Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Automatización de casetas modulares para mascotas

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autora: Lerma Sánchez, Sara

Tutor: Bernabeu Soler, Enrique Jorge

Curso 2017/2018

Agradecimientos

A mi madre, mi hermana, Toni y Rosa por apoyar mis decisiones y motivarme hacia nuevos objetivos.

A mi tutor por resolver mis dudas y creer en el desarrollo de este proyecto desde el principio.

Y, sobre todo, a mi padre, que con su entusiasmo, dedicación y paciencia ha aportado siempre todo lo que ha estado en su mano tanto en este proyecto como en mi formación en general y gracias a su financiación se ha podido llevar a cabo la muestra del producto.

Resumen

En este proyecto se implementa un prototipo de caseta domótica para mascotas que realiza automáticamente los procesos involucrados en el cuidado de ésta. Este proyecto trata de un sistema de control basado en la Raspberry Pi, la cual gestiona las entradas de los sensores incorporados en la caseta, haciendo posible el control de la temperatura y de los niveles de agua y de comida del que dispone la mascota en los recipientes. A partir de los valores de entrada de los sensores, la Raspberry Pi, basándose en el código de programación, realiza el control y, a modo de respuesta, gestiona la activación de los dispositivos actuadores como son el ventilador, la manta eléctrica o las válvulas de comida y agua. El control de la aplicación se puede realizar desde cualquier lugar y desde cualquier dispositivo mediante el programa de acceso remoto VNC. El sistema, al ser modular, ya que es una caseta domótica, puede constar de una sola unidad o varias, es decir, para uso individual o colectivo, como puede ser el caso de protectoras o residencias de animales.

Palabras clave: automatización, programación, Raspberry Pi, caseta domótica, cuidado mascota, sistema modular.

Resum

En este projecte s'implementa un prototip de caseta domòtica per a mascotes que realitza automàticament els processos involucrats en els cuidats d'esta. Este projecte tracta d'un sistema de control basat en la Raspberry Pi, la qual gestiona les entrades dels sensors incorporats a la caseta, fent possible el control de la temperatura i dels nivells d'aigua i de menjar de què disposa la mascota als recipients. A partir dels valors d'entrada dels sensors, la Raspberry Pi, basant-se en el codi de programació, realitza el control i, a manera de resposta, gestiona l'activació dels dispositius actuadors com són el ventilador, la manta elèctrica o les vàlvules de menjar i aigua. El control de l'aplicació es pot realitzar des de qualsevol lloc i des de qualsevol dispositiu mitjançant el programa d'accés remot VNC. El sistema, al ser modular, ja que és una caseta domòtica, pot constar d'una sola unitat o diverses, és a dir, per a ús individual o col·lectiu, com pot ser el cas de protectores o residències d'animals.

Paraules clau: automatització, programació, Raspberry Pi, caseta domòtica, cuidats mascota, sistema modular.

Abstract

In this project has been implemented a prototype of a domotic doghouse for pets, which performs automatically the processes involved in the care of our pets. This project is a control system based on the Raspberry Pi, which manages the inputs of the sensors incorporated in the kennel, making possible the control of temperatura, water and food levels available in the pet's bowls. Based on the input values of the sensors, the Raspberry Pi, according to the programming code, performs control and, in response, carries out the activation of actuator devices such as the fan, electric blanket or valves of food and water. The control of the application can be done from any place and from any device through the remote access program VNC. The doghouse is a modular system that may consist of a single or several units. In other words, it can be for individual or collective use, as in the case of dog pounds or animal residences.

Key words: automation, programming, Raspberry Pi, domotic doghouse, pet care, modular system.

Índice general

Apéndices

Ficheros adjuntos a la memoria:

- Manual de usuario
- Vídeo de funcionamiento

Índice de figuras

Índice de cuadros

1. Introducción

1.1 Motivación

En nuestra sociedad las mascotas ocupan un lugar muy importante en la vida de las personas ya que proporcionan compañía, diversión, seguridad y fidelidad, entre otras cosas. Sin embargo, en muchos casos no reciben la atención que necesitan debido al estrés y a la vida rápida de sus dueños. Las mascotas muchas veces deben quedarse solas durante un periodo de tiempo ya sea porque sus dueños viajan, por problemas personales o por el trabajo y a pesar de que existan residencias caninas para ello, en muchos casos por tiempo de previsión o por motivos monetarios no es posible el empleo de estos servicios.

Actualmente, por desgracia, sigue habiendo un gran número de abandonos de mascotas que intentan ser resueltos por las protectoras, criaderos y residencias caninas. En estos lugares también tienen problemas como es el hacinamiento, ya que deben cobijar a un gran número de animales, lo que supone un alto coste de mantenimiento, la contratación de personal para el cuidado y el control de estos animales.

Ante esta situación, y teniendo como referente que la Comunidad Valenciana es la que mayor número de perros tiene por persona del territorio español, se ha querido solventar o facilitar la resolución de estas necesidades que se plantean en la sociedad con respecto a las mascotas y, más concretamente, con los perros. Por este motivo, se ha pensado en crear una “caseta domótica”, la cual ayuda tanto al dueño, deshaciéndose de algunas responsabilidades del día a día, como a la mascota cumpliendo sus necesidades. Por ello este proyecto realiza la automatización del cuidado de una mascota en la estancia donde ésta reside, concretamente el proyecto consta de una caseta modular en la que se incorpora un control para actuar sobre las necesidades básicas de la mascota como son la alimentación, la hidratación, el control de la temperatura y la vigilancia. Al tratarse de un sistema aplicado en una caseta modular puede constar de una o varias unidades, es decir, de una o de varias casetas las cuales pueden ser de gran ayuda en las protectoras o residencias de animales ya que facilitan el control y permiten el ahorro de tiempo y dinero.

1.2 Objetivos

El objetivo principal del presente Trabajo de Fin de Grado (TFG) es automatizar una caseta de mascotas para satisfacer las necesidades básicas y mejorar la comodidad del animal que reside en ella, así como facilitar las tareas de cuidado al dueño de éste. Para ello se equipara la caseta, por una parte de sensores de temperatura, humedad y nivel, los cuales informan de las condiciones climatológicas y de los niveles de agua y comida, y por otra parte de actuadores que ayudan a mantener un buen clima, así como unos niveles de agua y comida adecuados. Todo ello es controlado por un dispositivo, mediante una programación, que almacena y gestiona todos esos datos. Por lo tanto, se debe tener en cuenta los siguientes objetivos específicos a la hora de desarrollar el TFG:

- Informar de la temperatura y de la humedad relativa.
- Informar de la necesidad de reponer la alimentación en el depósito de comida.
- Controlar que la temperatura sea la adecuada mediante la lectura de un sensor de temperatura y mandar la orden correspondiente al actuador que se necesita para que suba o baje la temperatura según las necesidades.
- Controlar la alimentación mediante sensores de nivel que se programa con un horario según las necesidades.
- Controlar la hidratación mediante sensores de nivel que indican las necesidades según su consumo.
- Vigilar a la mascota en cualquier momento.
- Desarrollar un programa que controle los sensores, los actuadores y muestre la información.

Esta memoria está compuesta por múltiples apartados los cuales siguen un flujo de desarrollo. En primer lugar, se introduce la idea general de proyecto y se exponen otros productos existentes que realizan funcionalidades similares al presente proyecto. A continuación, se muestra la solución de hardware propuesta justificando el motivo de su elección y la solución de software propuesta. Posteriormente, se presenta el diseño de la aplicación y del conexionado del hardware, y seguidamente se muestra el desarrollo y la implementación del proyecto. Finalmente se exponen las pruebas realizadas, las conclusiones y los trabajos futuros que se podrían realizar a partir de este proyecto.

2. Estado del arte

Actualmente existen varias casetas para mascotas automatizadas que realizan funciones similares a las que se desarrollan en este proyecto. Entre ellas se destacan la caseta Dog Parker que es la caseta que más funcionalidades parecidas tiene a la caseta desarrollada en este proyecto y la caseta Samsung Dream Doghouse la cual dispone de dos funcionalidades parecidas a las desarrolladas en el proyecto, pero ninguna de estos dos productos están destinados al mismo objetivo. A continuación, se exponen ambas casetas y se explican sus funcionalidades.

Dog Parker

Es una caseta inteligente de perros [1] ubicada en algunas tiendas de Estados Unidos donde no está permitida la entrada de perros. Esta caseta permite dejar a la mascota durante el tiempo que el dueño permanezca en la tienda. La casa Dog Parker cuenta con un dispensador de comida y bebida, un pequeño regulador de temperatura, una cama y luces ultravioletas para acabar con virus y bacterias. Además, mediante la aplicación se puede observar cómo se encuentra tu mascota y regular la temperatura. El precio de Dog Parker es de 25 dólares al año, además de 0,20 centavos por cada minuto que el perro pase dentro de esta caseta, los cuales tiene que pagar el dueño del perro.

Figura 1.1 Caseta Dog Parker

Samsung Dream Doghouse

Se trata de una caseta inteligente [2] que se divide en tres áreas; el comedor, la zona de entretenimiento y el espacio de descanso, con césped artificial, que ha sido creada por Samsung. Esta caseta incluye un dispensador de comida, el cual debe ser accionado por tu mascota pulsando un botón cada vez que tenga hambre, una tableta, un andador y una bañera de hidromasaje. Ha sido desarrollada por 12 diseñadores y un arquitecto, por lo que el precio de la caseta no baja de 30.500 dólares. Sin embargo, la compañía no tiene, por el momento, intención de sacarla a la venta, sino más bien ha sido una estrategia de marketing para el patrocinio de Samsung en una exposición canina.

Figura 1.2 Caseta Dream Doghouse

2.1 Crítica al estado del arte

Las casetas Dog Parker y Samsung Dream Doghouse son casetas que no están destinadas a su ubicación en una casa. Ninguna de las dos casetas permite establecer los horarios en los que se proporcionará comida a la mascota ni a la temperatura en la que se activará el ventilador o la manta eléctrica lo cual puede variar según el tipo de perro, su pelaje y su raza. La caseta desarrollada por Samsung solamente puede ser utilizada por perros de pequeño tamaño y, a pesar de que no se encuentra en el mercado, tiene un precio desorbitado en comparación con la caseta desarrollada en este proyecto.

2.2 Propuesta

La propuesta que se presenta trata de una caseta inteligente de mascotas creada con el objetivo de satisfacer las necesidades básicas y mejorar la comodidad de la mascota que reside en ella. La caseta desarrollada en este proyecto se muestra en la Figura 1.3 y dispone de un bebedero que se rellena de agua automáticamente cuando el nivel es bajo, de un comedero en el que se depositará comida en las horas que establezca el dueño, de un ventilador que se activará cuando la temperatura alcance o exceda la temperatura establecida y de una manta eléctrica la cual se activará cuando la temperatura sea menor o igual a la establecida previamente por el dueño. Además de estos modos automáticos, en cualquier momento el dueño puede activar y desactivar el paso del agua o de la comida, el ventilador y la manta eléctrica. El usuario puede observar la temperatura y la humedad actual, así como modificar la configuración de la caseta mediante la pantalla táctil que incorpora la caseta. A diferencia de las casetas presentadas en el anterior apartado, Dog Parker y Samsung Dream Doghouse, la caseta desarrollada a en este proyecto está destinada para su ubicación en una casa, la comida que se proporciona a la mascota es controlada, se puede observar la temperatura y la humedad del momento y es apta para cualquier tamaño de perro.

Figura 1.3 Caseta Domótica DomoDog

3. Análisis del problema

En este apartado se van a exponer los requisitos que se deben tener en cuenta para el desarrollo del producto, la identificación y el análisis de las posibles soluciones y la solución propuesta.

3.1 Especificación de requisitos

Los requisitos que se deben cumplir son los especificados a continuación:

- La mascota debe disponer de agua en todo momento.
- La mascota debe ser alimentada mínimo en dos momentos del día.
- La mascota debe estar en un clima adecuado.
- La mascota debe disponer de un lugar en el que residir.
- El usuario debe poder vigilar a su mascota mediante una cámara en cualquier momento.
- El usuario debe poder configurar los parámetros de la caseta mediante una aplicación.

3.2 Identificación y análisis de soluciones posibles

En este apartado se va a mostrar la solución en el ámbito de hardware que se ha elegido, con el objetivo del cumplimiento de los requisitos, comparándola con las posibles alternativas, determinando sus ventajas y sus desventajas y finalmente justificando el motivo de su elección. Concretamente se van a exponer todos los elementos hardware necesarios para cumplir los requisitos y de todos éstos se especificará una descripción, características, ventajas y desventajas y el motivo por el que se ha elegido dicho elemento y no otro. Con el objetivo de mostrar los elementos en un orden coherente, en primer lugar, se exponen los dispositivos que realizan las lecturas de la temperatura, de la humedad y de los diferentes niveles que se tratan. Posteriormente se explica, por un lado, los actuadores que son activados o desactivados según las condiciones y

el modo de funcionamiento y, por otro lado, los periféricos. Por último, se comenta el ordenador que controla todo el funcionamiento de la caseta.

3.2.1 Sensores

Un sensor es un dispositivo eléctrico de entrada que se encuentra en contacto directo con la magnitud física o química que se va a evaluar y al interactuar con éstas sufre cambios en sus propiedades y transforma la señal física o química, mediante un transductor, en variables eléctricas logrando la conversión de una señal no interpretable por el sistema, en otra variable interpretable por dicho sistema. Estas magnitudes físicas o químicas son llamadas variables de instrumentación y por ejemplo puede ser: distancia, humedad, inclinación, temperatura, fuerza, etc.

3.2.1.1 Sensor de humedad y temperatura

Para el cumplimiento del requisito de que la mascota debe estar en un clima adecuado, son necesarios varios elementos hardware. Por una parte, se debe obtener la temperatura y por otra se debe actuar en consecuencia. Respecto a la obtención de los grados, se utiliza un sensor que indica la temperatura y la humedad actual que hay en el ambiente.

I. Descripción

Un sensor de temperatura es un dispositivo que está midiendo constantemente una variable física, en este caso la temperatura, y la transforma en una señal eléctrica la cual es procesada por equipo eléctrico o electrónico. Mientras que un sensor de humedad relativa sirve para medir la humedad relativa en el ambiente, es decir, la cantidad de concentración de agua que hay en el ambiente. Concretamente, el sensor de humedad mide o detecta variables químicas o físicas que determinan el grado de humedad.

El DHT11 es un sensor digital de temperatura y de humedad el cual permite medir simultáneamente los grados y la humedad relativa proporcionando una salida de señal digital calibrada de ambas variables. Cabe destacar que se trata de un sensor activo, por lo que necesita potencia de alimentación externa, la cual puede ser suministrada por la Raspberry Pi. Para realizar la medición, el DHT11 utiliza un sensor de humedad capacitivo y un sensor de temperatura por resistencia, conectados a un microcontrolador interno de 8 bits el cual realiza la conversión de analógico a digital y envía una señal digital con la temperatura y la humedad.

Figura 2.1 Sensor DHT11

Respecto a la medición, se muestran los datos mediante una señal digital mediante el pin de datos. Al tratarse de un sensor digital emite una cadena de bits que sigue un patrón que hace posible obtener la información necesaria. En el caso del DHT11 [4], emite 40 bits de información, el cual se estructura en conjuntos de 8 bits, que es la representación de un número entero en el sistema binario. El primer conjunto de 8 bits corresponde a la parte entera de la humedad, el siguiente conjunto de 8 bits a la parte decimal de la humedad, mientras que los siguientes 8 bits pertenecen a la parte entera de la temperatura y los próximos 8 bits a la parte decimal de la temperatura. Finalmente, los últimos 8 bits hasta completar los 40 bits son de *checksum* que permiten detectar errores en la transmisión. El porcentaje de errores en la transmisión es significativo en este sensor por lo que es posible que haya que realizar varias lecturas del sensor para obtener una correcta. Este sensor tiene una frecuencia de refresco de dos segundos, es decir, por lo que las lecturas que se pueden realizar serán mínimo cada dos segundos.

Para el correcto funcionamiento del sensor es necesario conectar una resistencia de 4’7 KΩ a 10 KΩ en el pin VCC. Sin embargo, en el mercado se encuentra el sensor DHT11 soldado a una placa que tiene incorporada la resistencia.

II. Características técnicas [3]

- Tensión de alimentación: 3.5～5.5V CC
- Corriente de alimentación: 0.3mA típica, 60μA en espera
- Periodo de muestreo: más de dos segundos
- Calibrado directamente en Celsius (centígrados)
- Rango de temperatura: 0ºC a 50ºC
- Precisión de temperatura: ± 2%
- Rango humedad: 20% a 90% de humedad relativa (HR%)
- Precisión de HR %: ± 5%

III. Ventajas y desventajas

Ventajas

- En un mismo integrado suministra la salida con la humedad relativa y a la temperatura.
- Buena precisión.
- Salida de tensión lineal.
- Bajo consumo de energía.
- No requiere de circuitos adicionales para calibrarlo externamente.
- Económico.

Desventajas

- No admite condensación .
- Si se excede el voltaje de suministro se rompe el sensor.
- No capta temperaturas menores de 0ºC.
- El rango de humedad relativa se limita entre 20 y 90 %, lo ideal sería que fuese de 0-100%.

IV. Motivo de elección

Durante el proceso de selección del sensor se han considerado las siguientes características; su precio, su tamaño y que su salida no sea analógica. Las alternativas a la solución propuesta son las siguientes;

Temperatura

LM35 DZ es un sensor de temperatura analógico que proporciona una tensión de salida de 10mV por cada grado Celsius, es proporcional a la temperatura del dispositivo (voltaje de salida 200, quiere decir que la temperatura es de 20 grados centígrados). Al igual que el sensor DHT11, se trata de un sensor activo, por lo que necesita potencia de alimentación. El principal motivo por el que no se ha elegido este sensor es que, como se ha comentado anteriormente, se trata de un sensor de temperatura analógico y las Entradas/Salidas de Propósito General (GPIO) de la Raspberry Pi son digitales, por lo que, si se usara este sensor, sería necesario construir un circuito Convertidor de señal Analógica a Digital (ADC) con varios pines GPIO.

Humedad relativa

Existen distintos tipos de sensores de HR%, dependiendo del principio físico que siguen para calcular la HR% (capacitivos, resistivos, tensión, etc.). El hih-4000-001 se trata de un sensor capacitivo activo, por lo que necesita una alimentación aproximadamente de 5V. Proporciona una tensión de salida lineal, una humedad en un rango de humedad relativa (HR%) 0 a 100% con una precisión de 3’5%. Entre sus desventajas destacamos la sensibilidad al contacto y a la luz, así como la no admisión de condensación. Sin embargo, el principal motivo por el que no se ha elegido este sensor es porque, al igual que el sensor LM35 DZ, proporciona una salida analógica y, como se ha comentado anteriormente, las entradas GPIO de la Raspberry Pi son digitales.

Por lo tanto, se ha seleccionado el sensor DHT11, ya que permite la obtención digital de las dos variables deseadas, la temperatura y la humedad relativa, en una misma medición y con un mismo componente y conexionado.

3.2.1.2 Detector de nivel de líquidos

Para el cumplimiento del requisito de que la mascota debe disponer de agua en todo momento son necesarios varios elementos hardware. Por una parte, se debe obtener el nivel de agua del que dispone la mascota y, por otra parte, se debe actuar en consecuencia. Respecto a la obtención del nivel de líquido, se utilizan dos sensores de nivel que indique el nivel de agua que hay en el bebedero.

I. Descripción

El detector de nivel se trata de un dispositivo electrónico que detecta la presencia del líquido dentro de un tanque u otro recipiente, excepto de material metálico. El sensor XKC-Y25-V se muestra en la Figura 2.2 y es un detector de nivel sin contacto que puede detectar líquidos. Puede ser ubicado en una superficie (de espesor hasta 12mm) no metálica y no tiene la necesidad de entrar en contacto directo con el líquido para detectar la presencia de éste. Este detector incorpora un potenciómetro que permite ajustar el punto de referencia y la sensibilidad, de manera que, si se gira el tornillo del potenciómetro hacia la izquierda, aumenta la sensibilidad; de lo contrario, disminuye la sensibilidad.

Figura 2.2 Sensor de nivel de agua

El funcionamiento del detector de nivel XKC-Y25-V es similar al de un condensador. En cuanto al detector capacitivo, el material que presenta baja conductividad eléctrica es el aire, por lo que el valor de la capacitancia es muy bajo. En el momento en el que se aproxima un líquido, el valor

de capacitancia aumenta, por lo que el circuito integrado detecta esta variación e indica presencia de líquido, mediante emisión de una señal digital de nivel alto. En caso contrario, si no se antepone ningún líquido, no aumenta la capacitancia, por lo que no se activa la señal e indica nivel bajo. Cabe destacar que el detector dispone de un LED rojo el cual indica la presencia del líquido, de manera que si detecta presencia se enciende el LED, y en caso contrario se apaga.

II. Características técnicas [28]

- Tensión de alimentación: 5 ~ 24V CC
- Consumo de energía: 5mA
- Voltaje de salida (nivel alto): 5V CC
- Voltaje de salida (nivel bajo): 0V CC
- Corriente de salida: 1-50 mA
- Humedad: 5% -100%
- Tiempo de respuesta : 500ms
- Temperatura ambiente de funcionamiento: 0 ~ 105 Celsius

III. Ventajas y desventajas

Ventajas

- Detección precisa y estable.
- Rendimiento estable y larga vida.
- Alta estabilidad, alta sensibilidad, fuerte capacidad anti-interferencia.
- Sin contacto, lo que evita la contaminación con el líquido que se está monitoreando, como ejemplo el agua que ingiere la mascota.
- Punto de referencia ajustable.
- Económico.

Desventajas

- No es aplicable en superficie metálica.
- No detecta sólidos.
- No es hermético.

IV. Motivo de elección

Durante el proceso de selección del sensor se han considerado las siguientes características; que no sea necesario el contacto con el líquido para su detección, su precio, su tamaño y que su salida no sea analógica. Respecto a la característica de un sensor de nivel sin contacto, hace referencia a que sea externo, que no se deba de meter dentro del agua para que pueda calcular el nivel. Las alternativas a la solución propuesta son las siguientes;

Interruptores de flotador (boya)

Es un sensor de nivel [5] que permite determinar si el nivel que líquido almacenado alcanza o excede un nivel predeterminado mediante un flotador magnético el cual se mueve en la superficie del líquido y al ser movido por el líquido alcanza una inclinación que activa el microswitch interno. El movimiento del flotador se transmite por inducción magnética, por lo que no hay posibilidad de que se filtre líquido entre el recipiente y el interior del indicador de nivel. El principal motivo por el que no se ha elegido este sensor es por la peligrosidad que supone para la mascota tener un elemento flotante en su recipiente de bebida.

Sensor de nivel ultrasónico sin contacto

Este tipo de sensor incorpora, entre otras cosas, un procesador de señal analógica, un microprocesador, y un circuito de salida del controlador. El funcionamiento [6] es el siguiente; en primer lugar se envía un haz ultrasónico a la superficie del líquido, este haz rebota sobre la superficie, entonces el sensor de nivel recibe el eco de la superficie y envía dichos datos al microprocesador para una representación digital de la distancia entre el sensor y el nivel de la superficie. De esta forma, a través de una actualización constante de las señales que recibe el microprocesador, éste calcula los valores promedios y de esta manera se consigue medir el nivel de líquido. El motivo por el que no se ha elegido este tipo de sensor es por la posibilidad de que la mascota corte el haz ultrasónico, por lo que los valores recogidos no serían ciertos. Para incorporar dicho sensor es necesario que la zona esté despejada.

3.2.1.3 Detector de nivel de sólidos

Para el cumplimiento del requisito de que la mascota debe ser alimentada mínimo en dos momentos del día son necesarios varios elementos hardware, por una parte, se debe obtener el nivel de comida del que dispone la mascota en el comedero y si hay alimento suficiente en el depósito de comida, y por otra parte se debe actuar en consecuencia. Respecto a la obtención del

nivel de alimento en el comedero se pueden utilizar dos sensores de nivel que indique el nivel de pienso que hay disponible en el comedero, mientras que para saber si se dispone de comida suficiente en el depósito de alimento se puede incorporar un sensor de nivel de comida.

I. Descripción

El detector de nivel para sólidos es un dispositivo electrónico que detecta la presencia de un objeto dentro de un recipiente. Se ha utilizado como detector de presencia de comida un sensor de proximidad capacitivo de la marca DIANQI, se observa en la Figura 2.3. Este sensor es digital, por lo que cuando detecta objetos en una corta distancia emite un nivel alto, mientras que si no detecta ningún objeto su salida se encuentra en nivel bajo. Al igual que el sensor de nivel de agua, este sensor es regulable dado que incorpora un potenciómetro con el que permite ajustar el punto de referencia y la sensibilidad, de manera que, si se gira el tornillo del potenciómetro hacia la izquierda, aumenta la sensibilidad; de lo contrario, disminuye la sensibilidad. Además, para realizar la calibración y comprobar su correcto funcionamiento, dispone de un LED rojo que se activa cuando el sensor detecta un objeto.

Figura 2.3 Sensor de nivel de comida

El funcionamiento del sensor de proximidad capacitivo DIANQUI se basa en que el sensor genera un campo electrostático en el que, si un objeto entra en dicho campo, causa interferencia. Por lo que cuando se detecta que ha habido un cambio en el campo, se emite un nivel alto en el circuito de salida y hasta que no se detecte que deja de haber un objeto en el campo electrostático no se emite un nivel bajo.

II. Características técnicas [29]

- Tensión de alimentación: 6 - 36V CC
- Corriente de salida: 300mA
- Rango de detección: 1mm a 10mm
- Dimensiones: Diámetro 18mm y longitud 70mm
- Temperatura ambiente de funcionamiento -30 a 65°C

III. Ventajas y desventajas

Ventajas

- Hermético, por lo que no sufre ningún riesgo en el caso en el que tuviera contacto con la comida de la mascota.
- Rendimiento estable y larga vida.
- Punto de referencia ajustable.
- Económico.

Desventajas

- Para su colocación se debe de agujerear el recipiente.

IV. Motivo de elección

Durante el proceso de selección del sensor se han considerado las siguientes características; hermético, estable, que el sensor esté formado por una sola pieza, su precio, su tamaño, que su salida sea digital.

Sensor de proximidad ultrasónico

Es un tipo de sensor que permite detectar objetos que se ubican a una distancia desde centímetros hasta varios metros del sensor. Respecto al funcionamiento de este sensor, emite un sonido, el cual es reflejado en un objeto, el receptor del sensor recibe el eco producido por el objeto y lo convierte en señales eléctricas. El tiempo que tarda el sonido en ir hasta el objeto y volver es el que se utiliza para calcular la proximidad del objeto al sensor. Los principales motivos por los que no se ha seleccionado este tipo de sensor es que es menos hermético que el sensor de proximidad capacitivo, y porque se aconseja que objeto esté en perpendicular al sensor, y que la zona esté despejada, lo cual no se puede garantizar.

3.2.2 Actuadores

Un actuador es un dispositivo electrónico que se encarga de transformar una señal eléctrica en una variable física (luz, sonido, etc.). Los actuadores que se van a presentar en este proyecto son una válvula de comida, una válvula de agua, un ventilador y una manta eléctrica para mascotas. Antes de comenzar con la explicación de dichos actuadores, es necesario mencionar cómo se va a hacer posible la activación y la desactivación de dichos actuadores.

Placa de relés

La tensión de salida de la placa que se utiliza en el proyecto, Raspberry Pi, posteriormente explicada, es de 5 voltios. Con esta tensión se pueden alimentar LEDs y otros elementos que necesitan poco voltaje, sin embargo, los actuadores utilizados en este proyecto necesitan un mayor voltaje para su excitación. Por ello se va a emplear una placa que consta de un bloque de cuatro relés, gobernada desde la Raspberry Pi, para manejar la alimentación de los actuadores que funcionan a 220V CA. Los relés integrados en la placa soportan una intensidad de hasta 10 Amperios a 250V. Ninguno de los actuadores que se van a utilizar superan esta intensidad.

La placa de relés por la que se ha optado es la que se muestra en la Figura 2.4 que trata de un módulo de cuatro relés compatible con la Raspberry Pi, dado que funciona con alimentación de 5V. La función de este dispositivo es cerrar o abrir el circuito a parir de una señal desde la Raspberry Pi.

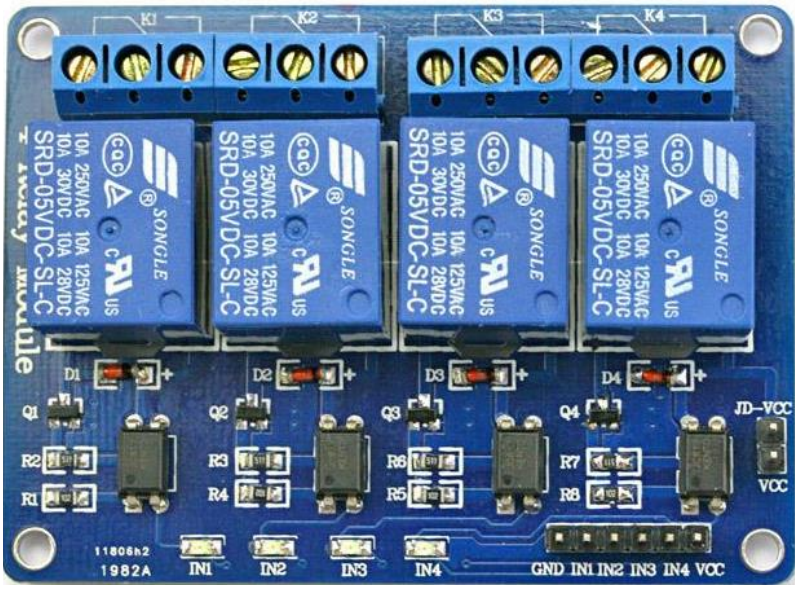


Figura 2.3 Placa de relés

3.2.1.4 Válvula de agua

Como se ha comentado anteriormente, para el cumplimiento del requisito de que la mascota debe disponer de agua en todo momento son necesarios varios elementos hardware, por una parte, se debe obtener el nivel de líquido del que dispone la mascota y por otra parte se debe actuar en consecuencia. Una vez obtenido el nivel de líquido, en el caso en el que dicho nivel sea bajo, se debe proporcionar agua, mediante una válvula de líquidos que permita el paso del fluido desde la toma de agua hasta el bebedero de la mascota.

I. Descripción

La válvula de agua es un instrumento encargado de controlar la cantidad de fluido que pasa a través de ella. Las tres funciones [7] más comunes de una válvula y el nombre que adoptan por su funcionamiento son las siguientes;

1. Detener e iniciar el flujo. Estas válvulas son llamadas **Stop / Start** y son utilizadas en sistemas que no necesitan el flujo regulado. La válvula se abre para permitir el paso del flujo y se cierra para detener el paso del flujo.
2. Regular el flujo, de ello se ocupan las válvulas **reguladoras** o de control las cuales controlan la velocidad y la cantidad de flujo que pasa a través del sistema.
3. Actuar como comprobación de no retorno del flujo, esta función la lleva a cabo las válvulas de **no retorno** o de retención controlan la dirección del flujo. El flujo en la dirección deseada abre la válvula, mientras que el flujo en la dirección opuesta fuerza a la válvula a cerrarse.

Para controlar el flujo del líquido las válvulas usan dos medios diferentes, las válvulas de **movimiento lineal** de forma que un elemento cierra o abre el paso del fluido o las válvulas de **movimiento giratorio** las cuales permiten regular el paso de mayor o menor cantidad de líquido.

La válvula de agua que se utiliza en este proyecto es una válvula de la marca KHAN, se trata de una válvula Stop/Start de movimiento lineal, ya que con ella se desea controlar el paso o detención del agua que fluye desde la toma de agua del lugar hasta el bebedero de la mascota. En la Figura 2.4 se puede observar la válvula elegida de planta y de perfil, en la imagen del perfil se puede apreciar que dispone de una membrana la cual solo deja el paso de líquidos.

Figura 2.4 Válvula de agua

Se trata de una válvula solenoide, lo que significa que es automática por lo que no es necesario que sea manipulada manualmente. Las válvulas solenoides [8] también son llamadas válvulas electromagnéticas o electroválvulas dado que son operadas magnéticamente ya que el solenoide es una bobina de alambre que produce un campo magnético cuando se le aplica energía, y este campo magnético es usado para mover el émbolo que abre o cierra la válvula como se puede observar en la Figura 2.5.

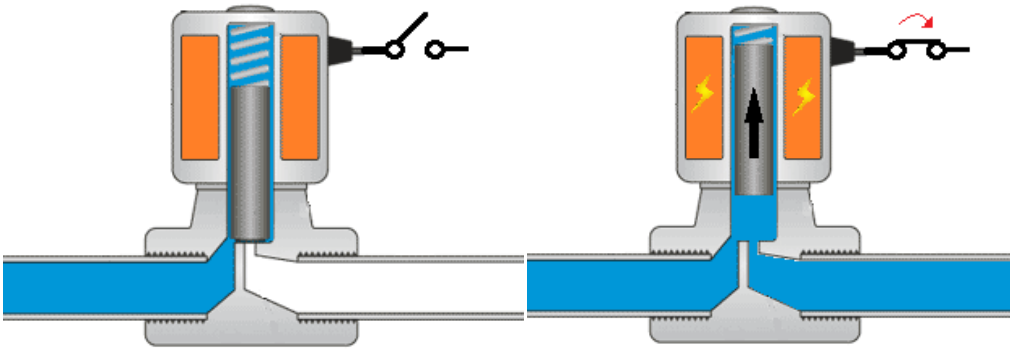


Figura 2.5 Funcionamiento válvula de agua

Una posible clasificación de las válvulas según su estado es Normalmente Cerrado (NC) o Normalmente abierto (NO), las válvulas normalmente cerradas se cierran cuando están en reposo (sin energía), mientras que las válvulas normalmente abiertas están abiertas cuando están en reposo. Las válvulas usadas en el proyecto son normalmente cerradas dado que solamente se desea que estén abiertas cuando se le ordene, es decir, cuando se produzca la excitación de la bobina.

II. Características técnicas [30]

- Potencia: Solenoide
- Tensión de alimentación: 220V CC
- Entrada y salida: Manguera para 1/2 "(diámetro exterior) manguera
- Modo de operación: normalmente cerrado
- Temperatura máxima: 100 °C
- Uso: fluidos de agua y baja viscosidad

III. Ventajas y desventajas

Ventajas

- El tiempo de respuesta de la válvula puede ser tan corto como de unos pocos milisegundos.
- El consumo de energía de la bobina de la válvula solenoide correctamente diseñado es muy bajo, es un producto que ahorra energía.
- Económica en comparación con otro tipo de válvulas.

Desventajas

- No puede ser utilizada con sólidos, sólo puede utilizarse con líquidos.
- No permite la regulación de flujo que pasa por ella.
- Es importante que se aplique el voltaje adecuado dado que si se aplica menor voltaje del requerido no se abrirá ni cerrará y causará mucho ruido, mientras que si se aplica mayor voltaje del requerido generará demasiado calor y desgastará el solenoide.

IV. Motivo de elección

Las alternativas a la solución propuesta son las siguientes;

Válvula de compuerta

La válvula de compuerta [9] se trata de una válvula que cierra el orificio que conecta las dos vías mediante un disco en vertical que se desliza en ángulos rectos. El principal motivo por el que no se ha elegido este tipo de válvula es porque el caudal que se necesita es de media pulgada y este tipo de válvulas se utiliza para caudales mayores.

Válvula de globo

Una válvula de globo [9] es un tipo de válvula que permite la regulación de la cantidad de flujo que pasa por ella y realiza un cierre hermético. El funcionamiento de esta válvula se basa en cerrar el orificio por el que pasa el fluido con un tapón obturador. En esta clase de válvulas el fluido no corre de manera directa y en una sola dirección sino que el fluido entra y sube dentro del cuerpo de la válvula, es estrangulado por el émbolo según lo abierta o cerrada que se encuentre la válvula, y después baja el fluido hacia la salida de la válvula. Los motivos por los que no se ha elegido este tipo de válvula es porque se utiliza para caudales mayores del necesario y son menos económicas.

3.2.1.5 Válvula de comida

Como se ha comentado en el apartado de sensores, para el cumplimiento del requisito de que la mascota debe ser alimentada mínimo en dos momentos del día son necesarios varios elementos hardware, por una parte, se debe obtener el nivel de comida del que dispone la mascota y por otra parte se debe actuar en consecuencia. Una vez obtenido el nivel, lo que es equivalente a averiguar si dispone de comida, en el caso en el que no disponga de alimento y sea la hora especificada por el usuario, se debe proporcionar alimento para lo que se utiliza una válvula para sólidos que permite el paso del pienso desde el depósito de comida hasta el comedero de la mascota.

I. Descripción

Las válvulas de sólidos [10] se utilizan para controlar el flujo de polvo, granulado y otros materiales sólidos a granel a lo largo de una línea de proceso. También se usan para controlar o regular el flujo de sólidos hacia y desde las unidades de proceso, los contenedores de almacenamiento, etc.

La válvula de comida que se utiliza es la que se muestra en la Figura 2.6, se trata de una válvula de bola motorizada de la marca SAILFLO de dos vías con un diámetro de 3/4 de pulgadas para que pueda fluir perfectamente el pienso de la mascota. Al igual que la válvula de agua, se trata de una válvula Stop/Start, lo que permite iniciar y detener el flujo de pienso, de movimiento lineal de forma que se cierra o se abre el flujo, no se regula. Con ello se consigue controlar el paso o detención del pienso que fluye desde el depósito de comida, que se sitúa encima de la válvula, hasta el comedero de la mascota.

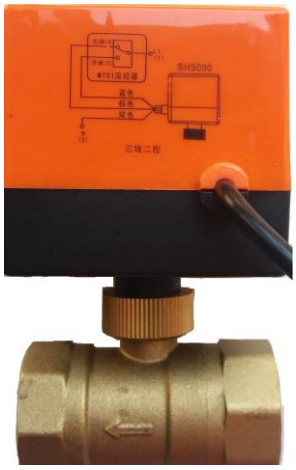


Figura 2.6 Válvula de comida

Respecto al funcionamiento de la válvula, se trata de una válvula de bola la cual utiliza una bola para controlar el flujo del material desde una vía hasta otra vía. La bola de la válvula dispone de un agujero, como se observa en la Figura 2.7, de manera que cuando la válvula esté en posición abierta, el agujero estará alineado con la tubería o dos vías y cuando la válvula esté en posición cerrada, el agujero quedará de forma perpendicular a la tubería de manera que imposibilitará el paso del flujo.

Figura 2.7 Válvula de comida abierta y cerrada

II. Características técnicas [31]

- Tensión de alimentación: 220 V AC
- Diámetro: dn20 (3/4 ")
- Modo de operación: recomendable encendido/apagado
- Potencia: 6 W (sólo cuando las válvulas se abren y cierran)
- Tiempo de ejecución: 10-15 segundos
- Temperatura de funcionamiento: 2 ° ~ 95 °

III. Ventajas y desventajas

Ventajas

- Posee la suficiente fuerza para que ningún grano de pienso bloquee el giro de la bola.
- Poseen un cierre hermético.
- No necesitan mantenimiento constante, ni lubricación.
- Permiten que la circulación de los fluidos sea en línea recta.
- Pueden funcionar sin interrupciones, las 24 horas del día, sin riesgo de sobrecalentamiento.

Desventajas

- No es adecuado su uso en una posición parcialmente abierta para regular el flujo, opera mejor totalmente abierta o totalmente cerrada.
- Tarda aproximadamente unos 15 segundos en girar 90º para su cierre o apertura.

IV. Motivo de elección

Las alternativas a la solución propuesta son las siguientes;

Válvula de mariposa

Las válvulas de mariposa [9] controlan el flujo a través de un disco circular montado en un eje rotativo de manera que girando el eje de pivote de la válvula en ángulo recto con respecto a la dirección del flujo en la tubería se logra abrir la válvula. Por lo que cuando está completamente cerrada, el disco bloquea totalmente la línea y cuando está completamente abierta, el disco se encuentra paralelo al flujo. El principal motivo por el que no se ha elegido este tipo de válvulas es porque no sellan completamente como las válvulas de bola.

Válvula de iris

La válvula de iris [11] es una válvula que su funcionamiento se basa en el diafragma del iris, un mecanismo que puede expandir o disminuir la apertura de un orificio circular de manera que permite controlar el paso del flujo del material. Esta válvula consta de dos anillos conectados por una manga flexible que puede ser de diferentes materiales y un extremo del mango se encuentra fijo mientras que el otro extremo está unido a un anillo de control el cual es manipulado por un mango externo. Por lo tanto en la medida que el mango gira la manga se retuerce hasta convertirse en una barrera sólida y plana. Esta es una buena alternativa y en el caso en el que se necesitara regular el flujo sería la opción escogida, sin embargo este tipo de válvulas son más difíciles de encontrar en el mercado.

3.2.1.6 Manta eléctrica

Como se ha comentado anteriormente, para el cumplimiento del requisito de que la mascota debe estar en un clima adecuado eran necesarios varios elementos hardware, por una parte, es necesaria la obtención de la temperatura, la cual ya ha sido explicada, y, por otra parte, es necesaria la actuación en consecuencia de la temperatura. Por lo que para poder actuar sobre los grados a los que se encuentra la caseta se controla la subida y la bajada de la temperatura. Para ello se propone el presente apartado que muestra un aparato para calentar y el posterior apartado, en el que expone un dispositivo para refrescar.

I. Descripción

Una manta eléctrica para mascotas se trata de una manta con un aparato calefactor eléctrico que se utiliza para mantener a la mascota caliente. La manta eléctrica seleccionada es la manta térmica para mascotas Comfy [12] que se muestra en la Figura 2.28, en la cual se puede observar que no dispone de un mando mediante el cual se puede regular la cantidad de calor que proporciona la manta, por lo que dispone de un mecanismo de encendido o apagado.

Figura 2.8 Manta eléctrica

Una vez enchufada la manta eléctrica a la red la temperatura comenzará a subir hasta conseguir una temperatura agradable en la superficie, de forma que se conseguirá una temperatura óptima cuando la mascota se tumbe sobre ella y con su peso corporal se logra una temperatura máxima de 40-45ºC. Cabe destacar que el cable está protegido hasta el empalme con el adaptador para evitar que la mascota lo deteriore.

II. Características técnicas [12]

- Temperatura en contacto con el cuerpo: máx. 40 - 45°C.
- Temperatura de la superficie: aprox. 30ºC.
- Con funda retirable y cierre con cremallera.
- Tensión de alimentación: 100-240 V, 50/60 Hz, 0,3 A.

III. Ventajas y desventajas

Ventajas

- Menor gasto eléctrico respecto a un calefactor.
- Menor gasto económico respecto a un calefactor.

Desventajas

- Cada cierto tiempo se debe limpiar la funda.

IV.Motivo de elección

Las alternativas a la solución propuesta son las siguientes;

Calefactor

Un calefactor o una estufa se trata de un dispositivo eléctrico que proporciona aire caliente en una estancia. Los motivos por los que no se ha elegido un calefactor es porque se considera más peligroso para la mascota que una manta eléctrica dado que se debería ubicar dentro de la caseta y porque consume más que una manta eléctrica, dado que el aire que emite es instantáneo y dura poco tiempo en la caseta debido a que no dispone de puerta.

Manta eléctrica con varios funcionamientos

Se ha elegido manta eléctrica para perros ya que éstas disponen de un sistema no regulable, es decir, que no proporciona diferentes modos de funcionamiento como por ejemplo, bajo, medio y alto, sino que solo se puede encender y apagar. Esto se debe a que la programación no se ha realizado para mantas eléctricas que dispongan de 3 o más modos de funcionamiento, solamente se puede configurar encender y apagar.

3.2.1.7 Ventilador

I. Descripción

Un ventilador es una máquina eléctrica con aspas que giran rápidamente y es usada para crear un flujo de aire logrando de esta manera la ventilación de ambientes, el refrescamiento de máquinas u objetos, etc.

El ventilador utilizado en este proyecto es un ventilador de cuadro eléctrico BLS12/92 que se muestra en la Figura 2.9 el cual permite la ventilación de la caseta de la mascota. Cabe destacar que el ventilador no enfría el aire. La función del ventilador es renovar el aire de la caseta.

Figura 2.9 Ventilador

II. Características técnicas [32]

- Dimensiones (A x A x B): 92 x 92 x 25 mm
- Caudal de aire: 74m³/h
- Corriente: 0.16A
- Velocidad: 2400 rpm
- Temperatura de operación: de -20°C a +40°C

III. Ventajas y desventajas

Ventajas

- Menor gasto eléctrico respecto a un aire acondicionado.
- Menor gasto económico respecto a un aire acondicionado.

Desventajas

- El ventilador no enfría el aire.

IV. Motivo de elección

Las alternativas a la solución propuesta son las siguientes;

Aire acondicionado portátil

Los aires acondicionados portátiles son aparatos que emiten aire frío sin instalación, por lo que permite enfriar una estancia no demasiado grande. Estos aparatos suelen tener un tamaño de 50 cm. de alto y 30 cm. de ancho, lo cual se considera grande para ser introducido en una caseta de perro debido a que ocuparía mucho espacio.

Ventilador para transportines y jaulas para mascotas

El ventilador para transportines de mascotas existente en el mercado se trata de un ventilador pequeño que es situado en la jaula de la mascota de manera que proporciona una corriente continua de aire renovado. El motivo por el que no se ha elegido dicho ventilador es porque funciona con pilas y no podría ser controlado mediante la programación.

3.2.3 Periféricos

Para que el usuario pueda configurar los parámetros de la caseta es necesario que interactúe con la aplicación desarrollada, y para interactuar con ella se puede incorporar en la caseta una pantalla táctil. Mediante la pantalla táctil el usuario puede observar la temperatura y la humedad relativa, activar o desactivar cualquier actuador, configurar los horarios de comida de la mascota, las temperaturas de consigna para las que se encenderán la manta eléctrica o el ventilador y vigilar a la mascota en cualquier momento. Respecto a la vigilancia de la mascota, se puede introducir una cámara de visión nocturna que permita visualizar el interior de la caseta en cualquier momento.

Un periférico [13] es un dispositivo auxiliar conectado a la Unidad Central de Procesamiento (CPU) de un ordenador utilizado para la conexión del ordenador con el exterior, es decir, para introducir información u obtener información del ordenador. Los dispositivos periféricos agregan funcionalidad al ordenador pero no son componentes que contribuyen a la función principal de la computadora, como son por ejemplo la CPU, la placa base y la fuente de alimentación, sino que ayudan a los usuarios finales a acceder y usar las funcionalidades de un ordenador. Existen

diferentes formas de categorizar los periféricos, una manera es dependiendo de la relación que tienen con el ordenador, donde existen tres tipos de periféricos:

1. **Periféricos de entrada**: este tipo de dispositivos son los que permiten introducir datos externos al ordenador para su posterior tratamiento por parte de la CPU, es decir, captan y, si es necesario, digitalizan los datos los cuales pueden provenir de distintas fuentes, siendo la principal el usuario u otro dispositivo y los envían al ordenador para ser procesados. Ejemplos de este tipo de dispositivos son el teclado, el ratón, cámara, micrófono, etc.
2. **Periféricos de salida**: son dispositivos que reciben información procesada por la CPU y muestran o proporcionan una salida de información hacia el exterior del ordenador de modo que sea perceptible por el usuario. La mayoría de estos periféricos son para informar, proyectar o proporcionar al usuario cierta información. Los periféricos de salida más utilizados son el monitor, el LED, el altavoz y la impresora.
3. **Periféricos de entrada/salida (E/S)**: se tratan de dispositivos que sirven para la comunicación entre el ordenador y el medio externo. Proporcionan la capacidad de introducir información en el ordenador, y viceversa, es decir, proporcionar información al exterior para realizar alguna tarea en un dispositivo determinado. Un claro ejemplo de este tipo de dispositivos es una pantalla táctil, la cual muestra información al usuario y éste puede introducir información al ordenador a través de la pantalla táctil.

Los periféricos que de los que se hacen uso en este proyecto son una pantalla táctil y una cámara, los cuales serán comentados a continuación.

3.2.3.1 Pantalla táctil

I. Descripción

Una pantalla táctil es un periférico de entrada y de salida de datos. Permite a un usuario la entrada de datos o el control del sistema mediante el contacto con la superficie de la pantalla ya sea con un lápiz especial o con los dedos y a su vez muestra los resultados introducidos previamente. Por lo tanto, la pantalla táctil permite al usuario interactuar directamente con lo que se muestra en lugar de utilizar un ratón, además se puede configurar un teclado virtual de manera que no sea necesario un teclado físico.

La pantalla elegida en este proyecto es la que se muestra en la Figura 2.10 y se trata de una pantalla táctil gráfica LCD de marca SHCHV de siete pulgadas. Esta pantalla es compatible con la Raspberry Pi 3 B por lo que su configuración ha resultado sencilla. Dispone de dos interfaces, la interfaz HDMI se utiliza para mostrar la imagen en la pantalla y la interfaz USB se utiliza para la

función táctil y transmisión de datos. Cabe destacar que el voltaje del puerto de datos no puede exceder 3.6V.

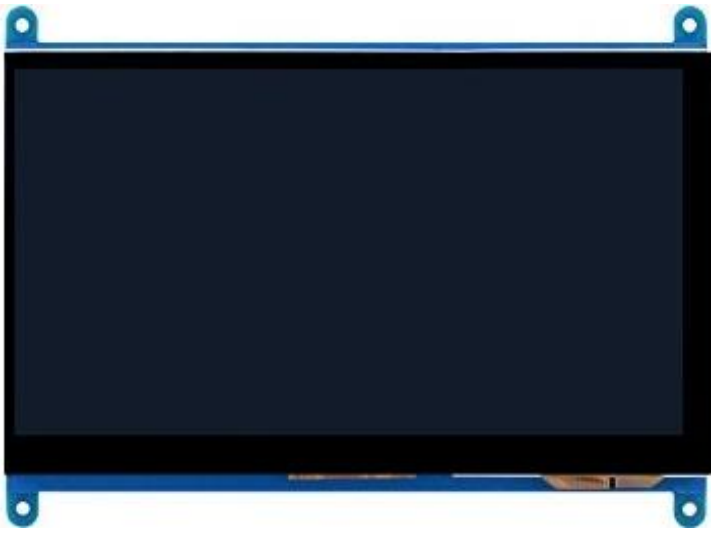


Figura 2.10 Pantalla táctil

II. Características técnicas [33]

- Resolución: 800x480
- Dimensión de la pantalla : 7 pulgadas
- Tensión de alimentación: 3 ~ 5V CC
- Pantalla en color: Sí
- No incluye batería recargable
- Temperatura de funcionamiento: -20 ~ +70 grados centígrados
- Retroiluminación LED
- Tipo de LCD: TFT

III. Ventajas y desventajas

Ventajas

- Comodidad.
- El polvo y humedad no afectan a su funcionamiento.

Desventajas

- Son frágiles al contacto con objetos afilados.
- La mayoría de pantallas táctiles suelen provocar una pérdida de brillo de un 25%.

IV. Motivo de elección

Las alternativas a la solución propuesta son las siguientes;

Pantalla no táctil con ratón

Una pantalla no táctil se trata de un periférico de salida, por lo que solamente muestra la información procedente del ordenador. Para que el usuario pueda interactuar con lo que se muestra en la pantalla es necesario un ratón. Esta alternativa es más económica pero menos cómoda y útil para este proyecto, por esa razón no se ha seleccionado una pantalla no táctil con ratón.

Pantalla LCD alfanumérica

Este tipo de pantallas son periféricos de salida compuestos por LEDs. La pantalla de LCD no es táctil y no puede mostrar la interfaz de usuario diseñada de la aplicación. El motivo por el que no se ha seleccionado es por la comodidad y visualización dado que solo puede informar.

3.2.3.2 Cámara

I. Descripción

Una cámara es un periférico de entrada que permite captar, almacenar y reproducir imágenes y vídeos. La cámara seleccionada en este proyecto es la mostrada en la Figura 2.11 y trata de una cámara de visión nocturna de la marca Waveshare junto con dos piezas de sensor IR luz LED que permiten visualizar la imagen emitida por la cámara en lugares oscuros. Esta cámara es totalmente compatible con la Raspberry Pi 3.

Figura 2.11 Cámara Raspberry Pi

II. Características técnicas [34]

- Tensión de alimentación: 3.3 V
- Dimensión: 25 mm x 24 mm x 6 mm
- Interfaz de la cámara de 15 pines, por lo que se conecta directamente a la placa Raspberry Pi
- Número de píxeles: 2592x1944.
- Campo de visión: 2,0x1,33 m a 2 m.
- Resolución: 1080p (1080 líneas horizontales)

III. Ventajas y desventajas

Ventajas

- Compatibilidad con la Raspberry Pi 3 B.
- Permite visión nocturna.

Desventajas

- Baja resolución.

IV. Motivo de elección

Las alternativas a la solución propuesta son las siguientes;

Cámara webcam

Las webcams son cámaras que se conectan mediante USB a la Raspberry Pi y no están hechas específicamente para el uso con la Raspberry Pi. Debido a que se deben pasar los datos del USB a la CPU se obtiene una sobrecarga, la cual no se obtiene con la cámara Raspberry Pi. Por lo que se ha elegido la cámara Raspberry Pi ante la webcam por el rendimiento, por la calidad de imagen y por el precio.

3.2.4 Ordenador del sistema

Todos los elementos hardware deben ser controlados y actuados, por ello son conectados a un ordenador que procesa la lectura de los sensores, manda órdenes a los actuadores y permite mostrar información mediante la interfaz gráfica. Concretamente el ordenador es el dispositivo donde se ejecuta la aplicación desarrollada que es la que realiza el control de todos los dispositivos.

I. Descripción

Ordenador de sistema hace referencia a una placa con un procesador y alguna memoria que sea capaz de recibir y procesar datos para convertirlos en información útil que posteriormente se envían a las unidades de salida. Los ordenadores se utilizan como sistemas de control en diversos dispositivos, es decir es el que gobierna el comportamiento de otros dispositivos.

En este proyecto se utiliza como ordenador del sistema la placa Raspberry Pi que es un ordenador, concretamente una placa base, de pequeño tamaño que cuenta con un procesador, RAM, almacenamiento mediante una tarjeta SD y los típicos puertos de hardware, por lo que se puede emplear en diversos proyectos. Se va a utilizar la Raspberry Pi 3 modelo B que es la más actual, la cual se puede observar en la Figura 2.12.

Figura 2.12 Raspberry Pi 3 B

La Raspberry Pi 3 B es el cerebro del proyecto, dado que es el ordenador que recibe datos de los sensores de temperatura, humedad y nivel, procesa dichos datos junto con el programa desarrollado y ordena operaciones a los actuadores, además de mostrar la información recogida en una pantalla táctil.

II. Características técnicas [35]

- Procesador: Broadcom BCM2837 1,2 GHz de cuatro núcleos
- RAM: 1GB SDRAM 400 MHz
- Almacenamiento: tarjeta microSD (16GB)
- USB: cuatro puertos USB 2.0
- Puerto Ethernet RJ45, Wifi, Bluetooth LE
- Salida de Video HDMI
- Puerto de cámara CSI para conectar una cámara Raspberry Pi
- Conector de alimentación micro USB
- Fuente de alimentación: 2.5A a 5V
- Puertos GPIO: 40 pines

III. Ventajas y desventajas

Ventajas

- Económico y bajo consumo.
- Ordenador completamente funcional.
- Pequeño tamaño.
- Fácil conexión a internet.
- Se puede programar usando una variedad de lenguajes de programación.

Desventajas

- Problemas de electricidad.
- Se puede producir calentamiento en el caso en el que no se utilicen disipadores de calor.
- Menor procesador y memoria RAM que un ordenador industrial.
- Necesario hardware adicional para la lectura de sensores analógicos.
- Se puede dañar por desconectarlo sin un apagado adecuado.

IV. Motivo de elección

Las alternativas a la solución propuesta son las siguientes;

Arduino

Arduino es una combinación del hardware y el software. Dado que se trata en lo que respecta al software de un programa de código abierto y, respecto al hardware, es un pequeño ordenador que se puede programar para leer y controlar los componentes eléctricos conectados a éste. Arduino viene con un paquete que tiene hardware y para controlar el hardware tienen su propio software llamado Arduino Software (IDE). Los motivos por los que no se ha elegido Arduino es porque tiene menor velocidad que la Raspberry Pi e implica mayor coste económico.

Ordenador industrial con tarjeta de adquisición de datos

Ordenador industrial hace referencia a cualquier tipo de ordenador, incluso una tableta, que realiza el procesamiento de los datos, pero para la adquisición de éstos se necesitaría una tarjeta de adquisición de datos, como por ejemplo Labjack, que se comunica con los sensores y actuadores para el funcionamiento de la aplicación. Respecto a fiabilidad, esta opción sería la más recomendable dado que un ordenador industrial garantiza disponibilidad total, es decir, 24 horas en funcionamiento los 7 días de la semana. Sin embargo el coste económico de un ordenador industrial junto con una tarjeta de adquisición de datos es mucho mayor que el de una Raspberry Pi, por ese motivo no se ha elegido un ordenador industrial junto con una tarjeta de adquisición de datos.

3.3 Solución propuesta

Dado que en el apartado anterior se ha expuesto la solución hardware elegida, en este apartado se propone la solución software. El producto que se presenta en esta memoria es una "caseta domótica". Empleando el término domótica como conjunto de sistemas que automatiza las acciones relacionadas con el cuidado de una mascota, como es el caso de la alimentación y la hidratación y acciones adicionales como el control del clima y la vigilancia. Con el fin de automatizar la caseta, como se ha comentado anteriormente, se han incorporado sensores y actuadores y se ha dotado de una programación la cual hace posible el control y permite a los usuarios adecuar la configuración a sus necesidades. La caseta consta de un ventilador, una manta eléctrica, una válvula de agua y otra de comida las cuales permiten el paso del agua y de la comida. Los actuadores de la caseta pueden ser controlados por el dueño ya que disponen de tres posibles modos de funcionamiento: Automático, Manual->Encender y Manual->Apagar. Cada modo de funcionamiento en cada actuador funciona de forma diferente. A continuación, en el Cuadro 3.1 se presenta lo que permite realizar cada modo en cada actuador.

	Automático	**Encender**	**Apagar**
Comedero	Permite establecer los horarios en los que se proporcionará automáticamente comida a la mascota. Se permite definir dos horas de comida, una por la mañana que debe establecerse en el rango de [6:00, 16:00] y otra por la tarde que debe establecerse en el rango de [17:00, 5:00].	Permite abrir la válvula de comida y el consecuente paso de comida indefinidamente.	Permite cerrar la válvula de comida por lo que hasta que no se cambie de modo de funcionamiento, no se proporcionará comida.

Bebedero	Permite que la mascota siempre disponga de agua mediante el control del nivel de líquido que se realiza. En el caso en el que el nivel de agua de la mascota sea bajo, se abrirá la válvula de agua hasta que el nivel sea alto, esté lleno, en este momento se cerrará la válvua, de manera que ni desbordará el agua ni a la mascota le faltará agua.	Permite abrir la válvula de agua y el consecuente paso de agua indefinidamente.	Permite cerrar la válvula de agua, por lo que hasta que no se cambie de modo de funcionamiento, no se proporcionará agua.
Ventilador	Permite establecer un punto de consigna, es decir, un número de grados definidos dentro de un rango [22,28] a partir de los cuales se activará el ventilador. Por ejemplo, si se establece como punto de consigna 24 grados, el ventilador será activado cuando la temperatura sea mayor o igual a 24 grados.	Permite encender el ventilador indefinidamente.	Permite apagar el ventilador indefinidamente.
Manta eléctrica	Permite establecer un punto de consigna, es decir, un número de grados definidos dentro de un rango [15, 20] hasta los que se activará la manta eléctrica. Por ejemplo, si se establece como punto de consigna 17 grados, la manta será activada cuando la temperatura sea menor o igual a 17 grados.	Permite encender la manta eléctrica indefinidamente.	Permite apagar la manta eléctrica indefinidamente.

Cuadro 3.1 Modos de funcionamiento de la aplicación

Cabe destacar que, en los modos automáticos, se han definido una serie de rangos debido que el usuario no pueda establecer horas o grados incoherentes. En el caso de la comida se han establecido lo rangos para que no se pueda dar el caso de programar dos comidas a la misma hora. Mientras que en el caso del ventilador y la manta eléctrica se ha basado en una temperatura ideal, 21 grados, por lo que cuando la temperatura sea 21 no se permitirá accionar ni el ventilador ni la manta eléctrica y por ello ambos rangos están definidos por encima y por abajo debajo de este nivel de temperatura. Cabe recordar que la activación el ventilador se producirá cuando la temperatura ambiente sea mayor o igual a la de consigna, mientras que la activación de la manta eléctrica se producirá cuando la temperatura ambiente sea menor o igual a la temperatura de consigna. Como se ha comentado anteriormente, el punto de consigna que se puede establecer para que se active el ventilador es entre 22 y 28 grados, se ha puesto un mínimo de 22 grados dado que a una temperatura menor de 21 grados no se considera que se deba encender el ventilador y se ha puesto un máximo de 28 dado que a partir de 28 grados se considera necesario la activación del ventilador, por lo que siempre que el modo sea automático y la temperatura ambiente sea mayor o igual que 28 grados se activará el ventilador. Respecto a la manta eléctrica se considera un máximo de 20 y un mínimo de 15 grados, dado que se considera que una temperatura menor de 15 grados es necesaria a activación de la manta eléctrica. De esta manera se proporciona al dueño parte de control de la caseta, pero dentro de rangos coherentes para que no sea posible el mal uso del producto.

Aparte del control del clima dentro de la caseta, de la alimentación y de la hidratación, dispone de un control de nivel de comida en el depósito de alimento, de manera que cuando se detecta un nivel bajo de pienso en el depósito aparece un mensaje en la pantalla en el apartado de Depósito de comida indicando “Recargar comida”, y en el caso en el que se detecte de que no hay un nivel bajo en el depósito aparece “Comida suficiente”. Además, dispone de una cámara de visión nocturna en el interior de la caseta que permite vigilar cuando se desea a la mascota.

La caseta tiene incorporada en la parte posterior, una pantalla táctil en la que se muestra la interfaz gráfica de la aplicación, desde esa interfaz gráfica se puede visualizar la temperatura, la humedad relativa, el nivel del depósito de comida, las configuraciones que se han elegido y observar el interior de la caseta con la cámara de visión nocturna. Para dotar al producto de más comodidad se puede visualizar y controlar dicha interfaz, también mostrada en la pantalla táctil, desde el móvil, desde una tableta o desde un ordenador mediante un programa de control remoto, de manera que se puede modificar las configuraciones, visualizar la interfaz y observar cuando se desee a la mascota desde cualquier lugar.

3.4 Presupuesto

En este apartado se muestra la valoración económica del producto desarrollado. En el Cuadro 3.2 se recogen tanto los gastos en mano de obra como los gastos en hardware. Cabe destacar que la unidad monetaria que se utiliza es el Euro (€) y que la unidad de tiempo del desarrollo del proyecto que se utiliza es la hora (h.).

	Coste unitario	Cantidad	Total
Sensor DHT11 [36]	2,06	1	2,06
Sensor de nivel de agua [28]	4,82	2	9,64
Sensor de nivel para sólidos [29]	7,70	3	23,1
Placa relés [37]	10,33	1	10,33
Válvula de agua [30]	6,65	1	6,65
Válvula de comida [31]	16,83	1	16,83
Manta eléctrica [12]	19,99	1	19,99
Ventilador [32]	2,47	1	2,47
Pantalla táctil [33]	44,70	1	44,70
Cámara y luz LED [34]	12,20	1	12,20
Raspberry Pi 3 [35]	49,47	1	49,47
Limitador de tensión L7805 [38]	0,10	1	0,10
Limitador de tensión L7806 [39]	0,14	1	0,14
Pila alcalina [40]	2,79	1	2,79

Caja hermética [41]	67,66	1	67,66
Depósito de comida [42]	14,90	1	14,90
Caseta de mascota [43]	85	1	85
Mano de obra	40	8	320
I + D			108
	SUBTOTAL		761
	I.V. A	21%	160
		TOTAL	921

Cuadro 3.2 Presupuesto

Cabe resaltar que se han invertido 540 horas en Investigación y Desarrollo (I+D), concretamente en investigación, diseño, desarrollo y pruebas realizadas en este proyecto. No se tiene en cuenta en I+D las horas dedicadas al montaje hardware de la caseta dado que éstas se cobrarán por unidad. El coste que conlleva las horas dedicadas en I+D es de 21.600 €. Suponiendo que se venderán 200 casetas, se divide el coste de I+D entre la cantidad de casetas que se pretenden vender, por lo que el coste de I+D por cada caseta sería de 108 €. Por consiguiente, el coste de una caseta al consumidor final constaría de tres bloques, el sobrecoste de I+D, que son 108 €, el coste del montaje hardware de una caseta, que son 8 horas * 40 € = 320 €, y el coste del material hardware para una caseta, 333’23 €, lo que conlleva a un coste final aproximado de 761 € sin I.V.A, y con I.V.A tendría un precio de **921 €.**

4. Diseño de la solución

4.1 Arquitectura del sistema

En este apartado se comenta la arquitectura del sistema en el ámbito del software, donde se muestran los posibles patrones arquitectónicos que se pueden adoptar en el proyecto y en el ámbito del hardware, donde se muestra la arquitectura hardware del proyecto en cuestión.

4.1.1 Patrones arquitectónicos

En esta sección se va a explicar tres patrones arquitectónicos utilizados en ingeniería del software, entendiendo como patrón arquitectónico una solución general y reutilizable a un problema común de la arquitectura de software. Los tres patrones que se van a tratar a continuación son los patrones Modelo Vista Controlador (MVC), Modelo Vista Presentador (MVP) y Factoría.

Modelo-Vista-Controlador

ModeloVista Controlador [14] es una solución de diseño que estructura una aplicación en tres capas diferentes: los datos (Modelo), la interfaz gráfica de usuario (Vista) y la lógica de negocio de una aplicación (Controlador). Cada uno de estos componentes está diseñado para manejar aspectos de desarrollo específicos de una aplicación.

Modelo contiene una representación de los datos con los que el sistema opera, es decir, corresponde a toda la lógica relacionada con los datos con la que trabaja el sistema y el usuario. Esto puede representar los datos que se transfieren entre los componentes Vista y Controlador o cualquier otro dato relacionado con la lógica de negocios.

El modelo es el responsable de:

- Acceder a la capa de almacenamiento de datos.
- Almacenar toda la información de la aplicación, así como la información interna del programa.
- Administrar todos los accesos a la información (lecturas, escrituras y modificaciones).

- Responder a la solicitud de la Vista y a las instrucciones del Controlador para actualizarse.

Vista es la interfaz de comunicación con el usuario, es decir, es la representación gráfica del modelo. Incluirá todos los componentes de la interfaz de usuario como cuadros de texto, menús desplegables, botones, etc. que se muestra al cliente y los mecanismos interacción con éste.

La vista, es la encargada de:

- Recibir datos del modelo y mostrarlos al usuario.
- Manejar la visualización de la información.
- Reaccionar ante las acciones del usuario.
- Realizar peticiones al controlador.

Controlador actúa como intermediario entre la Vista y el Modelo, recibe la petición de la Vista, dado que procesa dicha petición y realiza los cambios correspondientes sobre el Modelo. En otras palabras, la función del controlador es gestionar el flujo de información entre el Modelo y la Vista y realizar las conversiones correspondientes para adaptar los datos a las necesidades del Modelo y de la Vista.

El controlador es el responsable de:

- Recibir los eventos de entrada, es decir, las acciones de la Vista.
- Realizar peticiones al Modelo para modificarlo.
- Controlar la lógica de la aplicación.

En resumen, los Controladores actúan como una interfaz entre los componentes Modelo y Vista para procesar todas las solicitudes entrantes, manipular los datos utilizando el componente Modelo e interactuar con las Vistas para generar el resultado final.

En la Figura 4.1 se puede observar la comunicación entre los componentes del MVC.

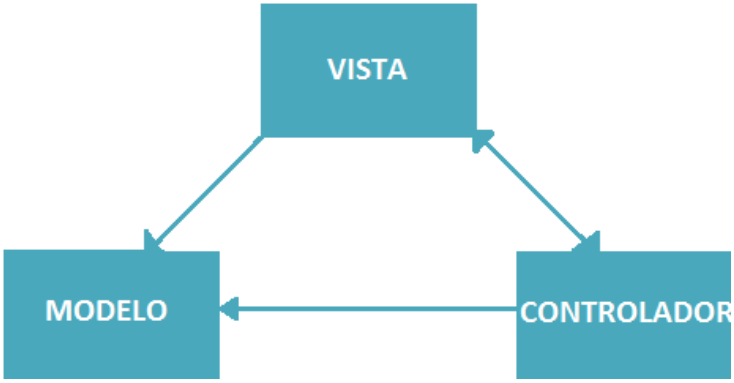


Figura 4.1 Interacción componentes MVC

El flujo de funcionamiento [15] del patrón MVC es el siguiente:

1. El Controlador recibe una solicitud.
2. El Controlador procesa la petición y modifica el Modelo de datos.
3. El Modelo procesado se pasa posteriormente a la Vista.
4. La Vista transforma el Modelo en el formato de salida apropiado y representa el resultado final.
5. Por último, se procesa la respuesta.

Gracias al MVC se logra la separación de conceptos, la posibilidad de reutilización de código, la no repetición de código, el fácil mantenimiento, la fácil legibilidad de código y la fácil comprobación mediante pruebas unitarias automatizadas.

Modelo-Vista-Presentador

El patrón Modelo Vista Presentador [15] es un patrón derivado del patrón de arquitectura MVC que fue creado para poder realizar pruebas de la representación de los datos sin necesidad de usar un entorno de interfaz gráfica ya que estos pruebas se ejecutan más rápido y son más sencillos de escribir, por ello, el objetivo principal de MVP es separar la capa de presentación de la lógica de la misma, de esta manera todo lo relacionado con la funcionalidad de la interfaz queda separado de su representación en la pantalla. Gracias a ello se pueden realizar pruebas reemplazando la interfaz real por una interfaz de mentira, también llamado *stub*, pudiendo realizar test unitarios que son mucho más rápidos que los test de interfaz ya que los test de interfaz requieren de un

emulador o entorno gráfico para poder funcionar simulando las acciones del usuario. Este patrón consiste en tres componentes:

Modelo: al igual que en el patrón MVC, el modelo es el componente que almacena y gestiona el estado del sistema.
Vista: Es la encargada de mostrar los datos en la interfaz de usuario de la forma más simple posible sin manejar lógica de presentación. Contiene una referencia al presentador, invocándolo cada vez que se realiza una acción sobre la interfaz. No puede actualizarse directamente desde el Modelo ya que la única forma en que se puede actualizar la Vista es a través del Presentador. La Vista se actualiza cuando el usuario lo solicita (presionando un botón 'Actualizar', por ejemplo). Todo el estado se gestiona en el presentador y no en la vista.

- Ventajas: Separación limpia de la Vista y el Modelo, por lo que el código de la Vista es más legible.
- Desventajas: Conlleva más trabajo ya que necesita crear por cada Vista una interfaz de java y una clase Presentador, en ocasiones este trabajo no está justificado.

Presentador: Es el responsable de la lógica operacional de la vista. Gestiona el estado de la Vista, decide qué mostrar en la interfaz y reacciona a las notificaciones de entrada del usuario desde la Vista, por lo que decide qué ocurre cuando se interactúa con la Vista. Además, recupera los datos del Modelo y se los devuelve formateados a la Vista. El presentador está desacoplado de la implementación de la Vista y habla con él a través de una interfaz.

El presentador se comunica con la vista de la siguiente manera:

1. La Vista se comunica con el Presentador directamente con llamadas a funciones de una instancia del Presentador sin esperar valores de retorno en los métodos.
2. El Presentador se comunica con la Vista mediante una interfaz java sin conocer la instancia concreta de la vista.

La comunicación entre los componentes del MVP se puede observar de manera gráfica en la Figura 4.2.

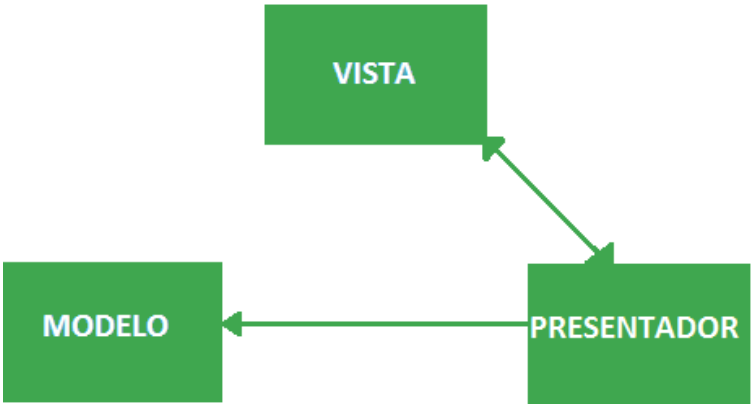


Figura 4.2 Interacción componentes MVC

En cuanto a las diferencias entre MVC y MVP. La diferencia fundamental entre estos dos patrones es la dependencia entre el Modelo y la Vista, la cual está presente en MVC, pero no en MVP.

Factoría

Se trata de un patrón de diseño creacional, lo que significa que se ocupa de los mecanismos de creación de objetos que permiten abstraer dicha creación. Una factoría [17] es un objeto que se encarga de la creación de un cierto tipo de objetos y se utiliza cuando la creación de un objeto implica algo más que una simple instanciación. Este patrón se suele utilizar cuando una clase no puede anticipar el tipo de objeto que se debe crear porque generalmente depende de algún parámetro de entrada, de manera que delega esa responsabilidad a una clase factoría la cual es capaz de especificar el tipo del objeto que se debe crear dependiendo del estado de la aplicación. Este patrón está compuesto por lo siguientes elementos:

- **Producto**: Define la interfaz de los objetos que se van a crear.
- **Producto concreto:** Es la implementación de la interfaz, tiene toda la funcionalidad concreta. El Creador crea y devuelve estos Productos concretos dependiendo del comportamiento que se le quiera dar a la aplicación.
- **Creador:** Se trata de una interfaz que define y declara la factoría que devuelve un objeto del tipo Producto.
- **Creador concreto:** Es la implementación del Creador y sobrescribe el método de la factoría para devolver una instancia concreta de un objeto Producto concreto.

En la Figura 4.3 se puede observar el diagrama UML [16] que muestra las relaciones entre los elementos de una factoría.

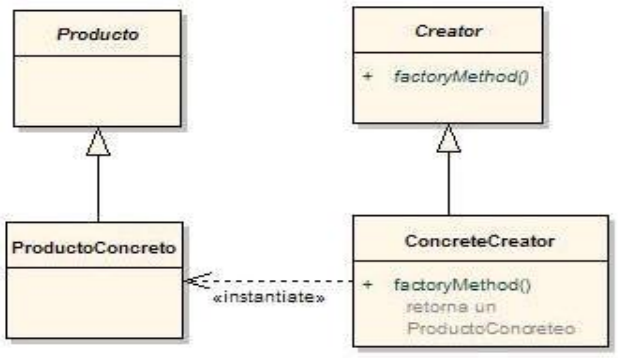


Figura 4.3 Interacción componentes patrón factoría

Por lo tanto, si en un futuro se desea añadir un nuevo Producto sólo habría que crear un nuevo Producto concreto y añadirlo como opción al Creador concreto sin que el programa cambie en absoluto, teniendo un impacto mínimo al tener que soportar nuevos productos.

4.1.2 Arquitectura hardware

En cuanto a la arquitectura hardware, se compone de una caseta sobre la que se incorpora todo el material de campo alrededor de ella. Como se puede observar en la Figura 4.4, en suelo de la caseta se depositan el bebedero, el comedero y la manta eléctrica. En el techo de la caseta se incorpora el depósito de la comida, en la fachada trasera la pantalla táctil junto con la cámara y en la delantera el ventilador. Mientras que por los laterales de la caseta se colocan las válvulas de comida y de agua y en la parte inferior de la caseta se dispone el cableado introducido dentro de canaletas para que la mascota no tenga accesibilidad a ellos.

Figura 4.4 Arquitectura hardware del sistema

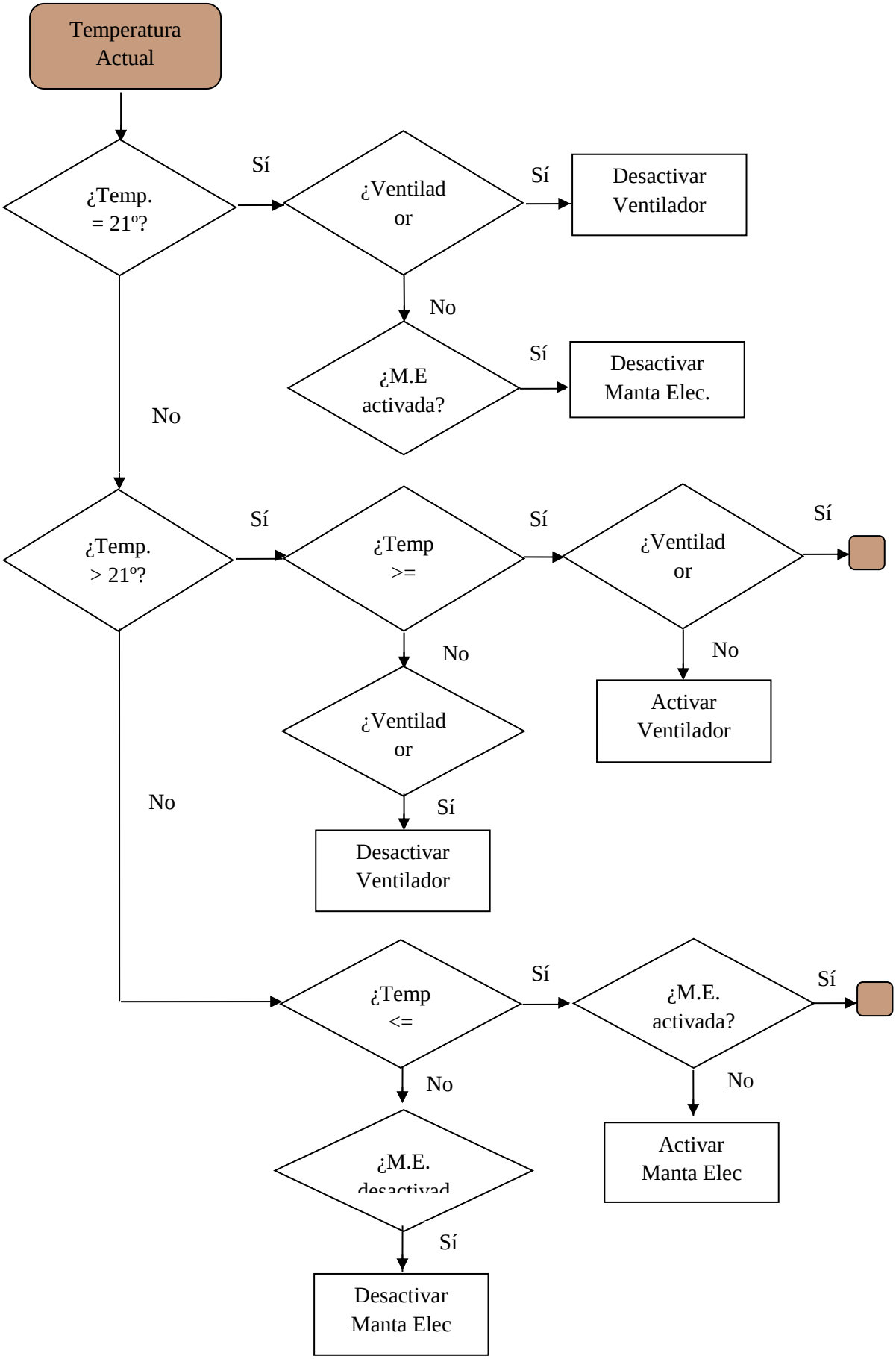
4.2 Diseño detallado

En este apartado se expone el diseño detallado de la parte software, los flujos de ejecución del programa, y la parte de hardware, que hace referencia a la conexión general de cada componente electrónico.

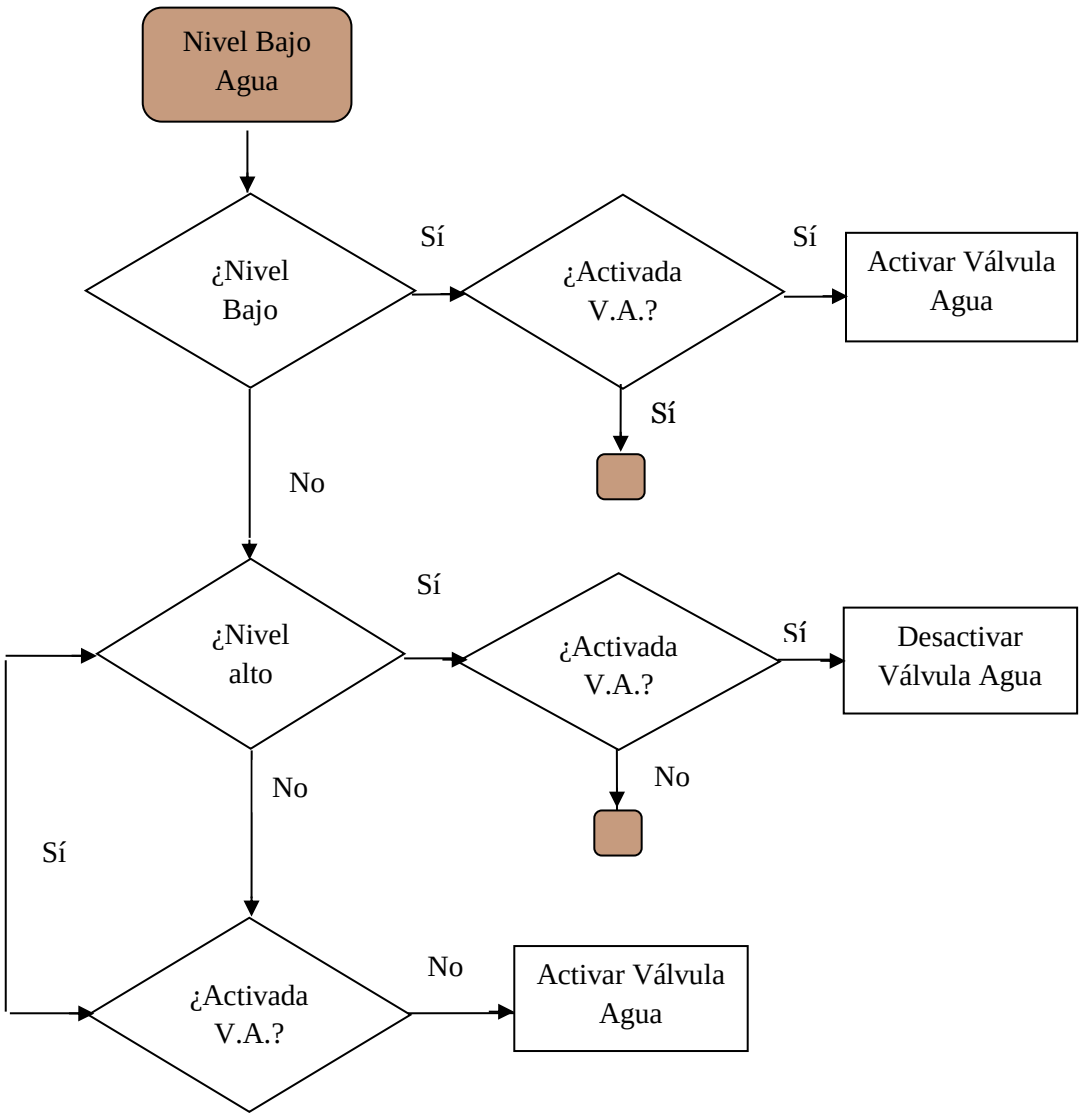
4.2.1 Diagramas de flujo

En este apartado se muestran los diagramas de flujo de la activación y desactivación de los actuadores cuando en la aplicación se selecciona el modo automático. Gracias a los diagramas de flujo se puede observar de forma gráfica el flujo de trabajo que se efectúa cuando se produce un cambio de hora o de temperatura. Respecto a la simbología y su significado; el óvalo representa el inicio y final del flujo de trabajo, el rectángulo hace referencia a la ejecución de una actividad mientras que el rombo es el símbolo que se utiliza para formular una pregunta.

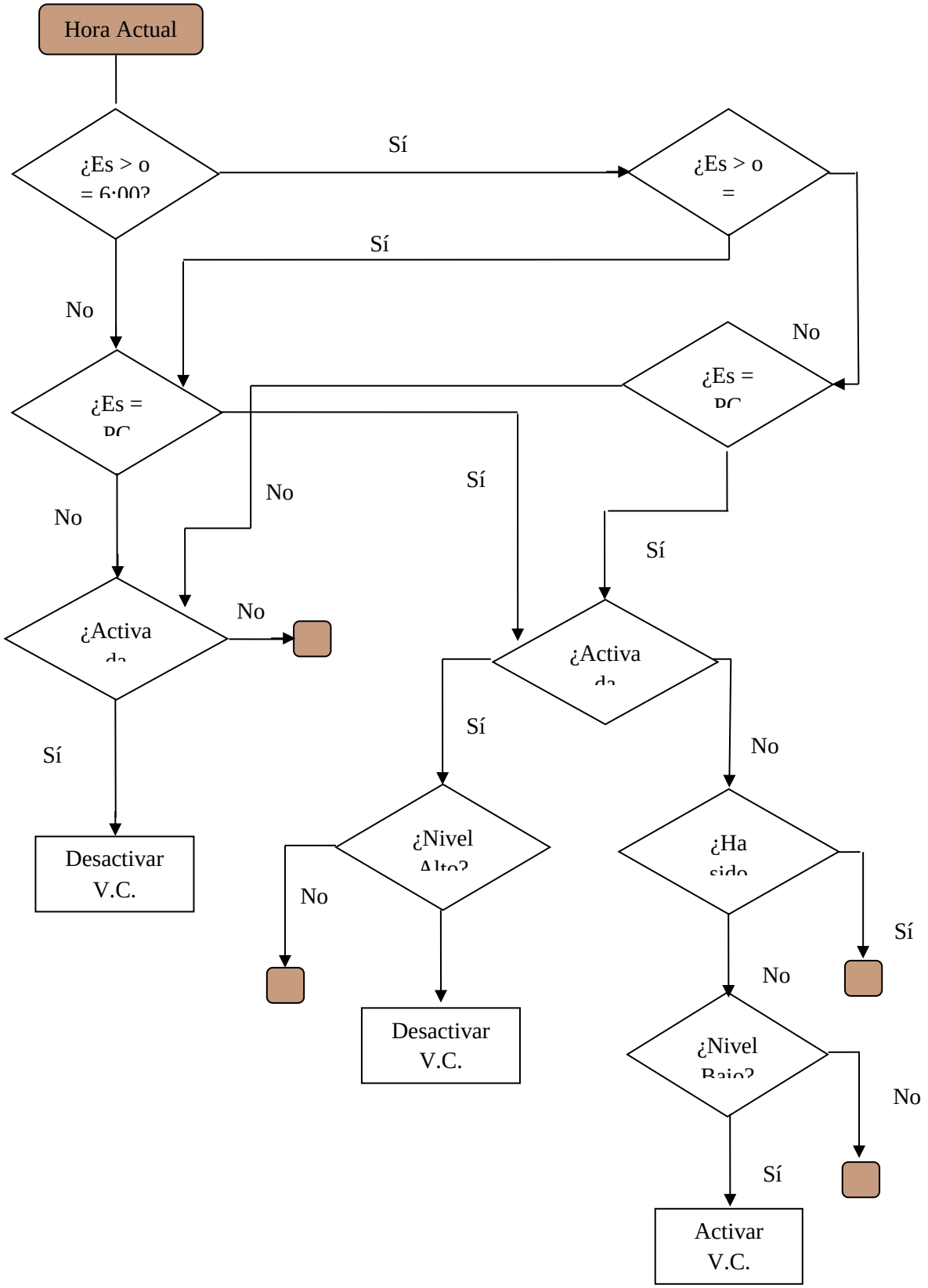
El diagrama de flujo que se muestra a continuación es el correspondiente al **modo automático de la temperatura**, donde la abreviatura "PCManta" hace referencia el Punto de Consigna que establece el usuario mediante la interfaz para que se active la manta eléctrica mientras que la abreviatura "PCVent" hace referencia el Punto de Consigna que se establece para que se active el ventilador. Cabe destacar que la abreviatura "M.E." hace referencia a la manta eléctrica.


Temperatura Actual
¿Temp. = 21º?
Sí
¿Ventilador
Sí
Desactivar Ventilador
No
¿M.E activada?
Sí
Desactivar Manta Elec.
No
¿Temp. > 21º?
Sí
¿Temp >=
Sí
¿Ventilador
Sí
No
¿Ventilador
No
Activar Ventilador
No
Sí
Desactivar Ventilador
¿Temp <=
Sí
¿M.E. activada?
Sí
No
¿M.E. desactivad
No
Activar Manta Elec
Sí
Desactivar Manta Elec

Respecto al diagrama de flujo que se sigue cuando se establece el **modo automático del bebedero** se muestra a continuación. Como se ha comentado anteriormente, en el bebedero se incorporan dos sensores de nivel para controlar el nivel del agua del que dispone la mascota en el bebedero, por lo que su gestión se basa en que el nivel de agua esté siempre entre los dos sensores, el sensor de nivel bajo y el sensor de nivel alto. Cabe destacar que la abreviatura "V.A." hace referencia a la válvula de agua.



Por último, se puede observar a continuación el diagrama de flujo correspondiente a la gestión del **modo automático de la comida**. En este modo automático, la activación de la válvula de comida, es decir, permitir el paso del alimento hacia el comedero, depende de dos factores, en primer lugar, las dos horas que establece el usuario mediante los elementos asociados al modo automático del comedero y, en segundo lugar, al igual que ocurre en el bebedero, si el nivel de alimento que dispone la mascota en el comedero es suficiente. Como se ha comentado uno de los factores para la activación de la válvula de comida son las horas que establece el usuario, éste puede establecer dos horas y en el diagrama de flujo estas dos horas son representadas mediante las abreviaturas “PC Mañana” que hace referencia al Punto de Consigna de la Mañana, es decir, la hora que está en el rango de la mañana que ha establecido el usuario para que se proporcione comida a la mascota y ”PC Tarde” que se refiere al Punto de Consigna de la Tarde. Cabe destacar que la abreviatura “V.C.” hace referencia a la Válvula de Comida.


Hora Actual
¿Es > o = 6:00?
Sí
¿Es > o =
Sí
No
No
¿Es = PC
¿Es = PC
No
Sí
No
Sí
¿Activa da
No
¿Activa da
Sí
Sí
No
Desactivar V.C.
¿Nivel Alto?
¿Ha sido
No
Sí
No
Desactivar V.C.
¿Nivel Bajo?
No
Sí
Activar V.C.

4.2.2 Conexión general del hardware

La conexión general del hardware hace referencia a la conexión de los sensores, de los actuadores y de los periféricos con el ordenador del sistema. Como se puede observar en la Figura 4.5, los elementos de los que se compone el proyecto están numerados para poder referenciarlos fácilmente.

1. Sensor de temperatura y humedad se conecta directamente a los pines VCC, GND y a la entrada de datos correspondiente de la Raspberry Pi.
2. Los sensores de nivel de agua se conectan a los pines VCC, GND y a la entrada de datos correspondiente de la Raspberry Pi.
3. La salida de la válvula de agua se conecta a la placa de relés y la placa de relés a los pines VCC, GND y a la salida correspondiente de la Raspberry Pi.
4. La salida de la manta eléctrica se conecta a la placa de relés y la placa de relés se conecta a los pines VCC, GND y a la salida correspondiente de la Raspberry Pi.
5. La salida del ventilador se conecta a la placa de relés y la placa de relés se conecta a los pines VCC, GND y a la salida correspondiente de la Raspberry Pi.
6. La salida de la válvula de comida se conecta a la placa de relés y la placa de relés se conecta a los pines VCC, GND y a la salida correspondiente de la Raspberry Pi.
7. Los sensores de nivel de comida se conectan a los pines VCC, GND y a la entrada de datos correspondiente de la Raspberry Pi.
8. La pantalla táctil se conecta directamente a la Raspberry Pi mediante un conector USB y un conector HDMI.
9. La cámara se conecta directamente al conector CSI de la Raspberry Pi mediante un cable conector CSI.

En resumen, los sensores y los periféricos se conectan directamente a la Raspberry Pi mientras que los actuadores deben conectarse a una placa de relés, y la placa de relés a la Raspberry Pi. En el apartado de Desarrollo hardware se expondrá el conexionado detallado de cada dispositivo.

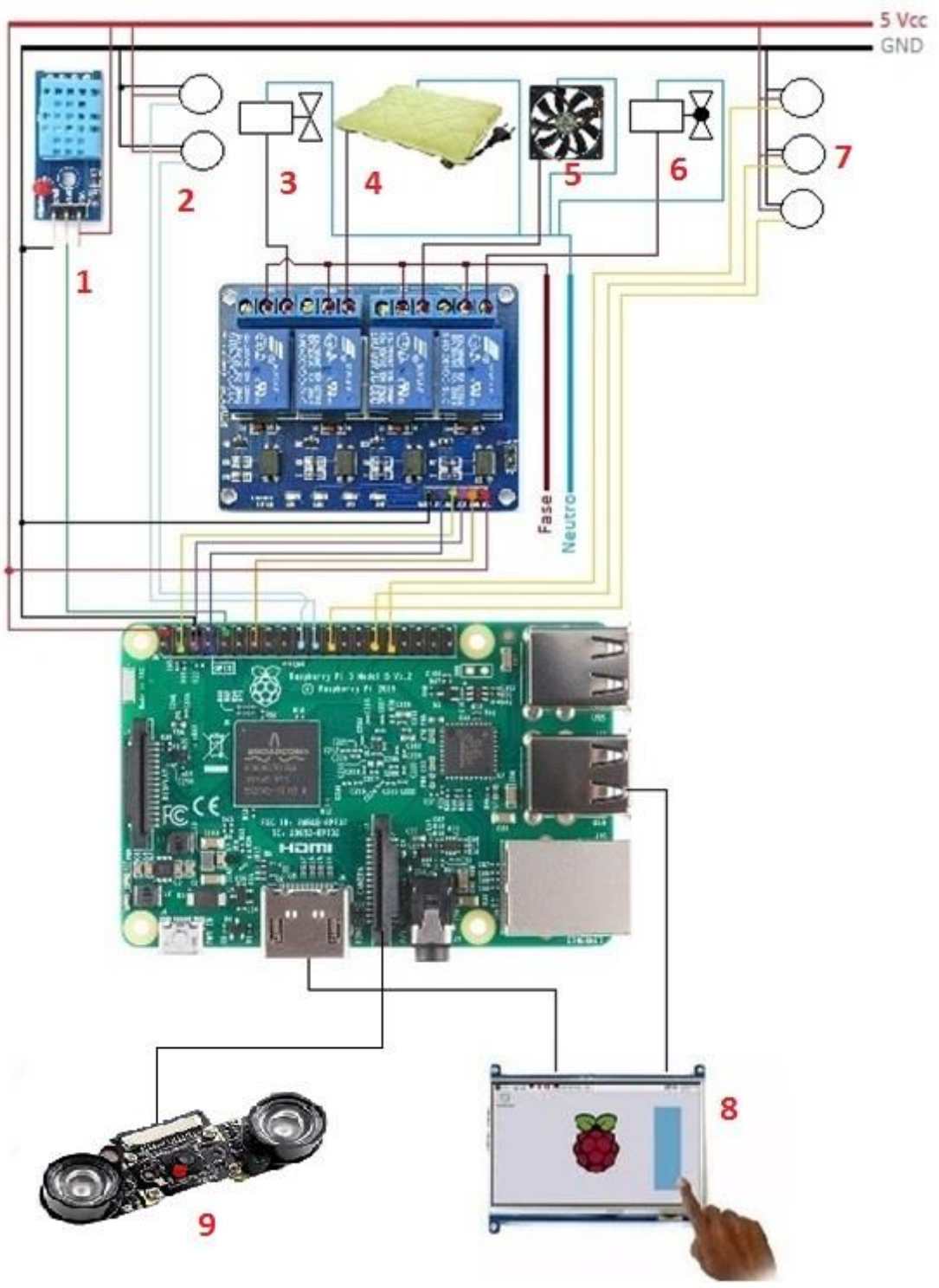


Figura 4.5 Conexión general hardware

4.3 Tecnología utilizada

En este apartado se comentan las diferentes tecnologías y herramientas software que se han utilizado en el proyecto para la implementación de la aplicación.

4.3.1 Sistema operativo

El sistema operativo que se utiliza es el principal sistema operativo compatible de la Raspberry Pi, Raspbian, que está basado en Debian y es gratuito. Está optimizado para el hardware de la Raspberry Pi y viene con más de 35.000 paquetes, que se trata de software precompilado incluido en un formato en el que resulta fácil su instalación en la Raspberry Pi. Cabe destacar que se trata de un proyecto comunitario en desarrollo activo, por lo que se sigue mejorando este sistema operativo con nuevas versiones.

Figura 4.6 Logotipo Raspbian

4.3.2 Tecnologías de implementación

Se ha usado como lenguaje de programación Java [20] dado que es el lenguaje que más se ha utilizado en el Grado de Ingeniería Informática. Java es un lenguaje de programación orientado a objetos similar a C++ que es concurrente, basado en clases, portable y distribuido.

Figura 4.7 Logotipo Java

El código compilado de Java se puede ejecutar en la mayoría de los ordenadores porque los intérpretes de Java y los entornos de tiempo de ejecución, conocidos como *máquinas virtuales Java (VM)*, existen para la mayoría de los sistemas operativos. Por esa razón es posible ejecutar código Java en la Raspberry Pi, aunque cabe decir que Java no es el lenguaje recomendado para desarrollar aplicaciones que se ejecutan en la Raspberry Pi.

4.3.3 Herramienta de desarrollo

Para el desarrollo de la aplicación se ha utilizado **Eclipse Oxygen 2017** que trata de una plataforma de programación y compilación que permite el desarrollo de aplicaciones gráficas, aplicaciones web, etc.

Figura 4.8 Logotipo Eclipse

El Entorno de Desarrollo Integrado (IDE) de Eclipse emplea complementos para proporcionar más funcionalidades según lo que necesite usuario. En el caso de este proyecto se utiliza el complemento *WindowBuilder* para la creación de la interfaz gráfica. La búsqueda e instalación de estos complementos es explicada en el apartado de Desarrollo software, en la sección Creación interfaz gráfica. Por lo tanto se ha utilizado esta herramienta de desarrollo porque se ha explicado en el Grado de Ingeniería Informática y porque es un entorno de desarrollo de software completo, ya que ofrece gran cantidad complementos que ayudan en la creación de aplicaciones.

4.3.4 Librerías

Dado que el lenguaje de programación que se va a utilizar es Java, para interactuar con los pines de comunicación Entrada/Salida de Propósito General (GPIO) de la Raspberry Pi 3 B se va a utilizar la **librería Pi4J**. Pi4J es un proyecto de código abierto por lo que los problemas que se producen con Pi4J se encuentran solucionados con la instantánea más actual. Para usar Pi4J se debe de instalar las bibliotecas Pi4J en la Raspberry Pi e importar la librería Pi4J en la herramienta de desarrollo, Eclipse.

Para instalar Pi4J en Raspberry Pi se debe ejecutar el siguiente comando directamente en la consola de la Raspberry Pi.

```
curl -s get.pi4j.com | sudo bash
```

El cual instalará las bibliotecas Pi4J y los archivos fuente de ejemplo en:

```
/ opt / pi4j / lib
/ opt / pi4j / examples
```

Para importar la librería Pi4J en Eclipse se debe descargar la librería más actual de la página web oficial [18] como se observa en la Figura 4.9.

Figura 4.9 Descarga librería Pi4J

Posteriormente descomprimir el fichero con extensión .zip y añadir en Eclipse las librerías necesarias de Pi4J.

Para compilar el programa escrito en Java que hace uso de la biblioteca Pi4J puede usarse el siguiente comando:

```
javac -classpath .:classes: / opt / pi4j / lib / '*' -d. programaJavaPi4j.java
```

o lo que es equivalente:

```
pi4j --compile programaJavaPi4j.java
```

Para ejecutar el programa puede usarse el siguiente comando:

```
sudo java-classpath .:classes: / opt / pi4j / lib / '*' programaJavaPi4j
```

o lo que es equivalente:

```
sudo pi4j --run programaJavaPi4j
```

En el caso en el que se desee ejecutar ejecutables con extensión .jar puede usarse el siguiente comando:

```
sudo java -classpath '.:classes:*classes:/opt/pi4j/lib/*' -jar programaJavaPi4j.jar
```

En este proyecto el usuario no debe ejecutar la aplicación mediante comandos dado que la aplicación se ejecutará automáticamente al arrancar el sistema operativo, cuya configuración será más adelante explicada. En el caso en el que se cierre la aplicación, lo cual no es fácil dado que se ha ocultado la barra superior de la aplicación que permite minimizar y cerrar, el usuario no deberá de ejecutar la aplicación a través de consola sino que sólo deberá pinchar sobre el icono de consola y se ejecuta automáticamente la aplicación.

En el caso en el que se quiera ejecutar por terminal se debe especificar qué placa se utiliza, en este caso, raspberry, mediante argumentos, lo cual se explica en la parte de desarrollo de software.

```
sudo java -classpath '.:classes:*classes:/opt/pi4j/lib/*' -jar programaJavaPi4j.jar raspberry
```

4.3.5 Conexión a escritorio remoto

El programa que se utiliza para poder observar la interfaz que se muestra en la pantalla de la caseta es VNC (Virtual Network Computing) el cual es un programa de software libre basado en una estructura cliente-servidor que permite controlar remotamente otro ordenador. Transmite los eventos del teclado y del ratón de un ordenador a otro, transmitiendo las actualizaciones de la pantalla gráfica en la otra dirección, a través de una red. De esa forma se consigue que, desde cualquier dispositivo, ya sea tableta, móvil u ordenador e independientemente del sistema operativo que dispongan dichos dispositivos, se pueda acceder remotamente a la pantalla que se ubica en la caseta. En este caso se ha utilizado una suscripción gratuita en la que se permite configurar hasta cinco equipos remotos y a cada equipo se puede conectar hasta tres ordenadores, esto es suficiente dado que para esta aplicación solamente se utiliza un equipo remoto, la Raspberry Pi, y a dicho equipo se podrán conectar hasta tres dispositivos. Raspbian dispone de VNC para ello se ha de habilitar en el menú de configuración, el cual se puede observar en la Figura 4.10.

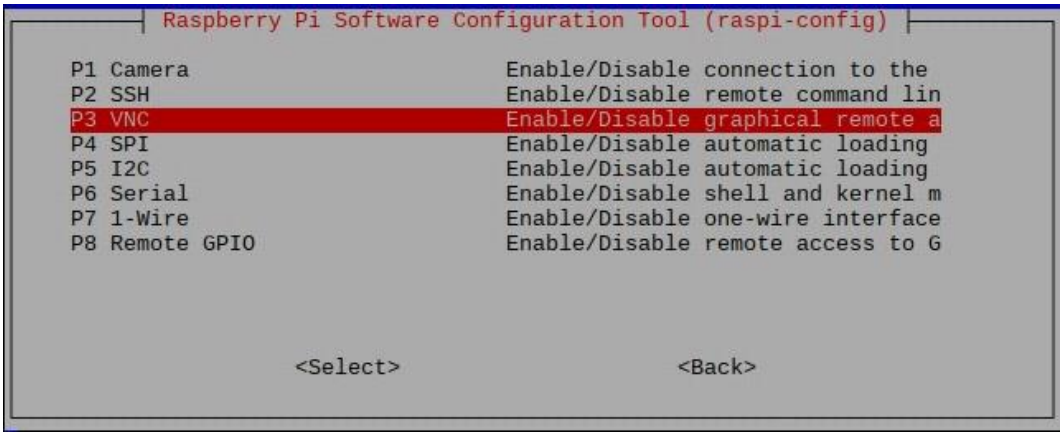


Figura 4.10 Menú configuración Raspberry Pi

Se habilita VNCServer en la Raspberry Pi dado que para que los clientes puedan ver e interactuar remotamente con su escritorio es necesario que disponga de VNCServer la Raspberry Pi.

Posteriormente se debe crear una cuenta en la página web de RealVNC [19] y a continuación licenciarse, para ello se pulsa en el menú situado arriba a la derecha, y se seleccionar *Licensing* como muestra la Figura 4.11.

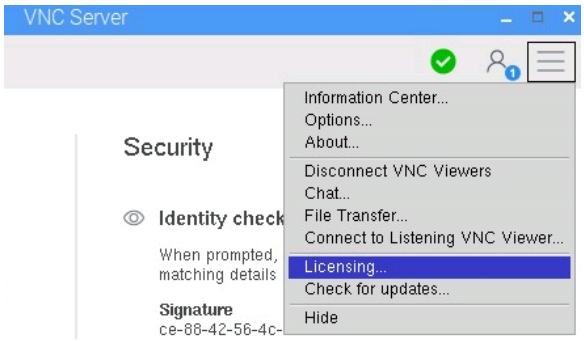


Figura 4.11 Licenciando VNC Server

Aparece una ventana como la que se observa en la Figura 4.12 y se selecciona la opción *Sign in to your RealVNC account* y Next.

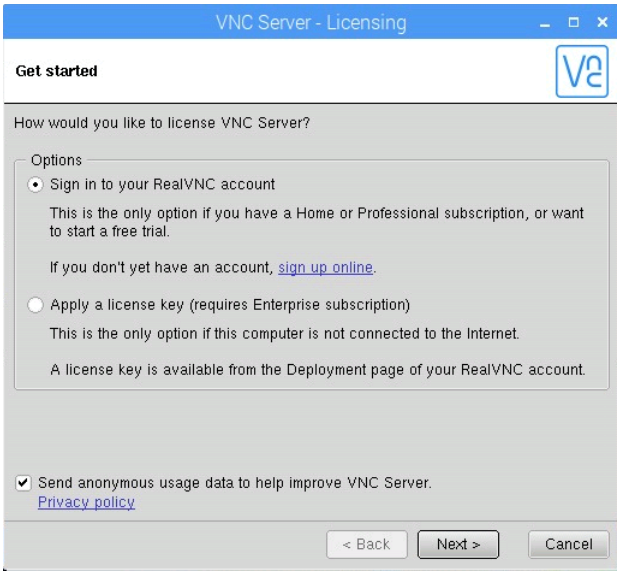


Figura 4.12 Licenciando VNC Server (I)

En la Figura 4.13 se debe introducir los datos de la cuenta previamente creada mediante la página web, por lo que se debe introducir email y contraseña.

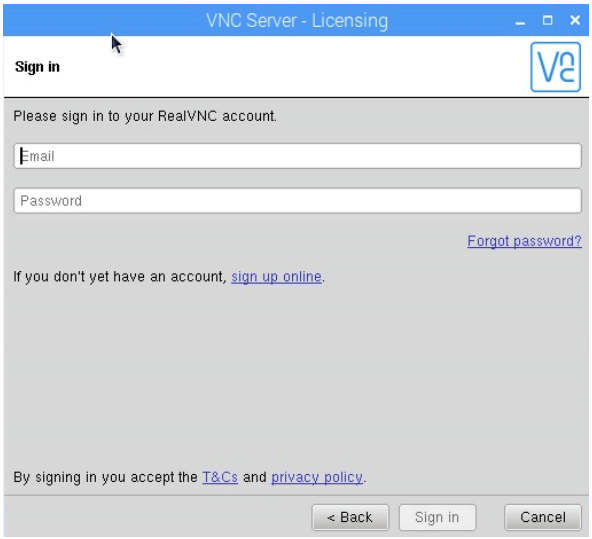


Figura 4.13 Licenciando VNC Server (II)

Por último, aparece una ventana como la de la Figura 4.14 donde se debe elegir "*Direct and cloud conectivity*" para poder establecer una conexión en la nube. Cabe destacar que una conexión en la nube es una conexión desde un dispositivo que ejecuta VNC Viewer a un equipo remoto que ejecuta el VNC Server, logrando así conectarse a un equipo remoto que no se ubique en la misma red de área local.

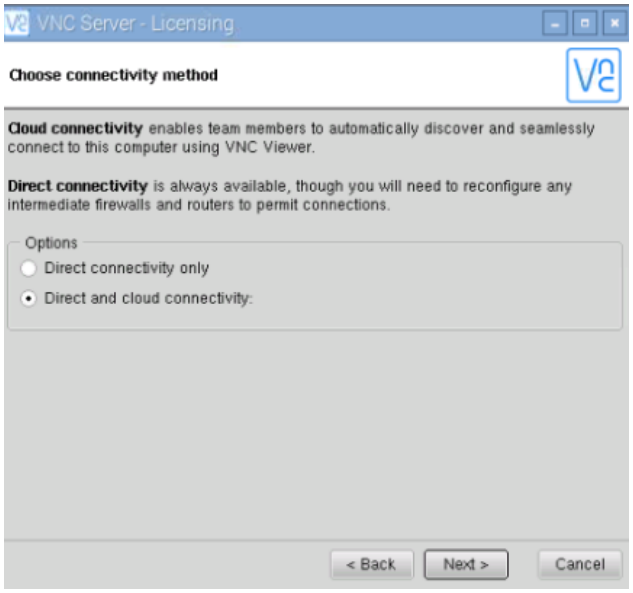


Figura 4.14 Licenciando VNC Server (III)

Y pulsando en Next aparece una nueva ventana como la de la Figura 4.15 donde se informa del número de equipos remotos que ya han sido configurados.

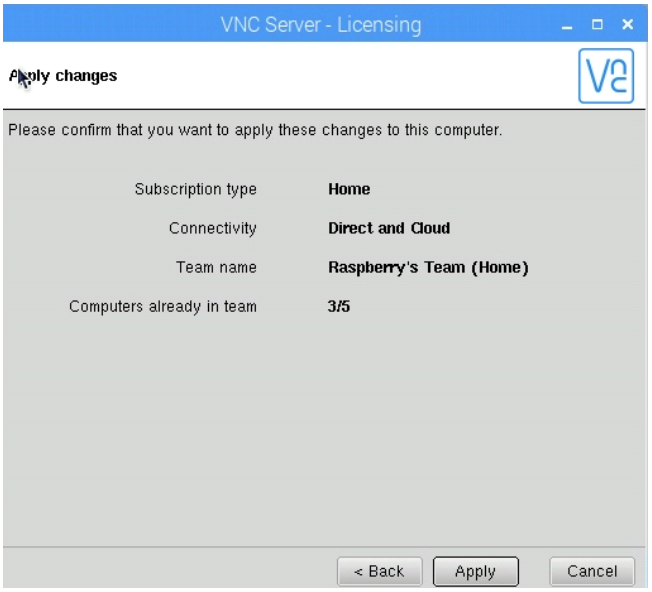


Figura 4.15 Licenciando VNC Server (IV)

Para poder mostrar la imagen grabada con la cámara de la Raspberry Pi mediante VNC se debe habilitar desde VNC Server la opción *"Enable experimental direct capture mode"* que se encuentra en *Options -> Troubleshooting* como muestra la Figura 4.16.

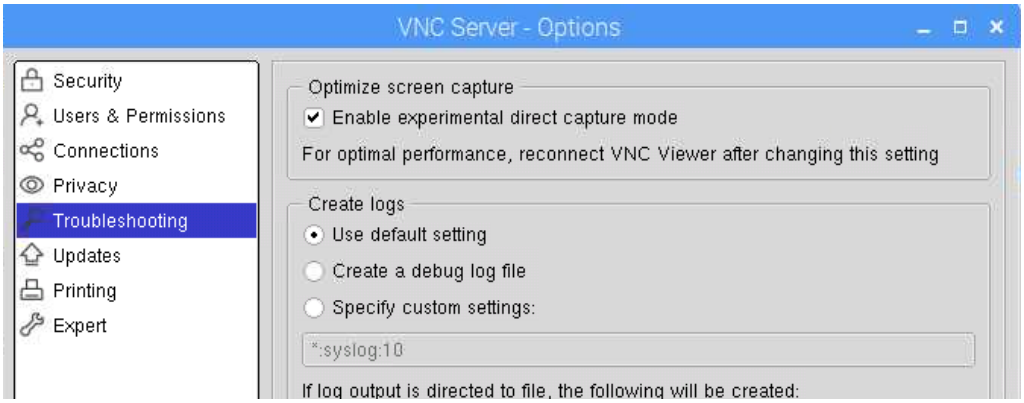


Figura 4.16 Habilitar modo captura

Una vez ya instalado y configurado el VNC Server se puede acceder a él desde un cliente, VNC Viewer, desde cualquier dispositivo. A continuación, se muestra como conectarse al servidor desde un ordenador y desde un móvil para poder ver lo que se muestra en la pantalla situada en la caseta y por lo consiguiente la aplicación de la caseta cuando está en ejecución.

Desde el lado cliente, se debe descargar VNC Viewer y registrarse con los datos de la cuenta previamente realizada. Una vez realizado el registro, aparecen los equipos remotos que se han registrado mediante VNC Server. Estos equipos son a los que se puede se pueden conectar remotamente. Por lo que se selecciona el equipo al que se desea conectar de los equipos disponibles, para ello se pincha sobre el equipo registrado previamente, en la Figura 4.17 se muestran los equipos disponibles.

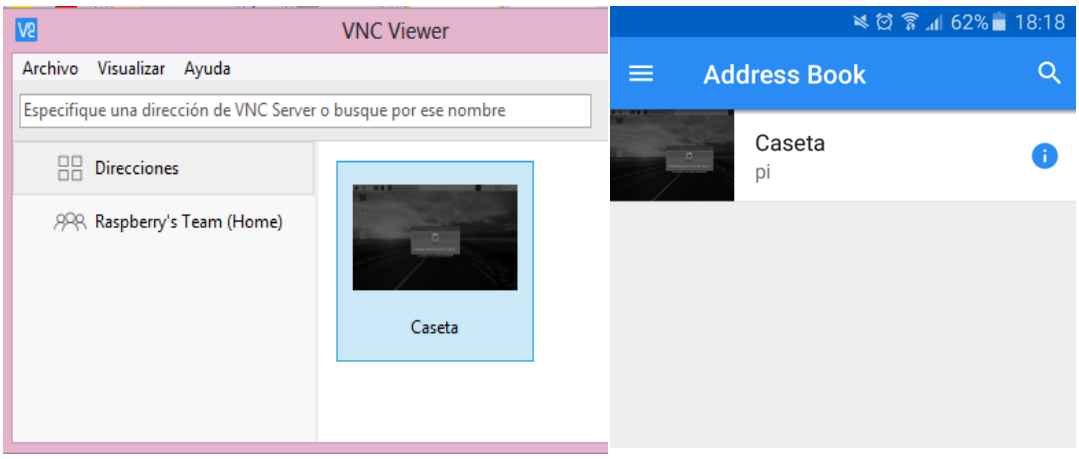


Figura 4.17 Conexión a un equipo remoto

Por último, aparece una ventana como la de la Figura 4.18 en la que se ha de realizar la autenticación a la Raspberry Pi, por lo que hay que introducir el usuario y la contraseña de acceso al escritorio de la Raspberry Pi.

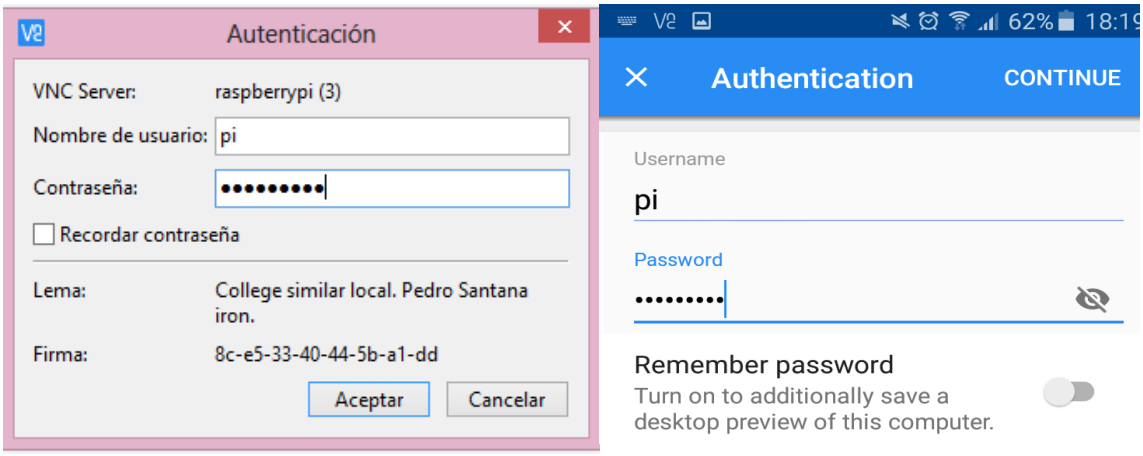


Figura 4.18 Autenticación

Una vez establecida la conexión con el servidor se puede observar el escritorio de la Raspberry Pi, como se muestra en la Figura 4.19 y permite el control sobre el escritorio lo que conlleva a poder interactuar con la aplicación.

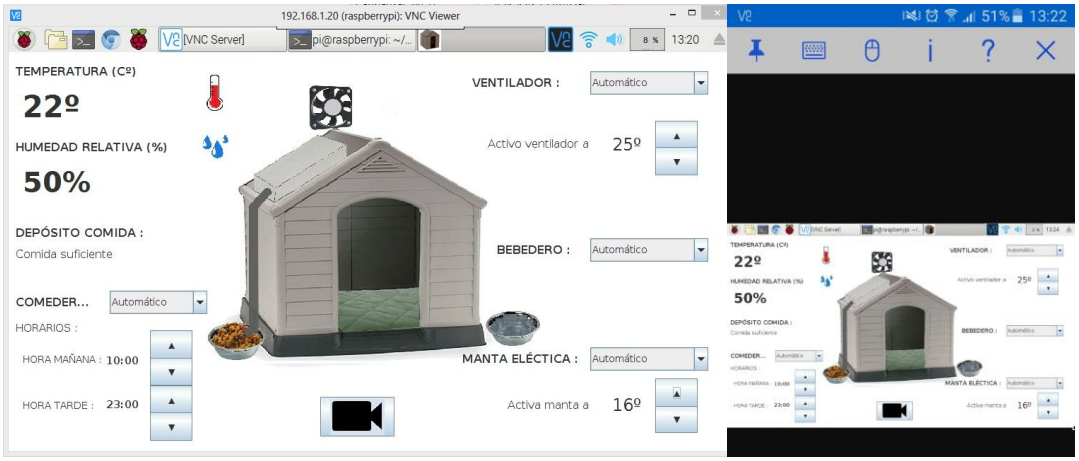


Figura 4.19 Conexión a escritorio remoto vía VNC

5. Desarrollo de la solución propuesta

En este apartado se expone el desarrollo de la solución propuesta tanto en el ámbito software, es decir, el desarrollo de la aplicación programada, como en el ámbito del hardware, donde se muestra paso a paso cómo se han interconectado todos los elementos electrónicos anteriormente presentados y como se han colocado dichos elementos interconectados alrededor de la caseta.

5.1 Desarrollo software

En esta sección se muestra el software del proyecto, es decir, la aplicación que se ocupa del control y automatización de los elementos electrónicos. La explicación se estructura en tres secciones; Modelo, Vista y Controlador, dado que en el diseño de la aplicación se ha basado en el patrón arquitectónico MVC explicado anteriormente.

La aplicación del proyecto se ha basado en el patrón Modelo-Vista-Controlador dado que el objetivo es la separación clara de las responsabilidades, y no la de desacoplar la vista de la presentación porque añade una dificultad que no es necesario abordar dado que la aplicación consta de dos pantallas muy simples y no se realizan pruebas automáticas. La aplicación no se ajusta exactamente a este patrón debido a que una parte de la aplicación necesita de una actualización continua de los elementos de la Vista sin interacción del usuario, sin embargo, se ha ajustado lo máximo posible al patrón MVC. A continuación, se va a explicar el diseño y la implementación de la aplicación siguiendo la estructura del patrón MVC.

Respecto a la estructura de la aplicación, como puede observarse en la Figura 5.1, consta de tres paquetes; Modelo, Vista y Controlador y dos clases que ejecutan y configuran la aplicación.

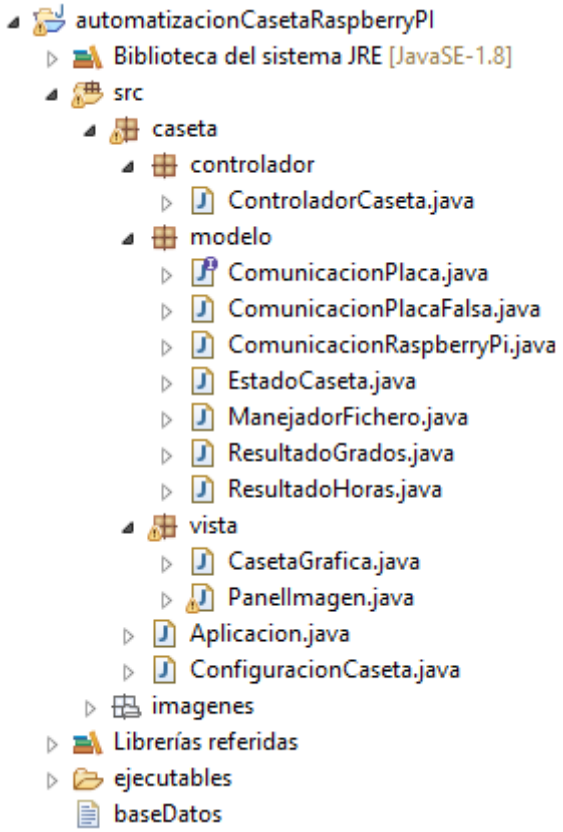


Figura 5.1 Estructura de la aplicación

❖ Clase Aplicación

En primer lugar, la aplicación se lanza desde esta clase, que es la clase principal ya que es la única clase que contiene el método main() y a partir de ella se instancia la clase de la interfaz gráfica que es la encargada de mostrar la pantalla y se realiza la configuración de la aplicación creando una instancia de la clase **ConfiguracionCaseta**.

❖ Clase ConfiguracionCaseta

La clase **ConfiguracionCaseta** es el creador concreto de la factoría dado que es la clase que devuelve una instancia concreta de un producto concreto. En esta clase es donde se deben realizar las instanciaciones de diferentes placas que podrían ser alternativas a la Raspberry Pi, como por ejemplo Arduino, Labjack, etc. o diferentes versiones de la misma placa pero que requiere una configuración o un comportamiento diferente. El código de esta clase se muestra a continuación, en el cual se observa que, para configurar la caseta, se necesita pasar por argumentos de la aplicación el tipo de placa que se desea utilizar para que la aplicación funcione con ese tipo determinado de placa. En el caso en el que no se haya pasado ningún argumento la implementación que se pasa es **ComunicacionPlacaFalsa** la cual más adelante será explicada. De esta forma se logra una capa más de abstracción sobre la placa que se está utilizando con el

objetivo de que si en un futuro este proyecto se implementara con otro tipo de placa que no fuese Raspberry Pi, solo debería crear una clase producto concreto con el código correspondiente a su comunicación con el material de campo y añadir el nombre en esta clase.

```
public class ConfiguracionCaseta {

	private String[] argumentos;

	public ConfiguracionCaseta(String[] args) {
		argumentos = args;
	}

	//Patron factory
	public ComunicacionPlaca getPlaca() {

		ComunicacionPlaca comunicacionPlaca;
		String tipoPlaca = getTipoPlaca();

		if(tipoPlaca.equals("raspberry")) {
			comunicacionPlaca = new ComunicacionRaspberryPi();
		} else {
			comunicacionPlaca = new ComunicacionPlacaFalsa();
		}
	}

	private String getTipoPlaca() {
		String tipoPlaca = "";
		if(argumentos.length == 0) {
			tipoPlaca = "falsa";//se inicializará la placa falsa
		}else {
			tipoPlaca = argumentos[0];
		}
		return tipoPlaca;
	}
```

Cabe destacar que esta clase no implementa la interfaz del creador porque no se ha creado una interfaz de creador ya que no resulta necesario crear dicha interfaz si sólo va a haber una implementación de ella.

5.1.1 Modelo

En el paquete Modelo que se observa en la Figura 5.2 se almacenan las clases relacionadas con la adquisición de datos, el manejo de la base de datos, el estado interno de la caseta y las representaciones de los datos necesarios con los que trabaja el sistema. Este modelo cumple con las responsabilidades que se han comentado en el apartado en el que se explica MVC.

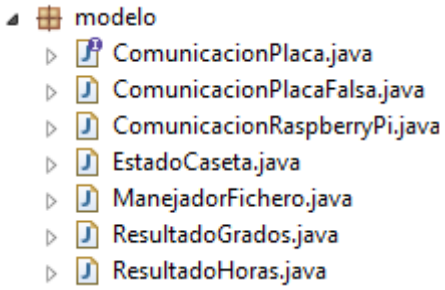


Figura 5.2 Paquete Modelo

❖ Clase ManejadorFichero

En primer lugar, la función de acceder a la capa de almacenamiento de datos se realiza mediante la clase **ManejadorFichero** la cual maneja la lectura y la escritura de los puntos de consigna que se guardan en la base de datos. Concretamente en esta clase se implementan los métodos de lectura y de escritura de las variables que deben ser guardadas en el caso en el que el programa tuviera que ejecutarse de nuevo basándose en los valores de las variables que se habían modificado en la anterior ejecución. En la Figura 5.3 se puede observar el contenido del fichero que representa la base de datos llamado "baseDatos" en los que se almacenan los valores de las variables "horaComidaManyana", "horaComidaTarde", "gradosVentilador" y "gradosMantaElectrica".

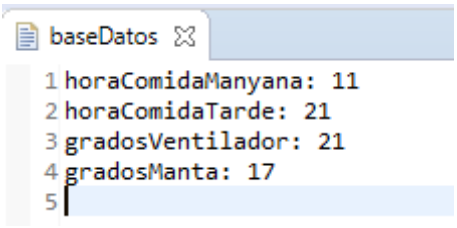


Figura 5.3 Fichero Base de datos

En segundo lugar, la función de almacenar toda la información de la aplicación, así como la información interna del programa se ve implementada en la clase **EstadoCaseta** la cual como su propio nombre indica almacena toda la información referente al estado de los actuadores, las horas de alimentación y los grados del ventilador y de la manta eléctrica, cabe destacar que es la clase que se comunica con la clase **ManejadorFichero** para transmitir una lectura o una escritura en la base de datos.

❖ Clase ComunicacionPlaca

Para la comunicación con las entradas y salidas de la placa, es decir, para la lectura de los sensores y la activación o desactivación de los actuadores, se implementa una interfaz que establezca los métodos que deben tener todas clases que implemente la comunicación entre la placa y la

aplicación. De esta forma se abstrae la comunicación con las posibles placas del controlador de la caseta ya que son independientes y en el caso en el que se quiera hacer otro caso de otra placa como por ejemplo Arduino, solamente se debe crear una clase que implemente la interfaz de **ComunicacionPlaca** e implementar los métodos conforme la configuración con la placa Arduino, sin cambiar más clases de la aplicación, excepto la clase ya nombrada **ConfiguracionCaseta** donde solo se debe introducir el nombre de la placa. Por lo tanto, se trata del elemento producto de una factoría ya que es la interfaz de objetos placa que se van a crear.

La clase **ComunicacionPlaca** contiene todos los métodos los cuales que pueden observarse en el siguiente código, son necesarios para obtener datos de los sensores que cualquier placa debería de proveer, como por ejemplo la humedad, temperatura, y el nivel de la alimentación y el agua y para activar y desactivar los actuadores. Los datos que son emitidos por la placa son gestionados mediante una de las implementaciones de la interfaz **ComunicacionPlaca**.

```
public interface ComunicacionPlaca {
	void inicializacionPines();
	void activarMantaElectrica();
	void desactivarMantaElectrica();
	void activarVentilador();
	void desactivarVentilador();
	void activarValvulaAgua();
	void desactivarValvulaAgua();
	void activarValvulaComida();
	void desactivarValvulaComida();
	int getHoraActual();
	int getTemperatura() throws IOException;
	int getHumedadRelativa() throws IOException;
	boolean getNivelBajoAgua();
	boolean getNivelAltoAgua();
	boolean getNivelBajoComida();
	boolean getNivelAltoComida();
	boolean getNivelDeposito();
	void cierre();
}
```

En resumen, es una abstracción de la comunicación con la placa de la aplicación en sí de manera, que si se necesita modificar algo relacionado con una placa solo se tiene que ir a su implementación concreta y modificarla. Además, gracias a esta abstracción es posible la completa separación del funcionamiento de la aplicación y de la placa se está utilizando ya que a la aplicación solo le interesa saber los valores de los sensores, no le interesa qué tipo de placa se ha utilizado para obtener dichos valores.

- **Clase ComunicacionRaspberry**

Se trata del producto concreto de la factoría por lo que implementa los métodos definidos por la interfaz **ComunicacionPlaca** con la correspondiente configuración de la Raspberry Pi. Para la interacción con los pines de comunicación Entrada/Salida de Propósito General (GPIO) de la placa Raspberry Pi se utiliza la librería Pi4J [18] de la cual se ha explicado su instalación en el apartado Librerías. El esquema de numeración de los pines GPIO que define la librería Pi4J es el mostrado en la Figura 5.4.

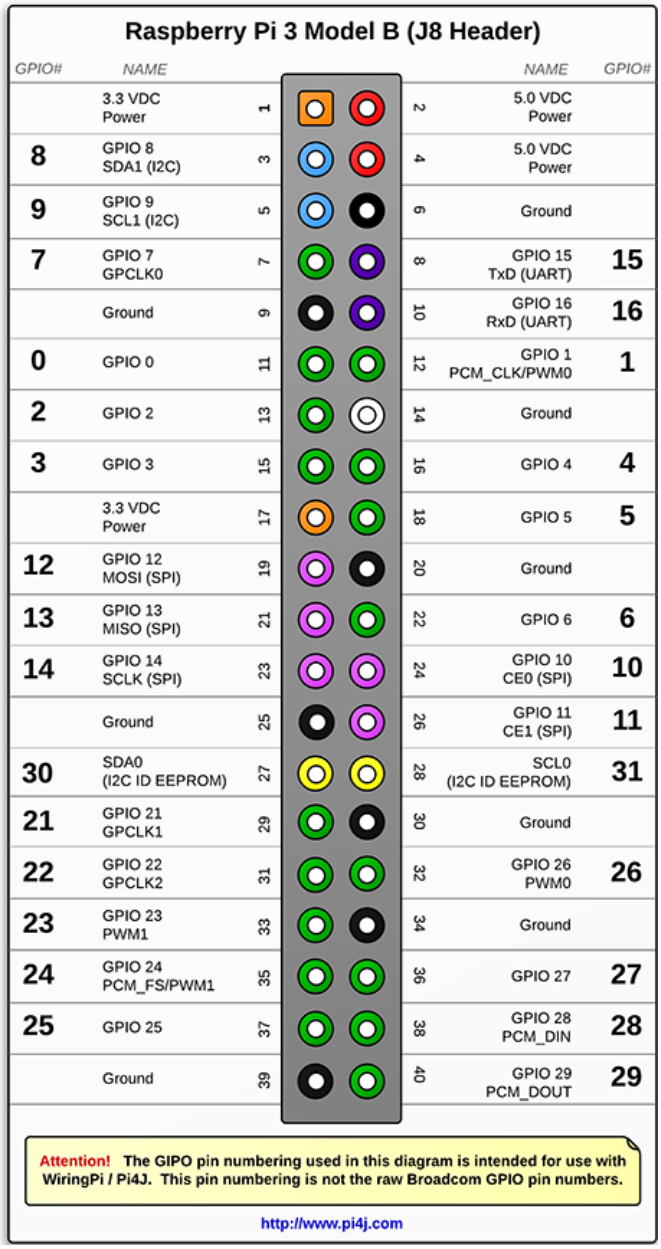


Figura 5.4 Esquema numeración Pi4J

A continuación, se expone los pines utilizados y su utilidad en la aplicación desarrollada:

- Pin 16: sensor humedad y temperatura DHT11. En el código en Python el pin correspondiente al sensor DHT11 es el pin 15.
- Pin 8: Manta eléctrica.
- Pin 9: Ventilador.
- Pin 7: Válvula de agua.

- Pin 2: Válvula comida.
- Pin 12: Sensor de nivel que representa el nivel bajo de agua.
- Pin 13: Sensor de nivel que representa el nivel alto de agua.
- Pin 14: Sensor de nivel que representa el nivel bajo de comida.
- Pin 21: Sensor de nivel que representa el nivel alto de comida.
- Pin 22: Sensor de nivel que representa el nivel del depósito de comida.

La librería Pi4J para la interacción con los pines GPIO proporciona una multitud de métodos entre los que se destacan los siguientes;

```
ControladorGpio = GpioFactory.getInstance();
```

De esta manera se crea la instancia del controlador GPIO. Solo se instancia una única instancia de controlador GPIO y esa instancia es compartida en todo el proyecto.

```
ControladorGpio.shutdown();
```

Este método es utilizado en el cierre de la aplicación y de esta forma se apaga el controlador de entradas y salidas de propósito general, es decir, detiene forzosamente todas las actividades/hilos GPIO cerrando el controlador GPIO.

Los métodos relacionados con las salidas digitales son los mostrados a continuación, se presentan como son realmente utilizados en el código de la aplicación, en este caso se presenta el uso de los métodos con la salida digital “MantaElectrica”.

```
pinMantaElectrica = ControladorGpio.provisionDigitalOutputPin(
RaspiPin.GPIO_08, "MantaElectrica", PinState.HIGH);
```

Mediante este método se especifica que se trata de una salida digital y como argumentos se introduce el pin en el que está conectado el actuador, una cadena de caracteres en la que se puede especificar el nombre del componente conectado y el estado inicial del pin.

```
pinMantaElectrica.setShutdownOptions(true, PinState.HIGH);
```

De esta forma se especifica que el estado del pin sea el especificado en el segundo argumento cuando se apague/cierre el programa. Esta configuración es aplicada automáticamente al pin cuando se termina la aplicación asegurándose así del apagado del actuador.

```
pinMantaElectrica.high(); //activar salida (excepto para los relés que es a la inversa)

pinMantaElectrica.low(); //desactivar salida (excepto para los relés que es a la inversa)
```

Con estos métodos se logra modificar el estado de la salida, concretamente, se cambia el estado de la salida a alto si se invoca el método .high() y a bajo si se invoca el método .low(). Cabe destacar que el funcionamiento de la placa de relés que se usa en este proyecto es a la inversa, dado que hace uso de la lógica negativa, es decir, cuando se desee activar el relé de debe desactivar el correspondiente pin de salida, por lo tanto, cuando se invoque el método .high() se desactivará el relé y cuando se invoque el método .low() se activará el relé. Por lo que como tratamos con relés la lógica es inversa, pero con salidas digitales funciona en modo normal.

Respecto a los métodos que hacen referencia a las entradas digitales se presentan, de la misma forma que los métodos relacionados con las salidas digitales, como son realmente utilizados en el código de la aplicación, en este caso se presenta el uso de los métodos con la entrada digital "nivel bajo agua".

```
pinNivelBajoAgua = ControladorGpio.provisionDigitalInputPin(
RaspiPin.GPIO_12,"NivelBajoAgua");
```

Mediante este método se especifica que se trata de una entrada digital y como argumentos se introduce el pin en el que está conectado el sensor y una cadena de caracteres en la que se puede especificar el nombre del componente conectado.

```
boolean estado = pinNivelBajoAgua.isHigh();
```

Con el método .isHigh() se puede leer el valor de la entrada digital especificada, concretamente devuelve true en el caso de que la entrada digital esté activa o false en caso contrario. También existe el método .isLow() en el que se aplicaría la metodología inversa.

```
pinNivelAltoComida.setShutdownOptions(true);
```

Al igual que el método setShutdownOptions() de la salida digital, en el momento en el que se cierre el programa se desactiva el pin por lo que no se realizan más lecturas hasta su activación.

```
ControladorGpio.unprovisionPin(pinMantaElectrica);

ControladorGpio.unprovisionPin(pinNivelBajoAgua);
```

El método unprovisionPin() realiza la operación inversa a la provisión de un pin, de esta manera se deshace toda la configuración que se haya realizado sobre el pin GPIO especificado en el argumento.

En cuanto a la lectura de las entradas digitales y la escritura en las salidas digitales se han realizado con los métodos proporcionados por la librería Pi4J mencionados anteriormente, excepto la lectura de la entrada digital a la que está conectado el sensor DHT11. El sensor DHT11 como se ha comentado anteriormente, se trata de un sensor que proporciona la temperatura y la humedad relativa, el problema reside en que tanto la temperatura y la humedad relativa son valores analógicos, no son digitales, por lo tanto, el sensor DHT11 no es un sensor digital que transmite un uno o un cero, sino es un sensor que transmite datos en binario. Para la lectura de dichos datos mediante los pines GPIO se ha utilizado un código ya existente en el lenguaje de programación Python [21]. Se ha realizado la lectura de este sensor en Python debido a que Java no es un lenguaje de tiempo real porque se ejecuta sobre la máquina virtual, por lo que no es suficientemente rápido para leer los datos emitidos por el sensor, para dicha lectura se pueden utilizar los lenguajes de programación Python o C. La lectura de dicho sensor se realiza en dos fases, la primera fase es la obtención de la temperatura y de la humedad mediante el código en Python y la segunda fase es la obtención de dichos valores en un formato determinado en Java mediante la ejecución del código en Python en el código de la aplicación Java.

❖ Clase ComunicacionPlacaFalsa

Para probar la aplicación se ha creado una instancia de **ComunicacionPlacaFalsa** la cual implementa los métodos definidos por la interfaz **ComunicacionPlaca** donde los métodos devuelven valores que se han definido intencionadamente para probar si los métodos realizan lo deseado, por ejemplo, el método getTemperatura() se ha definido que devuelva 20 grados. De esta manera no es necesario disponer de la placa para ejecutar la aplicación, por lo que se evita tener que comunicar con las placas para desarrollar partes de la aplicación que no tienen relación con la placa como es el caso del desarrollo de la interfaz gráfica sin necesidad de ejecutar la aplicación en la Raspberry Pi, o comprobar que el comportamiento de la aplicación es el correcto poniendo estados específicos, como por ejemplo una temperatura concreta. Por lo que se logra probar la aplicación en la herramienta de desarrollo Eclipse sin tener que ser probarlo desde la Raspberry Pi agilizando así el desarrollo de la aplicación.

❖ Clase ResultadoGrados

Esta clase se trata de una estructura de datos que contiene dos elementos; un boolean valido que es utilizado para saber si el valor de los grados que se desea poner como punto de consigna está dentro el rango especificado, en el caso en el que no esté, devuelve falso y en el caso contrario devuelve verdadero. El segundo elemento int grados es el punto de consigna que se desea poner mediante la interfaz en el ventilador y en la manta eléctrica.

❖ Clase ResultadoHoras

La clase **ResultadoHoras** es similar a la clase **ResultadoGrados** excepto la variable principal a gestionar, en vez de grados se trata de horas. Al igual que la clase **ResultadoGrados** se trata de una estructura de datos que contiene dos elementos; un boolean valido que indica si la hora se encuentra dentro de su rango especificado y el segundo elemento int horas es la hora especificada en los horarios de alimentación.

5.1.2 Vista

I. Interfaz de usuario

La Vista es la interfaz que se muestra al usuario y con la que interactúa. La interfaz de la aplicación se puede observar en la Figura 5.5, donde en la parte superior izquierda se muestra la temperatura, la humedad relativa y el nivel del depósito de comida mientras que en la parte inferior izquierda se muestra la configuración del comedero que se puede elegir mediante el menú desplegable el modo de funcionamiento y dependiendo del modo de funcionamiento escogido se muestran habilitados o deshabilitados los horarios de comida establecidos. En la parte superior derecha se puede observar la configuración referente al ventilador que dependiendo del modo de funcionamiento seleccionado se muestran habilitados o deshabilitados los botones de modificación de los grados de consigna de activación del ventilador y bajo de la configuración del ventilador se observa la configuración del bebedero que no tiene asociados ningún campo a configurar sea cual sea el modo de funcionamiento. Por último, en la parte inferior derecha se muestra la configuración de la manta eléctrica que se puede elegir mediante el menú desplegable y al igual que en la configuración del ventilador dependiendo del modo de funcionamiento seleccionado se muestran habilitados o deshabilitados los botones de modificación de los grados de consigna de activación de la manta eléctrica.

Figura 5.5 Interfaz de la aplicación

II. Usabilidad

Respecto a la usabilidad, se han tomado varias decisiones para que el uso de la aplicación sea fácil e intuitivo. En primer lugar, se decide poner una pantalla táctil en la caseta para que los usuarios puedan ver los valores de la temperatura y humedad, puedan observar la configuración de la que dispone actualmente la caseta y modificar dicha configuración. Para que la configuración de la caseta pueda ser observada y modificada por los usuarios se proporcionan textos, botones y desplegables separados con una distancia y un tamaño concreto para que se puedan cambiar mediante la pulsación en la pantalla táctil con cualquier dedo sin dificultad. Respecto a los desplegables, se han utilizado para que el usuario pueda elegir entre diferentes opciones y ocupar el menor espacio posible en la pantalla ya que mientras no se pulse en el desplegable, solo aparece la opción actualmente seleccionada, por lo que si se desea cambiar la opción tan solo se debe pulsar sobre el desplegable, el cual dispone de una flecha en su parte derecha indicando que hay más campos a seleccionar, y aparecerán las tres opciones disponibles. Por otro lado, se ha utilizado botones con un icono de una flecha hacia arriba para subir las horas o la temperatura y un botón con el icono de una flecha apuntando hacia abajo para bajar las horas o la temperatura. En segundo lugar, se ha elegido la distribución de los componentes alrededor de la caseta y el material de campo con el objetivo de que sea más fácil e intuitivo su uso sin dar una sensación de agrupación de componentes sin espacio en la pantalla. Por lo tanto, se ha incorporado un fondo que muestra una caseta y el respectivo material de campo para dar una sensación de espacio y crear una interfaz más intuitiva y amigable.

III. Creación interfaz gráfica

Como se ha comentado al principio del apartado de Software el complemento utilizado para crear la interfaz gráfica de la aplicación es *WindowBuilder*. Estos complementos se pueden encontrar e instalar en el entorno Eclipse, en Ayuda - Instalar Nuevo Software como se observa en la Figura 5.6.

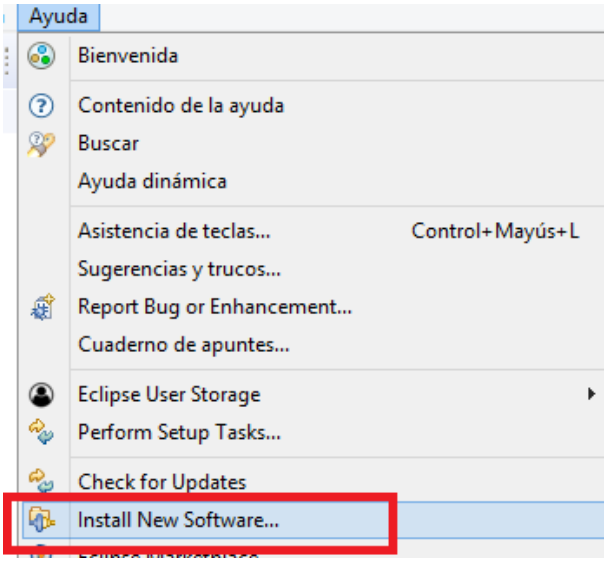


Figura 5.6 Instalar complemento en Eclipse

La interfaz gráfica se ha realizado mediante *WindowBuilder* que está compuesto por SWT Designer y Swing Designer. Para obtener dichas funcionalidades se han importado, como se muestra en la Figura 5.7, los complementos correspondientes a Swing Designer y a WindowBuilder.

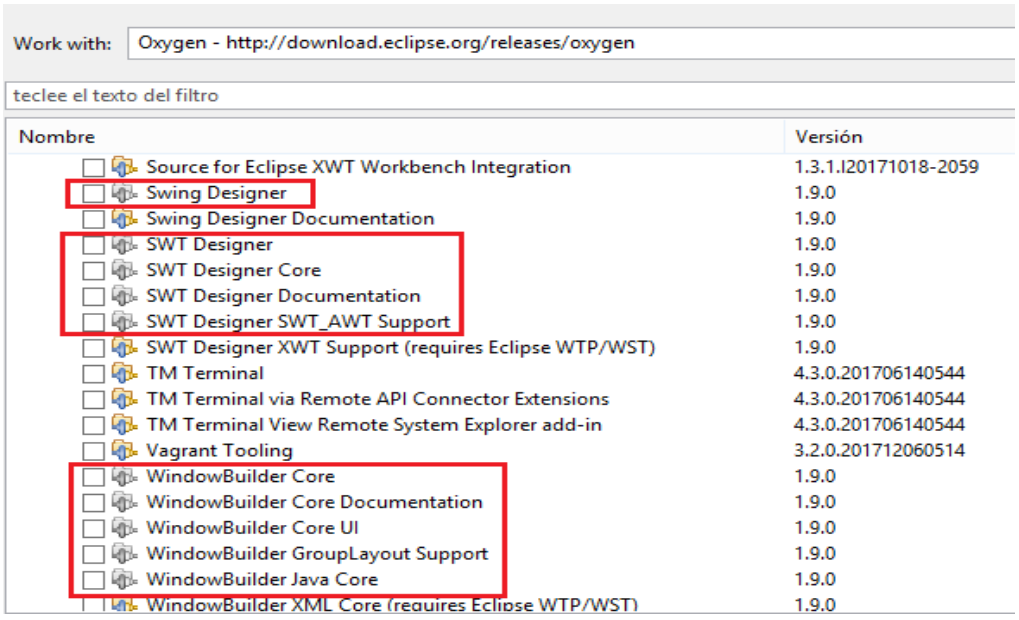


Figura 5.7 Complementos instalados

Tras la instalación del complemento *WindowBuilder*, para crear la interfaz gráfica con dicho complemento se ha de pulsar en el botón derecho del ratón, se ha de seleccionar Nueva y Otras como se pueden observar en la Figura 5.8.

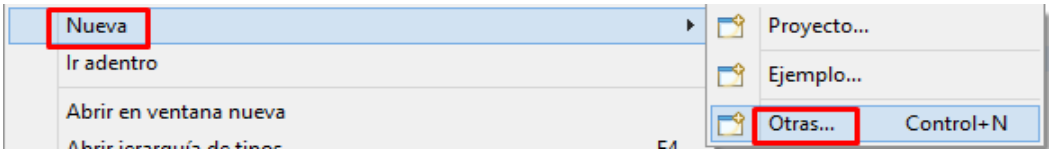


Figura 5.8 Crear nueva clase

Aparece una ventana como la de la Figura 5.9, dentro del complemento *WindowBuilder* se selecciona la clase de tipo *ApplicationWindow*.

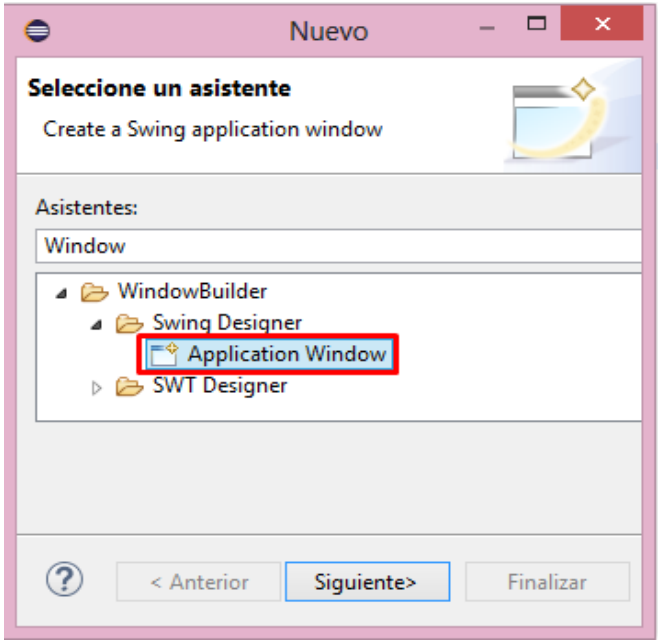


Figura 5.9 Crear clase de interfaz con complemento

Por último, aparece una ventana como la de la Figura 5.10 en la que se escribe el nombre que se desea asignar a la clase que representa la interfaz, es este caso **CasetaGrafica**.

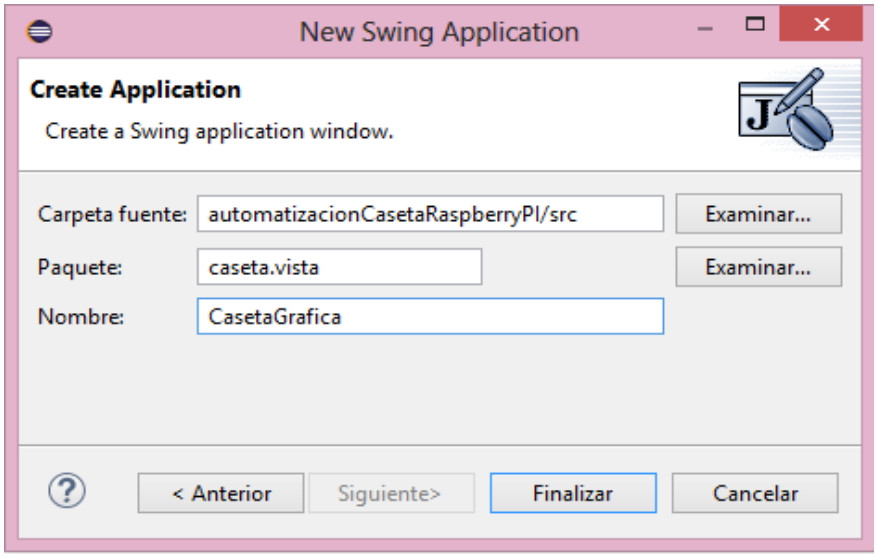


Figura 5.10 Asignación de nombre de la interfaz

WindowBuilder es un diseñador bidireccional de Interfaz Gráfica de Usuario (GUI) de Java que presenta dos vistas: Diseño y Fuente. Este diseñador bidireccional genera directamente código Java que puede modificarse en la Vista de Diseño o directamente en la Vista de Fuente, es decir, puede editar el código fuente o usar un diseñador gráfico para modificar la interfaz de usuario. Sincroniza ambas representaciones, por lo que todos los cambios realizados en el código fuente se reflejan en el diseñador gráfico y viceversa.

Por lo tanto, la **CasetaGrafica** por una parte se puede visualizar y gestionar la ventana que el usuario puede observar e interactuar mediante la vista de Diseño y por otra parte se puede agregar funcionalidades, modificar y dar forma al código derivado de la interfaz gráfica mediante la vista de Fuente.

Respecto a la vista Diseño, se puede contemplar en la pestaña *Design*, permite la creación del diseño de una interfaz para ello dispone de una barra de herramientas a la izquierda como se puede observar en la Figura 5.11 donde se pueden seleccionar todo tipo de elementos, entre ellos los botones, los desplegables, los textos, etc.

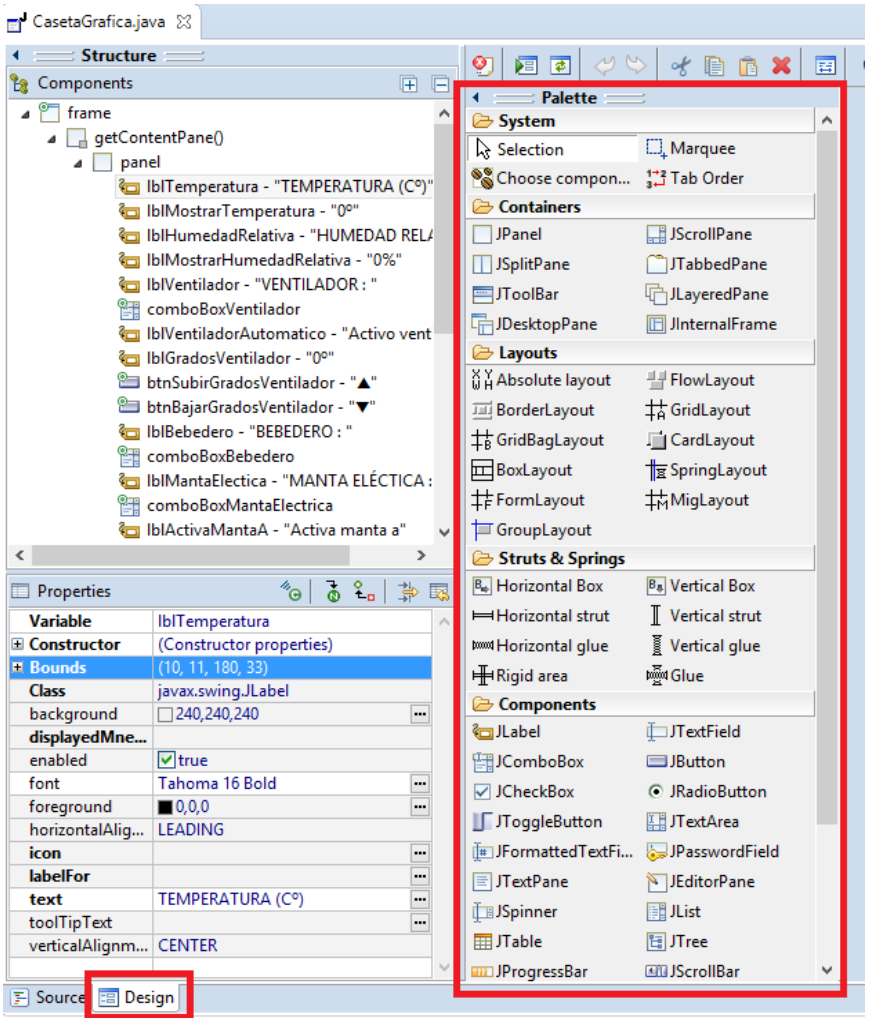


Figura 5.11 Pestaña Diseño

Para utilizar alguno de estos elementos en la interfaz solamente se tiene que seleccionar el elemento y desplazarlo hasta el lugar deseado como se observa en a Figura 5.12. En el caso de los botones y de los desplegables, para asociarle acciones cuando sean pulsados se debe pulsar doblemente sobre ellos y en la pestaña de Fuente aparece un escuchador asociado a dicho elemento.

Figura 5.12 Colocación elementos de la interfaz

Respecto a la vista Fuente, se puede observar en la pestaña *Source*, permite la creación de código relacionado con la interfaz gráfica y la modificación de la interfaz gráfica mediante código.

IV. Hilos en Java

Un hilo es un flujo secuencial de ejecución de una parte de código dentro de un programa. Una de las razones por las que se utilizan los hilos es la necesidad de la ejecución de varias tareas simultáneamente en un mismo programa. Por lo que para lograr que se realicen diferentes tareas al mismo tiempo o que el programa no se quede parado mientras realiza una tarea compleja, se emplean los hilos (*Threads* en inglés) [22]. Existen dos maneras de crear un hilo una forma es la que se presenta en el siguiente código y se trata de heredar de la clase *Thread* (*extends Thread)* y definir el método *run* () dentro del cual se introduce el código que se desea que se ejecute en un hilo separado. Para el arranque del hilo, es decir, para que se ejecute el código del método *run* (), se debe llamar al método *start* ().

```
public class CasetaGrafica extends Thread {
	public CasetaGrafica(ConfiguracionCaseta configuracion) {
		inicializacion(configuracion);
		start(); //Arranque del hilo creado
	}
public void run() {
//Código que ejecuta el hilo creado en segundo plano
}
```

La otra manera de crear un hilo es implementar la clase *Runnable* (*implements Runnable*) la cual sólo define un método, el método *run* (), lo que fuerza a que se implemente el método *run*() con las acciones a ejecutar del hilo y el hilo se pasa mediante argumentos en el constructor de *Thread*.

```
public class CasetaGrafica implements Runnable {
	public void run() {
		//Código que ejecuta el hilo creado en segundo plano
	}
	public static void main (String args []) {
		Thread thread = new Thread (this);
		thread.start();
	}
}
```

En todos los programas existe al menos un hilo, llamado hilo principal, el cual ejecuta el cuerpo del programa y tiene su inicio en la primera línea del método *main*(), que es donde se inicia una aplicación Java. A través del hilo principal se crean los hilos secundarios que comienzan su ejecución de forma concurrente al hilo principal y la creación puede ser debida a la ejecución de una tarea adicional al flujo del programa principal o al aprovechamiento del hardware disponible realizando la misma tarea segmentada en menor tiempo.

Existen dos planos de ejecución para un programa; el primer plano y el segundo plano.

El primer plano, *foreground* en inglés, es lo que el usuario visualiza, concretamente, es donde se ejecuta el hilo principal que trabaja con la interfaz gráfica que visualiza el usuario. En este plano es donde el usuario interactúa directamente con la aplicación por lo que no se debe realizar tareas que puedan bloquear este hilo que se ejecuta en primer plano porque entonces la aplicación irá lenta y el usuario se percatará de que la aplicación va lenta. Hay que tener en cuenta que cualquier modificación de un componente gráfico se ha de realizar en el primer plano.

El segundo plano, *background* en inglés, es donde se ejecuta los hilos "secundarios" y se ejecuta paralelamente al hilo principal. En segundo plano es donde debe ejecutarse los hilos con ejecuciones pesadas las cuales pueden llegar a tardar mucho tiempo y bloquean el hilo en el que se ejecutan por lo que son ideales para ejecutarse paralelamente al hilo principal sin bloquearlo ni ralentizarlo. Se debe tener en cuenta que en segundo plano se pueden ejecutar tareas que tarden mucho tiempo siempre y cuando la interfaz de usuario no deba mostrar valores dependientes de dichas tareas, en ese caso el usuario sí que se percatará.

V. Implementación

En el paquete de la vista se almacenan las clases relacionadas con la interfaz gráfica y todos sus componentes como se observa en la Figura 5.13. La vista recibe datos del modelo y los muestra en la interfaz, reacciona ante las acciones del usuario y realiza las peticiones correspondientes al controlador.

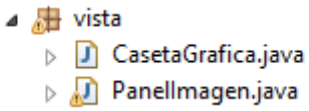


Figura 5.13 Paquete Vista

❖ Clase CasetaGrafica

En lo que respecta a la aplicación del proyecto, se trata de una clase que dispone de código asociado a la inicialización de la vista y a la gestión de los eventos que se producen por la interacción del usuario con la interfaz gráfica.

En la aplicación desarrollada se crea un hilo secundario que realiza dos funciones, una de ellas es la gestión de avisar al controlador si se ha pulsado el modo automático y la otra función es la actualización de los valores que muestra la interfaz, concretamente se crea una referencia al controlador y mediante su invocación, el hilo se encarga de preguntar al controlador los valores de la temperatura, la humedad, el nivel del depósito, los grados de consigna del ventilador y de la manta eléctrica y sustituir dichos valores en la interfaz logrando de esa manera que estén actualizados. Ambas funciones están introducidas en un bucle infinito que se repite cada medio segundo. Para mostrar los datos en la interfaz se debe volver a primer plano dado que para modificar o consultar la interfaz se debe realizar en primer plano como se ha comentado anteriormente, por lo que el hilo secundario primero consulta los valores al controlador en segundo plano y a la hora de mostrar dichos valores en la interfaz se pasa a primer plano y tras mostrar los valores vuelve a segundo plano. En el siguiente código se puede observar cómo se realiza el cambio de segundo a plano a primer plano y se muestran los valores en la interfaz. Este cambio de plano se realiza mediante el método *invokeLater* () de la clase Swing el cual simplemente planifica la tarea y regresa. El método *invokeLater* () recibe por parámetros un objeto *Runnable* que se ejecutará en el hilo principal que usa Swing para mostrar la interfaz.

```
SwingUtilities.invokeLater(new Runnable() {
  public void run() {
      lblMostrarTemperatura.setText(String.valueOf(temperatura) + "º");
      lblMostrarHumedadRelativa.setText(String.valueOf(humedadRelativa)+"%");
      lblMostrarDepositoComida.setText(controlador.autoDeposito());
      lblGradosVentilador.setText(String.valueOf(gradosVentilador) + "º");
      lblGradosManta.setText(String.valueOf(gradosManta) + "º");
  }
});
```

La clase **CasetaGrafica** además de mostrar información también gestiona los eventos que se producen, dichos eventos son gestionados mediante los escuchadores (*listeners* en inglés) asociados a los elementos de la interfaz. Los elementos que se encuentran en la interfaz que tengan asociados escuchadores son los botones y los desplegables (*ComboBox*). En el siguiente código se muestra el escuchador de un botón, en el momento en el que usuario pulse en el botón se ejecutará el código asociado. En el ejemplo mostrado, cuando se pulse el botón de bajar grados del ventilador se invocará el método bajarGradosVentilador() del controlador, el cual se ha comentado en el apartado del modelo en la clase de **ResultadoGrados**, dicho método devolverá el valor correspondiente de los grados a mostrar, en el caso en el que no sea válido es decir, sea el mínimo del rango establecido, el botón aparecerá deshabilitado y en el caso en el que sea válido el botón aparecerá habilitado por lo que podrá ser pulsado de nuevo para bajar un grado más. Posteriormente se muestra el número de grados del ventilador tras la operación aplicada y habilita el botón de subir grados por si estuviera deshabilitado dado que si estuviera deshabilitado sería porque había llegado al máximo de grados permitidos para configurar y al bajar un grado no se encuentra en el máximo, por lo que se puede pulsar de nuevo el botón de subir grados.

```
btnBajarGradosVentilador.addActionListener(newActionListener() {
 public void actionPerformed(ActionEvente) {
   try {
      ResultadoGradosresultadoGrados = controlador.bajarGradosVentilador();
      btnBajarGradosVentilador.setEnabled(resultadoGrados.esValido());
      lblGradosVentilador.setText(resultadoGrados.getGrados() + "º");
      btnSubirGradosVentilador.setEnabled(true);
   } catch (IOExceptione1) {
      e1.printStackTrace();
   }
 }
});
```

Antes de comentar el escuchador de un desplegable, cabe mencionar que los desplegables utilizados en la aplicación disponen, como se observa en la Figura 5.14, de tres modos de funcionamiento: Automático, Manual->Encender, Manual->Apagar. La funcionalidad de estos modos de funcionamiento es explicada en el apartado ya visto Solución Propuesta.

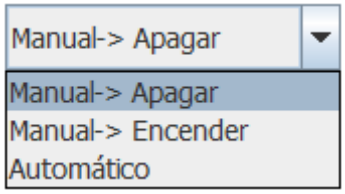


Figura 5.14 Modos de funcionamiento

En el código que se muestra a continuación se presenta el escuchador de un desplegable o *comboBox*, en el momento en el que usuario seleccione una opción se ejecutará el código asociado. En el ejemplo mostrado, cuando se seleccione el modo automático aparecerán habilitados todos los elementos asociados a la configuración del elemento a gestionar. Se consigue habilitar un elemento haciéndolo visible con el siguiente método: `lblVentiladorAutomatico.setEnabled(true);` y se deshabilita con el mismo método, pero con el argumento *false*, es decir, `lblVentiladorAutomatico.setEnabled(false);`. En el caso del ventilador y de la manta eléctrica se mostrarán los botones y los grados que representa el punto de consigna al que se activará el actuador, en el caso del bebedero se establecerá el modo automático y no aparecerá ningún elemento dado que no tiene configuración mientras que en el caso del comedero si se selecciona dicho modo aparecerán los botones y los textos asociados a la configuración de los horarios de comida de la mascota. Mientras que los otros dos modos en el momento en el que seleccionen aparecerán los elementos relacionados con el modo automático deshabilitados.

```
comboBoxVentilador.addActionListener(newActionListener() {
	publicvoidactionPerformed(ActionEvente) {
		if (comboBoxVentilador.getSelectedItem() != automatico) {
			controlador.ventiladorNoAutomatico();
			ventiladorAutomatico = false;
			if (comboBoxVentilador.getSelectedItem() == encender) {
				controlador.activarVentilador();
			} else {
				controlador.desactivarVentilador();
			}
			lblVentiladorAutomatico.setEnabled(false);
			lblGradosVentilador.setEnabled(false);
			btnSubirGradosVentilador.setEnabled(false);
			btnBajarGradosVentilador.setEnabled(false);
		} else {
			// MODO AUTOMÁTICO
			ventiladorAutomatico = true;
			lblVentiladorAutomatico.setEnabled(true);
			lblGradosVentilador.setEnabled(true);
			btnSubirGradosVentilador.setEnabled(true);
			btnBajarGradosVentilador.setEnabled(true);
		}
	}
});
```

Por último cabe destacar que el establecimiento de rangos de las horas de comida y de los grados de ventilador y de la manta eléctrica se gestionan en el momento en el que se pulsa el botón de subir o de bajar, en el momento en el que arranca la aplicación como no se ha pulsado a ningún botón aparecen habilitados los botones aunque estén en el mínimo o en el máximo de un rango, por lo que se puede al pulsar botón de bajar o subir y bajaría o subiría la temperatura o la hora y mostraría horas y temperaturas fuera de su rango. Por lo que para solucionar dicho problema al final de la clase **CasetGrafica** en el método de inicializar se han realizado las comprobaciones mostradas en el siguiente código para asegurarse de que, si está en el máximo o en el mínimo de algún rango, que no se pueda sobrepasar.

```
//6-16
if(controlador.getHoraComidaManyana()==6)
	btnBajarHoraManyana.setEnabled(false);

if(controlador.getHoraComidaManyana()==16)
	btnSubirHoraManyana.setEnabled(false);
//de 17-5
if(controlador.getHoraComidaTarde()==17)
btnBajarHoraTarde.setEnabled(false);

if(controlador.getHoraComidaTarde()==5)
btnSubirHoraTarde.setEnabled(false);
//18-28
if(controlador.getGradosConsignaVentilador() == 18)
	btnBajarGradosVentilador.setEnabled(false);

if(controlador.getGradosConsignaVentilador() == 28)
	btnSubirGradosVentilador.setEnabled(false);
//16-26
if(controlador.getGradosConsignaManta() == 16)
	btnBajarGradosManta.setEnabled(false);

if(controlador.getGradosConsignaManta()== 26)
	btnSubirGradosManta.setEnabled(false);
```

❖ Clase PanelImagen

Esta clase se trata de una clase personalizada (*Custom class* en inglés) que extiende de **JPanel** y sobrescribe el método paint(). **Jpanel** es un objeto contenedor donde se pueden agrupar otros objetos como botones, desplegables, textos, etc. y de esta manera se facilita la movilidad de todos esos componentes relacionados de un lugar a otro solo moviendo el **JPanel** y no cada uno de los elementos que este contiene. Cabe destacar que **Jpanel** no es un marco, por lo que se ha de introducir, por ejemplo, dentro de una ventana **JFrame**. Esta clase se crea con el objetivo de visualizar como fondo de la interfaz la imagen de la caseta junto con su material de campo. Para

ello, como se puede observar en el código [23], se sobrescribe el método *paint*() , al cual se le pasa como argumento un objeto **Graphics** que es el contexto en el cual se pinta la imagen.

```
public class PanelImagen extends JPanel {
	ImageIcon imagen;
	String urlImagen;

	public PanelImagen(String urlImagen) {
		this.urlImagen = urlImagen;
	}

	@Override
	public void paint(Graphics graphics) {
		Dimension size = getSize();
		URL ubicacionImagen = getClass().getResource(urlImagen);
		imagen = new ImageIcon(ubicacionImagen);
graphics.drawImage(imagen.getImage(),0,0,size.width,size.height,null);
		setOpaque(false);
		super.paint(graphics);
	}
}
```

En cuanto al código de la clase personalizada **PanelImagen** se ciñe en obtener la imagen a partir de la ubicación de la imagen y pintar el fondo del **Jpanel** con dicha imagen. Para que la imagen sea encontrada cuando se exporte el proyecto debe crearse una carpeta dentro del proyecto e introducir en ella la imagen que se desea utilizar de fondo. Para crear la una carpeta para que contenga las imágenes deseadas se debe pulsar el botón derecho del ratón sobre la carpeta "src" y seleccionar Nueva y Otras.

Se selecciona Carpeta y por último se escribe el nombre que se desea asignar a la carpeta, es este caso "imagenes" como puede observarse en la Figura 5.15.

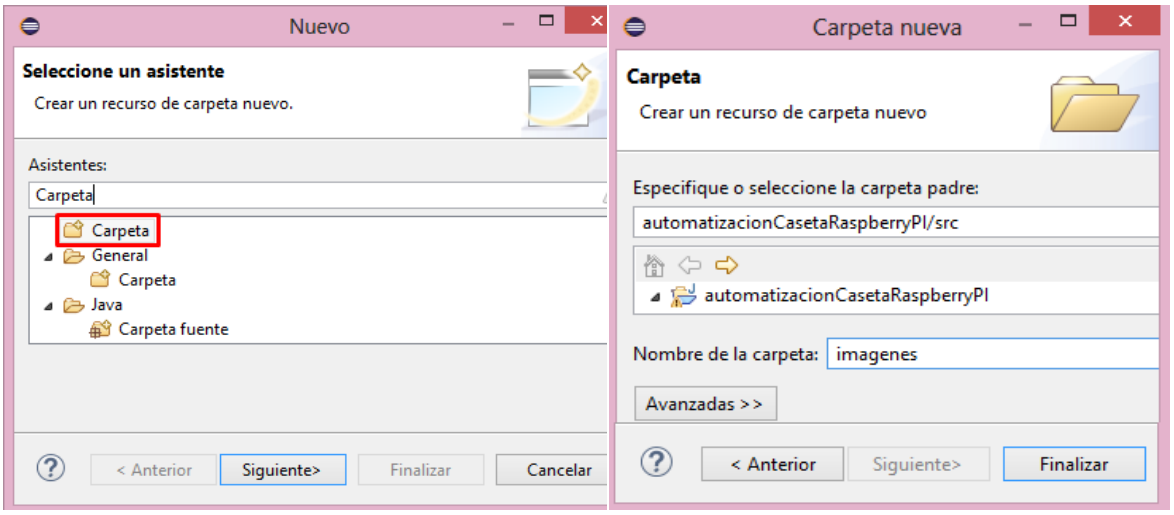


Figura 5.15 Crear nueva carpeta

Posteriormente la **CasetaGrafica**, como se puede observar en el siguiente código, crea un objeto **PanelImagen** con la ruta de la imagen que se desea establecer de fondo y este objeto panel se añade al marco de la interfaz, de manera que se observa la imagen de fondo de la interfaz.

```
PanelImagen panel = new PanelImagen("/imagenes/casetaPerroMedio.jpg");
panel.setLocation(0,0);
panel.setSize(1024,600);
frame.getContentPane().add(panel);
panel.setLayout(null);
```

❖ Clase CamaraGrafica

Esta clase se ocupa de mostrar las imágenes que graba la cámara en el instante actual. En el momento en el que se pulsa el botón de grabar que muestra la interfaz gráfica de **CasetaGrafica**, se invoca al método visualizar de la clase **CamaraGrafica** el cual inicia un proceso, como se puede observar el en código, que ejecuta un ejecutable llamado "preview.sh".

```
public void visible (boolean visible) throws IOException {
	frame.setVisible(visible);
	Runtime rt = Runtime.getRuntime();
	File archivo = new File("ejecutables/preview.sh");
	process = rt.exec("bash "+ archivo.getAbsolutePath());
}
```

El contenido de este ejecutable es el comando que aparece a continuación, el cual permite encender la cámara indefinidamente mostrando el contenido en una ventana emergente de tamaño 1024x545.

```
raspistill -t 0 --preview 0,0,1024,545
```

La interfaz gráfica de la clase **CamaraGrafica** se muestra en la Figura 5.16 y consta de un botón de Salir que permite cerrar la ventana de la cámara y muestra lo que visualiza la cámara en el instante actual.

Figura 5.16 Interfaz de la cámara

Concretamente a continuación se muestra el código asociado al botón de salir de la **CamaraGrafica** junto con el método terminarCamara() en los que se puede observar que en primer lugar deja de ser visible la ventana y que para apagar la cámara se invoca al método terminarCamara() en el cual obtiene el identificador del proceso que está ejecutando el comando de encender la cámara indefinidamente, y con dicho identificador se mata el proceso logrando que se apague la cámara.

```
btnSalirCamara.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e) {
		frame.setVisible(false);
		terminarCamara();
	}
});

public void terminarCamara() {
	try {
		Runtime rt = Runtime.getRuntime();
		Process psAux = rt.exec("pidof raspistill");
		BufferedReader bufferedReader = new BufferedReader(new
		InputStreamReader(psAux.getInputStream()));
		String pid = bufferedReader.readLine();
		rt.exec("sudo kill "+ pid);
	…
	}
```

5.1.3 Controlador

El controlador es el encargado de gestionar el flujo de información entre el Modelo y la Vista, exactamente recibe los eventos de entrada de la Vista, procesa dicha petición y realiza los cambios correspondientes sobre el Modelo. También se encarga de realizar las conversiones correspondientes para adaptar los datos a las necesidades del Modelo y de la Vista.

❖ **Clase ControladorCaseta**

El controlador de la aplicación en cuestión dispone de instancias a las clases del Modelo y éstas son invocadas cuando se necesita consultar o modificar algún dato o simplemente hacer uso de ellas como es el caso de las clases que representan estructuras de datos (**ResultadoGrados** y **ResultadoHoras**).

El usuario puede interactuar con las interfaces gráficas mediante botones y desplegables. Cuando el usuario interactúa con alguno de estos elementos, la Vista realiza una petición al Controlador y el Controlador es el que gestiona la operación correspondiente comunicándose con el Modelo cuando sea necesario. En el caso en el que el usuario pulse un botón, por ejemplo, el botón de subir hora de la comida por la mañana, las operaciones a realizar por parte del controlador son las que se muestran en el siguiente código y se listan a continuación;

1. Informar a la Vista de si debe estar habilitado el botón.
2. Comprobar si la hora seleccionada está en el límite de rango.
3. Realizar la operación correspondiente al botón modificando dicho dato en el modelo, en este caso subir una hora.
4. Almacenar en la base de datos dicho dato, comprobar si se debe activar dependiendo de la hora seleccionada y la hora actual.
5. Devolver un objeto con la información sobre la habilitación del botón y la hora tras aplicar la operación demandada.

```
public ResultadoHoras subirHoraComidaManyana() throws IOException {// de 6-16
    boolean mostrarBotonSubirHora = true;
    if (estadoCaseta.getHoraComidaManyana() == 15) //define el rango
            mostrarBotonSubirHora = false;
    estadoCaseta.setHoraComidaManyana(estadoCaseta.getHoraComidaManyana()+1);
    manejadorFichero.escribirHoraComidaManyana(
                                estadoCaseta.getHoraComidaManyana());

    if (estadoCaseta.getHoraComidaManyana() == this.getHoraActual())
            autoValvulaComida();

    return new ResultadoHoras(mostrarBotonSubirHora,
                                estadoCaseta.getHoraComidaManyana());
}
```

Sin embargo, si el usuario ha interactuado con el desplegable las operaciones a realizar por el controlador son diferentes a las operaciones de gestión de un botón. En el momento en el que un usuario selecciona una opción de un desplegable, por ejemplo, el modo automático del desplegable del ventilador, la Vista invoca a un método del Controlador que gestiona el modo automático del ventilador, el cual puede observarse a continuación en el código, que las acciones a realizar por el Controlador son las siguientes;

1. Obtener la temperatura actual del Modelo.
2. Comprobar si se debe de activar o desactivar el ventilador dependiendo de la temperatura actual y de los grados de consigna especificados.
3. Informar al Modelo de la actuación que se realiza.

```
public void autoVentilador() {
    int temperatura;
    try {
         temperatura = getTemperatura();
         if (estadoCaseta.getGradosConsignaVentilador() <= temperatura
                                 && estadoCaseta.isVentiladorActivado()) {
             activarVentilador();
             estadoCaseta.setVentiladorActivado(true);
             estadoCaseta.setVentiladorDesactivado(false);
         }
         if (estadoCaseta.getGradosConsignaVentilador()>temperatura
                                 && !estadoCaseta.isVentiladorDesactivado()) {
             desactivarVentilador();
             estadoCaseta.setVentiladorActivado(false);
             estadoCaseta.setVentiladorDesactivado(true);
         }
    } catch (IOException e) {
         e.printStackTrace();
    }
   }
```

5.2 Desarrollo hardware

En este apartado se muestra detalladamente como se ha conectado cada elemento electrónico entre sí y como se ha montado todo el conexionado sobre la caseta.

5.2.1 Conexionado hardware

A continuación, se explica cómo se ha realizado el conexionado de los dispositivos a la Raspberry Pi, tanto de las entradas como de las salidas. Cabe destacar que en el conexionado encontramos tanto la corriente continua, que es la corriente que fluye a lo largo de un circuito eléctrico en un solo sentido, como la corriente alterna, la cual fluye en ambos sentidos ya que cambia continuamente su polaridad, ésta última es la que permite el transporte de mayores cantidades de energía eléctrica y a mayor distancia. En este sistema se ha utilizado corriente alterna en las conexiones con los actuadores, los cuales necesitan un voltaje de 220V y no de 5V, que es lo máximo que puede proporcionar la Raspberry Pi. Mientras que, para la conexión con los sensores, la pantalla táctil y la placa de relés se hace uso de corriente continua, los cuales precisan una alimentación de 5V, excepto para el sensor de nivel de comida que precisa de 6V a 36V.

Por último, cabe destacar que se ha tenido que introducir una pila de 9V junto con un limitador de tensión a 5V para la alimentación de los dos sensores de nivel de líquidos dado que la Raspberry Pi no es capaz de alimentar la pantalla, el sensor de temperatura, la placa de relés y los sensores de nivel. Esta misma pila junto con un limitador de tensión a 6V, también alimenta los sensores de nivel de comida, dado que la Raspberry Pi solo puede proporcionar hasta 5V. El esquema de conexionado de la parte de la Raspberry Pi con los sensores y los actuadores se muestra en la Figura 5.17, este esquema es general por lo que no se muestra la pila y los limitadores de tensión.

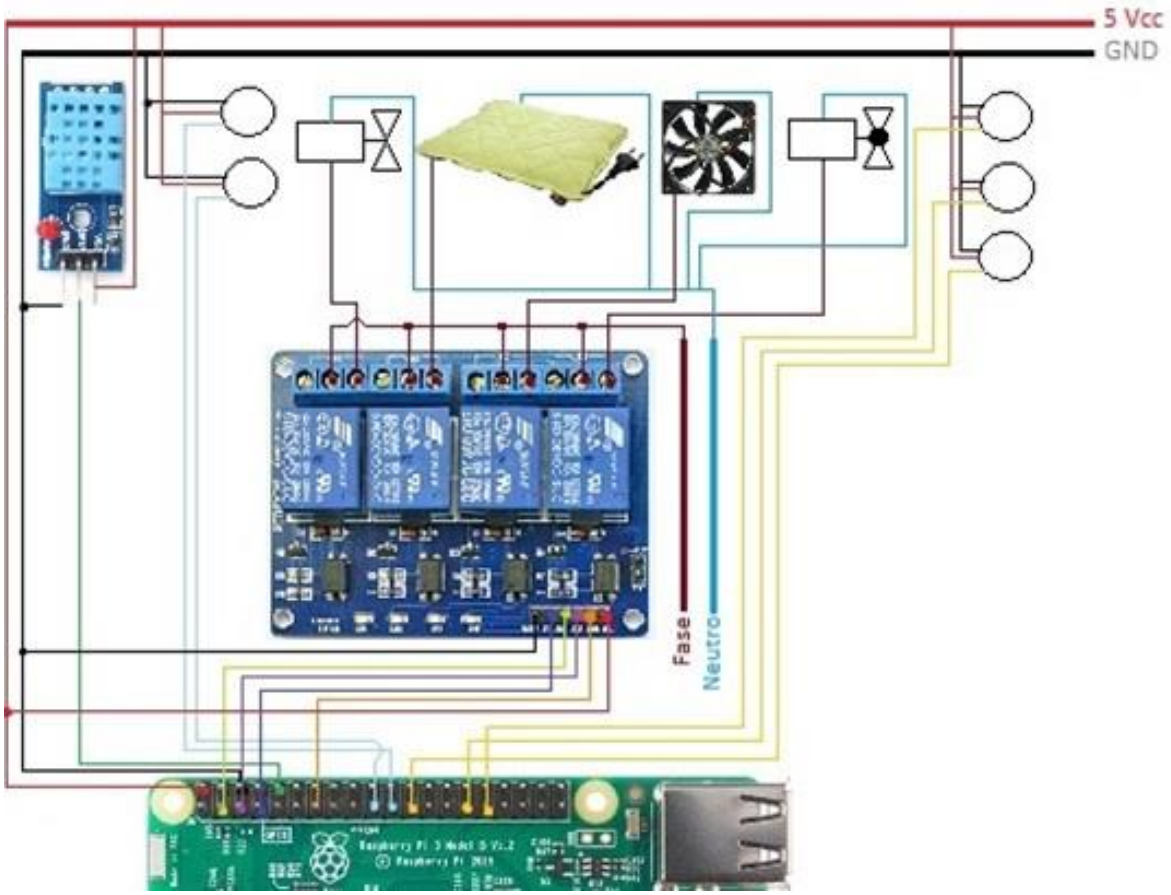


Figura 5.17 Conexionado parte sensores y actuadores

I. Entradas

En cuanto a la conexión de las entradas, se dispone en la Raspberry Pi de una entrada de datos para el sensor DHT11 y cinco entradas digitales para los sensores de nivel.

Sensor DHT11

Respecto al sensor DHT11, dispone de tres pines: Vcc, GND y DATA. Como se muestra en la Figura 5.18 y 5.19, se debe conectar el pin Vcc a uno de los pines de la Raspberry Pi que proporciona potencia positiva, ésta dispone de varios pines de salida de tensión de 3,3V y otros de 5V. Como el rango de voltaje que necesita el sensor es de 5V-24V, no se puede conectar al pin de la Raspberry Pi que proporciona 3’3V, por lo que se debe conectar al pin que proporciona 5V. El pin de GND se conecta a la potencia negativa, por lo que se conecta a uno de los pines de tierra de la Raspberry Pi, denominado como *Ground*. Por último, el pin DATA es el pin de datos, por el que se transmiten los datos de la temperatura y de la humedad relativa, por lo que se conecta con uno de los pines de entrada de datos de la Raspberry Pi para que realice la lectura de la temperatura y la humedad relativa.

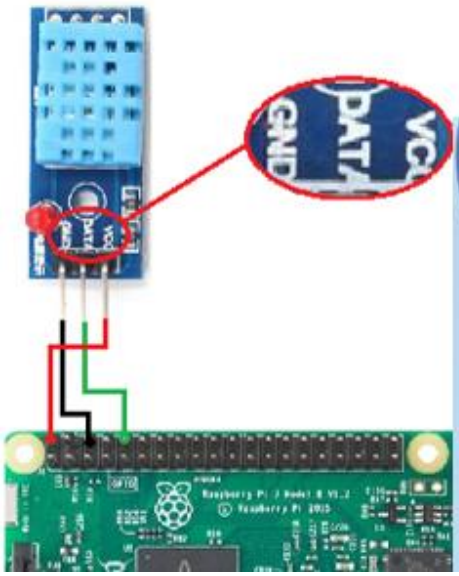


Figura 5.18 Conexionado sensor DHT11

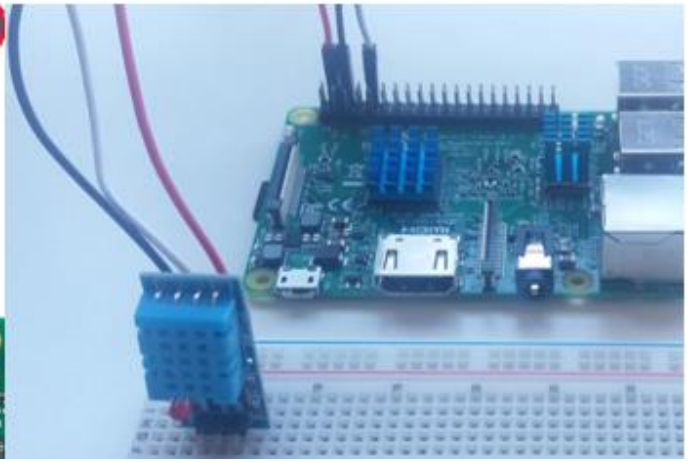


Figura 5.19 Conexionado sensor DHT11 real

Sensor de nivel de agua

En cuanto al sensor de nivel de agua se conectaría igual que el sensor de humedad y temperatura previamente explicado, como pueden observarse en las Figuras 5.20 y 5.21, el sensor de nivel dispone de cuatro cables: azul, marrón, amarillo y negro. El negro representa el modo por lo que no se utiliza, el azul se conectaría a GND de la Raspberry Pi, el marrón se conectaría a la alimentación de 5V de la Raspberry Pi y el amarillo se conectaría a otro pin de entrada de datos de la Raspberry Pi encargado de la lectura de nivel. Pero como se ha comentado anteriormente, tras las pruebas, se ha observado que la Raspberry Pi no tenía suficiente potencia para todos los dispositivos conectados, por lo que realmente, el cable marrón se ha conectado a la salida del limitador de tensión de 5V en vez de a la alimentación de la Raspberry Pi, y el cable azul de tierra se conecta con el GND de la pila, al que también se conecta el GND del limitador de tensión, que es el pin del medio, y el GND de la Raspberry Pi. Por último, para que el limitador funcione correctamente, se conecta el pin de la izquierda del limitador, que representa el pin de alimentación, con el conector positivo de la pila.

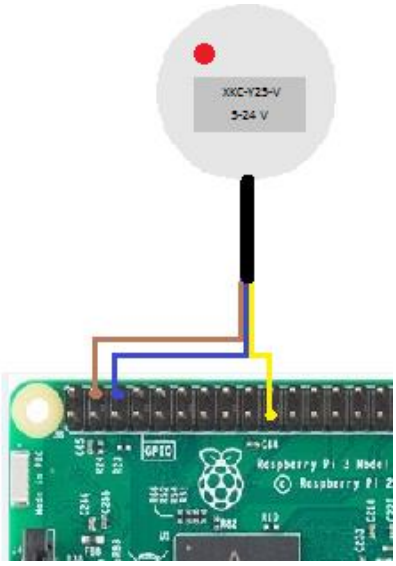


Figura 5.20 Conexionado sensor de nivel de agua

Figura 5.21 Conexionado sensor de nivel de agua real

Sensor de nivel para sólidos

Por último, el sensor de nivel para sólidos, que es empleado para saber el nivel del pienso que hay en el comedero o en el depósito de comida, dispone de tres cables; uno marrón que es el cable de alimentación, uno azul que representa la tierra (*Ground*) y el cable negro que es la salida del sensor. Por lo tanto, como se muestran en las Figuras 5.22 y 5.23, el cable marrón de alimentación se debe conectar a la salida del limitador de tensión de 6V, que es el pin de la derecha, ya que el sensor necesita una alimentación de entre 6-36V. El cable azul de tierra se conecta con el GND de la pila, al que también está conectado el GND del limitador de tensión, que es el pin del medio y el GND de la Raspberry Pi. Para que el limitador funcione correctamente se conecta el pin de la izquierda del limitador, que representa el pin de alimentación, con el conector positivo de la pila, en la Figura 5.22 se representa esta conexión con el cable rojo. Y por último el cable negro del sensor que es la salida de datos se conecta al pin de entrada de datos de la Raspberry Pi, encargado de la lectura de nivel de comida.

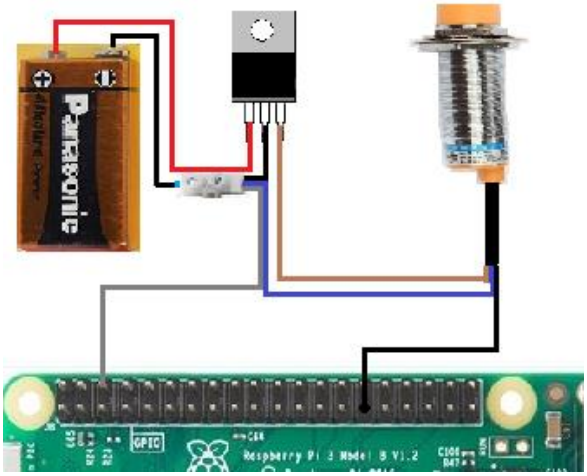


Figura 5.22 Conexionado sensor de nivel de comida

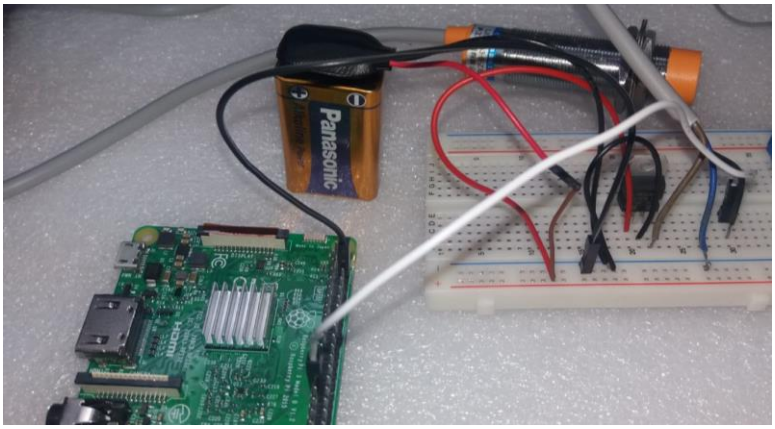


Figura 5.23 Conexionado sensor de nivel de comida real

II. Salidas

Respecto a la conexión con las salidas se exponen las conexiones con una salida digital (LED), con la válvula de agua, con la válvula de comida, con el ventilador y con la manta eléctrica.

Previamente al desarrollo del conexionado de las salidas, cabe exponer unos conceptos básicos. En primer lugar, el funcionamiento de los relés utilizados y sus contactos. Un relé está compuesto por una bobina que, al circular una pequeña corriente (3.3V, 5V o 12V) por ella, como se puede observar en la Figura 5.24, genera un campo magnético que produce el desplazamiento de la placa metálica hasta contactar con la entrada del circuito auxiliar, dejando pasar la corriente o no en función de si la placa metálica está en contacto y, por tanto, alimentando o dejando de alimentar el circuito independiente conectado, que es por el que circula un voltaje superior (220V). Respecto a los contactos, se pueden se observan en la imagen que son tres; C (Común), NC (Normalmente Cerrado), y NA (Normalmente Abierto) respecto al común. El contacto NA permanece abierto mientras no se actúe sobre él por lo que, cuando se actúa sobre él, se cierra y permite la circulación eléctrica a través de él. Por lo que “no deja pasar la corriente" mientras el mismo, o el dispositivo que lo hace funcionar, se hallan en un estado de espera o de reposo. Mientras que el contacto NC permanece cerrado mientras no se actúe sobre él, permitiendo el paso de corriente y en el momento en el que se actúe sobre él, se abre, por lo que deja de circular la corriente. Éste "deja pasar la corriente" mientras el mismo, o el dispositivo que lo hace funcionar, se hallan en un estado de espera o de reposo.

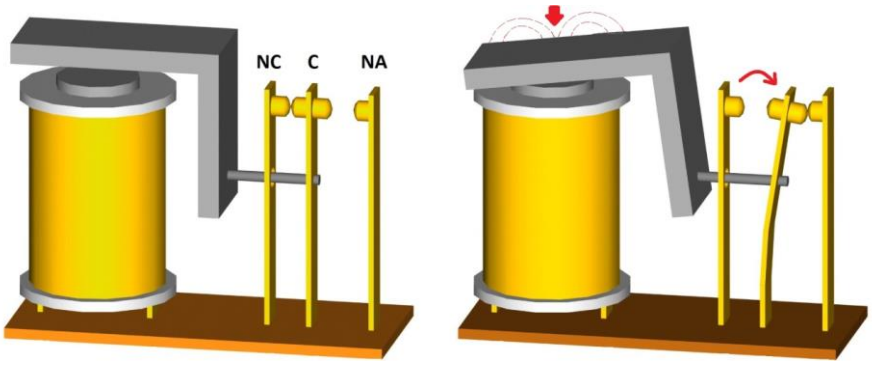


Figura 5.24 Funcionamiento de un relé

En segundo lugar, en el ámbito de la corriente alterna, un cable eléctrico es un elemento que conduce la electricidad. El material principal con el que está fabricado es con cobre por su alto grado de conductividad, aunque también se utiliza el aluminio que, a pesar de que su grado de conductividad es menor, resulta más económico que el cobre.

De otro lado, una manguera [24] es un cable que está formado por varios hilos los cuales son recubiertos cada uno con su aislante y todos ellos recubiertos por la cubierta protectora constituyendo un solo elemento. Los cables que contiene son los siguientes, los cuales se pueden observar en la Figura 5.25:

Cable de tierra: es una protección de todas las instalaciones eléctricas. Su finalidad es proteger la integridad de los equipos o las personas ante una liberación indebida de energía. Los colores que lo identifican suelen ser bicolor verde-amarillo.

Cable neutro o negativo: es el cable que no posee corriente eléctrica. El color que lo identifica suele ser el azul.

Cable fase o positivo: es el conductor que posee la corriente eléctrica. Los colores más comunes que lo identifican comunes son el marrón, el negro y el gris.

Figura 5.25 Colores de cables que contiene una manguera

Salida digital

Para poder observar la activación y desactivación de una salida digital sin la necesidad de realizar el conexionado con los actuadores reales, se ha utilizado un LED junto con una resistencia el cual nos informa si está activada la salida mediante el encendido del LED o si está desactivada mediante el apagado del LED. De esta forma se puede probar el funcionamiento del código desarrollado, con la excepción de que el funcionamiento de la placa de los relés es inverso, por lo que cuando se indique que se active la salida, la placa de relés ordena que se desactive la salida, y viceversa. Respecto a la conexión del LED con la Raspberry Pi, se realiza como se muestra en las Figuras 5.26 y 5.27 la patilla corta del LED, el negativo, se conecta con un extremo de una resistencia de 10 KΩ y el otro extremo de la resistencia al pin GND de la Raspberry Pi, mientras que la patilla larga del LED se conecta al pin de salida de datos correspondiente de la Raspberry Pi.

Figura 5.26 Conexión LED

Figura 5.27 Conexión LED real

Válvula agua

La conexión de la válvula de agua se compone de dos partes, la primera parte es el conexionado de la válvula con la placa de relés y la segunda parte trata del conexionado de la placa de relés con la Raspberry Pi. Respecto a la conexión de la válvula con la placa de relés, la válvula de agua dispone de dos contactos, como se observa en la Figura 5.28, a uno de los contactos se conecta el neutro del enchufe, representado en la figura por el cable azul, y al otro contacto se conecta un cable desde el contacto NA del relé que permita el paso de la corriente a la válvula de agua, en la figura representa el cable marrón, mientras que la fase del enchufe se conecta con el común del relé. En cuanto a la conexión de la placa de relés con la Raspberry Pi, el pin de la placa de relés que representa VCC se conecta con el pin de alimentación de 5V de la Raspberry Pi, el pin que representa GND en la placa de relés se conecta con un pin GND de la Raspberry Pi, mientras que el pin que representa el relé al que está conectado la válvula de agua se conecta al pin de salida correspondiente de la Raspberry Pi.

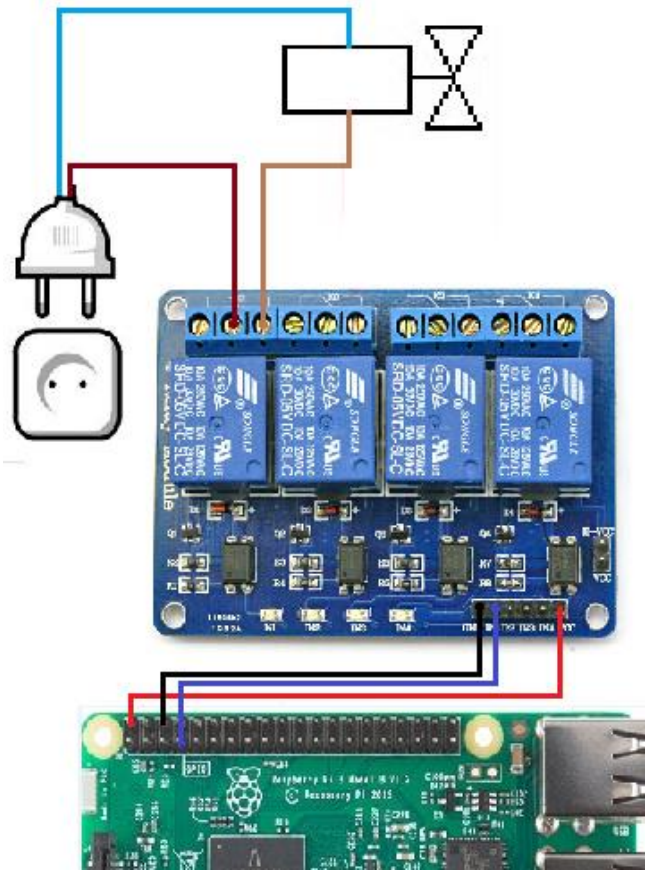


Figura 5.28 Conexión válvula de agua

Válvula comida

Al igual que en la conexión con la válvula de agua, la conexión de la válvula de comida se compone de dos partes, la primera parte es el conexionado de la válvula con la placa de relés y la segunda parte trata del conexionado de la placa de relés con la Raspberry Pi, la cual es igual a la explicada en el apartado anterior, excepto la conexión al pin de la salida de la Raspberry Pi, que es otro pin diferente al de la válvula anterior. Respecto a la conexión de la válvula de comida con la placa de relés, la válvula dispone de tres cables: el amarillo que es el común, el marrón que representa la válvula cerrada y el azul que representa la válvula abierta. Como se muestra en la Figura 5.29, la fase del enchufe, el cable de color granate, se conecta con el común del relé, y el neutro, el cable de color azul, se conecta con el común, el cable de color amarillo y verde de la válvula. El cable marrón de la válvula de comida, el que representa la válvula cerrada, se conecta al pin NC del relé de forma que hasta que no se excite la bobina, la válvula permanecerá cerrada. Mientras que el cable azul de la válvula, el que representa la válvula abierta, se conecta al NA del relé de forma que sólo se abre la válvula cuando se excite la bobina.

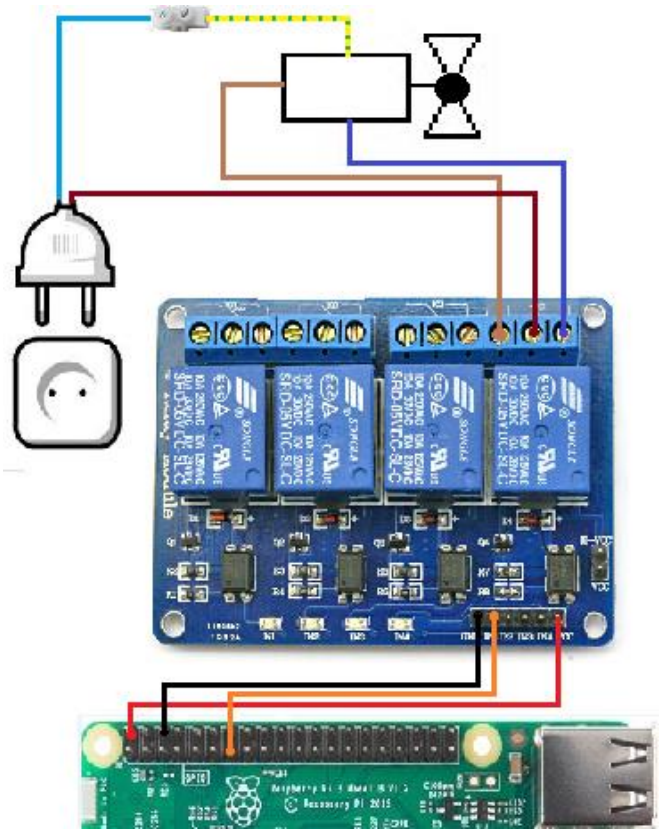


Figura 5.29 Conexión válvula de comida

Ventilador

Al igual que en las conexiones de las salidas explicadas anteriormente, la conexión del ventilador se compone de dos partes, la primera parte es el conexionado del ventilador con la placa de relés y la segunda parte trata del conexionado de la placa de relés con la Raspberry Pi y esta última es igual a la explicada en los apartados anteriores. Respecto a la conexión del ventilador, éste dispone de dos cables, el cable marrón que representa la fase y el cable azul que representa el neutro. La conexión se muestra en la figura 5.30, donde el cable de tensión del ventilador se conecta en el conector NA del relé y el cable neutro del ventilador se conecta con el neutro del enchufe, mientras que el cable de fase del enchufe se conecta al común del relé.

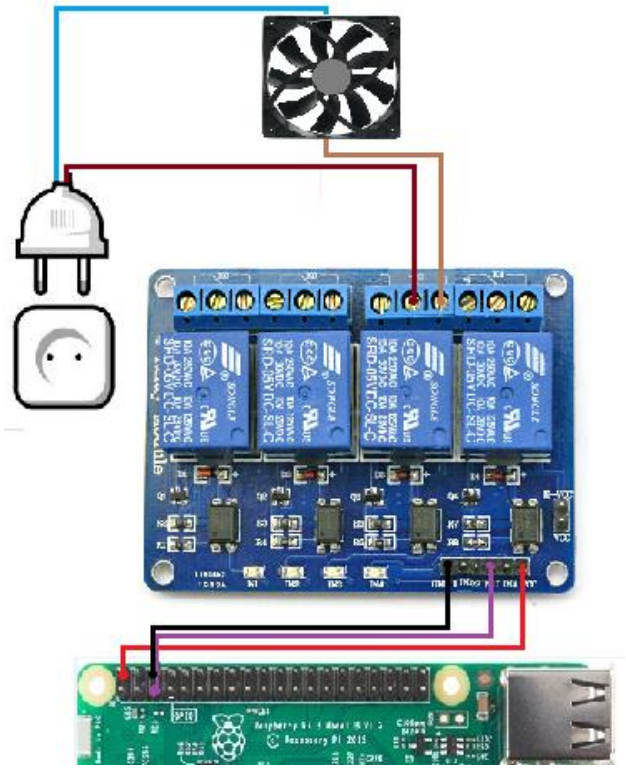


Figura 5.30 Conexión ventilador

Manta Eléctrica

Al igual que en las conexiones de las salidas explicadas anteriormente, la conexión la manta eléctrica se compone de dos partes, la primera parte es el conexionado de la manta eléctrica con la placa de relés y la segunda parte trata del conexionado de la placa de relés con la Raspberry Pi, la cual es igual a la explicada en los apartados anteriores. En cuanto a la conexión de la manta eléctrica con la placa de relés es igual que la conexión del ventilador dado que la manta dispone de dos cables, el cable marrón que representa la fase y el cable azul que representa el neutro. La conexión se muestra en la Figura 5.31, donde el cable de tensión de la manta se conecta en el conector NA del relé y el cable neutro de la manta se conecta con el neutro del enchufe, mientras que el cable de fase del enchufe se conecta al común del relé.

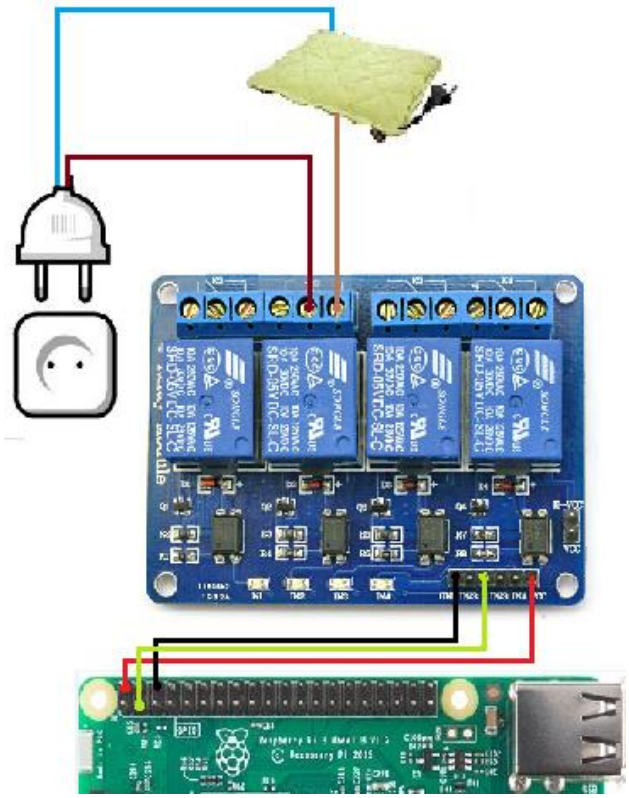


Figura 5.31 Conexión manta eléctrica

III. Periféricos

En cuanto a la conexión de la Raspberry Pi con los periféricos, se tratan dos periféricos, la pantalla táctil y la cámara. Los periféricos son conectados a la Raspberry Pi mediante los puertos que dispone la Raspberry Pi. El esquema de conexionado de la parte de la Raspberry Pi con los periféricos se muestra en la Figura 5.32.

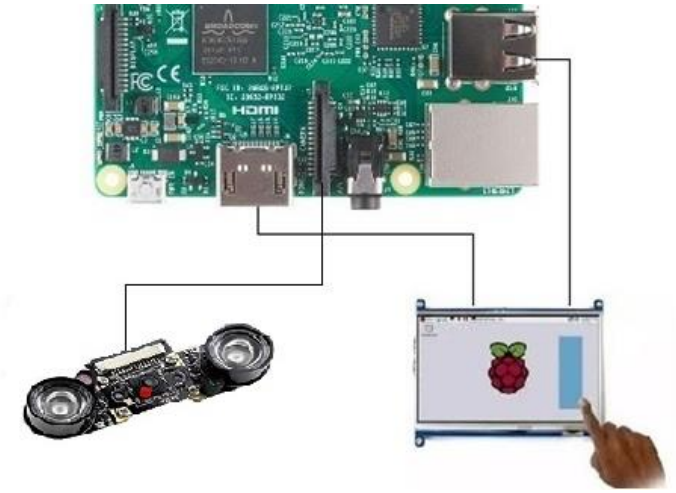


Figura 5.32 Conexión parte periféricos

Pantalla

La pantalla dispone de dos conectores, un conector HDMI y un conector micro USB. Respecto a la conexión, como se puede observar en la Figura 5.33, el conector HDMI se utiliza para mostrar la imagen en la pantalla y se conecta con la entrada HDMI de la Raspberry Pi. Mientras que el conector micro USB se utiliza para la función táctil y la transmisión de datos y éste se conecta a uno de los puertos USB de la Raspberry Pi. Cabe destacar que para que funcione es necesario configurar los controladores correspondientes en la Raspberry Pi, dicha configuración se encuentra en el apartado de Configuración controlador pantalla táctil que es explicado posteriormente.

Figura 5.33 Conexión pantalla táctil real

Cámara

La cámara que se ha utilizado en este proyecto está fabricada para su uso con Raspberry Pi, por lo que ésta dispone de un conector para esta función, como se observa en la Figura 5.34, se introduce en el conector de vídeo que está señalado en la placa con “CAMERA”.

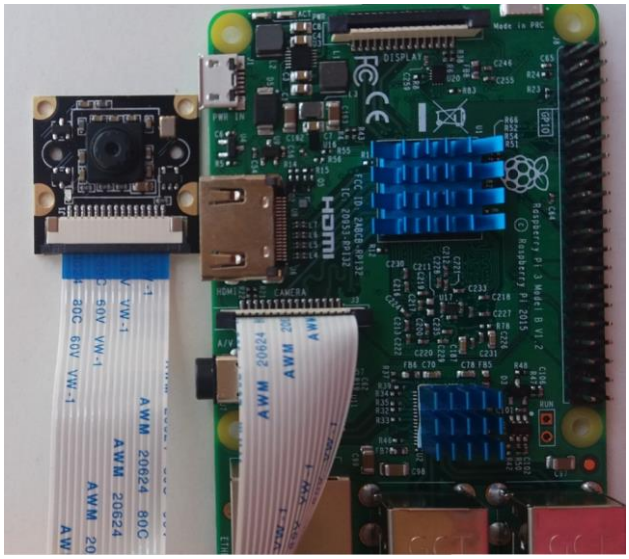


Figura 5.34 Conexión cámara real

5.2.2 Montaje hardware

Se ha intentado que la mayor parte de cableado se ubique en la parte inferior, en la base de la caseta. El proceso de montaje es el siguiente:

1. Se ha atornillado la placa de relés y una regleta de seis vías en una caja hermética. La regleta de seis vías permite realizar el conexionado de la fase y el neutro del enchufe que conecta con la toma de corriente y de los cuatro actuadores con la placa de relés.

Figura 5.35 Caja hermética con placa de relés y regleta

2. La caja hermética se ha situado en la parte trasera de la base. Se han puesto canaletas por las que se pasará el cableado hacia los actuadores y los sensores.

Figura 5.36 Caja hermética y canaletas atornilladas en la base de la caseta

3. Se han atornillado los cables de fase y de neutro de un enchufe a la regleta. Posteriormente se han conectado los cables fase de los actuadores a la placa de relés, se ha conectado la fase del enchufe que se conecta a la red eléctrica con los comunes de los relés y se han conectado en una regleta grande todos los cables neutro de los actuadores junto con el neutro del enchufe.

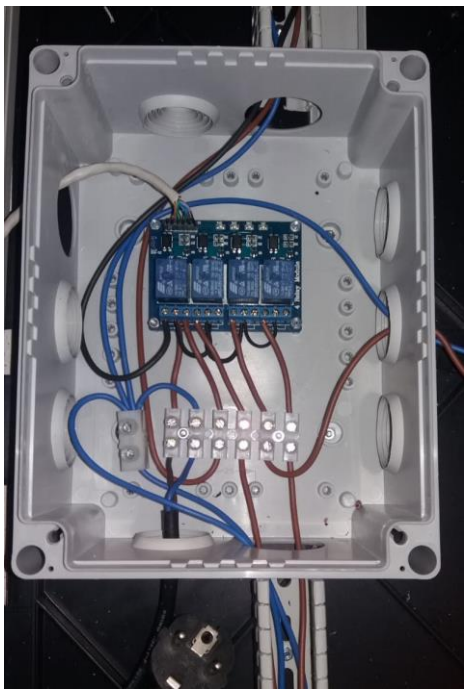


Figura 5.37 Caja hermética con conexionado actuadores

4. Se ha agujereado el recipiente de comida y el depósito de comida para la colocación de los sensores de nivel para sólidos.

Figura 5.38 Recipiente comida con sensores

Figura 5.39 Depósito comida con sensores

5. Se han pegado los sensores de nivel de agua en el bebedero mediante celo de doble cara.

Figura 5.40 Recipiente agua con sensores

6. Como conducto por el que debe pasar el agua y la comida, se han utilizado tubos de PVC y a dichos tubos se han acoplado unas piezas (codos) para que la comida y el agua desemboque en el recipiente y para unir las válvulas con los tubos (racor). Dichas piezas se han pegado a los tubos de PVC, como se observa en la Figura 5.41, mediante un pegamento especial para PVC.

Figura 5.41 Unión de tubos de PVC con piezas

7. Se han sujetado los tubos de PVC a unas escuadras con bridas y posteriormente se han unido dichos tubos a la válvula de agua y a la válvula de comida. Como puede observarse en la Figura 5.42, mediante dos piezas de PVC se ha unido la válvula de comida con el depósito de comida.

Figura 5.42 Tubos PVC junto con válvulas sujetos a las escuadras

8. Se ha introducido el sensor DHT11 dentro de una caja de pequeño tamaño con un agujero para que pueda obtener la temperatura y la humedad relativa dentro de la caseta, y dicha caja con el sensor se ha atornillado en la escuadra que sujeta el tubo por el que fluye la comida.

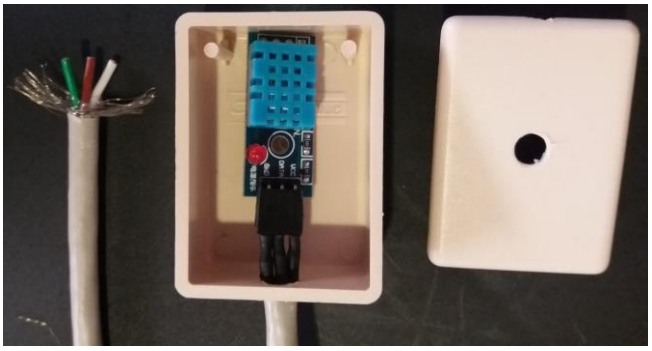


Figura 5.43 Caja con sensor DHT11

9. El ventilador se ha colocado en la parte superior de la caseta como puede observarse en la Figura 5.44.

Figura 5.44 Ventilador atornillado en la caseta

10. La manta eléctrica se ha situado sobre el suelo de la caseta y se ha realizado un agujero en el suelo para pasar el cable de la manta eléctrica.

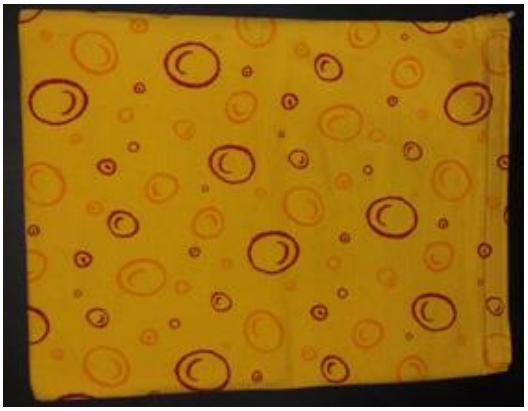


Figura 5.45 Manta eléctrica en la caseta

11. Se ha soldado todos los cables de los sensores y actuadores a unos pines que están conectados a los pines de la Raspberry Pi. Como se observa en la Figura 5.44, además de dicha soldadura, se han conectado los limitadores de tensión y la pila.

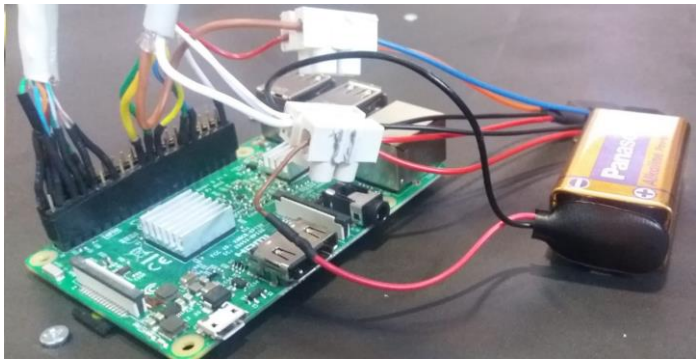


Figura 5.46 Soldadura de conexiones con Raspberry Pi

12. Se ha atornillado la Raspberry Pi y la pantalla táctil dentro de una caja hermética con tapa, lo que permite, levantando la tapa, interactuar con la aplicación mediante la pantalla táctil.

Figura 5.47 Caja con Raspberry Pi y pantalla táctil

13. Se ha atornillado la cámara de visión nocturna en la parte trasera de la caja hermética de forma que colocando la caja hermética en la fachada trasera de la caseta, pueda visualizarse el interior de la caseta, a través de un agujero realizado en la fachada para la introducción de la cámara.

Figura 5.48 Parte trasera de caja con cámara de visión nocturna

6. Implantación

En este apartado se exponen configuraciones necesarias para realizar la puesta en marcha del sistema desarrollado en este proyecto, se comienza explicando la previa configuración de software que se debe realizar para la implantación de la aplicación desarrollada y posteriormente se muestra la configuración previa que hace referencia al hardware para que funcione el hardware del sistema propuesto.

6.1 Configuración software

Previamente a la puesta en marcha del proyecto, se deben realizar una serie de configuraciones en la Raspberry Pi, en la pantalla táctil y en el sistema operativo para que cuando se ejecute la aplicación funcione según lo especificado.

6.1.1 Configuración Raspberry Pi

I. Instalación sistema operativo

En primer lugar, se instala una imagen del sistema operativo que se desea en una tarjeta SD para ello se debe introducir en el lector de tarjetas del ordenador la tarjeta. A continuación, se descarga la imagen de la página oficial de Raspberry Pi, en este caso se descarga la imagen del sistema operativo que se observa en la Figura 6.1, *Raspbian Strech* con escritorio para poder visualizar la interfaz gráfica.

Figura 6.1 Sistema operativo Raspbian

Se descarga el archivo de tipo ZIP como se muestra en la Figura 6.2 y se extrae el contenido de dicho ZIP, el contenido es un archivo de extensión ".img" el cual contiene sistema operativo.

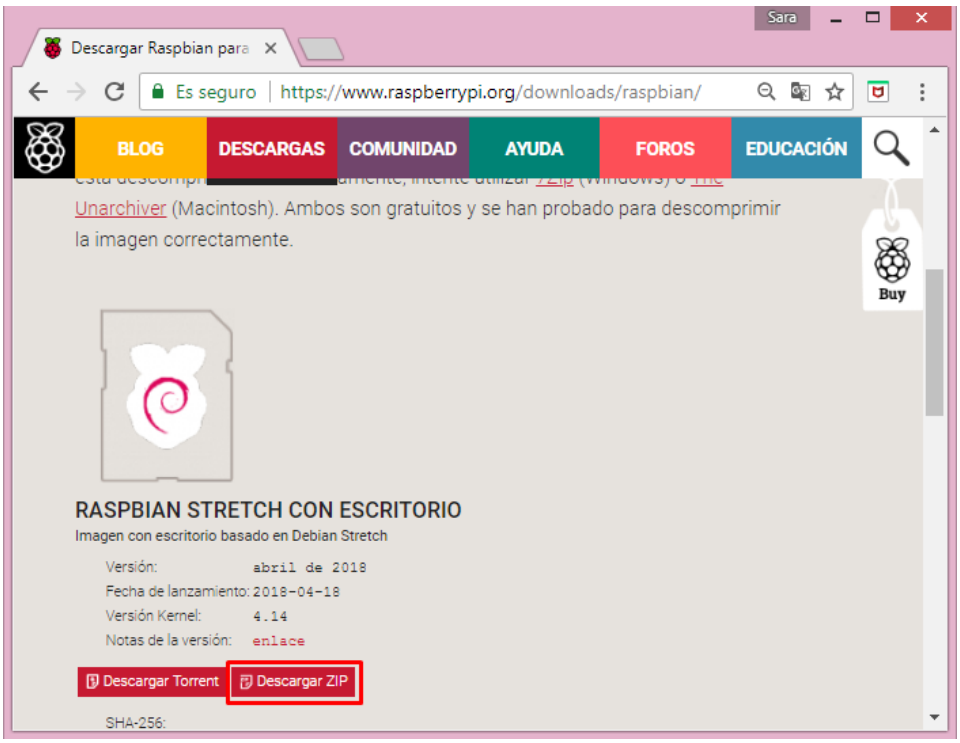


Figura 6.2 Descarga imagen de Raspbian

Para poder escribir en la tarjeta SD la imagen descargada se debe utilizar una herramienta que permita la escritura de imágenes, la página oficial de Raspberry Pi [27] informa de una herramienta de escritura en tarjetas SD que funciona en la mayoría de sistemas operativos llamada *Etcher*, para ello se descarga dicha herramienta de la página referenciada en el apartado de Referencias [25] y con la tarjeta introducida en el lector, se abre *Etcher*, y se siguen los pasos que se muestran en la Figura 6.3, primero se selecciona la imagen del sistema operativo previamente descargada, segundo se selecciona la tarjeta SD donde se desea escribir la imagen y por último se pulsa en "Flash!" para la comenzar la escritura de los datos en la tarjeta.

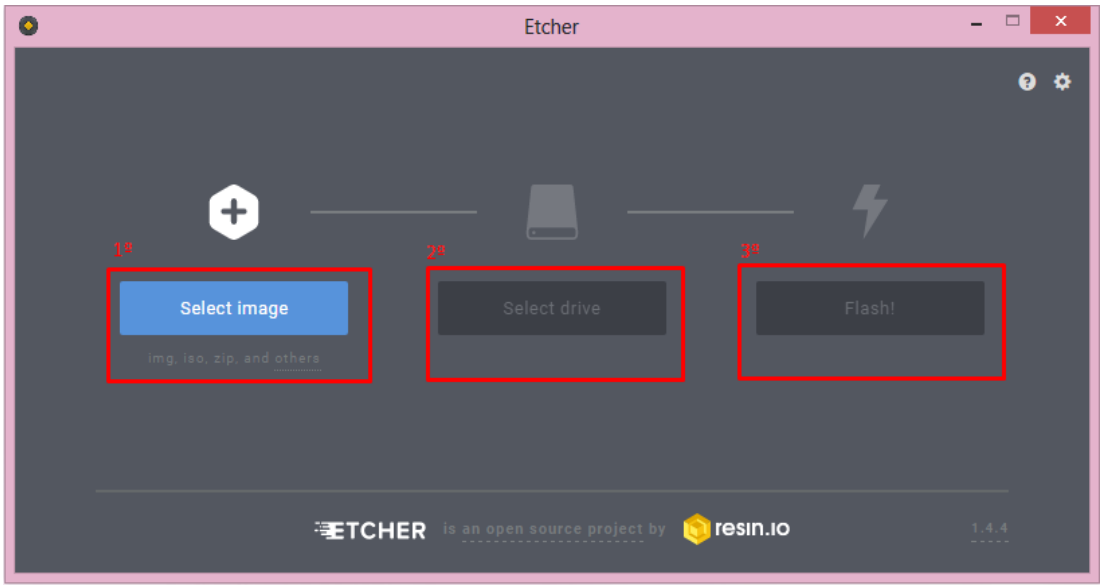


Figura 6.3 Copia de imagen de Raspbian en tarjeta SD

Cuando se complete la escritura de la imagen del sistema operativo en la tarjeta SD aparecerá una ventana como la que se puede observar en la Figura 6.4, desde ese momento se puede extraer la tarjeta SD del lector y, se introduce la tarjeta SD en la Raspberry Pi 3 y es posible que cuando se inicia la Raspberry Pi no lo haga en modo gráfico, no se observa el escritorio, por lo que para iniciar dicho modo se debe ejecutar el comando *startx* en la línea de comandos.

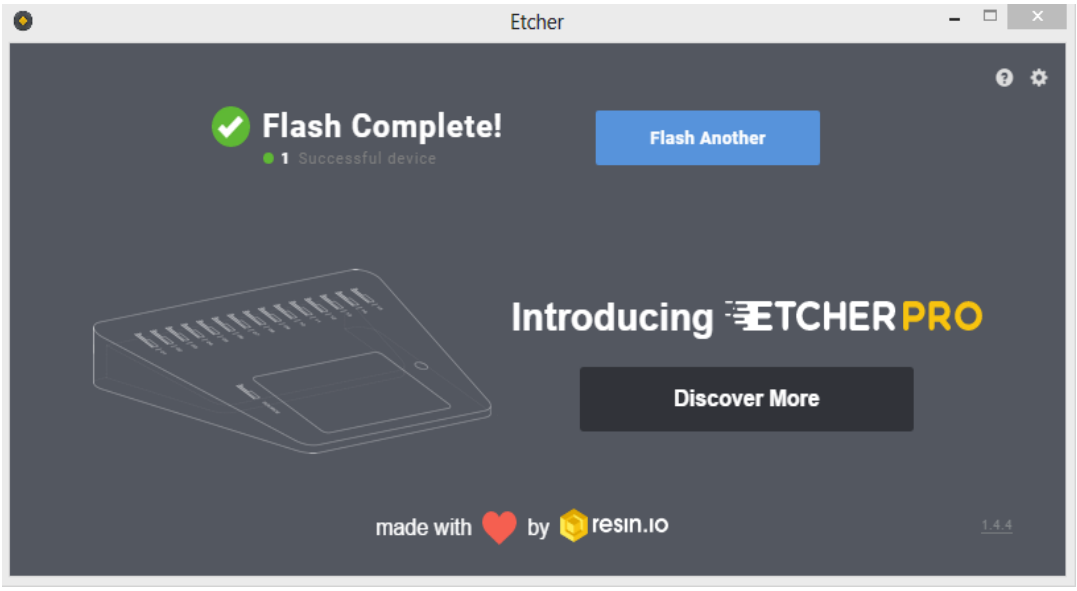


Figura 6.4 Finalización de copia en tarjeta SD

II. Configuración Raspberry Pi

Para configurar la Raspberry Pi [27] a las necesidades del proyecto se debe ir al menú de configuraciones de Raspberry Pi para ello nos ubicamos en el menú desplegable que se muestra en la Figura 6.5, se selecciona Preferencias y posteriormente Configuración Raspberry Pi.

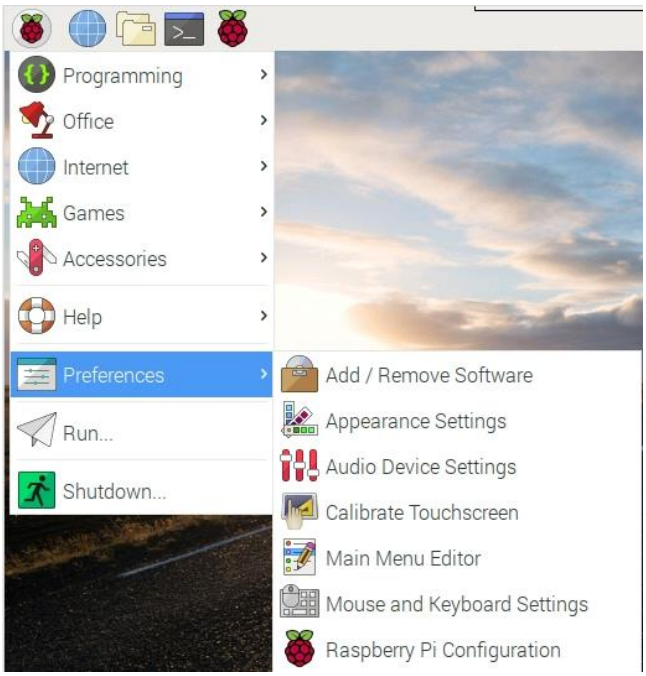


Figura 6.5 Menú desplegable Raspberry Pi

Se debe configurar la resolución de pantalla a la correspondiente, para ello en el menú de Configuración de Raspberry Pi que aparece en la Figura 6.6, en la pestaña de Sistema, se debe pinchar en el botón Modificar Resolución y seleccionar la resolución de la pantalla de la que se dispone.

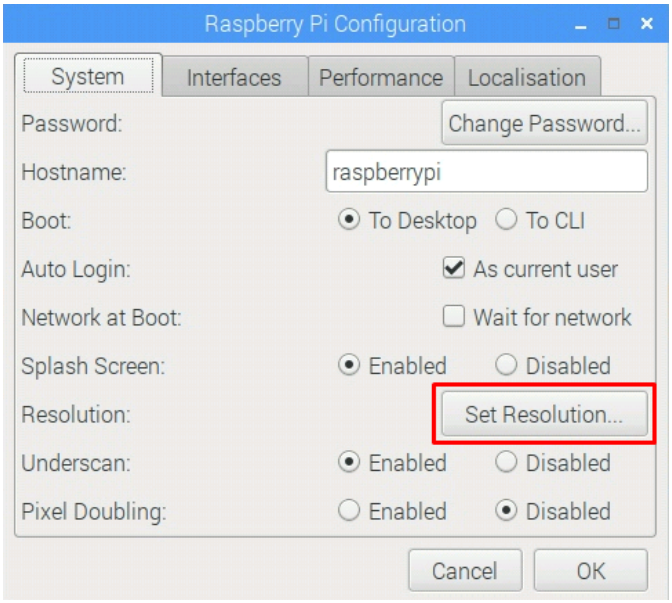


Figura 6.6 Modificar resolución

Para que el reloj se configure a la zona horaria en la que se reside se debe situar en la pestaña Localización que se muestra en la Figura 6.7, pinchar en el botón Modificar Zona Horaria y seleccionar la localización deseada, en este caso es Europa y Madrid.

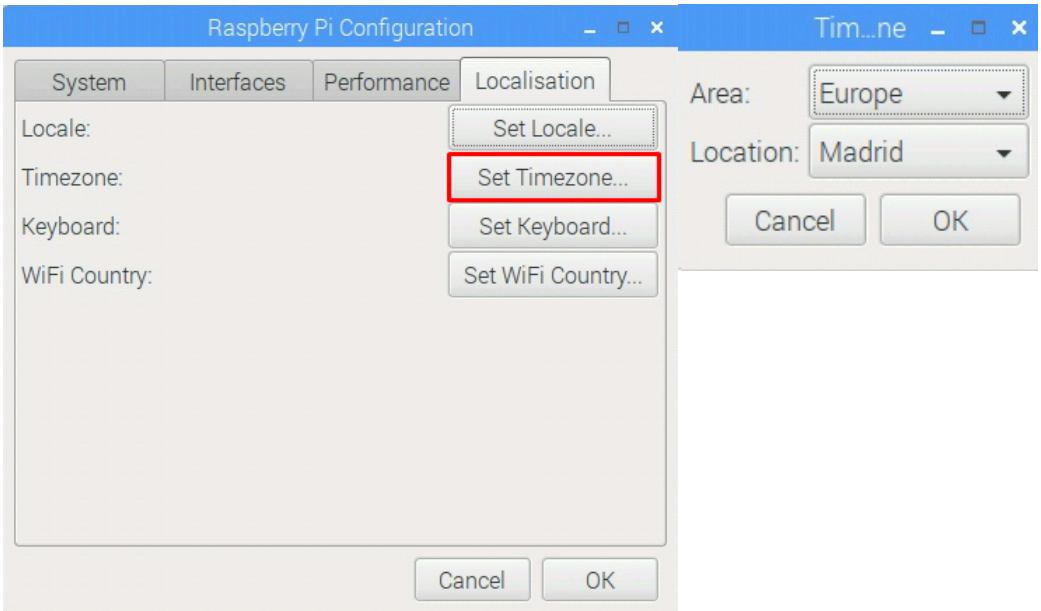


Figura 6.7 Modificar zona horaria

Para que funcione la cámara, el servicio VNC y los puertos GPIO de la Raspberry Pi se deben habilitar dichas interfaces en la pestaña de Interfaces como se puede observar en la Figura 6.8.

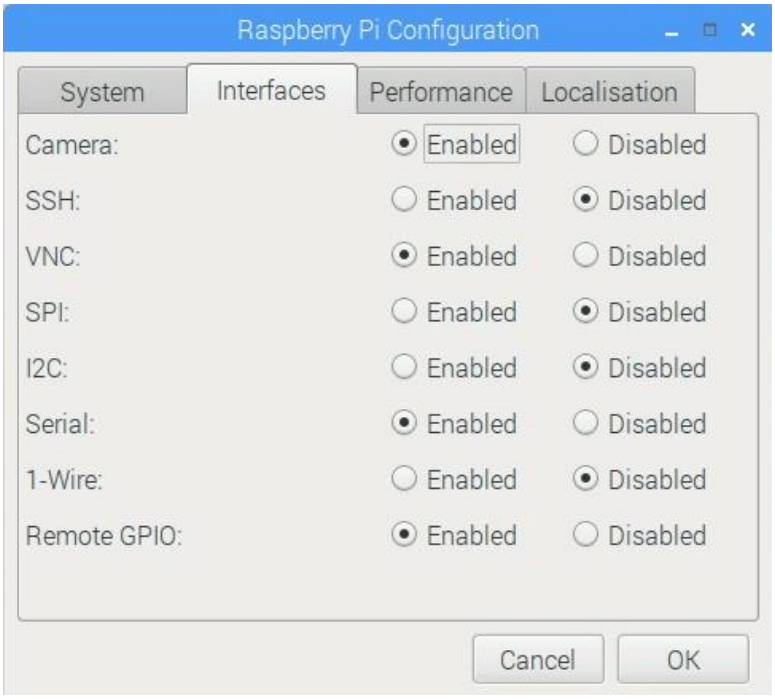


Figura 6.8 Habilitar interfaces

Para conectar la Raspberry Pi a internet se puede realizar mediante la conexión de un cable RJ45 en el conector RJ45 que dispone la Raspberry Pi o mediante wifi. Para conectarse mediante wifi se debe pulsar sobre el icono que representa que el wifi está apagado y seleccionar Encender y posteriormente elegir la red.

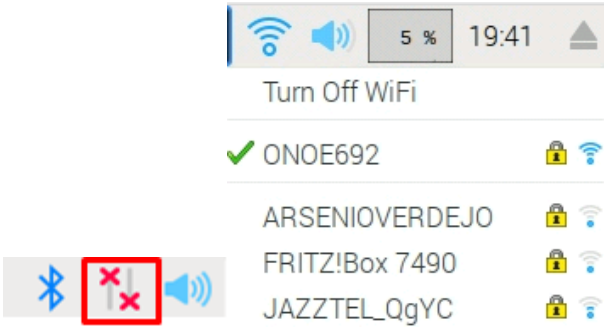


Figura 6.9 Conectar a una red wifi

6.1.2 Configuración controlador pantalla táctil

Cuando se conecta directamente la Raspberry Pi a la pantalla táctil aparece una pantalla en negro dado que no están los controladores instalados, por ello en este apartado se muestra como se han instalado dicho controlador. En primer lugar se deben ejecutar lo siguientes comandos por si hay actualizaciones disponibles.

```
sudo apt-get update
sudo apt-get updrade
```

Posteriormente se ha de agregar las siguientes líneas al archivo "config.txt" que se encuentra en */boot/:*

```
max_usb_current=1
hdmi_group=2
hdmi_mode=87
hdmi_cvt 1024 600 60 6 0 0 0
hdmi_drive=1
```

Para la obtención del controlador se debe descargar un archivo comprimido que contiene el controlador de la página referenciada en el apartado de Referencias [26] como puede observarse en la Figura 6.10.

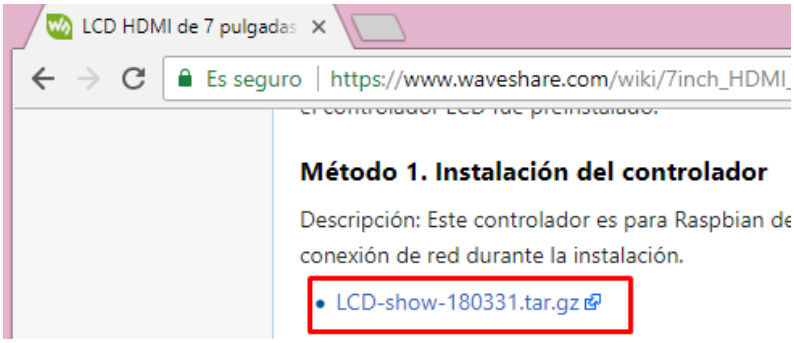


Figura 6.10 Controlador de la pantalla táctil

Se abre un terminal para instalar el controlador y se introduce las siguientes líneas. Tras reiniciar funcionará la pantalla correctamente e inclusive su función táctil.

```
tar xzvf LCD-show-*.tar.gz
cd LCD-show /
chmod + x LCD7-1024x600-show
./LCD7-1024x600-show
```

6.1.3 Ejecución aplicación al arranque

Con el objetivo de que el usuario no tenga que ejecutar ningún comando ni pulsar en el script ejecutable y de que si hay un apagón en la casa cuando vuelva la luz la aplicación se ejecute automáticamente, se ha decidido configurar los ficheros de inicio de Raspbian para que ejecute la aplicación nada más se inicie el sistema operativo. Para ello en primer lugar se crea un fichero ejecutable, el cual se ha llamado "ejecutable.sh", para crear dicho fichero se ejecuta el siguiente comando en el terminal:

```
sudo nano ejecutable.sh
```

En este fichero se introduce las siguientes líneas que permiten la ejecución de la aplicación:

```
cd /home/pi/Desktop/caseta/

sudo java -classpath '.:classes:*classes:/opt/pi4j/lib/*' -jar app.jar raspberry
```

Para que el fichero sea ejecutable se le debe dar permiso de ejecución, para ello se introduce en el terminal el siguiente comando:

```
sudo chmod +x ejecutable.sh
```

Posteriormente se añade al final del fichero ".bashrc" ubicado en la raíz, mediante *sudo nano .bashrc*, la siguiente línea para que se ejecute el ejecutable donde está la línea de ejecución de la aplicación. Si en alguna ocasión se desea que no se ejecute la aplicación al inicio es suficiente con comentar o eliminar la siguiente línea.

```
bash ejecutable.sh
```

Por último, se modifica el fichero "autostart" que se ubica en la ruta /.config/lxsession/LXDE-pi/autostart, mediante *sudo nano /etc/xdg/lxsession/LXDE-pi/autostart*, introduciendo la siguiente línea en el fichero:

```
@lxterminal
```

De esta forma se logra que cuando arranque el sistema operativo, se ejecute el terminal el cual ejecuta el fichero "ejecutable.sh" y dicho fichero ejecuta la aplicación desarrollada.

6.2 Configuración hardware

En este apartado se muestran una serie de pasos necesarios que se deben realizar para que sea posible el funcionamiento correcto del sistema desarrollado.

6.2.1 Conexión agua

Para la hidratación de la mascota es necesario realizar la conexión de agua potable a la caseta, para ello se debe conectar una manguera de la red de agua potable de la propiedad a la entrada de agua de la caseta.

Figura 6.11 Conexión agua

6.2.2 Conexión luz

Para que el hardware funcione es necesario conectar a la luz el cable que alimenta a todos los aparatos electrónicos de la caseta, constituido por un cable de alimentación y un enchufe macho, situado en la parte posterior de la caseta. Por lo que se necesita una toma de corriente que no diste mucho de la ubicación de la caseta o que se conecte un alargador de forma que pueda haber una distancia mayor desde la toma de luz hasta la ubicación de la caseta. Para ello se debe de suministrar a la caseta energía eléctrica, 220V CA.

Figura 6.12 Conexión luz

7. Pruebas

En este apartado se comentan las pruebas que se han realizado para observar el correcto funcionamiento del proyecto, los resultados obtenidos y las observaciones pertinentes.

7.1 Prueba de fiabilidad

Para probar la fiabilidad de la aplicación en ejecución en la Raspberry Pi se ha dejado en funcionamiento la Raspberry Pi, ejecutando la aplicación, durante tres días y no se ha observado ningún problema. Además, para observar su fiabilidad en diferentes climas, se ha dejado funcionando bajo el sol dentro de una caja hermética durante cinco horas.

7.2 Prueba de la aplicación

Con el objetivo de probar el correcto funcionamiento de la aplicación se ha probado manualmente todas las opciones que se pueden elegir. Se han seleccionado todos los modos de funcionamiento en cada uno de los actuadores y han funcionado correctamente. Los botones de subir y bajar realizan su función. Respecto a la ejecución de la aplicación al inicio del sistema se ha probado apagando el sistema y reiniciando el sistema y se ha iniciado la aplicación al inicio correctamente. Respecto a la prueba de la aplicación con el hardware, se ha probado a activar y desactivar todos los actuadores, se ha aumentado la temperatura de la caseta y el campo de la temperatura en la aplicación mostraba como subía la temperatura, se han probado los modos automáticos de todos los actuadores estableciendo los puntos de consigna y se activan y se desactivan correctamente. Para mostrar el correcto funcionamiento se adjunta un vídeo en el cual se puede observar el correcto funcionamiento previamente explicado.

7.3 Prueba de validación

Se ha dejado probar la aplicación a varios usuarios, entre ellos se destaca a mi primo, que es veterinario, y observa que debería haber otro horario de comida en modo automático dado que cuando son cachorros o tienen problemas intestinales se recomienda dar de comer de tres a cuatro veces al día raciones pequeñas.

8. Conclusiones

Este proyecto tenía como objetivo principal automatizar una caseta de mascotas para satisfacer las necesidades básicas y mejorar comodidad del animal que reside en ella, así como facilitar las tareas de cuidado al dueño de éste. Por lo tanto, el objetivo principal y los objetivos específicos que eran informar del valor de los sensores y controlar los actuadores mediante un programa se han cumplido.

Durante el desarrollo han surgido varios inconvenientes; el principal inconveniente es la corrupción de la tarjeta SD de la Raspberry Pi por lo que se debía instalar de nuevo la imagen en la tarjeta SD. Otro inconveniente es la fuente de alimentación de la Raspberry Pi, ya que al utilizar sensores, periféricos y la placa de relés que se alimentan de los 5V que proporciona la Raspberry Pi no se puede utilizar cualquier cargador. Por lo que se probaron varios cargadores de diferentes amperajes hasta que se encontró uno de 5’1 V y 2 A que dejó de dar problemas de alimentación.

Respecto a los errores cometidos en el proyecto, el principal error cometido fue comprar cinco sensores de presencia pensando que serían válidos tanto para la detección de líquido como de sólido pero fue una suposición errónea, dado que dicho sensor no detectaba sólidos, solo detectaba líquidos.

Pese a los problemas y errores cometidos, ha sido una experiencia muy enriquecedora que me ha servido en el ámbito personal como en el ámbito profesional. Respecto al ámbito personal, ha habido muchos momentos frustrantes y finalmente he sabido llevar la situación hacia adelante, he aprendido a gestionar el tiempo, priorizando las tareas a realizar y dedicándole el tiempo adecuado a cada tarea. Mientras que en el ámbito profesional este proyecto me ha ayudado mucho en mi formación en el campo de electrónica dado que era prácticamente nula, en el uso de la Raspberry Pi ya que nunca la había utilizado y he observado que con ella se pueden realizar fácilmente una amplia gama de proyectos. Respecto al software este proyecto me ha servido a poner en práctica patrones arquitectónicos y aprender a comunicarme mediante programación con los sensores y los actuadores.

8.1 Relación del trabajo desarrollado con los estudios cursados

En la carrera se han dado diversas asignaturas las cuales no están relacionadas directamente con el desarrollo de este proyecto pero estas materias han aportado habilidades y destrezas sobre algunos campos tocados en este proyecto. En cuanto a las asignaturas que han aportado conocimientos directos para el desarrollo de este proyecto son las asignaturas en las que se ha enseñado el lenguaje de programación Java, como es el caso de introducción a la informática y a la programación, concurrencia y sistemas distribuidos programación, ingeniería del software, entre otras. Sin embargo la asignatura que tiene relación con la parte software del proyecto es diseño y aplicaciones de los sistemas distribuidos dado que en esta asignatura se asigna un proyecto final en el que se tiene que programar en Java, se debe tener claros los conceptos referentes a los hilos, se crean interfaces gráficas y se utiliza la herramienta Eclipse.

9. Trabajos futuros

Realizando el proyecto se observan muchas ampliaciones y muchas mejoras las cuales se podrían haber puesto en desarrollo si se dispusiera de más tiempo. Respecto a las ampliaciones que se podrían realizar en este proyecto serían las siguientes;

- La incorporación de un sensor de cama mojada de manera que si la manta eléctrica se mojara por diferentes motivos se avisara al dueño.
- El desarrollo de una aplicación móvil donde se mostrara el mismo contenido que la aplicación actual, adaptando la interfaz a la vista en una pantalla móvil y de esta manera el usuario no debería conectarse mediante una aplicación de conexión a escritorio remoto para poder visualizar la interfaz y cambiar la configuración actual.
- El desarrollo de una aplicación web que, al igual que la anterior ampliación, mostrara el mismo contenido que la aplicación actual, solo que el usuario podría observar dicha interfaz desde internet sin tener que conectarse mediante conexión a escritorio remoto. Cabe destacar que sería necesario emplear algún mecanismo de seguridad para que no pudiese acceder cualquiera a dicha interfaz.
- La incorporación de paneles solares sobre el tejado de la caseta de manera que captara la energía de la luz del sol por el día y así se produjera electricidad que alimentara al material hardware del que dispone la caseta.

10. Referencias

[1] (07-06-2018) Brenda Macías, mediatrends. *Dog Parker: la casa inteligente en la que tu perro (casi) no te echará de menos*. Disponible en: https://www.mediatrends.es/a/102079/dog-parker-casa-inteligente-smart-home-perro/

[2] (07-06-2018) Idoia Estadella, mediatrends. *Samsung presenta Dream Doghouse, una Smart Home para tu perro, de 30.000 dólares.* Disponible en: https://www.mediatrends.es/a/27628/samsung-presenta-dream-doghouse-smart-home-para-perro/

[3] (17-02-2018) Akizuki Denshi Tsusho Co. *Temperature and humidity module*. Disponible en: https://akizukidenshi.com/download/ds/aosong/DHT11.pdf

[4] (17-02-2018) Pico.dev. *Obtener la temperatura y humedad con el sensor DHT11, la Raspberry Pi, C y Java.* Disponible en: https://picodotdev.github.io/blog-bitix/2017/03/obtener-la-temperatura-y-humedad-con-el-sensor-dht11-la-raspberry-pi-c-y-java/

[5] (05-03-2018) Altec, SA de CV. Monterrey. *Flotador*. Disponible en: http://www.altecdust.com/productos/controles-de-nivel/flotador/

[6] (05-03-2018) Omega Engineering. *Sensor de nivel.* Disponible en: https://es.omega.com/prodinfo/sondas-de-nivel-medicion.html

[7] (01-04-2018) Engineering360. *Water Valves Information.* Disponible en: https://www.globalspec.com/learnmore/flow_control_flow_transfer/valves/water_valves

[8] (02-04-2018) U.S. SOLID. *Solenoid Valves*. Disponible en: https://ussolid.com/solenoid-valves.html

[9] (02-04-2018) Enrique Jose Caroli, Monografías. *Válvulas*. Disponible en: http://www.monografias.com/trabajos11/valvus/valvus.shtml

[10] (10-04-2018) Engineering360. *Solids Valves Information.* Disponible en: https://www.globalspec.com/learnmore/flow_control_flow_transfer/valves/solids_valves

[11] (10-04-2018) Salina Vortex. *How Does an Iris Valve Work?*. Disponible en: https://www.vortexglobal.com/how-does-an-iris-valve-work/

[12] (13-04-2018) Zooplus. *Manta térmica Comfy con funda intercambiable.* Disponible en: http://www.zooplus.es/shop/tienda_perros/camas_cojines_cestas_mantas_perros/mantas_perros_termicas_refrigerantes/396291

[13] (5-05-2018) Wikipedia. *Periféricos*. Disponible en: https://es.wikipedia.org/wiki/Perif%C3%A9rico_(inform%C3%A1tica)

[14] (25-01-2018) Noelia Machicao. *Modelo Vista Controlador(MVC) y Modelo Vista Presentado (MVP).* Disponible en: http://tecnosemergentes1.blogspot.com/2013/03/modelo-vista-controlador-mvc.html

[15] (27-01-2018) Gilber Aranguren. *PATRONES DE ARQUITECTURA MVC, MV-VM, MVP.* Disponible en: https://ingsoftwarei2014.wordpress.com/category/comparacion-de-los-patrones-de-arquitectura-mvc-mv-vm-mvp/

[16] (22-02-2018) Maximiliano Juarez. *Factory Method.* Disponible en: http://migranitodejava.blogspot.com.es/2011/05/factory-method.html

[17] (23-02-2018) José María Pérez Ramos. *PATRÓN FACTORY METHOD - MÉTODO DE FACTORÍAS.* Disponible en: http://jmperezramos.net/programacion/patron-factory-method-metodo-de-factorias/

[18] (14-01-2018) Robert Savage y Daniel Sendula. *El proyecto Pi4J*. Disponible en: http://pi4j.com/

[19] (23-05-2018) RealVNC. Disponible en: https://www.realvnc.com/es/

[20] (20-03-2018) J. Steven Perri, IBM. *Conceptos básicos del lenguaje Java.* Disponible en: https://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava1/index.html

[21] (15-02-2018) Szazo, GitHub. *DHT11 Python*. Disponible en: https://github.com/szazo/DHT11_Python

[22] (17-04-2018) *Hilos en Java.* Disponible en: http://www.chuidiang.org/java/hilos/hilos_java.php

[23] (21-01-2018) Bartolomé Abellán, *Poner Imagen de fondo en un JPanel*. Disponible en: http://picarcodigo.blogspot.com.es/2012/08/poner-imagen-de-fondo-en-un-jpanel.html

[24] (8-05-2018) *CABLES ELECTRICOS Y TIPOS.* Disponible en: http://www.areatecnologia.com/electricidad/cables-conductores.html

[25] (16-01-2018) Herramienta ETCHER. *Flash OS images to SD cards & USB drives, safely and easily.* Disponible en: https://etcher.io/

[26] (27-04-2018) Waveshare. *7inch HDMI LCD.* Disponible en: https://www.waveshare.com/wiki/7inch_HDMI_LCD

[27] (14-01-2018) *DOCUMENTACIÓN DE RASPBERRY PI.* Disponible en: https://www.raspberrypi.org/documentation/

PRESUPUESTO

[28] AliExpress. *Sensor de nivel de líquido XKC-Y25-V*, Disponible en: https://es.aliexpress.com/store/product/1Pcs-XKC-Y25-V-Intelligent-Non-Contact-Liquid-Level-Sensor-Detection-Tool-for-Airtight-Container-Wholesale/2959023_32821693896.html?ws_ab_test=searchweb0_0,searchweb201602_1_10152_10065_10151_10344_10068_10342_10547_10343_10340_5722611_10341_10548_10698_10697_10696_5722911_5722811_10084_5722711_10083_10618_10307_10301_10303_5711211_10059_10184_10534_308_100031_10103_441_10624_10623_10622_10621_10620_5723011_5711311_5722511,searchweb201603_28,ppcSwitch_5&algo_expid=c5f8540b-3db7-4b7e-8251-7af62e6fc059-5&algo_pvid=c5f8540b-3db7-4b7e-8251-7af62e6fc059&transAbTest=ae803_2&priceBeautifyAB=0

[29] AliExpress. *Sensor de nivel para sólidos DIANQI*. Disponible en: https://es.aliexpress.com/store/product/capacitance-proximity-sensor-LJC18A3-B-Z-AX-18mm-diameter-10mm-detective-distance-DC-6-36V-NPN/521261_955022694.html?spm=a219c.search0104.3.29.6d591a8fZWZtwG&ws_ab_test=searchweb0_0,searchweb201602_1_10152_10151_10065_10344_10068_10547_10342_10343_10340_10548_10341_10696_10084_10083_10618_10307_10820_10821_10301_10303_10059_100031_524_10103_10624_10623_10622_10621_10620,searchweb201603_32,ppcSwitch_5&algo_expid=a8c0ce97-8ffd-4449-8268-1d676cebcbd4-4&algo_pvid=a8c0ce97-8ffd-4449-8268-1d676cebcbd4&transAbTest=ae803_2&priceBeautifyAB=0

[30] AliExpress. *Válvula de agua KHAN*. Disponible en: https://es.aliexpress.com/store/product/1pc-New-Practical-Electric-Solenoid-Valve-AC-220V-Water-Air-N-C-Normally-Closed-0-0/2321013_32818343577.html?spm=a219c.search0104.3.2.61077f513QYY84&ws_ab_test=searchweb0_0,searchweb201602_1_10152_10151_10065_10344_10068_10547_10342_10343_10340_10548_10341_10696_10084_10083_10618_10307_10820_10821_10301_10303_10059_100031_524_10103_10624_10623_10622_10621_10620,searchweb201603_32,ppcSwitch_5&algo_expid=89d564f3-350f-4a2e-b5a9-5f79906d2c80-0&algo_pvid=89d564f3-350f-4a2e-b5a9-5f79906d2c80&transAbTest=ae803_2&priceBeautifyAB=0

[31] AliExpress. *Válvula de comida SAILFLO*. Disponible en: https://es.aliexpress.com/store/product/Electric-Motorized-Brass-Ball-Valve-DN20-AC-220V-2-Way-3-Wire-with-Actuator/2972022_32839550310.html

[32] AliExpress. *Ventilador BLS12/92*. Céspedes Disponible en: http://www.cespedes.es/spa/item/1410585.html

[33] AliExpress. *Pantalla táctil*. Disponible en: https://es.aliexpress.com/store/product/7-inch-Raspberry-Pi-3-Touch-Screen-1024-600-7-inch-Capacitive-Touchscreen-LCD-HDMI-Interface/1381141_32799380449.html?spm=a219c.search0104.3.10.1def20fekqKyOl&ws_ab_test=searchweb0_0,searchweb201602_1_10152_10065_10151_10344_10068_10342_10547_10343_10340_5722611_10341_10548_10698_10697_10696_5722911_5722811_10084_5722711_10083_10618_10307_10301_10303_5711211_10059_10184_308_100031_10103_441_10624_10623_10622_10621_10620_5711311_5722511,searchweb201603_25,ppcSwitch_5&algo_expid=70936cbe-0b48-45e1-a4f2-d520d8efa662-0&algo_pvid=70936cbe-0b48-45e1-a4f2-d520d8efa662&transAbTest=ae803_2&priceBeautifyAB=0

[34] AliExpress. *Cámara de visión nocturna con Luz led*. Disponible en: https://es.aliexpress.com/store/product/New-Arrival-Raspberry-Pi-3-Camera-Module-1080p-5MP-Night-Vision-Camera-2-Pcs-IR-Sensor/615778_32833484581.html?spm=a219c.search0604.3.43.4f4e9959687FTt&ws_ab_test=searchweb0_0,searchweb201602_1_10152_10151_10065_10344_10068_10342_10547_10343_5

722611_10340_10548_10341_10696_5722911_5722811_5722711_10084_10083_10618_10307_10820_10301_10821_10303_10059_100031_10103_10624_10623_10622_10621_10620_5722511,searchweb201603_11,ppcSwitch_5&algo_expid=f8936dc7-ea72-44c0-9e5c-6b9bd19a647f-6&algo_pvid=f8936dc7-ea72-44c0-9e5c-6b9bd19a647f&transAbTest=ae803_2&priceBeautifyAB=0

[35] AliExpress. *Raspberry Pi 3*. Disponible en: https://es.aliexpress.com/store/product/2016-New-Raspberry-Pi-3-Model-B-BCM2837-1-2G-with-WIFI-and-Bluetooth/908915_32615114907.html?ws_ab_test=searchweb0_0,searchweb201602_1_10152_10065_10151_10344_10068_10342_10547_10343_10340_5722611_10341_10548_10698_10697_10696_5722911_5722811_10084_5722711_10083_10618_10307_10301_10303_5711211_10059_10184_10534_308_100031_10103_441_10624_10623_10622_10621_10620_5723011_5711311_5722511-10620,searchweb201603_28,ppcSwitch_5&algo_expid=bc234560-1512-4770-a780-3f2ac86584e9-2&algo_pvid=bc234560-1512-4770-a780-3f2ac86584e9&transAbTest=ae803_2&priceBeautifyAB=0

[36] AliExpress. *Sensor DHT11*. Disponible en: https://es.aliexpress.com/store/product/Nueva-DHT11-sensor-de-temperatura-y-humedad-relativa-para-Arduino/1266255_32840691852.html?ws_ab_test=searchweb0_0%2Csearchweb201602_1_10152_10065_10151_10344_10068_10342_10547_10343_10340_5722611_10341_10548_10698_10697_10696_5722911_5722811_10084_5722711_10083_10618_10307_10301_10303_5711211_10059_10184_10534_308_100031_10103_441_10624_10623_10622_10621_10620_5723011_5711311_5722511%2Csearchweb201603_28%2CppcSwitch_5&algo_expid=9cc41f6d-5fa8-4dcb-9d17-5b2376d59479-23&algo_pvid=9cc41f6d-5fa8-4dcb-9d17-5b2376d59479&transAbTest=ae803_2&priceBeautifyAB=0

[37] AliExpress. *Placa de relés*. Disponible en: https://es.aliexpress.com/store/product/4-way-5V-Relay-module-blue-plate-4-way-output-relay-DIY-Electronic-technology-2pcs/113275_32735292917.html?spm=a219c.search0104.3.2.60583ba1tQczjG&ws_ab_test=searchweb0_0,searchweb201602_1_10152_10151_10065_10344_10068_10342_10547_10343_5722611_10340_10548_10341_10696_5722911_5722811_5722711_10084_10083_10618_10307_10301_10303_10059_100031_10103_10624_10623_10622_10621_10620_5722511,searchweb201603_1,ppcSwitch_5&algo_expid=ecd5e631-e468-4a7e-a38e-83eb904cc7c5-0&algo_pvid=ecd5e631-e468-4a7e-a38e-83eb904cc7c5&transAbTest=ae803_2&priceBeautifyAB=0

[38] AliExpress. *Limitador de tensión L7805*. Disponible en: https://es.aliexpress.com/store/product/10pcs-LM7805-L7805-7805-Voltage-Regulator-IC-5V-1-5A-TO-220/2130127_32850676851.html?spm=a219c.search0104.3.9.a8da49ec2lnPm2&ws_ab_test=searchweb0_0,searchweb201602_1_10152_10151_10065_10344_10068_10342_10547_10343_5722611_10340_10548_10341_10696_5722911_5722811_10084_5722711_10083_10618_10307_10301_10303_5711211_10059_308_100031_10103_10624_10623_10622_10621_10620_5711311_5722511,searchweb201603_2,ppcSwitch_5&algo_expid=166934b6-34c1-4dc1-b0fe-0a028ed971e5-1&algo_pvid=166934b6-34c1-4dc1-b0fe-0a028ed971e5&transAbTest=ae803_2&priceBeautifyAB=0

[39] AliExpress. *Limitador de tensión L7806*. Disponible en: https://es.aliexpress.com/store/product/10pcs-lot-New-L7806CV-L7806-7806-LM7806-KA7806-MC7806-TO-220/418072_32850861231.html?spm=a219c.search0302.3.10.6fea17c6xdyp1t&ws_ab_test=searchweb0_0,searchweb201602_0_10152_10151_10618_10059_10696_10084_100031_10083_10547_10624_10821_10623_10307_10548_524_10341_10065_10340_10068_10343_10342_10301_10103_10620_10344_10303_10820_10622_10621,searchweb201603_0,ppcSwitch_0&algo_pvid

=f5a9628c-bba4-4f17-8108-10d3597860d1&algo_expid=f5a9628c-bba4-4f17-8108-10d3597860d1-1

[40] Amazon. *Pila alcalina*. Disponible en: https://www.amazon.es/Panasonic-Alkaline-Power-61-Pila/dp/B003AWVZIQ/ref=sr_1_4?s=electronics-accessories&ie=UTF8&qid=1527443143&sr=1-4&keywords=Panasonic

[41] AliExpress. *Caja hermética.* Disponible en: https://es.aliexpress.com/store/product/1-pcs-electrical-waterproof-plastic-hinge-box-IP68-control-enclosure-291-301-120mm-transparent-cover/1006252_32503747159.html?spm=a219c.search0104.3.2.339e4a74i8gZ0V&ws_ab_test=searchweb0_0,searchweb201602_5_10152_10151_10065_10344_10068_5722815_10342_10547_10343_10340_5722915_10548_10341_5722615_10696_10084_10083_10618_10307_10301_10303_5722715_10059_100031_10103_10624_10623_10622_5722515_10621_10620,searchweb201603_36,ppcSwitch_5&algo_expid=5629f3de-7816-4ce1-8f25-0593a0d9f94e-0&algo_pvid=5629f3de-7816-4ce1-8f25-0593a0d9f94e&priceBeautifyAB=0

[42] Wanimo. *Depósito de comida.* Disponible en: https://www.wanimo.com/fr/chiens/gamelle-et-distributeur-sc5/mini-boite-a-croquettes-petlife-sf6489/

[43] Leroy Merlin. Caseta de mascota. Disponible en: *http://www.leroymerlin.es/fp/17446135/caseta-de-resina-caseta-perro?idCatPadre=6762&pathFamilaFicha=010302*

Anexos

I. Manual de usuario

Caseta domótica DomoDog

Manual de usuario

Julio 2018

Tabla de contenidos

1. Control de caseta

En la imagen se muestra la aplicación que permite el control y la configuración de la caseta. A continuación, se explica detalladamente todos los apartados y el funcionamiento de la aplicación.

Temperatura

Muestra la temperatura actual que hay dentro de la caseta de la mascota. Es un campo solo informativo.

Humedad relativa

Muestra la humedad relativa actual que hay dentro de la caseta de la mascota. Es un campo solo informativo.

Depósito de comida

"Comida suficiente": Significa que el nivel de alimento en el depósito de comida incorporado en la caseta es suficiente.

"Recargar comida": Significa que el nivel de alimento en el depósito de comida incorporado en la caseta no es suficiente, por lo que debe rellenarse.

Comedero

Pulse en el desplegable del comedero para cambiar la configuración:

- Pulse en **Manual-> Encender** para proporcionar comida.

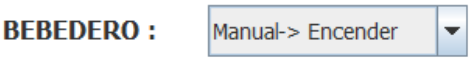
- Pulse en **Manual-> Apagar** para no proporcionar comida.

- Pulse **Automático** para que se proporcione comida automáticamente. En este modo se puede establecer el horario en el que se dosificará la comida. Se permite definir dos horas de comida, una por la mañana que debe establecerse en el rango de [6:00, 16:00] y otra por la tarde que debe establecerse en el rango de [17:00, 5:00].

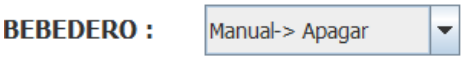
Bebedero

Pulse en el desplegable del bebedero para cambiar la configuración:

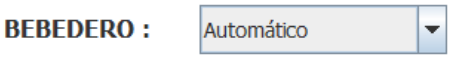
- Pulse en **Manual-> Encender** para proporcionar agua.



- Pulse en **Manual-> Apagar** para no proporcionar agua.



- Pulse **Automático** para que se proporcione agua automáticamente cuando en el bebedero haya un nivel bajo del líquido.



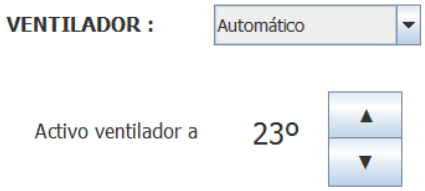
Ventilador

Pulse en el desplegable del ventilador para cambiar la configuración:

- Pulse en **Manual-> Encender** para encender el ventilador .

- Pulse en **Manual-> Apagar** para apagar el ventilador.

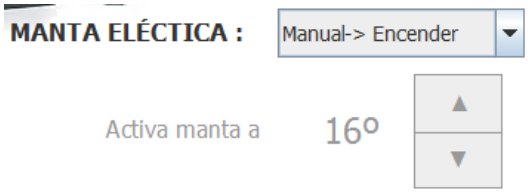
- Pulse **Automático** para que se encienda el ventilador automáticamente cuando la temperatura sea mayor o igual a la temperatura establecida.
  Se puede modificar la temperatura a partir de la cual se activa el ventilador mediante los botones de subir y bajar. Debido a que se considera que la temperatura ideal es 21 grados, en el ventilador se permite establecer un número de grados definidos dentro de un rango [22…28] a partir de los cuales se activará el ventilador.



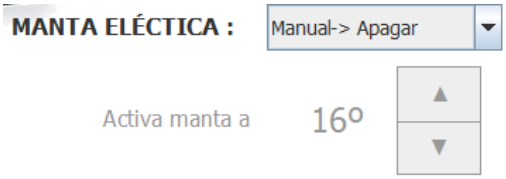
Manta eléctrica

Pulse en el desplegable de la manta eléctrica para cambiar la configuración:

- Pulse en **Manual-> Encender** para encender la manta eléctrica.

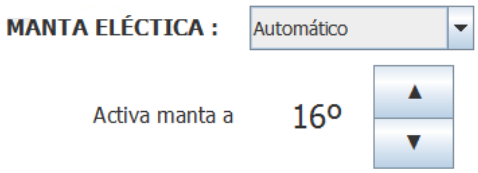


- Pulse en **Manual-> Apagar** para apagar la manta eléctrica.



- Pulse **Automático** para que se encienda la manta eléctrica de forma automática cuando la temperatura sea menor o igual a la temperatura establecida.

Se puede modificar la temperatura a partir de la cual se activa la manta eléctrica mediante los botones de subir y bajar. En la manta eléctrica se permite establecer un número de grados definidos dentro de un rango [15…20] hasta los que se activará la manta eléctrica.



Cámara

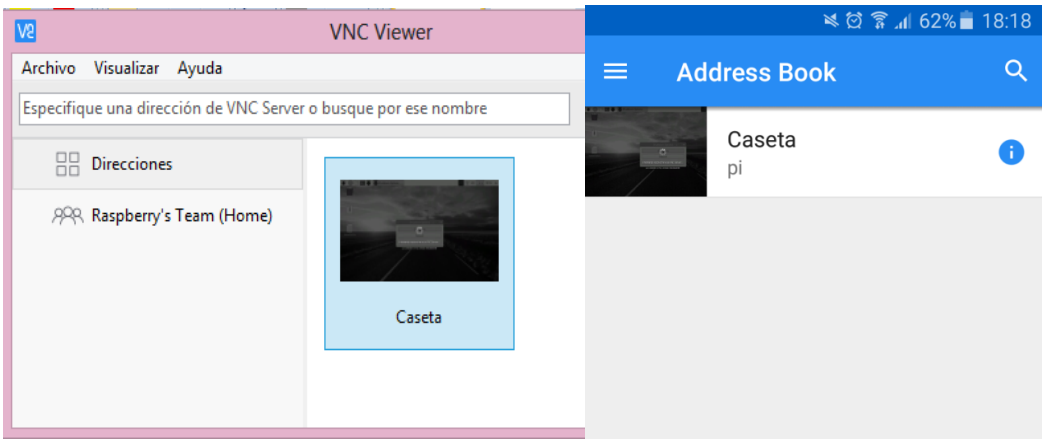
Para la vigilancia de la mascota se ha incorporado una cámara la cual visualiza el interior de la caseta. Pulse el botón con el icono de grabadora para poder visualizar lo que graba la cámara durante un tiempo indefinido.

Tras pulsar el botón de grabadora aparece una ventana como la siguiente en la que se muestra el interior de la caseta. Pulse el botón de Salir que se encuentra en la parte inferior para cerrar dicha ventana y volver a la ventana principal.

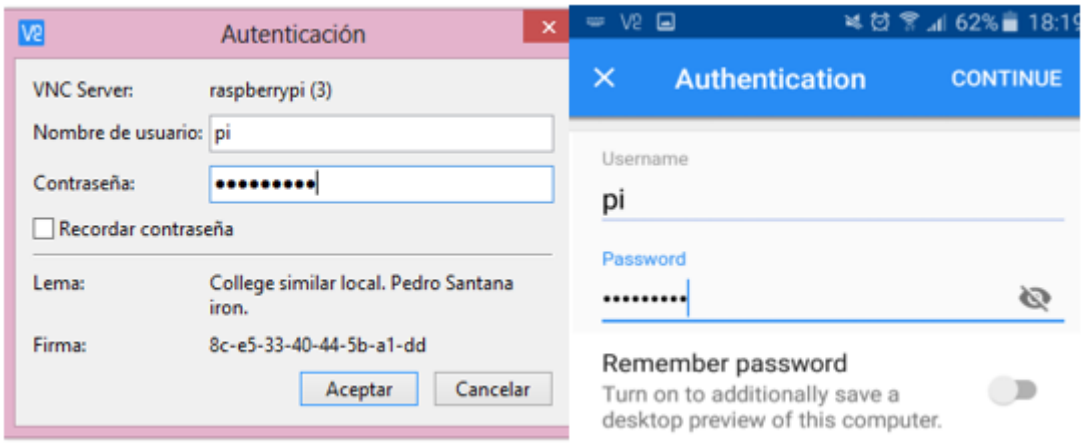
2. Conexión a la aplicación vía móvil u ordenador

A continuación, se muestra cómo controlar la caseta desde un ordenador y desde un móvil, para poder ver lo que se muestra en la pantalla situada en la caseta y por consiguiente la aplicación de la caseta cuando está en ejecución.

1. Debe descargar VNC Viewer desde el dispositivo desde el cual quiere controlar la caseta.
2. Registrarse mediante las credenciales de la suscripción previamente realizada.
3. Selecciona el equipo remoto al que desea conectar, para conectarse a la caseta pulse doblemente sobre el equipo llamado “Caseta”.



4. Por último, se ha de realizar la autenticación a la Raspberry Pi, por lo que hay que introducir el usuario y la contraseña de acceso al escritorio de la Raspberry Pi.



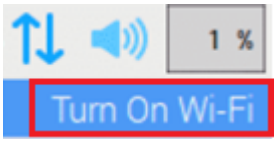
Una vez establecida la conexión con el servidor ya puede interactuar con la caseta desde el dispositivo que desee.

3. Conexión wifi

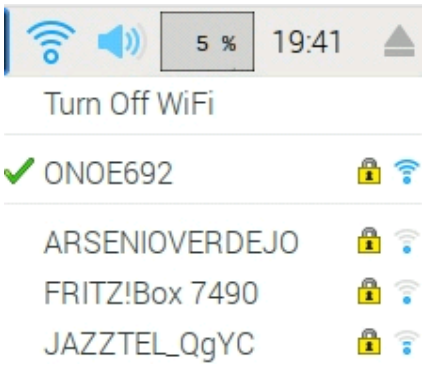
La conexión de la caseta a internet es necesario para que la hora sea correcta cuando se produzcan cambios de hora, y para poder controlar la caseta desde otro dispositivo. La conexión a internet se puede realizar mediante wifi. Para conectarse a una red wifi sigue los siguientes pasos:

1. Debe pulsar al icono que representa que el wifi está apagado.

2. Seleccione Encender (*Turn On* en inglés).



3. Elija la red.



4. Conexión con agua

Para realizar la conexión de agua potable a la caseta debe conectar una manguera de la red de agua potable de su propiedad a la entrada de agua de la caseta.

5. Conexión con luz

Para que la caseta funcione correctamente se le debe de suministrar energía eléctrica, 220V CA, para ello conecte el enchufe que se encuentra en la parte posterior de la caseta a una toma de electricidad de su propiedad.