PiCACSSO

Raspberry Pi

Control Agua Caliente

Sanitaria Solar

Autor: **Juan Villa Martínez**
Grado en Ingeniería Informática
Sistemas Empotrados

Consultor: **Jordi Bécares Ferrés**
Profesor: **Pere Tuset Peiró**

18/Junio/2017

FICHA DEL TRABAJO FINAL

Título del trabajo:	*PiCACSSO: Control de agua caliente sanitaria solar mediante una red de Raspberry Pi.*
Nombre del autor:	*Juan Villa Martínez*
Nombre del consultor/a:	*Jordi Bécares Ferrés*
Nombre del PRA:	*Pere Tuset Peiró*
Fecha de entrega:	07/2017
Titulación::	*Grado en Ingeniería Informática*
Área del Trabajo Final:	*Sistemas Empotrados*
Idioma del trabajo:	*Castellano*
Palabras clave	*Raspberry Pi, IOT, ACS*

Resumen del Trabajo*:*

La finalidad de este trabajo de fin de grado, que se incluye en el área de los sistemas empotrados, consiste en el desarrollo de un sistema de control automático que gestione el origen del ACS utilizada en mi vivienda.

Es un proyecto personal y totalmente adaptado a las características de mi vivienda tras la instalación de un equipo solar térmico de tipo termosifón. Ante la imposibilidad de instalar un dispositivo de control (centralita) cableado, se ha tenido que realizar un control totalmente manual del origen del ACS, seleccionado bien la procedente del termosifón, bien la procedente del acumulador auxiliar.

La metodología utilizada a consistido en un estudio previo de si era posible realizar algún sistema que se pudiera encargar del control del ACS, que dispositivos se podrían utilizar, que características se deseaba que tuviera el sistema, y si era posible tenerlo listo a tiempo para la presentación del TFG.

El resultado consiste en cuatro miniordenadores Raspberry Pi (con diferentes sensores y actuadores) conectados entre sí mediante una red Ethernet y dispositivos PLC. También se ha obtenido tres aplicaciones (dos en Java y una en PHP) que se encargan de la gestión del sistema.

Como conclusión, se han cumplido todos los objetivos previstos en la planificación inicial, consiguiendo un sistema funcional que da una solución práctica a un problema real.

Abstract:

The purpose of this end-of-degree work, which is included in the embedded systems area, is the development of an automatic control system that manages the origin of the ACS used in my home.

It's a personal project and it's totally adapted to the characteristics of my house. After the installation of a solar thermal equipment, and by the impossibility of installing a wired control device, it has been necessary to carry out a totally manual control of the origin of the ACS, selected either from the thermosyphon or from the auxiliary accumulator.

The methodology used consisted of a preliminary study of whether it was possible to accomplish a system that could be responsible for controlling the ACS, which devices could be used, what characteristics the system wished to have, and whether it was possible to have it ready in time for the presentation of the project.

The result consists of four Raspberry Pi mini-computers (with different sensors and actuators) connected together by an Ethernet network and PLC devices. It has also been obtained three applications (two in Java and one in PHP) that are in charge of the management of the system.

In conclusion, all the objectives foreseen in the initial planning have been fulfilled, obtaining a functional system that gives a practical solution to a real problem.

SUMARIO

Lista de figuras

Tablas

1 INTRODUCCIÓN

De media, el ACS o agua caliente de uso sanitario (también denominada de baja temperatura) representa entre el 25 y el 30% del consumo de energía que se realiza habitualmente en una vivienda. Este agua caliente que consumimos precisa de una temperatura ideal de salida de entre 37ºC y 42ºC.

En España, desde el año 2006, la normativa del nuevo Código Técnico de la Edificación o CTE [1] indica que es obligatoria la instalación, en general, de equipos solares térmicos de agua caliente sanitaria, para todas las nuevas construcciones residenciales o rehabilitadas, además de un equipo de apoyo o auxiliar

El CTE especifica que el equipo solar térmico ha de proporcionar un mínimo de entre el 30 y el 70% del agua caliente consumida, en función de la zona climática y el consumo en litros de ACS a 60ºC estimado por persona y día.

Para las construcciones anteriores a la entrada en vigor del nuevo CTE, no existe obligación de instalación, pero dado que algunas comunidades autónomas conceden subvenciones a su instalación, y en vista del ahorro energético que se produce, es muy recomendable la instalación de un equipo solar térmico que ayude a reducir el consumo energético.

El interés personal del proyecto viene dado por el hecho de poseer un equipo solar térmico instalado hace un par de años, pero ante la falta de una preinstalación existente, no se pudo instalar una centralita que controlase el uso del equipo solar térmico o del equipo auxiliar. Así, el control del origen del ACS es totalmente manual.

Y por tanto, la intención principal del presente proyecto consiste en la creación de un sistema de control que se encargue de decidir a el origen del ACS en función de la temperatura sistema.

1.1 Contexto y Justificación del Trabajo

Este es un proyecto totalmente personal y adaptado a las condiciones de mi vivienda unifamiliar adosada, adquirida de nueva construcción en 2005. La fuente de ACS instalada por el constructor en mi vivienda es una caldera de gasóleo mixta (calefacción y ACS), con un consumo medio anual aproximado de 1.500 litros de combustible.

Para tener disponible ACS en cualquier momento, la caldera tiene que estar encendida en todo momento, funcionando cada cierto tiempo para mantener el agua de su circuito primario por encima de 60ºC (según el manual de instalación). Este hecho se traduce en un consumo innecesario de gasóleo, con la consiguiente emisión de CO2 y otros productos contaminantes a la atmósfera, y en un desgaste de la caldera.

Para evitar los continuos encendidos y apagados, se instaló un acumulador de agua caliente, un interacumulador Fagor AFE-150 N1 conectado al circuito del agua caliente de la calefacción, tal y como se muestra en la figura 1. Esto permitía, salvo en invierno, encender la caldera una vez al día para calentar el agua del termo. De esta forma se reducía el consumo de energía y la consiguiente emisión de gases contaminantes.

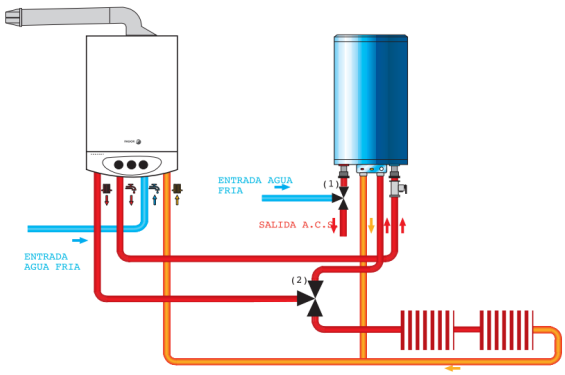


Figura 1: Instalación del AFE-xxx N1. (fuente: http://www.climamania.com/afe150n1.html)

Los hechos que llevaron a la decisión de instalar un equipo solar térmico fueron:

- Los sucesivos aumentos del precio de los combustibles (por ejemplo, el gasóleo de calefacción ha pasado de los 0,801€/litro en 2012 a 1,042€/litro en 2016 [3], lo que supone un aumento de gasto de unos 360€ anuales).
- La convicción personal de las ventajas que la energía solar proporciona en cuanto al hecho de ser una energía no contaminante que ayuda a la reducción de gases de efecto invernadero.
- La fiabilidad y duración de los equipos y la reducción de su coste.
- El ahorro que se obtendrá una vez amortizada la inversión,

Tras sopesar las diferentes posibilidades y costes de la instalación, se decidió instalar un equipo solar térmico de tipo termosifón IBESOL IB2 300 TG (el mismo equipo que el de la figura 13). Sin embargo, debido a la imposibilidad de introducir el cableado en la instalación eléctrica existente, no se pudo instalar un dispositivo de control o *centralita* que se encargase de controlar el sistema.

Al inicio de este proyecto, el sistema de control del ACS consta únicamente de un termómetro digital que muestra la temperatura del agua acumulada en el termosifón, siendo el control totalmente manual, lo que conlleva:

- Comprobar la temperatura del agua acumulada en el termosifón.
- Comprobar el estado de la caldera, y encenderla si es necesario.
- Calentar el agua del termo-acumulador auxiliar.

Todo esto requiere también de la apertura o cierre de diferentes válvulas o llaves de paso para que el agua caliente provenga del acumulador auxiliar o del termosifón.

El esquema de la instalación inicial es el que se muestra a continuación en la figura 2:

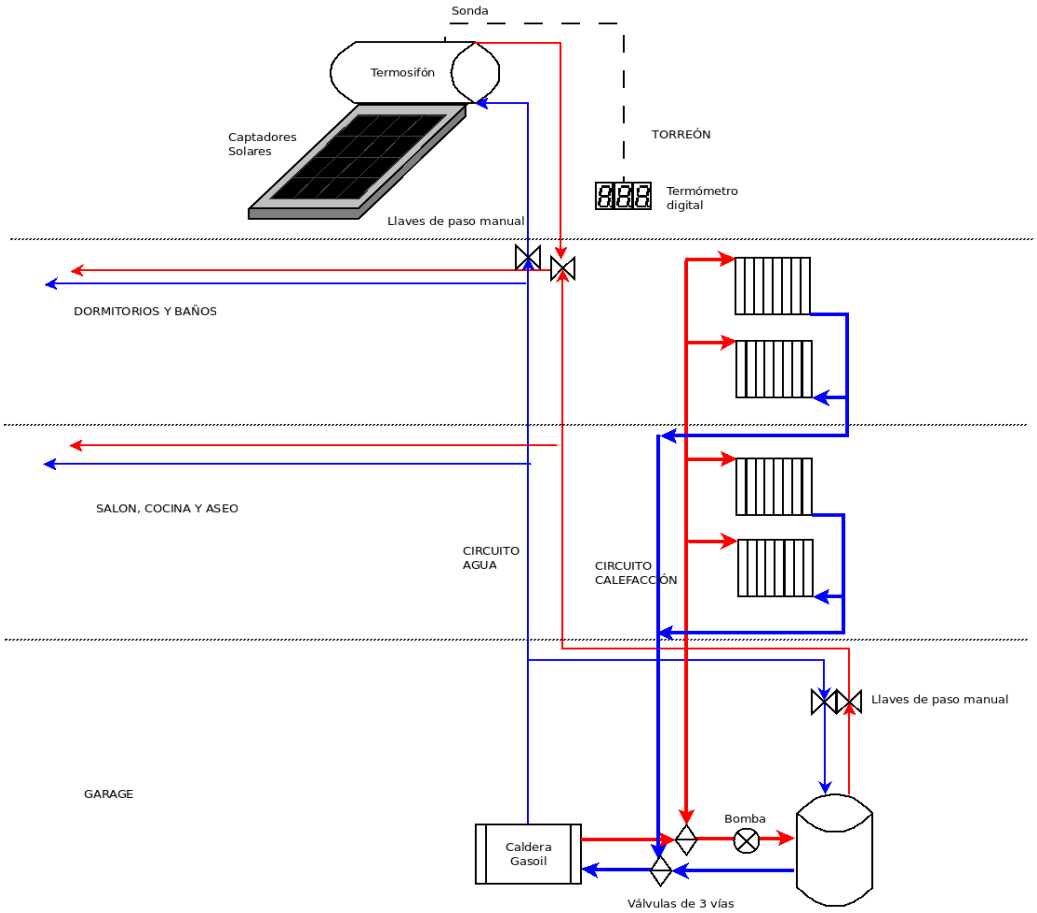


Figura 2: Esquema de la instalación inicial.

Por tanto, el principal propósito de mi proyecto es obtener un sistema, que utilizando dispositivos empotrados, se encargue de elegir el origen del ACS, y que en caso de necesidad sea capaz de calentar el agua y acumularla en el termo-acumulador auxiliar, para de esta forma, tener agua caliente siempre que se necesite.

1.2 Descripción del trabajo

En el mercado existen centrales de control o *centralitas* que permiten la automatización de la instalación de ACS solar a un coste razonable, pero en su gran mayoría son cableadas, lo que en mi caso implicaba un elevado coste de instalación, al tener que hacer reformas en la vivienda.

Al realizar la asignatura de Sistemas Empotrados, perteneciente al Grado en Ingeniería conocí los micro-controladores, y me plantee la posibilidad de realizar como trabajo de fin de grado, un sistema de control del ACS solar adaptado a mi vivienda utilizando sistemas empotrados, y así obtener una instalación final como la que se muestra en el siguiente esquema:

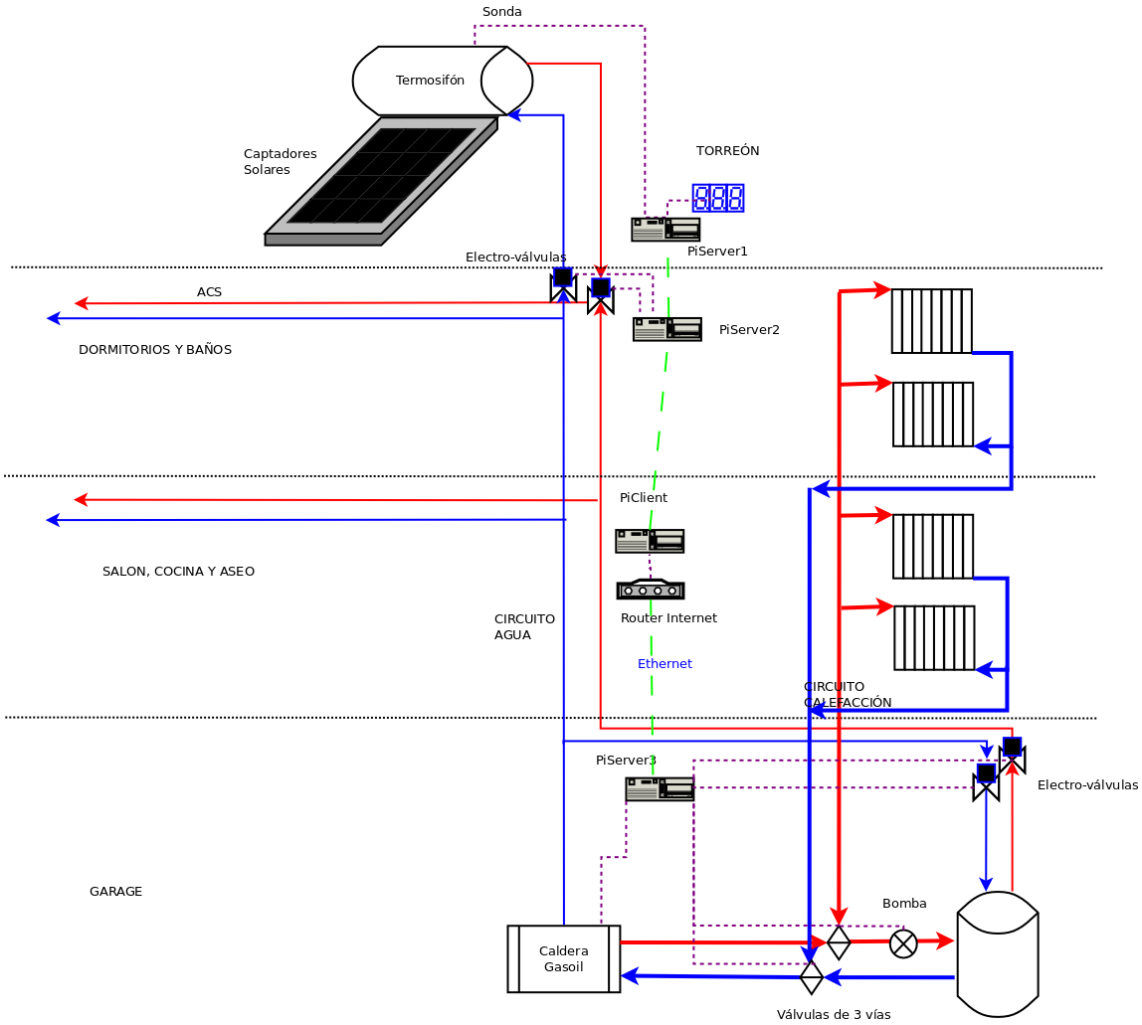


Figura 3: Esquema final previsto

Inicialmente se valoró la posibilidad de utilizar micro-controladores (Xbee, OpenMote,...) conectados mediante Wi-Fi, y crear así una red WSN (*Wireless Sensor Network* o red de sensores sin hilos) junto a un miniordenador Raspberry Pi, que se encargaría de controlar dichos micro-controladores.

Finalmente, para evitar problemas con las conexión Wi-Fi, se tomó la decisión de utilizar miniordenadores Raspberry Pi, conectados mediante red *Ethernet*, usando dispositivos PLC en aquellos lugares donde no sea posible llevar una cable *Ethernet*. Tres de las Raspberry actuarán como servidores, y enviaran información a la cuarta Raspberry (Cliente) cuando este se lo solicite.

La disposición y función de las diferentes Raspberry sería la siguiente:

- **PiServer1**: se encargará de detectar la temperatura almacenada en el termosifón, utilizando para ello la sonda de temperatura existente en dicho el termosifón. Opcionalmente, se conectará otra sonda de temperatura que detecte la temperatura exterior, y que permitirá observar la perdida de temperatura en el termosifón en función de la temperatura ambiente, y así poder establecer una temperatura mínima de funcionamiento óptima.

- **PiServer2**: Situada en el falso techo de uno de los cuartos de baño, se encargará únicamente de activar o desactivar, a petición de la Raspberry Pi cliente, unos relés que a su vez activaran/desactivaran dos electroválvulas. Estas controlará permiten que el agua del termosifón acceda o no al circuito del ACS.
- **PiServer3**: Situada en el garaje, al lado de la caldera de gasóleo y del termo-acumulador auxiliar. A petición de la Raspberry cliente se encargará de:
  - ☑ Detectar el estado de funcionamiento de la caldera, y si está apagada encenderla.
  - ☑ Detectar, mediante el termostato del termo-acumulador, que el agua ha alcanzado la temperatura optima.
  - ☑ Activar o desactivar las electroválvulas y bomba que controlan el circuito de agua de calefacción para calentar el agua del termo-acumulador.
- **PiClient**: Situada en la entrada de la casa, estará conectada mediante Ethernet al router de Internet, y se encargará de controlar todo el sistema, así como de almacenar en una base de datos un histórico de temperaturas.

1.3 Objetivos del TFG.

El objetivo principal es tener siempre agua caliente disponible para consumo, para lo cual se han de cumplir los siguientes objetivos por orden de importancia:

☑ **Detección de la temperatura del agua acumulada en el termosifón.**

Este es el primer objetivo, ya que es estrictamente necesario utilizar la sonda NTC de temperatura incorporada de fábrica en el termosifón. Como única referencia de la sonda, se posee unas hojas fotocopiadas y escaneadas en pdf, con las especificaciones del termistor y la tabla de correspondencias entre los valores máximo minino y promedio de la resistencia y la temperatura, tal y como se muestra en la figura 4.

10.00kΩ／25 ℃

温度（℃）	最大抵抗（kΩ）	標準抵抗（kΩ）	最小抵抗（kΩ）	温度許容差（℃）
50	4.239	4.160	4.083	− 0.6 + 0.6
51	4.103	4.026	3.949	− 0.6 + 0.6
52	3.972	3.896	3.821	− 0.6 + 0.6
53	3.846	3.771	3.697	− 0.7 + 0.7
54	3.725	3.651	3.579	− 0.7 + 0.7
55	3.609	3.536	3.464	− 0.7 + 0.7
56	3.496	3.425	3.354	− 0.7 + 0.7
57	3.388	3.318	3.249	− 0.7 + 0.7
58	3.284	3.215	3.147	− 0.7 + 0.7
59	3.184	3.116	3.049	− 0.7 + 0.7
60	3.087	3.020	2.954	− 0.8 + 0.8
			2.862	− 0.8 + 0.8

Figura 4: Correspondencia entre T (ºC) y R(kΩ) de la sonda NTC.

☑ **Conexión de todos los micro-ordenadores mediante red *Ethernet*.**

Se conectarán todos a red mediante la conexión *Ethernet* integrada en el dispositivo. En donde no sea posible realizar una conexión directa mediante cable *Ethernet*, se recurrirá a un adaptador *PLC* que transmita la señal *Ethernet* mediante los hilos de los enchufes de corriente.

- ☑ **Detectar el estado del termostato del acumulador auxiliar.**

  La detección de la temperatura del agua almacenada en el acumulador auxiliar determinará si es necesario encender la caldera para calentarla.

- ☑ **Control electrónico remoto del encendido y apagado de las diferentes electroválvulas.**

  El sistema ha de controlar a distancia, mediante órdenes enviadas a los diferentes micro-ordenadores, el encendido y apagado de las diferentes electroválvulas para proporcionar agua caliente del origen deseado.

- ☑ **Diseño de un sistema de control automático.**

  Este sistema de control ha de automatizar la elección del origen del agua caliente sanitaria, en función de su disponibilidad, aunque por defecto se preferirá siempre la calentada por energía solar. En caso de no haber disponible, ha de ser capaz de proporcionar agua calentada mediante la caldera de gasóleo.

- ☑ **Diseño de un sistema de control manual.**

  Este modo permitirá activar o desactivar determinados componentes del sistema. También permitirá cerrar la aplicación o reiniciar y apagar todos los dispositivos.

- ☑ **Detectar el estado de funcionamiento de la caldera.**

  Si es posible se intentará determinar si la caldera se encuentra o no en funcionamiento, y si no, encenderla. Para ello se dispone en formato PDF del manual de la caldera: "*Instrucciones de Instalación y Funcionamiento Clima Mix, Clima Mix FD-30*" enviado por el fabricante de la caldera.

- ☑ **Crear un página web sencilla de información y control.**

  Esta web ha de mostrar la temperatura del agua acumulada en el termosifón. Además, desde esta página web se efectuará el control manual.

Se establecen dos objetivos secundarios:

- ☑ **Aplicación web que muestre la evolución histórica de la temperatura**

  Se mostrará de forma gráfica, la evolución de la temperatura con el tiempo. De esta forma se podrá controlar cuantos días ha calentado el sol el agua al año, la temperatura máxima y mínima del agua almacenada en el termosifón en función de la temperatura ambiente, horas de funcionamiento de la caldera para calentar el ACS, etc.

- ☑ **Diseño de un modo autónomo.**

  Este modo autónomo permitirá a los diferentes dispositivos servidores funcionar sin la intervención del dispositivo cliente, debido a una caída o fallo en dicho dispositivo.

1.4 Enfoque y método seguido.

Como en cualquier otro proyecto, la primera tarea a realizar es la fase de información y de formación. Esta fase permite determinar si es posible llevar a buen fin el proyecto, y lo que es más importante, en el tiempo establecido.

A la hora de realizar un proyecto, existen 3 estrategias posibles para determinar si es posible llevar a buen puerto los objetivos principales establecidos,

- ☑ **Desarrollo de un producto totalmente nuevo.**

  En este caso, se partiría desde cero, y se desarrollaría el producto totalmente adaptado a nuestros objetivos y necesidades. Evidentemente, esta no es la mejor estrategia a elegir, ya que cualquier desarrollo desde cero conlleva una gran cantidad de recursos, tanto materiales como de tiempo.

- ☑ **Adaptación de un producto existente.**

  En este tipo de desarrollo se busca un producto existente que posea algunas de las características buscadas en el nuevo producto, y si es posible, se realizan las modificaciones oportunas para que realice las funciones deseadas.

  En el caso que nos ocupa, no es posible modificar centralitas ya existentes, al ser generalmente equipos cerrados y con muy poco o ninguna información, y en la mayoría de los casos, con *copyright*, lo que no permite modificarlos sin permiso del fabricante.

  Además, la adaptación de un producto preexistente puede reducir la funcionalidad del nuevo producto, bien por no poder modificar alguna característica, bien porque posee alguna limitación.

- ☑ **Desarrollo de un producto nuevo basado en equipos y elementos ya existentes.**

  Finalmente, en este caso se parte de elementos preexistentes con utilidades genéricas y que podemos adaptar a nuestras necesidades.

  Este es la estrategia elegida: coger un micro-controlador/micro-ordenador existente, así como diferentes elementos existentes en el mercado (como controladores LED, adaptadores de señales, etc.) de los que se posee bastante información, y adaptarlos a nuestras necesidades. Esto permite centrarse en los aspectos de funcionalidad del sistema.

En cuanto al desarrollo del proyecto, se ha efectuado, siempre que se ha podido, realizando modificaciones poco a poco, de tal forma que sea posible comprobar el buen funcionamiento de las aplicaciones. Esto no ha sido siempre posible, ya que por ejemplo, para comprobar que el cliente transmitía correctamente las instrucciones, el servidor debía estar a la espera, y el servidor no se podía comprobar si el cliente no estaba funcionando correctamente.

1.5 Planificación del Trabajo.

El trabajo se dividió en cuatro fases o etapas bien definidas. Las tres primeras se consideraron como obligatorias para dar por completado el proyecto, tal y como se había previsto inicialmente. La cuarta o fase de extras se pensó para, si no existían retrasos o problemas, dotar al sistema de algunas características que aunque no eran estrictamente necesarias para el funcionamiento del sistema, si le proporcionaban un valor añadido.

[1] Fases del proyecto.

A continuación se detallan las fases en que inicialmente se dividió la planificación del trabajo. Aunque el trabajo se dividió en cuatro fases, se ha añadido una quinta fase que se corresponde con la realización de esta memoria, y de la presentación final.

Fase 1 – Estudio.

Esta fase se dividió en dos etapas:

☑ **Estudio**: se realizó el estudio y búsqueda de información acerca de las diferentes tecnologías disponibles, seleccionando las que se han considerado como mejores opciones para desarrollar el proyecto.

Aunque la fase de estudio y búsqueda de información se concentró principalmente en esta fase, se ha seguido realizando a lo largo de todo el proyecto, bien para solventar algún problema, bien para implementar o mejorar alguna característica.

☑ **Planificación**: una vez realizado el estudio, se realizó una primera planificación, adaptando la duración prevista de cada fase a la duración disponible y a las fechas de entrega de las diferentes PEC's. Dicha planificación fue detallada mediante un diagrama de Gantt (Ver figura)

Fase 2 – Arquitectura y simulación.

Esta fase se subdividió en 3 etapas:

☑ **Arquitectura**: Montaje de la arquitectura, consistente en la instalación y configuración del sistema operativo y el software necesario en los dispositivos.

En esta fase también se instalará y configurará el entorno de desarrollo integrado (IDE) NetBeans en un ordenador de sobremesa. Desde dicho y mediante un compilador cruzado, se compilará y enviará la aplicación a las Raspberry.

☑ **Software Cliente-Servidor**: diseño e implementación tanto del software cliente como del servidor.

☑ **Simulación**: comprobación del intercambio de mensajes entre el cliente y los servidores, y simular por software los posibles estados de funcionamiento.

Fase 3 - Puesta en marcha.

Esta fase se ha subdividido en las 4 etapas:

☑ **Sensores y actuadores**: esta fase comprende la conexión de los diferentes sensores y actuadores correspondientes a cada Raspberry Pi, comprobando su funcionamiento. Las pruebas se realizaron inicialmente utilizando leds y pulsadores, y tras comprobar su funcionamiento, se comprobó con y en la medida de lo posible, sin estar instalados, y posteriormente instalados en su posición final.

☑ **Modo manual y modo automático**: comprobación del funcionamiento del sistema en el modo manual y automático. Estas dos fases son intercambiables, e incluso realizarse simultáneamente.

☑ **Aplicación Web**: Diseño de la aplicación de control utilizable mediante un navegador web.

Fase 4 – Extras

En esta fase, que transcurrirá a la par que la preparación del código final, se pretende si es posible, y el resto de fases se encuentran funcionando, realizar dos tareas que se corresponden con los objetivos secundarios:

☑ **Modo autónomo**: en este modo de funcionamiento, los dispositivos servidores serán capaces de funcionar de modo autónomo sin la intervención del dispositivo cliente.

☑ **Gráfico Web**: página web que muestra de manera gráfica la evolución de la temperatura a lo largo del tiempo.

Fase 5 – Memoria y presentación

En esta fase se creará la presente memoria del proyecto, así como una presentación en la que se explicará de forma clara y sencilla, el trabajo realizado y los resultados obtenidos.

[2] Diagramas de Gantt

Para realizar la planificación temporal de las diferentes fases y sus tareas se han utilizado los diagramas de Gantt. Estos diagrama de Gantt ha sido creados mediante una aplicación en línea proporcionada gratuitamente por un mes por "*Smartsheet: Work Management and Automation Solutions*".[3]

La siguiente figura 5 se corresponde con el diagrama de Gantt previsto inicialmente .

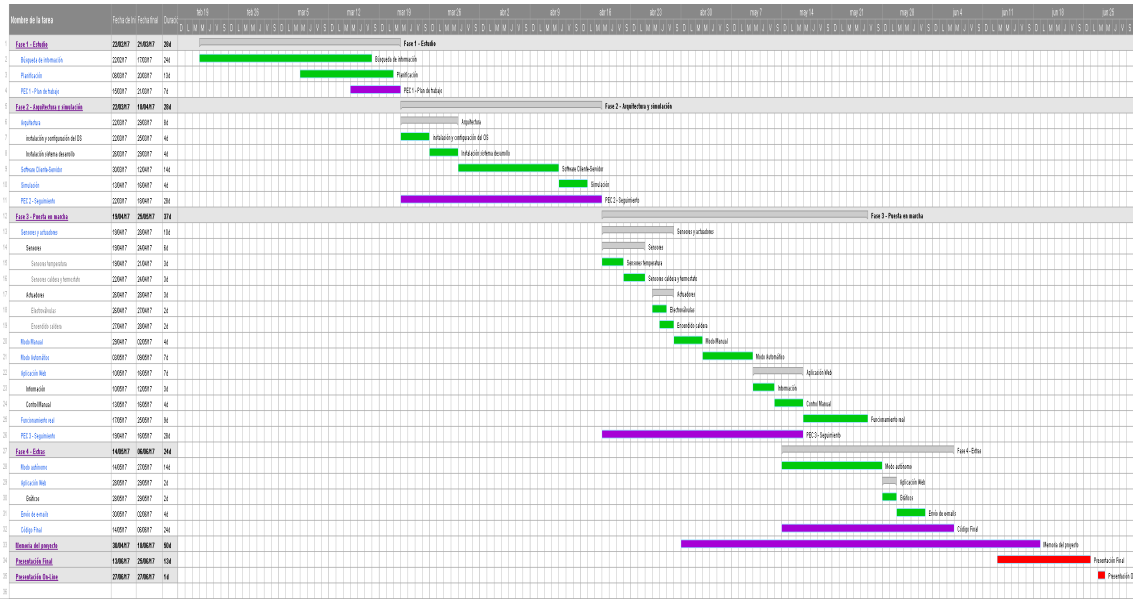


Figura 5: Diagrama de Gantt Inicial

Sin embargo, como suele ser común a la mayoría de proyectos de cualquier ámbito, tras esta planificación inicial, se producen retrasos debidos a imprevistos, bien por una fallo en la planificación, bien por algún problema externo al proyecto.

Algunos de los retrasos se han debido a:

- ☑ Retraso al recibir parte del material. Este hecho supuso que no se pudiera realizar en el tiempo previsto la etapa de “sensores y actuadores” dentro de la fase de “Arquitectura y simulación”.

  Sin embargo, este retraso permitió avanzar las fases de desarrollo del “modo Manual” y del “modo Automático”, que en lugar de realizarse en la “Fase 3 – Puesta en marcha”, se desarrollaron completamente en la Fase 2.

- ☑ En la fase 3 también se produjo un retraso en la recepción de parte del material de fontanería, lo que impidió el montaje de las electroválvulas en el tiempo previsto.

  Este tiempo fue aprovechado adelantar el desarrollo y parte de los extras, como el almacenamiento de las temperaturas (exterior y del agua del termosifón) en una base de datos, generar un gráfico con dichos datos. También se utilizó para implementar extras no previstos inicialmente y que surgieron sobre la marcha: añadir la posibilidad de enviar un correo electrónico en caso de algún problema y mostrar la temperatura en un display LED.

Ninguno de estos cambios implicó una modificación en las fechas de finalización previstas, ya que todas se mantuvieron modificando el orden de las tareas.

Por tanto, el diagrama final ha quedado como el que se muestra en la siguiente figura 6.

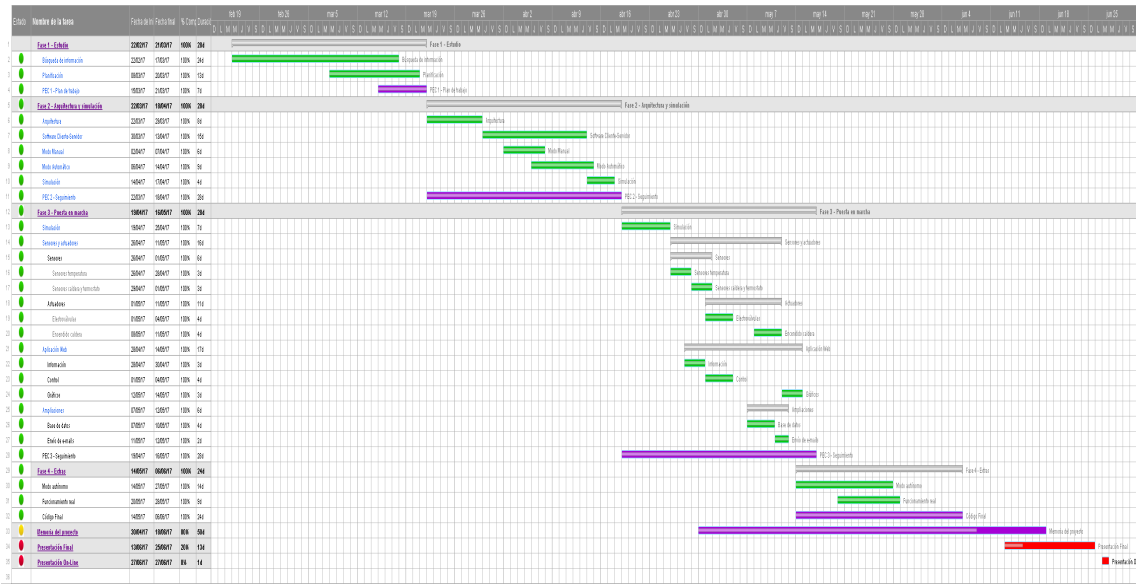


Figura 6: Diagrama de Gantt Final

Comentar que los gráficos se han colocado ampliados en el anexo 10.8.

1.6 Valoración de riesgos

En todo proyecto se producen incidencias. Estas que pueden conseguir desde que se retrase el proyecto hasta que sea imposible terminarlo. El éxito o fracaso de un proyecto depende en gran parte de que se hayan valorado correctamente sus riesgos, gastos, recursos,..Es por ello, que una parte importante de un proyecto es, antes de comenzarlo, realizar una valoración de riesgos.

En la siguiente tabla se encuentra la valoración de los riesgos inicial, y se han marcado en en color azul, la valoración de algunos de ellos si se han modificado su previsión durante el proyecto.

Nombre	Causa	Consecuencia	Tipo de incidencia.	Probabilidad	Impacto	Solución
Falta de Stock	Falta o retraso en el suministro de alguno de los componentes.	Se retrasa el montaje físico de los componentes y la puesta en marcha	~~Leve.~~ Media.	~~Baja.~~ Media.	~~Bajo.~~ Medio.	Localizar los componentes en otros proveedores. En este caso se ha esperado, aprovechando el tiempo en otras tareas.
Retraso en el desarrollo	Problemas en el establecimiento de la conexión entre las Raspberry por problemas en la red eléctrica que hiciera inviable el uso de los PLC.	Podría hacer que se retrasasen el resto de fases o se cancelara alguna.	Grave.	Media.	Alto.	Si es problema de la red eléctrica se podría cambiar de fabricante de PLC. Otro tipo de problemas de conexión requerirían plantearse la elección de dispositivos.
Retraso en la puesta en marcha.	Problemas a la hora de controlar los dispositivos . Problemas a la hora de leer las sondas y/o termostatos.	No se podrían hacer comprobaciones reales del sistema.	Media.	Media.	Media.	Podría simularse por programa , tanto el control de los dispositivos, como la lectura de las sondas y termostatos.
Retraso en los extras.	Retraso en el desarrollo de los extras	Cualquier retraso en las fases anteriores incidirá en esta, haciendo que no pueda completarse.	Leve.	Alta.	Bajo.	El sistema funcionará perfectamente sin los extras.
Fallos de hardware	Problemas con el hardware utilizado, especialmente con el PC de desarrollo, pueden conllevar perdidas de parte del trabajo.	Retrasos en el desarrollo en curso. Incluso pérdida de parte del trabajo ralizado.	~~Leve.~~ Media.	~~Baja.~~ Media	Alto	Tener copias de seguridad Tener un ordenador de reserva.

Tabla 1: Tabla con la valoración de riesgos.

1.7 Recursos empleados

Para la realización del proyecto se han utilizado los siguientes recursos:

- ☑ Recursos hardware:
  - ✗ PC de sobremesa con procesador AMD Phenom X4 965, 8GB de RAM y sistema operativo Ubuntu 17.04.
  - ✗ 4 miniordenadores Raspberry Pi de diferentes modelos: dos model 1B, un model 2B y un model 3B. Cada uno con su correspondiente fuente de alimentación.
  - ✗ Tarjetas microSD de diferentes tamaños y disco duro SSD de 32GB con carcasa USB.
  - ✗ Placa ADS1115, conversor analógico-digital de 4 canales de 16 bits, con conexión mediante bus I2C. (Características en anexo 10.1).
  - ✗ Dos display LED (uno blanco y otro amarillo) de 4 dígitos de 7 segmentos de 0,56" y controlador/adaptador (*backpack*) I2C modelo HT16K33 de Adafruit.
  - ✗ 2x placa *Level Shifter* (conversor de nivel lógico bidireccional 5V ↔ 3,3V) (Características en anexo 10.2).
  - ✗ Placa con 2 y 4 relés opto-aislados (características en anexo 10.3).
  - ✗ Sonda analógica NTC 10K.
  - ✗ 2 Dispositivos PLC TP-Link PA8010 de 1200Mbps (Características en anexo 10.4).
  - ✗ Componentes electrónicos como resistencias y diodos LED y otros componentes como *protoboards,* cables con conectores Dupont (M-M, M-H y H-H) y conectores.
  - ✗ Soldador y estaño. Polímetro digital.
  - ✗ Para la parte de fontanería, se ha utilizado corta-tubos, botella-soplete, estaño, decapante, taladro, llaves de paso, diferentes uniones y tubo de cobre de 22mm, tubos flexibles de acero, … y por supuesto, un par de llaves inglesas
  - ✗ Para la parte eléctrica, cable aislado de 1'5mm en colores azul, negro y amarillo-verde, conectores, bases de 3 enchufes Schuko hembra, cajas de superficie, tubo corrugado y un destornillador busca-polos.
  - ✗ Relés electromecánicos Finder 40.61 de 250V y 16A (figura 45).
- ☑ Recursos software:
  - ✗ Sistema operativo Raspbian Jessie Lite, una versión de Linux Debian específico para la Raspberry Pi, que funciona en modo comando sin entorno gráfico.
  - ✗ Java v8. Instalado en cada Raspberry, ha sido el lenguaje de programación en el que se ha creado la aplicación cliente y servidor.

- ✗ NetBeans 8.2: IDE que se utilizado para programar, compilar y distribuir a las Raspberry las aplicaciones cliente y servidor desarrolladas en Java. También se ha utilizado para desarrollar la aplicación Web en el lenguaje PHP.
- ✗ Librerías Java. En el desarrollo de las aplicaciones cliente-servidor se han utilizado diferentes librerías. A continuación se relacionan aquellas utilizadas que no forman parte de la API de Java:
  - ➢ **Pi4J**: Librería que proporciona acceso completo a las capacidades de E/S (puertos GPIO) de la Raspberry [4],[5].
  - ➢ **GSON**: Librería de código abierto desarrollada por Google para la serialización/deserialización de objetos Java y su notación en formato JSON [14], [15].
  - ➢ **JavaMail**: librería proporcionada por Oracle para enviar y recibir correos electrónicos desde aplicaciones Java [9],[10].
- ✗ LAMP. En el cliente se ha instalado un servidor de páginas web con soporte de base de datos SQL y lenguaje PHP. En lugar de utilizar Apache como servidor de páginas web se ha utilizado Lighttpd, un servidor más ligero, y en lugar de MySQL, una variación o *fork:* MariaDB. Dentro de la aplicación web se han instalado/utilizado dos librerías:
  - ➢ **Google Charts**: librería en JavaScript de Google para dibujar gráficos a partir de un conjunto de datos [15].
  - ➢ **SevenSeg**: Librería en JavaScript para crear ”displays” de 7 segmentos en páginas web [31].

1.8 Productos obtenidos

El producto obtenido consta de dos partes principales, una hardware y una software:

- **Hardware**, constituido por los 4 micro-ordenadores Raspberry Pi, los circuitos auxiliares necesarios para encender y apagar los diferentes dispositivos, así como para obtener la temperatura a partir de la sonda integrada en el termosifón.

  En las siguientes imágenes se muestra la instalación final de las tres Raspberry que actúan como servidores, en su caja con los correspondientes dispositivos. Nota: no se ha colocado foto del cliente, ya que es una Raspberry en su caja conectada a un disco duro USB.

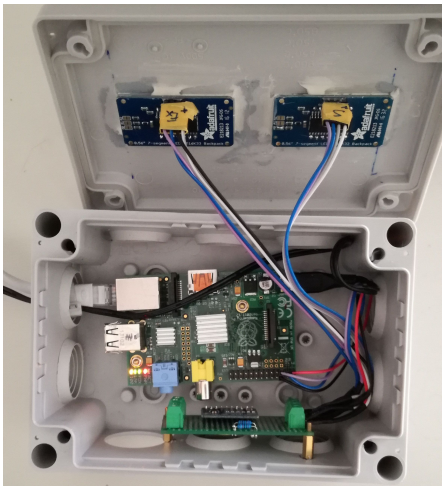


Figura 7: RPiServer1 – ADC + Displays

Figura 8: RPiServer 1 - Vista frontal.

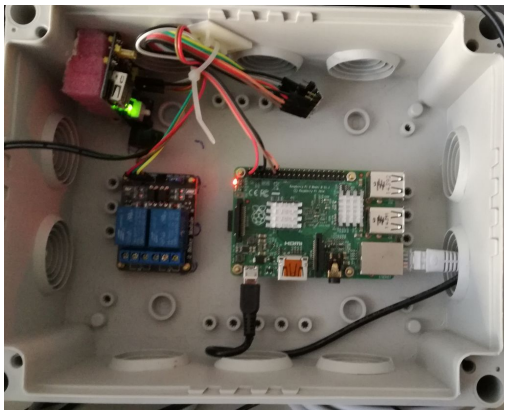


Figura 10: RPiServer1 en su caja, antes de su montaje en su posición definitiva.

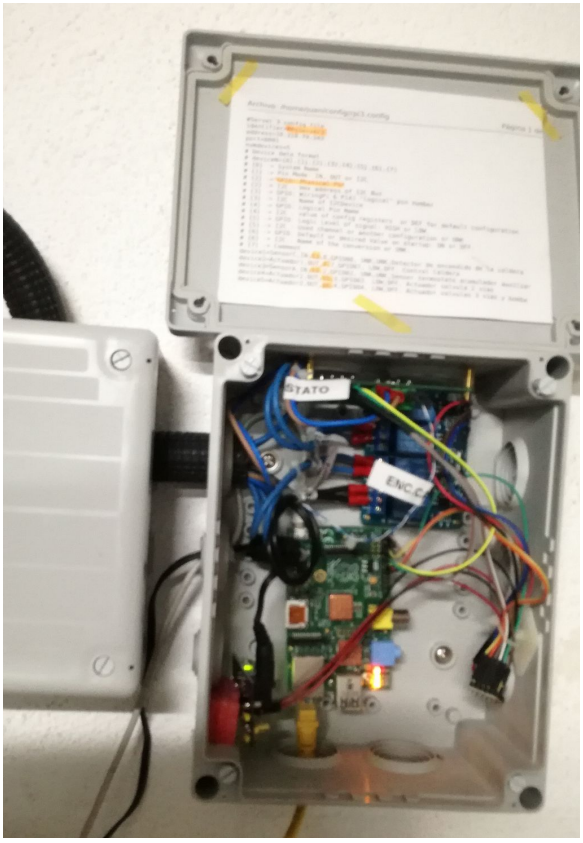


Figura 9: RPiServer3 en su localización definitiva.

- **Software**, constituido por 3 aplicaciones. 2 realizadas en el lenguaje Java, una cliente que funciona en una única Raspberry, y una servidor que funciona en el resto de Raspberry. La tercera aplicación es una aplicación web realizada en HTML y PHP, que se encarga de mostrar información sobre el estado del sistema, y permite realizar el control manual del sistema por parte de los usuarios. La aplicación servidor se ha intentado diseñar de tal forma que pueda ampliarse de una forma relativamente sencilla.

Además, aunque no es muy relevante desde el punto de vista de los sistemas empotrados, también se ha realizado :

- Una instalación o **adecuación de la instalación de fontanería** existente. Como se observa en la figura, se ha conectado cada electroválvula al circuito de agua correspondiente (fría o caliente). También se han colocado válvulas de *by-pass* manuales que permiten anular una electroválvula manualmente en caso de fallo y poder sustituirla fácilmente.

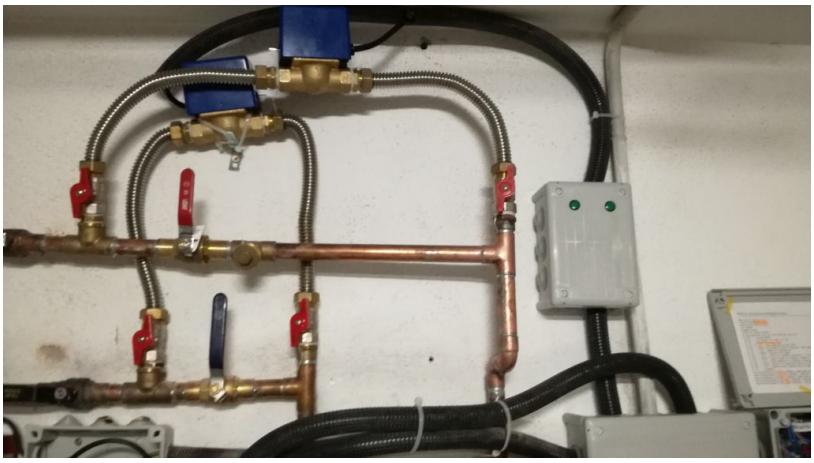


Figura 11: Nueva instalación de fontaneria. Zona termo auxiliar.

- Una **instalación eléctrica** para proporcionar energía a los dispositivos tanto electrónicos como a las electroválvulas. También se ha realizado la instalación eléctrica necesaria para detectar el estado (encendido o apagado) de la caldera de gasóleo y del termostato del termo-acumulador auxiliar, así como para encender la caldera. Para proteger la instalación eléctrica de la caldera, termo acumulador auxiliar, válvulas, y demás componentes, se ha instalado un interruptor diferencial para esa zona.

1.9 Breve descripción de los otros capítulos de la memoria.

Los siguientes capítulos de la memoria se han estructurado de la siguiente forma:

- **Antecedentes y estado del arte**: este capítulo da a conocer el estado de la tecnología utilizada en este proyecto, y su evolución en los últimos años.
- **Descripción funcional**: se explican las decisiones de diseño e implementación que se han tomado, así como una breve descripción de cada componente del sistema.
- **Descripción detallada**: se describen detalladamente, y de manera técnica, tanto la implementación de los dispositivos físicos como del software.
- **Valoración económica**: se realiza un presupuesto con los costes del desarrollo e instalación del proyecto.
- **Conclusiones**: se describen las conclusiones del proyecto.

2 ANTECEDENTES

2.1 Estado del arte

En este apartado se muestran las características de los dispositivos más importantes utilizados para el desarrollo del proyecto, así como sus diferencias, si es que existen, con otros dispositivos similares disponibles en el mercado.

[1] Equipos solares térmicos

Los equipos solares térmicos (figura 12) captan la energía de la radiación solar, mediante uno o varios captadores o placas solares. Esta energía se transfiere al fluido contenido en el circuito primario, el cual puede llegar a alcanzar los 180ºC de temperatura. Mediante un intercambiador de calor, se transfiere la energía al circuito secundario o de consumo, que contiene el ACS y que es almacenada en un termo-acumulador hasta su consumo.

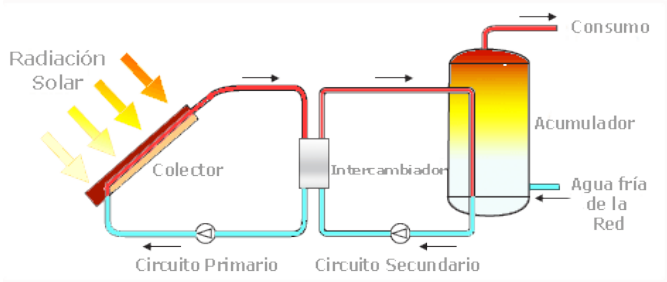


Figura 12: Esquema básico de una instalación solar térmica. (fuente: https://solar-energia.net/energia-solar-termica)

Cuando la temperatura del agua acumulada es inferior a la temperatura ideal, se requiere de un sistema de energía auxiliar (termo-acumulador eléctrico, caldera de gasóleo o de gas,…), que se encargará de realizar el calentamiento adicional hasta alcanzar la temperatura deseada.

Existen dos tipos básicos de equipos solares térmicos para baja temperatura:

☑ **Equipos de circulación natural o termosifón**, (figura 13) en los que la circulación del fluido del circuito primario se produce por las diferencias de densidad del fluido al calentarse. La zona de fluido más caliente tiene menos densidad y por tanto asciende y "flota" sobre la parte de fluido más fría. Requiere la colocación del termo-acumulador en la zona superior de los colectores.

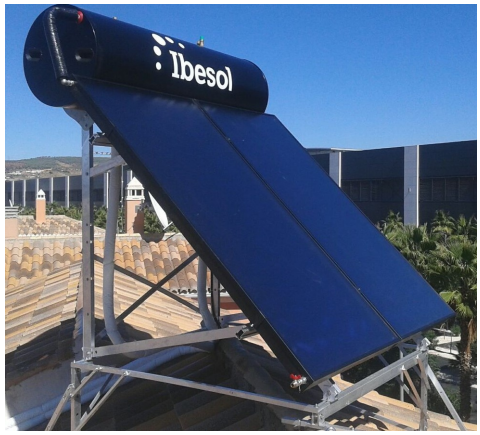


Figura 13: Termosifón (fuente: http://www.ibesol.com)

☑ **Equipos forzados** que utilizan una bomba para la circulación del fluido del circuito primario. Presenta la ventaja de que el termo-acumulador puede colocarse a distancia de los colectores. Como inconveniente, requiere la

instalación de una centralita, que en función de la temperatura alcanzada en el circuito primario, active la bomba, y de esta forma, calentar el agua del termo-acumulador.

Hace unos pocos años apenas existían fabricantes de equipos térmicos solares, pero en la actualidad hay decenas de fabricantes/vendedores en el mercado, y con gran variedad de precios en función de sus características.

Todos los equipos solares térmicos constan de 2 partes:

☑ **Captador solar**: entre sus características más importantes se encuentran la superficie de captación en m², el tipo de captador (placa plana o tubos de vacío), material de la cubierta transparente protectora (plástico, vidrio, vidrio con bajo contenido en hierro,..), aislamiento,..

☑ **Deposito o acumulador**: sus características más importantes son, junto con su tamaño, el material en que esta construido, el tipo y grosor del aislamiento y la seguridad incorporada, como válvulas de sobre-presión y vasos de expansión.

Hay que tener en cuenta también, que los equipos forzados requieren de bombas para mover el agua caliente desde los captadores solares hasta el deposito.

[2] Electroválvulas

Las electroválvula (figura 14), también denominadas ***válvulas de zona***, son dispositivos diseñados para controlar el flujo de un fluido en un conducto o tubería en modo todo o nada. Constan de dos partes:

☑ **Cabeza o parte eléctrica**, formada por un solenoide o un motor eléctrico, que al recibir corriente eléctrica actúa sobre la válvula, abriendo o cerrando el flujo del fluido.

☑ **Cuerpo, parte mecánica o cuerpo de la válvula**. Construido en diferentes materiales (latón, acero inoxidable, PVC,..) dependiendo del tipo de fluido que vaya a circular por ellas.

Aunque las electroválvulas llevan un motor eléctrico, no se deben confundir con las válvulas motorizadas, las cuales permiten posiciones intermedias.

Las características principales de una electroválvula son las siguientes:

☑ **Número de vías**. Normalmente son de 2 o de 3 vías. La de dos vías permiten abrir o cerrar el paso del fluido, mientras que las de 3 vías conmutan el flujo del fluido desde la entrada (normalmente indicada como AB) hacia una u otra de las salidas (indicadas como A o B).

☑ **Estado de la válvula en reposo**: indica el estado de la válvula cuando no se aplica corriente, y puede ser NC (Normalmente Cerrada) o NA (Normalmente Abierta).

☑ **Monoestable o biestable.** Las monoestables poseen unicamente un solenoide o un motor eléctrico que actúa sobre la válvula, de tal forma que al dejar de aplicar corriente, la válvula vuelve a su posición en reposo (NC o NA).

Las biestables suelen poseer dos solenoides o dos motores eléctricos, usando uno para abrir la válvula y otro para cerrarla. Estas por tanto, una vez se deja de aplicar la corriente mantienen el estado en que se encontraban cuando se dejó de aplicar la corriente.

- ☑ **Tipo de acción**, que puede ser directa, indirecta o mixta, según si el solenoide o motor actúa directamente sobre el émbolo que abre o cierra la válvula.
- ☑ **Material de construcción** de la electroválvula, que determina el fluido que puede utilizarse, así como la temperatura y presión máxima que soportará.
- ☑ **Tipo de alimentación**. Generalmente funcionan a 220V en corriente alterna, aunque suelen tener la posibilidad de funcionar a 12V o 24V en corriente continua.
- ☑ **Tipo de cierre**. En las válvulas de acción indirecta, se suele indicar la forma y el material utilizado para cerrar el paso del fluido. El tipo de cierre determina el tipo de fluido en que puede utilizarse. Las hay de bola (de acero, de goma,..), de clapeta, etc...

En vista de las altas temperaturas que puede alcanzar el agua en el termosifón, es muy importante tener en cuenta la característica de la temperatura de funcionamiento. Es por ello que los fabricantes mas importantes poseen una gama solar diferenciada.

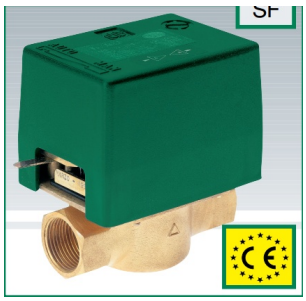


Figura 14: Valvula de 2 vías Mut SF-20 Solar.

Por economía en cuanto a consumo de corriente, habría sido deseable instalar válvulas biestables, pero son poco comunes, y su coste triplicaba el de una válvula monoestable.

A continuación, en la siguiente tabla 2, se muestran las características de las electroválvulas utilizadas.

	Válvula de 2 vías	Válvula de 3 vías
Fabricante	Mut Meccanica Tovo	
Modelo	SF20	SF20 SOLAR
Estado en reposo	Normalmente cerrada	AB → A
Acción	Indirecta	
Tipo de cierre	Bola de goma	
Alimentación	230V AC	
Consumo	6W	
Conexión	Hembra 3/4”	
Presión Nominal	10 Kg/cm2	
Temperaturas del fluido	+5 ~ +110 ºC	+5 ~ +120 ºC [150ºC]
Tiempo Apertura/Cierre	10 / 4 sg	20 / 6 sg
Precio unidad (IVA Inc.)	106 €	118 €

Tabla 2: Características de las electroválvulas utilizadas.

[3] Microcontroladores/Miniordenadores

Un sistema empotrado o embebido es un sistema de computación que ha sido diseñado para realizar un conjunto de tareas específicas. Los sistemas empotrados de diseñan alrededor de microcontroladores. Un microcontrolador posee un procesador, memoria y controladores de entrada/salida en un único chip denominado SoC (System On a Chip).

Muchos de los miniordenadores existentes hoy en día, como la Raspberry Pi, utilizan soluciones SoC, de tal forma que es difícil determinar la diferencia entre miniordenador o microcontrolador. Seguramente, las diferencia determinante entre uno sea el hecho de que los microcontroladores utilizan sistemas operativos en tiempo real.

Algunos de los microcontroladores/miniordenadores existentes en el mercado son:

☑ **Raspberry Pi** [25] [26], en la figura 18, es un miniordenador de tamaño reducido, desarrollado por la Raspberry Pi Foundation para promocionar la enseñanza de los fundamentos de informática en las escuelas y en los países en desarrollo.

Aunque no es hardware libre, su gran aceptación en el mercado con más de 11 millones de unidades vendidas ha hecho que existan multitud de miniordenadores basados en ella, con diferentes variaciones o mejoras, ejemplo de las cuales tenemos las Orange Pi [41], Banana Pi [42] u Odroid C2 [43].

☑ **BeagleBone** [35], en la figura 15, es un miniordenador de tamaño reducido basado en el procesador Sitara ARM Cortex-A8. Desarrollada y producida por Texas Instruments en colaboración con dos distribuidores/productores de componentes electrónicos: Digi-Key y Newark element14.

☑ **LPC1769** [36], en la figura 16, es un micro-controlador con un alto nivel de integración. Desarrollado por NXP para iniciarse en el uso y la programación de sistemas empotrados. Es el sistema utilizado en la asignatura de Sistema Empotrados.

☑ **Onion Omega2** [34], en la figura 17, es un miniordenador Linux diseñado específicamente como ordenador IOT, posee un tamaño reducido y combina la eficiencia de Arduino con la potencia y la flexibilidad de la Raspberry Pi.

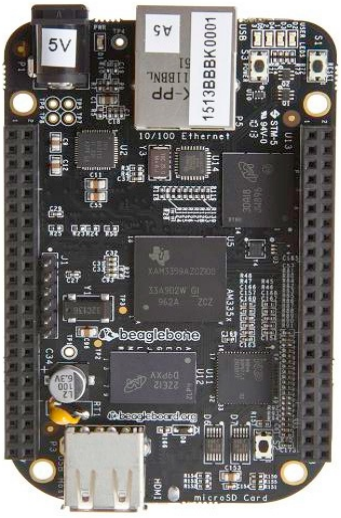


Figura 15: Beaglebone Black
(fuente: http://beagleboard.org/bone)

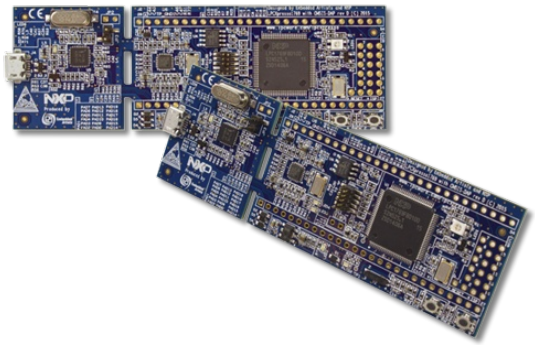


Figura 16: NXP LPC1769 LPCXPRESSO
(fuente: https://www.embeddedartists.com/)

Figura 17: Onion Omega2
(fuente: https://onion.io/omega2/)

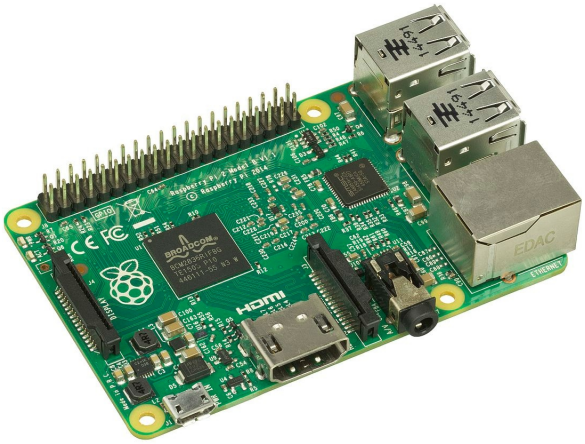


Figura 18: Raspberry Pi model 2B
(fuente: Wikipedia)

En la tabla 3 se muestran las características más importantes de los dispositivos revisados en este apartado.

Características	BeagleBone	Onion	NXP LPCEXPRESSO	Raspberry Pi
Modelo	Black Wireless	Omega2 Plus	LPC1769 with CMSIS-DAP	Model 3B
Arquitectura	ARMv7-A (32 bits)	MIPS32 (32 bits)	ARMv7-M (32 bits)	ARMv8-A (64/32 bits)
SoC	Sitara AM3358/9	Mediatek MT7688	NXP LPC1769	Broadcom BCM2837
CPU	ARM Cortex -A8 @ 1GHz 2x PRU 32-bit micro-controladores	MIPS32 24KEc @ 580MHz	ARM Cortex-M3 @ 120 MHz	ARM Cortex-A53 @ 1.2 GHz
RAM	512 MB DDR3 8-bit eMMC flash 4GB	128 MB DDR2 32 MB flash RAM	64KB SRAM para datos 512 KB flash RAM	1 GB
GPU	PowerVR SGX530 @ 200MHz	No	No	Broadcom VideoCore IV
USB 2.0	1x, 1x miniUSB	1x	No	4x
Conexiones y red	MicroSD Ethernet (10/100 Mbps) WiFi 802.11n	MicroSD, WiFi 802,11n	MicroUSB	MicroSD, Ethernet (10/100 Mbps) WiFi 802.11n Bluetooth 4.1
Periféricos	40x GPIO, 1x HDMI , 4x UART, 8x PWM LCD, GPMC, MMC1, 2x SPI, 2x I2C A/D Converter, 2x CAN bus, 4 Timers	21x GPIO, compartidos con 2x PWM, 2x UART, 1x I2C, 1x SPI, 1x I2S Audio	4x UART, 2x ADC (10 y 12 bits) 4x Hardware Timers, 8x Canales DMA 1x Ethernet con interface RMII, 3x I2C	1x HDMI, LCD, 26x GPIO, compartidos con 1x UART, 1x I2C, 1x SPI, 1x I2S Audio
Sistemas operativos	Linux (Debian, Ubuntu, Fedora, Angström,...) , Android, FreeBSD, OpenBSD, QNX, Minix, RISC OS Windows Embedded	LEDE	RTX RTOS FreeRTOS Contiki	Linux (Raspbian, Ubuntu, Arch, ….hasta cerca de 30 distribuciones) RISC OS, Chromium OS Windows 10 IOT Core
Alimentación y consumo	5V, 2.3 W max.	3.3V	3.3V	5V microUSB, 6.7W max.
Tamaño	86.4 x 53.3 mm	42.9 x 26.4 mm	35 x 140 mm	85,6 x 56,5 mm
Precio[1]	56.69 US$	29.00 US$ [2]	43.10 US$	35.94 US$

Tabla 3: Características de diferentes microcontroladores.

1 Todos los precios se han obtenido de Amazon.com para poder comparar precios. No se incluyen gastos de envío.

2 El precio incluye el dispositivo *Dock* necesario para poder conectarlo al PC y programarlo.

[4] Conexiones de Red

Dentro de las diferentes formas de establecer conexiones de red tenemos disponible:

☑ **PLC** (*Power Line Communications* o comunicación de datos mediante líneas de potencia) hace referencia a diferentes tecnologías de transmisión y recepción de datos que utilizan las líneas de transmisión energía eléctrica convencionales. La tecnología PLC aprovecha la red eléctrica para convertirla en una línea digital de alta velocidad de transmisión de datos, permitiendo, entre otras cosas, el acceso a Internet mediante banda ancha.

Los dispositivos PLC se conectan a una toma eléctrica convencional (enchufe de corriente), transmitiendo la señal que recibe en su puerto RJ-45 (Ethernet) por los cables de la corriente eléctrica y viceversa. Para ello utiliza una onda portadora (de entre 20 y 200KHz) en la que se modulan las señales digitales, de forma similar a como funciona la tecnología ADSL sobre las líneas de teléfono.

Cada dispositivo PLC posee una dirección única, por lo que antes de poder utilizarlo hay que emparejarlo con otro PLC, generalmente del mismo modelo. Según el modelo, se pueden utilizar hasta 6 u 8 en una misma red de PLC [8], ya que aunque existen estándares de facto, como las HomePlug y HomePlug AV, cada fabricante suele incluir mejoras que solo funcionan con sus equipos.

En la imagen se puede ver la aplicación del fabricante TP-Link que permite visualizar el estado de los PLC del mismo modelo sincronizados en una red.

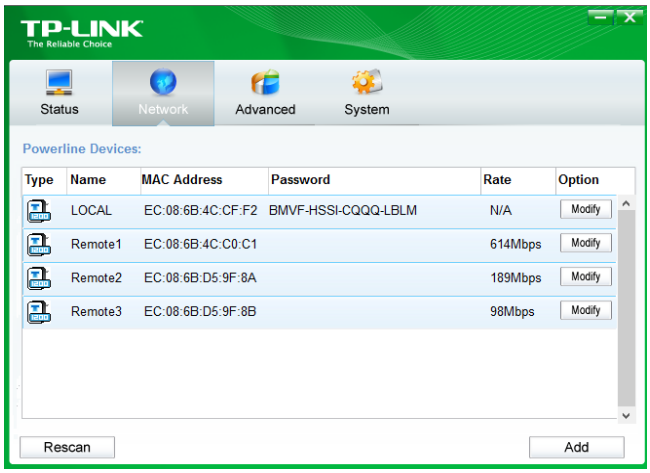


Figura 19: Estado de los dispositivos PLC en la red.

El mayor inconveniente en el uso de PLC vienen dado por las interferencias (que determinados dispositivos eléctricos provocan cuando se usan), por los picos de tensión en la línea y por el estado del cableado eléctrico, que suelen provocar una reducción en la velocidad de conexión

☑ **Wi-Fi**: hace referencia al conjunto de estándares IEEE 802.11 que permiten la conexión de dispositivos en una WLAN (*Wireless LAN*). Una de las característica más importantes del estándar IEEE802.11 es el hecho de ser una modificación del estándar *Ethernet* (IEEE 802.3), en el que se han modificado las capas físicas y de control de acceso al medio. Por tanto, las redes Wi-Fi son totalmente compatibles con los servicios de las redes Ethernet.

Uno de los problemas actuales de Wi-Fi es que sus estándares más utilizados (802.11b, 802.11g y 802.11n, que se correspondes con las velocidades de 11, 54 y 300Mbps) utilizan la banda de frecuencias de los 2,4 GHz, la misma que utilizan otros tipos de tecnologías inalámbricas como Bluetooh o Zigbee.

Esto, unido a la gran cantidad de usuarios que utilizan este tipo de tecnología hace que se produzcan numerosos problemas de saturación de la señal de radio, lo que suele incidir en un reducción de la velocidad de conexión, o en pérdidas aleatorias de la conexión.

Otro de los inconvenientes que posee el Wi-Fi, es que su alcance depende del entorno. Así, la norma 802.11n indica un radio de cobertura en el interior de hasta 120m, pero dependiendo de la ubicación y de la estructura del edificio puede no llegar ni a 10m.

Y para finalizar, el hecho de que estén expuestas a las interferencias externas (como las que puede provocar un simple horno-microondas, una lavadora centrifugando o el motor del ascensor) hace que este tipo de tecnología no sea una buena elección, ya que la fiabilidad depende de circunstancias externas.

☑ **Ethernet**: es una familia de protocolos usados habitualmente tanto en redes locales (LAN) como en el ámbito de MAN y WAN. Inventada por Xerox, fue desarrollada conjuntamente por Xerox, DEC e Intel y apareció en el mercado en 1980. Su característica principar fue su alta velocidad de 10Mbps para la época.

En 1985 se convirtió en el estándar del IEEE 802.3. Por este motivo, aunque la norma estándar es la IEEE 802.3 (y sucesivas), se conoce habitualmente por *Ethernet*. Hoy en día es, seguramente, la tecnología de redes LAN más utilizada en el mundo, pasando su velocidad de los 10Mbps iniciales en 1980 a 100Mpbs (FastEthernet), 1Gbps (GigaEthernet) y 10Gbps.

Este protocolo utiliza una topología en bus y una política de acceso al medio CSMA-CD. Soporta multitud de medios físicos, siendo el más común en LAN el cable de par trenzado de cobre, aunque también soporta cable coaxial y diferentes tipos de fibra óptica.

☑ **ZigBee**: es el nombre que se le da al conjunto de protocolos de alto nivel de comunicación inalámbrica. ZigBee surgió como un forma de estandarizar las redes MESH, y está promocionada por una alianza de distintos fabricantes. A pesar de ello, puede darse el caso de que distintos dispositivos compatibles ZigBee no puedan comunicarse entre ellos, al no existir una definición completa, .

Está basada en el estándar IEEE 802.15.4 de redes WPAN, y su funcionamiento es similar al del conocido Bluetooh, aunque con algunas diferencias debido a su diseño para ser utilizado en domótica con multitud de diferentes dispositivos de distintos fabricantes.

Entre sus características se encuentran la posibilidad de tener hasta 65535 nodos, distribuidos en subredes de 255 nodos. Además, ZigBee proporciona tanto la red de

comunicaciones como la seguridad, al poseer algoritmos de cifrado integrados, y también proporciona servicios para aplicaciones que operan encima de la capa IEEE 802.15.4 permitiendo topologías de red tan variadas como se deseen.

2.2 Estudio de mercado.

Si hojeamos cualquier catálogo de empresas instaladoras de equipos térmicos solares, o simplemente de fontanería y calefacción, nos encontramos con una gran variedad de dispositivos de control o centralitas, con una o varias entradas de sensores, y con la posibilidad de controlar de una a varias decenas de salidas. Algunas diseñadas para funcionar unicamente con determinados equipos, ya sean solares o de calefacción, de un fabricante.

Estas centralitas se pueden dividir en dos categorías:

- ☑ **Pre-programadas**. Incluyen una serie de programas para funcionar en diferentes sistemas
- ☑ **Programables**, que permiten programar, generalmente mediante una serie de comandos o pulsaciones de teclas, unas determinadas acciones en función de las entradas.

Una de las características más importante de las centralitas de ámbito doméstico es su forma de conexión con los diferentes sensores y actuadores. Esta conexión suele ser de dos tipos:

- ☑ **Cableadas**. Sistema utilizado por la mayoría de centralitas, que implica llevar un cable (generalmente de 2 o 3 hilos) a cada uno de los diferentes dispositivos que se quieran controlar (sensores y actuadores).
- ☑ **Vía radio**. Estos sistemas suelen indicar dos valores de alcance de la señal de radio. Uno superior a 100m en campo abierto, y otro de unos 25m dentro de un edificio. Sin embargo, suelen indicar que el alcance dentro de un edificio no puede garantizarse, ya que este depende en gran medida de la naturaleza del edificio.

Algunos ejemplos de centralitas serían las siguientes:

- ☑ Coltech CT04-S [37]: “CT04-S es una centralita electrónica para la regulación de instalaciones solares, con capacidad para controlar hasta cuatro sondas de temperatura y cuatro salidas para accionamiento de bomba y sistema auxiliar en sistemas formados por: - Campo de colectores solares, hasta 2 depósitos (2 depósitos de ACS o depósito ACS + depósito inercia) o depósito más piscina, sistemas de calentamiento auxiliar y sistema de refrigeración”. Posee 9 modos de funcionamiento predefinidos.
- ☑ Kaysun KCS1 [38]: “KCS1 Regulador Diferencial de Temperatura. Pantalla iluminada con modo de gráficos y textos completos. Asesoramiento de la configuración por el asistente de puesta en marcha integrado / Protección contra el sobrecalentamiento y la congelación / Programas especiales para arrancar sistemas de tubo de vacío y de vacío automático. Para 5 diferentes sistemas: Solar con deposito, Solar con piscina, Caldera con deposito, Cambio de carga de depósitos y Mezcla del retorno de la calefacción. Para

controlar la función del sistema se puede medir la energía producida - 3 Entradas para Sensores de temperatura Pt1000- 1 Salida de relé 230VAC para bombas/válvulas".

- ☑ ESCOSOL DeltaSol TT [39]: "Centralita de regulación solar especial para termosifón. 3 entradas para sondas PT1000. Salidas: 2 relés (1 semiconductor + 1 electromecánico) y salida PWM. Funciones: Termostato diferencial, sistema termosifón con calentamiento rápido, termostato temporizador, contador horas". PVP: 129€ (IVA no incluido).
- ☑ RESOL DeltaSol BX Plus [39]: "Centralita de regulación de sistema Solar/Calefacción. 8 entradas sondas (PT1000, PT500, sonda radiación CS10,..) . Salidas: 5 relés (4 semiconductor + 1 electromecánico) y 2 salidas PWM. Funciones: Comunicación VBus, tarjeta SD, contador energía, hasta 2 módulos extensión (máx. 21 sondas y 15 relés)". PVP: 395€ (IVA no incluido).

Como se puede observar, la mayoría son cableados y generalmente posen pocas posibilidades a la hora de controlar un gran número de dispositivos. Algunas centralitas, como la última de la lista anterior (DeltaSol BX Plus) incorporan la posibilidad de conectar dispositivos mediante adaptadores de interfaz Vbus/LAN, pero cada adaptador cuesta 170€ (IVA no incluido) [39].

Por tanto, parece que no existe en el mercado ningún sistema de control del ACS, por lo menos en el ámbito doméstico y a precio razonable, que posea las características del sistema de control desarrollado en este proyecto.

3 DESCRIPCIÓN FUNCIONAL.

El sistema está pensado para controlar el origen del ACS en función de la temperatura del agua almacenada en el equipo térmico solar. Si el agua del termosifón no alcanza una temperatura preestablecida, el sistema comprobará si el agua acumulada en el termo-acumulador auxiliar está o no caliente, y en el caso que no este caliente procederá a encender la caldera y calentarla.

A continuación se describen las principales características del diseño funcional, las decisiones tomadas y el motivo por el cual se han tomado.

3.1 Microcontroladores, tipo de conexión y lenguaje de programación.

La primera decisión en cuanto al diseño del sistema fue elegir el micro-controlador a utilizar, para lo cual había que decidir ante antes el mejor tipo de conexión.

Inicialmente se planteó utilizar conexión WiFi, pero los problemas habituales en este tipo de conexión, como las bajas velocidades de conexión o la pérdida de la conexión en determinados momentos hizo que se descartara su uso.

Este hecho, unido al conocimiento del buen funcionamiento de los dispositivos PLC, hizo que la balanza por una conexión cableada mediante Ethernet, utilizando adaptadores PLC en aquellos sitios donde no puede llevarse un cable Ethernet. Una vez elegida esta conexión, la cantidad de micro-controladores se ve reducida a aquellos que poseen dicho interfaz de forma nativa.

La primera opción sería utilizar el LPC1769, ya utilizado anteriormente en la asignatura de Sistemas Empotrados y que incorpora una conexión Ethernet con interface RMII. Su principal ventaja frente a otros microcontroladores es la utilización de un sistema operativo en tiempo real, lo que implica que el sistema responderá a una acción dentro de un tiempo estrictamente definido.

Sin embargo, para este sistema de control del ACS solar no es necesario un tiempo de respuesta estricto, ya que por ejemplo, las electroválvulas tardan 30 segundos en abrirse, la caldera tarda casi un minuto en arrancar, y el incremento de un grado la temperatura en el termosifón puede requerir horas.

Por tanto, un sistema en tiempo real no es estrictamente necesario. Además, para utilizar el LPC1769 sería necesario desarrollar los drivers a bajo nivel de la interfaz Ethernet, lo que llevaría un tiempo que habría que quitar de algún otro apartado del proyecto.

Llegados a este punto, hubo que decidir que lenguaje de programación utilizar: un lenguaje de medio-bajo nivel como el C, o un lenguaje de alto nivel como Java.

El lenguaje C nos dará más velocidad, un menor consumo de memoria y un acceso completo al sistema. Sin embargo es más “delicado” de programar, ya que un error en la programación puede hace que el sistema se bloquee.

Java por contra es más lento que C, y consume más recursos, pero proporciona seguridad y robustez. Además, su recolector de basura nos evita tener que preocuparse excesivamente por el consumo de memoria.

Java utiliza el paradigma de POO, que proporciona abstracción, encapsulación, polimorfismo y herencia. Y además, es portable, por lo que una aplicación Java podría correr en dispositivos distintos (salvo en el caso de utilizar por ejemplo los puertos GPIO).

Por tanto, la elección es clara: Java. Ahora se necesita un sistema de computación que permita utilizar Java, y con soporte de los puertos de E/S en dicho lenguaje. En este caso había diferentes sistemas donde elegir, como BeagleBone, Orange Pi o la Raspberry Pi.

Aunque algunos de los sistemas en teoría compatibles con la Raspberry, como Orange Pi o Banana Pi poseen una mejor relación precio/potencia, tienen el inconveniente de una menor comunidad de soporte.

La BeagleBone tampoco posee una gran comunidad que la soporte. Eso unido al precio y al hecho de no poseer una librería que soporte los puertos de E/S bajo Java, hizo que se descartara.

Al final la solución elegida fue utilizar Raspberry Pi, por su capacidad de proceso y de memoria y por su soporte del lenguaje Java, características que permiten centrarnos en el desarrollo de la aplicación. Y en caso de dudas o problemas, es seguramente el miniordenador con mayor soporte en el mundo, gracias al elevado número de unidades vendidas, 11millones en noviembre de 2016 (Fuente: Raspberry Pi Foundation [26]).

3.2 Arquitectura de red

La siguiente decisión en cuanto al diseño del sistema fue elegir el tipo de arquitectura de red. Había dos posibilidades:

- **Arquitectura Peer-to-Peer** o P2P, en la que no hubiera clientes ni servidores, y por tanto, todos los nodos fueran iguales. La mayor ventaja, desde el punto de vista del diseño del proyecto, es que solo habría que desarrollar una única aplicación que correría en cada uno de los micro-ordenadores.

  Sin embargo, los algoritmos de las redes distribuidas son más complejos, y no tienen mucho sentido su utilización en una red tan pequeña como la que nos ocupa, con solo 4 nodos.

- **Arquitectura cliente-servidor**. En este caso, había que decidir entre dos posibilidades: utilizar un servidor y tres clientes, o tres servidores y un cliente. Normalmente el cliente es el equipo que realiza peticiones a algún servidor, y por tanto, el diseño más evidente es tener un cliente y tres servidores.

  Así, el cliente solicitará información a cada uno de los clientes, y en función de los datos, pedirá a cada servidor que realice una serie de acciones.

El diagrama de bloques del sistema sería el que se muestra en la siguiente figura 20:

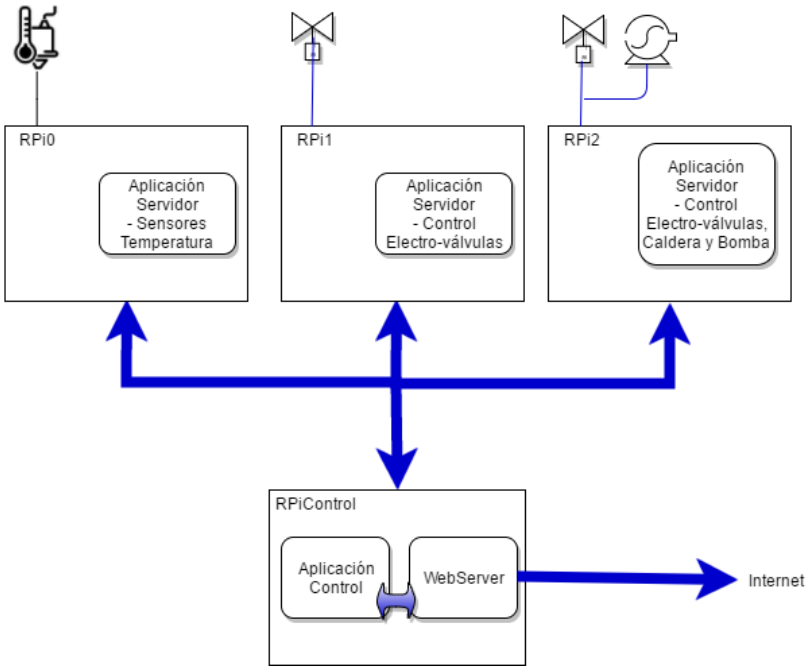


Figura 20: Diagrama de bloques del sistema.

La descripción de cada uno de los bloques sería la siguiente:

- ☑ **RpiControl**: bloque encargado de controlar el resto de bloques. Este bloque consta de una Raspberry Pi (RPi), la cual, conectada mediante Ethernet al resto de bloques, actuará como cliente y se encargará de pedir al resto de bloques información sobre el estado de los diferentes sensores y actuadores. En función de las respuestas obtenidas se encargará de enviará las ordenes necesarias.

  Además, posee un servidor Web que permite acceder desde Internet a la información del estado del sistema, así como su control.

- ☑ **RPi0**: bloque encargado de obtener la información de la temperatura del agua acumulada en el termosifón, y enviarla, bajo petición, al bloque de control. Opcionalmente, obtendría también información de la temperatura exterior.

  Como elementos físicos, este bloque está formado por una Raspberry Pi, un dispositivo ADC (debido a que las Raspberry no incluyen ninguno) y una o dos sondas de temperatura.

- ☑ **RPi1**: bloque encargado de activar o desactivar las electro-válvulas que permiten el paso del agua caliente proveniente del termosifón al circuito de ACS, así como de detectar su estado. Está formado por una RPi y el correspondiente circuito de control de las electro-válvulas.

☑ **RPi2**: este bloque es el encargado de controlar las electro-válvulas que controlan el uso de agua caliente proveniente del acumulador o del termosifón y su posición. También controla las electro-válvulas que desvían el agua caliente del circuito de calefacción para calentar el agua del termo.

También se encargara de detectar el estado de la caldera, así como el estado del termostato del termo-acumulador que indica que el agua acumulada ha alcanzado la temperatura deseada. Este bloque está formado por una RPi y los correspondientes circuitos de control de las electro-válvulas, y de detección del estado de la caldera y del termostato del termo-acumulador auxiliar.

3.3 Protocolo de comunicaciones

Una vez decidido la arquitectura de red, los dispositivos, y el lenguaje de programación que se iban a utilizar, la siguiente decisión fue elegir el protocolo de comunicaciones a utilizar.

☑ Sockets TCP: Java proporciona dentro de su API dos clases que permiten crear de una manera sencilla una conexión entre un cliente y un servidor. Estos sockets pueden crearse para utilizar tanto el protocolo de transporte TCP, como el UDP.

Inicialmente se crearon las aplicaciones cliente servidor utilizando sockets, consiguiendo con éxito el intercambio de mensajes entre el cliente y el servidor. Además, como el protocolo TCP está orientado a conexión, cuando se produce algún error en la transmisión, el destinatario lo detecta y solicita el reenvío de la información.

Sin embargo, como los sockets TCP son conexiones a bajo nivel, cada vez que se requiere enviar un mensaje hay que establecer la conexión entre los dos extremos del socket, gestionar los posibles errores de comunicación (como que uno de los extremos cierre o pierda la conexión), y lo que es más importante, hay que gestionar el protocolo de intercambio de los mensajes.

Esto generalmente se hace enviando un mensaje con un texto como “HELLO”. Si el servidor está disponible para aceptar peticiones, le contestara por ejemplo con un “OK”. Cuando el cliente recibe el “OK”, entonces enviará un mensaje con la petición y esperará a recibir el mensaje indicando que el servidor ha recibido la petición correctamente.

Es decir, hay que establecer todo el conjunto de reglas para garantizar un intercambio fiable de mensajes. Aunque esto no es difícil de implementar, la cantidad de mensajes intercambiados puede complicar la cantidad de combinaciones a detectar.

☑ HTTP: El protocolo de transferencia de hipertexto, es un protocolo de la capa de aplicación, que utiliza en el nivel de transporte al protocolo TCP. En el diseño de la aplicación, esto implica que el cliente solo enviará una petición al servidor, y que será el protocolo HTTP, utilizando el protocolo TCP, el encargado de enviar y controlar toda la petición. El servidor le responderá de la misma forma.

Esto evita tener que preocuparse de abrir el socket, enviar un mensaje para comprobar si el servidor está disponible, etc. Es el protocolo HTTP quien se encarga de toda esta gestión.

JAVA proporciona una serie de clases y métodos para utilizar el protocolo HTTP, por lo que finalmente este es el protocolo elegido para la transmisión y recepción de los mensajes.

El protocolo HTTP implementa un conjunto de métodos de petición , que indican al cliente cual es la acción a realizar sobre un recurso determinado. Los dos métodos más habituales para realizar peticiones en HTTP son GET y POST.

- ☑ GET: se utiliza para solicitar una información a un recurso determinado.
- ☑ POST: se utiliza para enviar información para ser procesada por un determinado recurso.

En la tabla 4 (fuente w3schools.com [7]) se muestran las principales diferencias entre Get y Post .

	GET	**POST**
BACK button/Reload	Harmless	Data will be re-submitted (the browser should alert the user that the data are about to be re-submitted)
Bookmarked	Can be bookmarked	Cannot be bookmarked
Cached	Can be cached	Not cached
Encoding type	application/x-www-form-urlencoded	application/x-www-form-urlencoded or multipart/form-data. Use multipart encoding for binary data
History	Parameters remain in browser history	Parameters are not saved in browser history
Restrictions on data length	Yes, when sending data, the GET method adds the data to the URL; and the length of a URL is limited (maximum URL length is 2048 characters)	No restrictions
Restrictions on data type	Only ASCII characters allowed	No restrictions. Binary data is also allowed
Security	GET is less secure compared to POST because data sent is part of the URL Never use GET when sending passwords or other sensitive information!	POST is a little safer than GET because the parameters are not stored in browser history or in web server logs
Visibility	Data is visible to everyone in the URL	Data is not displayed in the URL

Tabla 4: HTTP Methods: GET vs. POST
(fuente: https://www.w3schools.com/tags/ref_httpmethods.asp)

Como se puede ver, el protocolo HTTP indica que la forma habitual de funcionamiento es utilizar GET para solicitar información a un recurso, y POST para enviar información al servidor.

Tal y como está diseñado el sistema, habría que utilizar GET para enviar las ordenes a los servidores, y por tanto, se podría haber utilizado únicamente dicho método para enviar todas las peticiones a los servidores. Sin embargo, el método GET posee algunas limitaciones como el tamaño en la URL, o el hecho de que solo admita caracteres ASCII.

En consecuencia, y en vista de que el servidor implementado no es estrictamente un servidor Web, se decidió utilizar ambos métodos, de tal forma que el método empleado para enviar un petición, para sirve para indicar el tipo de órdenes.

Así, se ha utilizado el método GET para enviar ordenes que no requieren un recurso, si no que le indican al servidor indique si está activo, que cierre la aplicación o que apague o reinicie el sistema. Este tipo de órdenes solo se envían porque se han solicitado manualmente desde la aplicación Web. El resto de órdenes se envían mediante el método POST.

3.4 PiCACSSO

En la siguiente figura se muestra el diagrama de bloques funcional del sistema Pi Control de Agua Caliente Sanitaria Solar:

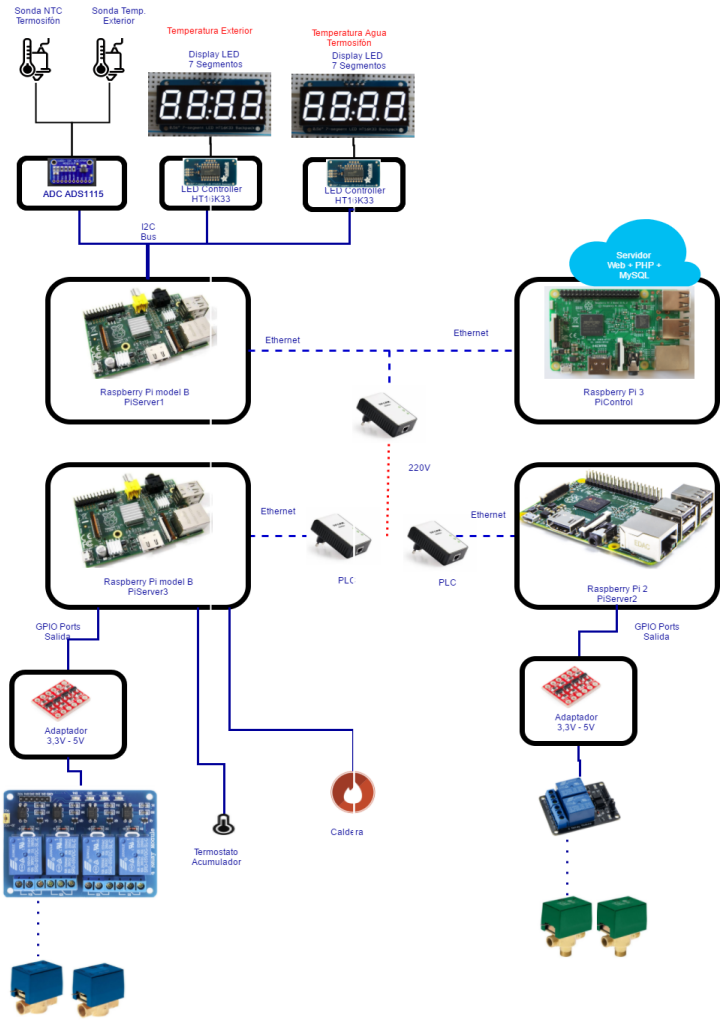


Figura 21: Diagrama de bloques funcional del sistema.

Como se puede observar en el diagrama de bloques funcional del sistema tenemos 4 Raspberry Pi. Todas se encuentran en la misma red local conectadas mediante cable Ethernet, utilizando para ellos tres dispositivos PLC, dos conectados a la Raspberry y el tercero al router de fibra óptica. Todas se han configurado con dirección IP fija.

En la siguiente figura 22 se muestra un diagrama de bloques del funcionamiento del sistema y el intercambio de mensajes, una versión reducida de un diagrama de flujo.

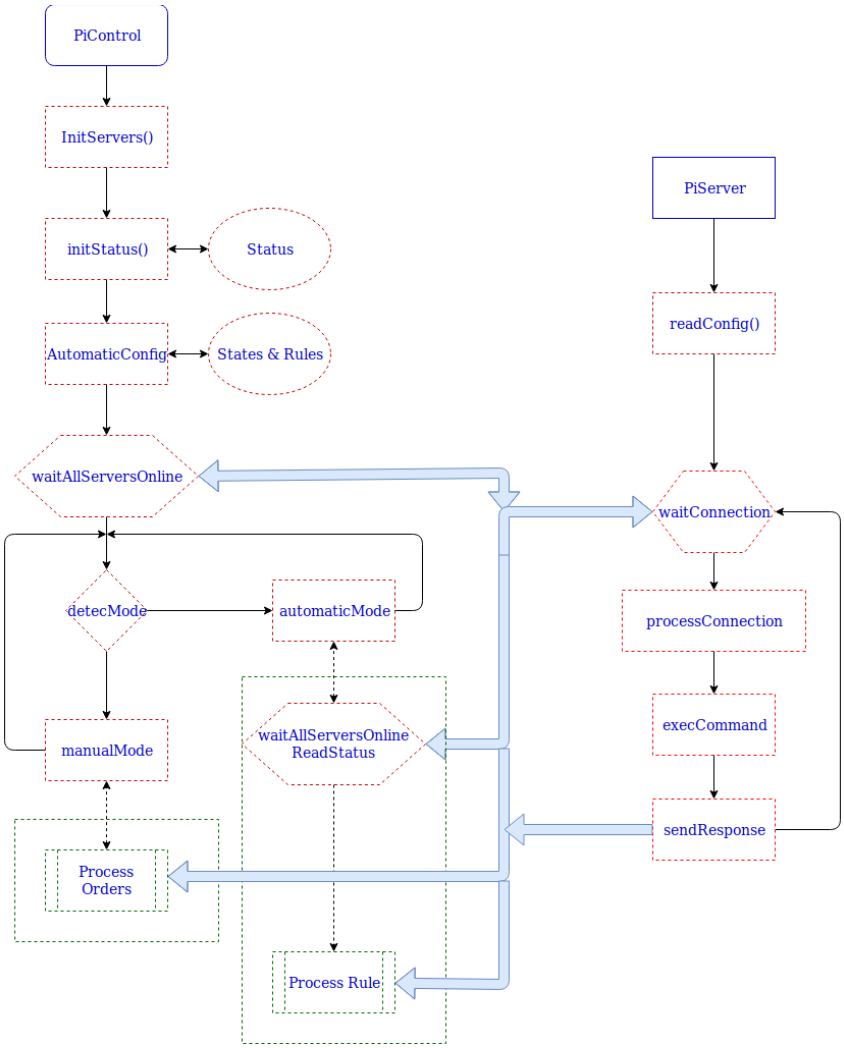


Figura 22: Diagrama de flujo básico.

A grandes rasgos, el cliente se encarga de controlar todo el sistema. Primero espera a que todos los servidores estén en línea. Luego detecta el modo de funcionamiento, comprobando si existen órdenes manuales que el usuario haya indicado mediante la aplicación Web PiSolar. A continuación determina el estado del sistema, en base a la información solicitada a los servidores, y cual sería el estado óptimo, enviando las órdenes necesarias a cada servidor, para alcanzar dicho estado óptimo.

Los servidores están siempre a la espera de recibir solicitudes por parte del cliente y de responderlas, bien devolviendo alguna información acerca de alguno o de todos sus dispositivos o bien ejecutando algún comando en alguno de los dispositivos.

Previsto como una característica extra, se ha implementado el modo de funcionamiento autónomo en los servidores, de tal forma que si detectan que ha transcurrido demasiado tiempo desde la última comunicación por parte del servidor, pasan a funcionar en este modo.

3.5 PiControl

La aplicación piControl funciona en la Raspberry cliente, y es la encargada de toda la gestión del sistema, para lo cual, cada cierto tiempo establecido manualmente, se encarga de realizar las siguientes funciones:

- ✗ Comprobar si existe alguna orden manual, y en caso afirmativo procesarla.
- ✗ Si no existe ninguna orden manual se encarga de:
  - ➢ Solicitar información a cada uno de los clientes.
  - ➢ Determinar el estado deseado del sistema en función de la información enviada por cada servidor.
  - ➢ Enviar las ordenes necesarias a los servidores para pasar al estado deseado.
- ✗ Almacenar el estado actual en la base de datos SQL que se encuentra instalada en el mismo cliente..
- ✗ Obtener el promedio de las temperaturas y almacenarlo en la base de datos.

3.6 PiServer

La aplicación piServer funciona en cada una de las Raspberry servidor. La aplicación es la misma para todos los servidores, y la configuración específica de cada uno de los servidores se encuentra almacenada en un fichero de texto, que la aplicación lee al iniciarse. La aplicación servidor se encarga de:

- ☑ Detectar si es la Raspberry que posee las sondas de temperatura. En este caso ejecutará cada cierto tiempo (indicado en el fichero de configuración) una tarea que leerás las temperaturas de las sondas NTC, y actualizará los valores correspondientes en los display LED.
- ☑ Esperar a recibir alguna petición por parte del cliente, y procesarla, leyendo la información solicitada de algún dispositivo o activando/desactivando alguno de los actuadores conectados a esa Raspberry.
- ☑ Comprobar si hay que ejecutar en modo autónomo, para lo cual ejecuta cada cierto tiempo preestablecido una tarea que determina si hay algún problema con el cliente. Para ello comprobará la fecha de la última petición por parte del cliente, y en el caso de que supere el tiempo establecido en el fichero de configuración, pasará a ejecutar las tareas previstas en el modo autónomo.

3.7 Web PiSolar

Esta aplicación web ha sido realizada en lenguaje PHP. Su diseño se ha realizado lo más sencilla posible en vista de los usuarios habituales de dicha aplicación. Además, se ha diseñado teniendo en mente que se usará la mayoría de las veces desde el teléfono móvil.

A continuación se muestran algunas capturas de dicha aplicación. Las dos primeras se han capturado en un teléfono móvil, y las siguientes en el navegador del PC.

Tal y como se observa en la figura 23, en la página principal se muestran las temperaturas (agua acumulada en el termosifón y temperatura exterior) y las acciones que se pueden realizar.

Figura 23: Página principal.

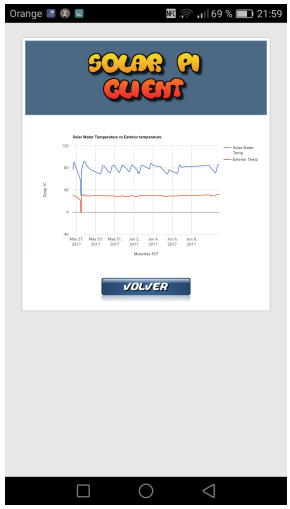


Figura 24: Gráfico de temperaturas en el móvil.

En las figuras 24 y 25 se muestra como se visualiza el gráfico con la evolución de las temperaturas en el navegador del teléfono móvil y del PC.

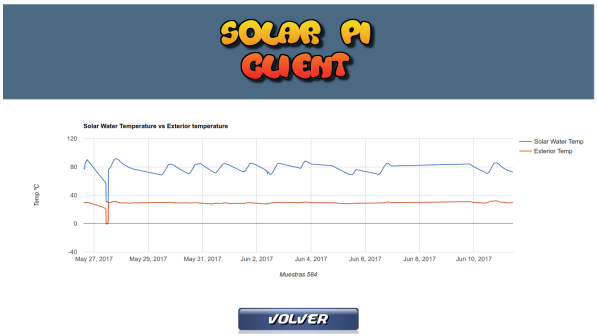


Figura 25: Gráfico de temperaturas en navegador..

Figura 26: Estado del sistema.

En figura 26 se muestra el estado actual del sistema, y como se observa, desde él se pueden activar o desactivar manualmente todos los actuadores del sistema.

4 DESCRIPCIÓN DETALLADA

A continuación se detalla de forma técnica las características del sistema, tanto en el apartado de hardware como en el de software.

4.1 Hardware

En este apartado se detallan las características del montaje físico realizado en cada uno de los componentes del Sistema PiCACSSO.

[1] PiControl

En el cliente no se conectado ningún dispositivo, salvo un disco duro SSD de 32GB. Se ha modificado el arranque (*boot*) para cargar el sistema operativo Raspbian desde el disco duro SSD.

[2] PiServer1

Este Raspberry, tal y como se puede observar en la figura 27, tiene conectados los siguientes tres dispositivos mediante el bus I2C:

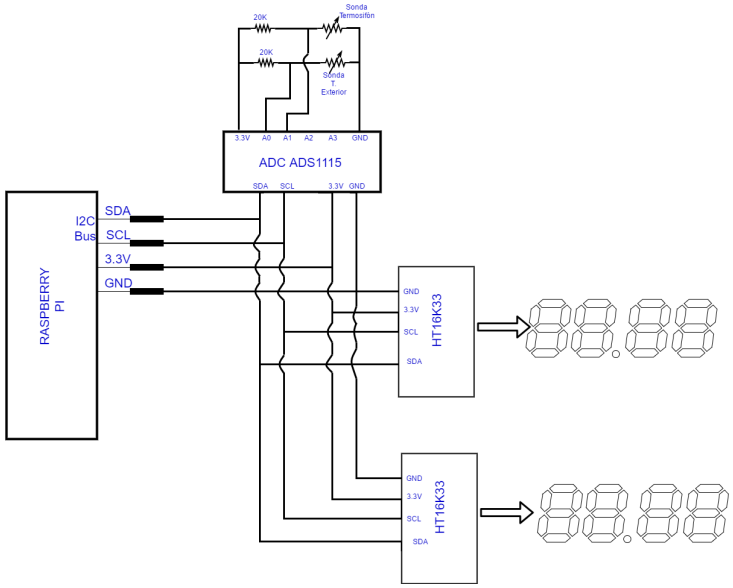


Figura 27: Conexiones PiServer1.

Para poder utilizar dispositivos I2C en la Raspberry, antes hay que habilitar los módulos del kernel que permiten acceder a él. También es recomendable instalar las aplicaciones `i2c-tools`, ya que permiten detectar los dispositivos conectados al bus I2C.

En la figura 28 se muestra la salida del comando `i2cdetect`, que muestra las direcciones de los dispositivos I2C conectados a la Raspberry PiServer1.

```
pi@RPi1:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: 70 71 -- -- -- -- -- --
pi@RPi1:~ $
```

Figura 28: Detectando las direcciones de los dispositivos I2C

☑ ADC. El conversor analógico-digital es un ADS1115 (anexo 10.1), que se conecta a la Raspberry mediante el bus I2C. A este ADC se han conectado las dos sondas de temperatura (termistores NTC). Para calcular el valor de la resistencia de la sonda se ha utilizado en las dos sondas un divisor de tensión como el de la figura 29.

El ADC nos devuelve el valor digital correspondiente a la tensión Vo. Como conocemos Vi (3.3V) y R (he utilizado una resistencia de 20KΩ). Aplicando la ley de Ohm, obtenemos el valor de la resistencia de la sonda de temperatura (el desarrollo completo está en el anexo 10.5):

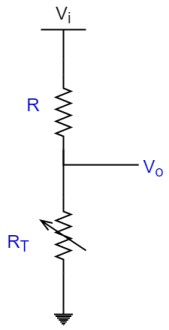


Figura 29: Divisor de tensión.

$$R_T = \frac{(R_1 \cdot V_o)}{(V_i - V_o)}$$

La única información disponible de la sonda de temperatura son unas hojas fotocopiadas y escaneadas en pdf. En ellas se encuentran las especificaciones (figura 30) del termistor y la tabla de correspondencias

3) RATING

3-1) Rated zero-power resistance. R_{25} : 10 kΩ ± 1 % (at 25 ℃)

3-2) B value. $B_{25/85}$: 3,435 K ± 1 %

* The B value is calculated using the zero-power resistance values measured at 25℃ and 85℃.

3-3) Dissipation factor.	: Approx. 3	mW/℃	(in air)
3-4) Thermal time constant.	: Approx. 75	s	(in air)
3-5) Maximum power rating.	: 15	mW	(at 25℃)
3-6) Category temperature range (= Operating temperature range)	: −50 ~ 105	℃	

Figura 30: Valores característicos de la sonda del termosifón.

entre los valores máximo, mínimo y promedio de la resistencia y la temperatura, tal y como se muestra en la figura 4.

Había dos posibilidades a la hora de obtener la temperatura. Trasladar todos estos datos a una tabla para poder relacionar la resistencia con la temperatura, o utilizar los valores característicos indicados en las especificaciones.

Por rapidez y comodidad se decidió utilizar los valores característicos. Así, aplicando la **ecuación B simplificada de Steinhart-Hart** se pueden obtener valores aproximados de la temperatura a partir de los valores característicos:

$$\frac{1}{T}=\frac{1}{T_0}+\frac{1}{B}\cdot\ln\left(\frac{R}{R_0}\right)$$

Comprobando los valores obtenidos con los obtenidos por el termómetro digital que inicialmente poseía la instalación, la diferencia de temperatura es de 1 a 2ºC en 70ºC, lo que da un margen de error del 1.1%, no muy diferente del 1% especificado en las característica de la sonda.

Todo este cálculo se ha utilizado igualmente, aplicando los valores característicos indicados por el vendedor, para obtener la temperatura de la sonda exterior.

☑ Display LED 4 dígitos de 7 segmentos. Estos dos display se conectan también a la Raspberry mediante el bus I2C. Vienen por defecto configurados en la dirección 0x70, pero como en el bus no puede haber dos dispositivos con la misma dirección, hay que modificar uno de ellos. Como se observa en la figura 31, traen 3 jumpers marcados A0 A1 A2, para unir mediante un punto de soldadura, de tal forma que la dirección se modifica al valor base, 0x70, más el valor en binario de los jumpers: A2 A1 A0.

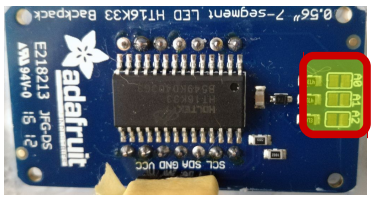


Figura 31: Controladora (Backpack) HT16K33.

Para poder utilizar los displays LED, se ha utilizado una librería publicada en el foro el fabricante de la placa [21] por Eric Eliason [22]. Se ha modificado adaptándola a las necesidades de la aplicación, y al modelo de display LED de 7 segmentos utilizado.

[3] PiServer2

Esta Raspberry se encarga de controlar las dos electroválvulas que controlan el paso del agua caliente del termosifón al circuito del ACS.

Tal como se muestra en la figura 32, solamente se han conectado dos de los GPIO de la Raspberry a un *level shifter* (anexo 10.2), conectado a su vez a los puertos de activación del módulo de relés opto-aislado.

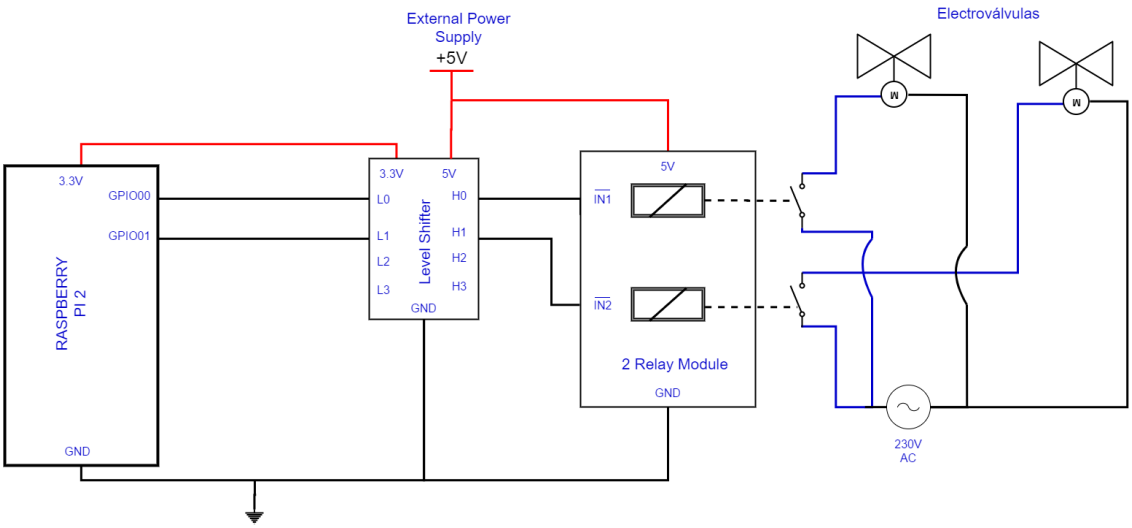


Figura 32: Conexiones PiServer2.

Se ha conectado el neutro de la corriente eléctrica de 220V al común del relé, y desde el otro borne del relé a la electroválvula. La fase se ha conectado al otro borne de la electroválvula. De esta forma, al cerrarse el relé mediante una orden de la Raspberry, la electroválvula comienza a funcionar.

En el anexo 10.3 se comenta con más detalle el funcionamiento de los relés y la conexión utilizada.

[4] PiServer3

Esta Raspberry se encarga de controlar cuatro electroválvulas y una bomba:

☑ Dos que controlan la entrada y salida de agua del acumulador auxiliar al circuito de ACS.

☑ Dos que controlan el paso del agua caliente del circuito de la calefacción al circuito de calentamiento del acumulador auxiliar y la bomba que mueve el agua de la calefacción.

Para ello se han utilizado dos relés de un módulo de relés de 4. El diseño, para esta parte, es idéntico al del servidor PiServer2, por lo que no se muestra. La única diferencia son los puertos GPIO utilizados (GPIO03 y GPIO04),y que en lugar de activar o desactivar una única electroválvula, uno de los relés activa o desactiva dos electroválvulas y el otro dos electroválvulas y la bomba de agua de la caldera.

Esta Raspberry se encarga también de detectar el estado de la caldera, del termostato del acumulador auxiliar, y de controlar de encendido de la caldera.

En la figura 34 se muestra el esquema eléctrico de la caldera de gasóleo obtenido del manual de instalación de la caldera. En dicha figura se ha resaltado la parte donde se encuentra el interruptor de encendido, y el piloto de puesta en marcha que indica que la caldera se encuentra o no encendida.

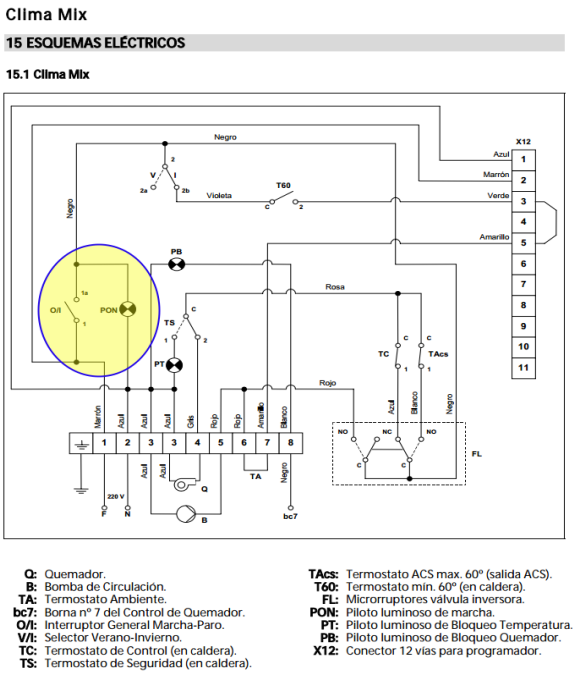


Figura 34: Esquema eléctrico de la caldera.

Como se muestra en la siguiente figura 33, el mecanismo utilizado para encender la caldera a distancia ha sido puenteado el interruptor de encendido, y conectado dicho puente a uno de los relés de la placa de relés que controla la Raspberry. De esta forma, la Raspberry, activando el relé puede encender y apagar la caldera. Si la caldera ya estaba encendida mediante el interruptor manual, la activación/desactivación desde la Raspberry no tiene ningún efecto en el estado de la caldera.

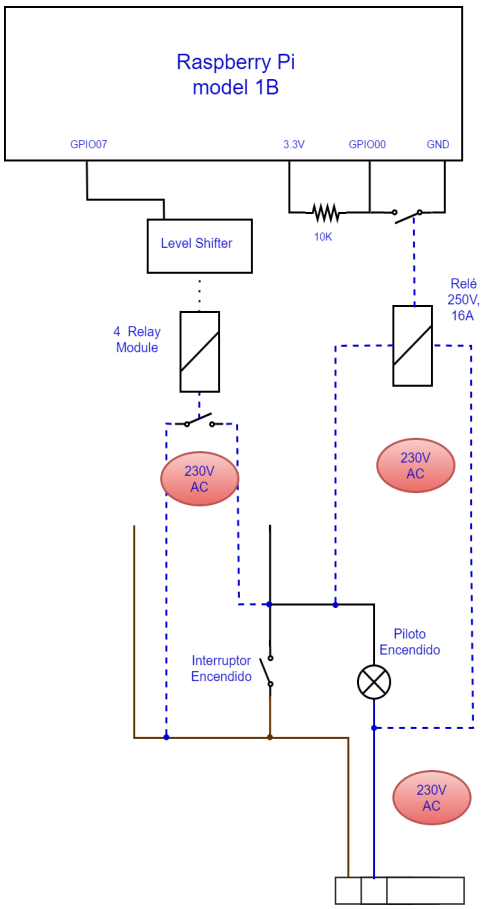


Figura 33: Modificaciones al circuito eléctrico de la caldera.

Para detectar el estado de la caldera, se ha realizado un puente desde el piloto de encendido hacia un relé electromecánico de 220V.

Cuando la caldera está encendida, el piloto se encuentra iluminado, el relé se activa y cierra el interruptor, cuyo estado es detectado desde la Raspberry.

En este caso, se ha colocado una resistencia de *pull-up* de 10KΩ, ya que los primeros modelos de Raspberry no incluían resistencias de *pull-up* en algunos de los puertos GPIO, y la Raspberry utilizada es un modelo antiguo.

En cuanto a la detección del estado del termostato del acumulador auxiliar, en la figura

36 se muestra el esquema eléctrico de dicho acumulador auxiliar, un Fagor AFE150N1. En dicha figura se observa, marcada con un círculo, una ficha de conexiones al lado del termostato de funcionamiento. Aunque no se especifica su uso en el manual de instalación, en las características de la gama de interacumuladores se indica que posee un "interruptor de conexión bomba circulación sanitaria" [40].

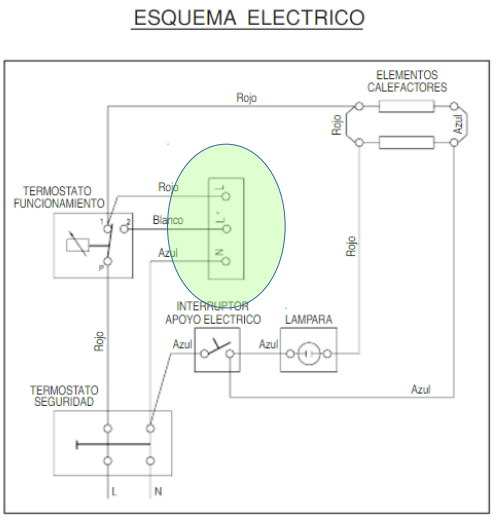


Figura 36: Esquema eléctrico del Interacumulador Fagor AFE-150N1.

Por lo tanto, para detectar el estado del termostato se ha realizado el mismo sistema que para detectar el estado de funcionamiento de la caldera, es decir, conectar la fase y el neutro del conector a un relé electromecánico, de tal forma que la Raspberry, pueda detectar su estado, tal y como se muestra en la figura 35

Tal y como se indica en el apartado de software, todas las conexiones a los puertos GPIO de la Raspberry se encuentran especificadas en un fichero de configuración, el cual es leído por cada servidor al iniciar.

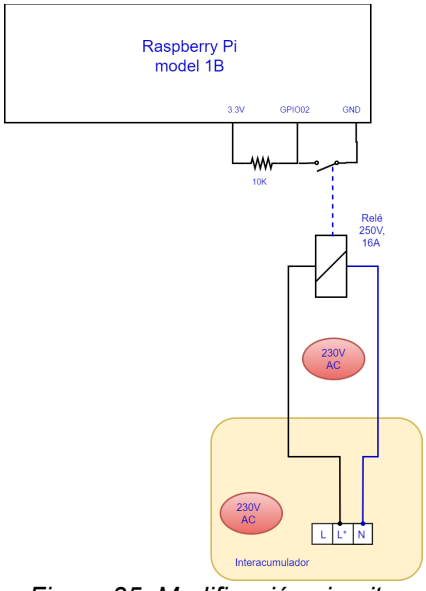


Figura 35: Modificación circuito electrico del termo acumulador.

4.2 Software

[1] Comunes

En este primer apartado de este capítulo, se indican las características comunes a las dos aplicaciones realizadas en Java.

☑ Log4J: se ha utilizado la librería Apache Log4J [12] para generar el log de las aplicaciones. Se ha configurado para que únicamente almacene en un fichero los niveles de log "*Warning*", "Severe" y "Fatal".

También se ha configurado para que genere un log diario, y que todos los días los comprima. Además se ha indicado que no almacene los mensajes inmediatamente, si no que espere a rellenar la caché. De esta forma se reduce la degradación del las tarjetas de memoria SD, ya que se escriben los sectores cuando están llenos de información.

☑ Siempre se ha evitado los posibles errores y excepciones "envolviendo" el código problemático en un bloque *try-catch*. Si se produce una excepción, se detecta si el sistema puede o no continuar, y en función de ello, se genera el mensaje para el log, y se continua o se sale de la aplicación.

[2] PiControl

A continuación se describe de forma detallada el funcionamiento del cliente, la implementación del autómata finito determinista que implementa el modo de funcionamiento automático y las clases implementadas junto con el diagrama UML:

Funcionamiento:

El cliente se encarga de realizar las siguientes funciones:

☑ Leer la configuración de tres ficheros de texto, utilizando la clase de la API de Java `java.util.Properties`. Los tres ficheros son los siguientes:

- ✗ `client.config`: en este fichero se encuentra información como el número de servidores del sistema, cuantas veces se intenta enviar un mensaje mediante http antes de contabilizarlo como error, si se envían correos electrónicos de aviso, cuantos errores se cuentan antes de enviar el correo electrónico, con cuantos valores se calcula el promedio de la temperatura, los tiempos de espera y la temperatura mínima óptima, por encima de la cual tiene que estar el agua almacenada en el termosifón para poderse utilizar.
- ✗ `automatic.config`: en este fichero se encuentra la información relativa al autómata finito determinista que implementa el modo automático de funcionamiento.
- ✗ `mail.config`: en este fichero se encuentra la información relativa al usuario y contraseña desde la cual el sistema enviará los mensajes de correo electrónico.

☑ Antes de comenzar con el proceso, el cliente comprueba si los tres servidores se encuentran en línea. Si alguno no está en línea, envía un correo electrónico y sigue a la espera un tiempo antes de volver a comprobar si se encuentran en linea.

Cuando se alcance el número de veces especificado en la configuración sin haber conectado, vuelve a enviar el correo electrónico. Se estima que en caso de error envíe un correo electrónico de aviso cada 8 horas aproximadamente.

El cliente no puede comenzar a controlar el sistema hasta que todos los servidores estén en línea. Entonces repite indefinidamente (o hasta que una orden manual lo cierre) las siguientes tareas:

- Comprueba si existe alguna orden manual. Para ello inicialmente se comprobaba si existía un fichero de configuración con las órdenes manuales. Tras la implementación de la Web PiSolar, esta almacena las ordenes manuales en la tabla “manual” de la base de datos SolarPi.

  Existen dos tipos de órdenes manuales:

  - Específica: afecta a un único dispositivo de un servidor. El cliente lee la orden, determina a que dispositivo va destinada y se la envía. Al ser una orden manual no modifica su estado, y seguirá ejecutándola hasta que manualmente se desactive.
  - Global: afecta a todos los servidores. El cliente primero modifica el estado de la orden para marcarla como procesada, modificando un campo en la base de datos. Tras enviarla a todos los servidores, la ejecuta el mismo.

- Si no existe ninguna orden manual, realiza el proceso automático, consistente en:

  - Comprobar si todos los servidores se encuentran en línea y obtener su estado.
  - Determinar el estado ideal y enviar las ordenes necesarias a cada uno de los servidores para que el sistema cambie al nuevo estado. Para ello se ha definido un autómata finito determinista, que se comenta a continuación.

Autómata Finito Determinista

El autómata finito determinista o DFA implementado en el cliente define todos los posibles estados en que puede estar el sistema, y que cambios de estado se producen en función de las posibles entradas al sistema: la temperatura del agua acumulada en el termosifón, y el estado del termostato del acumulador auxiliar.

El autómata, que posee 3 estados, es el que se muestra en la figura 37:

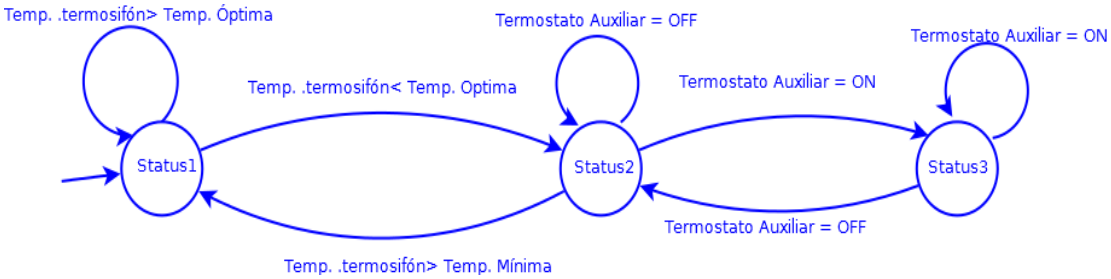


Figura 37: Automata Finito Determinista que implementa el modo Automático.

El sistema comienza en el estado “Status1”, y permanece en él mientras la temperatura del agua acumulada en el termosifón sea superior a la temperatura óptima. Si la temperatura es menor o igual que la óptima se pasa al estado “Status2”, donde permanece hasta que o bien salte el termostato del acumulador auxiliar o el agua acumulada en el termosifón supere la temperatura óptima.

En el estado “Status3” es en el que se calienta el agua del acumulador auxiliar mediante la caldera, estado en el que permanece mientras el termostato del acumulador auxiliar este activo.

Para implementar el autómata se ha utilizado un fichero de configuración en el que se definen una serie de reglas, que son de dos tipos:

- ☑ Reglas C (Cambio): se utilizan para cambiar de estado.
- ☑ Reglas P (Permanencia): estas reglas se aplican sin cambiar de estado.

Cada regla consta de varias partes:

- ☑ Nombre de la regla, e indica el estado al que pertenece la regla y el número de esta.
- ☑ Tipo de regla: C o P.
- ☑ Nombre del servidor al que afecta la regla.
- ☑ Tipo de dispositivo al que afecta (I en caso de entrada u O en caso de salida).
- ☑ Nombre del dispositivo.
- ☑ Condición ó estado deseado.
  - ✗ En el caso de una regla de cambio en este campo se almacena la condición que se tiene que cumplir para cambiar de estado.
  - ✗ En el caso de una regla de permanencia, en este campo se almacena el estado (encendido, apagado o indiferente) que se desea en el dispositivo.
- ☑ Estado al que se cambia, si se es una regla de tipo C y la condición se cumple.
- ☑ Comentario.

En la siguiente tabla 5 se muestran dos de las reglas utilizadas en la configuración del autómata, y su significado:

`St3Rule1=C,RPiServer1,I,SensorT,TMAX,Status1, Si temperatura termosifon >TMAX =>Status1`
Esta regla, la 1 del estado 3, indica que es una regla de cambio, y por tanto, si el dispositivo de entrada de nombre SensorT tiene un valor mayor que TMAX (temperatura optima), cambiamos al estado 3
`St3Rule6=P,RPiServer3,O,Actuador1,ON,Status3, Encender la caldera`
Esta regla es la número 6 del estado 3, e indica que es una regla de permanencia. Por tanto, el dispositivo de salida con nombre Actuador1 tiene que estar en estado encendido (ON), y se permanece en el mismo estado. Esta regla enciende o mantiene encendida la caldera.

Tabla 5: Ejemplo de reglas de configuración del autómata.

Clases

En la figura 38 se muestra parte del diagrama de bloques UML (creado mediante la aplicación web yUML [46]), solo con las clases más importantes, sin los atributos y los métodos. El código UML completo se encuentra en el anexo 10.6.

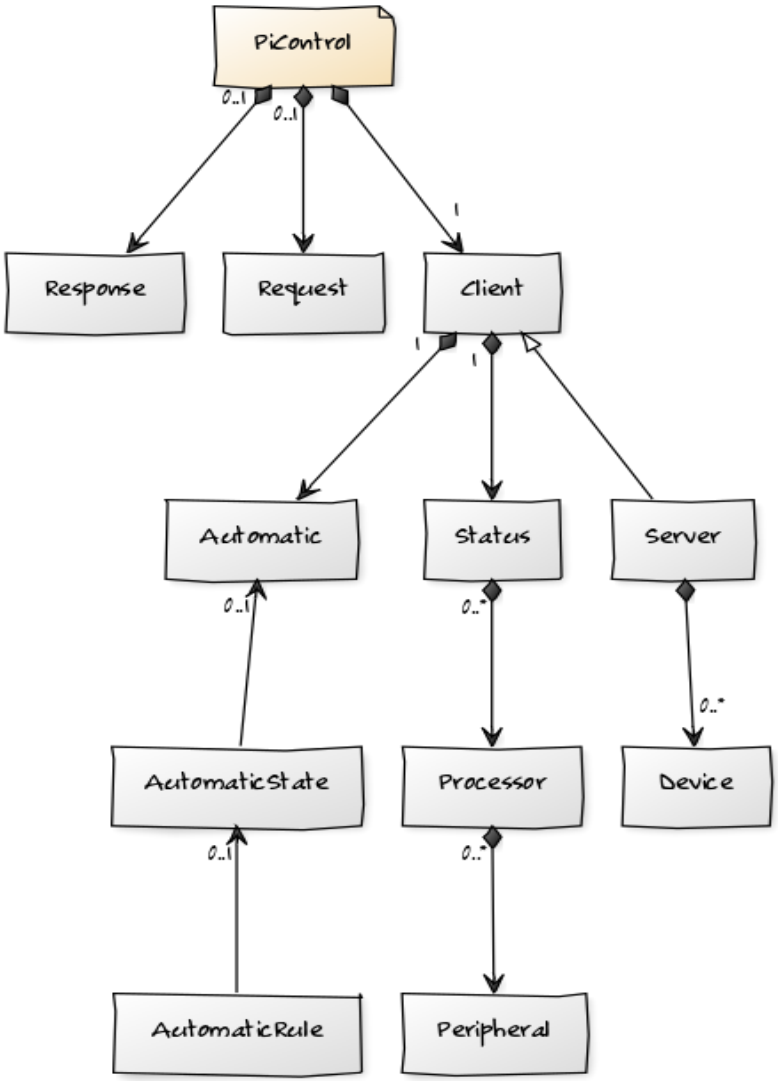


Figura 38: PiClient: UML reducido.
(Creado con: https://yuml.me/diagram)

A continuación se detallan estas clases más importantes. El resto de clases no representadas en la figura 38, son clases que únicamente implementan diferentes métodos, estando todos los métodos de un tipo, en una misma clase. Así mismo, se ha creado una clase con todas las constantes utilizadas en la aplicación.

Las clases son las siguientes:

- ☑ **PiControl**: Esta es la clase principal, que se encarga de utilizar el resto de métodos y clases.
- ☑ **Request**, **Response**: Estas dos clases implementan los objetos que contienen la petición que el cliente envía a los servidores, y la respuesta que el servidor envía al cliente. Estas dos clases son enviadas a través del protocolo HTTP entre cliente y servidor, de forma serializada en formato JSON mediante los objetos y métodos proporcionados por la librería GSON. Estas dos clases se utilizan exactamente igual en la aplicación servidor.

- ☑ **Server**: clase que representa la información lógica de una Raspberry: nombre del servidor, puerto y dirección IP, fecha de última conexión. También contiene una lista de objetos Device, con la información de los dispositivos conectados.
- ☑ **Client**: Esta clase amplía (*extends*) la clase server para representar a la Raspberry que funciona como cliente. En el cliente se almacenan todas las configuraciones leídas del fichero de configuración, así como la lista de servidores, y los objetos que representan el estado del sistema (Status) y el autómata finito (Automatic).
- ☑ **Device**: clase que implementa todos los valores lógicos necesarios para configurar y definir un dispositivo físico.
- ☑ **Status**: es la clase en la que se almacena el estado global del sistema. Está formada por una lista de objetos Processor (parte lógica de un servidor) cada uno de los cuales representa la parte lógica de un servidor dentro del estado.
- ☑ **Processor:** implementa la información lógica necesaria para el estado del sistema (nombre, dirección IP, estado) de cada uno de los servidores presentes en la red, y cada objeto Processor implementa una lista de objetos Peripheral.
- ☑ **Peripheral**: esta clase implementa la información lógica de cada dispositivo conectado a un servidor. Esta información lógica es la que necesita el estado del sistema: nombre del dispositivo, nombre lógico del puerto en el que está conectado, el modo de funcionamiento (entrada, salida u otros como I2C), y el valor que tiene (encendido, apagado, temperatura,…).
- ☑ **Automatic**: clase que implementa el autómata finito determinista. Almacena todos los diferentes estados en que puede estar el sistema, así como el estado actual. Consta de una lista de AutomaticState, que son los estados definidos en el DFA.
- ☑ **AutomaticState**: esta clase implementa un estado del DFA, y está formado por el conjunto de reglas que se aplican en este estado.
- ☑ **AutomaticRule**: clase que implementa una regla de un determinado estado. Los atributos de esta clase son todos los campos que se encuentran en el fichero de configuración `automatic.config`, y que se cargan leyendo dicho fichero al iniciar el cliente.

Además de estas clases se han definido las siguientes en el cliente:

- ☑ **Constants**: clase que implementa todas las constantes del sistema, como nombre de los sensores, nombre de los ficheros de configuración, tipos de errores, nombre de la base de datos y de las tablas, cadenas con las consultas SQL,…
- ☑ **Communication**: clase que implementa todos los métodos de comunicaciones.
- ☑ **ManualOrder**: cuando existen ordenes manuales en la base de datos, el cliente las almacena en objetos de esta clase, para luego procesarlas y ejecutarlas.

- ☑ **Processing**: clase que implementa los métodos mas importantes, como los encargados de determinar el siguiente estado del sistema, leer los ficheros de configuración o ejecutar las ordenes manuales.
- ☑ **SQLUtils**: clase con los métodos de consultas a la base de datos.
- ☑ **SendMail**: esta clase implementa los métodos para enviar los correos electrónicos en caso de error.
- ☑ **Utilities**: clase con algunos métodos útiles, como obtener la red y la dirección de una dirección IP (suponiendo redes de tipo /24).

[3] PiServer

A continuación se describe de forma detallada el funcionamiento del servidor, como se ha realizado la implementación del modo de funcionamiento autónomo, las tareas que se ejecutan, y las clases implementadas junto con el diagrama UML:

Funcionamiento

La aplicación PiServer es independiente del servidor en el que funciona. La configuración de cada uno de los servidores se encuentra en un fichero de configuración.

La aplicación piServer se encarga de realizar las siguientes funciones:

- ☑ Leer la configuración de dos ficheros de texto, para lo cual se utiliza la clase de la API de Java. `java.util.Properties`.
  - ✗ `rpi.config`: en este fichero se encuentra la información relativa al servidor, como el nombre, la dirección Ip, y el número de dispositivos que tiene conectados. Para cada dispositivo hay una línea con la siguiente información separada por comas:
    - ➢ Número de dispositivo. Solo identifica al dispositivo para poder leer la configuración.
    - ➢ Nombre que le damos al dispositivo. Deben de coincidir con los nombres dados en la clase donde se especifican las constantes.
    - ➢ Pin mode: Indica si el dispositivo es de entrada, salida o es un dispositivo I2C, es decir va a indicar si el pin va a funcionar como GPIO, bien entrada o salida, o
    - ➢ Si es un dispositivo que utiliza un GPIO, este campo contiene el nombre lógico usado en la librería Pi4J, mientras que si es un dispositivo I2C, contiene su nombre.
    - ➢ Si es un dispositivo que utiliza un GPIO, este campo contiene el nombre lógico, mientras que si es un dispositivo I2C, contiene contiene el valor de los registros de configuración, o el valor DEF (default) para utilizar la configuración por defecto.
    - ➢ Si es un dispositivo que utiliza un GPIO, este campo indica el nivel lógico de la señal, es decir si se activa a LOW (0V) o a HIGH (+Vcc). Si es un dispositivo I2C, el canal utilizado del dispositivo, o UNK si es desconocido o no utilizado.

- Si es un dispositivo que utiliza un GPIO, este campo indica el valor deseado (encendido o apagado) por defecto al iniciar el sistema. Si es un dispositivo I2C, el nombre de la conversión a utilizar (solo se ha implementado la NTC, para sondas NTC) o UNK si es desconocido o no utilizado.
- Comentario sobre el dispositivo.

Además, al implementar el modo autónomo se han añadido algunos valores constantes de configuración del servidor, como cada cuanto tiempo se actualizan los displays LED o cada cuanto tiempo se ejecuta el proceso que detecta si el cliente está caído y pasa a ejecutar el modo autónomo.

✗ `rules.config`: en este fichero se encuentra la información relativa a las reglas a utilizar en caso de funcionar en modo autónomo.

☑ Una vez leídos los ficheros de configuración, el servidor comprueba si es el que tiene conectado los displays de temperatura. Y en caso afirmativo lanza la tarea que se encargará de actualizar cada cierto tiempo, especificado en el fichero de configuración, los datos de las temperaturas en los displays LED de 7 segmentos.

☑ Cada servidor lanza la tarea que cada cierto tiempo comprueba si el cliente está funcionando.

☑ Luego pasa a quedar a la espera de recibir una petición por parte del cliente, donde se queda indefinidamente, o hasta que el cliente le envíe una orden manual de cierre.

☑ Cuando recibe una petición, determina el tipo de petición según el método utilizado para solicitarla:

✗ `GET`: este método indica que la orden es o bien una petición para determinar si el servidor está disponible, o es una orden de cierre de la aplicación, o de cierre de la aplicación y apagado o reinicio de la Raspberry. La orden se envía en la URL.

✗ `POST`: este método indica que la petición se envía mediante un objeto `Request` serializado en formato GSON/JSON. Los comandos implementados en este método son los siguientes:

- `CONFIG`: el cliente solicita la configuración inicial del servidor.
- `STATUS`: el cliente solicita el estado actual del servidor.
- `READ`: el cliente solicita que el servidor lea el estado de alguno de sus dispositivos.
- `ACTIVATE/DEACTIVATE`: el cliente solicita al servidor que active o desactive (encienda/apague) algún dispositivo.

  El cliente no tiene que preocuparse de si el dispositivo se activa con señales a LOW o a HIGH. El servidor, mediante el fichero de configuración de su dispositivo, conoce cual es la señal correcta que hay que enviar al dispositivo para realizar la orden solicitada.

- `DEFAULT`: el cliente solicita al servidor que establezca un dispositivo en su valor por defecto, que es el que viene indicado en el fichero de configuración.
- `UPDATE`: el cliente informa al servidor que la información enviada corresponde al estado actual del sistema. Así el servidor puede actualizar el estado global.

Tareas

Todos los servidores lanzan al iniciar la tarea *AutonomousMode*, para lo cual se utiliza un *Timer* de la API de Java. Esta tarea se encarga, de acuerdo al tiempo indicado en el fichero de configuración, de comprobar la fecha y hora en que se recibió la última petición, y si supera el tiempo indicado en el fichero de configuración, el servidor pasa al modo autónomo, que consiste en procesar un conjunto de reglas especificadas.

En el caso del servidor que posee las sondas de temperatura, se lanza también mediante el *Timer* de la API de Java una tarea que se encarga de leer las temperaturas cada cierto tiempo, y mostrarla en los display LED de 7 segmentos.

Modo Autónomo

Previsto inicialmente como un extra, el modo autónomo permite a los servidores funcionar si el cliente falla. El servidor posee una serie de reglas que debe de comprobar y ejecutar a partir del último estado conocido del sistema.

Para ello, cada servidor recibe, por parte del cliente, el estado global del sistema y se encarga de almacenarlo, manteniendo el estado actual y el anterior. Cuando el servidor detecta que el cliente ha fallado, se ejecuta el modo autónomo, que lo que hace es comprobar las reglas (que al inicio del servidor se leyeron del fichero `rules.config`) y determina si hay que cumplirlas o no.

Para comprobar las reglas, se decidió tomar un enfoque diferente al utilizado en el modo automático en el cliente: escribir las reglas en JavaScript, y comprobar si se cumplen o no utilizando las clases `javax.script.ScriptEngine` y `javax.script.ScriptEngineManager`.

La aplicación preprocesa las reglas para cambiar el nombre de los sensores por sus valores, o valores como ON u OFF reemplazarlos por las cadenas JavaScript ‘ON’/’OFF’, antes de comprobarlo con el método `eval()`, y obtener si la regla es verdadera o falsa.

A continuación se muestran como ejemplo las reglas implementadas en el servidor PiServer3

```
# Rules for autonomous mode
numRules=3
# Actuador1 = Encendido caldera
rule1=(SensorT < TOptimal) && (SensorA === ON ), Actuador1, ON
# Actuador2 = Válvula de 2 vías
rule2=(SensorT < TOptimal), Actuador2, ON
# Válvula de 3 vías y bomba
rule3=(SensorT < TOptimal) && (SensorA === ON ), Actuador3, ON
```

La primera parte, en negrita, es la regla escrita en *JavaScript*. La segunda y tercera parte son el sensor y el estado en que hay que ponerlo si la es cierta. Esta forma de definir reglas, permite crear reglas más complicadas, y en una sola línea, que el sistema utilizado en la implementación del sistema automático.

Clases

En la figura 39 se muestra parte del diagrama de bloques UML, solo con las clases más importantes, sin los atributos y los métodos. El UML completo se encuentra en el anexo 10.7. Y a continuación se detallan estas clases más importantes. El resto de clases no representadas en la figura , son clases que únicamente implementan diferentes métodos, estando todos los métodos de un tipo, en una misma clase. En este gráfico faltan también las clases que manejan los displays LED .

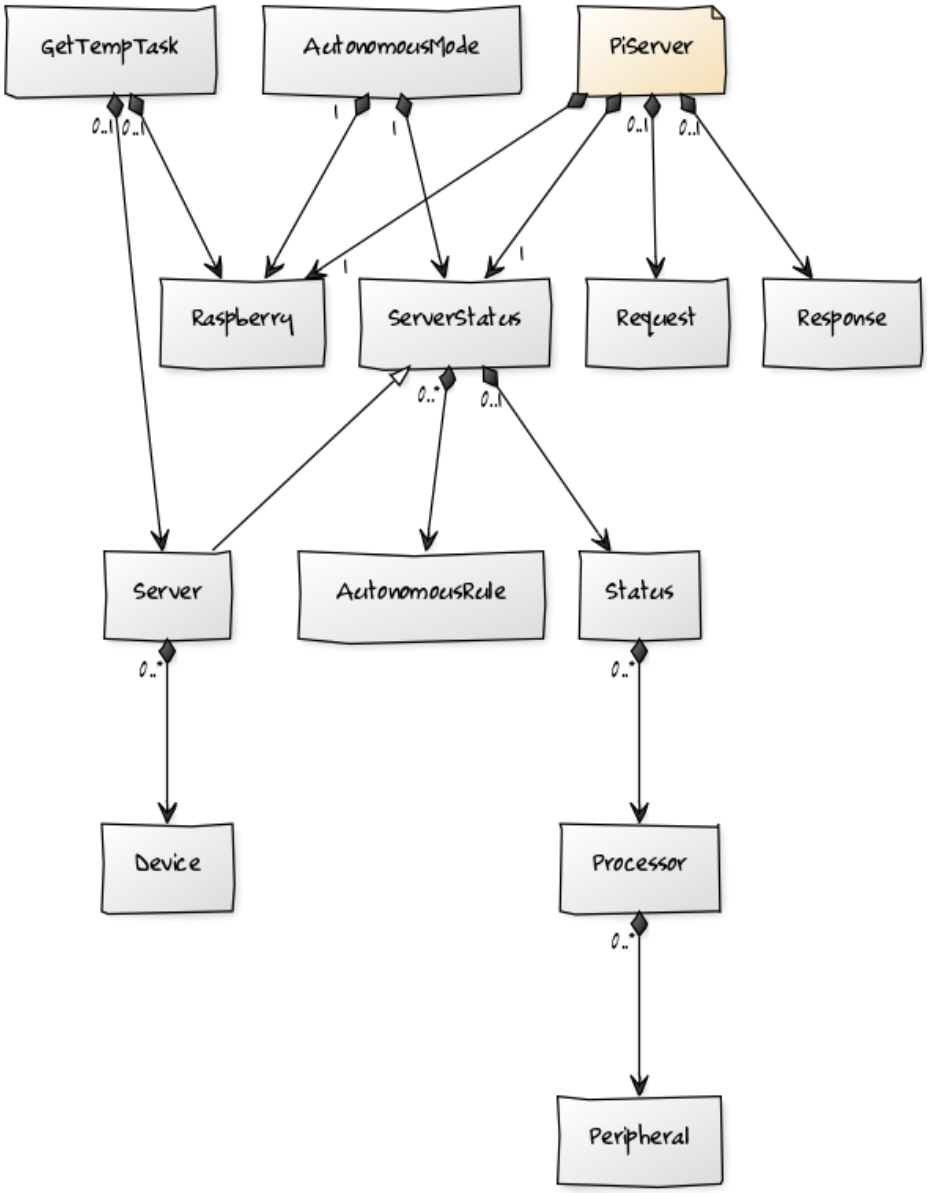


Figura 39: PiServer: UML reducido.
(Creado con: https://yuml.me/diagram)

Las clases son las siguientes:

- ☑ **PiServer**: Esta es la clase principal, que se encarga de utilizar el resto de métodos y clases. En esta clase inicia las tareas que se ejecutarán cada cierto tiempo.
- ☑ **Request**, **Response**: Estas dos clases implementan los objetos que contienen la petición que el cliente ha enviado, y la respuesta que el servidor enviará al cliente. Estas dos clases son enviadas a través del protocolo HTTP entre cliente y servidor, de forma serializada en formato JSON mediante los objetos y métodos proporcionados por la librería GSON. Estas dos clases se utilizan exactamente igual en la aplicación servidor.
- ☑ **Server**: clase que representa la información lógica de una Raspberry: nombre del servidor, puerto y dirección IP, fecha de última conexión. Contiene una lista de objetos Device, con la información de los dispositivos conectados. En el objeto instanciado también se almacena la configuración leída del fichero de configuración.
- ☑ **Device**: clase que representa la información lógica de cada uno de los dispositivos conectados a un servidor.
- ☑ **Raspberry**: clase que implementa toda la parte física de una Raspberry. Contiene los objetos instanciado de la librería Pi4J que representan al controlador GPIO de la Raspberry, al bus I2C y a todos los puertos GPIO utilizados.

  Todos los puertos GPIO se reservan al inicializar el servidor como puertos multipropósito, `GpioPinDigitalMultipurpose`. Esto permite utilizar una sola lista para almacenarlos, en lugar de tener dos, una con los de entrada y otra con los de salida. En función de como se hayan indicado en el fichero de configuración, se usan de entrada o de salida.
- ☑ **ServerStatus**: clase que extiende la clase Server con la información necesaria para funcionar en modo autónomo. Incluye una lista con todas las reglas a utilizar en dicho modo.
- ☑ **AutonomousRule**: clase que implementa una regla de funcionamiento autónomo en un servidor para el dispositivo indicado.
- ☑ **Status**: clase que implementa el estado global del sistema, y es la misma clase que la utilizada en parte cliente, por lo que está formada igualmente por una lista de objetos Processor, y cada uno de ellos con una lista de Peripheral.
- ☑ **Processor**: misma clase que en la parte cliente. Implementa la información lógica necesaria para el estado del sistema de cada uno de los servidores, cada uno de los cuales implementa una lista de objetos Peripheral.
- ☑ **Peripheral**: esta clase implementa la información lógica de cada dispositivo conectado a un servidor, y es la misma que la de la parte cliente.
- ☑ **GetTempTask**: clase que extiende la clase *TimerTask*, y por tanto, se ejecuta cada cierto intervalo de tiempo. Se encarga de obtener las temperaturas y de mostrarlas en los displays LED.

☑ **AutonomousMode**: esta clase también extiende la clase *TimerTask*, y por tanto, se ejecuta cada cierto tiempo preestablecido. Comprueba si ha transcurrido más de un determinado tiempo sin recibir peticiones por parte del cliente, y en caso afirmativo ejecuta las reglas establecidas para el modo de funcionamiento autónomo.

Otras clases definidas en el servidor son:

☑ **Constants**: clase que implementa todas las constantes del sistema, como nombre de los sensores, nombre de los ficheros de configuración, tipos de errores,.. así como las características de los dispositivos físicos, como las direcciones de los diferentes registro del ADC, o los valores característicos de las sondas NTC.

☑ **Communication**: clase que implementa los métodos de comunicaciones.

☑ **I2CDisplay**: clase que implementa el manejo de la controladora *Adafruit 0.54 inch Alphanumeric FeatherWing Display*. Modificada de la original de Eric Eliason para adaptarla a las necesidades de la aplicación y al display LED utilizado.

☑ **InputOutput**: clase que implementa los métodos específicos para acceder a los dispositivos hardware de la Raspberry, como activar o desactivar los puertos GPIO, leer el estado de un puerto, o leer un dato de un dispositivo I2C.

☑ **Processing**: clase que implementa métodos de procesamiento, como el de las ordenes recibidas, la actualización del estado global en el servidor o la conversión de un dato digital en temperatura.

☑ **Utilities**: clase con algunos métodos útiles, como la lectura de los ficheros de configuración, métodos de tratamiento de cadenas o conversores de hexadecimal o binario a bytes.

[4] Web PiSolar

La aplicación web se encuentra instalada en el cliente, y se se encarga de las siguientes funciones:

☑ Mostrar las temperaturas, para lo cual lee estos valores del estado actual almacenado en la base de datos SQL.

☑ Mostrar la evolución de las temperaturas a lo largo del tiempo de forma gráfica, leyendo los datos almacenados por la aplicación cliente en la base de datos.

☑ Almacenar las órdenes manuales que indique el usuario en la base de datos SQL.

El diagrama de bloques de esta aplicación es bastante sencillo, y es el que se muestra en la figura 40:

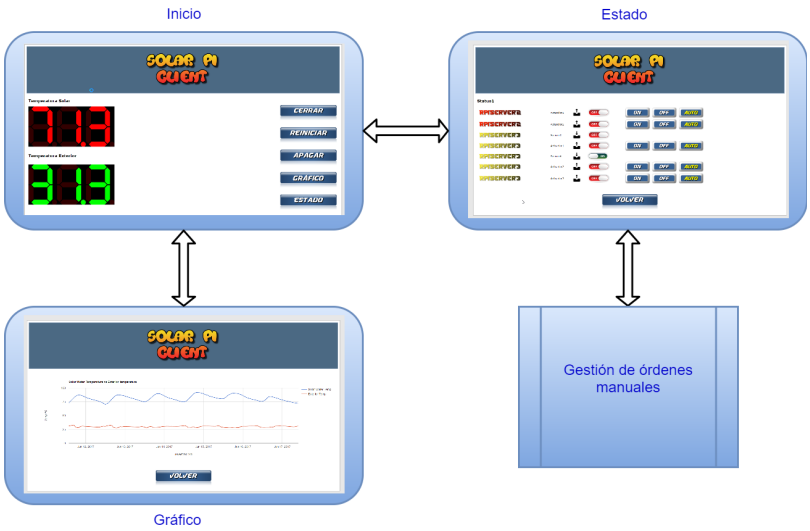


Figura 40: Diagrama de bloques de la aplicación PiSolar Web.

En el bloque de Inicio se muestran las temperaturas. Estas no son las temperaturas actuales, si no el último valor que almacena la aplicación piClient en la base de datos, el cual es el promedio de las últimas 30 temperaturas leídas. Este valor se puede modificar en el fichero de configuración del cliente.

Desde el inicio se pueden ejecutar las órdenes manuales que apagan o reinician todas las Raspberry que conforman la aplicación, o cierran la aplicación en todas ellas. Estas órdenes manuales son procesadas por la aplicación piControl

El bloque gráfico lee todas las temperaturas actualmente almacenadas en la base de datos, y utilizando la librería en JavaScript Google Charts [11] , se representa el gráfico.

En el bloque de estado, se muestra el último estado del sistema. Desde este bloque se pueden modificar manualmente el estado del sistema. Estas órdenes manuales permanecerán activas hasta que se vuelvan a poner manualmente en modo automático, cuando el sistema volverá a tomar el control de ellos.

No se ha implementado ningún control de acceso ni de seguridad, ya que solo se ha previsto su uso en la misma red local en la que se encuentran los dispositivos. En caso de querer utilizarla desde Internet se podrían efectuar las siguientes modificaciones:

- ☑ Validarse mediante usuario o contraseña.
- ☑ Habilitar las conexiones HTTPS: solo habría que configurar el servidor web de forma que acepte únicamente conexiones seguras desde los dispositivos que tengan el certificado y la clave privada generada por nosotros.
- ☑ Implementar las dos anteriores.

BASES DE DATOS

El servidor se ha creado un base de datos denominada Solar Pi que posee las siguientes tablas:

- ☑ **`manual`**: en esta tabla la aplicación Web Solar Pi se encarga de almacenar todas las ordenes manuales. Antes de almacenarla, se comprueba que no exista una alguna en proceso que afecte al mismo dispositivo, y en caso afirmativo, se modifica el estado en que hay que colocarlo.

  Esta tabla es leída por el cliente para comprobar si existe alguna orden, y en caso afirmativo procesarla.

- ☑ **`status`**: esta tabla contiene el estado actual del sistema, y es la aplicación piClient quien se encarga de actualizarla. La aplicación Web Solar Pi solo se encarga de leerla para mostrar el estado de forma gráfica.

- ☑ **`temperatura`**: en esta tabla se almacenan todos los valores de las temperaturas.

5 VIABILIDAD TÉCNICA

Es evidente que no solo es viable implementar el sistema, si no que en estos momentos se encuentra funcionando perfectamente, salvo problemas puntuales de comunicación con alguno de los servidores.

Existen en el mercado multitud de centralitas capaces de controlar el suministro de agua caliente sanitaria solar. Pero aquellas que permiten controlar tantos dispositivos como los utilizados en este sistema (6 electroválvulas, 1 bomba, 2 sondas de temperatura, control del encendido de la caldera y control del termostato del termo auxiliar y de la caldera) tienen un coste elevado.

Además, todas son cableadas, lo que puede requerir, si no se posee preinstalación en la vivienda, un coste añadido bastante elevado para realizar la instalación del cableado.

5.1 Puntos fuertes

Como puntos fuertes del sistema desarrollado se pueden considerar los siguientes:

- ☑ Utilización de microordenadores ampliamente disponibles y sobradamente conocidos, como es el caso de la Raspberry Pi.
- ☑ Bajo coste del material, sin tener en cuenta electroválvulas, bombas,... que habría que adquirir de igual forma utilizando cualquier otro dispositivo de control.
- ☑ Gran capacidad de ampliación.

  Tal y como está el sistema final, se podrían controlar hasta 4 sondas de temperatura con el ADC instalado. Se podrían instalar un total de 8 ADC iguales (cada uno con una dirección I2C diferente), lo que permitiría controlar 32 sondas de temperatura sin tener que modificar prácticamente el código.

  Lo mismo ocurre con los dispositivos actuadores (electroválvulas, bombas, …) ya que se pueden controlar tantos como puertos GPIO haya disponibles en cada servidor.

  Únicamente habría que modificar los ficheros de configuración de cada servidor, y la clase `Constants` para definir los nuevos nombres de las sondas y actuadores. Evidentemente habría que añadir nuevas reglas de funcionamiento.
- ☑ Conexión cableada, bien mediante cable *Ethernet* directo o bien mediante PLC, lo que implica que pueden situarse en cualquier lugar donde haya un enchufe de corriente.
- ☑ Sistema sencillo de mantener. En el caso que el software de una Raspberry falle, teniendo una imagen de la tarjeta, se puede volver a tener en marcha en cuestión de minutos.
- ☑ Si lo que falla es un componente físico, todos son fáciles de sustituir y se localizan fácilmente.

5.2 Puntos débiles

Como puntos débiles del sistema se pueden considerar los siguientes:

- ☑ Es un proyecto personal y totalmente adaptado a las condiciones de mi vivienda, por lo que puede no ser fácilmente extrapolable a otras situaciones.
- ☑ El código, aunque se ha comprobado, pudiera no estar lo suficientemente depurado, y siempre cabe la posibilidad de algún fallo no previsto.
- ☑ En el código se ha previsto unicamente la utilización de un modelo de ADC, y de un modelo de controlador de display LED, aunque añadir cualquier otro modelo no debería más complicado que añadir algunos métodos nuevos.
- ☑ No se puede acceder a la aplicación web de control desde Internet, aunque esto tiene fácil solución.
- ☑ Las tarjetas SD utilizadas en las Raspberry Pi tienen tendencia a degradarse, con los fallos de corriente. Esto daña el sistema operativo instalado en ellas, y una posible solución sería montar algún sistema que actúe como sistema de alimentación ininterrumpida, como por ejemplo, una batería de recarga de teléfonos móviles.
- ☑ La caja en la que están colocados los display LED no es muy atractiva.

6 VALORACIÓN ECONÓMICA

A continuación se detalla el presupuesto del proyecto.

6.1 Coste del material

Para la realización del presupuesto en lo relativo al coste del material, se han tenido en cuenta las siguiente consideraciones:

- En el presupuesto no se han incluido los gastos de electroválvulas, tuberías, bases de enchufes, etc. Tampoco gastos de envío de algunos componentes.
- Tampoco se ha incluido los gastos de la instalación de la fontanería ni la electricidad, ya que en condiciones normales, dichos trabajos lo realizaría un profesional del ramo, y es de suponer que en menos tiempo, aunque a un coste mayor. También hay que tener en cuenta que en muchas instalaciones, especialmente de fontanería, si hay dos personas realizando la instalación, el tiempo se reduce prácticamente a una cuarta parte.
- Si se han incluido algunos elementos como las Raspberry o las fuentes de alimentación, ya que aunque tenía algunos disponibles, en caso tener que desarrollarlo desde cero, si se necesitarían.
- En el presupuesto tampoco se incluye el disco duro SSD de 32GB, ya que no es necesario en el proyecto.
- Los precios indicados son IVA incluido.

Descripción	Unidades	Precio Ud.	Importe
Rasperry Pi model 3	4	39,90 €	159,60 €
Caja Raspberry Pi	1	9,88 €	9,88 €
Fuente de alimentación microUSB 5V, 2.5A	4	13,50 €	54,00 €
Fuente de alimentación 9V para placas relés.	2	9,99 €	19,98 €
Adaptador 9V a 5v/3.3V	2	4,99 €	9,98 €
Tarjeta microSD 16GB	4	8,99 €	35,96 €
Módulo Reles 5V 2 módulos	1	3,30 €	3,30 €
Módulo Reles 5V 4 módulos	1	7,49 €	7,49 €
Módulo level shifter 3.3V ⇔ 5V	2	2,38 €	4,76 €
Sonda NTC 10K	1	3,85 €	3,85 €
Kit 2 PLC TL-PA8010	1	49,00 €	49,00 €
Display LED 4digit 7segments Yellow + Backpac	1	20,82 €	20,82 €
Display LED 4digit 7segments White+ Backpack	1	22,63 €	22,63 €
Protoboard	2	0,50 €	1,00 €
Resistencias 10K (bolsa 20 unidades)	1	1,00 €	1,00 €
Conectores PCB 2pin (paquete 10 ud)	1	3,00 €	3,00 €
Cajas de superficie	4	5,00 €	20,00 €
Total			**426,25 €**

Tabla 6: Costes de materiales electrónicos.

6.2 Coste del desarrollo

Para la realización del presupuesto en lo relativo al coste del material, se han tenido en cuenta las siguiente consideraciones:

- El total de horas estimadas (mas de 500) incluye el estudio y búsqueda de información, la instalación de fontanería y electricidad. De ellas, se estima que unas 300 se utilizaron en el desarrollo del proyecto, tanto hardware como software.
- El precio de la hora de desarrollo se ha considerado de 25€, que se ha obtenido dividiendo el salario bruto promedio de un programador Java con entre 5 y 10 años de antigüedad en una empresa [47], y utilizando los cálculos encontrados en [48].
- Generalmente en un proyecto habrían participado diferentes personas, cada una con una tarea diferente: programador web, analista, programador java, ingeniero electrónico, … Cada uno de ellos emplearía un número de horas diferente y con diferente coste la hora.
- Se considera que el prototipo no tiene costes de mantenimiento, o son muy bajos. El único mantenimiento preventivo sería acceder a las Raspberry cada cierto tiempo a borrar los ficheros de log para que la tarjeta SD no se llene, pero para ello tienen que pasar muchos meses.
- También habría que comprobar que falla en caso de recibir el correo electrónico de aviso. En caso de problemas por la degradación de la tarjeta SD, bastaría con cambiarla por otra con la imagen.

Descripción	Unidades	Precio Ud.	Importe
Horas de desarrollo	260	24,00 €	6.240,00 €
Instalación y pruebas	40	24,00 €	960,00 €
Total			**7.200,00 €**

Tabla 7: Costes de desarrollo e instalación.

6.3 Coste de industrialización

En el coste de la industrialización se ha considerado lo siguiente:

- No se incluye el coste del desarrollo, ya que se supone se amortizará al añadir el margen de venta del producto.
- Se supone un margen de reducción del coste al fabricarse de forma industrial del 40%, aunque este margen podría reducirse hasta casi el 80% en algunos componentes dependiendo del lugar donde se fabrique.
- En el proyecto se han utilizado 2 fuentes de alimentación en los servidores con relés. Una para la Raspberry y otra que alimenta la placa de relés, lo que requiere adaptador de

corriente a 5V. En una producción industrial, se usaría una única fuente de alimentación con salidas aisladas. Modelos similares para alimentar tiras de LED salen por 6€.

- Los PLC se podrían sustituir por modelos más baratos y lentos. En este proyecto se eligió el modelo PA8100 de 1200Mbps para emparejarlos con los ya existentes.
- Las placas ADC y adaptadores de nivel, junto con las placas utilizadas para crear los divisores de tensión y las resistencias pull-up se podrían fabricar también en una única placa a un coste muy reducido, y podría integrarse en todas las Raspberry, de forma que se conectara cuando se necesite. Lo mismo podría hacerse con los módulos de relés. Se ha realizado el cálculo suponiendo que cada servidor incorpora un módulo de 4 relés. También podría añadirse un display LED o LCD a cada una de ellas.

Descripción	Unidades	Precio Ud.	Importe
Rasperry Pi model 3	1	23,94 €	23,94 €
Fuente de alimentación 5V	1	6,00 €	6,00 €
Tarjeta microSD 16GB	1	5,39 €	5,39 €
Módulo Reles 5V 4 módulos	1	4,49 €	4,49 €
Placa electrónica integrada	1	1,00 €	1,00 €
Sonda NTC 10K	1	2,31 €	2,31 €
PLC	1	20,00 €	20,00 €
Display LED o LCD	1	12,49 €	12,49 €
Caja	1	2,00 €	2,00 €
Total			77,63 €

Tabla 8: Coste de industrialización de 1 dispositivo servidor.

Como se puede observar, el sistema completo saldría por menos de 300€, y eso sin considerar que algunos componentes podrían ser modulares y no instalarse salvo que se necesiten.

El coste de mantenimiento del sistema vendría dado por ejemplo por sustituir alguno de los componentes por otros, como pudiera ser el ADC.

Y en cuanto a las certificaciones, considero que no es necesario, ya que se supone que cada uno de los componentes tendría sus propias certificaciones.

7 CONCLUSIONES

En este proyecto se ha podido comprobar que un conjunto de pequeños miniordenadores como la Raspberry Pi pueden desarrollar tareas relativamente complejas, y como alguien dijo alguna vez “el único límite es tu imaginación”.

7.1 Conclusiones del trabajo y lecciones aprendidas

Al inicio del proyecto se plantearon la siguiente serie de objetivos principales:

- ☑ Detección de la temperatura del agua acumulada en el termosifón.
- ☑ Conexión de todos los micro-ordenadores mediante red Ethernet.
- ☑ Detectar el estado del termostato del acumulador auxiliar.
- ☑ Control electrónico remoto del encendido y apagado de las diferentes electroválvulas.
- ☑ Diseño de un sistema de control automático.
- ☑ Diseño de un sistema de control manual.
- ☑ Detectar el estado de funcionamiento de la caldera.
- ☑ Crear un página web sencilla de información y control.

Y dos objetivos secundarios:

- ☑ Aplicación web que muestre la evolución histórica de la temperatura
- ☑ Diseño de un modo autónomo.

Además de cumplirse tanto los objetivos primarios como los secundarios, durante el desarrollo del proyecto se han añadido algún otro extra no previsto, como el envío de correos electrónicos de aviso de fallo, o el añadir un par de displays para mostrar la temperatura.

En cuanto a las lecciones aprendidas, aunque ya había utilizado la Raspberry Pi, siempre la había utilizado como servidor de páginas web y de ownCloud, nunca había programado en ella, y menos aún había utilizado sus puertos GPIO. Por tanto, ha sido bastante satisfactorio el hecho de programar con ella, consiguiendo leer puertos I2C y controlar electroválvulas.

En general, considero que lo mejor del proyecto es que me ha servido para poner “juntas” muchas de las cosas aprendidas en diferentes asignaturas durante estos años de estudio.

7.2 Autoevaluación.

En general estoy bastante satisfecho con todo el trabajo realizado. Ha sido muy duro, ya que al final eran muchas las tareas a realizar, y he conseguido llevarlo a cabo, aunque he tenido que aprovechar todo el tiempo disponible, incluyendo fines de semana, festivos e incluso pidiendo días de vacaciones en el trabajo. A pesar de todo, considero que todo el esfuerzo ha merecido

la pena ya que he conseguido realizar un proyecto que llevaba tiempo pensando, y aunque no se ha llevado a cabo tal y como inicialmente tenía en mente, el sistema utilizado me da mucha más confianza a la hora de evitar problemas y fallos que el que había pensado.

Durante el proyecto me he tropezado con puntos complicados, como el desarrollo y la implementación del modo automático de funcionamiento, que con paciencia y buenas ideas he sacado hacia adelante. Y a veces, errores tontos en la programación me han dado días de quebraderos de cabeza

Ha habido problemas con la planificación de determinadas tareas, debido a algunos retrasos en el envío de algún material electrónico, y un problema con los tubos de acero flexible encargados a medida, hizo que el fabricante, AceroSolar [49], tuviera que reenviarlos. Fueron muy amables y rápidos en la solución del problema, y lo hicieron sin coste alguno por mi parte.

Sin embargo, los retrasos en la recepción del material hicieron que pudiera dedicar el tiempo a adelantar algunas fases o a implementar alguna tarea no prevista inicialmente.

En definitiva, durante el desarrollo del proyecto, no se han cumplido las objetivos previstos en la planificación inicial, pero reordenando las diferentes tareas, se ha conseguido llevar a buen término todas las tareas, incluidas las extras y alguna no prevista inicialmente, y todo dentro de los plazos previstos.

7.3 Las líneas de trabajo futuro.

Aunque todas las tareas previstas inicialmente se han llevado a cabo, el conocer todas las posibilidades que proporciona la Raspberry me ha llevado a pensar de momento en un par posibles futuras ampliaciones como:

- ☑ Añadir un display LCD al Cliente, de tal forma que a la vez que genera el log, vaya mostrándolos en el display.
- ☑ Implementar el envío de correos de aviso en la aplicación servidor, de tal forma que envíe un correo si pasa a funcionar en modo autónomo, lo que implica que el cliente está caído.
- ☑ Implementar uno de los servidores, o uno nuevo, en un miniordenador Orange Pi, en teoría compatible con la Raspberry, aunque utiliza un SoC totalmente diferente.
- ☑ Como ya tengo una Raspberry Pi en cada planta de la casa, podría colocar en cada una de ellas sensores de temperatura, de tal forma que se pueda controlar la calefacción individualmente por planta.

  El control del calefacción por planta sería relativamente sencillo y barato, ya que a la salida de la caldera existe un distribuidor que reparte el agua caliente de la calefacción por plantas. Colocando unas electroválvulas se controlaría el reparto a cada planta.

8 GLOSARIO

A

ACS: Agua caliente sanitaria.

ADC:*Analogic-Digital Converter*. Conversor analógico a digital.

API: *Application Programming Interface*, hace referencia al conjunto de funciones y procedimientos,u objetos y métodos en el caso de la POO, que proporciona una librería o biblioteca para ser utilizado desde otras aplicaciones.

ARM: Originalmente proviene de *Acorn RISC Machine*, ya que fue inicialmente diseñada por Acorn Computers, y hace referencia a una arquitectura RISC. Luego pasó a hacer referencia a *Advanced RISC Machine.*

C

CSMA/CD: *Carrier Sense Multiple Access / Collision Detection*: Acceso Múltiple por Detección de Portadora con Detección de Colisión. Es la política de acceso al medio utilizada en el protocolo Ethernet.

G

GPIO: *General Purpose Input Output*. Hace referencia a un pin en un dispositivo, cuyo comportamiento se puede especificar por programa.

GSON: o Google GSON, es el nombre de la librería Java que permite serializar/deserializar objetos en formato JSON.

I

I2C ó IIC: *Inter-Integrated Circuit*. Es un bus de datos en serie desarrollado por Philips.

IDE: *Integrated Development Environment* o entorno de desarrollo integrado.

IOT: *Internet of Things* o Internet de las cosas, es un concepto que hace referencia a la inter-conexión de objetos de uso cotidiano a Internet.

J

JSON: *JavaScript Object Notation*. Es un formato de texto definido inicialmente para intercambiar datos en JavaScript. Hoy en día se considera un formato independiente del lenguaje.

L

LAMP: *Linux Apache MySQL PHP*. Este acrónimo hace referencia a los servidores que utilizan Linux como sistema operativo, Apache como servidor de páginas web, MySQL/MariaDB como gestor de base de datos y PHP como lenguaje de programación de páginas web dinámicas.

LAN: *Local Area Network* o red de área local.

LEDE: Linux Embedded Development Environment. Adaptación de OpenWRT, con kernel Linux v 2.

LOG4J: Librería Java desarrollada por la Apache Software Foundation para manejar el registro o log de los mensajes de aviso y error.

M

MESH: es una red en malla implementada sobre una red inalámbrica. Es una red en la que se mezclan las dos topologías de redes inalámbricas: descentralizadas o *ad hoc* y de infraestructura, donde existe un dispositivo (punto de acceso) que centraliza todas las comunicaciones.

N

NTC: *Negative Temperature Coefficient.* Un termistor sensor de temperatura es aquel que varía su resistencia de acuerdo a la temperatura. En el caso de los NTC, disminuyela resistencia al aumentar la temperatura.

O

OpenWRT: distribución de Linux diseñada para ser utilizada principalmente en routers caseros o semiprofesionales.

P

PHP: Lenguaje de programación de uso general diseñado inicialmente para el desarrollo de aplicaciones web dinámicas. Permite tanto programación procedural como orientada a objetos.

Pi4J: Librería Java que permite el acceso a los puertos de comunicación GPIO de la Raspberry.

PLC: Power Line Communications. Tecnologías de comunicaciones que utilizan como medio de conexión las lineas eléctricas.

POO: Programación Orientada a Objetos.

R

RISC: *Reduced Instruction Set Computer*, ordenador con conjunto de instrucciones reducido.

RTOS: Sistema operativo en tiempo real.

S

Socket: Un socket es un concepto abstracto, un tipo de fichero especial, un mecanismo que permite establecer un enlace de comunicación bidireccional entre dos programas que se ejecutan en la red.

V

Vbus: Bus de conexión diseñado por *RESOL - Elektronische Regelungen GmbH* para conectar controladores electrónicos entre sí mediante una conexión de 2 o 4 hilos.

W

WPAN: *Wireless Personal Area Network*, o red de área personal inalámbrica. Es una red local de ámbito reducido, donde los dispositivos están muy cercanos al punto de acceso (máximo 10 metros).

9 BIBLIOGRAFÍA

[1] Boletín Oficial del Estado: Real Decreto 314/2006, de 17 de marzo, por el que se aprueba el Código Técnico de la Edificación, publicado en el «BOE» núm. 74, de 28 de marzo de 2006.

[https://www.boe.es/buscar/doc.php?id=BOE-A-2006-5515], 12/05/2017

[2] Evolución del precio gasóleo Calefacción Andalucía.

[https://www.clickgasóleo.com/c/evolucion-del-precio-gasóleo-calefaccion-andalucia], 10/06/2017

[3] Smartsheet: Work Management and Automation Solutions.

[https://app.smartsheet.com/b/home], 19/03/2017, 18/04/2017, 02/05/2017, 15/05/2017, 10/06/2017.

[4] The Pi4J Project: Java I/O library for the Raspberry Pi:

[http://pi4j.com/], 18/03/2017, 21/03/2017, 30/04/2017, 02/05/2017, 04/05/2017

[5] GitHub: Java I/O library for Raspberry Pi (GPIO, I2C, SPI, UART)

[https://github.com/Pi4J/pi4j], 02/05/2017, 04/05/2017

[6] Energía Solar: Energía solar térmica.

[https://solar-energia.net/energia-solar-termica], 02/03/2017, 12/06/2017

[7] W3Schools.com: HTTP Methods: GET vs. POST

[https://www.w3schools.com/tags/ref_httpmethods.asp], 05/03/2017

[8] TP-LINK.

[http://www.tp-link.es], 13/06/2017

[9] Java Mail:

[https://javaee.github.io/javamail/], 12/04/2017

[10] Enviar un correo con JavaMail:

[http://www.chuidiang.org/java/herramientas/javamail/enviar-correo-javamail.php]

[11] Google Charts

[https://developers.google.com/chart/], 12/05/2017

[12] Apache Log4j 2.

[https://logging.apache.org/log4j/2.x/], 15/05/2017

[13] Raspberry Pi GPIO Header Pin Protection

[http://www.petervis.com/Raspberry_PI/Raspberry_Pi_GPIO_Header/Raspberry_Pi_GPIO_Header_Pin_Protection.html], 24/03/2017

[14] GitHub: A Java serialization/deserialization library to convert Java Objects into JSON and back

[https://github.com/google/gson], 10/04/2017

[15] Google Gson Homepage

[https://sites.google.com/site/gson/Home], 10/04/2017

[16] Mkyong: How to convert Java object to / from JSON (Gson)

[https://www.mkyong.com/java/how-do-convert-java-object-to-from-json-format-gson-api/], 10/04/2017

[17] MariaDB

[https://mariadb.com/], 28/04/2017

[18] About MariaDB Connector/J

[https://mariadb.com/kb/en/mariadb/about-mariadb-connector-j/], 28/04/2017

[19] SQLite Homepage

[https://www.sqlite.org/], 15/04/2017

[20] PHP: SQLite Manual

[http://php.net/manual/es/book.sqlite.php], 15/04/2017

[21] Adafruit: Learn Raspberry Pi

[https://learn.adafruit.com/], 18/03/2017

[22] Adafruit Forums: Quad Alphanumeric Displays

[https://forums.adafruit.com/viewtopic.php?f=47&t=103621] , 21/04/2017

[23] Gordon Projects

[https://projects.drogon.net/], 18/03/2017

[24] Learn Sparkfun

[https://learn.sparkfun.com/], 18/03/2017

[25] Physical computing with Raspberry Pi

[https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/robot/buttons_and_switches/], 18/03/2017

[26] Raspberry Pi - Teach, Learn, and Make with Raspberry Pi

[https://www.raspberrypi.org/], 22/03/2017

[27] Arduino-info Wiki!: RELAYS and OPTO-ISOLATION CONSIDERATIONS

[https://arduino-info.wikispaces.com/RelayIsolation], 20/04/2017

[28] ADS111x Datasheet

[http://www.ti.com/lit/ds/symlink/ads1114.pdf], 02/05/2017

[29] Bristolwatch.com: ADS1115 4-Channel ADC Uses I2C with Raspberry Pi

[http://www.bristolwatch.com/rpi/ads1115.html], 05/05/2017

[30] Coderwall: Run a script on startup in a detached screen on a Raspberry Pi

[https://coderwall.com/p/quflrg/run-a-script-on-startup-in-a-detached-screen-on-a-raspberry-pi], 05/06/2017

[31] SevenSeg.js: A tiny JavaScript jQuery UI plugin for creating vector-based (SVG) seven-segment displays in HTML.

[http://brandonlwhite.github.io/sevenSeg.js/], 12/05/2017

[32] Sparkfun Electronics.

[http://cdn.sparkfun.com/datasheets/BreakoutBoards/Logic_Level_Bidirectional.pdf], 16/05/2017

[33] Newark element14

[http://www.newark.com/], 12/06/2017

[34] Onion: Omega 2

[https://onion.io/omega2/], 12/06/2017

[35] Beagleboard.org: BeagleBone Black

[http://beagleboard.org/black],

[36] NXP: LPCXpresso board for LPC1769 with CMSIS DAP probe

[http://www.nxp.com/products/software-and-tools/software-development-tools/software-tools/lpcxpresso-boards/lpcxpresso-board-for-lpc1769-with-cmsis-dap-probe:OM13085]

[37] Coltech: Centralita solar CT04

[http://www.coltech.es/centralita-solar.html], 13/06/2017

[38] Kaysun – Centralitas.

[http://www.kaysun.es/es/productos/detalle/centralitas], 13/06/2017

[39] Salvador Escoda: Tarifa Energías Renovables.

[http://www.salvadorescoda.com/tarifas/Energias_Renovables_Tarifa_PVP_SalvadorEscoda.pdf], 13/06/2017

[40] Climamania: Interacumulador de apoyo electrico AFE150N1.

[http://www.climamania.com/afe150n1.html], 15/03/2017

[41] Orange Pi

[http://www.orangepi.org/], 14/06/2017

[42] Banana Pi

[http://www.banana-pi.org/], 14/06/2017

[43] HardKernel: Odroid products.

[http://www.hardkernel.com/main/main.php], 14/06/2017

[44] Acorn Computers: Creadores de la arquitectura ARM.

[https://es.wikipedia.org/wiki/Acorn_Computers], 15/06/2017

[45] James Reuben Knowles: 5V to 3.3V Logic Level Shifting Stragety Notes

[http://jamesreubenknowles.com/level-shifting-stragety-experments-1741], 23/04/2017

[46] yUML: Create and share simple UML diagrams in your blogs, wikis, forums, bug-trackers and emails.

[https://yuml.me/diagram], 17/06/2017

[47] OpenWebinars: ¿Cuánto gana un programador Java en España?

[https://openwebinars.net/blog/cuanto-gana-un-programador-java-en-espana/], 17/06/2017

[48] Pymes y autónomos: ¿Sabes calcular el precio por hora de tu trabajo?

[https://www.pymesyautonomos.com/vocacion-de-empresa/sabes-estimar-el-precio-por-hora-de-tu-trabajo], 17/06/2017

[49] Acero Solar: fabricante de tubos de acero flexible especial para uso en STE.

[http://www.acerosolar.com/], 12/03/2017

10 ANEXOS

10.1 ADS1115

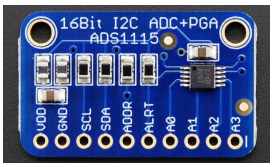


Figura 41: Placa ADC 16bit ADS1115

(fuente: Adafruit.com)

El ADS1115 [28],[29] es un convertidor analógico a digital cuyas principales características son:

- ☑ 4 canales con una precisión de 16bits. En realidad la resolución es de 15bits, ya que uno de los bits se utiliza como bit de signo.
- ☑ Puede configurarse para utilizar los cuatro canales independientemente, o como dos canales diferenciales.
- ☑ Ratio de muestra programable desde 8 hasta 860 SPS (muestras por segundo).
- ☑ Bus I2C, con dirección seleccionable
- ☑ Incluye un amplificador de ganancia programable (PGA) de hasta x16.
- ☑ Puede funcionar con tensiones entre 2 y 5.5V.
- ☑ Librerías para su utilización en Arduino y Raspberry Pi, con soporte nativo para Python (*Adafruit Pi Python Library*). También tiene soporte en Java mediante la librería Pi4J [4], [5].

El ADS1115 posee 3 registros:

- ☑ Register Pointer. Es un registro puntero de 8 bits utilizado para indicar en cual de los otros dos registros de 16 bits se quiere acceder. Si se almacena un 0, se accede al registro de datos, y si es un 1 al registro de configuración.
- ☑ Data Register. Este registro de 16 bits almacena el valor leído por el ADC de acuerdo a la configuración establecida en el registro de configuración. En realidad solo hay un dato de 15bits, ya que el bit más alto se utiliza para indicar cuando el dato está listo. En el modo diferencial se usa como bit de signo.
- ☑ Configuration Register: Este registro de 16 bits indica la configuración del ADC, tal y como se muestra en la tabla 9.

Bits	Significado
15	OS: Flag bit for single shot
14-12	MUX2, MUX1, MUX0: Select the input: 100 ANC0; 101 ANC1; 110 ANC2; 111 ANC3

11-09	PGA2, PGA1, PGA0: Internal Amp gain. Default to 010
08	MODE: Operational mode of the ADS1115. 0: Continuous conversion mode. 1: Power-down single-shot mode (default). In this case, bit D15 serves as"Ready" flag, and it's set when data is ready.
07-05	DR2, DR1, DR0: Set data rate. Default to 100 for 128SPS
04-00	COMP_MODE, COMP_POL, COMP_LAT, COMP_QUE1, COMP_QUE0: Bits 4-0 are comparator functions.

Tabla 9: ADS1115 Configuration Register.

10.2 Level Shifter

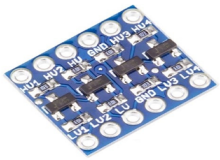


Figura 42: Placa Level Shifter

(fuente: Amazon.es)

Un *level shifter* es dispositivo que convierte los niveles lógicos de las señales. Muchos dispositivos poseen entradas y salidas de 5V, mientras que otros poseen 3.3V.

Aplicar una tensión de 5V a una entrada de 3.3V puede terminar por dañar el dispositivo de 3.3V, por lo que si queremos conectarlos sin riesgos hay que adaptar las señales. Una forma es utilizar un adaptador de señales (*Level Shifter* o *Logic Level Converter*).

Para el proyecto, las relés elegidos funcionan a 5V, por lo que lo mejor es adaptar la señal de los 3.3V de la Raspberry a los 5V que requieren la placa de relés optoaislados.

Las forma habitual de adaptar la señal es bien utilizando un transistor y resistencias, como se puede ver en la figura 43, o bien utilizar un circuito integrado especializado como el 4050 o el 74LVX245 , entre otras posibilidades [45] :

Para construir este adaptador se requieren 2 ó 3 resistencias y un transistor. Sin embargo, una placa con cuatro adaptadores de señal como el de la figura 44 cuesta menos de 3 euros, con el consiguiente ahorro de tiempo a la hora de cablear y soldar.

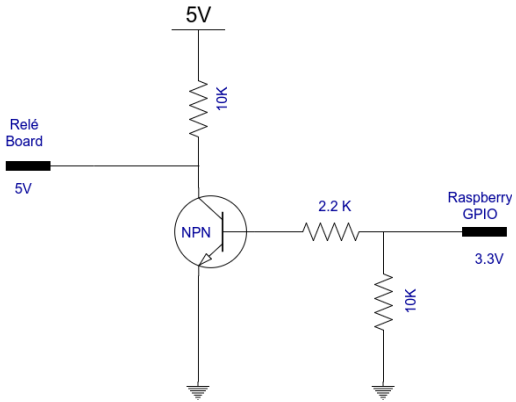


Figura 43: Esquema del adaptador de nivel 3.3 -> 5V.

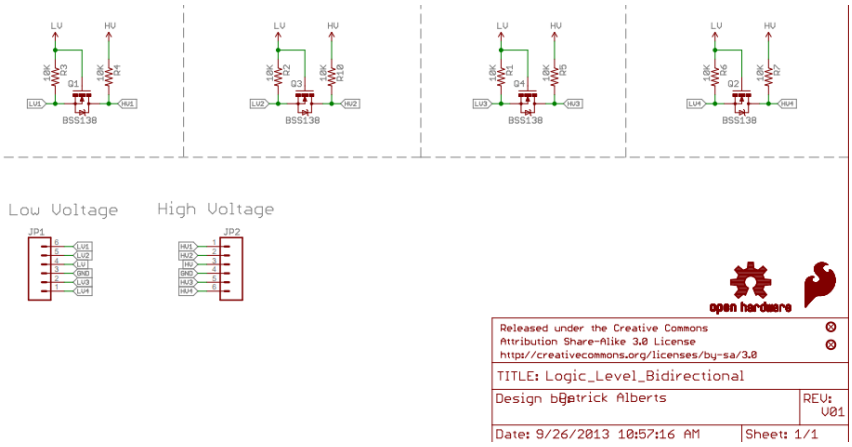


Figura 44: Adaptador de niveles bidireccional.
(fuente: https://www.sunfounder.com/)

10.3 Relés

Un relé se define como un dispositivo, en el cual al aplicarse una corriente electríca avre o cierra un circuito. En el proyecto se han utilizado 2 tipos de reles.

El primero, ha sido Finder 40.51, que se corresponde con el de la figura 45. Es un relé electro-magnético de propósito general que funciona a 230V CA, y que soporta 250V y 16A en los contactos. Se ha utilizado para detectar si la caldera se encuentra o no encendida y si el termostato del acumulador auxiliar se encuentra activo o no, ya que ambos funcionan a 230V,

Figura 45: Relé Finder 40.61, 16A 250V.
(fuente: ebay.es)

El segundo tipo utilizado han sido relés de estado sólido y que se encuentran opto-aislados como el de la figura 46, y que se han utilizado en un módulo de 2 relés y otro de 4 relés.

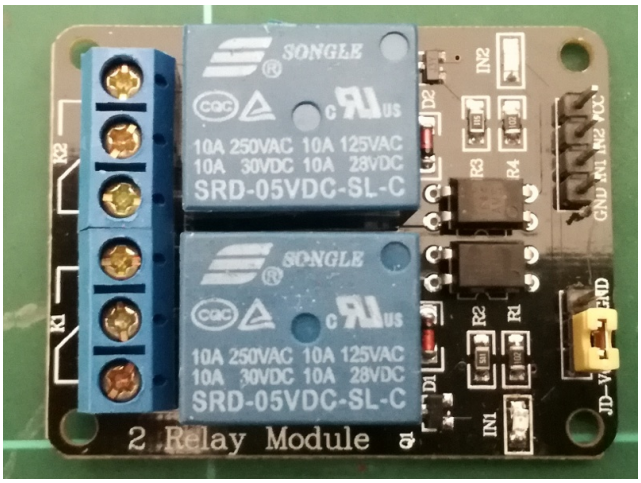


Figura 46: Placa con 2 reles opto-aislados.

Estos relés, tal y como indica su serigrafía, soportan hasta 10A con 250V AC. Cada electroválvula consume 6W a 230V, que son unos 26mA, muy lejos de los 10A soportados por los relés

En la página “Relays and Opto-Isolatión Considerations” [27] se ha localizado el esquema de un relé opto-aislado y que se muestra en la figura :

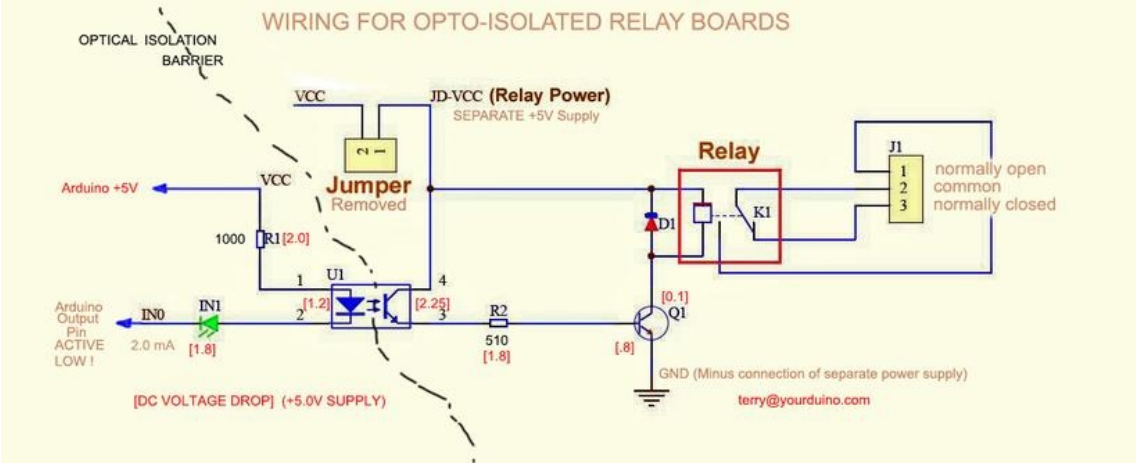


Figura 47: Esquema de relé opto-aislado
(fuente: https://arduino-info.wikispaces.com/RelayIsolation)

Para mantener el aislamiento, hay que utilizar fuentes de alimentación diferenciadas para el relé y para el dispositivo (Raspberry, Arduino o similar..) y eliminar el *jumper* entre Vcc JD-VCC para que las alimentaciones se encuentren separadas.

En este caso, se ha realizado otro aproximación. Se mantienen las dos fuentes de alimentación diferenciadas. Una microUSB para la Raspberry, y otra de 9V conectada a un módulo de alimentación para placas de pruebas (breadboard) como el de la figura 48, que proporciona 3.3V y 5V. Desde esta placa se alimenta la placa de relés.

Figura 48: Módulo de alimentación para breadboards.

La conexión entre los GPIO de la Raspberry y los puertos de entrada del módulo de relés se encuentran aislados al utilizar un adaptador de nivel de 3.3V a 5V.

10.4 PLC TL-PA8010

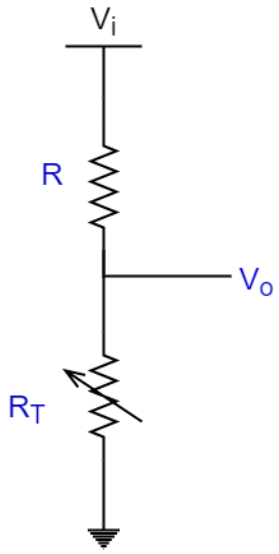
El PLC elegido ha sido el modelo TL-PA8010b debido a su relación precio/velocidad más que razonable, unido al hecho de que ya poseía dos unidades de dicho modelo. Finalmente se ha formado una red de 4 PLC como puede verse en la figura 19. Entre sus características tenemos las siguientes (fuente: [8]) :

- ☑ Tipo de enchufe: Schuko (U.E. y U.K.)
- ☑ Protocolos y estándares: HomePlug AV2, HomePlug AV, IEEE1901, IEEE802.3, IEEE802.3u, IEEE802.3ab.
- ☑ Velocidad máxima teórica: 1200Mbps.
- ☑ Interfaz: 1 puerto Gigabit Ethernet.
- ☑ Consumo: 6W, 0.5W en reposo.
- ☑ Alcance: 300m en el circuito eléctrico.
- ☑ Soporte tecnología MIMO (*Multiple Input Multiple Output*): se utilizan los tres cables de la línea eléctrica (Fase, Neutro y Tierra) para establecer múltiples rutas aumentando el rendimiento.

Figura 49: Kit PLC TL-PA8010
(fuente: www.tp-link.es)

10.5 Divisor de tensión

Aplicando la ley de Ohm en el divisor de tensión tenemos que:



$$V_o = \frac{V_i \cdot R_T}{(R + R_T)} \Rightarrow$$

$$V_o \cdot (R + R_T) = V_i \cdot R_T \Rightarrow$$

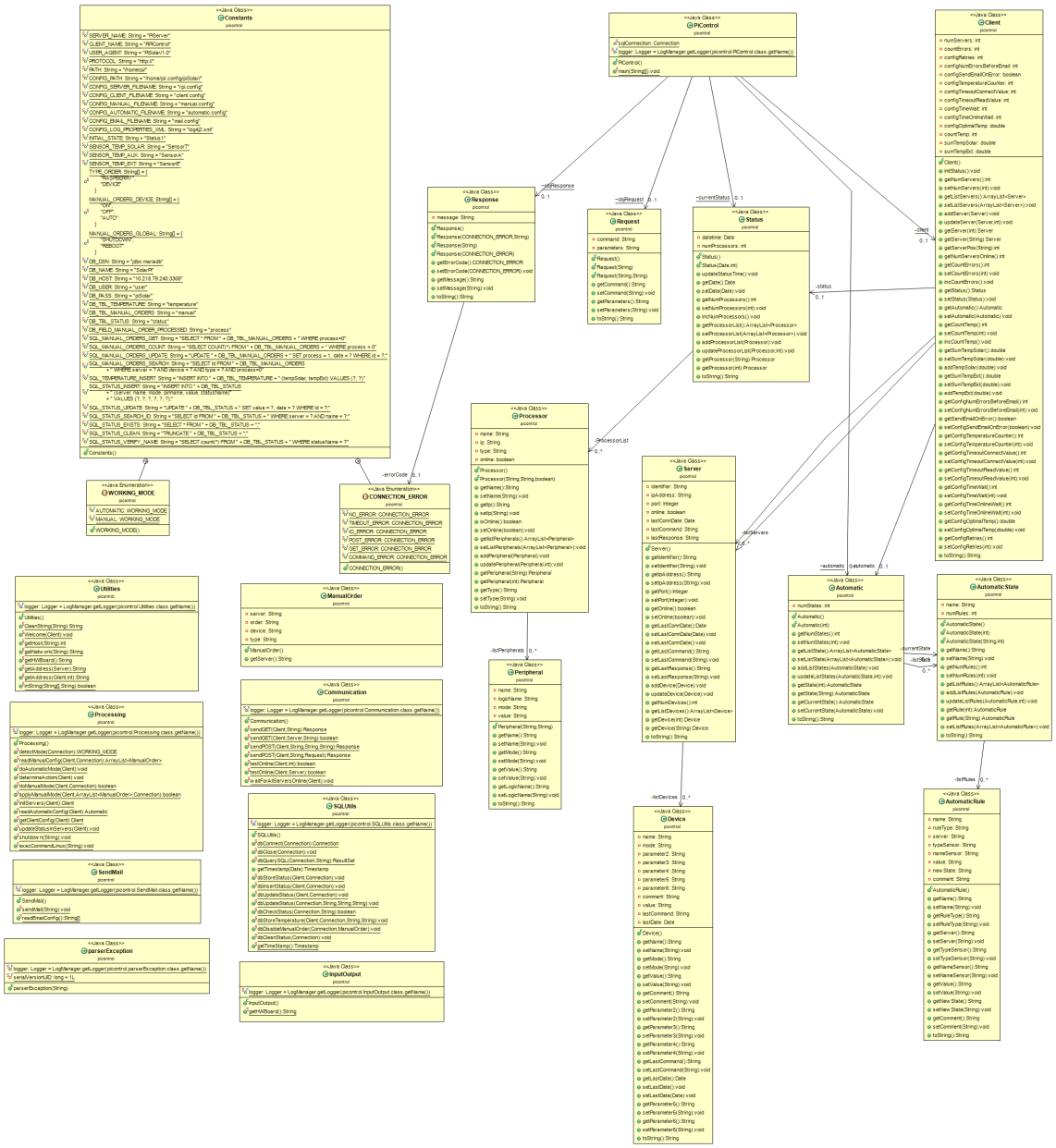
$$V_o \cdot R + V_o \cdot R_T = V_i \cdot R_T \Rightarrow$$

$$V_o \cdot R = V_i \cdot R_T - V_o \cdot R_T \Rightarrow$$

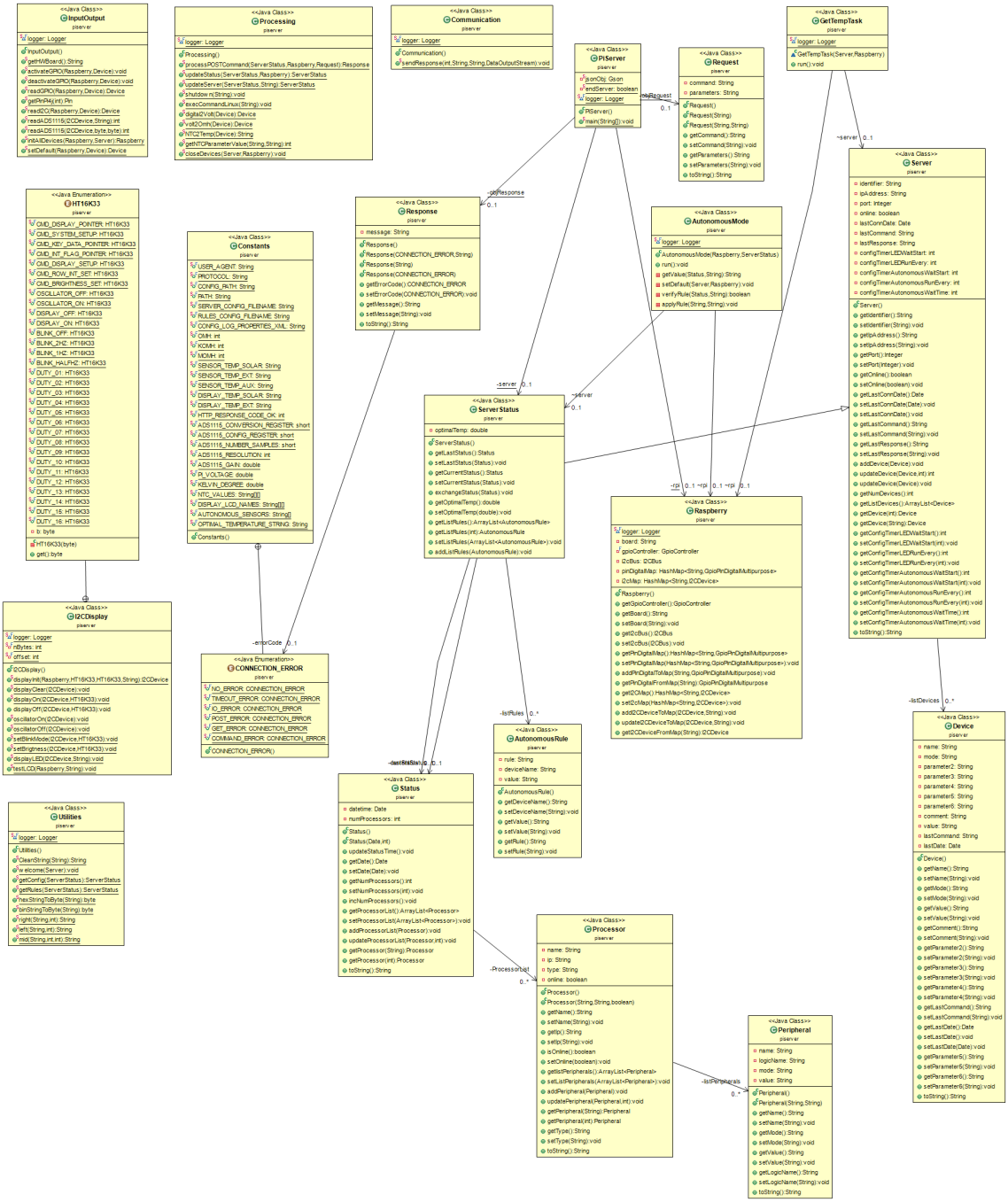
$$V_o \cdot R = (V_i - V_o) \cdot R_T \Rightarrow$$

$$\boxed{R_T = \frac{R \cdot V_o}{(V_i - V_o)}}$$

10.6 PiControl UML

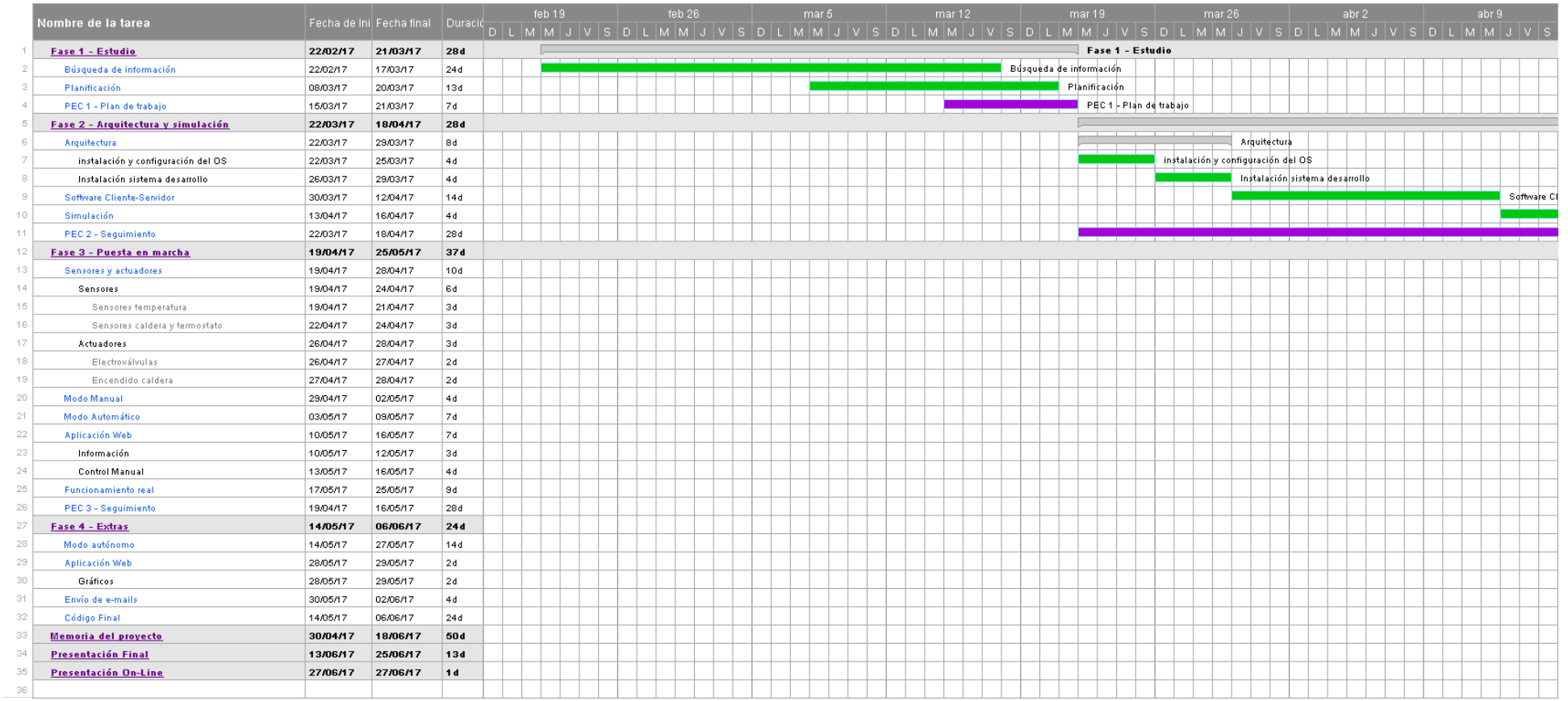


10.7 PiServer UML



10.8 Diagramas de Gantt

[1] Inicial



	Nombre de la tarea	Fecha de Ini	Fecha final	Duraci
1	**Fase 1 - Estudio**	**22/02/17**	**21/03/17**	**28d**
2	Búsqueda de información	22/02/17	17/03/17	24d
3	Planificación	08/03/17	20/03/17	13d
4	PEC 1 - Plan de trabajo	15/03/17	21/03/17	7d
5	**Fase 2 - Arquitectura y simulación**	**22/03/17**	**18/04/17**	**28d**
6	Arquitectura	22/03/17	29/03/17	8d
7	instalación y configuración del OS	22/03/17	25/03/17	4d
8	Instalación sistema desarrollo	26/03/17	29/03/17	4d
9	Software Cliente-Servidor	30/03/17	12/04/17	14d
10	Simulación	13/04/17	16/04/17	4d
11	PEC 2 - Seguimiento	22/03/17	18/04/17	28d
12	**Fase 3 - Puesta en marcha**	**19/04/17**	**25/05/17**	**37d**
13	Sensores y actuadores	19/04/17	28/04/17	10d
14	Sensores	19/04/17	24/04/17	6d
15	Sensores temperatura	19/04/17	21/04/17	3d
16	Sensores caldera y termostato	22/04/17	24/04/17	3d
17	Actuadores	26/04/17	28/04/17	3d
18	Electroválvulas	26/04/17	27/04/17	2d
19	Encendido caldera	27/04/17	28/04/17	2d
20	Modo Manual	29/04/17	02/05/17	4d
21	Modo Automático	03/05/17	09/05/17	7d
22	Aplicación Web	10/05/17	16/05/17	7d
23	Información	10/05/17	12/05/17	3d
24	Control Manual	13/05/17	16/05/17	4d
25	Funcionamiento real	17/05/17	25/05/17	9d
26	PEC 3 - Seguimiento	19/04/17	16/05/17	28d
27	**Fase 4 - Extras**	**14/05/17**	**06/06/17**	**24d**
28	Modo autónomo	14/05/17	27/05/17	14d
29	Aplicación Web	28/05/17	29/05/17	2d
30	Gráficos	28/05/17	29/05/17	2d
31	Envío de e-mails	30/05/17	02/06/17	4d
32	Código Final	14/05/17	06/06/17	24d
33	**Memoria del proyecto**	**30/04/17**	**18/06/17**	**50d**
34	**Presentación Final**	**13/06/17**	**25/06/17**	**13d**
35	**Presentación On-Line**	**27/06/17**	**27/06/17**	**1d**
36				

Figura 50: Diagrama de Gannt Inicial, parte 1.

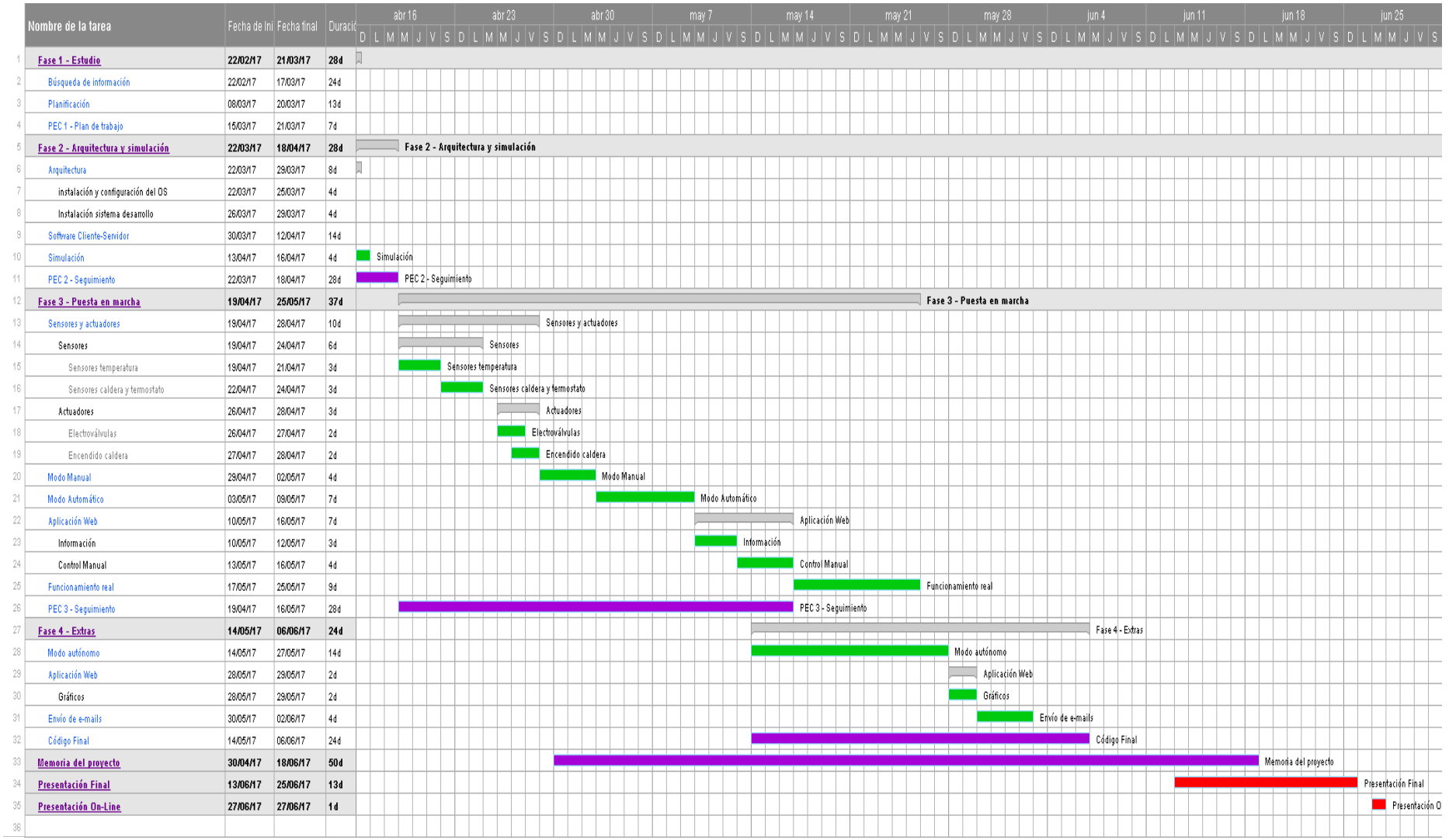


Figura 51: Diagrama de Gannt Inicial, parte 2.

[2] Final

	Nombre de la tarea	Fecha de Ini	Fecha final	% Comp	Duració
1	**Fase 1 - Estudio**	**22/02/17**	**21/03/17**	**100%**	**28d**
2	Búsqueda de información	22/02/17	17/03/17	100%	24d
3	Planificación	08/03/17	20/03/17	100%	13d
4	PEC 1 - Plan de trabajo	15/03/17	21/03/17	100%	7d
5	**Fase 2 - Arquitectura y simulación**	**22/03/17**	**18/04/17**	**100%**	**28d**
6	Arquitectura	22/03/17	29/03/17	100%	8d
7	Software Cliente-Servidor	30/03/17	13/04/17	100%	15d
8	Modo Manual	02/04/17	07/04/17	100%	6d
9	Modo Automático	06/04/17	14/04/17	100%	9d
10	Simulación	14/04/17	17/04/17	100%	4d
11	PEC 2 - Seguimiento	22/03/17	18/04/17	100%	28d
12	**Fase 3 - Puesta en marcha**	**19/04/17**	**16/05/17**	**100%**	**28d**
13	Simulación	19/04/17	25/04/17	100%	7d
14	Sensores y actuadores	26/04/17	11/05/17	100%	16d
15	Sensores	26/04/17	01/05/17	100%	6d
16	Sensores temperatura	26/04/17	28/04/17	100%	3d
17	Sensores caldera y termostato	29/04/17	01/05/17	100%	3d
18	Actuadores	01/05/17	11/05/17	100%	11d
19	Electroválvulas	01/05/17	04/05/17	100%	4d
20	Encendido caldera	08/05/17	11/05/17	100%	4d
21	Aplicación Web	28/04/17	14/05/17	100%	17d
22	Información	28/04/17	30/04/17	100%	3d
23	Control	01/05/17	04/05/17	100%	4d
24	Gráficos	12/05/17	14/05/17	100%	3d
25	Ampliaciones	07/05/17	12/05/17	100%	6d
26	Base de datos	07/05/17	10/05/17	100%	4d
27	Envío de e-mails	11/05/17	12/05/17	100%	2d
28	PEC 3 - Seguimiento	19/04/17	16/05/17	100%	28d
29	**Fase 4 - Extras**	**14/05/17**	**06/06/17**	**100%**	**24d**
30	Modo autónomo	14/05/17	27/05/17	100%	14d
31	Funcionamiento real	20/05/17	28/05/17	100%	9d
32	Código Final	14/05/17	06/06/17	100%	24d
33	**Memoria del proyecto**	**30/04/17**	**18/06/17**	**80%**	**50d**
34	**Presentación Final**	**13/06/17**	**25/06/17**	**20%**	**13d**
35	**Presentación On-Line**	**27/06/17**	**27/06/17**	**0%**	**1d**
36					

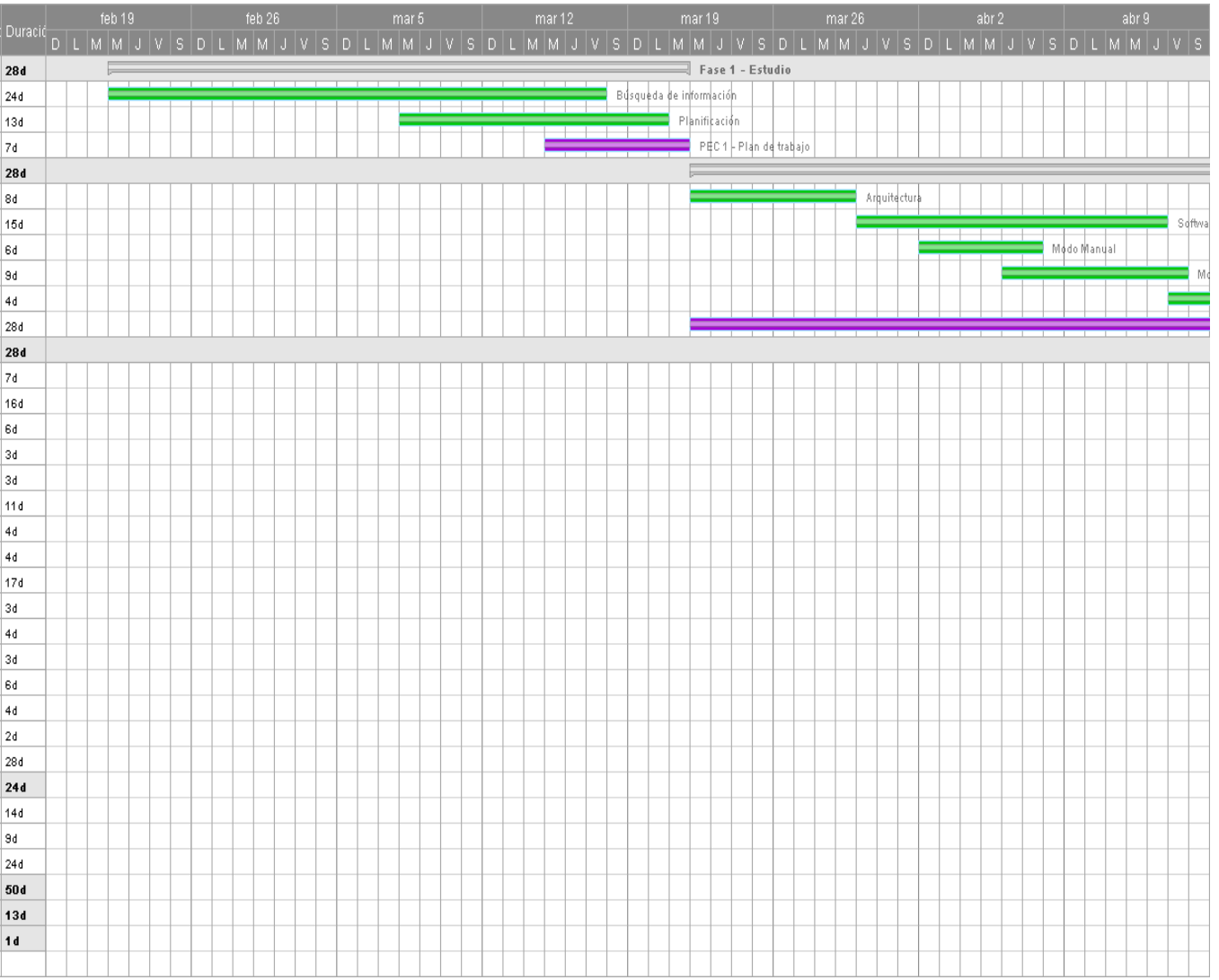


Figura 52: Diagrama de Gannt Final, parte 1.

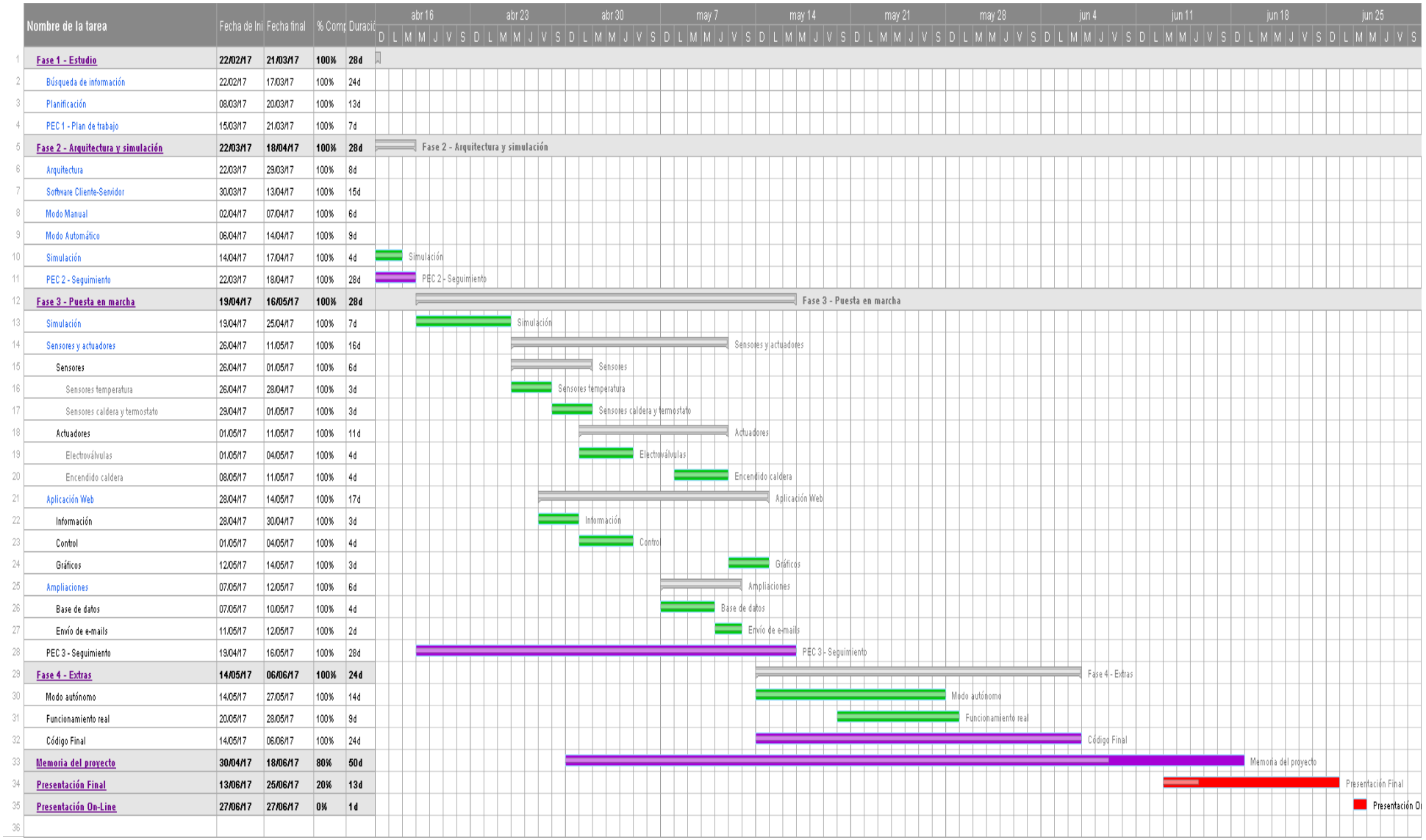


Figura 53: Diagrama de Gannt Final, parte 2.