza el estado del sensor: 0 (apagado) ó 1 (encendido).

- **controlador EncendidoApagadoRegulador.php**: gestiona las peticiones de encendido y apagado del refrigerador, llamando a los archivos reguladorON.py y reguladorOFF.py respectivamente.

- **controladorEnviarAlerta.php**: este programa gestiona las peticiones de envío de alertas, para ello llamará al fichero send-telegram.sh que recibirá como parámetros los datos introducidos por el usuario al cumplimentar el formulario de alertas: destinatario, código de la alerta, descripción de la alerta y emisor.

- **controladorEstadoCamara.php**: este script selecciona el estado de la cámara: 0 (apagada) ó 1 (encendida). Si el estado de la cámara es 1 (encendida), automáticamente lanzará el programa que enciende la cámara (encenderCamara.sh), de esta forma, cuando el sistema se reinicie se encontrará en las mismas condiciones.

- **controladorEstadoDHT11.php**: este script selecciona el estado del sensor de temperatura y humedad: 0 (apagado) ó 1 (encendido).

- **controladorEstadoHumo.php**: este script selecciona el estado del detector de gases: 0 (apagado) ó 1 (encendido).

- **controladorInformesPeriodicos.php**: este script gestiona las peticiones de usuario relacionadas con la automatización de informes periódicos. Por un lado se encargará de escribir en el crontab las líneas necesarias para llevar a cabo la automatización de los informes teniendo en cuenta las opciones de periodicidad seleccionadas por el usuario. Además mantendrá la configuración de periodicidad vigente de forma que el usuario pueda verla desde el sitio web.

- **controladorOperacionesCamara.php**: gestiona las peticiones en las que el usuario desea realizar alguna operación con la cámara, ya sea tomar una fotografía o grabar un vídeo, llamando a los archivos capturar_imagen.sh y grabar_video.sh respectivamente. Sin embargo, debemos tener en cuenta que durante la transmisión de un streaming nuestra cámara no puede ejecutar otra acción, con lo cual lo primero que haremos será comprobar el estado de la cámara y sólo si esta se encuentra encendida deberemos: detenerla (apagarCamara.sh), realizar la operación y volver a dejarla como estaba (encenderCamara.sh). Si se encuentra apagada basta con realizar la operación.

- **controladorServo.php**: gestiona las peticiones que mueven el servomotor hacia la izquierda, hacia el centro y hacia la derecha llamando a los archivos servo_izquierda.py, servo_centrado.py y servo_derecha.py respectivamente.

- **controladorTermometros.php**: es el archivo que proporciona la información a la petición Ajax incluida en el archivo inicio.php. Para ello, accede a la Base de Datos y selecciona la última muestra (la más actual) de temperatura y humedad recogida por el sensor DHT-11.

Directorio “acceso”

En este directorio se encuentran los dos ficheros que implementan la seguridad de nuestro sitio web, impidiendo el acceso a personal no autorizado:

- **seguridad.php**: cualquier página restringida hace un llamamiento a este fichero, en él se verifica si el usuario está correctamente identificado, si es así permite el acceso a la página solicitada, pero si no está autentificado nos devuelve a la página de registro index.php.

- **control_acceso.php**: recibe el usuario y la contraseña de index.php y comprueba si existe esa combinación de credenciales. En caso afirmativo se creará una variable de sesión que será comprobada por el archivo seguridad.php y accederemos a inicio.php. Si los credenciales no son válidos volveremos a la página de registro index.php.

8.3. Crontab

Una de las partes primordiales en la implementación del sistema es la realizada en el fichero /etc/crontab, gracias al cual podremos automatizar algunas de las tareas que requieren de esta automatización.

Los sistemas operativos Unix tienen un demonio cron (en nuestra distribución alojado en /etc/init.d) que se encarga de administrar procesos en segundo plano. Su funcionamiento es sencillo, se ejecuta cada minuto revisando la tabla de tareas crontab.

Crontab es un simple archivo de texto que guarda una lista de comandos a ejecutar en un tiempo especificado por el usuario, verificará la fecha y hora en que se debe ejecutar el script, los permisos de ejecución y lo realizará en segundo plano. Cada usuario puede tener su propio archivo crontab, en nuestro caso hemos empleado en archivo crontab del usuario root (/etc/crontab) para la automatización de las tareas del sistema.

Para agregar tareas a /etc/crontab basta con emplear la estructura del crontab y así hacer posible la automatización. Esta estructura se explica con claridad en la Figura 8.4.

Figura 8.4: resumen de crontab

Centrándonos en nuestro fichero /etc/crontab, vamos a ir enumerando y explicando las líneas que hemos agregado para conseguir las automatizaciones requeridas por el sistema. Empezamos con aquellas relacionadas con el encendido/apagado de los sensores DHT-11 y MQ-135 (ver Figura 8.5).

```
17 */5 * * * * root sudo python /home/pi/Sensor_DHT11/AdafruitDHT.py 11 17 27 2
18 * * * * * root sudo python /home/pi/Sensor_Humo/humo.py
```

Figura 8.5: crontab para sensores

La línea 17 está reservada para el sensor de temperatura y humedad DHT-11. Cuando el sensor se apaga, esta línea se comenta indicando para qué está reservada. Sin embargo, cuando el sensor está activado nos encontramos con la línea 17 que aparece en la captura de la Figura 8.5, en ella se ejecuta el fichero AdafruitDHT.py atendiendo a la configuración del sensor impuesta por el usuario de la aplicación web. En este caso se ejecutaría el programa cada 5 minutos fijando el valor máximo de temperatura permitido en 27ºC y el valor de histéresis en 2ºC.

La línea 18 se ha reservado para el detector de gases MQ-135. Cuando el detector se apaga, esta línea se comenta indicando para qué está reservada. Sin embargo, cuando el detector está activado nos encontramos con la línea 18 que aparece en la captura de la Figura 8.5, en ella se ejecuta el fichero humo.py, que se ejecutará cada minuto revisando si hay presencia de gases en el habitáculo. La periodicidad con la que se ejecuta este programa no es configurable, viene impuesta por el sistema. Su frecuente periodicidad radica en los elevados daños potenciales que pudiesen desencadenarse debido a la detección tardía de un posible incendio. Es decir, a la hora de controlar la temperatura del habitáculo, no es tan crítica una medición constante: la temperatura no variará drásticamente y si durante algunos minutos el habitáculo se encuentra por encima de la temperatura

máxima permitida, no debería ocasionar mayores problemas. Desde luego que estos problemas serían mucho mayores si durante esos minutos hay un incendio en la habitación.

Otra tarea que está íntimamente relacionada con el fichero /etc/crontab es la de generar automáticamente informes periódicos, para ello nos situaremos entre las líneas 21 y 29 de nuestro crontab (ver Figura 8.6).

```
20 #AUTOMATIZACION DE INFORMES:
21 20 * * * * root sudo python /home/pi/informes_periodicos/DHT11periodico.py 11 17
22 10 0 * * * root php /home/pi/informes_periodicos/pdf_diario.php
23 15 0 * * * root php /home/pi/informes_periodicos/borra_muestreo.php
24 5 0 * * * root php /home/pi/informes_periodicos/graba_muestreodiario.php
25 11 0 1 * * root php /home/pi/informes_periodicos/pdf_mensual.php
26 16 0 1 * * root php /home/pi/informes_periodicos/borra_muestreo_diario.php
27 6 0 1 * * root  php /home/pi/informes_periodicos/graba_muestreomensual.php
28 12 0 1 1 * root php /home/pi/informes_periodicos/pdf_anual.php
29 17 0 1 1 * root php /home/pi/informes_periodicos/borra_muestreo_mensual.php
```

Figura 8.6: crontab para la periodicidad de informes

Estas filas aparecerán comentadas (indicando para qué están reservadas) si no se requiere la automatización de sus tareas asociadas. La captura de la Figura 8.6 no muestra ninguna línea comentada, eso significa que nos encontramos ante una configuración en la que el usuario ha optado por activar la creación automática de informes diarios, mensuales y anuales.

La fila 21 está reservada para obtener la temperatura y la humedad cada hora. Para ello haremos uso del programa python DHT11periodico.py, que se ejecutará todas las horas a y veinte minutos y guardará los datos recogidos en la tabla “muestreo”. Esta línea debe aparecer siempre que queramos generar algún tipo de informe, ya sea diario, mensual o anual. Es el punto de contacto con el habitáculo y a partir de los datos recogidos generaremos otros datos que nos proporcionen información organizada sobre el conjunto de la muestra.

La fila 22 está reservada para generar informes diarios. Para ello haremos uso del programa pdf_diario.php, que se ejecutará sólo si el usuario ha activado la creación automática de informes diarios. Este programa se ejecuta diariamente a las 00:10, accediendo a la tabla “muestreo” y teniendo en cuenta todas las muestras (máximo 24) que se han tomado a lo largo del día.

La fila 23 está reservada para vaciar la tabla “muestreo” (resultados de cada hora). Para ello haremos uso del programa borra_muestreo.php, que se ejecutará diariamente a las 00:15 vaciando la tabla “muestreo”. Esta línea debe aparecer siempre que queramos generar algún tipo de informe, ya sea diario, mensual o anual. Por supuesto, el vaciado de la tabla tendrá lugar después de que se hayan generado los informes correspondientes y se hayan grabado en otras tablas datos que requerían la participación de aquellos almacenados en la tabla “muestreo”.

La fila 24 está reservada para guardar la media de “muestreo” en “muestreodiario”. Para ello haremos uso del programa graba_muestreodiario.php, que se ejecutará diariamente a las 00:05 calculando la media de las muestras de la tabla “muestreo” y grabándola en “muestreodiario”. Esta línea debe aparecer siempre que queramos generar informes mensuales o anuales. Como es lógico, la grabación de la media tendrá lugar antes de que se vacíe la tabla “muestreo” y antes de que se generen los posibles informes.

La fila 25 está reservada para generar informes mensuales. Para ello haremos uso del programa pdf_mensual.php, que se ejecutará sólo si el usuario ha activado la creación automática de informes mensuales. Este programa se ejecuta mensualmente el primer día del mes a las 00:11, accediendo a la tabla “muestreodiario” y teniendo en cuenta todas las muestras (máximo 31) que se han tomado a lo largo del mes.

La fila 26 está reservada para vaciar la tabla “muestreodiario” (resultados de cada día). Para ello haremos uso del programa borra_muestreo_diario.php, que se ejecutará mensualmente el primer día de cada mes a las 00:16 vaciando la tabla “muestreodiario”. Esta línea debe aparecer siempre que queramos generar informes mensuales o anuales. Por supuesto, el vaciado de la tabla tendrá lugar después de que se hayan generado los informes correspondientes y se hayan grabado en otras tablas datos que requerían la participación de aquellos almacenados en la tabla “muestreodiario”.

La fila 27 está reservada para guardar la media de “muestreodiario” en “muestreomensual”. Para ello haremos uso del programa graba_muestreomensual.php, que se ejecutará mensualmente el primer día del mes a las 00:06 calculando la media de las muestras de la tabla “muestreodiario” y grabándola en “muestreomensual”. Esta línea debe aparecer siempre que queramos generar informes anuales. Como es lógico, la grabación de la media tendrá lugar antes de que se vacíe la tabla “muestreodiario” y antes de que se genere el posible informe.

La fila 28 está reservada para generar informes anuales. Para ello haremos uso del programa pdf_anual.php, que se ejecutará sólo si el usuario ha activado la creación automática de informes anuales. Este programa se ejecuta anualmente el primer día del año a las 00:12, accediendo a la tabla “muestreomensual” y teniendo en cuenta todas las muestras (máximo 12) que se han tomado a lo largo del año.

La fila 29 está reservada para vaciar la tabla “muestreomensual” (resultados de cada mes). Para ello haremos uso del programa borra_muestreo_mensual.php, que se ejecutará anualmente el primer día del año a las 00:16 vaciando la tabla “muestreomensual”. Esta línea debe aparecer siempre que queramos generar informes anuales. Por supuesto, el vaciado de la tabla tendrá lugar después de que se haya generado el informe correspondiente.

8.4. Porciones de código a destacar

Todo el código fuente que se ha desarrollado se incluye en el soporte digital que acompaña a esta memoria, por supuesto, con los pertinentes comentarios que facilitarán la comprensión de los programas a cualquier persona externa al proyecto que se interese por el mismo. Por lo tanto, el objetivo de esta sección es ofrecer al lector aquellas porciones de código que destacan y son dignas de mención.

8.4.1. Consultar el estado de los elementos

A lo largo del proyecto nos encontramos en repetidas ocasiones con la necesidad de saber si un elemento del sistema se encuentra encendido o apagado, para así reflejarlo en tiempo real en la aplicación web. Se ha considerado interesante señalar las diferentes formas mediante las cuales se ha llevado a cabo esta tarea.

Por un lado tenemos el método empleado para conocer el estado del sensor de temperatura y humedad, de la Raspberry Pi Camera y del detector de gases. Este método se basa en mantener una tabla en nuestra BD que almacena los estados de los distintos elementos (ver Figura 8.7), variando este estado entre 1 (encendido) y 0 (apagado).

estado_DHT11	estado_humo	estado_PiCamera
0	0	1

Figura 8.7: estado de los elementos mediante BD

Al realizar el usuario las operaciones de encendido/apagado de los distintos elementos desde la aplicación web, se modificarán los registros pertinentes en la tabla “estado_sensores”. En la Figura 8.8 se muestra un fragmento del fichero controladorEncendidoApagadoCamara.php para ilustrar este mecanismo, de forma análoga se encenderán y apagarán el resto de elementos que emplean este mecanismo.

```php
//si se presiona el botón de encendido...
if (isset($_POST['CamON'])){
    //se llevan a cabo las tareas propias de la cámara para encenderse
    exec('sudo sh /home/pi/piCamera/encenderCamara.sh');
    //se actualiza el estado de la cámara
    @mysql_query("UPDATE estado_sensores SET estado_PiCamera='1'");
    //se guarda el estado en una variable a la que accederemos cuando queramos conocerlo y
    //mostrarlo en la aplicación web
    $resultado_estado_PiCamera = @mysql_query("SELECT estado_PiCamera FROM estado_sensores");
}
```

Figura 8.8: código estado de los elementos mediante BD

El segundo método empleado puede llevarse a cabo siempre que el elemento del cual queremos conocer su estado se encuentre conectado a un pin GPIO. Consiste en preguntar directamente al pin GPIO si se encuentra en alta o en baja. Así conocemos el estado de la alarma y del regulador de temperatura para notificarlo en la página web. Se hace uso del programa python consultaEstadoGPIO.py al cual le pasamos como parámetro un entero correspondiente al pin GPIO en cuestión. En la Figura 8.9 vemos un fragmento de código donde preguntamos si el GPIO 'pin' se encuentra en alta o en baja.

```python
if (GPIO.input(int(pin))):
 print 1 #devuelve un '1' si el GPIO esta en alta
else:
 print 0 #devuelve un '0' si el GPIO esta en baja
```

Figura 8.9: fragmento de código de consultaEstadoGPIO.py

8.4.2. Modificar el crontab, comando 'sed'

Como hemos visto recientemente, en nuestro fichero /etc/crontab se encuentran una serie de líneas reservadas para automatizar determinadas tareas que se comentan y descomentan en función de las necesidades del sistema. Para ello se empleará el editor de flujo sed, con la opción –i haremos cambios en el archivo original, concretamente reemplazaremos líneas (opción c, de *change*) alternando entre líneas que siguen la sintaxis del crontab y comentarios dependiendo de las necesidades. En la Figura 8.10 mostramos las líneas de código (del archivo controladorEncendidoApagadoHumo.php) referentes a las modificaciones del fichero /etc/crontab que llevamos a cabo con la herramienta sed para automatizar y detener la automatización de la detección de gases, de forma análoga se automatizarán otras tareas.

```php
if (isset($_POST['activarHumo'])){
    //se va a automatizar el empleo del detector de gases una vez cada minuto
    exec ('sudo sed -i "18c* * * * * root sudo python /home/pi/Sensor_Humo/humo.py" /etc/crontab');
    //...
}
if (isset($_POST['desactivarHumo'])){
    //se detiene la detección automática de gases
    exec ('sudo sed -i "18c#FILA 18: reservada para el detector de gases MQ-135" /etc/crontab');
    //...
}
```

Figura 8.10: fragmento de código empleando el comando sed

Como vemos, antes de la opción "c" que nos permitirá reemplazar una línea, debemos indicar la línea que debe ser reemplazada, recordemos que cada línea de nuestro crontab tiene una funcionalidad específica.

8.4.3. Variables de sesión

Las sesiones son una forma sencilla de almacenar datos para usuarios de manera individual usando un ID de sesión único. Esto se puede usar para hacer persistente la información de estado entre peticiones de páginas.

Quizá el uso más extendido de las sesiones es el que empleamos para el control de acceso en control_acceso.php, donde crearemos una sesión con la variable de sesión "autentificado" que será comprobada siempre que queramos acceder a alguna página de nuestro sitio web. La duración de la sesión es de 1440 segundos (24 minutos).

Sin embargo, vamos a aprovechar la propiedad de las sesiones de hacer persistente la información de estado entre páginas en otro contexto diferente. Un buen escenario para poner en práctica esta propiedad es en las vistas estadisticas.php y guardarEstadisticas.php. Ambas vistas comparten información, concretamente son dos formas diferentes de ver el mismo informe: una de forma online y otra en formato PDF pero con los mismos datos en su interior.

En estadisticas.php se calculan una serie de valores (valor máximo, mínimo, media, desviación estándar y coeficiente de variación) que deberán ser mostrados tanto en esta vista como en guardarEstadisticas.php. Para no calcular estos datos por partida doble, lo que hacemos es calcularlos en estadisticas.php y crear variables de sesión para recuperarlos en guardarEstadisticas.php (ver Figura 8.11).

```
$_SESSION['Tmax'] = mysql_result($resultadoTmax,0);
$_SESSION['Tmin'] = mysql_result($resultadoTmin,0);
$_SESSION['Tmed'] = round(mysql_result($resultadoTmedia,0),2);
$_SESSION['Tdesviacion'] = round(mysql_result($resultadoTdesviacion,0),2);
$_SESSION['Tcv'] = round($resultadoTcv,2);
$_SESSION['Hmax'] = mysql_result($resultadoHmax,0);
$_SESSION['Hmin'] = mysql_result($resultadoHmin,0);
$_SESSION['Hmed'] = round(mysql_result($resultadoHmedia,0),2);
$_SESSION['Hdesviacion'] = round(mysql_result($resultadoHdesviacion,0),2);
$_SESSION['Hcv'] = round($resultadoHcv,2);
```

Figura 8.11: fragmento de código variables de sesión

8.4.4. Descargar contenido

A la hora de presentar contenido descargable en nuestro sitio web, haremos uso del fragmento de código que aparece en la Figura 8.12, ya sea una imagen, un vídeo o un informe.

```
<a href='forzar_descarga.php?file=$ruta_archivo&root=$directorio'>
```

Figura 8.12: fragmento de código contenido descargable

Como vemos, se presenta el contenido como un hipervínculo (etiqueta <a>) al fichero php que se encargará de gestionar la descarga (forzar_descarga.php), al cual se le pasarán dos variables por URL. Estas variables son la ruta del archivo y el directorio donde se encuentra.

8.4.5. Histéresis

Aplicamos un ciclo de histéresis al comportamiento de los actuadores ante una situación de temperatura excesiva. Dicho de otro modo, si únicamente activamos y desactivamos los mecanismos de respuesta en función de un único valor de temperatura excesiva, lo que haremos será generar una situación de conflicto en la cual los actuadores se activarán y desactivarán continuamente al estar muy cerca del valor máximo de temperatura permitida bien superándolo o bien acercándose a él. Aplicando esta propiedad de histéresis lo que haremos será crear un rango en el cual los actuadores una vez que se han activado, mantendrán su estado hasta alcanzar un

valor inferior determinado por la histéresis asignada, evitando así las continuas activaciones y desactivaciones de los actuadores. En la Figura 8.13 podemos ver el código que hace alusión a la histéresis:

```
# Si la temperatura es superior a la maxima permitida, se ponen en marcha los mecanismos de actuacion
if temperature > int(temperatura_maxima):
  print "Temperatura Alta"
  # Para activar la alarma:
  os.system("sudo python /home/pi/Actuadores/alarmaON.py")
  # Para activar el regulador de temperatura:
  os.system("sudo python /home/pi/Actuadores/reguladorON.py")
  # Para enviar la alerta automaticamente:
  os.system("/home/pi/Scripts_Alertas/send-telegram.sh Ejemplo 'MENSAJE AUTOMATICO: TEMPERATURA ELEVADA ("+str(temperature)+"ºC)'")
# Si no...
else:
  # Si la temperatura ya ha descendido tanto que es menor que el valor resultante de la resta entre la temperatura maxima permitida
  # y el valor de histeresis, entonces ya se pueden desactivar los mecanismos de respuesta. Como vemos, los mecanismos no se
  # detienen nada mas bajar del valor de temperatura maxima permitida, ya que en ese caso lo que sucederia es que nada mas activarse
  # el regulador de temperatura, esta descenderia un poco inmediatamente, y el regulador se desactivaria, pero al momento
  # la temperatura volveria a subir y deberia accionarse de nuevo, entrando en un bucle. De ahi la importancia de la histeresis
  if temperature < ( int(temperatura_maxima) - int(histeresis_superior) ):
    # Para desactivar la alarma:
    os.system("sudo python /home/pi/Actuadores/alarmaOFF.py")
    # Para apagar el regulador de temperatura:
    os.system("sudo python /home/pi/Actuadores/reguladorOFF.py")
```

Figura 8.13: fragmento de código histéresis

Como se aprecia en el código, partiendo de unas condiciones normales, los actuadores sólo se activarán en el caso de que la temperatura exceda del valor máximo permitido. Cuando suceda ese supuesto, es de esperar que la temperatura comience a descender, pero aun así los actuadores se mantendrán activos durante un rango de temperatura (aunque esta sea menor que la temperatura máxima permitida), hasta que sea inferior a la diferencia entre el valor máximo permitido y el valor de histéresis asignado. En ese momento los actuadores se desactivarán y sólo volverán a activarse si la temperatura vuelve a sobrepasar el valor máximo permitido.

8.4.6. Informes periódicos automáticos

Las operaciones implicadas en generar informes periódicos automáticos se llevan a cabo al finalizar el día/mes/año, por lo tanto se ejecutarán en los primeros instantes del nuevo día/mes/año. Esto nos lleva a tomar algunas consideraciones a la hora de operar con los registros. De forma resumida, deberemos añadir la cláusula WHERE en nuestras consultas SQL trabajando con el conjunto de datos del día/mes/año anterior. En la Figura 8.14 veremos cómo seleccionamos el valor máximo de temperatura de la tabla 'muestreodiario' (almacena medias diarias) teniendo en cuenta sólo los registros cuya fecha correspondan al mes anterior (aunque lo guardemos en la variable $mesactual):

```
$mesactual = date('m', strtotime('now - 1 month'));
$sql = "SELECT MAX(temperatura) FROM muestreodiario WHERE MONTH(fecha)='$mesactual'";
```

Figura 8.14: fragmento código pdf_mensual.php

Para calcular el año anterior lo hemos hecho siguiendo el mismo método, sin embargo, a la hora de calcular el día anterior, no lo hemos almacenado en una variable, sino que hemos hecho uso de la función DATE_SUB, que genera fechas a partir de un intervalo elegido. En nuestro caso seleccionábamos el día anterior del siguiente modo:

```
DATE_SUB(CURDATE(), INTERVAL 1 DAY)
```

De forma análoga, cuando en los archivos graba_muestreodiario.php (inserta un registro en la tabla 'muestreodiario' que contiene las medias de temperatura y humedad de un día concreto) y graba_muestreomensual.php (inserta un registro en la tabla 'muestreomensual' que contiene las medias de

temperatura y humedad de un mes concreto), insertamos los registros con las medias, en el campo de fecha ingresaremos el día/mes anterior, para que corresponda con los datos del día/mes empleados para calcular la media. Si no tuviésemos en cuenta esta manipulación en el campo de la fecha, en el registro correspondiente al primer día del mes quedarían almacenadas las medias del último día del mes anterior y en el registro del primer mes del año quedarían almacenadas las medias del último mes del año anterior.

Capítulo 9: Batería de pruebas

En este capítulo nos centraremos en las pruebas a las que se ha sometido el producto una vez finalizada su implementación. El desarrollo de la batería de pruebas tiene dos objetivos principales:

- Evaluar la calidad del producto.
- Mejorar el producto identificando defectos (típicamente implica deficiencias de calidad) y problemas (afecta a la funcionalidad del sistema final).

Debido a la modularidad de nuestro sistema, realizaremos pruebas de ámbito unitario con las que verificaremos el funcionamiento aislado de las piezas de software. Sin embargo, también se realizará alguna prueba de ámbito de sistema para poder verificar el comportamiento del sistema en su conjunto.

En el caso de las pruebas unitarias estaremos alerta y en busca de fallos relacionados con los requisitos funcionales, sin embargo en las pruebas de sistema perseguiremos comprobar los requisitos no funcionales.

La técnica empleada en las pruebas será la de "caja negra" (los casos de prueba se basan en el comportamiento de entrada/salida), estas pruebas están orientadas a la comprobación de los requisitos funcionales y nosotros de antemano ya conoceremos la salida esperada en cada prueba.

Antes de comenzar con la batería de pruebas, es conveniente aclarar que durante el desarrollo del sistema, concretamente a lo largo de la implementación, se han llevado a cabo verificaciones y validaciones (VV) paralelas al desarrollo del software. Estas VV son un conjunto de procedimientos, actividades y técnicas que pretenden asegurar que el software resuelve el problema inicialmente planteado. Debido al trabajo con estas VV en la fase de implementación, nos encontramos ante un sistema que se ha ido testando y perfeccionando, aun así es imprescindible realizar una batería de pruebas sobre el producto final.

P-01 Autenticación fallida

Objetivo	Comprobar el correcto funcionamiento del módulo de autenticación
Descripción	Se introducen diferentes combinaciones de credenciales no admitidos: usuario y contraseña incorrectos, usuario correcto con contraseña incorrecta y usuario incorrecto con contraseña correcta
Entrada	Credenciales erróneos
Salida	No permite el acceso al sitio web
Resultado	Esperado

Tabla 9.1: prueba-01 "Autenticación fallida"

P-02 Autenticación correcta

Objetivo	Comprobar el correcto funcionamiento del módulo de autenticación
Descripción	Se introducen los usuarios admitidos con sus correspondientes contraseñas
Entrada	Credenciales correctos

Salida	Permite el acceso al sitio web
Resultado	Esperado

Tabla 9.2: prueba-02 "Autenticación correcta"

P-03 Navegabilidad

Objetivo	Comprobar la navegabilidad de la aplicación web
Descripción	Se llevan a cabo todas las combinaciones de navegación a través de los menús y submenús del sitio web
Entrada	Selección de un menú o submenú
Salida	Cambio de vista a su página correspondiente
Resultado	Esperado

Tabla 9.3: prueba-03 "Navegabilidad"

P-04 Activación/Desactivación manual de sensores y actuadores

Objetivo	Verificar que la pulsación de los botones desencadena las acciones asociadas	
Descripción	Se prueban todos los botones existentes en el sitio web que activan y desactivan los distintos componentes del sistema	
Entrada	**1)** Activar DHT11	**2)** Desactivar DHT11
	3) Activar sensor humo	**4)** Desactivar sensor humo
	5) Activar cámara	**6)** Desactivar cámara
	7) Activar alarma	**8)** Desactivar alarma
	9) Activar regulador Tª	**10)** Desactivar regulador Tª
Salida	**1)** Comienza a medir la Tª y la humedad y queda reflejado en la página de inicio	**2)** Deja de medir la Tª y la humedad y así queda reflejado en la página de inicio
	3) Se pone en marcha el programa asociado y queda reflejado en la página de inicio	**4)** Se pone detiene el programa asociado y queda reflejado en la página de inicio
	5) Se muestra la imagen en tiempo real y queda reflejado en la página de inicio	**6)** Se oculta la imagen en tiempo real y queda reflejado en la página de inicio
	7) Se enciende el LED rojo y queda reflejado en la página de inicio	**8)** Se apaga el LED rojo y queda reflejado en la página de inicio
	9) Se enciende el LED verde y queda reflejado en la página de inicio	**10)** Se apaga el LED verde y queda reflejado en la página de inicio

Resultado	Esperado

Tabla 9.4: prueba-04 “Activación/Desactivación manual de sensores y actuadores”

P-05 Funcionamiento sensor DHT-11

Objetivo	Comprobar el correcto funcionamiento del sensor DHT-11
Descripción	Detendremos y pondremos en funcionamiento el sensor DHT-11 para comprobar si las acciones que esto genera son correctas
Entrada	**1)** Se inicia el sensor DHT-11 **2)** Se detiene el sensor DHT-11
Salida	**1)** En la web se refleja que el sensor está encendido, comienzan las mediciones de temperatura y humedad y se repiten periódicamente de acuerdo a la configuración del usuario **2)** En la web se refleja que el sensor está apagado y se detienen las mediciones de temperatura y humedad
Resultado	Esperado

Tabla 9.5: prueba-05 “Funcionamiento sensor DHT-11”

P-06 Valores coherentes de temperatura y humedad

Objetivo	Se pretende verificar que los valores de temperatura y humedad son coherentes
Descripción	Se comparan los resultados de temperatura y humedad que recoge el sensor DHT-11 con los de un termómetro ajeno al sistema
Entrada	Ejecuciones del programa que mide la temperatura y la humedad
Salida	Valores iguales o similares a los de otro termómetro ajeno al sistema
Resultado	Esperado

Tabla 9.6: prueba-06 “Valores coherentes de temperatura y humedad”

P-07 Movimiento de la cámara

Objetivo	Comprobar que la cámara se mueva en las tres posiciones esperadas
Descripción	Se realizan diferentes movimientos de la cámara de forma que se va posicionando en las tres posiciones

Entrada	Pulsación de los botones “izquierda”, “centro” y “derecha”
Salida	Movimiento del servomotor que sostiene la cámara hacia la izquierda, hacia el centro y hacia la derecha
Resultado	Esperado

Tabla 9.7: prueba-07 “Movimiento de la cámara”

P-08 Sacar fotografías

Objetivo	Comprobar el correcto funcionamiento de la cámara capturando imágenes
Descripción	Se sacan fotografías partiendo de un estado en el que la cámara estaba encendida y también estando apagada
Entrada	Se ajusta la calidad de la imagen y se presiona el botón que sacará una fotografía
Salida	La imagen capturada está disponible en la galería de contenido multimedia
Resultado	Esperado

Tabla 9.8: prueba p-08 “Sacar fotografías”

P-09 Grabación de vídeos

Objetivo	Comprobar el correcto funcionamiento de la cámara grabando vídeos
Descripción	Se graban vídeos partiendo de un estado en el que la cámara estaba encendida y también estando apagada
Entrada	Se ajusta la duración del vídeo y se presiona el botón que iniciará la grabación
Salida	El vídeo grabado está disponible en la galería de contenido multimedia
Resultado	Esperado

Tabla 9.9: prueba p-09 “Grabación de vídeos”

P-10 Descargar contenido desde la aplicación web

Objetivo	Verificar que el contenido descargable al cual se puede acceder desde la página web efectivamente se descarga
Descripción	Se accede a la galería multimedia y se descarga una imagen y un vídeo. Posteriormente se va a la sección de la web donde se almacenan los informes guardados y se descarga uno al azar

Entrada	Se selecciona la imagen, vídeo o informe a descargar
Salida	Comienza la descarga de la imagen, vídeo o informe
Resultado	Esperado

Tabla 9.10: prueba p-10 "Descargar contenido desde la aplicación web"

P-11 Envío de una alerta

Objetivo	Comprobar el correcto funcionamiento del sistema de alertas en tiempo real
Descripción	Se lleva a cabo una simulación donde generamos la alerta que deseamos enviar y se controla que llegue en tiempo real y con la información introducida por el usuario
Entrada	Se cumplimentan los campos del formulario para el envío de la alerta y se envía
Salida	Se recibe un mensaje de Telegram con el contenido de la alerta configurada
Resultado	Esperado

Tabla 9.11: prueba p-11 "Envío de una alerta"

P-12 Reacción del sistema ante una temperatura excesiva

Objetivo	Comprobar que funcionan correctamente los mecanismos de actuación del sistema cuando la temperatura es superior a la permitida
Descripción	Se va a forzar una situación en la que el sistema detecte que la temperatura es excesiva para verificar su correcta toma de decisiones
Entrada	Al programa que mide la temperatura y realiza la toma de decisiones, le pasaremos por parámetro una temperatura máxima inferior a la de la habitación
Salida	Se activa la alarma, el regulador de temperatura y se envía un mensaje vía Telegram que indica que la temperatura es excesiva
Resultado	Esperado

Tabla 9.12: prueba p-12 "Reacción del sistema ante una temperatura excesiva"

P-13 Informes instantáneos

Objetivo	Se pretende verificar el correcto funcionamiento a la hora de generar informes instantáneos configurados por el usuario

Descripción	Esta prueba comprende la verificación del proceso completo de generación de informes instantáneos: desde que el usuario configura el informe a generar hasta que obtiene el archivo PDF con el informe en cuestión. Los datos se comparan con los almacenados en la Base de Datos y además se calculan de forma ajena al sistema
Entrada	Se introduce fecha y hora de inicio y fin. Los datos que se encuentran dentro del rango son susceptibles de aparecer en el informe. También se selecciona la magnitud: temperatura y/o humedad
Salida	La gráfica representa adecuadamente los valores y su fecha asociada. Los datos de la tabla son correctos. El archivo informe PDF contiene la misma información que el informe online y se descarga con normalidad
Resultado	Esperado

Tabla 9.13: prueba p-13 "Informes instantáneos"

P-14 Informes periódicos

Objetivo	En esta prueba se pretende verificar la correcta automatización de informes periódicos
Descripción	Esta prueba comprende la verificación del proceso completo de automatización de informes periódicos: desde que el usuario selecciona la periodicidad hasta que los informes aparecen disponibles para su descarga en la página web
Entrada	**1)** Se prueban todos los programas que intervienen en la automatización de informes periódicos **2)** Se indica al sistema que queremos generar informes diarios, mensuales y anuales. En el caso de los informes mensuales y anuales modificamos la fecha de ejecución del programa para evitar esperas, pero siempre manteniendo el orden de ejecución de los programas **3)** Se seleccionan para descargar informes periódicos ya generados
Salida	**1)** Los nueve programas que intervienen funcionan correctamente: capturando información del sensor DHT-11, generando medias y grabándolas en otras tablas, limpiando las tablas y generando los informes en PDF **2)** La automatización es correcta **3)** Las descargas se llevan a cabo con normalidad
Resultado	Esperado

Tabla 9.14: prueba p-14 "Informes periódicos"

Bloque VI: Conclusiones y Anexos

Capítulo 10: Conclusiones

Este capítulo final de la memoria se escribe una vez que el desarrollo del proyecto ha llegado a su fin, en él se pretende hacer un balance honesto donde destacar los principales logros alcanzados y se reflexione sobre lo que este TFG ha aportado.

Tras meses de trabajo se ha desarrollado una herramienta que a través de una aplicación web nos permite monitorizar un lugar y gracias a ello, hacerlo más seguro. Por tanto, una vez más queda latente como la tecnología simplifica nuestras tareas: si queremos tener controlada la temperatura de una sala, no nos hará falta presenciarnos en ella para comprobarla, ahora disponemos de todo un sistema que recoge información del mundo real mediante sensores, procesa esta información y toma decisiones sobre si poner en marcha o no mecanismos de actuación.

El trabajo en este proyecto lo considero absolutamente enriquecedor, debido a sus características, el abanico de vertientes informáticas implicadas es muy amplio: desde el contacto con el propio hardware, hasta la implementación de programas escritos en diferentes lenguajes, pasando por el desarrollo de un entorno web ante el que se enfrentará el usuario. Por supuesto, el iniciar desde cero un proyecto informático de cierta envergadura y trabajar en todas y cada una de sus fases, considero que es una experiencia positiva y necesaria para la transición entre el mundo de la docencia y el mundo laboral, ayuda a madurar y a asentar conceptos.

Trabajar con la plataforma Raspberry Pi es otro de los puntos fuertes en la realización de este TFG, sin duda es un elemento muy accesible, versátil y con una gran comunidad tras él que invita a tenerlo en cuenta como solución a cualquier problema que queramos resolver en nuestro día a día. Recomendable para cualquier amante de la tecnología, estoy seguro de que volveré a trabajar con la plataforma, aunque sea en pequeños proyectos personales.

La toma de decisiones continuas a lo largo del desarrollo del sistema ha sido un aspecto muy relevante a mi juicio. Al abordar este proyecto no había un guión a seguir como en las prácticas de laboratorio, la responsabilidad de qué lenguaje emplear para un determinado sensor, qué librería usar o qué elemento comprar ha recaído en mi persona. Considero que la capacidad de decidir y poder justificar una decisión son valores que ayudan a crecer como profesional.

Finalmente, concluyo con que se han alcanzado los objetivos propuestos, se ha ofrecido una solución adecuada al problema y sobretodo, se ha aprendido.

10.1. Líneas de trabajo futuro

"Hay una forma de hacerlo mejor, encuéntrala."

La frase de Thomas Edison, aplicable a cualquier proyecto sea del campo que sea, invita a la autocrítica, a investigar sobre vías de evolución que mejoren una solución propuesta. Por tanto, en esta sección se van a proponer algunas líneas de trabajo futuro que podrían servir a alguien interesado en dar continuidad a este proyecto, tomándolo como punto de partida y a partir de él implementar mejoras en el sistema. Estas líneas de trabajo futuro son:

- Añadir más elementos al sistema, por ejemplo: sensor de movimiento, sistema de identificación por radiofrecuencia (tarjetas RFID), sensores de luminosidad que automaticen el encendido/apagado de luces…
- Desarrollar una aplicación móvil para Android o iOS que contenga la misma funcionalidad que la aplicación web.
- Aumentar la escalabilidad global del sistema, de forma que existan varias placas Raspberry monitorizando diferentes lugares y a la vez se centralice todo el funcionamiento del sistema global en una única Raspberry. De esta forma el número de datos aumentaría drásticamente y se podrían trabajar en líneas de Big Data.
- Hacer uso de JavaScript y Ajax para mejorar la interactividad del usuario con la aplicación web.
- Ampliar la sección referente a los informes, de forma que se generen informes que midan parámetros más allá de la temperatura y la humedad.
- Traducir la aplicación web a otros idiomas, de forma que desde la propia interfaz web seleccionemos el idioma en el que se quiera trabajar.

Anexo I: Manual de usuario

En este capítulo se ofrece un manual de usuario cuyo objetivo es instruir al lector sobre el modo de llevar a cabo las distintas funcionalidades del sistema, explicando los pasos a seguir y los elementos con los que interactuar.

AI.1. Acceder al sistema RaSpy

Para acceder al sistema basta con introducir la dirección del sitio web (http://tfgjgp.ddns.net/) en tu navegador, introducir los datos de acceso (combinación de usuario + clave) y pulsar el botón entrar. En la Figura AI.1 vemos la interfaz de acceso:

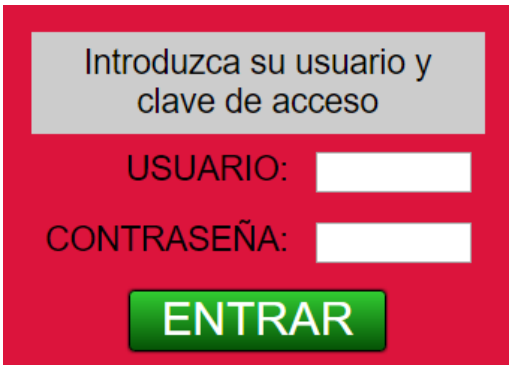


Figura AI.1: captura acceso a RaSpy

AI.2. Menú de navegación

Una vez dentro del sistema el menú de navegación (Figura AI.2) estará presente en todas las páginas y nos permitirá movernos de unas a otras.

Figura AI.2: captura menú de navegación

Los títulos asignados a los diferentes menús o secciones son bastante descriptivos. Así por ejemplo, el menú "Inicio" nos lleva a la página de inicio de la aplicación, desde ella controlaremos el sensor DHT-11 (temperatura y humedad) y el detector de gases. Además podremos comprobar el estado de los demás componentes del sistema.

Pinchando sobre el menú "Cámara" llegaremos a la página relacionada con la Raspberry PiCamera. Este menú contiene el submenú "Galería", desde donde podremos descargar las fotografías y vídeos capturados por la PiCamera.

Si pulsamos en el menú “Informes”, iremos a la web relacionada con la creación y gestión de los informes. Este menú contiene un submenú llamado “Informes Guardados” donde se podrán descargar los informes periódicos que se han generado automáticamente.

El menú “Alertas” nos envía a la página donde el usuario puede generar y enviar alertas a través de Telegram.

El menú “Control” nos envía a una página donde el usuario podrá controlar manualmente los distintos actuadores del sistema.

AI.3. Inicio

Esta es la página a la cual se nos dirige automáticamente una vez que nos identificamos correctamente ante el sistema. De alguna manera es la página principal del sitio web ya que ofrece una vista completa de los estados de todos los componentes que forman RaSPy. Su interfaz es la que se muestra en la Figura AI.3:

Figura AI.3: captura inicio

Como apreciamos en la Figura AI.3, la interfaz de “Inicio” tiene tres zonas bien diferenciadas.

La zona superior está dedicada a uno de los principales elementos del sistema: el sensor de temperatura y humedad. Como vemos, en la parte izquierda se encuentran el termómetro y el higrómetro que nos indican la temperatura y la humedad del habitáculo en tiempo real, siempre y cuando en sensor esté activado (podemos activarlo y desactivarlo desde esta misma zona). En la parte de la derecha se encuentra la zona de configuración del sensor, donde podremos introducir y modificar los valores de: temperatura máxima (a partir de la cual el sistema pondrá en marcha sus mecanismos de actuación), histéresis (este valor refleja los grados de margen hasta que el sistema vuelva a la normalidad, es decir, si la temperatura máxima son 25ºC, al alcanzarla se pondrá en marcha el refrigerador y de forma casi instantánea la temperatura descenderá de esos 25ºC y el sistema volverá a la normalidad apagando el refrigerador, pero la temperatura volverá a subir. Para evitar estas desconcertantes situaciones, se fija un valor de histéresis, por ejemplo 4ºC, de esta forma al alcanzar los 25ºC el refrigerador saltará y se mantendrá activado hasta que la temperatura baje de 21ºC) e intervalo de medida. La

configuración se hace efectiva al pulsar el botón “Activar DHT11” y el sistema nos recuerda la configuración vigente.

En la zona central el usuario podrá encender/apagar el detector de gases y comprobar su estado.

En la parte más baja se indica al usuario el estado de los distintos componentes del sistema que no manejamos desde esta página, consiguiendo que además de monitorizar la temperatura y la humedad del habitáculo, también monitorizamos el estado de todos los componentes del sistema, sabiendo cuáles están encendidos y cuáles apagados (en la Figura AI.3 estaría encendida la cámara y apagados el regulador de temperatura y la alarma).

AI.4. Cámara

Toda la funcionalidad relacionada con la Raspberry Pi Camera se lleva a cabo desde esta página. En la Figura AI.4 nos encontramos con su interfaz:

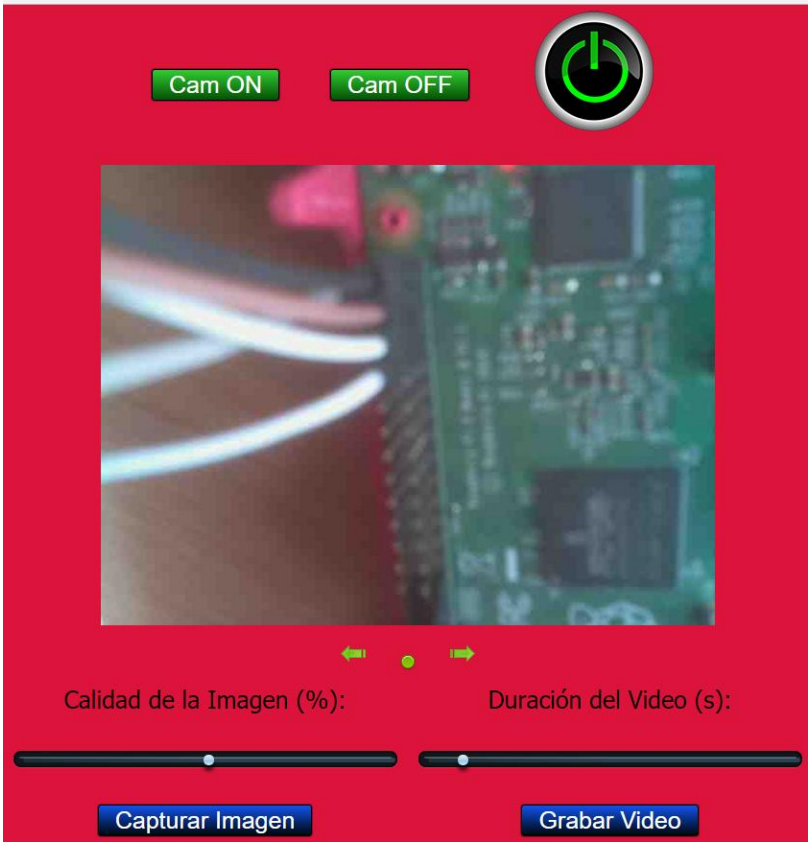


Figura AI.4: captura cámara

En la parte superior podremos encender y apagar la cámara pulsando en los botones “Cam ON” y “Cam OFF” respectivamente. También se puede comprobar su estado.

Inmediatamente debajo nos encontramos con la imagen en directo, siempre y cuando la cámara esté encendida.

Debajo de la imagen en directo nos encontramos con los botones que moverán la cámara hacia tres posiciones: izquierda, centro y derecha. Realmente lo que movemos es el servomotor sobre el cual está instalada la cámara.

En la parte inferior de esta interfaz nos encontramos con las operaciones que se pueden realizar con la cámara: sacar una foto y grabar un vídeo. Para realizar estas acciones deberemos configurar la calidad de la imagen (en una barra que muestra el porcentaje, siendo este de forma predeterminada el 50%) antes de pulsar sobre el botón "Capturar Imagen" si queremos sacar una fotografía, y deberemos ajustar la duración del vídeo (en segundos, de forma predeterminada 5s) antes de pulsar sobre el botón "Grabar Vídeo" si lo que pretendemos es grabar un vídeo.

Las imágenes y los videos serán almacenados en el sistema y el usuario podrá acceder a su descarga en el submenú "Galería". Para que se inicie la descarga es suficiente con pinchar sobre el archivo en cuestión y esta comenzará automáticamente. En la Figura AI.5 se muestra la interfaz de la galería de contenido multimedia, donde primero se presentan las imágenes y a continuación los vídeos.

Figura AI.5: captura galería

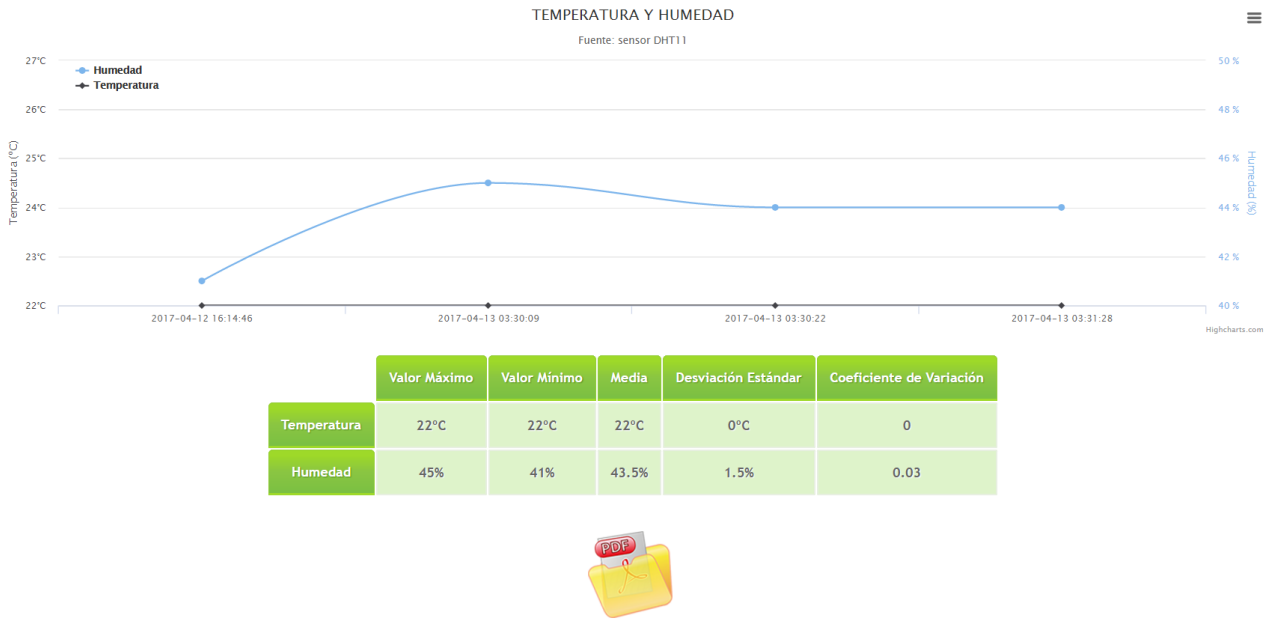
AI.5. Informes

En esta sección del menú (ver Figura AI.6) se recoge toda la funcionalidad relativa a la creación de informes de temperatura y humedad, empleando las muestras recogidas por el sensor DHT-11. Esta sección se divide en dos apartados, el primero de ellos proporciona al usuario las opciones de configuración necesarias para generar informes instantáneos que podrán ser vistos online y descargados. El segundo de los apartados simplemente pregunta al usuario la periodicidad con la cual se crearán informes de forma automática, esta periodicidad puede ser diaria, mensual y anual (en la captura de ejemplo de la Figura AI.6 se estarían generando de forma automática informes diarios y mensuales). Cuando el usuario presione el botón "Enviar" la configuración se hará efectiva (esta configuración de la periodicidad queda reflejada destacando las opciones vigentes para comodidad del usuario).

Figura AI.6: captura informes

Como hemos comentado, el primer apartado (situado en la parte superior de la interfaz) ofrecerá un pequeño formulario mediante el cual el usuario indicará los momentos entre los que desea generar un informe. El usuario deberá introducir la fecha de inicio, la hora de inicio, la fecha final y la hora final. Por lo tanto y siguiendo la captura de ejemplo de la Figura AI.6, en nuestro informe aparecerán todos los datos recogidos entre el 12 de abril de 2017 a las 15:30 y el 13 de abril de 2017 a las 22:30. Además podremos seleccionar la magnitud: temperatura y/o humedad.

Una vez finalizada la configuración, al pulsar sobre el botón “Generar Gráfica”, el sistema nos dirige a una página web que contiene el informe online generado a partir de los datos introducidos, como podemos ver en la Figura AI.7:



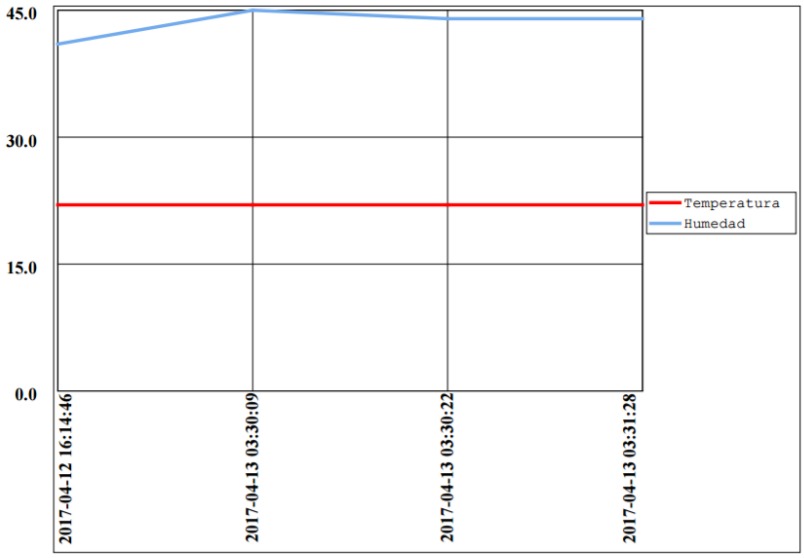
	Valor Máximo	Valor Mínimo	Media	Desviación Estándar	Coeficiente de Variación
Temperatura	22°C	22°C	22°C	0°C	0
Humedad	45%	41%	43.5%	1.5%	0.03

Figura AI.7: captura informe online

El primer elemento del informe que podemos ver en esta página es una gráfica que representa los valores de temperatura y/o humedad atendiendo a la configuración del usuario.

A continuación, podemos ver una tabla que proporciona algunos de los datos más significativos de la anterior gráfica: el valor máximo, el mínimo, la media, la desviación estándar y el coeficiente de variación.

Por último, se incluye en la página una imagen (parte inferior) que nos dirige a una versión PDF del informe lista para descargar (ver Figura AI.8).



	MÁXIMO	MÍNIMO	MEDIA	DESVIACIÓN	C.VARIACIÓN
TEMP. (ºC)	22	22	22	0	0
HUMEDAD (%)	45	41	43.5	1.5	0.03

Figura AI.8: captura informe PDF

El contenido de este informe es exactamente el mismo que el de la versión web, únicamente cambia la estética, más austera y apropiada para ser imprimida en la versión PDF del informe.

AI.6. Alertas

En esta sección del menú (ver Figura AI.9), el usuario podrá configurar una alerta y enviarla desde la propia aplicación web.

Figura AI.9: captura alertas

A través de un formulario, el usuario indicará el destinatario de la alerta (en nuestro ejemplo, el mensaje Telegram llegará al encargado de mantenimiento), el tipo de incidencia (en nuestro ejemplo será un error de cámara con código 06), la descripción de la incidencia (en nuestro ejemplo será el siguiente texto: "La cámara genera un ruido extraño.") y por último, el usuario se identificará como emisor de la alerta (en el ejemplo, Javier García). Una vez rellenado el formulario, para que se produzca el envío del mensaje de alerta a través de Telegram deberemos pulsar el botón "Enviar Alerta" y recibiremos el mensaje que se muestra en la Figura AI.10:

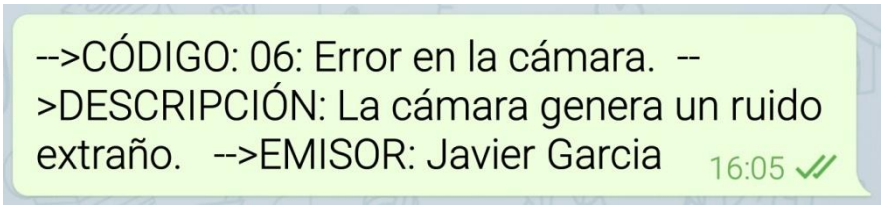


Figura AI.10: captura alerta Telegram

AI.7. Control

En esta sección (ver Figura AI.11) el usuario podrá manejar los distintos actuadores que se encuentran en el sistema de forma manual, es decir, podremos encender y apagar dichos actuadores de forma ajena a la toma de decisiones del sistema.

Para encender la alarma pulsaremos el botón “Alarma ON” y para apagarla “Alarma OFF”.

Para encender el regulador de temperatura hay que pulsar el botón “Regulador ON” y para apagarlo “Regulador OFF”.

Figura AI.11: captura control

AI.8. Cerrar sesión

Para cerrar sesión, a la derecha del menú de navegación encontraremos un botón (“Cerrar Sesión”) destinado para tal efecto. Una vez pulsado se cierra la sesión y nos guía a la página de la figura AI.12, donde se proporciona un enlace a la página de autentificación de la aplicación. Si el usuario quiere acceder nuevamente al sistema es necesario que se autentifique.

Figura AI.12: captura cerrar sesión

Anexo II: Contenido del soporte digital

A continuación se describe el contenido del CD-ROM entregado al concluir este Trabajo de Fin de Grado:

- Fichero **memoria.pdf**
- Directorio **codigoFuente**: contiene el código fuente de todo el software desarrollado a lo largo del TFG. Este directorio se encuentra ordenado en tres subdirectorios:
    - Directorio **codigoWeb**: contiene todo lo referente al sitio web implementado.
    - Directorio **codigoProgramas**: se divide en subdirectorios que hacen referencia a los distintos módulos, dentro de ellos se encuentran los programas implementados que llevan a cabo la funcionalidad.
    - Fichero **crontab**: dada la importancia que tiene en la automatización de tareas se incluye este fichero en una versión donde todas las automatizaciones posibles se encuentran activadas.

Bibliografía

[01] Fernando Alonso, Loïc Martínez, Fco. Javier Segovia. *Introducción a la Ingeniería del Software. Modelos de desarrollo de programas*. Delta Publicaciones, 2005.

[02] Ian Sommerville. *Ingeniería de Software*. 6ª Edición. Addison Wesley, 2002.

[03] Craig Larman. *UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Pearson Prentice Hall, 2003.

[04] Larry Ullman. *PHP: paso a paso*. Anaya Multimedia, 2009.

[05] Scott McCracken. *Curso de programación web con HTML5, CSS, JavaScript, PHP 5/6 y MySQL*. Inforbook's, 2011.

[06] PHP Official Website. [Internet] Disponible en: http://php.net/

[07] Fritzing Official Website. [Internet] Disponible en: http://fritzing.org/

[08] Código fuente MJPG-Streamer. [Internet] Disponible en: https://sourceforge.net/projects/mjpg-streamer/files/latest/download

[09] Librería RPi.GPIO. [Internet] Disponible en: http://downloads.sourceforge.net/project/raspberry-gpio-python/RPi.GPIO-0.5.4.tar.gz

[10] How to build and run MJPG-Streamer on the Raspberry Pi. [Internet] Disponible en: https://blog.miguelgrinberg.com/post/how-to-build-and-run-mjpg-streamer-on-the-raspberry-pi

[11] Código fuente Telegram. [Internet] Disponible en: https://github.com/vysheng/tg

[12] Datasheet micro servo 9g SG90 Tower Pro. [Internet] Disponible en: http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

[13] FPDF Library: PDF generator. [Internet] Disponible en: http://www.fpdf.org/

[14] Cron & crontab. [Internet] Disponible en: https://blog.desdelinux.net/cron-crontab-explicados/

[15] Highcharts. [Internet] Disponible en: https://www.highcharts.com/

[16] Google Charts: Chart Gallery. [Internet] Disponible en: https://developers.google.com/chart/interactive/docs/gallery

[17] LIBROSWEB. [Internet] Disponible en: http://librosweb.es/

[18] w3schools: the world's largest web developer site. [Internet] Disponible en: https://www.w3schools.com/

[19] Desarrolloweb. [Internet] Disponible en: https://desarrolloweb.com/