Grado en Ingeniería Telemática

TRABAJO FIN DE GRADO

Diseño e implementación de un sistema de control centralizado para climatización

Autor: Carlos Gómez Garrido

Tutor: Higinio Rubio Alonso

Leganés, Junio de 2015

Título: **Diseño e implementación de un sistema de control centralizado para climatización.**

Autor: Carlos Gómez Garrido

Tutor: Higinio Rubio Alonso

EL TRIBUNAL

Presidente: Marcelo Gabriel Bagnulo Braun

Vocal: Raquel Aparicio Morenilla

Secretario: Pedro Contreras Lallana

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día 10 de Julio de 2015 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

RESUMEN

El fin principal de la domótica no es otro que el de ofrecer un mayor nivel de confort, seguridad y ahorro al usuario en su vivienda. Los sistemas domóticos son caros, sin embargo, en la actualidad han surgido diversas tecnologías, como la plataforma Arduino, que pertenecen a lo que conocemos como hardware y software libre, a las que cualquiera puede acceder y que han permitido una reducción de los costes de un sistema domótico. Por tanto, en este momento resulta indispensable poder ampliar el uso de la domótica más allá de los restringidos usos anteriores, para llegar hasta cualquier edificio de nuestra vida cotidiana.

El objetivo principal del presente proyecto es el de diseñar un sistema de control domótico de bajo coste, basado en una plataforma de hardware libre y software libre, como Arduino, que se adapte a las características de un sistema de climatización, es decir, a un sistema cotidiano. Con el fin de alcanzar dicho objetivo, se ha diseñado e implementado un sistema de control para un sistema de climatización existente, adaptándose a sus características de funcionamiento reales.

Para llevarlo a cabo, se han evaluado diferentes plataformas de hardware libre, que puedan cumplir la funcionalidad requerida por el sistema convencional a controlar en el edificio, para proponer una arquitectura sencilla y robusta. La arquitectura, a su vez debe permitir la comunicación a partir de la red local Ethernet del propio edificio, empleando un protocolo con proyección de futuro para las tecnologías domóticas, flexible y fiable que permita automatizar de la manera deseada el sistema a controlar en nuestro edificio. Además, se valorarán diferentes sensores de temperatura y de humedad para monitorizar el comportamiento del sistema de climatización y ejecutar diferentes acciones a través de actuadores para controlar su comportamiento.

El proyecto concluye con el desarrollo de una aplicación web, accesible desde el navegador del usuario de su propia red local, con el fin de integrar una interfaz central de usuario, para poder controlar y visualizar los datos del sistema domótico.

ABSTRACT

Home automation main goal is to offer a better feeling of security and comfort, together with less energy used, to any user in their homes. The main issue is that home automation systems are expensive. However, new technologies, like the Arduino platform, which belong to what is known as open hardware and software have appeared. This technologies are low-cost and open to anyone interested. Home automation systems can take advantage of those technologies, resulting in a cost reduction. Therefore, now it is essential to expand automation systems to any other building from our daily lives.

The main purpose of this project is to design a low-cost automation system, based on an open hardware and software platform, like the Arduino platform, which can be adapted to the characteristics of a regular system, like a heating and air cooling system. With that purpose in mind, we have designed and implemented a control system for an existing heating and air cooling system, according to its behaviour characteristics.

In order to complete our task, first we have evaluated some open hardware electronics platform, which can fulfil the required functionalities by the regular system to control in our building, so we can propose a simple but robust automation architecture. In addition, the archictecture should allow its nodes to communicate using the local Ethernet network of the building where it is located, using a communication protocol with a bright future in automation systems, which is reliable and flexible in order to automate according to the user preferences the system to control. Moreover, we will consider different temperature and humidity sensors to monitor the performance of the heating and air cooling system and execute different orders in several actuators to control its behaviour.

Finally, our project will be completed after developing a web application, reachable from a web browser in the local Ethernet network in the building. The purpose of the web application is to offer the user a centralised interface, where the automation system can be controlled and the user can see its data.

Índice General

Índice de Figuras

Índice de Tablas

Capítulo 1: Introduction

1.1 Motivation

Home automation systems is one of the technologies with a promising future ahead. However, we should not limit it's scope to people's homes.

Over the recent years, there have appeared several new technologies which have brighten automation systems future. For example, there is an open hardware prototyping platform, called Arduino, based on a microcontroller whose aim is to allow as many people as posible, with or without electronics knowledge, to design their own electronics project, like home automation, in an easy way. It can be done since Arduino's programming language is open software and easy to learn and understand.

Arduino has a community with multiple users that offers a lot of support and many open software already tested and ready to use in our projects. As a open prototyping platform it has different electronic boards, but each of them with a low cost, so it allows everyone to minimize the cost to develop an automation system. Other technologies that have appeared are credit-card size, low cost computers, like Raspberry Pi. This low-cost computers offers us a way to minimize the cost when using an central element to manage our automation system and coordinate its devices.

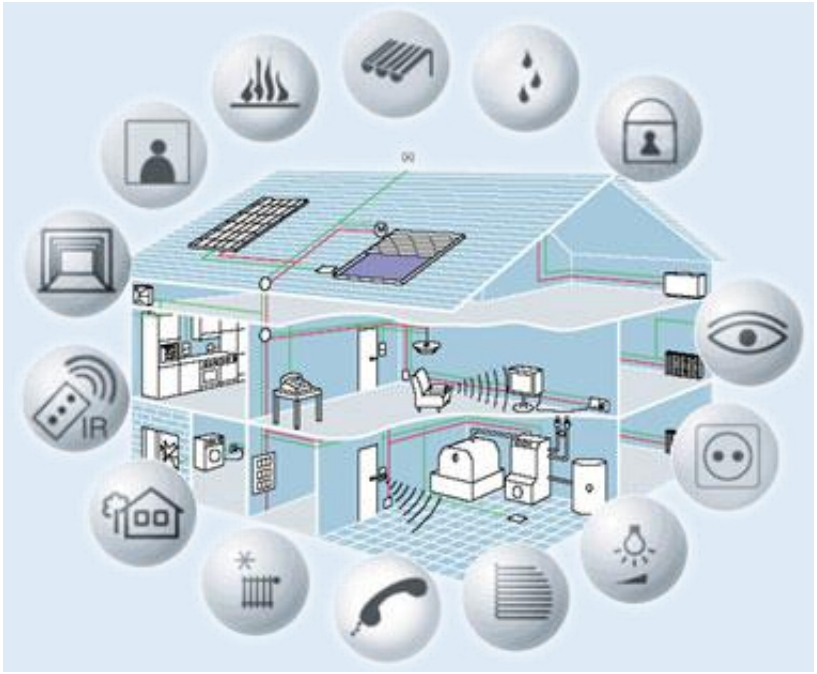


Figura 1.1 *Incorporación de un sistema domótico en el hogar.*

On the other hand, almost every building has a local Ethernet network installed. So it would be great if we can take advantage of that and use it for the communication among the components of the architecture used in an automation system. In addition, there is an extra motivation by doing this project in order to use a communication protocol, over the local Ethernet network, which provides us a flexible and fiable exchange of messages, whose format can be defined by the automation system we want to develop and has a bright future in automation systems.

Finally, we want to offer home automation the opportunity to extend its scope into more buildings, like enterprises or schools, which people use almost every day. Thanks to all this new technologies, which allows home automation to have a lower cost, now it is the right time to do so. Moreover, another motivation for this project is to create a guide for other people that want to include an automation system for increasing their comfort, in any type of building, where the automation system should have a low cost and behave according to the characteristics of the system to control.

1.2 Goals

The **main goal** of this project is to design and develop a low-cost automation system, based on the real characteristics of a heating and air cooling system, already in place in a building. An automation system, whose components belong to what it's known as open hardware and software, communicating over a local Ethernet network. A design and implementation that could be integrated in a building in the future, since the scope does not include its integration.

In order to achieve this, what we are going to do is to first study how the heating and air cooling system behaves for determining the strategies to control it. What follows next is designing a viable architecture according to its characteristics. Finally, we are going to choose its hardware components, where we can develop the software of our automation system and whose communication is based on a promising protocol for automation system over the local Ethernet network of the building. Our system can also interact with the user across a web application. In addition, we can find some secondary goals in our project:

1. Learn from the different low cost open hardware prototyping technologies, like Arduino, which we can use as control unit in our automation system. And low-cost computers, which we can use as the central element of our system.
2. Design a control system that can send different orders to its actuators according to the data given by some temperature sensors to control a heating and air cooling system.
3. Design a centralised architecture, where the central element can store data about the performance of each single element controlling the heating and air cooling system and where the user can interact with the control elements.
4. Learn from different open source communication protocols that can be used in automation systems and use one to implement the communication using the local Ethernet network in our system.

5. Desing and implement a web application that can be used as user interface for our system.
6. Design and develop an automation system that can serve as a basic guide to anyone that wants to develop a system, whose control procedure is adapted to the characteristics of a regular system in a building, available to people with capabilities to develop this kind of project, who are also trying to minimaze the cost to implement it.

1.3 Document organization

In this section we will present the organisation of this document, which is:

- **Chapter 1: Introduction**. In this chapter we will present the motivation of our project and its main and secondary goals. Finally, we will present the organisation of the whole document.
- **Chapter 2: State of technology**. In this chapter we will cover several technologies that surround our project, in order to let the reader know everything that is necessary to understand our project. First, we will cover automation and control systems. Secondly, we will see different electronics prototyping platforms and temperature and humidity sensors. Finally, we will talk about low-cost computers.
- **Chapter 3: Building complex description**. In this chapter we will the building complex where our automation system is going to be located and how the heating and air cooling system to control works.
- **Chapter 4: Control strategies analysis**. In this chapter we will see the strategies propose to control the existing heating and air cooling system, according to its characteristics, in order to develop our automation system.
- **Chapter 5: Methodology**. In this chapter we will cover some methodologies that can be applied to organise our project. Secondly, we will choose one among them as a guideline of the different tasks in our proyect. Finally, we will conclude presenting the organisation which we have followed and the requirements that our automation system should fulfil.
- **Chapter 6: Design of the automation system architecture**. In this chapter we will first introduce the automation architecture that we have designed. Secondly, we will describe how the communication protocol which we have selected works, in order to introduce the reader the architecture that the nodes of our system are using to communicate. In addition, we will present the format of the messages interchanged using the procol and the format of the tables in our database to store information about our automation system. Finally, we will describe the structure web application that we will develop as the user interface.
- **Chapter 7: Hardware components**. In this chapter we will describe the different hardware components selected in our design of the automation system.

- **Chapter 8: Software development**. In this chapter we will describe every detail of the software that we have developed to complete our design of the automation system.
- **Chapter 10: Performance tests**. In this chapter we will complete some tests in the automation system developed to check its overall performance.
- **Chapter 10: Budget**. In this chapter we will show the complete budget to develop our project.
- **Chapter 11: Conclusions**. In this chapter we will see the conclusions after finishing our project and some future work that would be interesting to do.
- **Chapter 12: References**. In this chapter we will show the bibliographic and web references used to write this document.

Finally, we will include one annex. This annex will be an extended English summary of the whole document

Capítulo 2: Estado de la tecnología

A lo largo de este capítulo vamos a situar el contexto en el que se encuentra nuestro proyecto. Primero, introduciremos al lector en el concepto de domótica, detallando sus puntos más importantes y los tipos de arquitectura presentes en los sistemas domóticos. Después, vamos a explicar los sistemas de control que pueden emplearse en aplicaciones domóticas para después pasar a explicar los diferentes elementos que los componen. Estos elementos suelen ser una plataforma electrónica de control, sensores y actuadores. Además, describiremos al lector diferentes ordenadores de bajo coste que pueden ser empleados para coordinar cada uno de los sistemas de control incorporados a dicha aplicación domótica. El objetivo de cada explicación será el de introducir al lector en los diferentes dispositivos y tecnologías que comprenden nuestro proyecto.

2.1 Domótica e inmótica

En la actualidad, el término **domótica** es ampliamente utilizado. Sin embargo, no siempre de forma correcta, ya que en la mayoría de los casos se emplea para indicar cualquier tipo de automatización. La palabra domótica proviene de la unión de la palabra “domo”, que proviene del latín “domus”(casa) y del sufijo “tica”, que según qué autor afirmará que proviene de la palabra automática o que está a su vez formado por “tic”, de tecnologías de la información y “a” de automatización.

El objetivo principal de la domótica es el de proporcionar a los usuarios de una vivienda un aumento del grado de confort, seguridad, ahorro energético y facilidades de comunicación. En concreto, podemos entender a la domótica como el conjunto de técnicas utilizadas, basadas en un conjunto de automatismos o dispositivos inteligentes, gracias a la incorporación de microcontroladores, para garantizar la automatización de la gestión y la comunicación de las viviendas unifamiliares.

Por otro lado, el término **inmótica** se refiere a la gestión técnica de grandes edificios, por ejemplo, hoteles, museos, bloques de aulas, etc. A diferencia de la domótica, más orientada a viviendas unifamiliares, la inmótica se orienta hacia edificios de mayor tamaño, con unos fines diferentes que se orientan no solo a mejorar la calidad de vida, sino que también hacia la mejora de la calidad del trabajo. Por ello, la inmótica empleará las mismas técnicas de automatización que la domótica pero centradas en los sistemas de automatización que desea incorporar.

2.1.1 Arquitectura de sistemas domóticos

La arquitectura de un sistema domótico o inmótico describe las distintas maneras de ubicar a los elementos de control del sistema. Las tres arquitecturas más usadas son:

Arquitectura centralizada. En esta arquitectura, figura 2.1, el conjunto de elementos que se van a controlar y supervisar, es decir, luces, sensores, válvulas… han de cablearse hasta el sistema de control central, formando una red en estrella, ya que toda la comunicación se realiza a través del centro de control. En este caso, toda la información recopilada por los sensores se envía al controlador central para que tome las decisiones oportunas y se las comunique a los actuadores. El controlador central es el corazón del sistema, ante cuyo fallo todo el sistema se viene abajo, aunque su despliegue es más sencillo.

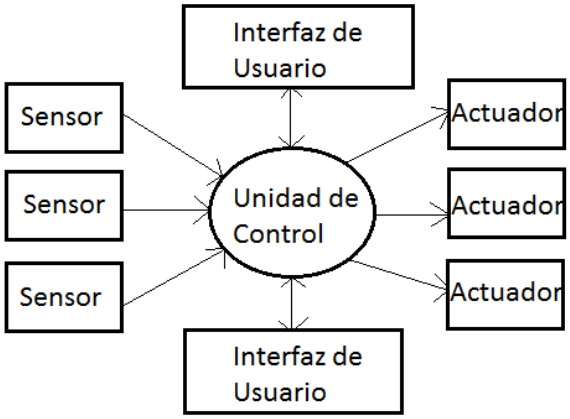


Figura 2.1 *Arquitectura de un sistema domótico centralizado.*

Arquitectura descentralizada. Esta arquitectura, figura 2.2, es la opuesta a la arquitectura vista en el párrafo anterior. En esta arquitectura, todos los elementos del sistema disponen de inteligencia, siendo totalmente independientes, es decir, los sensores y actuadores son a su vez controladores. El sistema debe de proporcionar un bus compartido entre los elementos básicos que permita a todos comunicarse, en función de su configuración. Los sensores serán capaces de transmitir la información recibida y los actuadores serán capaces de recibir las órdenes a través de este bus.

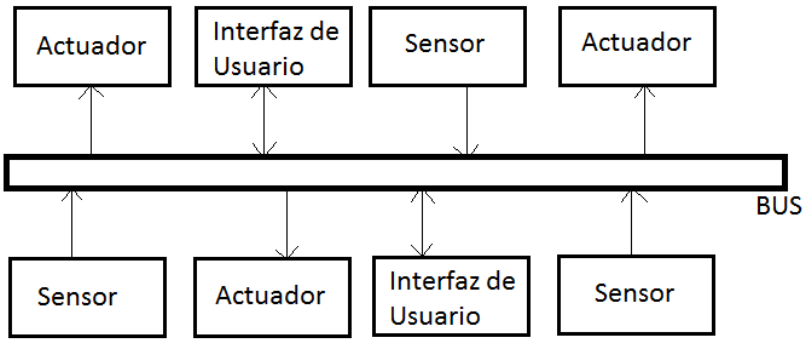


Figura 2.2 *Arquitectura de un sistema domótico descentralizado.*

Arquitectura distribuida. Esta arquitectura, figura 2.3, es optimizar las dos anteriores con la idea de acercar el elemento de control hacia el elemento a controlar. En este caso, ya no existe un elemento de control único encargado de gobernar todo el sistema, sino que existen varios elementos que se reparten las tareas de control. Estos elementos reciben el nombre de nodos, cada uno dispone de sus sensores y actuadores, es decir, a ellos se conectan los

elementos básicos. Además, estos controladores o nodos estarán conectados entre sí a través de un bus de datos compartido, que permite la comunicación entre ellos. La ventaja principal que presenta esta arquitectura es la robustez en cuanto al funcionamiento, ya que un fallo en un controlador no afecta al resto. Sin embargo, sufre el inconveniente ser más costosa y complicada de desplegar.

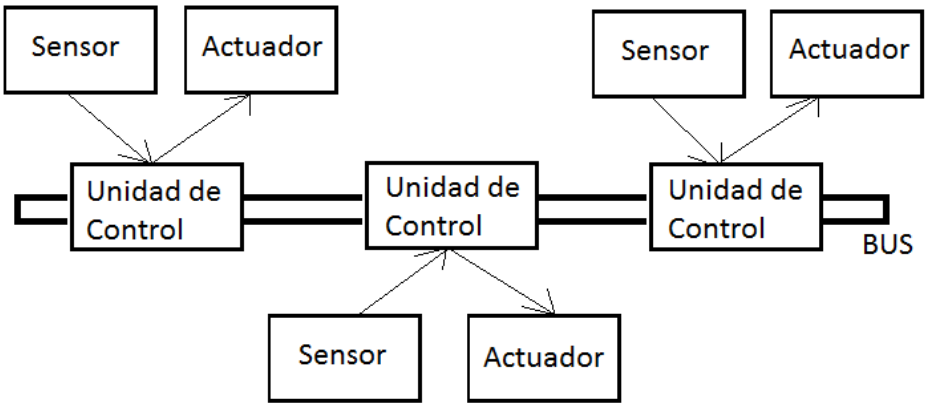


Figura 2.3 *Arquitectura de un sistema domótico distribuido.*

Sin embargo, siempre podemos crear una arquitectura mixta, que combine varias de las tres anteriores, para lograr un sistema domótico que se adapte en mayor grado a nuestras necesidades.

2.2 Sistemas de control

Denominaremos **sistema de control** al conjunto de dispositivos encargados de coordinar, dirigir y regular el comportamiento de otro sistema, para obtener los resultados esperados y reducir la probabilidad de fallo.

En el contexto en el que se sitúa nuestro proyecto, cada sistema de control se encargará controlar un determinado sistema de climatización. Dicho sistema de control de contará con los siguientes elementos:

- Uno o varios **sensores** de temperatura, encargados de realizar las mediciones correspondientes a cada bloque de climatización.
- Un **controlador**, que empleará los datos obtenidos por los sensores para determinar la acción de salida del sistema.
- Uno o varios **actuadores**, encargados de ejecutar la orden de salida o control por parte del controlador.

Por otro lado, en función de la estrategia de control que deseemos implementar, podemos distinguir dos tipos de sistemas de control:

- **Sistemas de lazo abierto** o sistemas sin realimentación, en los que el controlador determinará la acción de salida únicamente en función de la entrada al sistema de control. En este caso, la salida no tiene efecto sobre el

sistema de control. La mayor parte de estos sistemas serán automatismos, muy limitados para tomar decisiones inteligentes, al no tener forma de conocer los resultados de su decisión anterior. En la figura 2.4 podemos visualizar un ejemplo de sistema de control de lazo abierto.

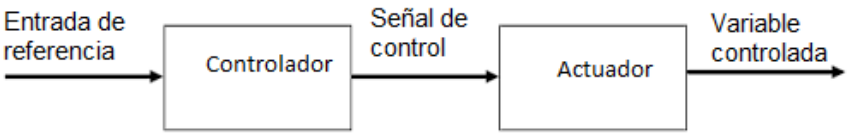


Figura 2.4 *Esquema de un sistema de control de lazo abierto.*

- **Sistemas de lazo cerrado** o sistemas retroalimentados, en los que la señal de salida realimenta al sistema de control a través de los sensores. En este caso, el controlador determinará la acción de salida en función del error, el cual es la diferencia entre la señal de entrada y la de salida (realimentada). Este sistema es más flexible y será capaz de reaccionar ante resultados esperados. Podemos ver su esquema en la figura 2.5.

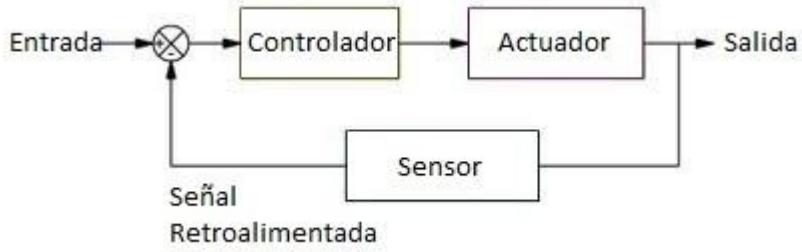


Figura 2.5 *Esquema de un sistema de control de lazo cerrado.*

En la mayoría de los sistemas de control se van a emplear un sistema de lazo cerrado. Gracias al dibujo anterior, podemos identificar en ellos a los siguientes elementos:

- Una **señal de entrada** que se corresponderá con la temperatura deseada o de consigna para cada uno de los bloques de climatización anteriores.
- Una **señal de salida** o variable controlada, que será la temperatura real a la que se encuentra nuestro sistema de climatización a controlar, en un determinado momento.
- Un **sensor** encargado de medir la temperatura a la que se encuentra nuestro sistema a controlar en un determinado momento para enviar la señal realimentada a nuestro controlador.
- Un **controlador** que se encargará de tomar la decisión adecuada para el correcto funcionamiento del sistema. La decisión variará en función de la señal de error que reciba el controlador. Esta señal de error será la diferencia entre la señal de entrada y la de salida.
- Un **actuador** que se encargará de ejecutar la orden del controlador. En nuestro caso, se corresponderá con los servomotores asociados a las válvulas mezcladoras de cada uno de los bloques de climatización.

A continuación, en los próximos apartados, describiremos los dos tipos de sistemas de control de lazo cerrado más comunes. El primero será un sistema de control de tipo On/Off y el segundo de tipo PID. En función del grado de control requerido en el en sistema a controlar se suele emplear uno u otro.

2.2.1 Sistema de control ON/OFF

El sistema de control por lazo cerrado más sencillo que podemos encontrar se trata del sistema de control ON/OFF. Este sistema de control se corresponde con los termostatos tradicionales que podemos encontrar en casa, sólo ofrecerá como salida dos estados posibles a su bloque de climatización, un estado completamente encendido (100% ON) o completamente apagado (100% OFF). Un estado será usado cuando la temperatura medida en cada bloque se encuentre por debajo de la temperatura de consigna y el otro cuando la temperatura medida se encuentre por encima de la temperatura de consigna.

Por ejemplo, en invierno, cuando nuestros bloques de climatización ofrezcan calor, si la temperatura medida es inferior a la de consigna, el estado de salida al bloque será 100% ON, mientras que si la temperatura medida es superior a la de consigna, el estado será 100% OFF. Sin embargo, en verano, los estados de salida para cada bloque sucederán a la inversa. A partir del siguiente dibujo podemos comprender en mayor detalle su funcionamiento:

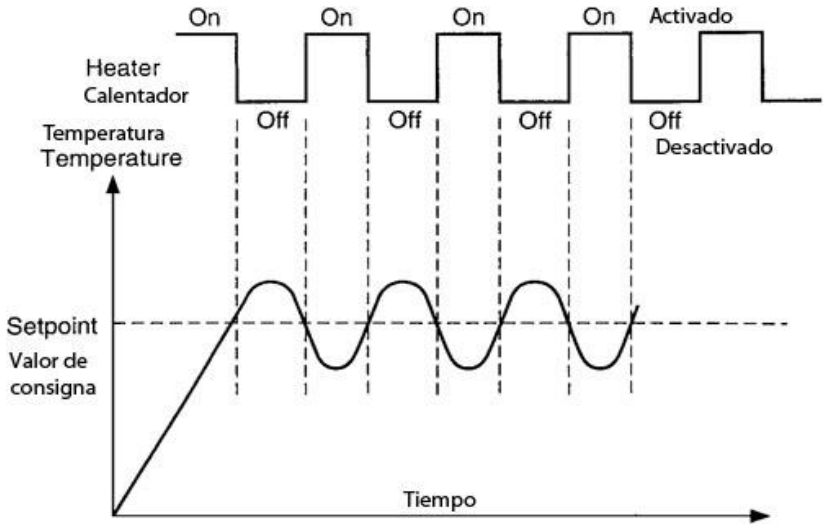


Figura 2.6 *Gráfica de rendimiento de un sistema de control ON/OFF a lo largo del tiempo, fuente [52].*

Como podemos observar en el dibujo, el sistema de control de tipo ON/OFF presenta dos problemas:

- El primer problema será que la temperatura en cada bloque va a oscilar, ya que la temperatura medida deberá cruzar el punto de consigna para que el controlador actúe.
- El segundo problema será la aparición de “overshoot” y “undershoot”. El “overshoot” será la magnitud en que la temperatura medida rebasa al punto de consigna hasta que el controlador actúa. El “undershoot” será lo contrario.

A la hora del diseño de estos sistemas de control, se va a tratar de reducir en lo posible tanto esta oscilación como el "overshoot" y "undershoot". Para ello, se suele definir un rango de histéresis, que medirá la sensibilidad del controlador. Es decir, el controlador sólo actuará cuando la diferencia entre la temperatura medida en cada bloque y la temperatura de consigna sea superior al rango de histéresis definido.

2.2.2 Sistema de control PID

Un sistema de control proporcional, integral y derivativo, es decir, un control PID aparece en situaciones en las que se requiere un sistema de control más preciso. Al igual que el sistema On/Off, se trata de un sistema de control por lazo cerrado. Sin embargo, el sistema de control PID implementa un algoritmo que combina la acción de tres tipos de control, un control proporcional en el tiempo, un control integral y un control derivativo. Cada uno de ellos proporciona una mayor precisión al controlador PID. En la figura 2.7, podemos visualizar el esquema de un sistema de control PID.

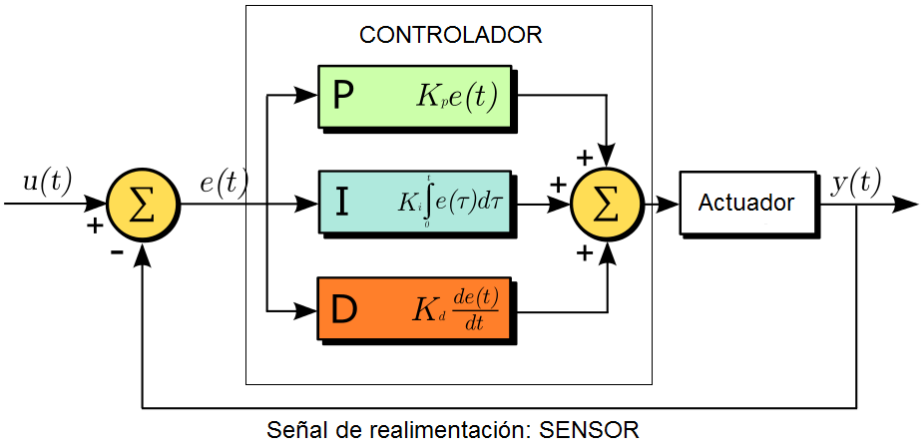


Figura 2.7 *Esquema de los elementos de un controlador de tipo PID.*

El objetivo principal del controlador PID será el de reducir el error (señal e(t)) a cero en estado estacionario, es decir, la diferencia entre la temperatura de consigna (señal de entrada u(t)) y la temperatura medida en el bloque (señal de salida y(t)). Además tratará de solucionar los dos problemas que aparecen con los sistemas de control de tipo On/Off, es decir, la aparición del "overshoot" y "undershoot" y la oscilación de la temperatura medida en el bloque de climatización a controlar.

Como mencionamos anteriormente y podemos observar en el dibujo, el controlador PID combina la acción de tres controladores [55]. Estos controladores serán los siguientes:

1. En el **sistema de control proporcional** la salida del controlador es proporcional a la señal de error, es decir, entre la temperatura medida en el bloque y la temperatura de consigna. Por tanto, la respuesta de este control será instantánea ante cualquier variación del error. Como podemos ver en la siguiente fórmula:

$$\boldsymbol{Y(t) = Kp * e(t)}$$

Como podemos observar en la fórmula, la señal de salida será el resultado del producto entre la señal de error y la banda o constante proporcional Kp. Definiremos a esta banda proporcional al lugar por debajo del punto de consigna en el que el control proporcional actuará. Por tanto, a medida que aumentemos el valor de Kp, aumentaremos la respuesta del sistema y reduciremos el error en estado estacionario. Sin embargo, existe un límite para el aumento de la constante proporcional. A medida que aumentamos el valor de Kp, se producirá un mayor "overshoot", es decir, habrá una mayor sobre oscilación. Además, en caso de escoger un valor pequeño para Kp, el sistema oscilará como en el caso del control On/Off. En la figura 2.8 podemos comprobar el efecto de la variación de Kp en un controlador PID.

2. En el **sistema de control integral** la salida del controlador es proporcional al error acumulado a lo largo del tiempo. El objetivo del control integral será el de reducir y eliminar el pequeño error tras el control proporcional. Para ello, el control integral seguirá la siguiente fórmula:

$$\boldsymbol{Y(t) = Ki * \int_0^t e(t)dt}$$

Como podemos ver la acción de control será resultado del producto entre la constante integral Ki por un sumatorio del error obtenido en un período determinado. Debido a esta integración, hasta el menor error posible causará un pequeño incremento en la señal de salida del control integral. Por tanto, la señal de salida de control aumentará hasta que dicho error sea cero. Como hemos visto, este control nos ofrecerá una mayor estabilidad a la respuesta del sistema tras añadirlo al control proporcional. Sin embargo, a medida que aumentamos el valor de la constante proporcional Ki, la respuesta del sistema se hará más oscilatoria, pudiendo llegar a desestabilizarse. En la figura 2.9 podemos comprobar el efecto de la variación de Ki en un controlador PID.

3. El objetivo del **control derivativo** será el de mejorar la estabilidad del sistema, corrigiendo el posible "overshoot" y "undershoot" aún presente tras la acción proporcional e integral. Como vimos anteriormente, se va a tardar un cierto tiempo en que la acción el controlador se note a la salida del sistema. La acción derivativa tratará de anticiparse a este cambio, se basa en introducir una acción de predicción sobre la señal de error. La fórmula que sigue esta acción será:

$$\boldsymbol{Y(t) = Kd * (\partial e(t)/\partial t)}$$

Como podemos ver en la fórmula, al obtener la derivada del error en un período de tiempo, nos indica el cambio instantáneo de la acción de control. Si incrementamos el valor de la constante de derivación Kd, nuestro sistema tendrá una reacción más veloz ante cambios en la señal de error, pudiendo llegar a reducirlo antes de que se haga demasiado grande en sistemas como nuestros bloques de climatización, con una gran inercia. Sin embargo, normalmente se usa un valor para la constante Kd pequeño, ya que la acción derivativa es sensible al ruido. De tal forma que habrá que escoger su valor con cuidado, ya que si la señal realimentada por el sensor introduce ruido y el sistema de control tiene un tiempo de respuesta alto, un valor alto de Kd puede

saturar o desestabilizar el sistema. En la figura 2.10 podemos comprobar el efecto de la variación de Kd en un controlador PID.

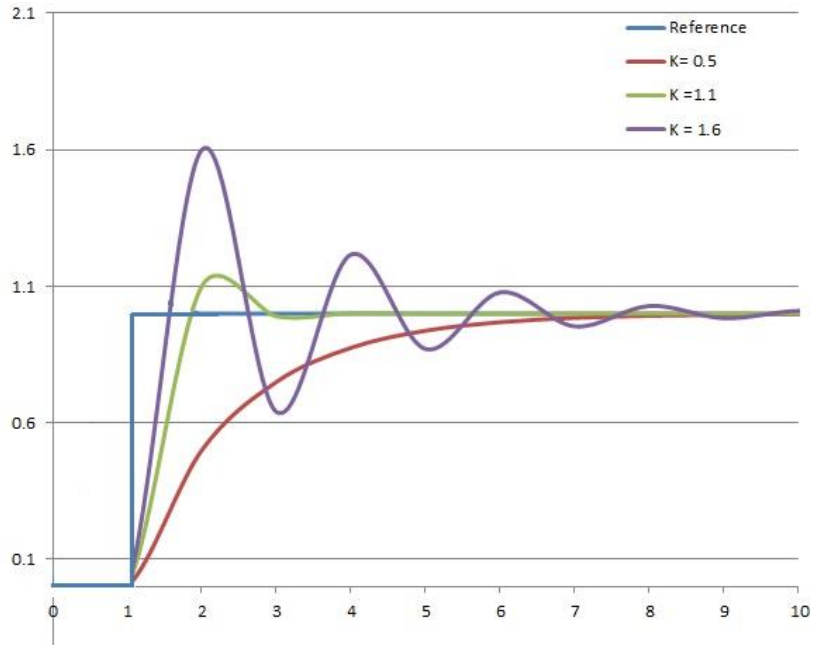


Figura 2.8 *Gráfica del rendimiento de un sistema de control proporcional en función del valor de la constante proporcional Kp a lo largo del tiempo.*

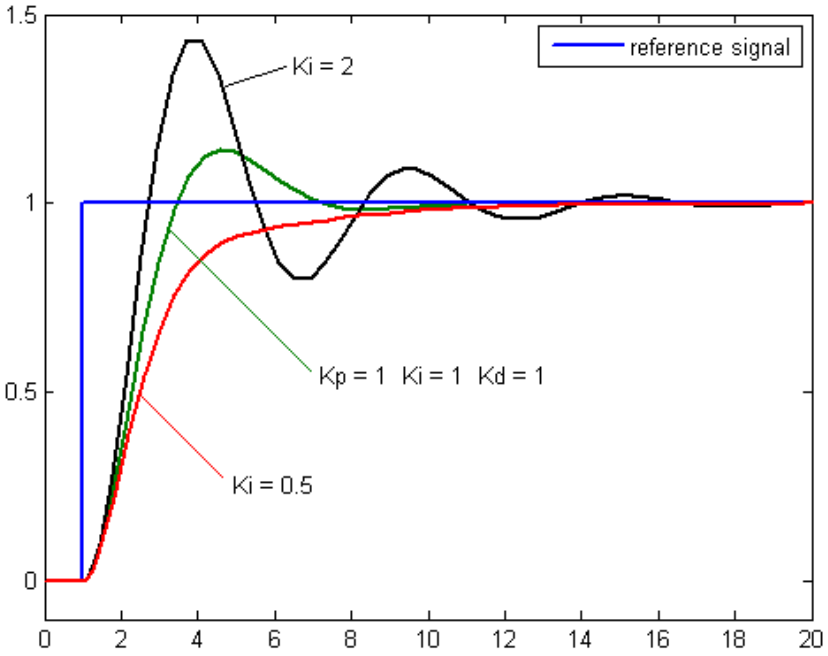


Figura 2.9 *Gráfica del rendimiento de un sistema de control integral en función del valor de la constante proporcional Ki a lo largo del tiempo.*

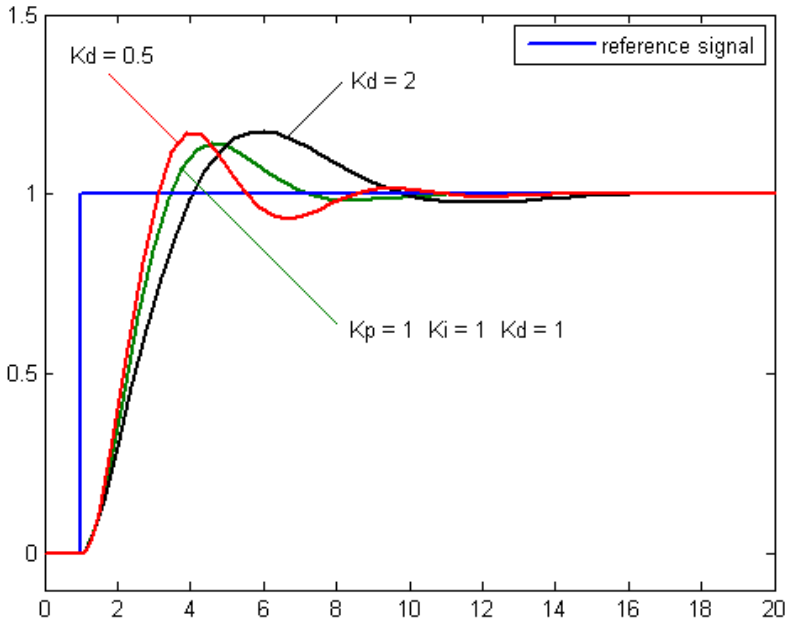


Figura 2.10 *Gráfica del rendimiento de un sistema de control derivativo en función del valor de la constante proporcional Kd a lo largo del tiempo.*

Como hemos visto, la acción conjunta de la acción proporcional, integral y derivativa forman el controlador PID. Este controlador es el más usado para cualquier aplicación de control industrial, ya que el grado de control que nos ofrece es bastante elevado y suficiente para la mayoría de aplicaciones. En la figura 2.11 podemos ver la fórmula que sigue el controlador PID.

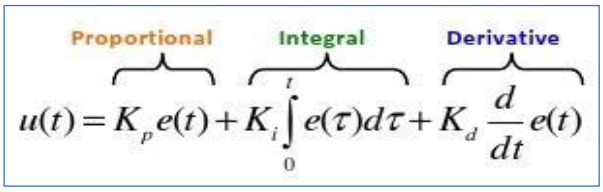


Figura 2.11 *Función de salida de un controlador PID.*

Sin embargo, como vimos en los párrafos correspondientes a la acción proporcional, integral y derivativa, hay que elegir cuidadosamente los valores de las constantes Kp, Ki y Kd para que el sistema de control funcione adecuadamente. Para ello, se realiza un proceso de sintonización previo a la puesta en marcha. En el siguiente apartado, describiremos los dos métodos más comunes de sintonización para los controladores PID.

2.2.3 Sintonización del controlador PID

Como concluimos en el apartado anterior, para conseguir que el sistema de control responda de manera adecuada, será necesario elegir adecuadamente los valores de las constantes Kp, Ki y Kd.

El **método de Ziegler-Nichols** [56], permite ajustar de manera empírica, o sintonizar, un controlador PID, antes de su puesta en marcha, sin necesidad de conocer las ecuaciones del sistema a controlar. En 1942, Ziegler y Nichols propusieron unas reglas de ajuste para

conseguir que la respuesta de la mayoría de los sistemas de control realimentados tenga un valor con buenas características de actuación y estabilidad. Por ello, se trata de uno de los métodos más usados.

Ziegler y Nichols definieron dos reglas de sintonización para determinar los valores de las constantes Kp, Ki y Kd. En concreto, una regla será para sistemas cuya respuesta se adapta mejor a un sistema de lazo abierto y otra regla para aquellos sistemas que se ajustan mejor a un sistema de lazo cerrado.

1. Primer método: sintonización por la respuesta al escalón

El **primer método** será la sintonización por la respuesta al escalón. Este método se empleará en sistemas cuya respuesta se adapta bien a un sistema de lazo abierto y que sufren de un tiempo de retardo amplio entre la orden emitida por el controlador y su ejecución por parte del actuador. Para hallar los valores de las constantes, primero se retirará el controlador PID y en su lugar se colocará una señal escalón que se aplicará en el actuador. Tras esto, se procederá a registrar la respuesta de la señal de salida del sistema de control, la cual será una curva con forma de S:

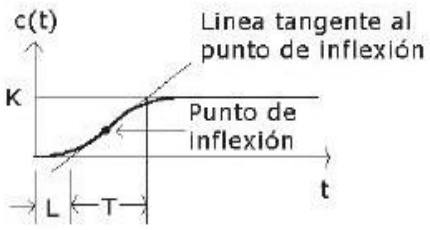


Figura 2.12 *Curva de la respuesta del sistema ante una función de entrada en escalón.*

Una vez obtenida esta respuesta, podemos hallar los dos parámetros que caracterizan dicha curva, el tiempo de retardo L y la constante de tiempo T. Para ello, se va a dibujar una recta tangente en el punto de inflexión de la curva, para después determinar las intersecciones de dicha tangente con el ejes del tiempo y $c(t) = K$. Con estos datos, podemos aproximar el sistema a controlar como un sistema de primer orden más un retardo, con la siguiente función de transferencia:

$$\boldsymbol{K * e\text{^}Ls/(Ts + 1)}$$

Aplicando los valores obtenidos en la curva de respuesta, según la tabla 2.1, determinaremos valores de las constantes Kp, Ki y Kd:

Controlador	Kp	$Ti = \frac{1}{Ki}$	$Td = \frac{1}{Kd}$
P	$\frac{T}{K * L}$	∞	0
PI	$0.9 * \frac{T}{K * L}$	$\frac{T}{0.3}$	0
PID	$1.2 * \frac{T}{K * L}$	$2 * L$	$0.5 * L$

Tabla 2.1 Valores para sintonizar las constantes del controlador en función de los datos obtenidos tras respuesta a escalón.

2. Segundo método: ganancia crítica

El **segundo método**, no requerirá retirar el controlador PID, ya que se realizará sobre un sistema de lazo cerrado. Inicialmente, reduciremos al mínimo las acciones integral y derivativa del controlador PID mientras que iremos incrementando gradualmente el valor de la constante proporcional hasta que la respuesta del sistema oscile de forma continua y lineal, de amplitud constante. En dicho momento, mediremos el valor de la ganancia proporcional, llamada ganancia crítica Kc y el período de oscilación Tc.

A partir de los valores de la ganancia crítica Kc y el período de oscilación Tc determinaremos los valores de las constantes Kp, Ki y Kd a través de la tabla 2.2.

Controlador	Kp	$Ti = \frac{1}{Ki}$	$Td = \frac{1}{Kd}$
P	$0.5 * Kc$	∞	0
PI	$0.45 * Kc$	$0.83 * Tc$	0
PID	$0.59 * Kc$	$0.5 * Tc$	$0.125 * Tc$

Tabla 2.2 Valores para sintonizar las constantes del controlador en función de los datos obtenidos tras método ganancia crítica.

2.3 Plataformas de electrónica libre

2.3.1 Microcontroladores

Un microcontrolador, o MCU, del inglés "Microcontroller Unit", puede definirse como un pequeño ordenador con un único circuito integrado programable, es decir, con un único microchip, podemos ver un ejemplo en la figura 2.13. Podemos afirmar que se trata de un pequeño ordenador ya que dicho circuito integrado contiene las principales unidades funcionales de un ordenador, es decir, una unidad central de procesamiento, unidades de memoria, entradas programables y periféricos de salida.

Figura 2.13 Ejemplo de microcontrolador.

Para comprender de una forma más completa el concepto de microcontrolador, vamos a describir en mayor detalle sus componentes principales:

- **Puertos de entrada y salida (E/S)**. En realidad estos puertos son pines que recogen y generan señales digitales a otros circuitos, formando las interfaces entre un microcontrolador y el mundo exterior. Cualquier sensor o actuador que vaya a comunicarse con el microcontrolador lo hará a través de dichos puertos. Al tratarse de puertos digitales, el microcontrolador recibirá dicha información en forma de bits.

- **CPU**, o unidad central de procesamiento, se encargará de realizar todos los cálculos en base a la programación y de interactuar con circuitos externos.

- La **unidad de memoria**, que incluye tanto al programa que está siendo ejecutado, como espacio para almacenar la información generada por el mismo. La capacidad de esta unidad de memoria es limitada, por lo que debe ser gestionada correctamente.

- Un **bus** con dos líneas de **comunicación serial**, una para transmitir información desde el microcontrolador y otra para que reciba información, permitiendo el intercambio de bits de forma secuencial, a través de dichos interfaces.

- Además, la mayoría de microcontroladores incluyen el uso de dos dispositivos adicionales. Un módulo **temporizador** que permite al microcontrolador realizar tareas de forma periódica. Un **conversor analógico-digital**, ADC, del inglés "Analog to Digital Converter", que permite al microcontrolador aceptar entradas analógicas.

La **función principal de un microcontrolador** es la de controlar a otros elementos a través de los interfaces de entrada y salida, para ello, será necesario programarles. En sus inicios, los microcontroladores eran programados únicamente en lenguaje ensamblador. Sin embargo, en la actualidad se dispone también de lenguajes de mayor nivel para programarles, como C. Sin embargo, en la mayoría de los casos será necesario un hardware adicional que cargue nuestro programa en los microcontroladores, conocido como **programador de microcontroladores**. El problema principal que acarrea dicho hardware es que para programas con un grado de computación alto puede producir tiempos de carga largos. Para paliar dicho efecto, hay microcontroladores que incorporan un pequeño software previamente cargado llamado "**Bootloader**". Este software será cargado una única vez a través de un programador de microcontroladores para que, a partir de ese momento, se pueda cargar nuestro programa al microcontrolador sin necesidad de un programador.

En la actualidad, los microcontroladores son uno de los dispositivos más comunes en dispositivos electrónicos modernos, aumentando sus prestaciones. Podemos encontrar microcontroladores en un gran número de estos dispositivos, desde nuestra lavadora o frigorífico hasta el sistema de arranque de nuestro vehículo. Los microcontroladores más populares son los Basic Stamp, PICAXE, procesadores ARM y Atmel AVRs. Por ejemplo, para dispositivos Arduino, se emplea el microcontrolador Atmel ATMega, siendo el cerebro de las diferentes placas Arduino, como veremos a continuación.

2.3.2 Plataforma Arduino

De acuerdo con sus creadores [35], Arduino es una plataforma de electrónica libre para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Gracias a la plataforma Arduino podremos desarrollar proyectos electrónicos propios, con una serie de entradas, como sensores o pulsadores, podremos controlar diferentes dispositivos de salida, ya sean luces, motores o diversos aparatos electrónicos. En resumen, Arduino puede ser el cerebro de una gran variedad de proyectos electrónicos. Los programas Arduino pueden ser autónomos, si la placa una vez programada no necesita conexión a un ordenador y funciona de manera independiente. En otros casos, la placa Arduino debe estar conectada de forma permanente, vía USB, Ethernet… con otra aplicación con un software específico que permita el intercambio de información entre la placa y la aplicación.

Desde una perspectiva más técnica [1], Arduino es una placa de hardware libre formado por un microcontrolador programable y un conjunto de pines hembra unidos internamente a las entradas y salidas de dicho microcontrolador. Con el término placa hardware, nos referimos, en concreto, al término PCB, del inglés “Printed Circuit Board”. Una PCB es una superficie fabricada a partir de un material no conductor, sobre la que se laminan pistas de un material conductor, para conectar eléctricamente a los componentes electrónicos soldados en ella. Por lo tanto, podemos describir a Arduino como una placa PCB que implementa un diseño específico de circuitería interna. En la figura 2.14 podemos ver un ejemplo de una placa Arduino.

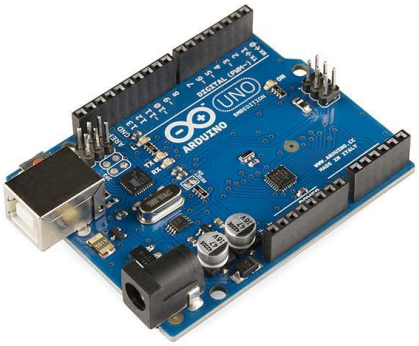


Figura 2.14 *Ejemplo de placa Arduino.*

Las características principales de Arduino son a su vez sus grandes ventajas, dichas características son las siguientes:

- Las placas Arduino son de **bajo coste** en comparación con otras plataformas de microcontroladores. La mayoría de placas Arduino pueden adquirirse pre ensambladas por un precio menor de 50€ [6]. Sin embargo, también existe la posibilidad de comprar sus componentes por separado y montar la placa por cuenta propia, reduciendo aún más su precio.

- Su entorno de programación es **multiplataforma**. El software se puede instalar y ejecutar en sistemas Windows, Mac OS y Linux. Además, su **entorno de programación es claro y sencillo** de utilizar o aprender para usuarios noveles, con un simple botón podemos cargar un programa en la placa y comprobar su resultado. Además, ofrece un entorno muy completo y flexible que los

usuarios más avanzados pueden usar a su favor. Además, aporta una documentación completa, con numerosos ejemplos de programas y funciones integradas en librerías de acceso libre.

- El **software** de la plataforma Arduino es **libre y extensible**, ya que cualquiera puede mejorar tanto el entorno de desarrollo software como el lenguaje de programación. El hecho de que el software Arduino sea libre significa que cualquier usuario puede ejecutarlo, copiarlo, distribuirlo, estudiarlo, cambiarlo y mejorarlo sin necesidad de obtener permisos de parte del autor. Existen multitud de librerías software de libre acceso, que nos permiten añadir una mayor funcionalidad a nuestro proyecto facilitándonos el propio desarrollo software. Además, cualquiera puede ampliar y adaptar dichas librerías según sus necesidades. Este software de código abierto permite la colaboración de una multitud de personas que radica en un mayor desarrollo para la plataforma Arduino.

- El **lenguaje de programación** de Arduino es **sencillo**, está compuesto por un conjuntos de funciones de alto nivel escritas en C/C++ y el propio entorno de desarrollo dispone de un compilador. No será necesario conocer lenguaje ensamblador ni otros detalles más técnicos del microcontrolador de la placa, como sus registros, para desarrollar un programa para Arduino. Sin embargo, usuarios más avanzados tienen la opción de poder hacerlo, cualquier interesado en conocer los detalles más técnicos del software de la plataforma Arduino puede emplear el lenguaje AVR C que emplea el microcontrolador Atmel de la placa.

- El **hardware** de la plataforma es **libre y extensible**. El hardware de la plataforma Arduino es libre ya que los ficheros esquemáticos de la placa Arduino están publicados bajo licencia Creative Commons, es decir, bajo una licencia libre que permite a cualquier persona interesada estudiarlos para construir su propia versión de la placa, extenderla o mejorarla. Además, esto permite que exista un gran abanico de variantes de las placas no oficiales.

- Arduino dispone de una **comunidad con un gran número de usuarios**, hecho que enriquece la documentación existente de la plataforma y favorece enormemente a su desarrollo.

- Las placas Arduino son **reutilizables y versátiles**, ya que una misma placa puede ser reutilizada para diferentes proyectos. Y versátiles ya que las placas disponen de varios tipos diferentes de entradas y salidas, permitiendo obtener información de múltiples sensores y enviar órdenes a actuadores de diferentes maneras.

Como podemos comprobar, Arduino no es un simple microcontrolador, sino que es una plataforma de electrónica libre formada por diversas placas hardware y un único entorno de programación [2]. La mayor parte de las placas hardware Arduino son desarrolladas por el organismo oficial de la plataforma, aunque es posible encontrar réplicas de las placas más

comunes, como Uno o Mega, de otros vendedores. La mayoría de las placas oficiales de Arduino incorporan un microcontrolador de la familia megaAVR, en concreto del tipo ATmega. La mayoría de las placas también incorporan un regulador de voltaje de 5V y un oscilador de 16MHz. Además, su microcontrolador incorpora un "Bootloader" pre-cargado que simplifica la carga de programas a la memoria flash del microcontrolador, dicho "Bootloader" también es de código libre, no necesitará hardware adicional para la carga de programas. Las diferentes placas Arduino se diferencian en el tamaño y funcionalidades, en términos de procesador y del número de entradas y salidas, que presentan. En la figura 2.15 podemos ver las placas más conocidas:

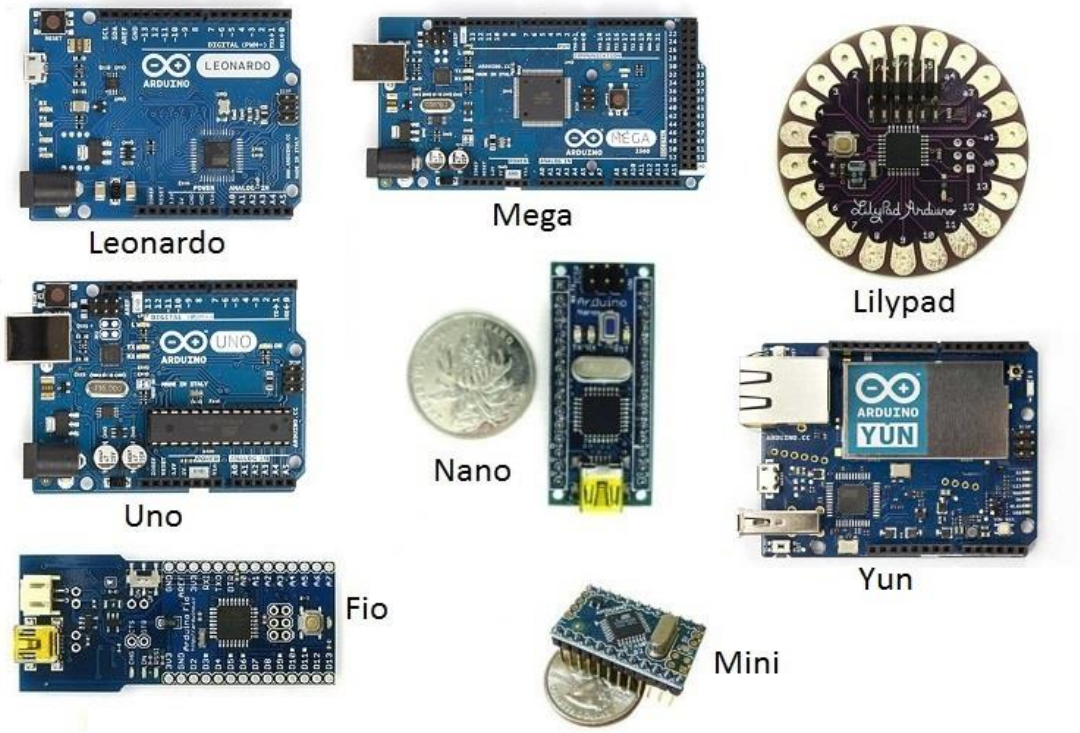


Figura 2.15 *Los tipos de placas Arduino más comunes.*

- La placa **Arduino Leonardo** es una de las más recientes, se basa en el microcontrolador ATMega32u4. Dispone de 12 entradas analógicas y 20 pines de entrada/salida digitales, de los cuales 7 pueden ser usados como salida PWM. Además, la placa dispone de todo lo necesario para el correcto funcionamiento del microcontrolador, es decir, dispone de un oscilador de cristal de 16MHz, conexión micro USB, 6 pines ICSP, del inglés "In Circuit Serial Programming", que permite cargar el "Bootloader". La diferencia de la placa Leonardo respecto a otras placas Arduino es que incorpora comunicación USB, por lo que al conectarlo con nuestro PC, nos aparecerá como un teclado o un ratón, en lugar de puerto serial virtual (COM). El precio de la placa Leonardo ronda los 18€ en la tienda oficial.

- La placa **Arduino Uno** es una de las placas más utilizadas, se basa en el microcontrolador ATMega328. Dispone de 6 entradas analógicas y 14 pines de entrada/salida digitales, de los que dos pueden ser usados como salida PWM. Al igual que la placa anterior, dispone de los elementos necesarios para el correcto funcionamiento del microcontrolador. Una vez adquirida, la placa viene

completamente preparada para ser usada. Además, podemos alimentarla a través de una conexión USB o por una fuente de alimentación externa. El precio de la placa Uno ronda los 20€ en la tienda oficial.

- La placa **Arduino Fio** es una de las placas considerada como mejor opción para proyectos en los que se requiere conectividad inalámbrica, ya que incorpora un socket XBee, que gracias al "Bootloader" incorporado permite programar la placa de forma inalámbrica. Además, permite ser alimentado a través de una conexión mini USB o a través de La placa se basa en el microcontrolador ATmega 328P, que a diferencia de las placas anteriores, opera a 3.3V y a 8MHz. La placa también dispone de 8 entradas analógicas y 14 pines de entrada/salida digitales. El precio de la placa Fio ronda los 23€ en la tienda oficial.

- La placa **Arduino Mega** es una placa más avanzada que la Uno, dispone de un microcontrolador ATmega2560 con mayor memoria que el de la placa Uno. Además cuenta con un mayor número de entradas y salidas. En concreto, dispone de 54 pines de entrada/salida digitales y 16 entradas analógicas. El precio de la placa Mega ronda los 35€ en la tienda oficial.

- La placa **Arduino Nano** es una placa de menor tamaño que la placa Uno, sin embargo proporciona una funcionalidad similar, se trata de una placa completa. La diferencia entre esta placa y la placa Uno es que disponemos de una memoria menor en el microcontrolador, no dispone de un conector de alimentación y trabaja con un conector mini USB. Sin embargo, su coste es más reducido.

- La placa **Arduino Lilypad** fue diseñada para proyectos "wearable" ya que tanto la placa, como la fuente de alimentación, sensores y actuadores que emplee pueden coserse a la tela empleando hilo conductor. La placa incorpora el mismo número de pines de entrada y salida que la placa Uno. Sim embargo, necesita de un adaptador externo para cargar programas en el microcontrolador de la placa. El precio de la placa Lilypad ronda los 23€ en la tienda oficial.

- La placa **Arduino Mini** es la versión más pequeña de todas las placas Arduino. Sin embargo, presenta las mismas funcionalidades que Arduino Uno, ya que emplea el mismo microcontrolador. Al igual que la placa Lilypad, necesita un adaptador externo para cargar programas en la placa. Además, al igual que la placa Nano, dispone de pines macho para conectarse directamente en la "protoboard". El precio de la placa Mini ronda los 14€ en la tienda oficial.

- La placa **Arduino Yun** es una placa similar a la placa Uno pero con una serie de capacidades extra. La placa Yun presenta capacidades nativas para conexión Ethernet, WiFi, USB y micro SD sin necesidad de emplear módulos adicionales. El precio de la placa Yun ronda los 52€ en la tienda oficial.

2.3.3 Plataforma Wiring

Según la página web oficial de la plataforma [7], la plataforma Wiring es un framework de acceso libre para microcontroladores. La plataforma Wiring permite escribir software multiplataforma para controlar dispositivos conectados a un rango amplio de placas electrónicas, ya que no está limitado al uso único de hardware de la plataforma Wiring, permite usar otras plataformas como Arduino. De esta forma la plataforma Wiring creó originalmente su framework con la idea de poder ser útil en multitud de proyectos diferentes e impulsar una comunidad en la que tanto novatos como expertos compartan diseños, conocimiento e ideas.

Por otro lado, la plataforma dispone de más de cien librerías que extienden el software del framework. Sin embargo, existen varios tipos entre las librerías disponibles: librerías programadas específicamente para un hardware concreto, librerías multiplataforma útiles para cualquier hardware y librerías contribuidas por la comunidad a través de Internet.

En la figura 2.16 podemos ver la placa electrónica más común de la plataforma Wiring, llamada **Wiring S**. Las características principales de la placa son las siguientes: está basada en un microcontrolador ATMega644p. Dispone de treinta y dos pines de entrada/salida digitales, de los cuales seis pueden ser usados como salida PWM, y algunos están reservados para usos especiales, como comunicación SPI e interrupciones externas. También dispone de seis pines de entrada analógicos. Dispone de una conexión para un alimentador de corriente externo, ya que la placa puede trabajar tanto a 5V como a 3,3V. Además, dispone de una entrada USB para conectarlo con nuestro ordenador.

Figura 2.16 *Placa Wiring S.*

2.3.4 Plataforma Netduino

Netduino [8], es una plataforma de electrónica libre basada en microcontroladores con arquitectura ARM de 32-bits, por tanto disponen de una mayor capacidad que varias de las placas Arduino. Además, emplea el framework .NET micro. Las ventajas que ofrece la plataforma es que permite combinar lenguajes de programación de nivel alto con las características de su microcontrolador con mayor capacidad, permite programación basada en eventos, programación multitarea y mejor depuración de código. Por otro lado, la mayoría de las placas de la plataforma Netduino tienen sus pines de entrada y salida diseñados de tal manera que sean compatibles con la mayoría de módulos adicionales para la plataforma Arduino.

En la figura 2.17, podemos encontrar las tres placas más comunes de la plataforma Netduino. Estas placas son:

- **Netduino 2** [9]. Es la placa Netduino más común, la capacidad de su microcontrolador STMicro STM32F2 es mayor que la placa más común Arduino, es decir, Arduino Uno. Emplea el sistema operativo .NET Micro Framework 4.2. Por defecto no tiene capacidad de conexión a red ni almacenamiento externo, sin embargo permite añadirle módulos adicionales tanto para ofrecerle conectividad como almacenamiento a través de una tarjeta SD externa. Por último, dispone de 22 pines GPIO, del inglés "General Purpose input/output", pines de un circuito integrado que pueden ser de entrada o salida a determinación del usuario durante la ejecución, seis de ellos ofrecen salidas PWM, cuatro ofrecen salidas UART para comunicación serial y permite el uso de los protocolos I2C e ISP. Además, dispone de seis pines analógicos de entrada. La disposición de los pines garantiza compatibilidad con la mayoría de módulos adicionales para Arduino.

- **Netduino 2 Plus**. Esta placa emplea un microcontrolador con mayor capacidad que el de la placa Netduino 2. Sin embargo, emplea el mismo sistema operativo. Por defecto, incorpora una conexión RJ-45 que ofrece a la placa una conexión a la red Ethernet de hasta 10 Mbps y una entrada para tarjeta micro-SD de hasta 2Gb. El número de pines de entrada y salida es el mismo que para la placa Netduino 2. Por tanto, su disposición garantiza compatibilidad con la mayoría de módulos adicionales para Arduino.

- **Netduino 3 WiFi**. Esta placa emplea un microcontrolador con mayor capacidad que el de la placa Netduino 2 Plus. Sin embargo, emplea el mismo sistema operativo. Por defecto, incorpora un receptor WiFi que ofrece a la placa una conexión inalámbrica y una entrada para tarjeta micro-SD de hasta 2Gb. El número de pines de entrada y salida es el mismo que para la placa Netduino 2. Por tanto, su disposición garantiza compatibilidad con la mayoría de módulos adicionales para Arduino.

Figura 2.17 *Placas principales de la plataforma Netduino.*

2.4 Sensores

De acuerdo con [2] un **sensor** es un dispositivo que puede medir un fenómeno físico (temperatura, humedad, radiación…) o detectar una concentración química (humo) y cuantificarlo. En otras palabras, un sensor produce una representación medible de un fenómeno dentro de un rango específico, por ejemplo, en un determinado rango de voltaje. En la mayoría de las ocasiones un sensor se conecta a un pin de entrada de un microcontrolador, como en el caso del microcontrolador de nuestra placa Arduino. Por otra parte, podemos dividir los sensores en dos categorías:

1. **Sensores analógicos**. Un sensor analógico se conecta a un circuito de tal manera que va a tener una salida con un rango específico de voltaje, normalmente, entre 0 y 5V, es decir, produce una señal de salida analógica, continua en tiempo y amplitud. Esta señal de salida analógica se dirige a un pin de entrada analógico del microcontrolador que usa un conversor analógico a digital, ADC, para convertir el voltaje de salida del sensor en un valor numérico que podamos leer y procesar más adelante. Por ejemplo, en el caso de nuestro Arduino Uno cada pin analógico dispone de un conversor analógico digital con una resolución de 10 bits, es decir, que nos ofrece un rango de valores numéricos entre 0 y 1024, siendo 0 el valor más bajo del rango de voltaje (0V) y siendo 1024 el valor más alto (5V).

2. **Sensores digitales**. Un sensor digital produce una señal de salida discreta o voltaje, la cual es una representación digital de la cantidad medida. Los sensores digitales producen una señal binaria de salida cuya lógica 1 representa un voltaje de 5V y su lógica 0 representa un voltaje de 0V. Una señal digital no produce valores continuos, sino discretos. En comparación con las señales analógicas, éstas son bastante más precisas, ya que una señal digital no se ve tan afectada por el ruido. Además, la precisión de una señal digital depende del número de bits empleados para representar la cantidad medida.

En muchos casos, para conectar cualquier tipo de sensor a un microcontrolador se suelen emplear resistores llamados **“pull-down”** y **“pull-up”**, para prevenir que un exceso de corriente del circuito del sensor pueda llegar al microcontrolador. Por ejemplo, para entender el uso de ambos resistors, consideremos el ejemplo de un pulsador con dos estados, “on” y “off”, en el que uno genera 5V en el pin con el que se conecta al microcontrolador y el otro de 0V.

Como podemos ver en la figura 2.18, si colocamos una resistencia de tipo “pull-up” para el pulsador, cuando el pulsador se encuentra abierto, el flujo de corriente circulará por el pin de entrada al microcontrolador ofreciendo en la señal digital una lógica 1, es decir, un valor alto. En el momento de pulsar el pulsador, quedará cerrado, por lo que el flujo de corriente se dirigirá a tierra, indicando en la señal digital una lógica 0 en el pin de entrada del microcontrolador. Por el contrario, en caso de conectar la resistencia en modo “pull-down”, cuando el pulsador esté presionado la señal digital de entrada al microcontrolador tendrá una lógica 1 y cuando no se encuentra pulsado tendrá una lógica 0.

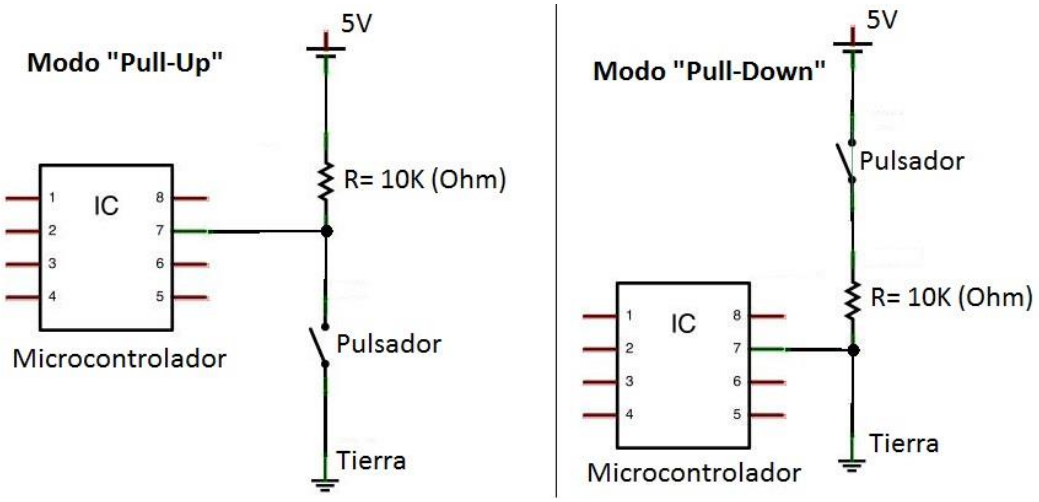


Figura 2.18 *Resistores en modo "pull-up" y "pull-down", fuente [2].*

2.4.1 Sensores de temperatura

En este apartado vamos a ver tres sensores de temperatura que suelen emplearse en conjunto con dispositivos de una plataforma de electrónica libre.

Sensor de temperatura digital DS18B20:

El primer sensor que vamos a tratar es el sensor digital de temperatura DS18B20. De acuerdo con su hoja de datos [10], el sensor DS18B20 es un termómetro digital cuyas mediciones de temperatura en grados Celsius tienen una resolución mínima de 9 bits y máxima de 12 bits, seleccionable por el usuario. Este sensor digital utiliza el protocolo 1-Wire para comunicarse a través de un único cable de datos con un microcontrolador. Además, cada sensor de temperatura DS18B20 dispone de un código identificador único de 64-bits. A través del protocolo 1-Wire pueden conectarse varios sensores digitales DS18B20 a un mismo pin de entrada digital del microcontrolador y gracias al código identificador de cada sensor podemos identificar las mediciones de temperatura de cada uno. Estas características lo hacen ideal para aplicaciones en las que se requiere monitorizar temperaturas dentro de un edificio y en sistemas de control.

Como podemos comprobar en su hoja de datos [referencia], el sensor de temperatura digital DS18B20 tiene las siguientes características:

- Puede ser alimentado a través de un alimentador externo o a través de la línea de datos que emplea para comunicarse con el microcontrolador.
- El rango de temperaturas que puede medir empieza en -55ºC y termina en 125ºC. Dispone de una precisión mínima de 0,5ºC y máxima de 0,0625ºC, ajustable por el usuario, para el rango entre -10ºC y 85ºC.
- El tiempo máximo que tarda en convertir una temperatura medida en una señal digital de 12 bits es de 750ms.
- Ofrece un modelo resistente al agua.

En la figura 2.19 podemos ver tanto un modelo genérico del sensor DS18B20 como un módulo del mismo sensor resistente al agua. El precio en el mercado del modelo genérico ronda los 2€ mientras que el precio en el mercado del modelo resistente al agua ronda los 4€.

Figura 2.19 *Modelos del sensor digital ds18b20, fuente [10].*

Sensor analógico de temperatura TMP36

De acuerdo con su hoja de datos [11], el sensor analógico TMP36 es un sensor de temperatura con una precisión de centígrados y de bajo voltaje. El sensor analógico TMP36 está formado por un circuito integrado cuyo voltaje de salida será mayor o menor en función de la temperatura medida.

La impedancia de salida de dicho circuito integrado es baja, por lo que sumado a qué ofrece una salida lineal y calibrada, simplificará la interacción con el conversor analógico a digital del pin de entrada analógico del microcontrolador al que se conecta. El hecho de que la señal de salida analógica esté calibrada significa que el valor de la salida sigue un escala de 10mV/ºC. Además, el sensor TMP36 permite realizar medidas de temperatura en un rango entre -40ºC y 125ºC, ofreciendo un voltaje máximo en su señal de salida de 750mV para una medida de 125ºC.

El sensor TMP36 puede ser alimentado con una tensión entre 2,7V a 5,5V. Además, puede trabajar correctamente con corrientes de alimentación de hasta 50 micro-Amperios, lo que garantiza que no se sobrecaliente demasiado. En la figura 2.20 podemos ver un modelo genérico de un sensor TMP36, cuyo precio en el mercado ronda los 1,5€.

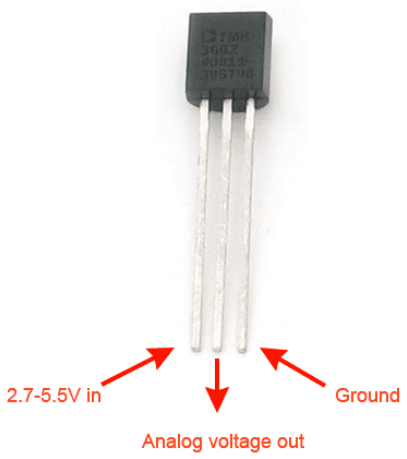


Figura 2.20 *Sensor analógico TMP36, fuente [20].*

Termistor NTC

Un termistor no es más que una resistencia cuyo valor varía en función de la temperatura. En realidad, cualquier resistencia varía con la temperatura. Sin embargo, el valor de un termistor varía más drásticamente ante un cambio en la temperatura que cualquier otra resistencia, pudiendo variar hasta 100 Ohmios por cada grado de temperatura en el cambio.

Como podemos ver en su hoja de datos [12], el termistor NTC, del inglés "Negative Temperature Coefficient", decrece el valor de su resistencia a medida que la temperatura aumenta. Sin embargo existen otros termistores, de tipo PTC, del inglés "Positive Temperature Coefficient", cuyo valor de la resistencia aumenta a medida que la temperatura también aumenta.

El problema que presenta este tipo de sensor analógico es que el valor del voltaje de su señal de salida no se encuentra calibrado. Por tanto, para determinar el valor de la temperatura a través del valor del voltaje de dicha señal habrá que realizar un estudio previo con el sensor para poder calibrarlo, ya que como podemos ver en la hoja de datos, la variación del valor de la resistencia no varía de forma lineal con la temperatura, sino que lo hace de forma exponencial. En cualquier caso, existen diversos métodos y funciones matemáticas que nos permiten calibrarlo vía software. Sin embargo, no ofrece una precisión mínima como los dos sensores anteriores, sino que su precisión dependerá del diseño que se le ofrezca. Por último, cabe destacar que es el sensor más económico en el mercado, cuyo precio ronda los 0.75€.

Figura 2.21 *Termistor NTC, fuente [12].*

2.4.2 Sensores de temperatura y humedad

Sensor DHT22

El sensor DHT22 es un sensor digital capaz de medir la temperatura del aire de una sala y su nivel relativo de humedad que puede ser utilizado por cualquiera de las placas de una de las plataformas de electrónica libre vistas anteriormente. Para medir la temperatura, el sensor DHT22 emplea un termistor, es decir, una resistencia cuyo valor varía drásticamente con la temperatura. Sin embargo, a diferencia del termistor NTC visto en el apartado anterior, como podemos comprobar en su hoja de datos [13], el sensor produce una salida digital de 8 bits de resolución calibrada, con una precisión de 0,5ºC. El rango de temperatura en el que puede trabajar el sensor es entre -40ºC hasta 80ºC.

Para medir la humedad emplea un sensor capacitivo, cuyo condensador tendrá un valor para su capacidad que variará en función del nivel de humedad presente en la sala donde se ubica. De esta forma, el dieléctrico del condensador (inicialmente el aire de la sala)

variará en función de la humedad relativa, lo que hace que varíe su capacidad. El sensor será capaz de medir el nivel de humedad relativa en un rango entre el 0% y el 100% con una precisión del 2%.

Por último, el rango de voltaje recomendado para alimentar este sensor está comprendido entre los 3,3V y los 6V, lo que permite una distancia máxima de transmisión de datos entre el sensor y el dispositivo al que se conecta de 20m. El precio en el mercado del sensor DHT22 ronda los 6€. En la figura 2.22 podemos ver una imagen del sensor DHT22.

Sensor DHT11

El sensor DHT11 es un sensor digital capaz de medir la temperatura del aire de una sala y su nivel relativo de humedad que puede ser utilizado por cualquiera de las placas de una de las plataformas de electrónica libre vistas anteriormente. Para medir la temperatura, emplea un termistor igual que en el caso del sensor DHT22, cuya salida digital está calibrada. Sin embargo, la precisión de la medida de la temperatura de este sensor es de 2ºC y opera en un rango entre 0ºC y 50ºC, de acuerdo con su hoja de datos [14].

En este caso, el sensor emplea un sensor resistivo para medir la humedad, hecho sobre una pequeña tableta de un polímero capaz de absorber agua. De esta forma, se mide la resistencia eléctrica del polímero, que cambia según el contenido de agua. Este sensor de la humedad relativa dispone de una precisión del 5% y trabaja en un rango entre el 20% y el 90% de la humedad relativa de la sala.

En cuanto a la alimentación, el rango de voltaje recomendado está comprendido entre los 3,3V y 5V. En caso de que la distancia entre el sensor DHT11 y el dispositivo al que está conectado es menor de 20m se requiere el uso de una resistencia “pull-up”, en caso de ser mayor, no. El precio del sensor DHT11 en el mercado ronda los 3€. En la figura 2.22 podemos ver una imagen del sensor DHT11.

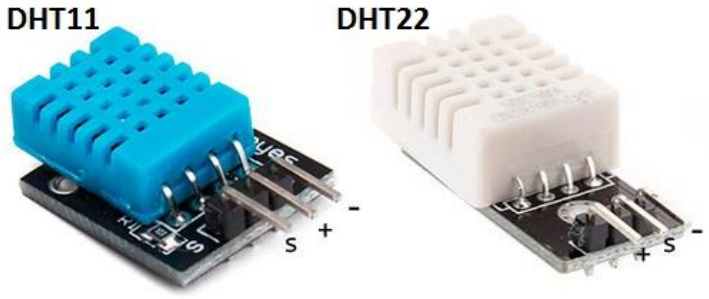


Figura 2.22 *Sensores de Temperatura y Humedad, fuente [21].*

2.5 Ordenadores de bajo coste

En este apartado, vamos a ver diferentes ordenadores de bajo coste que tienen un tamaño apenas superior a una tarjeta de crédito. Para ello, vamos a ver tres plataformas que disponen de uno o varios ordenadores de bajo coste.

2.5.1 Raspberry Pi

De acuerdo con la descripción que podemos ver en su web oficial [16], la fundación Raspberry Pi es una organización educativa registrada en UK, del inglés “United Kingdom”. El principal objetivo de la fundación es de impulsar la formación, tanto de adultos como de menores, en el campo de la informática y similares.

Una Raspberry Pi es un ordenador de bajo coste, cuyo tamaño es apenas superior al de una tarjeta de crédito, que puede conectarse a un monitor y emplear teclado y ratón, es capaz de hacer lo mismo que cualquier ordenador de sobremesa, como navegar por Internet o visualizar vídeos. Además, es un dispositivo que ofrece a las personas que exploren la informática y que aprendan lenguajes de programación como Python. Por último, es un dispositivo con la habilidad de interactuar con el mundo externo, ideal para multitud de proyectos de distinta índole, por ejemplo, un sistema domótico de control de la temperatura.

En la figura 2.23, podemos observar las tres Raspberry Pi que podemos encontrar los tres modelos de Raspberry Pi que podemos encontrar en el mercado, desde el modelo A+ hasta el último, el modelo 2. Las características de cada una de las placas son las siguientes:

- **Raspberry Pi Modelo A+**. Como podemos observar en [17], dispone de 40 pines GPIO, del inglés “General Purpose Input/Output” para interactuar con otros dispositivos. Dispone de una entrada USB y otra para una tarjeta micro-SD. Este es el modelo que reemplazó a la Raspberry Pi original en Noviembre de 2014, entre otras cosas, consume menos potencia y la calidad del audio que produce ofrece menor nivel de ruido que el modelo anterior. Es un modelo ideal para proyectos que requieren un dispositivo con un bajo consumo y que no requieran uso de Ethernet o múltiples puertos USB. Su precio en el mercado ronda los 21€.

- **Raspberry Pi Modelo B+**. Como podemos ver en [18], este modelo supone la revisión final del modelo original Raspberry Pi, en Julio de 2014. Dispone de 40 pines GPIO, 4 puertos USB 2.0 una ranura para tarjeta micro-SD y un puerto de entrada RJ-45. Al igual que el modelo A+, ofrece un menor consumo y mejora la calidad del audio respecto al modelo original. Es un modelo perfectamente compatible para su uso en escuelas, ya que ofrece una mejor flexibilidad para los estudiantes que el modelo A+ gracias a sus múltiples puertos USB y RJ-45. Su precio en el mercado ronda los 22€.

- **Raspberry Pi 2**. Como podemos ver en [19], supone la segunda generación de la Raspberry Pi reemplazando al anterior modelo B+ en Febrero de 2015, ya que entre otras cosas, dispone de un procesador ARMv7 de mejores características y 1Gb de memoria RAM. En cuanto a puertos GPIO, entradas USB y demás elementos dispone del mismo número que el modelo B+. Sin embargo, su procesador más potente permite instalar cualquier distribución GNU/Linux, incluido Ubuntu, y Microsoft Windows 10.

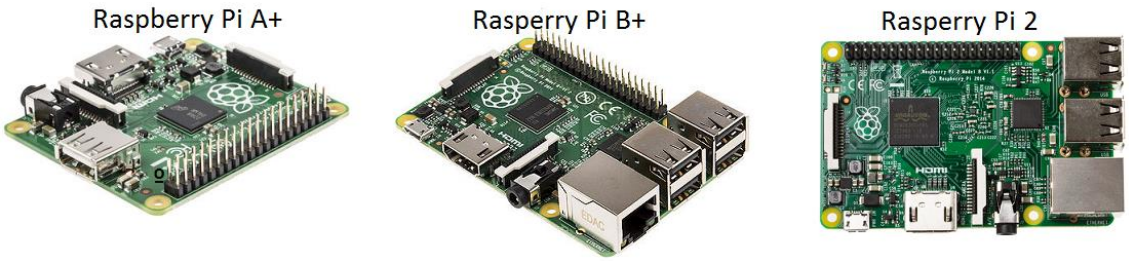


Figura 2.23 *Modelos de Raspberry Pi, fuente [22].*

2.5.2 BeagleBoard

Como podemos observar en su página web oficial [23], la fundación BeagleBoard es una fundación sin ánimo de lucro en pos de la educación e impulso al diseño y uso de software y hardware libre en proyectos de informática embebida. A través de su página web oficial se puede acceder a un foro en el que tanto creadores como desarrolladores de software y hardware libre intercambian ideas, conocimiento y experiencia.

Al igual que la fundación Raspberry Pi, la fundación BeagleBoard ofrece un ordenador de bajo coste al mercado. En este caso, el nombre del ordenador de bajo coste es BeagleBone Black. En la figura 2.24 podemos observar dicho ordenador, cuyas características de acuerdo a [24], son las siguientes:

- Dispone de un procesador ARM cuya capacidad permite instalar sistemas operativos Android, Debian y Ubuntu entre otros.
- Dispone de una capacidad de almacenamiento flash de 4Gb flash y 512 Mb de RAM.
- Dispone de un acelerador de gráficos 3D y dos microcontroladores de 32-bits.
- Dispone de una entrada USB, tanto para alimentación como comunicación. Un puerto de entrada RJ-45, 2 bloques de 46 pines cada uno para interactuar con otros dispositivos y una salida HDMI.
- Su precio en el mercado ronda los 55€.

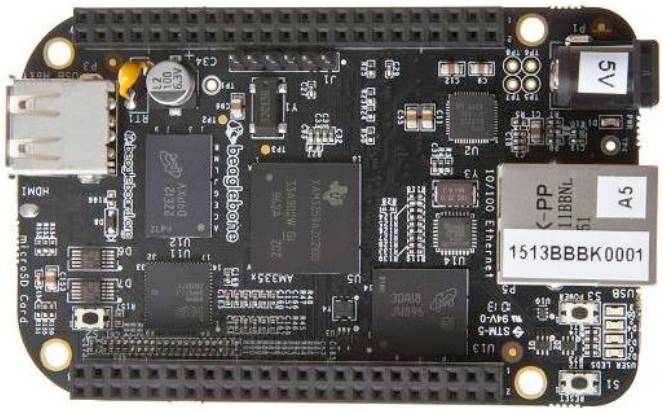


Figura 2.24 *Ordenador de bajo coste BeagleBone Black.*

2.5.3 pcDuino

Un dispositivo pcDuino [25] es un ordenador de bajo coste, con características similares a los dos tipos vistos anteriormente. Sin embargo, la peculiaridad de este dispositivo es que dispone de pines compatibles con shields y placas de Arduino y dispone de un entorno de programación para Arduino.

En la figura 2.25, podemos encontrar varios de los dispositivos pcDuino disponibles en el mercado. Dichos dispositivos son los siguientes:

- **pcDuino Lite** [26]. Es el dispositivo pcDuino de menor capacidad, no tiene almacenamiento flash interno. Como podemos ver en [26], dispone de 512Mbs de memoria RAM, por lo que no permite la instalación del sistema operativo Android. Sin embargo, permite otros sistemas operativos como Ubuntu. Por otra parte, dispone de una ranura para micro-SD de hasta 32Gb y una entrada RJ-45. Además, dispone de dos entradas USB y una salida HDMI. Este dispositivo fue diseñado específicamente para diseñar proyectos que demandad una mayor capacidad de computación extendiendo los shields para Arduino. Dispone de una API que permite al usuario acceder a todas las funciones que emplearíamos con el lenguaje para Arduino. Su precio en el mercado ronda los 36€.

- **pcDuino 2** [27]. Este dispositivo pcDuino ofrece una funcionalidad muy similar al dispositivo anterior. Sin embargo, dispone de unas características superiores que extienden al anterior modelo. En primer lugar, dispone de 1Gb de memoria RAM, por lo que también permite la instalación del sistema operativo Android. Además, dispone de 2Gb de almacenamiento interno flash. Por otro lado, dispone de un módulo Wifi incorporado, adicional a la entrada RJ-45 y dispone de unos pines compatibles con los pines de Arduino. Su precio en el mercado ronda los 50€.

- **pcDuino 3** [28]. Es el modelo pcDuino más potente que existe, extiende las funcionalidades vistas en el anterior modelo. Por un lado, soporta el uso de múltiples lenguajes de programación, dispone de pines para entradas analógicas que al igual que las anteriores son compatibles con Arduino e incluso más funcionalidades extra como podemos ver en [28]. Su precio en el mercado ronda los 60€.

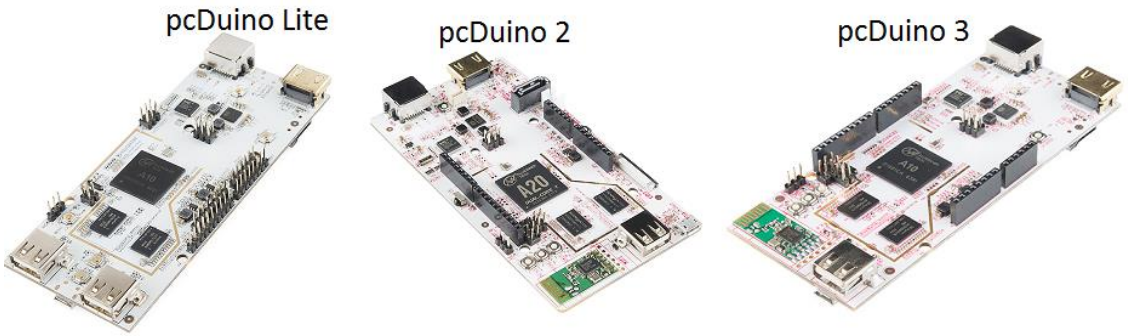


Figura 2.25 *Modelos de pcDuino fuente: [26], [27] y [28].*

Capítulo 3: Descripción del complejo

A lo largo de este capítulo vamos a describir en primer lugar las los diferentes edificios que forman el complejo en el que se sitúa nuestro sistema domótico. En segundo lugar, explicaremos los bloques de climatización que forman el sistema de climatización a controlar por nuestro sistema domótico.

3.1 Descripción de las instalaciones

A lo largo de esta sección, trataremos de describir los diferentes edificios que forman el complejo del colegio. Nuestro complejo se divide en varios edificios que presentaremos a continuación. Dividiremos cada edificio en bloques para la posterior explicación del funcionamiento de los sistemas de climatización empleados al final del capítulo.

Nuestro complejo está formado por tres edificios principales, como podemos comprobar en la figura 3.1. A continuación, veremos una pequeña descripción de cada uno de estos edificios.

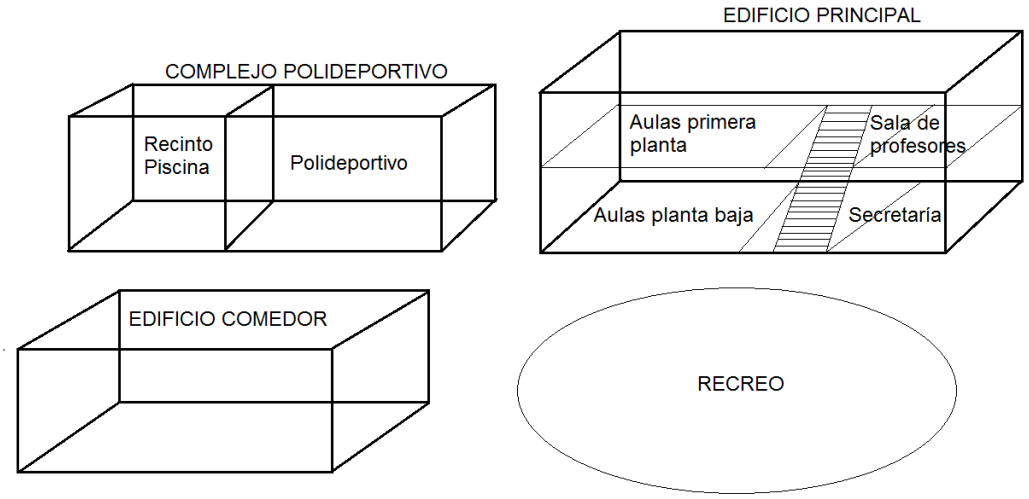


Figura 3.1 *Esquema del complejo del colegio.*

3.1.1 Edificio principal

El primer edificio, se trata del edificio principal o aulario. En este edificio, encontraremos el total de aulas que forman parte de nuestro colegio. En concreto, dispondremos de un total de veinte aulas repartidas a lo largo de las dos plantas del edificio principal. Además, podremos encontrar la sala de profesores en la primera planta, y debajo de la misma, en la planta baja, encontraremos la secretaría o sala de administración.

Por otra parte, este edificio dispone de una planta adicional, localizada en el subsuelo, bajo la planta baja. En esta planta, habrá una sala, en la cual estará tanto la caldera como la bomba de calor y frío.

3.1.2 Complejo polideportivo

El segundo edificio se corresponde con el complejo polideportivo. En este edificio, podremos encontrar dos recintos. El primer recinto será polideportivo cubierto. El segundo recinto será el recinto piscinas. En él, podremos encontrar dos piscinas climatizadas, una grande y otra pequeña.

3.1.3 Edificio comedor

El tercer edificio de nuestro complejo contendrá únicamente el espacio acondicionado para funcionar como comedor.

3.2 Descripción del sistema de climatización

Durante esta sección, trataremos de describir de forma general el funcionamiento del sistema de climatización del complejo (figura 3.1). Para ello, dividiremos los diferentes sistemas empleados en bloques para la posterior descripción de esquema de funcionamiento y elementos que lo componen.

3.2.1 Bloque 1: Climatizador

Este modelo de climatización será el que podremos encontrar para el recinto polideportivo y para el comedor. Este sistema será el utilizado tanto en invierno, ofreciendo calefacción al polideportivo y al comedor, como en verano, refrigerando ambas estancias. Como podemos observar en la figura 3.2, nuestro sistema se compone de varios elementos.

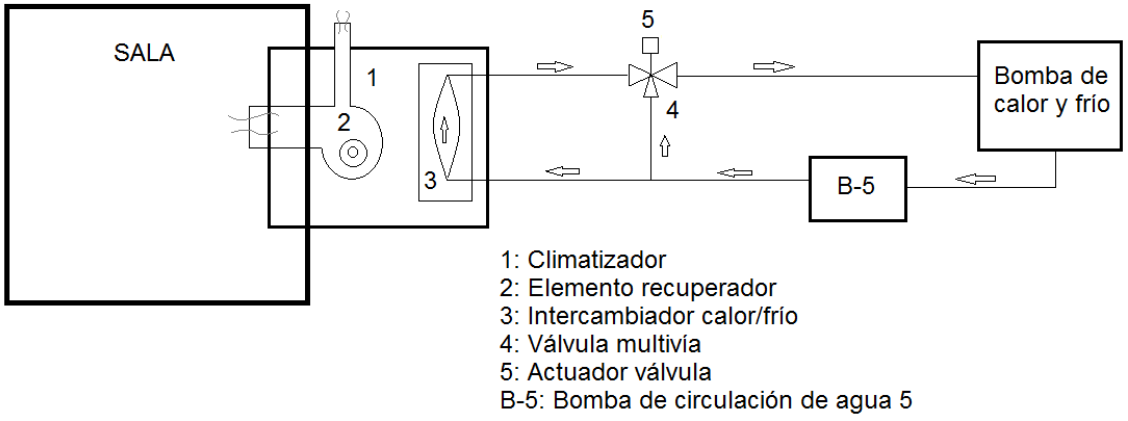


Figura 3.2 *Esquema de funcionamiento del climatizador.*

Elementos que forman el climatizador:

La **bomba de calor y frío** ofrecerá el flujo de agua que alimenta y recorre el circuito del climatizador para el recinto polideportivo y el comedor. En invierno, el flujo de agua ofrecido se encontrará a una temperatura entre 35ºC y 45ºC. Sin embargo, en verano el flujo de agua ofrecido se encontrará a una temperatura entre 7ºC y 12ºC.

La **bomba de circulación de agua B-5** se encargará de distribuir por el circuito del sistema el agua recibida desde la bomba de calor y frío. En nuestro sistema, la bomba de circulación impulsará el agua recibida por dos sub-circuitos independientes, uno dirigido al recinto polideportivo y otro al comedor. Como podemos observar, la figura 3.2 dispone de unas flechas con las que podemos comprobar la dirección que sigue el caudal agua a lo largo de nuestro sistema.

El **climatizador**, se compone de varios elementos. El primer componente, será la **unidad climatizadora**. A través de esta unidad, circulará el agua impulsada desde la bomba de circulación. En esta unidad tendrá lugar la transferencia de calor o frío entre el agua que circula por ella y el aire recogido en el recuperador. El segundo elemento, será el **recuperador**. Este componente se encargará de extraer cierta cantidad del aire interior de la sala para mezclarlo con otra parte de aire que recoge del exterior. Posteriormente, tras producirse la transferencia de calor o frío dentro de la unidad climatizadora, el recuperador devolverá el aire a la sala, para climatizar la sala y renovar el aire de su interior.

El último elemento del sistema será una **válvula multivía**. En nuestro caso, se trata de una válvula mezcladora de tres vías que consta de dos entradas y una salida. La primera entrada será la que entra a la válvula desde su izquierda, proveniente del climatizador. La segunda entrada será la que entra a la válvula desde abajo, cuyo flujo de agua viene desde la bomba b-5. La salida se situará en la parte derecha de la válvula. Además, el flujo de agua de salida será una mezcla de los flujos de entrada. Por último, la válvula dispondrá de un actuador, en este caso será un servomotor, que se encargará de regular el paso de los caudales de entrada que formarán el flujo de salida.

Funcionamiento del climatizador:

Vamos a distinguir entre dos modos de funcionamiento tanto en el climatizador del polideportivo como en el climatizador del comedor, los cuales son:

1. **Modo invierno o calefacción**, en el que el agua en circulación por el sistema se encontrará entre 35ºC y 45ºC. Al pasar por el interior de la unidad, se producirá una transferencia de calor entre el agua en circulación y el aire recogido y mezclado por el recuperador. En concreto, esta unidad se encargará de enfriar el agua para así calentar dicho aire, ya que recibirá el calor perdido por el agua. Tras esta operación, el recuperador se encargará de devolver el aire a la sala, a mayor temperatura, ofreciendo calefacción a la misma. La temperatura del sistema se encontrará entre 18ºC y 24º.

2. **Modo verano o climatización**, en el que el agua en circulación por el sistema se encontrará a una temperatura entre 7ºC y 12ºC. En este caso, se producirá una transferencia inversa. En concreto, este agua a menor temperatura, al

pasar por la unidad se calentará. De tal forma que el frío perdido se pasará al aire recogido y mezclado por el recuperador. Tras esta operación, el recuperador se encargará de devolver este aire, ahora con una temperatura menor, para poder así acondicionar la sala en verano. La temperatura del sistema se encontrará entre 20ºC y 28º.

Por otro lado, dependiendo de la diferencia entre la temperatura de consigna, es decir, la deseada para el polideportivo o para el comedor, y la temperatura medida en ellos, tendremos un comportamiento diferente dentro de la válvula multivía. En concreto, la posición determinada por el actuador dependerá de la diferencia entre la temperatura deseada en la sala y la temperatura actual de la misma. En la figura 3.3 podemos ver la válvula.

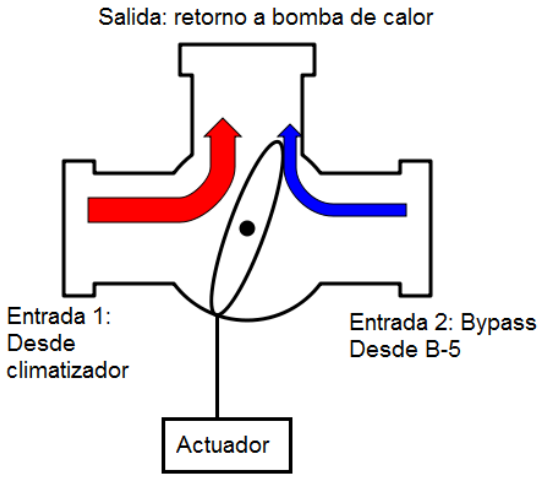


Figura 3.3 *Esquema de funcionamiento de la válvula mezcladora de tres vías del climatizador.*

Como podemos comprobar en la imagen anterior, el actuador decidirá el paso de cada entrada a la válvula. Cuanto mayor sea la diferencia, mayor paso dejará el actuador al caudal que proviene desde la primera entrada, es decir, desde el climatizador. En esta situación, una mayor cantidad de agua podrá circular a través de la unidad climatizadora, produciendo una transferencia de calor o frío mayor, para así reducir la diferencia entre la temperatura deseada y la actual en menor tiempo. Por otro lado, a medida que se vaya reduciendo la diferencia entre la temperatura deseada y la medida en la sala, el actuador ofrecerá mayor paso a la entrada del caudal de la bomba B-5.

3.2.2 Bloque 2: Distribución de suelo radiante

Como vimos en la descripción del recinto, disponemos de un edificio principal donde estarán ubicadas tanto las aulas, como la sala de profesores y la secretaría. El sistema de climatización de este edificio está dividido en dos partes. La primera parte será la distribución del suelo radiante y refrescante, será el que expliquemos en este bloque. La segunda parte, será el propio sistema individual empleado en cada aula, sala de profesores y secretaría, que tendrá una dependencia directa de la distribución suelo radiante y explicaremos en el próximo bloque.

En nuestro caso, el edificio principal consta de dos plantas más un sótano, por lo que dispondremos de dos sistemas de distribución de suelo radiante y refrescante. Uno para la planta baja y otro para la primera planta, cuyo funcionamiento es el mismo. En la figura 3.4, podemos observar de forma general cual será el funcionamiento del sistema de distribución de suelo radiante y los elementos que lo componen.

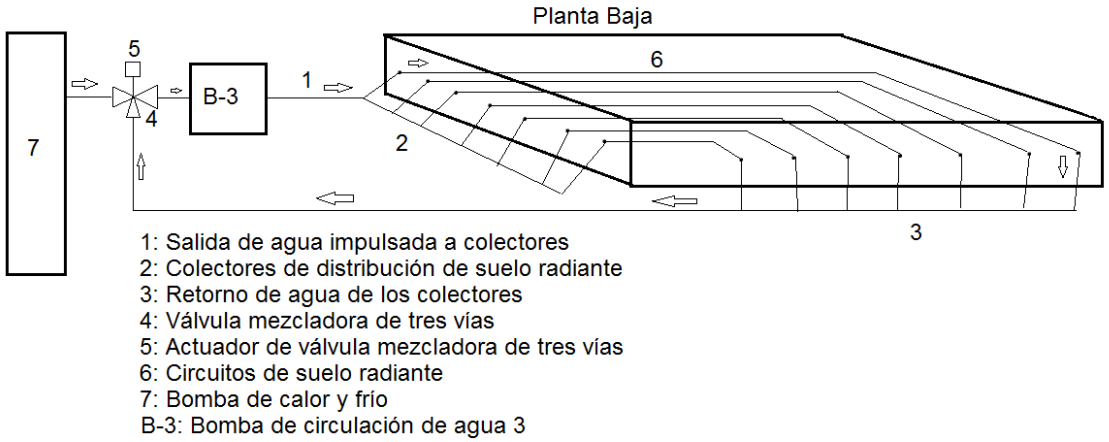


Figura 3.4 *Esquema de funcionamiento del sistema de distribución de suelo radiante.*

Elementos que forman el sistema de distribución de suelo radiante:

La **bomba de calor y frío** se encargará de abastecer a la bomba de circulación de agua B-3 en la planta baja y la bomba de circulación de agua B-4 para la primera planta. En invierno, el agua ofrecida se encontrará a una temperatura entre 32ºC y 37ºC. Sin embargo, en verano el agua ofrecida se encontrará a una temperatura entre 12ºC y 19ºC.

Tras la bomba de calor y frío, nos encontramos ante una **válvula multivía**. Al igual que en el bloque anterior, se trata de una vía mezcladora que dispone de dos entradas y una salida, cuyo flujo será mezcla de ambas entradas. En este caso, la primera entrada, es decir la del lateral izquierdo en la válvula, recibe su caudal de agua desde la bomba de calor y frío. La segunda entrada, desde abajo, recibirá su caudal desde el retorno de los colectores de distribución del suelo radiante. Además, la válvula contará con un servomotor, a modo de actuador, que se encargará de regular el nivel de entrada de cada caudal. Por último, la salida de la válvula se encuentra en el lateral derecho, su flujo será el caudal impulsado hacia los colectores de distribución del suelo radiante y refrescante.

Cada **bomba de circulación de agua,** B-3 para la planta baja y B-4 para la primera planta, se encargarán de impulsar el flujo de agua recibido desde la válvula mezcladora al conjunto de colectores de suelo radiante y refrescante distribuidos bajo el suelo de cada plantas. Como podemos observar, el dibujo dispone de unas flechas con las que podemos comprobar la dirección que sigue el caudal agua a lo largo de nuestro sistema.

Los **colectores de distribución del suelo radiante y refrescante** ofrecerán el flujo de agua recibido por la bomba de circulación a todos los circuitos que componen el suelo radiante y refrescante a lo largo de la planta a climatizar.

El **retorno de los colectores de distribución de suelo radiante y refrescante** recibirá el flujo de vuelta a partir de los circuitos que componen el suelo radiante a lo largo de la planta a climatizar. Además, dicho flujo continuará hasta llegar a la segunda entrada de la válvula multivía, desde abajo, repitiéndose la circulación a lo largo del sistema.

Funcionamiento del sistema de distribución de suelo radiante:

Al igual que en el bloque anterior, el sistema puede funcionar en dos modos, los cuales son:

1. **Modo invierno**, en el que la bomba de calor y de frío se encargará de abastecer con agua caliente, a una temperatura entre 35ºC y 45ºC, a la primera entrada, desde el lateral izquierdo, a la válvula multivía. De tal forma que gracias al funcionamiento del actuador, explicado más abajo, lograremos que durante este modo, el agua a lo largo del sistema de colectores de distribución de suelo radiante circule entre 32ºC y 37ºC.

2. **Modo verano**, en el que la bomba de calor y de frío se encargará de abastecer con agua fría, a una temperatura entre 7ºC y 12ºC, a la primera entrada de la válvula multivía. De tal forma que gracias al funcionamiento del actuador, explicado más abajo, lograremos que durante este modo, el agua a lo largo del sistema de colectores de distribución de suelo radiante circule entre 14ºC y 19ºC.

Por otro lado, dependiendo de la diferencia entre la temperatura de consigna, es decir, la temperatura deseada en los colectores de suelo radiante, y la temperatura medida en su retorno, tendremos un comportamiento diferente dentro de la válvula multivía. En concreto, la posición determinada por el actuador dependerá de esta diferencia.

En la figura 3.5 podemos ver la válvula mezcladora de este bloque, cuyo funcionamiento es el siguiente: durante el invierno, el caudal de la primera entrada, desde la bomba de calor y frío, tendrá una temperatura mayor que el de la segunda entrada, desde el retorno de los colectores de suelo radiante. Sin embargo, en verano el caudal de la primera entrada tendrá una temperatura inferior al de la segunda entrada. En nuestra válvula, el actuador regulará el paso de cada entrada en función de la temperatura necesitada a la salida de la misma. Por ejemplo, cuanto mayor sea la diferencia entre la temperatura de consigna en los circuitos de suelo radiante y la medida a su retorno, mayor paso dejará el actuador a la entrada cuyo caudal viene de la bomba de calor y frío. Por el contrario, a medida que la diferencia se vaya reduciendo, menor paso le dejará a la primera entrada respecto a la entrada del retorno de los colectores.

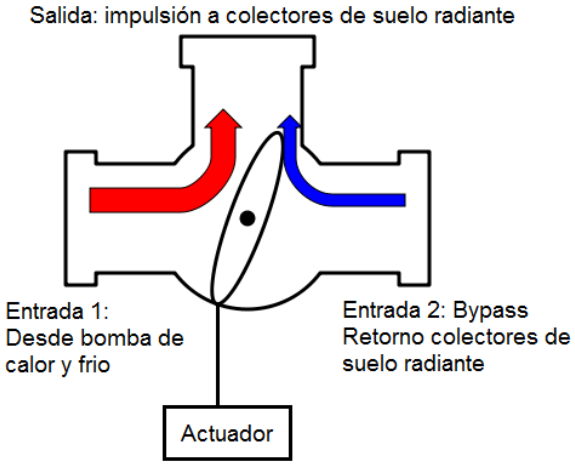
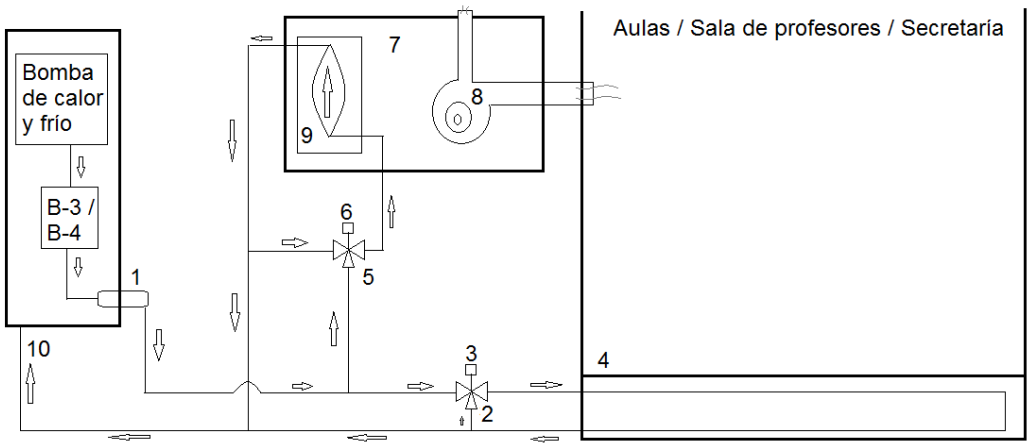


Figura 3.5 *Esquema de funcionamiento de la válvula mezcladora de tres vías del sistema de distribución de suelo radiante.*

3.2.3 Bloque 3: Termostato de dos etapas

Como vimos durante la descripción de las instalaciones, en el edificio principal o aulario, disponemos de dos plantas. La planta baja dispone de diez aulas y una secretaría, mientras que la primera planta dispone de otras diez aulas más la sala de profesores. En este bloque, vamos a tratar el sistema de climatización para cada una de estas salas, en todas igual. En concreto, se trata de un termostato de dos etapas, la primera etapa empleará el suelo radiante que hemos visto en el bloque anterior, y la segunda etapa será un pequeño climatizador auxiliar, como el del bloque uno, pero de menor tamaño. En la figura 3.6 podemos ver su esquema de funcionamiento.



1: Colectores de distribución de suelo radiante
2: Válvula mezcladora de tres vías, primera etapa
3: Actuador de válvula primera etapa
4: Suelo radiante y refrescante
5: Válvula mezcladora de tres vías, segunda etapa
6: Actuador de válvula segunda etapa
7: Climatizador auxiliar
8: Recuperador
9: Unidad climatizadora
10: Retorno colectores de suelo radiante
B-3 / B-4: Bombas de circulación de agua por planta

Figura 3.6 *Esquema de funcionamiento del termostato de dos etapas.*

Elementos que forman el termostato de dos etapas:

El primer elemento de este bloque son los **colectores de distribución de suelo radiante**. Este elemento pertenece al bloque anterior, como vimos, estos colectores se encargan de abastecer a los circuitos repartidos bajo el suelo de cada planta, es decir, se encargan de distribuir el flujo de agua a la temperatura adecuada para el funcionamiento del suelo radiante y refrescante.

Por otra parte, este sistema se trata de un termostato de dos etapas, en el que cada etapa se activará en función del modo en el que nos encontremos y la situación concreta en cada sala.

La **primera etapa del termostato**, dispone de una **válvula multivía**. El esquema de esta válvula mezcladora es el mismo que en los bloques anteriores. Dispone de dos entradas, la primera desde el lateral izquierdo, cuyo flujo viene directamente desde los colectores de suelo radiante. Y una segunda entrada, desde abajo, cuyo flujo viene del retorno del circuito de suelo radiante en el aula en cuestión. A diferencia del bloque que la anterior, la válvula dispondrá de un relé a modo de actuador, ya que el actuador sólo permitirá dos posiciones, como podemos comprobar en las estrategias de control en el próximo capítulo. La primera posición es la que abrirá el paso completamente a la primera entrada, desde los colectores y cerrándolo a la segunda, desde el retorno del circuito del suelo radiante. La segunda posición, cerrará el paso completamente a la primera entrada para abrirlo a la segunda. La salida de la válvula dirigirá un flujo de agua hacia el circuito del suelo radiante para la sala en cuestión.

La **segunda etapa del termostato**, se compone de dos elementos. El primero, será otra **válvula multivía**, idéntica a la de la primera etapa. La primera entrada, desde la izquierda, recibe su flujo desde el retorno del climatizador. La segunda entrada, desde abajo, recibe su flujo desde los colectores de suelo radiante, es decir, el flujo desde el bloque dos que alimenta el bloque actual. Al igual que la anterior, la válvula dispondrá de un relé a modo de actuador, ya que sólo permitirá dos posiciones, como veremos más adelante en las estrategias de control en el capítulo cuatro. La primera posición en la que abrirá el paso completamente a la primera entrada, desde los colectores y cerrándolo a la segunda, desde el retorno del climatizador. La segunda posición, cerrará el paso completamente a la primera entrada para abrirlo a la segunda. La salida de la válvula dirigirá un flujo de agua que alimentará el climatizador.

El segundo elemento será el **climatizador auxiliar**, cuyo funcionamiento es idéntico al explicado en el bloque uno.

Por último, el caudal que circula a través del circuito del suelo radiante de la sala tendrá un **retorno a los colectores** de distribución de suelo radiante y refrescante. De este modo, el agua volverá al sistema del bloque dos, y desde el bloque dos volverán a este sistema, repitiéndose el proceso de nuevo.

Funcionamiento del termostato de dos etapas:

Al igual que en los dos bloques anteriores, el sistema puede funcionar en dos modos, los cuales son:

1. Durante el **modo invierno**, se ofrecerá calefacción a la sala, por tanto en cuanto la temperatura medida en la sala sea menor que la temperatura

deseada, se activará la primera etapa. Además, cuando la diferencia entre la temperatura deseada y la medida en la sala supere un mínimo, establecido en las estrategias de control, se activará el climatizador auxiliar. El climatizador ofrecerá apoyo a la calefacción por suelo radiante.

2. Durante el **modo verano**, se va a climatizar la sala. En este caso se activará cuando la temperatura medida en la sala sea mayor que la temperatura deseada. Al igual que antes, cuando la diferencia entre la temperatura deseada y la medida en la sala supere un mínimo, establecido en las estrategias de control, se activará el climatizador auxiliar. Sin embargo, durante este modo el climatizador tendrá una función adicional además de ofrecer soporte al suelo refrigerante. En verano, podemos sufrir condensación, debido a que la temperatura del suelo radiante será menor que la de la sala. Por tanto, el climatizador se activará en caso de que la humedad de la sala sea superior a un nivel en las estrategias de control. Durante esta situación, el recuperador se encargará de introducir aire en la sala para reducir el nivel de humedad.

El **funcionamiento de la válvula multivía de la primera etapa** tendrá dos eventos. El primero, en el que el circuito de suelo radiante de la sala demande un flujo con una temperatura mayor o menor, dependiendo del modo en el que nos encontremos, ya que la temperatura medida en la sala no coincide con la deseada. Durante este evento, el relé abrirá el paso a la primera entrada, circulando el agua desde los colectores de suelo radiante, a través del circuito para retornar al bloque anterior. De este modo, se aumentará o reducirá la temperatura que circula por el circuito del suelo de la sala gracias al bloque anterior. Durante el segundo evento, la temperatura del flujo de agua que circula por el circuito del suelo de la sala será la adecuada. Se abrirá la segunda entrada, por lo que durante esta situación el flujo se mantendrá en circulación únicamente a través del circuito del suelo y la válvula.

El **funcionamiento de la válvula multivía de la segunda etapa** será muy similar al de la válvula empleada en el primer bloque. Sin embargo, como mencionamos en la descripción de la válvula, más arriba, en este sistema la válvula el actuador sólo permite dos posiciones. Por tanto, en caso de ser necesario activar el climatizador para ofrecer soporte al suelo radiante y refrescante, se abrirá el paso a la segunda entrada cuyo flujo viene desde los colectores del bloque dos y se cerrará el paso al retorno de la unidad climatizadora. El caudal circulará desde los colectores. A través del climatizador para volver al retorno de los colectores, produciéndose la transferencia de calor o frío previamente explicada gracias a la producción del bloque anterior. En el caso contrario, se abrirá el paso únicamente al retorno del climatizador y no se producirá la transferencia previa, ya que el agua circulará a través de la válvula y el climatizador con un flujo que ya produjo su transferencia de calor o frío.

3.2.4 Bloque 4: Intercambiador de calor

Como vimos en el primer apartado de este capítulo, disponemos de un complejo deportivo con un recinto para dos piscinas interiores. En su interior, encontramos dos piscinas climatizadas, una grande y una pequeña. Durante este bloque, vamos a explicar el sistema de climatización

empleado para ambas piscinas, habrá uno igual para cada una. En concreto, este sistema se trata de un intercambiador de calor, en la figura 3.7 podemos ver su esquema de funcionamiento:

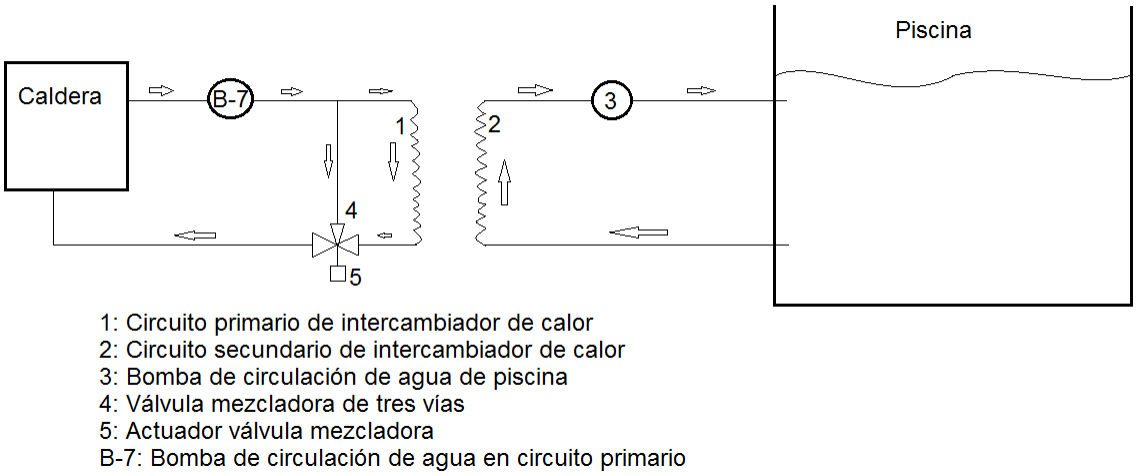


Figura 3.7 *Esquema de funcionamiento del intercambiador de calor.*

Elementos que forman el intercambiador de calor:

El primer elemento que podemos observar se trata de la **caldera**, la cual se encarga de abastecer a nuestro sistema de agua caliente, a una temperatura de 85ºC. En este caso, no empleamos una bomba de calor y frío, ya que este sistema sólo funcionará en un modo, en concreto, en **modo calefacción**, para calentar el agua de cada una de las piscinas.

El elemento principal de nuestro sistema, se trata del **intercambiador de calor**, el cual podemos ver en el centro de la imagen. Está formado por dos circuitos, el **circuito primario**, situado a la izquierda, y el **circuito secundario**, a la derecha. Cabe destacar que estos dos circuitos circulan por conductos separados, nos garantizan que el caudal de agua de cada circuito no se mezcle. Necesitamos que no se mezclen, ya que en el primario el agua que circula no lleva el cloro que circula por el secundario para el mantenimiento de la piscina.

Por otro lado, nuestro sistema dispone de dos **bombas de circulación de agua**. La bomba de circulación B-7, tendrá dos subcircuitos, uno para cada piscina. Esta bomba se encargará de impulsar el flujo de agua a través del circuito primario del intercambiador de calor. Además, cada piscina contará con una bomba de circulación, en concreto, la bomba b-12 para la piscina grande y la bomba b-13 para la piscina pequeña. Estas dos bombas se encargarán de impulsar el flujo de agua a través de la piscina y el circuito secundario del intercambiador.

Además, contamos con una **válvula multivía** idéntica a la de los anteriores bloques, podemos verla en la figura 3.8. Se trata de una válvula mezcladora de tres vías, con dos entradas, una desde el flujo de la caldera y otra tras el paso de dicho flujo por el intercambiador, y una salida, cuyo flujo mezclará el caudal de ambas entradas y retornará a la caldera. La válvula también contará con un servomotor, a modo de actuador, que regulará el caudal proveniente de cada entrada.

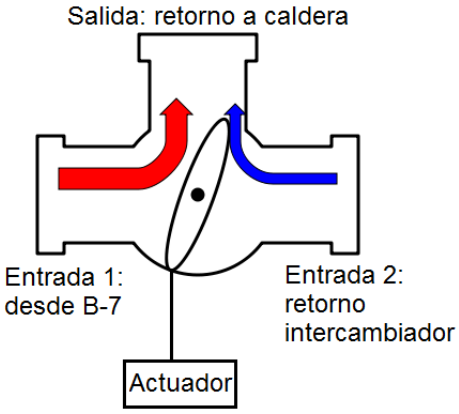


Figura 3.8 *Esquema de funcionamiento de la válvula mezcladora de tres vías del intercambiador de calor.*

Funcionamiento del intercambiador de calor:

El funcionamiento del **intercambiador de calor** es sencillo. Como dijimos antes, dos flujos de agua circulan a través de dos conductos situados en cada circuito del intercambiador, separados por una pequeña distancia. En el circuito primario, gracias a la caldera, circulará agua caliente, mientras que por el circuito secundario fluirá otro caudal, en dirección contraria, que atraviesa la piscina. El agua caliente del circuito primario perderá temperatura, transfiriendo parte del calor perdido al caudal del circuito secundario, calentando el agua que atravesará la piscina.

Por tanto, cuanto mayor sea la diferencia entre la temperatura deseada y la temperatura de la piscina, mayor paso dejará el actuador al flujo de la segunda entrada, es decir, la entrada cuyo flujo atraviesa el circuito primario del intercambiador. De esta forma, mayor caudal atravesará el circuito primario para conseguir una transferencia de calor mayor. Por el contrario, cuanto menor sea esta diferencia, mayor paso dejará el actuador a la primera entrada, desde la bomba de circulación b-7, por lo que un menor flujo atravesará el circuito primario del intercambiador, produciendo una transferencia de calor menor.

3.2.5 Bloque 5: Producción de calor y frío

En este bloque vamos a tratar el último componente del sistema de climatización de nuestro colegio. Se trata del bloque encargado de abastecer tanto de agua caliente en invierno, como de agua fría en verano, a los bloques previamente explicados. Además, como comentamos en la primera parte de este capítulo, la sala que contiene este bloque se encuentra bajo la planta baja del edificio principal. En esta sala encontraremos la bomba de calor y frío más la caldera, mencionadas en anteriores bloques, podemos ver dicho bloque en la figura 3.9.

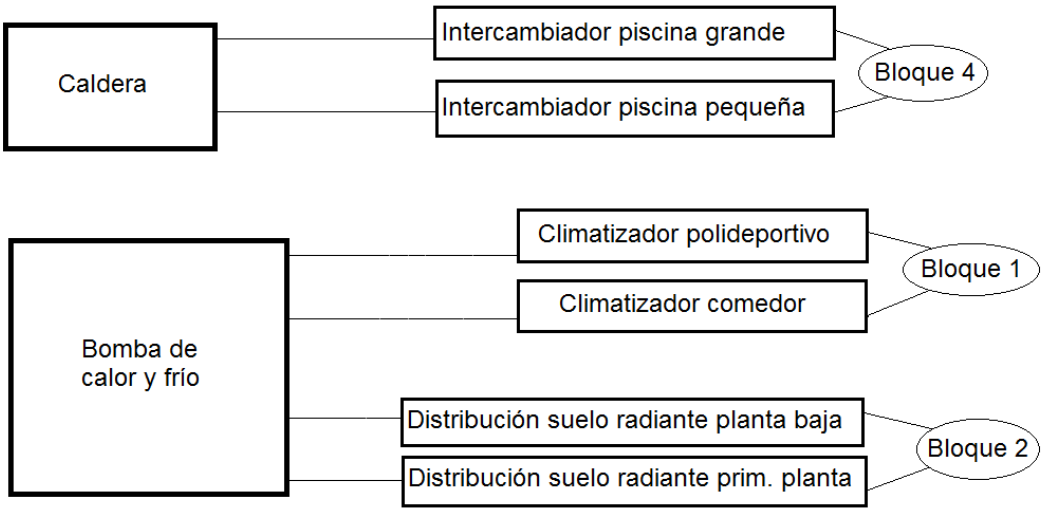


Figura 3.9 *Esquema de funcionamiento del sistema de producción de calor y frío.*

Como vimos en el bloque cuatro, la **caldera** se encargará de abastecer durante todo el año de agua caliente, a una temperatura de 85ºC, a los intercambiadores de calor de cada piscina.

Como vimos en los bloques uno, dos y tres, la **bomba de calor y frío**, se encargará de abastecer de agua caliente, entre 35ºC y 45ºC, a los sistemas de cada bloque durante el invierno. Y de agua fría, a una temperatura entre 7ºC y 12ºC, durante el verano. Cabe destacar que un sistema de un determinado bloque no podrá demandar a la bomba agua fría mientras que otro sistema de otro bloque demande agua caliente. En nuestro caso, la bomba ofrecerá agua caliente o agua fría a todos los sistemas de todos los bloques conectados a ella. De esta forma, será el regulador de cada sistema el encargado de detectar el modo de funcionamiento de la bomba de calor y frío para ejecutar las órdenes correspondientes al modo en el que se encuentre.

Capítulo 4: Análisis de estrategias de control

En este capítulo, vamos a describir las estrategias a que vamos a seguir para el control de los sistemas de climatización de los bloques descritos en el capítulo anterior. A partir de lo descripción de los sistemas de control vista en el capítulo dos, explicaremos aquellos que vamos a emplear para cada uno de los bloques anteriores. Las estrategias de control determinarán las necesidades que debe ofrecer cada regulador a los bloques de climatización vistos en el capítulo anterior. Cada uno de nuestros sistemas de control estará formado por un controlador que coordinará a un conjunto de sensores y actuadores, empleará un controlador de entre los vistos en el capítulo dos. En los siguientes apartados explicaremos en mayor detalle las funcionalidades requeridas para cada sistema de control

4.1 Estrategia para bloque 1: climatizador

En este apartado, vamos a describir las funcionalidades requeridas para el sistema de control encargado de coordinar, dirigir y regular el primer bloque de climatización visto en el capítulo tres, es decir, el climatizador. Como dijimos en el capítulo tres, éste será el sistema de climatización empleado para el polideportivo y para el comedor. Por tanto, habrá dos sistemas de control, uno para el polideportivo y otro para el comedor. En la figura 4.1 podemos ver el esquema del sistema de control para dicho bloque.

El sistema de control empleado para este bloque será un **controlador PID de lazo cerrado**, ya que este bloque requiere un grado de control más sofisticado. Como vimos en el capítulo dos, el controlador PID emplea un algoritmo formado por tres acciones de control, la acción proporcional, integral y derivativa. Para que funcione adecuadamente, será necesario sintonizar correctamente sus respectivas constantes, Kp, Ki y Kd, a la hora de integrar el regulador con el bloque de climatización.

Por otro lado, los elementos que podemos distinguir en el esquema anterior son los siguientes:

- El **regulador o controlador** será un microcontrolador encargado de coordinar, dirigir y monitorizar el conjunto de elementos que forman el bloque de climatización. Por un lado, dispondrá de tres entradas digitales para monitorizar el funcionamiento de los tres aparatos principales del bloque de climatización, pudiendo detectar posibles averías. Estos aparatos son la bomba de circulación b-5, el recuperador y la unidad climatizadora. Las averías serán detectadas de forma general, ya que el controlador sólo podrá detectar una ausencia en su señal de entrada. Por otro lado, dispondrá de dos salidas digitales para encender y apagar el recuperador y la unidad climatizadora. Por último, dispondrá de una salida digital que simulará una analógica mediante PWM para enviar la acción de control determinada por el algoritmo PID al actuador, es decir, al servomotor de la válvula mezcladora.

- La **sonda de temperatura** principal se encargará de realimentar el controlador PID. Esta sonda medirá el aire de la sala donde esté situada, es decir, en el comedor o en el polideportivo. Gracias a estas mediciones, el controlador PID podrá conocer los resultados de sus acciones de control. Además, habrá una sonda de temperatura adicional a la salida de la bomba de circulación de agua. De esta forma, se detectará si va a funcionar en modo invierno o en modo verano para poder ejecutar el algoritmo PID correspondiente a cada modo.

- El **actuador** será el servomotor de la válvula mezcladora vista en la explicación de dicho bloque de climatización. Se encargará de ejecutar la orden de control recibida desde el controlador PID. A través de esta orden, se encargará de regular el paso de las dos entradas de la válvula multivía según el funcionamiento explicado en el apartado de dicho bloque en el capítulo tres. En resumen, para el valor mínimo de la señal PWM, la entrada a la válvula desde el climatizador quedaría cerrada, dejando paso completo a la entrada desde la bomba de circulación. A medida que la señal PWM aumente, mayor paso se dejaría a la entrada desde el climatizador, dejándola paso completo ante el valor máximo de la señal PWM.

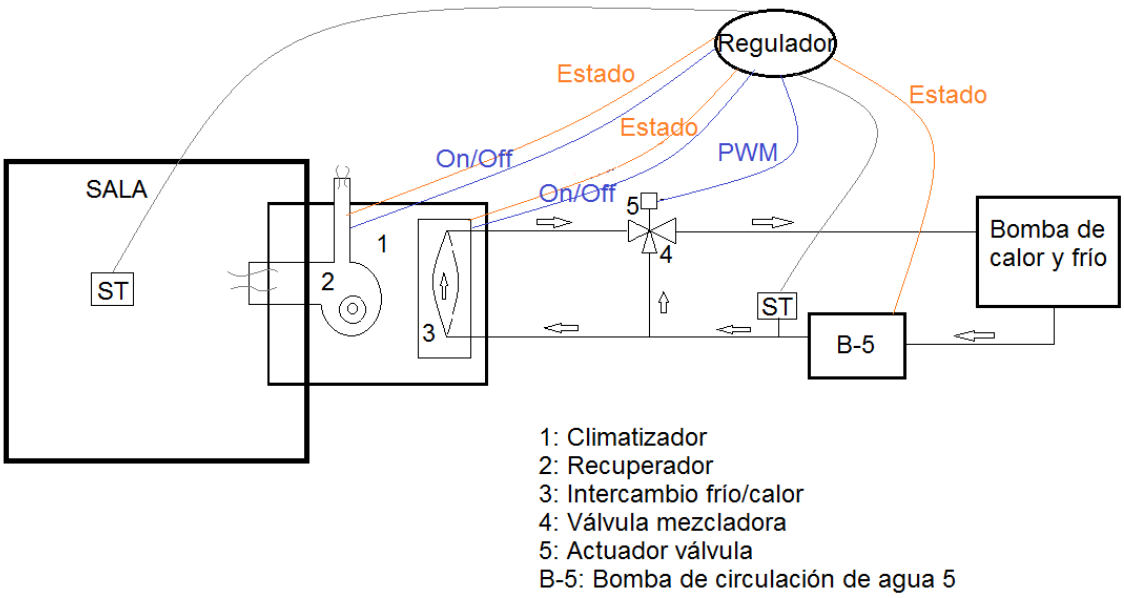


Figura 4.1 *Esquema estrategia de control para climatizador.*

Por último, para proporcionar una interfaz de usuario que permita interactuar con el bloque de climatización, el regulador debe de ser capaz tanto de transmitir información, con respecto al estado del bloque, como de recibir órdenes a través de una interfaz de usuario remota y central. En ella se debe de poder visualizar el estado de la unidad climatizadora, del recuperador y de la bomba de circulación de agua. Además, ofrecerá la última lectura de temperatura de la sonda encargada de medir la temperatura del aire de la sala, así como la temperatura de consigna asignada al bloque. A su vez, dicha interfaz central podrá enviar la orden de encendido o apagado al bloque y podrá modificar su temperatura de consigna. Por otro lado, el regulador debe ser capaz de ofrecer una interfaz auxiliar, de tal forma que muestre los mismos datos que la remota a través de una pantalla auxiliar que se le añada. Sin

embargo, esta interfaz auxiliar sólo permitirá controlar la temperatura de consigna asignada al regulador para este bloque.

4.2 Estrategia para bloque 2: distribución de suelo radiante

En este apartado, vamos a describir las funcionalidades requeridas para el sistema de control encargado de coordinar, dirigir y regular el segundo bloque de climatización visto en el capítulo tres, es decir, de la distribución de suelo radiante y refrescante. Como vimos en el capítulo tres, éste será el sistema de climatización encargado de abastecer los sistemas de suelo radiante situados en las plantas baja y primera del edificio principal. Por tanto, habrá dos sistemas de control, uno para cada planta. En la figura 4.2 podemos ver un esquema del sistema de control para dicho bloque.

Al igual que en el bloque anterior, el sistema de control empleado para este bloque será un **controlador PID de lazo cerrado**, ya que este bloque requiere un grado de control más sofisticado Por otro lado, los elementos que podemos distinguir en el esquema anterior son los siguientes:

- El **regulador o controlador** será un microcontrolador encargado de coordinar, dirigir y monitorizar el conjunto de elementos que forman el bloque de climatización. Por un lado, dispondrá de una entrada digital para monitorizar el funcionamiento de la bomba de circulación de agua correspondiente, pudiendo detectar posibles averías. Las averías serán detectadas de forma general, ya que el controlador sólo podrá detectar una ausencia en su señal de entrada. Por otro lado, dispondrá de una salida digital para poder encender y apagar dicha bomba de circulación. Por último, dispondrá de una salida digital que simulará una analógica mediante PWM para enviar la acción de control determinada por el algoritmo PID.

- La **sonda de temperatura** medirá la temperatura del agua a la salida de la bomba, cuyo flujo se dirige hacia los colectores de suelo radiante y refrescante de dicha planta. Esta sonda, se encargará de realimentar el controlador PID. Gracias a estas mediciones, el controlador PID podrá conocer los resultados de sus acciones de control. Además, el controlador el controlador tendrá una sonda de temperatura adicional a la salida de la bomba de calor y frío. De esta forma, detectará si va se encuentra funcionando en modo invierno o en modo verano para poder ejecutar el algoritmo PID correspondiente a cada modo.

- El **actuador** será el servomotor de la válvula mezcladora vista en la explicación de dicho bloque de climatización. Se encargará de ejecutar la orden de control recibida desde el controlador PID. A través de esta orden, se encargará de regular el paso de las dos entradas de la válvula multivía según el funcionamiento explicado en el apartado de dicho bloque en el capítulo tres. En resumen, para el valor mínimo de la señal PWM, la entrada a la válvula desde la bomba de calor y frío quedaría cerrada, dejando paso completo a la entrada desde el retorno de los colectores de suelo radiante. A medida que la señal PWM aumente, mayor paso se dejaría a la entrada desde la bomba de calor y frío, dejándola paso completo ante el valor máximo de la señal PWM.

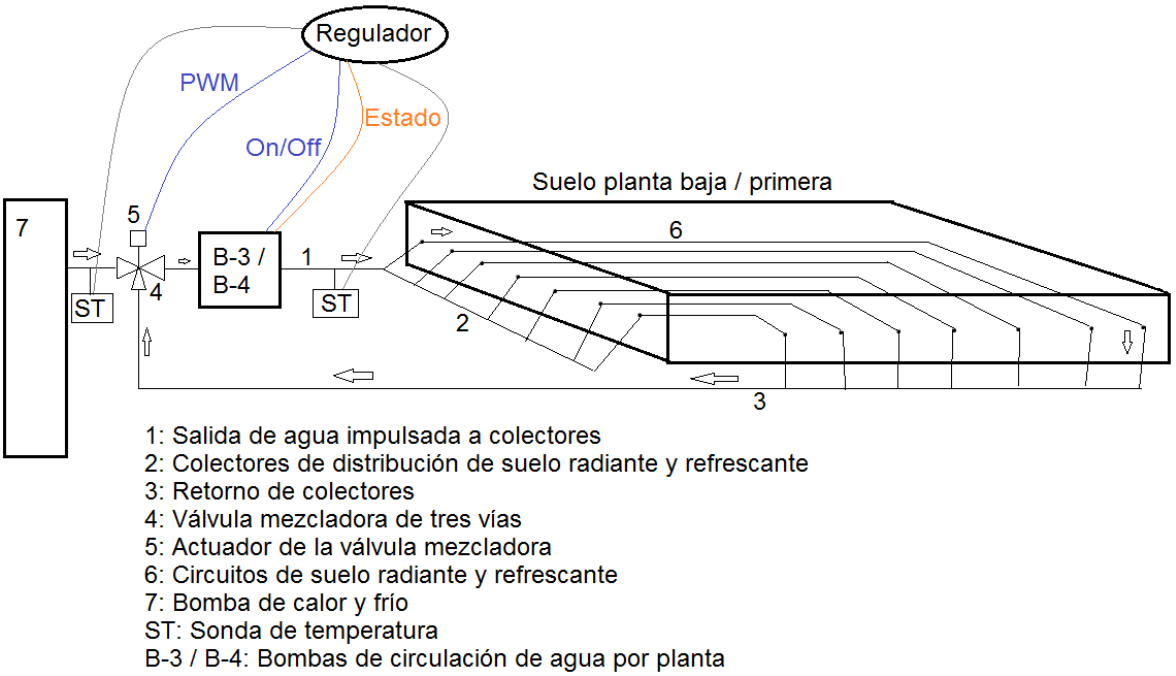
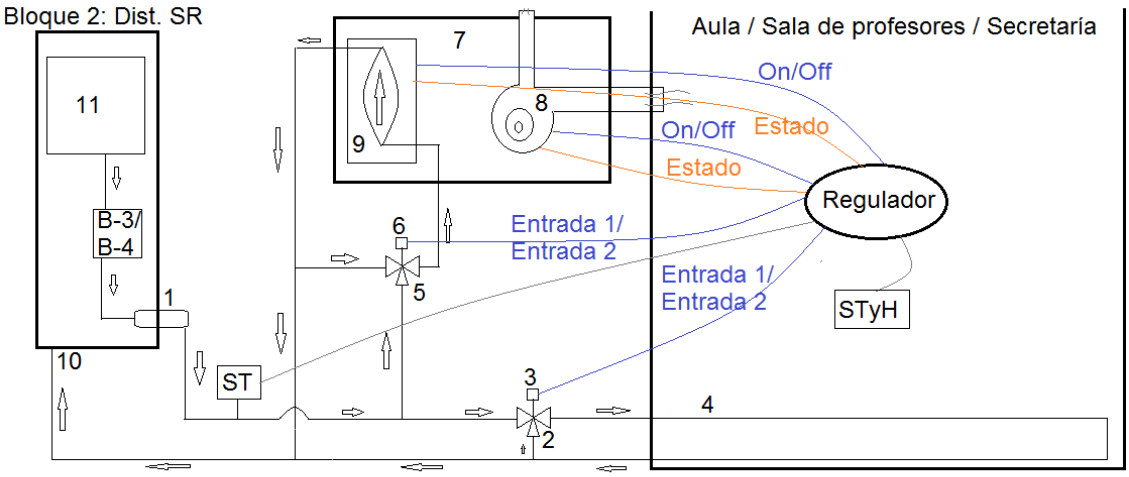


Figura 4.2 *Esquema estrategia de control para distribución de suelo radiante.*

Al igual que en el bloque anterior, para proporcionar una interfaz de usuario que permita interactuar con el bloque de climatización, el regulador debe de ser capaz tanto de transmitir información, con respecto al estado del bloque, como de recibir órdenes a través de una interfaz de usuario remota y central. En ella se debe de poder visualizar el estado de la bomba de circulación de agua de cada planta. Además, ofrecerá la última lectura de temperatura de la sonda encargada de medir la temperatura del agua de los colectores de suelo radiante, así como la temperatura de consigna asignada al bloque. A su vez, dicha interfaz central podrá enviar la orden de encendido o apagado al bloque y podrá modificar su temperatura de consigna. Por otro lado, el regulador debe ser capaz de ofrecer una interfaz auxiliar, de tal forma que muestre los mismos datos que la interfaz remota a través de una pantalla auxiliar añadida al regulador. Sin embargo, esta interfaz auxiliar sólo permitirá controlar la temperatura de consigna asignada al regulador para este bloque.

4.3 Estrategia para bloque 3: termostato de dos etapas

En este apartado, vamos a describir las funcionalidades requeridas para el sistema de control encargado de coordinar, dirigir y regular el tercer bloque de climatización visto en el capítulo tres, es decir, del termostato de dos etapas. Como dijimos en el capítulo tres, éste será el sistema de climatización empleado en cada una de las aulas distribuidas por ambas plantas del edificio principal, de la sala de profesores y de la secretaría. Por tanto, habrá un sistema de control para cada una de estas salas. En la figura 4.3, podemos visualizar el bloque.



1: Colectores de distribución de suelo radiante (B. 2)
2: Válvula mezcladora de tres vías, primera etapa
3: Actuador de válvula primera etapa
4: Suelo radiante y refrescante
5: Válvula mezcladora de tres vías, segunda etapa
6: Actuador de válvula segunda etapa
7: Climatizador auxiliar
8: Recuperador
9: Intercambio calor/frío
10: Retorno colectores suelo radiante
11: Bomba de calor y frío (B.2)
B-3 / B-4: Bomba de circulación de agua por planta (B.2)
ST: Sonda de temperatura
STyH: Sonda de temperatura y humedad

Figura 4.3 *Esquema estrategia de control para termostato de dos etapas.*

El sistema de control empleado para este bloque será un **controlador On/Off doble**, ya que el controlador podrá encender y apagar las dos etapas del termostato. En este bloque no empleamos un controlador más sofisticado por diversos motivos. El primero se a que el bloque de climatización que abastece cada termostato de dos etapas, es decir el bloque dos, distribución de suelo radiante ya emplea un controlador de tipo PID. Dicho sistema de control, junto a la inercia radiante propia de los sistemas de suelo radiante provocará que el caudal de agua que abastece cada termostato no sufra cambios bruscos de temperatura, en un período corto de tiempo, y que la temperatura se mantenga dentro de un rango adecuado. Además, el termostato debe de asegurar que en ningún momento la humedad ambiente de cada sala sobrepase un nivel del 80% en verano, para no sufrir condensación. Esto se logrará con mayor eficacia con un controlador que active en cuanto sea necesario el climatizador auxiliar para reducir dicho nivel de humedad.

Por otro lado, los elementos que podemos distinguir en el esquema anterior son los siguientes:

- El **regulador o controlador** será un microcontrolador encargado de coordinar, dirigir y monitorizar el conjunto de elementos que forman el bloque de climatización. Por un lado, dispondrá de dos entradas digitales para monitorizar el funcionamiento del recuperador y de la unidad climatizadora del climatizador auxiliar, pudiendo detectar posibles averías. Las averías serán detectadas de forma general, ya que el controlador sólo podrá detectar una ausencia en su señal de entrada. Por otro lado, dispondrá de

dos salidas digitales para encender y apagar sendos aparatos del climatizador auxiliar. Por último, dispondrá de dos salidas digitales para poder encender y apagar, es decir, ofrecer paso a una entrada u otra de cada las válvulas mezcladoras que forman las dos etapas del termostato.

- La **sonda de temperatura y humedad** medirá tanto la temperatura del aire de la sala como el nivel de humedad en la misma. Esta sonda, se encargará de realimentar el controlador On / Off. Gracias a estas mediciones, el controlador podrá conocer los resultados de sus acciones de control y activar una o ambas etapas en función de la temperatura y humedad medidas. Además, el controlador el controlador tendrá una sonda de temperatura adicional a la salida de los colectores de distribución de suelo radiante, es decir, a la salida del bloque dos de climatización. De esta forma, detectará si va a funcionar en modo invierno o en modo verano.

- El controlador dispondrá de dos **actuadores**, uno para cada válvula de las dos etapas del termostato. Cada actuador será un relé que admite dos posiciones, ofreciendo paso total a una entrada y cerrándolo a la otra. Diremos que el actuador de la primera etapa estará activado cuando ofrezca paso a la entrada cuyo flujo viene desde los colectores de distribución de suelo radiante, por la izquierda, cerrándolo para la entrada del retorno del suelo radiante, desde abajo. Diremos que está desactivado cuando de paso a la entrada desde el retorno, cerrando la entrada de los colectores. De igual manera, diremos que el actuador de la segunda etapa estará activado cuando ofrezca paso a la entrada desde los colectores de distribución, desde abajo, para cerrarle el paso a la entrada que retorna del climatizador auxiliar, por la izquierda. Este actuador estará desactivado cuando se abra paso a la entrada del retorno, desde la izquierda, para cerrarlo a la entrada de los colectores, desde abajo.

Antes de terminar, vamos a describir las condiciones según el modo de funcionamiento en las que el controlador activará cada una de las etapas del termostato. Durante el **modo invierno**, el controlador activará la primera etapa cuando la temperatura del aire medido en la sala es inferior a la temperatura deseada. Además, si la diferencia entre ambas temperaturas es superior a 2ºC, el controlador activará la segunda etapa. Durante el **modo verano**, el controlador activará la primera etapa cuando la temperatura medida en la sala sea superior a la temperatura deseada. Sin embargo, el controlador activará la segunda etapa en dos situaciones. La primera, cuando la diferencia entre ambas temperaturas sea superior a 2ºC. La segunda, cuando el nivel de humedad medida en la sala supere el 80% en verano, para evitar que se produzca un fenómeno de condensación en el suelo.

Por último, al igual que en los bloques anteriores, para proporcionar una interfaz de usuario que permita interactuar con el bloque de climatización el regulador debe de ser capaz tanto de transmitir información, con respecto al estado del bloque, como de recibir órdenes a través de una interfaz de usuario remota y central. En ella se debe de poder visualizar el estado de la unidad climatizadora y del recuperador del climatizador auxiliar. Además, ofrecerá la última lectura de temperatura y humedad de la sonda encargada de medir la temperatura y humedad relativa del aire de la sala, así como la temperatura de consigna asignada al bloque. A su vez, dicha interfaz central podrá enviar la orden de encendido o apagado al bloque y podrá modificar su temperatura de consigna. Por otro lado, el regulador debe ser capaz de ofrecer una interfaz auxiliar, que muestre los datos de temperatura y humedad relativa del aire de la sala tras realizar la última medición con la sonda, así como la temperatura de consigna asignada al bloque. Además, esta interfaz auxiliar sólo permitirá controlar la temperatura de consigna asignada al regulador para este bloque.

4.4 Estrategia para bloque 4: intercambiador de calor

En este apartado, vamos a describir las funcionalidades requeridas para el sistema de control encargado de coordinar, dirigir y regular el cuarto bloque de climatización visto en el capítulo tres, es decir, del intercambiador de calor de cada piscina. Como dijimos en el capítulo tres, éste será el sistema de climatización encargado de climatizar la piscina grande y la piscina pequeña. Por tanto, habrá dos sistemas de control, uno para cada piscina. En la figura 4.4 podemos ver un esquema del sistema de control para dicho bloque.

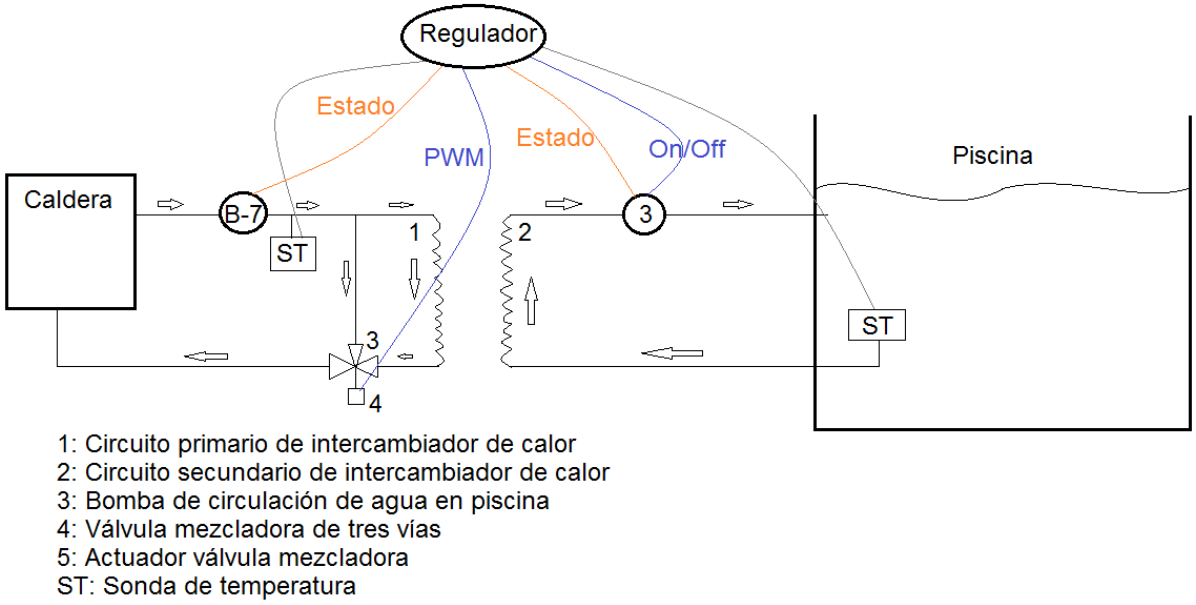


Figura 4.4 *Esquema estrategia de control para intercambiador de calor.*

Al igual que en la estrategia de control para los bloques uno y dos, el sistema de control empleado para este bloque será un **controlador PID de lazo cerrado**, ya que este bloque requiere un grado de control más sofisticado Por otro lado, los elementos que podemos distinguir en el esquema anterior son los siguientes:

- El **regulador o controlador** será un microcontrolador encargado de coordinar, dirigir y monitorizar el conjunto de elementos que forman el bloque de climatización. Por un lado, dispondrá de dos entradas digitales para monitorizar el funcionamiento de la bomba de circulación de agua para cada circuito, pudiendo detectar posibles averías. Las averías serán detectadas de forma general, ya que el controlador sólo podrá detectar una ausencia en su señal de entrada. Por otro lado, dispondrá de una salida digital para poder encender y apagar la bomba de circulación del circuito secundario, es decir, de la piscina. Por último, dispondrá de una salida digital que simulará una analógica mediante PWM para enviar la acción de control determinada por el algoritmo PID.

- La **sonda de temperatura** medirá la temperatura del agua de la piscina. Esta sonda, se encargará de realimentar el controlador PID, cuyas mediciones darán a conocer los resultados de sus acciones de control al microcontrolador. Además, el

controlador tendrá una sonda de temperatura adicional a la salida de la bomba que impulsa el agua de la caldera por el circuito primario. De esta forma, detectará que dicho flujo se encuentre a la temperatura adecuada.

- El **actuador** será el servomotor de la válvula mezcladora vista en la explicación de dicho bloque de climatización. Se encargará de ejecutar la orden de control recibida desde el controlador PID. A través de esta orden, se encargará de regular el paso de las dos entradas de la válvula multivía según el funcionamiento explicado en el apartado de dicho bloque en el capítulo tres. En resumen, para el valor mínimo de la señal PWM, la entrada a la válvula desde la salida del primario del intercambiador quedaría cerrada, dejando paso completo a la entrada desde la impulsión de la bomba de circulación 7. A medida que la señal PWM aumente, mayor paso se dejaría a la entrada desde la salida del primario, dejándola paso completo ante el valor máximo de la señal PWM.

Al igual que en los bloques anteriores, para proporcionar una interfaz de usuario que permita interactuar con el bloque de climatización el regulador debe de ser capaz tanto de transmitir información, con respecto al estado del bloque, como de recibir órdenes a través de una interfaz de usuario remota y central. En ella se debe de poder visualizar el estado de las bombas de circulación de los circuitos primario y secundario. Además, ofrecerá la última lectura de temperatura de la sonda encargada de medir la temperatura del agua de la piscina, así como la temperatura de consigna asignada al bloque. A su vez, dicha interfaz central podrá enviar la orden de encendido o apagado al bloque y podrá modificar su temperatura de consigna. Por otro lado, el regulador debe ser capaz de ofrecer una interfaz auxiliar, de tal forma que muestre los mismos datos que la interfaz remota a través de una pantalla auxiliar añadida al regulador. Sin embargo, esta interfaz auxiliar sólo permitirá controlar la temperatura de consigna asignada al regulador para este bloque.

4.5 Estrategia para bloque 5: producción de calor y frío.

Como vimos en el capítulo tres, éste sistema de climatización estará formado por la caldera y la bomba de calor/frío. Dichos aparatos tendrán la función de abastecer de agua caliente o fría, en función de la temporada, a los bloques de climatización vistos anteriormente.

Se ha considerado que para este bloque no va a ser necesario un sistema de control propiamente dicho, ya que sólo se encargaría de encender o apagar la caldera y la bomba de calor/frío. Además, en función de la temporada, sólo se encargaría de seleccionar el modo de funcionamiento de la bomba generadora de calor/frío. Por tanto, este bloque de climatización no dispondrá más que de un control manual sobre ambos aparatos.

Capítulo 5: Metodología de trabajo

A lo largo de este apartado vamos a detallar distintos aspectos de la metodología a seguir durante nuestro proyecto. En este momento, ya hemos analizado las estrategias de control para controlar cada uno de los bloques de climatización que engloban nuestro proyecto. Vamos a poder concretar los requisitos que debe cubrir nuestro sistema de control, que nos permita seleccionar más adelante, los componentes hardware e interfaces de usuario adecuadas, tanto la tecnología y protocolo de comunicación a emplear como el formato de los mensajes intercambiados en ella, el modelo de desarrollo de software más adecuado y el modelado de los datos para almacenar la información de cada uno del sistema de control a desarrollar. Una vez conocidos todos estos detalles, podemos estudiar los diferentes modelos de proceso software que más se adecuan a nuestro caso y decantarnos por uno de ellos como nuestra metodología de trabajo.

5.1 Modelo de proceso software

A lo largo de este apartado vamos a describir la metodología a seguir para el desarrollo del software de nuestro prototipo para un sistema de control centralizado para el sistema de climatización existente, que cumpla con las estrategias de control propuestas al inicio de este capítulo para cada uno de los bloques de climatización y que cumpla con los requisitos que hemos identificado en el apartado anterior. En concreto, vamos a emplear la metodología del modelo de un proceso software que más se adecue a nuestras circunstancias.

Como podemos comprobar en [3] un proceso software es la colección de actividades de trabajo, acciones y tareas que se van a realizar para un determinado producto. Cada una de estas tareas se engloba dentro de un modelo de desarrollo software. Una estructura general para un modelo de desarrollo software define cinco actividades estructurales: comunicación, planificación, modelado, construcción y despliegue. Además, a lo largo del desarrollo se llevarán a cabo ciertas actividades sombrilla, como la gestión del riesgo, seguimiento y control del proyecto, etc.

Un flujo de proceso define la secuencia u organización de las actividades estructurales del modelo y de las tareas y acciones de cada una de ellas. Existen varios tipos de flujo de proceso. El primero sería un flujo de proceso lineal, que ejecuta las actividades estructurales en secuencia, desde la comunicación hasta el despliegue. Un flujo de proceso iterativo repite una o más actividades antes de pasar a la siguiente. Un flujo de proceso iterativo realiza las actividades de forma “circular” en el que cada vuelta mejora el producto final. Por último, un flujo de proceso paralelo ejecuta una o más actividades al mismo tiempo. Ahora que comprendemos los elementos que forman un modelo de desarrollo software y como se encuentran organizados, vamos a pasar a exponer los modelos de software entre los que elegiremos uno para nuestro caso.

5.1.1 Modelo en cascada

El modelo en cascada suele emplearse en situaciones en las que los requerimientos para un cierto problema pueden comprenderse bien, por ejemplo, en situaciones en las que van a hacerse adaptaciones o mejoras bien definidas en un sistema ya existente, como en nuestro caso, dotar a un sistema de climatización de un sistema de control centralizado.

El modelo en cascada un enfoque sistemático y secuencial, en el que las actividades que lo engloban se suceden de forma lineal. Normalmente, comienza con la especificación de los requisitos del cliente para continuar con la planificación, modelado, construcción y despliegue para terminar con el despliegue software.

Sin embargo, este modelo presenta varias limitaciones, ya que raramente los proyectos reales siguen una secuencia lineal, un cambio puede generar confusión a medida que el proyecto avanza. Además, los clientes raramente expresan todos los requerimientos al comienzo del proyecto y no será hasta el final del proyecto cuando el cliente disponga de un producto funcional. En cualquier caso, como dijimos anteriormente, será un modelo de proceso adecuado en situaciones con requerimientos fijos cuyo trabajo avanza de forma lineal hasta el final. En la figura 5.1 podemos ver el esquema de un modelo en cascada.

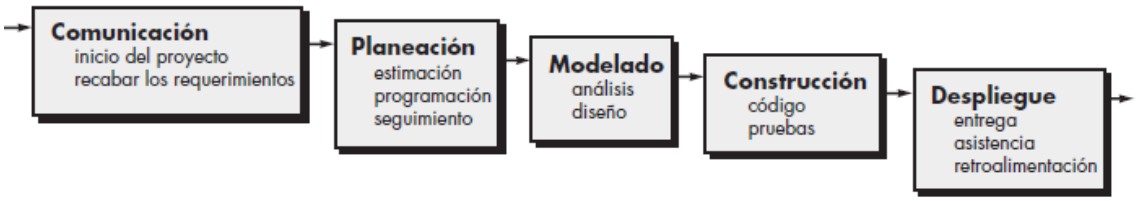


Figura 5.1 *Modelo en cascada, fuente [3].*

5.1.2 Modelos de proceso incremental

Un modelo de proceso incremental se va a emplear en situaciones en las que los requerimientos iniciales del software están razonablemente bien definidos mientras que el alcance del desarrollo no permite una secuencia lineal. Además, se requiere ofrecer rápidamente una cierta funcionalidad limitada de software a los usuarios que se verá aumentada en entregas posteriores.

El modelo de proceso incremental combina los flujos de proceso lineal y paralelo, de tal forma que aplica secuencias lineales de forma escalonada a medida que avanza el desarrollo del proceso. Cada secuencia lineal produce un incremento o entrega de un producto que ya opera, de tal forma que las primeras entregas serán productos desnudos de la versión final, que irán completándose a medida que avanza el desarrollo.

Por último, un modelo de proceso incremental es útil en situaciones en las que se dispone de personal limitado para el desarrollo del proceso software, ya que las primeras iteraciones requieren poco personal para llevarse a cabo. De esta forma, en caso de ser bien recibido el producto básico, puede asignarse mayor personal al desarrollo para futuras iteraciones. En la figura 5.2 podemos ver el esquema seguido por un modelo incremental.

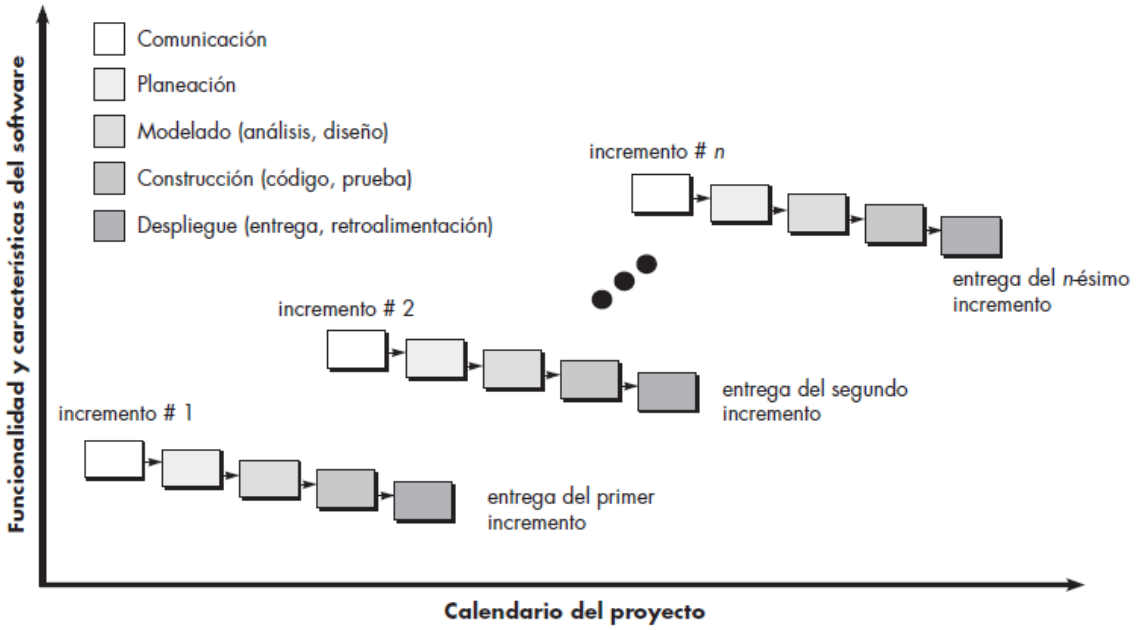


Figura 5.2 *Modelo incremental, fuente [3].*

5.1.3 Modelo de proceso evolutivo

Un modelo de proceso evolutivo se emplea en situaciones en las que los requerimientos del producto cambian a medida que avanza el desarrollo, el cliente define los objetivos generales del producto, pero no identifica sus requerimientos en detalle. En estas situaciones, no podemos avanzar de forma lineal hacia el producto final y se necesita un modelo de proceso que evolucione con el tiempo. El modelo de proceso evolutivo es iterativo, en el que cada iteración resulta en una versión más completa del software final.

Un ejemplo de modelo de proceso evolutivo es el **modelo en espiral**. En un modelo de proceso en espiral. La idea de este modelo es que durante cada iteración se realice un prototipo, que además de ofrecer un producto con cierta limitación, permita definir los requisitos que debe cumplir nuestro producto final, se basa en una serie de entregas evolutivas. Además, cada iteración realiza un prototipo siguiendo un modelo en cascada, que permite hacer un desarrollo rápido de versiones cada vez más completas.

Como podemos ver en la figura 5.3, el modelo en espiral se divide en un conjunto de actividades estructurales que se realizan en cada iteración. De esta forma, la primera vuelta en torno a la espiral resulta en una especificación del producto, a medida que se realizan más iteraciones se va completando. Cada vez que una iteración pasa por la fase de planificación, se ajusta el plan del proyecto, así como la programación de las actividades tras cada entrega. A diferencia de otros modelos de proceso que finalizan cuando se entrega el software final, el modelo en espiral puede aplicarse a lo largo de todo el ciclo de vida del software, en el que cada iteración marca las diferentes etapas del ciclo.

Por último, debido a su capacidad de evolución y adaptación es un modelo con un enfoque bastante realista. Sin embargo, también presenta diversos riesgos, ya que es más difícil de controlar cada una de sus etapas.

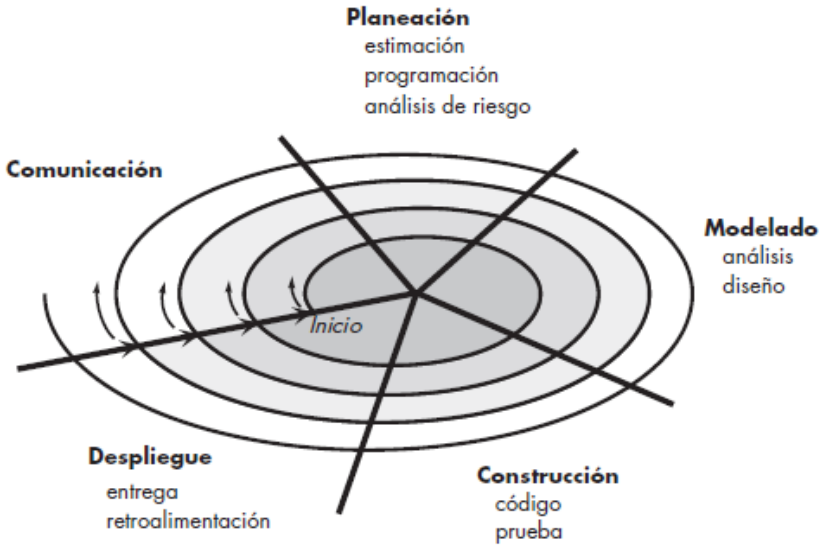


Figura 5.3 *Modelo en espiral, fuente [3].*

5.2 Planificación

Para la planificación de nuestro proyecto vamos a seguir la metodología de un modelo de desarrollo software existente de entre los tres que vimos en el apartado anterior, que nos permita desarrollar un sistema de control centralizado del sistema de climatización del colegio.

Tras valorar los diferentes modelos del punto anterior, vamos a basar nuestra metodología de trabajo en un **modelo de proceso en cascada** que permita dividir la planificación de nuestro proyecto en una serie de actividades estructurales que siguen una secuencia lineal, es decir, que una actividad no va a comenzar hasta que acabe la anterior.

El modelo de proceso en cascada es adecuado para nuestro proyecto ya que se encuentra en una situación en la que se va a añadir una mejora bien definidas en un sistema ya existente, en nuestro caso, dotar a un sistema de climatización de un sistema de control centralizado. Además, esta situación nos va a permitir determinar todos los requisitos necesarios del proyecto antes de desarrollarlo, lo que hace que el modelo en cascada sea adecuado para esta ocasión. No va a ser necesario un modelo más flexible, con mejor adaptación a un proyecto susceptible a cambios en los requerimientos a medida que evoluciona. Por último, el modelo en cascada también será el adecuado ya que no hay urgencia por tener un prototipo del mismo hasta el final del proyecto, ya que el alcance de nuestro proyecto, como vimos en el capítulo introductorio, se sitúa en el desarrollo de un prototipo del sistema de control centralizado, no en su integración.

Ahora que ya entendemos cuáles son los motivos que nos hacen decantarnos por un modelo de desarrollo en cascada, vamos a describir cuáles serán las actividades estructurales que formarán nuestro modelo. Como podemos ver en la figura 5.4, nuestro modelo está formado por seis actividades estructurales, por lo que nuestro proyecto estará comprendido por seis fases, una por cada actividad estructural.

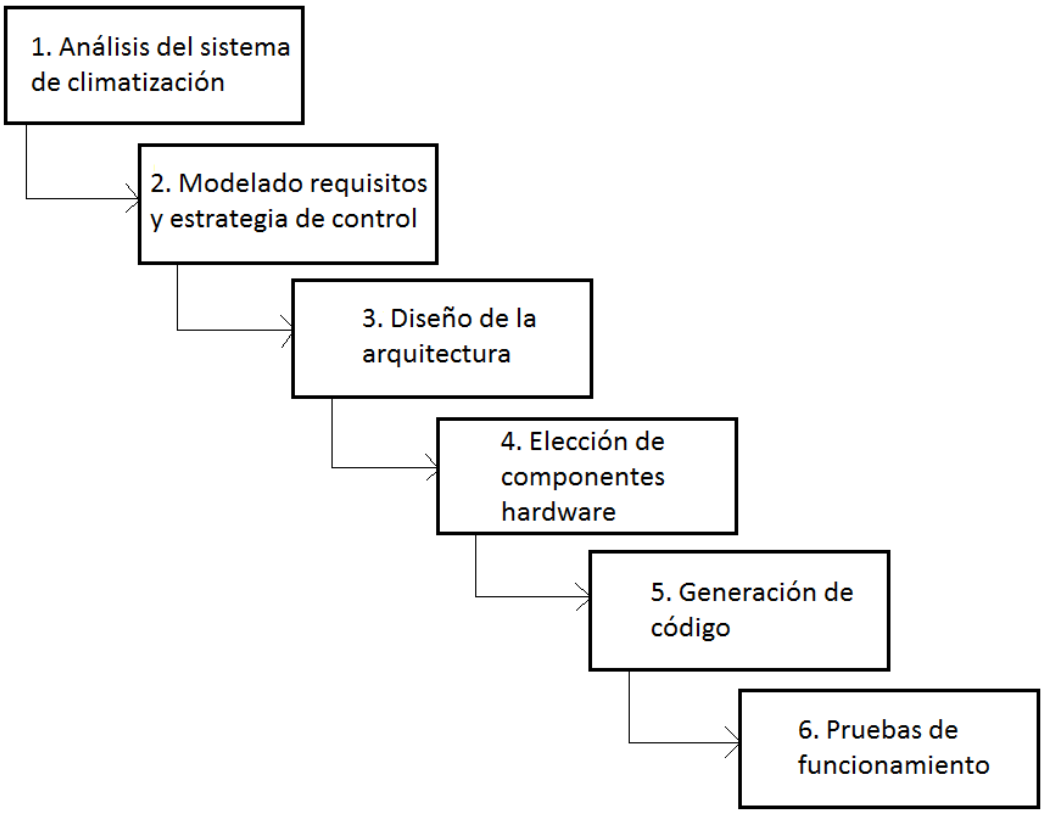


Figura 5.4 *Esquema de las fases del modelo en cascada que emplearemos.*

Gracias a la figura 5.4, podemos comprender el orden en el que se van a desarrollar las actividades de nuestro modelo en cascada. Además, a partir del diagrama que podemos encontrar a continuación, podemos encontrar las tareas a realizar dentro de cada una de las seis actividades estructurales. El objetivo de cada fase es el siguiente:

- **Fase 1: Análisis del sistema de climatización**. A lo largo de esta fase vamos a estudiar el comportamiento de los bloques de climatización del complejo para que nuestro sistema de control se comporte y se adapte a las características de un sistema real.
- **Fase 2: Modelado de requisitos y estrategias de control**. En esta fase, una vez que hemos analizado los bloques de climatización podremos analizar el resto de requisitos del cliente para determinar las estrategias de control de nuestro sistema para cada bloque y los requisitos de los componentes de nuestro sistema.
- **Fase 3: Diseño de la arquitectura**. En esta fase diseñaremos la arquitectura de nuestro sistema de control y de la comunicación entre los elementos que lo forman. Además, determinaremos el formato de los mensajes intercambiados en la comunicación y determinaremos como almacenar la información generada por cada bloque.
- **Fase 4: Elección de los componentes hardware**. En esta fase determinaremos los componentes hardware que van a formar parte de la arquitectura propuesta para nuestro sistema de control, así como los sensores, actuadores u otros elementos para proporcionar una interfaz al usuario que necesitemos.

- **Fase 5: Generación de código**. En esta fase desarrollaremos el software de nuestro sistema de control. En primer lugar, desarrollaremos el software de las unidades de control de cada bloque. En segundo lugar desarrollaremos el software del elemento central y la comunicación entre el elemento central y cada unidad de control. Por último, desarrollaremos una interfaz central para el usuario en la que pueda interactuar con cualquier elemento del sistema de control.
- **Fase 6: Pruebas de funcionamiento**. En esta fase realizaremos una serie de pruebas sobre el software que hemos desarrollado en los componentes hardware que hemos elegido, para comprobar que la funcionalidad requerida se cumple y que la arquitectura que hemos propuesto es válida.

En las figuras 5.5 y 5.6 podemos ver el diagrama Gantt de la planificación seguida, en la que hemos realizado 20 horas semanales repartidas en 5 días. Como podemos ver en el diagrama, el desarrollo del proyecto ha tenido lugar en tres meses y una semana, con un total de 300 horas de trabajo.

En la tabla 5.1, podemos visualizar el reparto de horas que hemos seguido para las actividades que forman cada una de las fases que acabamos de mencionar.

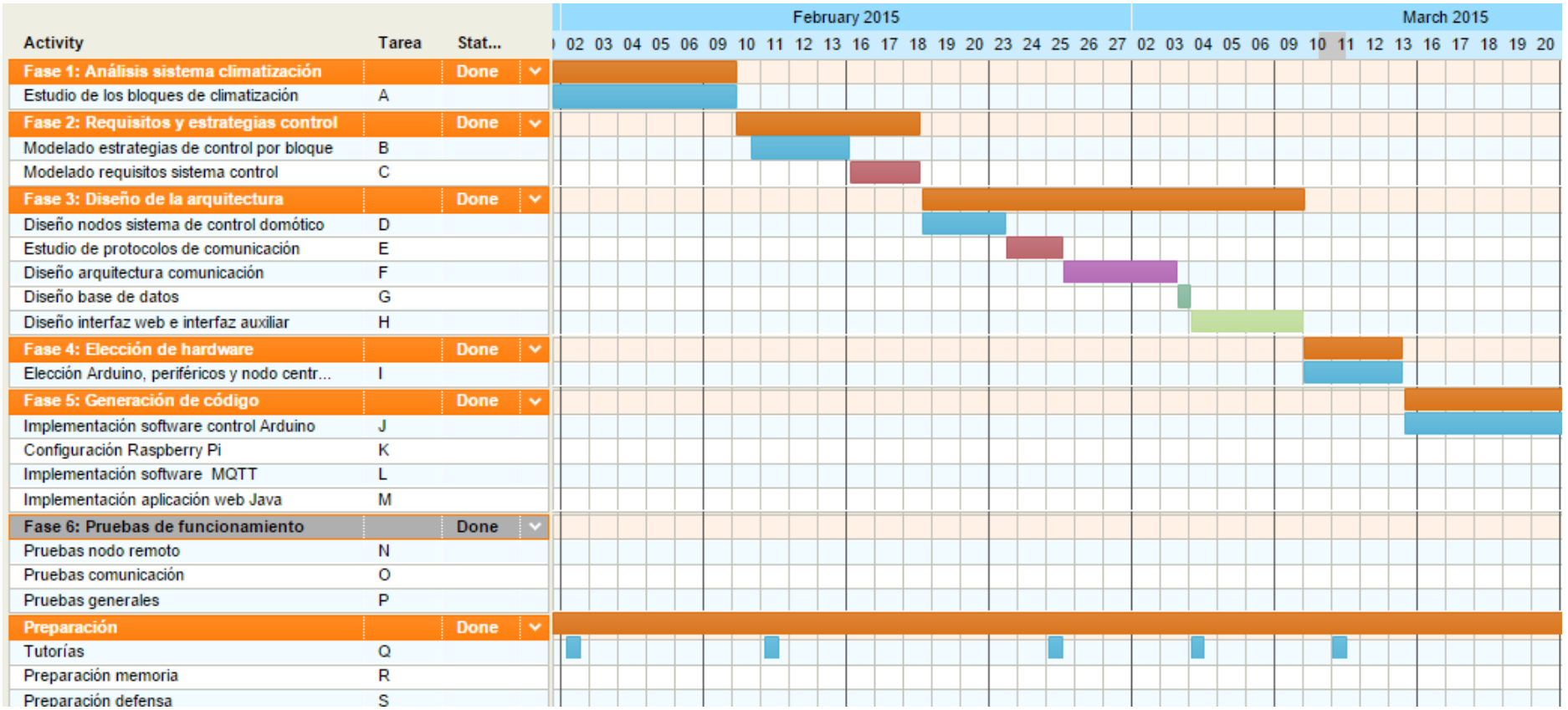


Figura 5.5 *Primera mitad de diagrama Gantt de la planificación seguida.*

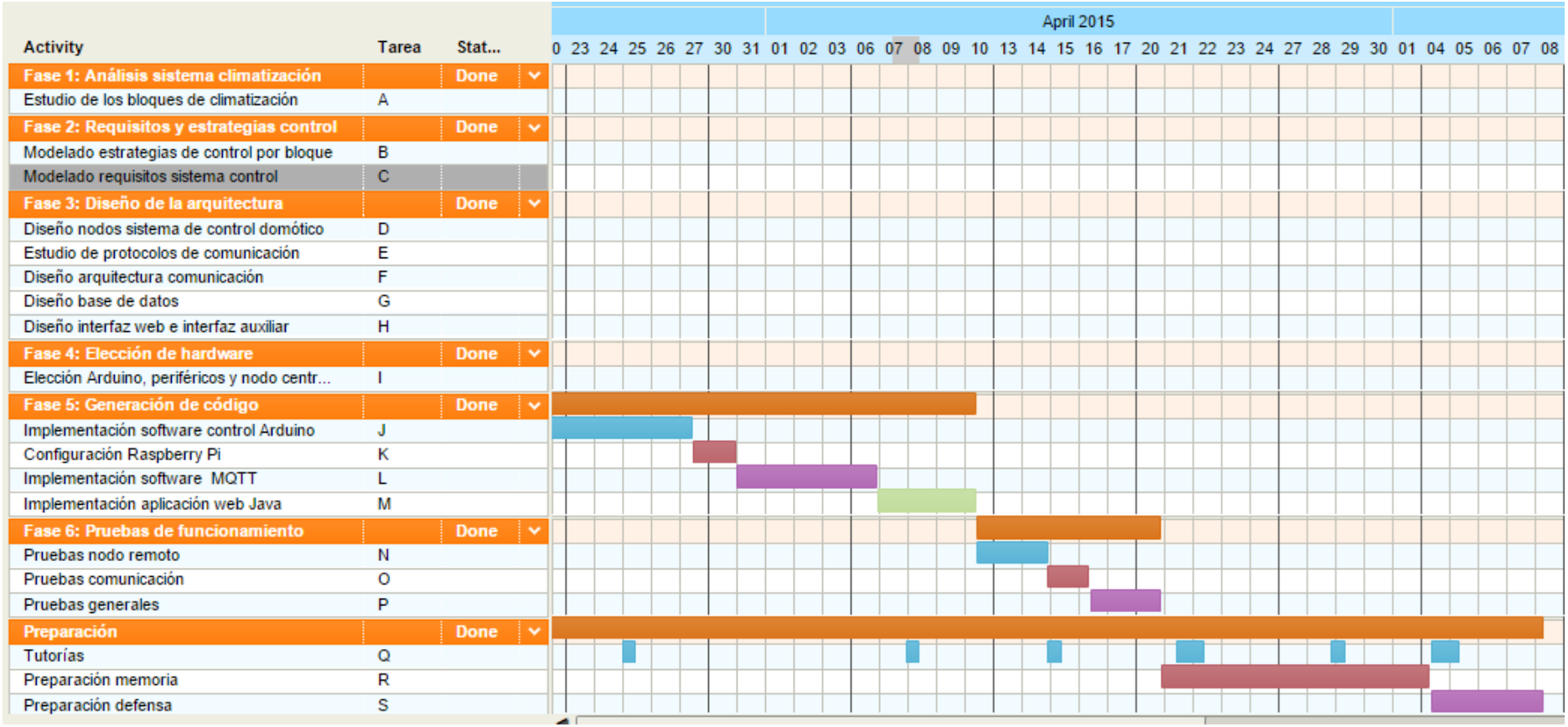


Figura 5.6 *Segunda mitad de diagrama Gantt de la planificación seguida.*

Fase	Tarea	Duración	ID Tarea
Fase 1	Estudio sistema de climatización y requisitos del cliente	25 horas	A
Fase 2	Modelado de estrategias de control para el sistema de climatización	15 horas	B
Fase 2	Modelado de requisitos de componentes del sistema de control	10 horas	C
Fase 3	Diseño nodos del sistema de control domótico	15 horas	D
Fase 3	Estudio de diversos protocolos de comunicación	8 horas	E
Fase 3	Elección de un protocolo y diseño arquitectura de comunicación y formato de mensajes.	15 horas	F
Fase 3	Diseño de la base de datos para almacenado de datos.	2 horas	G
Fase 3	Diseño interfaz web Java e interfaz auxiliar en nodo remoto.	15 horas	H
Fase 4	Elección de placa Arduino, sensores de temperatura y de humedad, actuadores y ordenador de bajo coste.	15 horas	I
Fase 5	Implementación software de control Arduino e interfaz auxiliar.	40 horas	J
Fase 5	Configuración Raspberry Pi, instalación y configuración de software requerido en nodo central.	5 horas	K
Fase 5	Implementación software comunicación, cliente MQTT Arduino y Raspberry Pi e intercambio de mensajes JSON.	20 horas	L
Fase 5	Implementación de la aplicación web Java.	15 horas	M
Fase 6	Pruebas de funcionamiento en nodo remoto, comprobación sensores, actuadores e interfaz auxiliar.	10 horas	N
Fase 6	Pruebas de comunicación entre Arduino y nodo central.	5 horas	O
Fase 6	Prueba de funcionamiento general: aplicación web central y nodos remotos.	10 horas	P
Preparación	Tutorías	25 horas	Q
Preparación	Desarrollo de memoria del proyecto	40 horas	R
Preparación	Defensa oral	10 horas	S
	TOTAL:	300 horas	

Tabla 5.1 *Planificación de las actividades y fases por horas seguida.*

5.3 Requisitos

A lo largo de este apartado, vamos a describir los requisitos que debe cumplir el sistema de control centralizado que vamos a proponer en este proyecto. En este momento, ya sabemos el objetivo principal del proyecto, como vimos en el primer capítulo. Además, ya entendemos el funcionamiento del sistema de climatización del complejo y disponemos de unas estrategias de control para el sistema. Por tanto, si juntamos todo esto más los requisitos específicos del cliente, en la tabla 5.2, podemos concretar el resto de requisitos para un posterior diseño y propuesta del sistema de control centralizado.

Podemos clasificar los requisitos como:

- Requisitos del cliente
- Requisitos de la unidad de control por bloque de climatización
- Requisitos de la comunicación.
- Requisitos del elemento central del sistema centralizado.

5.2.1 Requisitos del cliente

En primer lugar, vamos a analizar las condiciones impuestas por el cliente. En la tabla 5.2 podemos encontrar los requisitos del cliente.

Requisito	Descripción
Plataforma Electrónica Libre	La elección de la unidad hardware de control para cada uno de los bloques de climatización debe situarse entre las plataformas de electrónica libre analizadas en el capítulo dos. Además, se debe elegir un único dispositivo de entre los disponibles para la plataforma elegida que cumpla con los requerimientos vistos en las estrategias de control.
Ordenador de Bajo Coste	La elección de la unidad central de control, encargada de coordinar nuestro sistema de control centralizado, debe situarse entre los ordenadores de bajo coste vistos en el capítulo dos.
Sensor de Temperatura	La elección de un sensor de temperatura debe estar en función de la precisión ofrecida. Además, debemos emplear un único tipo de sensor en todos los bloques de climatización, que sea capaz de medir tanto la temperatura del aire como del agua. El último factor decisivo será el coste por unidad.
Sensor de Temperatura y Humedad	La elección de un sensor de temperatura debe estar en función de la precisión tanto para medir la temperatura como la humedad relativa ambiente. El segundo factor decisivo será el coste por unidad.
Ethernet	La tecnología de comunicación a emplear por el sistema de control debe ser vía Ethernet, aprovechando la red local. Para ello, tanto la unidad de control de cada bloque como la unidad central de control deben ser capaces de comunicarse a través de Ethernet. Por tanto, deben ser capaces de admitir una conexión y configuración para dicha tecnología.
Interfaz Web	La interfaz central de usuario debe ser una aplicación web, que se encuentre alojada en un servidor web local en la unidad central del sistema. La aplicación web local sólo debe ser accesible para el encargado del funcionamiento del sistema de climatización del complejo. A través de ella, podrá visualizar información del estado de cada uno de los bloques de climatización del colegio en un determinado momento, en un espacio reservado para cada uno, que permita identificar unívocamente el bloque a tratar en la aplicación web. Además, permitirá interactuar con cada bloque, es decir, el usuario podrá enviar órdenes a través de la aplicación web a cada uno de los bloques. Además, se deberá añadir una interfaz auxiliar de apoyo a la aplicación web en caso de fallo de la misma. Esta interfaz permitirá visualizar la información necesaria para comprender el estado del bloque en un determinado momento, se encontrará en el mismo lugar que el bloque de climatización. Además, el usuario podrá interactuar con el bloque, pero de una forma más limitada que la interfaz remota de usuario.

Tabla 5.2 *Tabla con requisitos del cliente.*

5.2.2 Requisitos de la unidad de control por bloque de climatización

Requisito	Descripción
Elección	Como vimos en los requisitos del cliente, la elección de la unidad de control para cada bloque se encuentra entre las plataformas de electrónica libre vistas anteriormente. Además, necesitamos que el dispositivo elegido sirva para implementar las estrategias de control de cada bloque, cuya funcionalidad podemos comprobarla en la tabla 5.2. Además, la elección de la plataforma de electrónica abierta se verá determinada por el que más desarrollo y soporte software tenga.
Periféricos	El componente hardware seleccionado como unidad de control de cada bloque debe ser capaz de interactuar con todos los periféricos del mismo. Debe ser capaz de entenderse con ellos vía software y debe admitir tantos dispositivos de entrada y salida como se necesiten. Debe de disponer de tantas entradas/salidas digitales, como entradas analógicas requiera cada bloque de climatización para implementar la estrategia de control propuesta.
Software de Control	El componente hardware debe ser capaz también de implementar vía software el controlador requerido por el bloque, debe recibir información de los sensores correspondientes para enviar órdenes a los actuadores en función de la temperatura deseada y la medida en cierto momento, debe ser capaz de ejecutar un control PID u ON/OFF según la estrategia de control requerida.
Comunicación	El componente de control debe ser capaz de comunicarse, vía Ethernet, con un elemento central, enviando información consistente respecto al estado de su bloque en un determinado momento y siendo capaz de recibir órdenes a través del elemento central. Por tanto, debe ser capaz de entender dichas órdenes para poder asignar una nueva temperatura deseada por el usuario al bloque y al recibir una orden de encendido o apagado del bloque debe ser capaz de enviar las órdenes necesarias a los elementos del bloque para su ejecución.
Averías	El componente de control seleccionado para cada bloque debe ser capaz de encontrar posibles averías tanto en los sensores como en los elementos que forman el bloque, pudiendo reportarlas al elemento central, que las mostrará por la interfaz remota e incluso en ciertos casos en la interfaz auxiliar. Sin embargo, estas averías serán detectadas a un nivel general, ya que el dispositivo sólo detectará una ausencia en la señal de entrada correspondiente al elemento averiado. Será el usuario final quién debe determinar el alcance de la avería, pudiendo ir desde una mala conexión hasta un fallo total del elemento en cuestión.
Interfaz Auxiliar	El componente de control seleccionado para cada bloque debe ser capaz de mostrar los datos a través de una interfaz auxiliar física, que se encuentra situada en el mismo lugar que el bloque en cuestión, para además poder interactuar con ella, pudiendo recibir únicamente órdenes de un cambio en la temperatura deseada para el bloque por el usuario.

Tabla 5.3 Requisitos de unidad hardware de control para cada bloque de climatización.

En segundo lugar, para que podamos implementar las estrategias de control propuestas, vamos a necesitar un componente hardware que ejerza de dispositivo de control por bloque, que nos permita dotar de una mayor inteligencia a cada uno de los bloques de climatización. Por tanto, ahora que conocemos las necesidades que debe cumplir el regulador para garantizar las estrategias de control vistas en el capítulo anterior y conocemos los requisitos del usuario, podemos concretar los **requisitos** que debe cumplir el **componente hardwar**e que seleccionemos como **unidad de control de cada bloque de climatización**, es decir, el cerebro de los bloques de climatización que queremos dotar de un mayor grado de inteligencia. En la tabla 5.3 podemos ver dichos requisitos.

5.2.3 Requisitos de la comunicación

En tercer lugar, como podemos ver en la tabla 5.4 los **requisitos** respecto a la **comunicación** que vamos a emplear en nuestro proyecto.

Requisito	Descripción
Tecnología	La tecnología de comunicación que se va a emplear será Ethernet, ya que viene impuesto por las condiciones del cliente. Todos los elementos que participen en la comunicación deben ser capaces de emplearla.
Protocolo	El protocolo de comunicación a utilizar debe permitir un envío fiable respecto a los mensajes u órdenes enviadas por el elemento central hacia cada uno de los dispositivos de control, debe garantizar el envío y recepción de dichos mensajes. Sin embargo, no será necesario una garantía de envío y recepción de los mensajes de estado por parte de los dispositivos de control. En un sistema de climatización los cambios no pueden ser notificados instantáneamente, son sistemas que tardan un cierto tiempo en actuar. Por ejemplo, cuando en uno de los bloques el usuario asigna una nueva temperatura deseada, el propio bloque tardará un cierto tiempo en poder alcanzar dicha temperatura. Por tanto, si por ejemplo, un controlador de cierto bloque es capaz de enviar diez mensajes del estado del mismo al elemento central, durante el tiempo que tarda el controlador, a través de los sensores, en detectar un cambio en la temperatura del mismo tras un cambio de la temperatura deseada, no va a suponer un problema que no llegue un mensaje del total de enviados durante el intervalo.
Formato de los mensajes	El formato de los mensajes para el intercambio de información debe ser legible por los dos elementos que forman la comunicación. Además, debe de identificar unívocamente tanto las órdenes enviadas del elemento central al dispositivo controlador como los mensajes de estado del bloque recibidos por el elemento central desde el dispositivo controlador, permitiendo identificar cada uno de los campos correspondientes a cada elemento del bloque de climatización y su estado.

Tabla 5.4 Requisitos de la comunicación

5.2.4 Requisitos del elemento central del sistema centralizado

Por último, vamos a describir en la tabla 5.5 los **requisitos** que debe cumplir el **elemento central del sistema de control** propuesto, encargado de coordinar y dirigir todos los elementos de control de cada bloque.

Requisito	Descripción
Ordenador de bajo coste	La elección del elemento central, según las condiciones del cliente, debe de encontrarse entre los ordenadores de bajo coste presentados en el capítulo (referencia). La elección ordenador de bajo coste se decidirá en función de los siguientes factores: en primer lugar, el coste del ordenador. En segundo lugar, debe de garantizar un rendimiento suficiente para realizar todas las actividades que requiere, dichas actividades serán los próximos requisitos, es decir, alojar la aplicación web local y el servidor local de base de datos. Además, debe ser capaz de alojar un software que reciba los datos de cada uno de los dispositivos de control de cada bloque y los procese para su almacenado en la base de datos local. Por último, otro elemento decisivo será el grado de soporte a través de la comunidad que lo rodee.
Aplicación web local	El ordenador de bajo coste elegido debe ser capaz de alojar una aplicación web local que sirva de interfaz remota para el usuario encargado del sistema de control de la climatización. Además, debe de garantizar un rendimiento de la misma adecuado.
Base de datos local	El ordenador de bajo coste elegido debe ser capaz de alojar un servidor de base de datos local, en los que almacene los datos de cada uno de los bloques de climatización en un determinado momento. Además, debe permitir el empleo de dicha información en la aplicación web.
Software de recepción	El ordenador de bajo coste elegido debe ser capaz de implementar un software adicional que reciba los datos aportados por las distintas unidades hardware de control de cada bloque de climatización. Además, una vez recibida dicha información, de forma correcta, debe ser capaz de procesarla e introducirla en la base de datos correspondiente.
Comunicación	El ordenador de bajo coste debe emplear el mismo protocolo que las unidades hardware de control de cada bloque de climatización. Además, debe utilizar, según un requisito del cliente, Ethernet como tecnología de comunicación.

Tabla 5.5 *Requisitos del elemento central del sistema de control.*

Capítulo 6: Arquitectura y diseño del sistema

A lo largo de este capítulo, vamos a explicar la arquitectura propuesta para el sistema de control de los bloques de climatización del complejo. Primero, vamos a explicar la distribución de los elementos principales que componen nuestra arquitectura. Después, explicaremos el protocolo de comunicación que vamos a emplear, incluyendo el formato de los mensajes intercambiados, y el modelado de los datos a emplear en nuestra arquitectura, es decir, el formato de las tablas de la base de datos que vamos a emplear. Por último, describiremos la interfaz de usuario que hemos propuesto para nuestro sistema.

6.1 Nodos

La arquitectura que va a presentar nuestro sistema domótico se trata de una arquitectura mixta, formada por dos niveles, cada nivel presenta una arquitectura de las vistas en el capítulo dos. El primer nivel comprende una arquitectura centralizada y el segundo nivel una arquitectura distribuida.

El primer nivel estará formado por el elemento central de todo el sistema de control que vamos a diseñar, al que a partir de ahora llamaremos **nodo central**, ya que será el elemento central de dicha arquitectura centralizada. Como nodos secundarios de la arquitectura centralizada de este primer nivel encontraremos a cada uno de los elementos hardware de control para cada uno de los lugares de nuestro complejo con uno de los bloques de climatización vistos anteriormente. A partir de ahora, conoceremos dichos nodos secundarios como **nodos remotos**. El segundo nivel estará formado por cada uno de los nodos remotos junto a los sensores y actuadores correspondientes al bloque de climatización de un determinado lugar del complejo, en el que ejercen como unidad de control.

Como vimos en el análisis de requisitos del apartado anterior, una condición impuesta por el cliente era que empleásemos un ordenador de bajo coste como nodo central, en nuestro caso, hemos escogido una **Raspberry Pi, modelo B+** para actuar como nodo central. El nodo central se encargará de coordinar y unificar cada uno de los sistemas de control para cada bloque de climatización en un sistema de control global del complejo. El nodo central tendrá alojado un servidor local de base de datos, dónde almacenará los datos recibidos por cada uno de los nodos remotos. Además, alojará la aplicación web que servirá de interfaz del sistema global de control para el usuario. Desde esta aplicación web, el usuario podrá visualizar el estado de cada bloque de climatización del complejo así como enviarles órdenes. En concreto, podrá enviar órdenes de cambio de la temperatura deseada y apagado o encendido a los nodos remotos. Además, el nodo central alojará un bróker local, encargado de distribuir los mensajes según el protocolo MQTT.

De igual manera, debíamos escoger entre las plataformas de electrónica libre vistas en el capítulo dos nuestro nodo remoto, para ello, hemos escogido la plataforma Arduino, en concreto la placa **Arduino Uno**. El nodo remoto se corresponderá con cada uno de los lugares del complejo que emplean un determinado bloque de climatización. De esta forma, el nodo

remoto será el cerebro del sistema de control del bloque de climatización de un determinado lugar del complejo. En función de las órdenes recibidas a través del nodo central, el modo de funcionamiento del bloque y la temperatura medida por sus sensores, ejecutará una serie de órdenes sobre sus actuadores para cumplir el cometido del bloque de climatización que le corresponde. Estos actuadores que va a controlar cada Arduino Uno serán un servomotor de rotación continua de 180º para las válvulas mezcladoras y relés para los aparatos del bloque. A su vez, los sensores que va a emplear serán sensores digitales de temperatura DS18B20 y el sensor de temperatura y humedad DHT22, en el caso del bloque 3 (termostato de dos etapas). Por último, la interfaz auxiliar de cada nodo remoto dispondrá de una pantalla LCD de tamaño 16x2 junto a un pulsador y un potenciómetro para interactuar con el Arduino. Además, incluirá un módulo Ethernet para la comunicación. En las figuras 6.1, 6.2, 6.3 y 6.4 podemos ver un esquema de los nodos remotos para cada bloque de climatización.

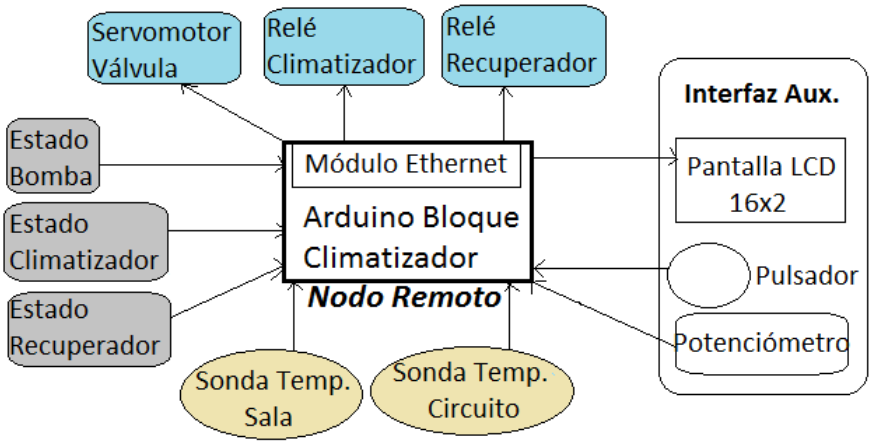


Figura 6.1 *Esquema nodo remoto bloque 1 climatización: climatizador.*

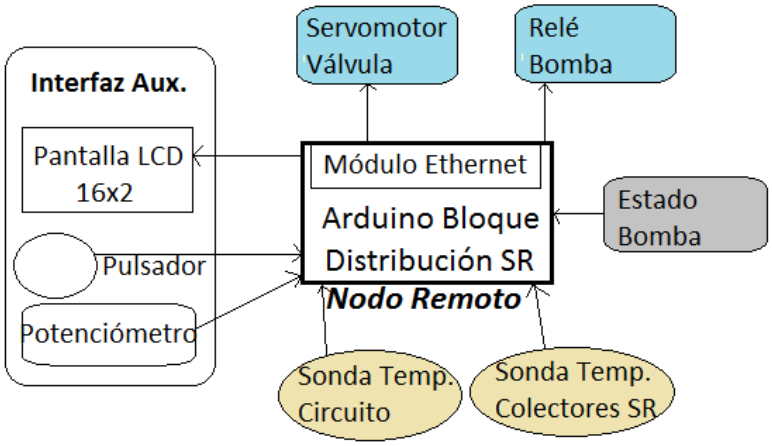


Figura 6.2 *Esquema nodo remoto bloque 2 climatización: distribución de suelo radiante.*

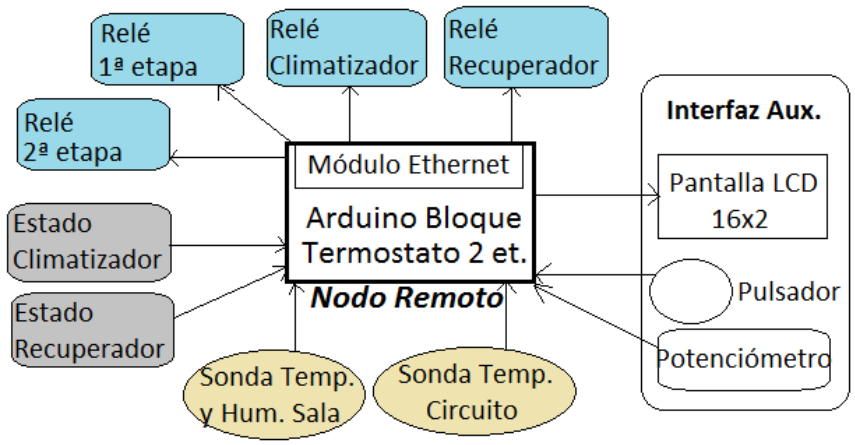


Figura 6.3 *Esquema nodo remoto bloque 3 climatización: termostato de dos etapas.*

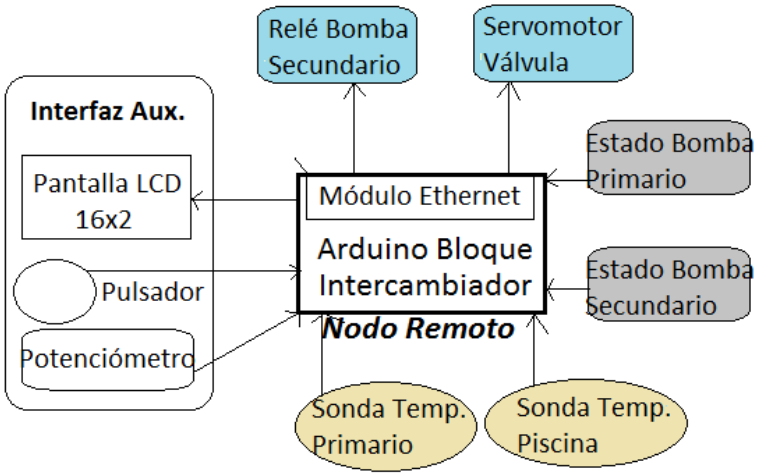


Figura 6.4 *Esquema nodo remoto bloque 4 climatización: intercambiador de calor.*

Por otro lado, vamos a disponer de 18 nodos remotos, que identificaremos por el id que demos a cada una de las placas Arduino Uno. En concreto, los **identificadores de los nodos remotos** serán:

- **Bloque 1: Climatizador**. La placa Arduino Uno del polideportivo tendrá el identificador 1. La placa Arduino Uno del comedor tendrá el identificador 2.
- **Bloque 2: Distribución de suelo radiante**. La placa Arduino Uno de la distribución de la planta baja del edificio principal tendrá el identificador 3. La placa Arduino Uno de la distribución de la primera planta del edificio principal tendrá el identificador 4.
- **Bloque 3**: **Termostato de dos etapas**. Las placas de los Arduino de las cinco aulas de la planta baja tendrán respectivamente los identificadores 5, 6, 7, 8 y 9, mientras que la sala de administración el identificador 10. . Las placas de los Arduino de las cinco aulas de la primera planta tendrán respectivamente los identificadores 11, 12, 13, 14 y 15, mientras que la sala de profesores tendrá el identificador 16.

- **Bloque 4: Intercambiador de calor**. La placa Arduino Uno de la piscina grande tendrá el identificador 17. La placa Arduino Uno de la piscina pequeña tendrá el identificador 18.

La elección de todos los componentes la conoceremos en un mayor detalle en el próximo capítulo. Como veremos en el próximo apartado, La comunicación entre el nodo central y cada uno de los nodos remotos se realizará a través de la red local Ethernet del complejo, empleando el protocolo MQTT que explicaremos a continuación. Por último, en la figura 6.5, podemos ver un esquema de la arquitectura propuesta.

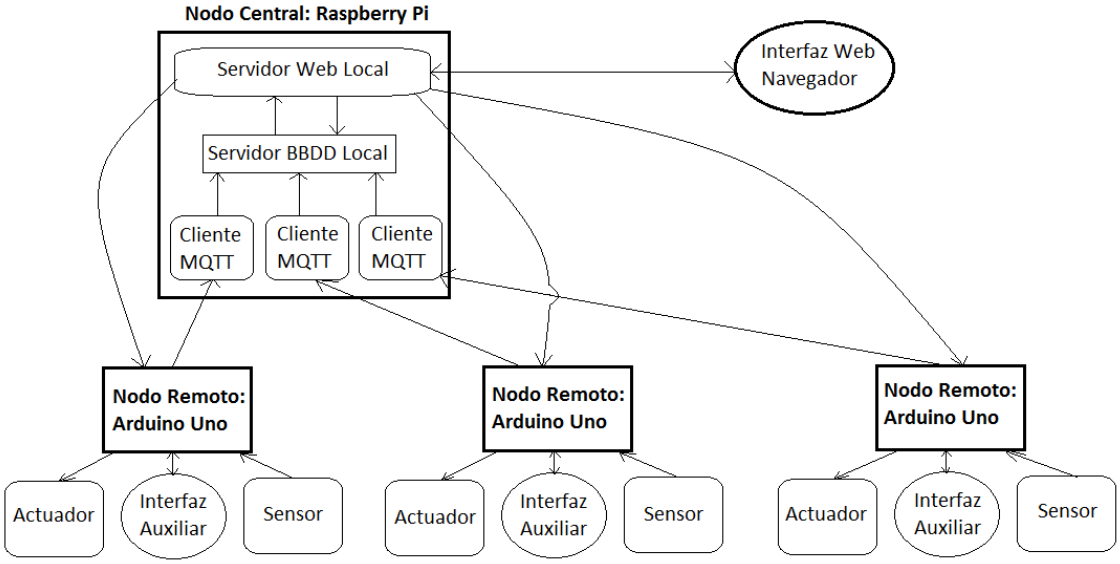


Figura 6.5 *Esquema de la arquitectura propuesta.*

6.2 Comunicación

Ahora que ya entendemos la arquitectura en la que se encuentran situados los nodos de nuestro sistema de control, es momento para entender cómo se produce la comunicación entre ellos. Como dijimos anteriormente, la tecnología de comunicación que van a emplear va a ser Ethernet, cada uno de los nodos tendrá una conexión RJ-45 disponible para conectarse a la red Ethernet de área local de nuestro complejo sobre la que se producirá la comunicación. Además, como mencionamos en el apartado anterior, el protocolo de comunicación. En primer lugar, vamos a explicar el funcionamiento básico del protocolo MQTT de comunicación que vamos a emplear. En segundo lugar, compararemos el protocolo MQTT con otras dos alternativas que habríamos podido elegir. Por último, describiremos la arquitectura de la comunicación propuesta y el formato de los mensajes intercambiados..

6.2.1 MQTT

El término MQTT proviene del inglés “Message Qeueing Telemetry Transport”, según su especificación oficial en [32], MQTT es un protocolo de mensajería que sigue un modelo de “publish/subscribe”. Es un protocolo ligero, de acceso libre y diseñado para que su implementación sea sencilla, lo que hace que sea ideal en situaciones en las que los dispositivos que participan en la comunicación tienen recursos y capacidad limitada.

El modelo “publish/subscribe” ofrece una alternativa al tradicional modelo de cliente-servidor, en el que el cliente demanda un determinado servicio al servidor y éste le envía un mensaje con la respuesta. En el modelo alternativo de “publish/subscribe” existen dos tipos de cliente. Un cliente que envía un determinado mensaje, o **editor**, y uno o más clientes que reciben dicho mensaje, llamados **suscriptores**, de tal forma que ni el editor conoce a sus suscriptores ni viceversa. Además, existe un tercer elemento, denominado “**bróker**”, que conoce tanto a un editor como a sus suscriptores, dado que los suscriptores se van a suscribir a un tema sobre el que va a publicar mensajes un editor. El bróker se va a encargar de recibir los mensajes de un editor y distribuirlos a sus suscriptores. En la figura 6.6 podemos ver un esquema sencillo del funcionamiento del protocolo MQTT, en ella, primero, los clientes suscriptores Raspberry Pi y el terminal móvil Android se suscriben al tema “Temperatura”, a partir de ese momento, cada vez que el editor publique un mensaje, el bróker se encargará de hacérselo llegar.

Por tanto, las principales **ventajas del modelo “publish/subscribe”** son: tanto el editor como el suscriptor no necesitan conocerse, por ejemplo, no necesitan conocer la dirección IP y el puerto de cada uno, sino únicamente la del bróker. La comunicación sigue siendo efectiva aunque el editor y el suscriptor no participen en el mismo momento, ya que se realiza a través del bróker. Además, las operaciones que se estén realizando en un cliente no tienen por qué bloquearse mientras se publica o se recibe un mensaje. Por último, es un modelo mucho más escalable que el modelo cliente-servidor, ya que el bróker puede realizar múltiples operaciones en paralelo, es decir, recibir publicaciones desde múltiples editores y repartirlos entre sus suscriptores, ya que permite almacenar mensajes en espera.

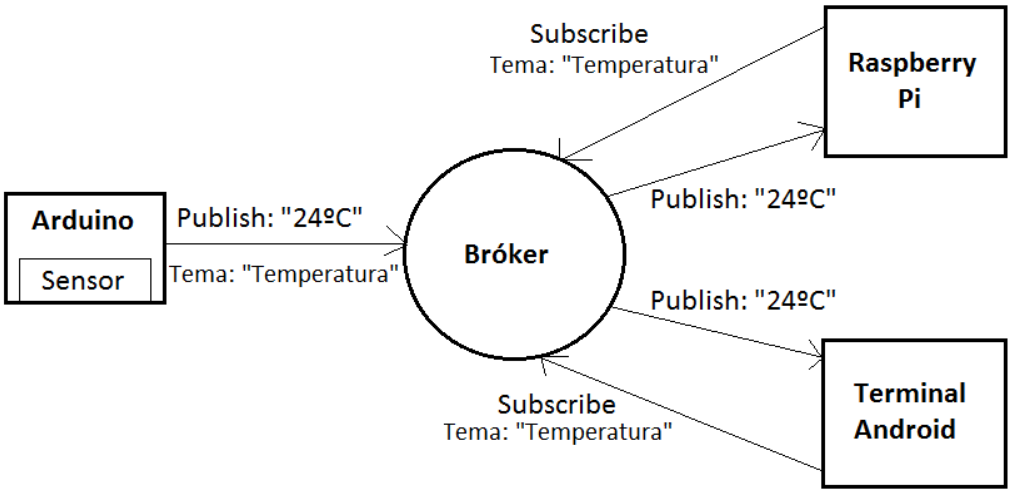


Figura 6.6 *Esquema modelo publish/subscribe de protocolo MQTT.*

En el caso del protocolo MQTT [31], un cliente puede ser cualquier dispositivo, desde un microcontrolador hasta un servidor público, el cual dispone de una librería MQTT en ejecución, es capaz de conectarse al bróker a través de una red y dispone de stack TCP/IP, sobre el que trabaja el protocolo MQTT. Siempre se conectará un cliente con el bróker, nunca se conectarán dos clientes entre sí. La conexión entre el bróker y un cliente siempre será iniciada a través un mensaje de tipo **"Connect"** del cliente al bróker que será confirmado por el bróker. A través de este mensaje "Connect" el cliente le indica ciertos campos que servirán para posteriores procesos.

- Los tres campos obligatorios que debe incluir el cliente son: el identificador de cliente, único por bróker, y el "flag" de sesión. Si el "flag" de sesión está desactivado, el bróker mantendrá las suscripciones previas del cliente y mensajes que no haya recibido en función de la QoS seleccionada. Si el "flag" está activado, el bróker no almacenará ningún dato del cliente y limpiará los datos de sesiones previas con el "flag" desactivado. El último campo obligatorio es el campo de "KeepAlive", que define el tiempo máximo que pueden estar el cliente y bróker sin enviarse un mensaje. El protocolo MQTT usa este mecanismo para asegurar que se mantiene la conexión entre el bróker y el cliente, de tal forma que ambos se enviarán mensajes de "Ping" sin exceder dicho intervalo para mantener la conexión. En caso de exceder el intervalo, el cliente debe iniciar un nuevo proceso de conexión.

- Por otra parte, el mensaje de tipo "Connect" permite añadir unos campos adicionales: permite añadir un usuario y contraseña para autenticar al cliente. Además, permite añadir un último testamento. Este último testamento es un mensaje específico en un tema específico para el cliente que permite notificar tanto al bróker como a otros clientes suscritos al tema del último testamento de dicho cliente que ha perdido la conexión de manera inesperada.

Una vez que un cliente está conectado con el bróker, puede publicar mensajes. Para publicarlo, debe enviar un mensaje de tipo **"Publish"** al bróker. En dicho mensaje, el cliente incluirá el tema al que pertenece el mensaje, para que el bróker dirija el mensaje a los clientes suscritos a dicho tema. La calidad de servicio, "QoS", que va a emplear. El protocolo MQTT nos permite seleccionar tres tipos:

1. Calidad de Servicio 0, que garantiza un servicio de entrega "best effort", es decir, que hará todo lo posible para entregar el mensaje pero no puede ofrecer una garantía del 100%, ya que la entrega del mensaje no será confirmada.
2. Calidad de Servicio 1, que garantiza que el mensaje va a ser entregado por lo menos una vez. Sin embargo, el mensaje puede ser entregado más de una vez, ya sólo confirma el bróker al cliente editor.
3. Calidad de Servicio 2, que garantiza que el mensaje va a ser entregado una única vez al cliente suscriptor. Es la calidad de servicio más segura, pero a su vez la más lenta, ya que necesita una doble confirmación entre cliente editor y bróker.

Además, el cliente editor puede activar el "flag" de conservación del mensaje, que provocará que el bróker almacene dicho mensaje y su calidad de servicio del tema correspondiente. De tal forma que cuando un suscriptor nuevo se suscriba a dicho tema reciba el mensaje conservado por el bróker inmediatamente, sin necesidad de esperar a que el editor publique de nuevo un mensaje. Por último, los mensajes del protocolo MQTT no definen un formato

específico para la información que contienen, ya que se transmite en bytes, por lo que el usuario puede usar el formato que quiera, como una lista de caracteres, XML o JSON.

De igual manera, un cliente puede suscribirse a un tema específico con una calidad de servicio de entre las tres que vimos anteriormente, para recibir mensajes de dicho tema, y podrá cancelar la suscripción en cualquier momento.

6.2.2 Alternativas

Ahora que conocemos en mayor detalle el funcionamiento del protocolo MQTT, antes de exponer la arquitectura de la comunicación que hemos propuesto, vamos a concretar dos alternativas frente al protocolo que hemos seleccionado para así ofrecer al lector los motivos de nuestra elección.

La decisión respecto al protocolo de comunicación que vamos a emplear en nuestra arquitectura ha venido determinada por la capacidad limitada de la plataforma Arduino. Una de las condiciones necesarias para la comunicación que analizamos en los requisitos del capítulo tres fue que la comunicación entre el nodo central y cada Arduino fuese bidireccional. Además, había que garantizar que el envío de órdenes del nodo central a cierto Arduino fueran recibidas. Por tanto, necesitamos un protocolo que fuera ligero, con una implementación sencilla y que nos ofrezca ciertas garantías para determinados mensajes.

Como veremos más adelante en el capítulo siete, la placa Arduino elegida requiere de un módulo adicional con una conexión RJ-45 que permita a Arduino conectarse a una red local Ethernet. Como podemos ver en [39], la librería Ethernet.h para Arduino nos proporciona diversas funcionalidades. Por ejemplo, nos permite emplear nuestro Arduino como cliente web, lo que permite el envío de peticiones HTTP a un servidor. En otro caso, nos permite el envío y recepción de mensajes UDP en nuestro Arduino. Por tanto, estas son las dos alternativas que hemos contemplado:

- Uso del **protocolo HTTP**. Mediante el empleo de este protocolo, nuestro Arduino puede enviar datos respecto a un bloque de climatización al servidor local en el nodo central mediante el uso de una petición POST. En dicha petición, puede incluir los campos correspondientes al estado del bloque. Sin embargo, la creación y envío de dicha petición requiere una mayor capacidad de nuestro Arduino que la publicación de un mensaje en cierto tema en el protocolo MQTT. Por otro lado hay otros dos hechos que hacen que el uso de MQTT requiera una mayor capacidad de nuestro Arduino: el primero, para que un Arduino pueda recibir órdenes a través del nodo central, necesita enviar peticiones cada poco tiempo al servidor para que este le responda con la orden en el momento que se produzca, ya que el modelo cliente-servidor no permite que el servidor inicie la comunicación, mientras que con el modelo que sigue MQTT, como vimos en el apartado anterior, esta situación no se produce. El segundo hecho es la distribución de la información en HTTP es punto a punto, por lo que en caso de que Arduino necesite distribuir el estado de un bloque de climatización a varias entidades interesadas, necesitaría el envío de varias peticiones POST. Sin embargo, el protocolo MQTT permite el envío 1 a 1, 1 a ninguno o 1 a N gracias al modelo “publish/susbcribe”.

- Uso de un **protocolo de diseño propio sobre UDP.** En este caso, gracias a que la librería Ethernet.h de Arduino permite el envío y recepción de mensajes UDP tenemos la posibilidad de diseñar un protocolo propio, creando un diseño de mensajes que se intercambiarían a través de UDP entre cada Arduino y el nodo central. El principal problema que nos presenta este diseño es que UDP no dispone de mecanismos para garantizar el envío y recepción de mensajes. Por ello, para garantizar que las órdenes del nodo central a cierto Arduino sean recibidas correctamente, habría que añadirle una lógica adicional a nuestros Arduinos. El hecho de añadir una lógica adicional requiere una mayor capacidad de nuestro Arduino que no sería necesaria en caso de emplear el protocolo MQTT.

6.2.3 Arquitectura de la comunicación

En este momento ya entendemos el funcionamiento del protocolo MQTT elegido y los motivos por los que lo hemos hecho. Por tanto, podemos presentar en la figura 6.7, el diseño de la arquitectura de la comunicación propuesta.

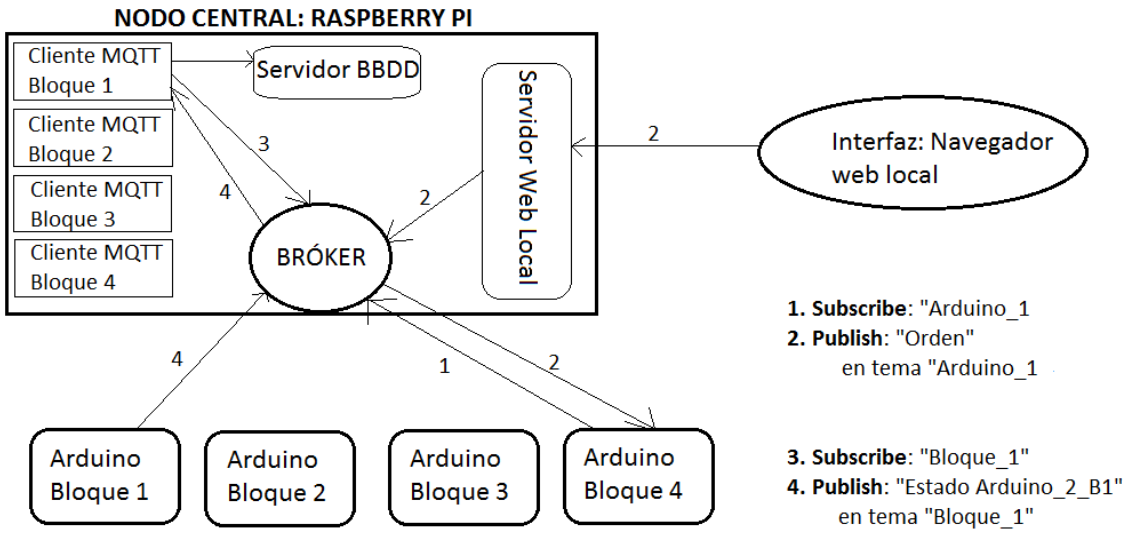


Figura 6.7 *Esquema comunicación propuesta.*

El funcionamiento de la arquitectura de la comunicación que hemos propuesto se divide en dos situaciones:

- **Envío de una orden** desde el navegador web local a un Arduino de un determinado bloque de climatización. En este caso, cada Arduino estará suscrito a su propio tema. El formato del nombre del tema será el siguiente: "Arduino_ID", dónde ID es el identificador del Arduino. Por tanto, cuando quería enviar una orden a un determinado Arduino el servidor web atenderá la petición de dicha orden y a través de una instancia de un cliente MQTT publicará la orden en el tema del Arduino correspondiente con calidad de servicio uno, por lo que el bróker se encargará de que al menos llegue una vez.

- **Envío de un informe de estado** de un Arduino de un determinado bloque al nodo central. En este caso, el nodo central dispondrá de cuatro clientes MQTT en ejecución, uno por cada bloque, suscritos cada uno al tema correspondiente al bloque de climatización. De tal forma que cada vez que un Arduino de dicho bloque publique un nuevo mensaje informando del estado en el que se encuentra, el bróker le entregará dicho mensaje para que el cliente introduzca el informe de estado en el servidor de base de datos local, en la base de datos correspondiente.

6.2.4 Formato de los mensajes

Para la comunicación entre los elementos de nuestro sistema de control vamos a utilizar JSON como formato de nuestros mensajes. El término JSON viene del inglés "JavaScript Object Notation", como podemos comprobar en [33], JSON es un formato de intercambio de mensajes ligero.

JSON está diseñado de tal forma que resulta sencillo de escribir y leer para los humanos y para las máquinas resulte sencillo de generar y analizar. Está basado en una parte del lenguaje de programación JavaScript. Sin embargo, es un formato de texto independiente de cualquier lenguaje, cuyas propiedades son familiares para prácticamente cualquier lenguaje de programación, lo que hace que sea ideal para el intercambio de información. Como veremos en el formato de nuestros mensajes, JSON emplea una notación de clave-valor. Cada mensaje JSON contiene una lista de claves, elegidas por nosotros, con su valor correspondiente.

A continuación podemos encontrar dos tablas. En la tabla 6.1 podemos encontrar el formato de los mensajes de estado que enviará cada uno de los nodos remotos de cada bloque de climatización al cliente MQTT correspondiente del bloque en el nodo central. Gracias a estos mensajes, el cliente obtendrá la información de cierto bloque en un determinado momento para introducirlo a la base de datos.

Por último, en la tabla 6.2 podemos ver el formato de las órdenes enviadas por el usuario a través de la interfaz web a cualquier nodo remoto de un bloque de climatización. Dicho mensaje será enviado por el servidor web al nodo remoto a través del bróker.

Bloque de Climatización	Formato de JSON
1. Climatizador	{ “arduino”: id, “estado”: “on/off”, “climatizador”: “on/off/error”, “recuperador”: “on/off/error”, “bomba”: “on/off/error”, “sonda_circuito”: “on/error”, “temp_sala”: valor, “consigna”: valor; }
2. Distribución de suelo radiante	{ “arduino”: id, “estado”: “on/off”, “bomba”: “on/off/error”, “sonda_circuito”: “on/error”, “temp_colectores”: valor, “consigna”: valor; }
3. Termostato de dos etapas	{ “arduino”: id, “estado”: “on/off”, “climatizador”: “on/off/error”, “recuperador”: “on/off/error”, “sonda_circuito”: “on/error”, “temp_sala”: valor, “humedad_sala”: valor, “consigna”: valor; }
4. Intercambiador de calor	{ “arduino”: id, “estado”: “on/off”, “bomba_cprimario”: “on/off/error”, “bomba_csecundario”: “on/off/error”, “sonda_cprimario”: “on/error”, “temp_piscina”: valor, “consigna”: valor; }

Tabla 6.1 *Formato de mensajes de estado por bloque de climatización.*

Orden	Formato de JSON
Cambio de consigna	{ “arduino”: id, “orden”: 0, “consigna”: valor ;}
Apagado/Encendido remoto	{ “arduino”: id, “orden”: 1, “estado”: 0/1 ;} (0 = Off y 1 = On)

Tabla 6.2 *Formato de órdenes.*

6.3 Modelado de datos

Como mencionamos al final del apartado anterior, en este apartado vamos a describir el formato de los mensajes que vamos a emplear en la comunicación y el formato de las bases de datos que vamos a emplear para almacenar la información del estado de cada uno de los bloques de climatización.

Primero, veremos que es MySQL de forma breve y por qué lo vamos a utilizar. Después, veremos el diseño, es decir, los campos que forman las tablas de la base de datos de nuestro sistema de control centralizado.

6.3.1 MySQL

MySQL es un sistema gestor de bases de datos relacionales [4], es decir, es un programa que nos ofrece soporte al uso de bases de datos relacionales. MySQL es un sistema de acceso libre, podemos descargarlo sin coste alguno desde su página web oficial [34]. Además, es un sistema sencillo de usar, en comparación con otros gestores, y multiplataforma. Por otro lado, hay que mencionar que una base de datos relacional es una base de datos que permite establecer relaciones entre los datos de las tablas que la forman.

Como veremos en el próximo apartado, al final de este capítulo, la aplicación web que vamos a realizar es una aplicación web en Java. Hemos seleccionado MySQL como el sistema gestor de nuestra base de datos ya que es el sistema más utilizado en aplicaciones web Java. Esto es debido a que MySQL nos ofrece una serie de funcionalidades:

1. **Ofrece soporte a SQL**, del inglés “Structured Query Lenguage”, que es un lenguaje estándar para trabajar con los datos almacenados en una base de datos relacional. En nuestra aplicación web, esto nos permite desarrollar aplicaciones Java que usan declaraciones SQL para acceder a los datos almacenados en una base de datos de MySQL.
2. **Ofrece soporte a múltiples clientes**, que pueden acceder a la base de datos, incluso empleando lenguajes de programación diferentes.
3. **Ofrece conectividad a través de una red**, por ejemplo, a través de Internet.
4. **Ofrece seguridad**, ya que sólo permite el acceso a usuarios autenticados.
5. **Ofrece integridad referencial**, es decir, que una fila de una determinada tabla sólo se relacione con otras filas válidas, que existan en la base de datos.
6. **Ofrece soporte a transacciones**, es decir, un conjunto de declaraciones SQL que deben ejecutarse como una unidad.

6.3.2 Diseño de la base de datos

En este apartado, veremos el diseño que hemos seleccionado para las tablas de nuestra base de datos. En este caso, vamos a tener una tabla por cada bloque de climatización, en la que almacenaremos los datos de cada nodo remoto por cada bloque. En la tabla 6.3 podemos encontrar los campos que vamos a emplear en las tablas de nuestra base de datos.

Tabla	Campos de la tabla en BBDD
Bloque 1: Climatizador	**arduino** INT NOT NULL, **estado** VARCHAR(5) NOT NULL, **climatizador** VARCHAR(5) NOT NULL, **recuperador** VARCHAR(5) NOT NULL, **bomba** VARCHAR(5) NOT NULL, **sonda_circuito** VARCHAR(5) NOT NULL, **temperatura** DECIMAL (4,2), **consigna** DECIMAL (4,2), **fecha** TIMESTAMP; Clave primaria: arduino.
Bloque 2: Distribución de suelo radiante	**arduino INT NOT NULL,** **estado** VARCHAR(5) NOT NULL, **bomba** VARCHAR(5) NOT NULL, **sonda_circuito** VARCHAR(5) NOT NULL, **temperatura** DECIMAL (4,2), **consigna** DECIMAL (4,2), **fecha** TIMESTAMP; Clave primaria: arduino.
Bloque 3: Termostato de dos etapas	**arduino INT NOT NULL,** **estado** VARCHAR(5) NOT NULL, **climatizador** VARCHAR(5) NOT NULL, **recuperador** VARCHAR(5) NOT NULL, **sonda_circuito** VARCHAR(5) NOT NULL, **temperatura** DECIMAL (4,2), **humedad** INT NOT NULL, **consigna** DECIMAL (4,2), **fecha** TIMESTAMP; Clave primaria: arduino.
Bloque 4: Intercambiador de calor	**arduino INT NOT NULL,** **estado** VARCHAR(5) NOT NULL, **bomba_cprimario** VARCHAR(5) NOT NULL, **bomba_csecundario** VARCHAR(5) NOT NULL, **sonda_cprimario** VARCHAR(5) NOT NULL, **temperatura** DECIMAL (4,2), **consigna** DECIMAL (4,2), **fecha** TIMESTAMP; Clave primaria: arduino.
Usuarios	**id INT NOT NULL autoincrement,** **nombre** VARCHAR(50), **contraseña** VARCHAR(20); Clave primaria: id.

Tabla 6.3 *Diseño de los campos de las tablas en la base de datos.*

5.4 Interfaz de Usuario

Este es el último apartado para terminar de describir la arquitectura propuesta. En él, detallaremos la interfaz de usuario que vamos a proponer. Primero, explicaremos brevemente qué es una aplicación web en Java y los elementos que la componen para así poder entender mejor el diseño que hemos propuesto. Por último, describiremos la interfaz auxiliar de la que dispone cada nodo remoto.

5.4.1 Aplicación web con Java

Una aplicación web es un conjunto de páginas web generadas en respuesta a peticiones del usuario, emplea un modelo cliente-servidor. Para acceder al contenido de una aplicación web, el cliente accede a través de un navegador web a los recursos de la aplicación almacenados en un servidor. El servidor no es más que un ordenador que se encuentra ejecutando un servidor web que permite el envío de páginas web a los navegadores. El acceso a dicho servidor web puede realizarse a través de Internet o en el interior de una red local.

En nuestro caso, el servidor web se localizará en el nodo central, es decir, en la Raspberry Pi. El software que vamos a emplear es el servidor más común en aplicaciones web Java es "Apache Tomcat", pertenece a la fundación Apache, por lo que es de acceso libre. En el capítulo dos de (referencia a libro Murach) encontramos los pasos a seguir para instalarlo. Por otro lado, como mencionamos anteriormente, el nodo central dispondrá de un servidor local MySQL, que almacenará los datos de nuestro sistema y será accesible para la aplicación web que vamos a emplear. En (referencia a web mysql install) encontramos un enlace con los pasos para instalar dicho servidor MySQL en un entorno Linux. La aplicación web será únicamente accesible a través de la red local del complejo.

La aplicación web que vamos a desarrollar es una aplicación web Java, será la encargada de realizar las tareas de procesado y presentación para que accedamos a la información e interactuemos con nuestro sistema de control a través de la interfaz web en nuestro navegador. En la figura 6.8 podemos ver un esquema de los componentes de una aplicación web en Java, dichos componentes son:

- **Servlet**. Un servlet es un programa Java que se ejecuta en el servidor web y se encargan del procesado de los parámetros de la petición HTTP del navegador web del cliente, del acceso a la base de datos y de la lógica de la aplicación.

- **JSP**. Una página JSP, del inglés "JavaServer Pages" es una página HTML que incorpora elementos dinámicos, entre etiquetas especiales e incluyen fragmentos de código. La página JSP sirve para la presentación de las vistas de la aplicación, de tal forma que el código HTML aparece a la salida del navegador sin modificaciones, pero le añade elementos dinámicos, con pequeños fragmentos de código Java que se ejecutan en el servidor al momento de generar la respuesta.

- **Java Beans**. Un Java Bean es una clase Java que cumple el siguiente convenio: contiene propiedades, es decir, atributos de instancia, privados, que se acceden a través de "get", "set" e "is" y tiene un constructor nulo.

Un Java Bean se emplea en las aplicaciones web para representar un determinado objeto. Por ejemplo, como veremos a continuación, podemos definir un Java Bean como una clase que representa un Arduino de un determinado bloque de climatización, con sus características como atributos de instancia.

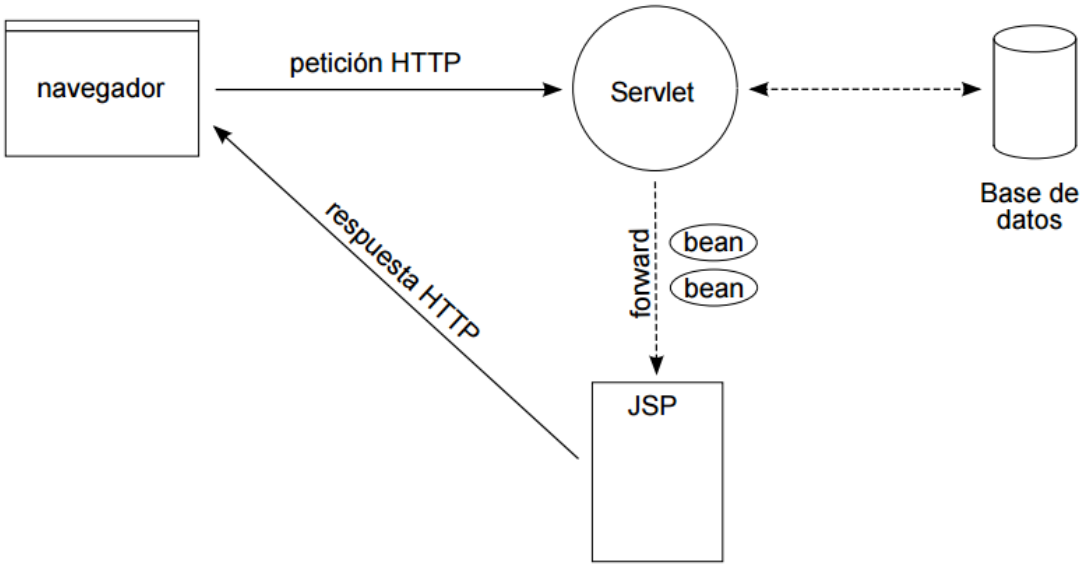


Figura 6.8 *Esquema de la arquitectura de una aplicación web Java, fuente [5].*

En el capítulo nueve, veremos más en profundidad la implementación de la aplicación Web Java. Sin embargo, la arquitectura de la aplicación Web Java que proponemos tiene la siguiente estructura:

- Dispone de **tres vistas**, las cuales son:
    1. **Login**, que sólo aparece cuando el usuario no está autenticado. Únicamente el usuario encargado del sistema de control propuesto podrá autenticarse, ya que será el único usuario registrado en la tabla Usuarios de la base de datos y será el único que podrá visualizar el resto de vistas de la aplicación. Habrá una página JSP encargada de mostrar dicha vista.

    2. **Pantalla principal**, que aparece una vez que el usuario se ha autenticado correctamente. En esta pantalla aparecerán cinco columnas. Cada columna se corresponde con un bloque de climatización, menos los elementos del bloque tres, que al ser más, se dividen en dos columnas. Cada columna tendrá una lista, en la que cada fila se corresponderá con un lugar del complejo que utiliza dicho bloque de climatización. Cada fila nos llevará a la vista de bloque de dicho lugar. La primera columna tendrá dos filas que nos enlazará a la vista de bloque de ambos lugares, una para el comedor y otra para el polideportivo, que emplean el primer bloque de climatización. La segunda columna tendrá también dos filas, una para la distribución del suelo radiante de la primera planta del edificio principal y otra para la segunda planta, ambas nos enlazan a

su vista de bloque. La tercera columna tendrá seis filas, se corresponde con las cinco aulas y la sala de administración de la primera planta del edificio principal, emplean el tercer bloque de climatización. La cuarta columna tendrá también seis filas, para los lugares de la segunda planta que emplean el tercer bloque, es decir, las cinco aulas de la segunda planta y la sala de profesores. La quinta y última columna tendrá dos filas, una para cada piscina, que emplean el cuarto bloque de climatización. Habrá una página JSP encargada de mostrar dicha vista.

3. **Vista de bloque**. A esta vista podemos acceder a través de cualquiera de las filas de las columnas que encontramos en la pantalla principal. En esta vista encontraremos la información de cada uno de los nodos remotos, que se corresponden, como vimos al comienzo de este capítulo, con cada uno de los lugares que emplea un determinado bloque de climatización. En ella encontraremos toda la información del estado actual de un bloque de climatización de un determinado lugar del complejo. Además, a través de esta vista podremos cambiar la temperatura deseada del bloque de climatización de dicho lugar e incluso apagarlo o encenderlo. Habrá un total de cuatro JSP, uno por cada bloque de climatización, que será común entre los lugares que empleen dicho bloque.

- Los **Servlets** de la aplicación web Java serán los siguientes:

    1. **Servlet de Login**, encargado de autenticar al usuario, para ello comprobará que los datos introducidos en el formulario de Login son los mismos que en la tabla Usuarios de la base de datos local. En caso de autenticarse correctamente, será reenviado a la vista principal. En caso de fallo, no pasará de la vista de Login.

    2. **Servlet principal**, encargado de mostrar la información necesaria en la vista principal.

    3. **Servlet de bloque,** encargado de atender las peticiones del usuario desde la pantalla principal que desea cargar la vista de un lugar de un determinado bloque de climatización. Este Servlet recuperará el estado del lugar solicitado del último informe existente de la tabla del bloque en la base de datos local. Una vez recuperados los datos, se los comunicará a la vista correspondiente encargada de mostrarlos.

    4. **Servlet de órdenes,** encargado de atender las peticiones del usuario desde la vista de bloque al enviar una orden al bloque de climatización del lugar de la vista. Una vez completada la orden, redirigirá la petición al Servlet de bloque correspondiente para que cargue de nuevo la vista adecuada.

- Por último, habrá un **Java Bean** por cada uno de los cuatro bloques de climatización de los lugares del complejo, es decir, por cada tipo de nodo remoto. Además de otro por el usuario. Los atributos de dicho Java Bean los podemos ver en la tabla 6.4.

Nombre de Java Bean	Atributos de instancia(privados)
Nodo_Bloque1	• **int** arduino; • **String** estado, climatizador, recuperador, bomba, sonda_circuito; • **double** temperatura, consigna;
Nodo_Bloque2	• **int arduino;** • **String** estado, bomba, sonda_circuito; • **double temperatura, consigna;**
Nodo_Bloque3	• **int arduino, humedad;** • **String** estado, climatizador, recuperador, sonda_circuito; • **double temperatura, consigna;**
Nodo_Bloque4	• **int arduino;** • **String** estado, bomba_cprimario, bomba_csecundario, sonda_cprimario; • **double temperatura, consigna;**
Usuario	• **String nombre, contraseña, token;**

Tabla 6.4 *Lista de Java Bean y sus atributos.*

En la figura 6.9 podemos ver un esquema de los casos de uso de la aplicación web que proponemos como interfaz central de usuario para nuestro sistema de control:

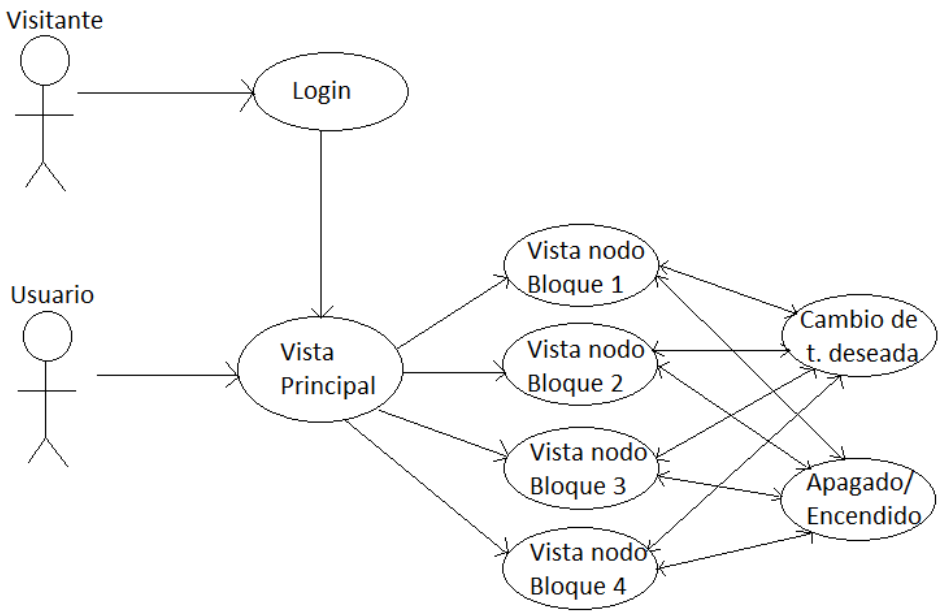


Figura 6.9 *Casos de uso de la aplicación web Java.*

6.4.2 Interfaz Auxiliar

En cada uno de los nodos remotos correspondiente a un lugar del complejo que emplea un determinado bloque de climatización concretamos que se dispondría de una interfaz auxiliar. Esta interfaz auxiliar permite que sea más accesible a otros usuarios el interactuar con el bloque de climatización de un lugar, por ello está limitada a solo poder enviar órdenes de cambio de la temperatura deseada al bloque, y nunca orden de encendido u apagado.

Esta interfaz auxiliar es útil en determinados lugares, como en las aulas, ya que permiten que el usuario que se encuentra en un determinado momento en dicho lugar pueda modificar la temperatura deseada sin necesidad de emplear la interfaz web. Además, en caso de fallo de la interfaz web, nos permitirá visualizar los datos del bloque de climatización de un determinado lugar.

Como podemos observar en las figuras 6.1, 6.2, 6.3 y 6.4, es decir, los esquemas de cada tipo de nodo remoto, la interfaz auxiliar está formada por una pantalla LCD de 16x2 píxeles, que veremos en mayor detalle en el próximo capítulo, un pulsador y un potenciómetro. La pantalla LCD nos mostrará la información actual de dicho bloque de climatización. El pulsador nos permitirá encender y apagar la luz de fondo de la pantalla LCD, permitiendo apagarla en el momento que no se vaya a usar, para ahorrar energía. Y el potenciómetro, como veremos en el próximo capítulo en mayor detalle, nos permitirá escoger (dentro de un rango predeterminado) una temperatura deseada en dicho bloque de climatización.

Capítulo 7: Componentes Hardware

En este capítulo, vamos a explicar los componentes hardware que vamos a emplear en nuestro proyecto. En el primer apartado comprenderemos en mayor detalle el hardware de la placa Arduino Uno que vamos a emplear, así como el hardware del shield Ethernet que va a incorporar. Además, podremos ver la asignación de los pines que vamos a usar para cada una de estas placas. En el segundo apartado, veremos en mayor detalle los sensores de temperatura y humedad escogidos, comparándolos con otras alternativas vistas en el capítulo dos. En el tercer apartado, describiremos los actuadores que hemos empleado. Por último, detallaremos en mayor medida el modelo de Raspberry Pi escogido como nodo central.

7.1 Arduino Uno

La placa electrónica de la plataforma Arduino que vamos a emplear en nuestro sistema es la placa Arduino Uno [49], como vimos en el capítulo dos en la descripción de la plataforma Arduino, esta placa se trata de una de las placas Arduino más utilizadas, ya que aunque la capacidad de su microcontrolador es más limitada que la de otras placas, nos ofrece un funcionamiento más que adecuado para la mayoría de proyectos a un precio muy atractivo para el usuario. En la figura 7.1 podemos observar un esquema detallado de la placa Arduino Uno. El uso que vamos a darle, como dijimos en el apartado uno del capítulo anterior, será el de coordinar y dirigir el sistema de control empleado en cada lugar que dispone de un determinado bloque de climatización, es decir, en cada nodo remoto.

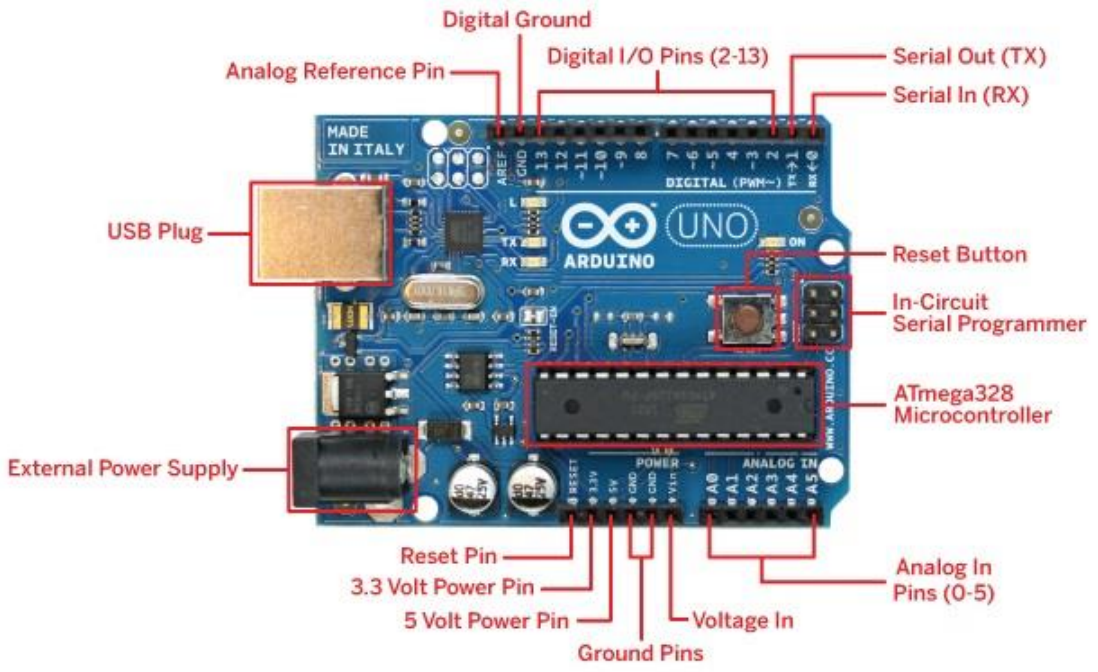


Figura 7.1 *Esquema Arduino Uno en detalle, fuente [29].*

La placa Arduino Uno no incluye por defecto ningún tipo de conexión inalámbrica ni Ethernet. Para ello, como veremos a continuación en este capítulo, necesitaremos incluirla un módulo adicional. Sin embargo, como podemos observar en la figura 7.1, dispone los siguientes pines de entrada y salida:

- **14 pines de entrada/salida digitales**. Entre estos 14 pines, los dos primeros pines, con posición 0 y 1, suelen utilizarse para la transmisión serial con el microcontrolador ATMega328 de la placa (0 = "Rx" y 1 = "Tx") , para poder programarlo a través de "USB-to-serial converter". Como podemos ver en la figura 7.1, los pines marcados con una raya curvada son los pines que ofrecen una salida PWM. En concreto, Arduino uno ofrece seis salidas PWM de 8 bits de resolución a través de los pines 11, 10, 9, 6, 5 y 3. Los pines 2 y 3 pueden ser empleados como interruptores externos, pudiendo ser configurados para activar la interrupción cuando toman un valor de nivel bajo, un valor de nivel alto o con un cambio de valor. Los pines 10(SS), 11(MOSI), 12(MISO) y 13(SCLK) permiten el uso de SPI, del inglés "Serial Peripheral Interface". SPI es un protocolo síncrono de intercambio de información serial empleado por microcontroladores para comunicarse de manera veloz con sus periféricos o con otros microcontroladores. Si configuramos estos pines correctamente podemos emplear la comunicación SPI con nuestro Arduino como maestro para controlar los periféricos con los que va a comunicarse. Además, el pin 13 dispone de una resistencia integrada que permite conectar un LED directamente. Por último, cada pin ofrece una corriente DC de salida de 40mA.

- **6 pines de entrada analógicos**, como podemos ver en la figura 7.1, desde el pin A0 hasta el A5. Estos pines disponen de una resolución de 10 bits, es decir, admiten 1024 valores diferentes. Además, el pin A4 (SCL) y el pin A5 (SDA) pueden emplearse en otros protocolos de comunicación serial como I2C, que veremos más adelante.

- **Pines de alimentación**. El pin Vin introduce un voltaje a la placa Arduino a través de un alimentador externo. El pin 5V ofrece un voltaje de salida 5V a través del regulador de la placa a partir de un alimentador externo conectado a la placa. El pin de 3,3V ofrece un voltaje de salida regulado de 3,3V.

- **Pines digitales adicionales**. En la placa Arduino Uno podemos emplear los pines de entrada analógicos como pines de entrada/salida digitales. Para ello habrá que configurarlos como tal vía software. Si queremos llevarlo a cabo, denominaremos en el software al pin A0 como pin digital 14, al pin A1 como pin digital 15, al pin A2 como pin digital 16, al pin A3 como pin digital 17, al pin A4 como pin digital 18 y al pin A5 como pin digital 19.

Por otro lado, la placa Arduino puede operar con un rango de voltaje entre 6V y 20V. En caso de ser alimentada con menos de 7V, el pin de 5V puede obtener menos de 5V y por tanto la placa puede ofrecer un comportamiento inestable. Sin embargo, en caso de usar más de 12V, el regulador de voltaje podría sobrecalentarse y dañar la placa. Por tanto, el rango recomendado de alimentación está entre 7V y 12V. Las formas entre las que puede ser alimentada son:

- Una **entrada USB**, que nos permite conectar la placa con nuestro ordenador para cargarla programas que hayamos desarrollada y alimentarla.
- Una **conexión para un alimentador** de corriente externo al que se le puede conectar un adaptador de corriente. Sin embargo, la placa Arduino Uno también puede ser alimentada a través de una batería empleando los pines Gnd y Vin de los pines de entrada reservados para alimentación. La placa Arduino uno elegirá automáticamente la fuente de alimentación entre las que se encuentren disponibles de las tres que hemos visto. En nuestro caso emplearemos un adaptador de corriente que ofrece a nuestra placa Arduino 9V y 1000mA.
- Los **pines de entrada de alimentación** vistos en la descripción de los pines la placa más arriba.

Por último, la memoria de la que dispone el microcontrolador ATMega328 es de 2KB para SRAM, 1KB para EEPROM y 32KB de memoria flash, de la que 0,5KB ya están ocupados por el "Bootloader", que nos permite cargar nuestros programas directamente en la placa sin necesidad de utilizar un hardware adicional. Aunque a priori parezca que su capacidad es algo limitada, para la mayoría de los proyectos es suficiente, lo que nos ofrece una buena relación calidad precio. Además, es posible extender la funcionalidad de la placa a través de unos shields adicionales como el que vamos a ver a continuación.

Las **funciones** que debe cumplir la placa Arduino Uno, como cerebro de cada nodo remoto, son las siguientes:

1. Regular el comportamiento del bloque de climatización con el que se corresponde, mediante el uso de sondas para ejecutar una serie de órdenes en los actuadores. De tal forma que cumpla con las estrategias de control vistas en el cuarto capítulo y los requisitos analizados en la metodología.
2. Informar al nodo central del estado en el que se encuentra el bloque de climatización cada 30 segundos.
3. Informar al usuario del estado en el que se encuentra el bloque de climatización a través de la interfaz auxiliar.
4. Comprender las órdenes recibidas por el usuario a través de la interfaz auxiliar.
5. Comprender las órdenes recibidas por el usuario a través del nodo central.
6. Entender la lectura de los sensores de temperatura, o de temperatura y humedad, según corresponda, y enviar la orden correcta a los servomotores y relés para que la ejecuten.
7. Detectar e informar de posibles averías que puedan producirse en los elementos del bloque de climatización.

7.1.1 Arduino Ethernet Shield

En la figura 7.2 podemos visualizar el shield Ethernet compatible con la placa Arduino Uno. De acuerdo con la web oficial de Arduino [50], el shield Ethernet nos permite conectar nuestro Arduino a Internet en apenas unos minutos. Para ello, simplemente tenemos que conectar

este módulo a nuestra placa Arduino tal y como vemos en la figura 7.3, gracias a que el shield incorpora unos pines macho que permiten conectar el shield con la placa. En la figura 7.4, podemos ver el shield Ethernet montado sobre la placa Arduino Uno que hemos empleado.

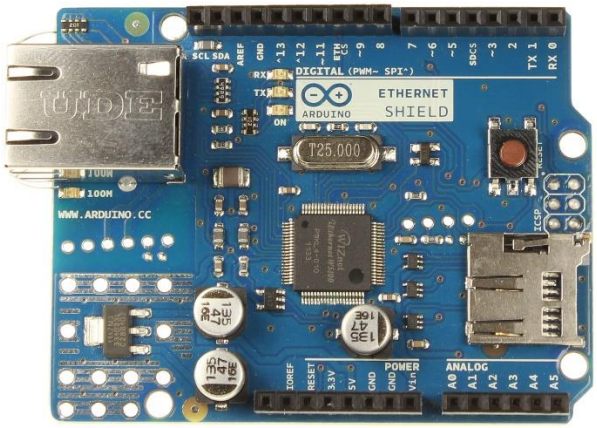


Figura 7.2 *Arduino Ethernet Shield, fuente [50].*

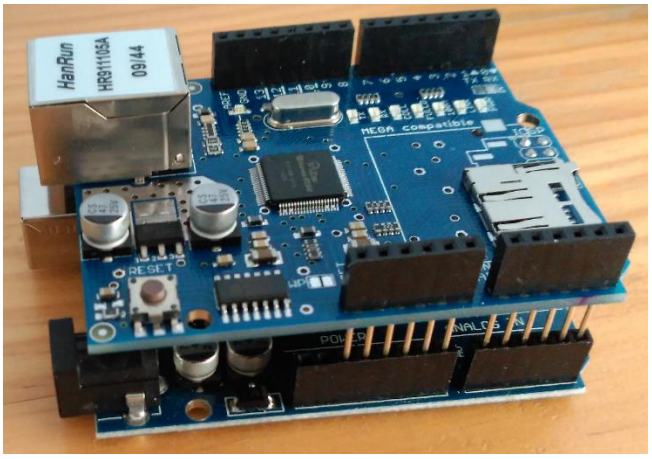


Figura 7.3 *Dispositivo Arduino Uno junto a shield Ethernet empleado.*

Como podemos observar en [50], el shield ofrece conexión a Internet a través de una entrada RJ-45 estándar. El shield está basado en el chip Ethernet Wiznet W5100, que proporciona un stack IP capaz de soportar los protocolos TCP y UDP. Además, permite hasta cuatro conexiones de tipo socket abiertas al mismo tiempo. Gracias a la librería Ethernet.h, permite crear programas para que nuestro Arduino pueda acceder a Internet de diferentes maneras, podemos encontrar diversos ejemplos en [39].

Por otro lado, como podemos ver en la figura 7.2, el shield Ethernet dispone de un botón de reset, situado justo encima de la ranura para tarjeta micro-SD, para garantizar que el chip Wiznet se reinicia correctamente tras alimentar el shield. Además, para alimentar el shield, dispone de un módulo PoE, del inglés “Power over Ethernet”, que extrae potencia de cable de par trenzado Cat5 Ethernet conectado a través del conector RJ-45 estándar para alimentar al shield.

Por último, podemos observar que el shield Ethernet incorpora además una ranura para una tarjeta micro-SD. Sin embargo, tanto la tarjeta micro-SD como el chip Ethernet

Wiznet se comunican con el microcontrolador de la placa Arduino a través del protocolo SPI, por tanto, los pines 10, 11, 12 y 13 quedan reservados para la comunicación entre Arduino y el shield, no podrán ser usados como pines de entrada/salida digitales. El hecho de que el chip Ethernet Wiznet y la tarjeta micro-SD compartan el bus de comunicación SPI provoca que solo pueda emplearse uno de los dos a la vez. El pin 10 de la placa Arduino Uno sirve para seleccionar el chip Wiznet mientras que el pin 4 sirve para seleccionar la tarjeta micro-SD, ambos pines tampoco pueden ser usados como pines de entrada/salida digitales. Para desactivar el uso de uno de los dos, tendremos que configurar su pin como una salida digital y asignarle un valor alto vía software.

7.1.2 Alternativas

El dispositivo de control para cada uno de los nodos remotos que podíamos haber escogido podría haber sido otra placa Arduino. Sin embargo, hemos escogido esta ya que el número de pines del que dispone nos es suficiente para cada uno de los nodos remotos, como podemos ver en la tabla 7.1. Además, la memoria que nos ofrece el microcontrolador Arduino Uno soporta la lógica de la programación requerida para cada nodo remoto, dicha lógica podremos entenderla en mayor detalle en el próximo capítulo. Por otro lado, es compatible con la gran mayoría de shields adicionales disponibles para Arduino.

En segundo lugar, podríamos haber escogido alguna de las placas de las otras dos plataformas de electrónica libre que describimos en el segundo capítulo. Por ejemplo, de la plataforma Wiring podríamos haber escogido la placa Wiring S, cuyo precio es similar a la placa Arduino Uno y además es compatible con la mayoría de shields para Arduino. Sin embargo, no dispone de entradas/salidas digitales disponibles para todos los nodos remotos que forman nuestro sistema de control. Por ello no ha sido escogido.

Por último, podríamos haber escogido cualquiera de las placas de la plataforma Netduino, ya que sus microcontroladores tienen una capacidad mayor que el microcontrolador de la placa Arduino Uno, ya que incluso permiten el uso de multitarea. Sin embargo, aunque se podría haber llevado a cabo el proyecto con cualquiera de sus placas, también hubiera elevado el precio de nuestro sistema, ya que cualquiera de ellas tiene un precio más elevado en su tienda oficial que la placa Arduino Uno. Además, la plataforma Arduino tiene una comunidad con un mayor número de usuarios que la plataforma Netduino, lo que supone que haya un mayor número de librerías para extender el software de la plataforma Arduino disponibles y depuradas para su uso.

7.1.3 Asignación de pines

Para terminar con la descripción de Arduino Uno, en la tabla 7.1 podemos ver la asignación de los pines de la placa para cada uno de los nodos remotos en función del diseño que hicimos en el primer apartado del capítulo seis para los nodos remotos y sus respectivos bloques de climatización en uso. La asignación de los pines a emplear nos servirá de guía para llevar un orden en el desarrollo software de cada uno de los nodos remotos, como veremos en el próximo capítulo.

Nodo Remoto	Asignación pines Arduino Uno
Bloque 1: Climatizador	**Pines Digitales:** • 4, 10, 11, 12, 13: reservados shield Ethernet. • 2: entrada pulsador luz pantalla LCD. • 3: entrada estado climatizador. • 5: entrada estado recuperador. • 6: entrada estado bomba. • 7: entrada sonda de temperatura sala. • 8: entrada sonda de temperatura circuito. • 9: salida PWM servomotor válvula. • 14(pin A0 convertido): salida relé climatizador. • 15(pin A1 convertido): salida relé climatizador. **Pines Analógicos:** • A5: SDA para i2c de pantalla LCD. • A4: SCL para i2c de pantalla LCD. • A3: entrada potenciómetro selección de temperatura deseada.
Bloque 2: Distribución SR	**Pines Digitales:** • 4, 10, 11, 12, 13: reservados shield Ethernet. • 2: entrada pulsador luz pantalla LCD. • 5: entrada estado bomba. • 6: entrada sonda de temperatura colectores de suelo radiante. • 7: entrada sonda de temperatura circuito. • 8: salida relé bomba. • 9: salida PWM servomotor válvula. **Pines Analógicos:** • A5: SDA para i2c de pantalla LCD. • A4: SCL para i2c de pantalla LCD. • A3: entrada potenciómetro selección de temperatura deseada.
Bloque 3: Termostato 2 etapas	**Pines Digitales:** • 4, 10, 11, 12, 13: reservados shield Ethernet. • 2: entrada pulsador luz pantalla LCD. • 3: salida relé válvula etapa 1. • 5: salida relé válvula etapa 2. • 6: entrada estado climatizador. • 7: entrada estado recuperador. • 8: entrada sonda de temperatura y humedad sala. • 9: entrada sonda de temperatura circuito. • 14(pin A0 convertido): salida relé climatizador. • 15(pin A1 convertido): salida relé climatizador. **Pines Analógicos:** • A5: SDA para i2c de pantalla LCD. • A4: SCL para i2c de pantalla LCD. • A3: entrada potenciómetro selección de temperatura deseada.
Bloque 4: Intercambiador calor	**Pines Digitales:** • 4, 10, 11, 12, 13: reservados shield Ethernet. • 2: entrada pulsador luz pantalla LCD. • 3: entrada estado bomba circuito primario. • 5: entrada estado bomba circuito secundario. • 6: entrada sonda de temperatura piscina. • 7: entrada sonda de temperatura circuito primario. • 8: salida relé bomba circuito primario. • 9: salida PWM servomotor válvula. **Pines Analógicos:** • A5: SDA para i2c de pantalla LCD. • A4: SCL para i2c de pantalla LCD. • A3: entrada potenciómetro selección de temperatura deseada.

Tabla 7.1 *Asignación de pines a Arduino Uno de cada nodo remoto.*

7.2 Sensores

En este apartado, vamos a describir los dos tipos de sensores que hemos empleado, es decir, sensores de temperatura y de temperatura/humedad. Para ello, partiremos los conceptos básicos sobre sensores vistos en el capítulo 2. En primer lugar, vamos a describir tanto el sensor de temperatura escogido como dos alternativas que podríamos haber escogido para nuestro proyecto. Después, describiremos el sensor de temperatura y humedad que hemos escogido y otra alternativa que podríamos haber seleccionado.

7.2.1 Sensor de temperatura DS18B20

Como vimos en el capítulo dos, el sensor DS18B20 es un sensor de temperatura digital, cuya precisión depende los bits de resolución de la señal digital de salida seleccionados por el usuario. En nuestro caso, la elegiremos en el programa de cada Arduino Uno. En concreto, puede ser:

- Para una resolución de 9 bits, una precisión de 0.5ºC.
- Para una resolución de 10 bits, una precisión de 0.25ºC.
- Para una resolución de 11 bits, una precisión de 0.125ºC.
- Para una resolución de 12 bits, una precisión de 0.0625ºC.

Además, otra de las características principales que tenía era que empleaba el protocolo 1-Wire para la comunicación entre el sensor, y en nuestro caso, la placa Arduino Uno. Como podemos ver en [15] se trata de un protocolo de comunicación serial, bi-direccional y half-duplex, entre un dispositivo 1-Wire maestro y unos o más dispositivos 1-Wire esclavos, mediante un único cable de datos junto a un cable de tierra como referencia. En concreto, que el dispositivo maestro, en nuestro caso la placa Arduino Uno, inicie la comunicación con uno o varios dispositivos 1-Wire en el mismo bus 1-Wire. Para identificar a cada uno de los dispositivos esclavos 1-Wire, se emplea un identificador de 64-bits único, que como ya vimos, cada sonda DS18B20 dispone de uno.

La ventaja que supone este protocolo es que podemos ahorrar pines en situaciones en las que tenemos numerosas sondas de temperatura para un mismo Arduino, ya que podemos conectar varias sondas DS18B20 en el mismo pin de Arduino compartiendo el mismo bus 1-Wire. Como vimos en el segundo capítulo, el sensor DS18B20 puede alimentarse de la propia línea de datos, llamado modo parásito, o alimentarse de forma externa, llamado modo normal. Si queremos conectar varios sensores DS18B20 en un único pin de Arduino debemos alimentarlos con el modo parásito. Además, el empleo del modo parásito limita el número de sondas que podemos conectar al mismo pin, ya que para la conversión que cada una realiza de temperatura a la señal digital se alimentan de la misma línea de datos. El esquema de conexión de ambos modos lo podemos visualizar en la figura 7.4. Como podemos ver, requiere una resistencia de “pull-up” entre la línea de datos y Vcc.

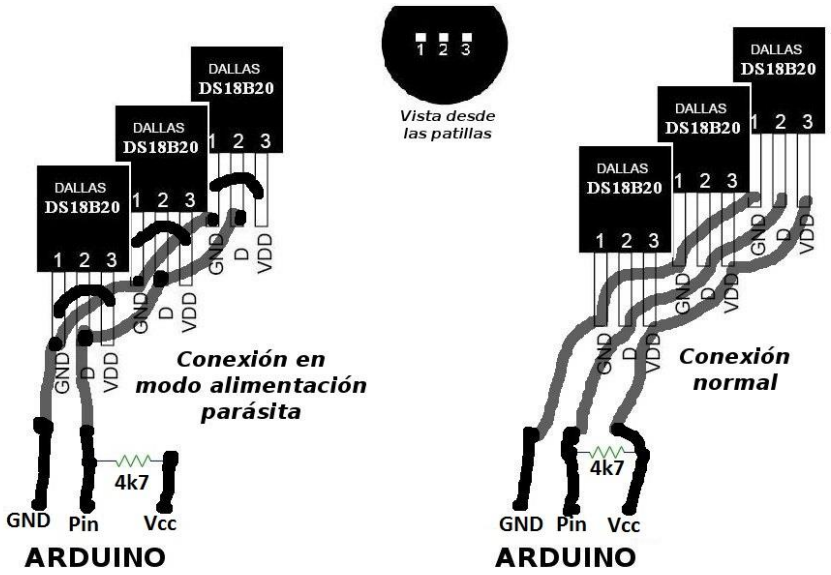


Figura 7.4 *Modos de conexión Sensor DS18B20, fuente [57].*

En nuestro caso, sólo tendremos un único sensor de temperatura DS18B20 por cada entrada a nuestra placa Arduino Uno. Por tanto, emplearemos el modo de alimentación normal. Para ello, conectaremos la patilla Vcc del sensor con el pin de salida de la placa Arduino Uno que ofrece un voltaje de salida fijo con un valor de 5V.

Los motivos que nos han hecho seleccionar el sensor digital de temperatura DHT22 frente a los que vimos en el capítulo dos son los siguientes:

- Se puede adquirir una modalidad del sensor resistente al agua. Lo que nos permite adquirir el mismo sensor para cualquiera de los nodos remotos y usar el mismo sensor tanto para medir la temperatura del aire o del agua según se requiera en el bloque de climatización en uso por el nodo remoto.
- Su precisión es bastante buena, como vimos al comienzo del apartado, y al tratarse de una señal digital sus mediciones de temperatura serán más robustas frente al ruido.
- El rango de temperatura de trabajo de nuestros bloques de climatización se encuentran dentro del rango sobre el que trabaja el sensor, por tanto, podrá medir todas las temperaturas requeridas.
- El hecho de que cada sensor disponga de una dirección única de 64-bits permite que podamos identificar una avería en un determinado sensor.

7.2.1 Sensor de temperatura y humedad DHT22

El sensor de temperatura y humedad que hemos escogido para nuestro proyecto, en concreto, para medir la temperatura y humedad relativa de las salas que emplean el termostato de dos etapas (2º bloque de climatización). Como vimos durante la descripción tanto del bloque de climatización del termostato de dos etapas como su estrategia de control, durante el verano el suelo radiante distribuido por el segundo bloque se va a emplear en la sala en la que se encuentra el termostato. En este caso, el agua que circula a través del circuito por el suelo de la sala se encuentra a una temperatura inferior a la temperatura del aire de la sala puede aumentar los niveles de humedad relativa ambiente y provocar condensación de agua en el suelo. Para que esto no ocurra, disponemos del sensor DHT22 conectado a un pin de entrada digital de nuestro Arduino Uno, de tal forma que cuando el nivel de humedad relativa supere un nivel que estableceremos en el programa de nuestro Arduino, éste active el climatizador auxiliar de la sala.

7.2.3 Alternativas

En cuanto a los sensores de temperatura que vimos en el capítulo dos y que podíamos haber usado en nuestro proyecto son el sensor de temperatura analógico TMP36 y el termistor NTC.

De haber escogido el sensor analógico TMP36 en cuanto a precisión ofrecida a la hora de medir temperaturas, la que nos ofrece es adecuada para el desarrollo de nuestro sistema de control. Sin embargo, hay que tener en cuenta que el sensor escogido, el sensor digital DS18B20, permite al usuario escoger una precisión aún mayor. En cualquier caso, el motivo principal por el que no hemos escogido este sensor es porque en nuestro sistema de control va a ser necesario medir tanto temperatura de agua como de aire. Por tanto, en caso de haber seleccionado este sensor para nuestro proyecto, tendríamos que procurar añadir a cada sensor empleado para medir agua un elemento que le permita sumergirse.

En caso de haber escogido el termistor NTC para nuestro proyecto habría requerido una serie de pruebas adicionales sobre cada bloque de climatización para calibrar la conversión de la temperatura medida a un valor legible por nuestras placas Arduino Uno. Por este motivo, junto a que también habría que añadirle un elemento extra para hacerle resistente al agua, creemos que de haber seleccionado este sensor el desarrollo de nuestro proyecto hubiese sido más complejo.

En cuanto a los sensores de temperatura y humedad, en el capítulo dos vimos otro sensor alternativo al escogido, el sensor DHT11. Como vimos, el sensor DHT11 ofrece una solución de menor coste que el sensor DHT22. Sin embargo, las características técnicas de dicho sensor, como vimos, ofrecen una precisión tanto para medir la temperatura como la humedad relativas notablemente inferiores a las del sensor DHT22. Por ese motivo, creemos que la elección del sensor DHT22 es la adecuada, para conseguir que nuestro sistema de control sea preciso.

7.3 Actuadores

Un actuador es un dispositivo que actúa en base a una condición específica o a un evento disparador. Para nuestro proyecto, vamos a usar dos tipos de actuadores diferentes: relés para controlar aparatos de un determinado bloque de climatización conectados a nuestro Arduino Uno de cada nodo remoto. Servomotores s de rotación continua de 180º para las válvulas mezcladoras de un determinado bloque de climatización conectadas a nuestro Arduino Uno de cada nodo remoto.

7.3.1 Relé

Como podemos ver en [2], un relé es un interruptor que opera eléctricamente. De tal forma que la corriente que fluye a través de la bobina del relé crea un campo magnético para atraer una palanca que cambia los contactos del interruptor. Dicha corriente que atraviesa la bobina puede estar en dos modos: “on” y “off”. Por lo que el comportamiento del relé puede resumirse en el mismo que el de un interruptor con dos posiciones. En nuestro caso, permitirá encender o apagar un determinado aparato, como por ejemplo, la unidad climatizadora o el elemento recuperador del primer bloque de climatización mediante una placa Arduino Uno. Para ello, vamos a emplear un módulo que nos permite conectar hasta cuatro relés y nos permite controlar a través de nuestro Arduino aparatos que trabajan con una corriente elevada. En la figura 7.5 podemos ver el módulo de cuatro relés que vamos a emplear.

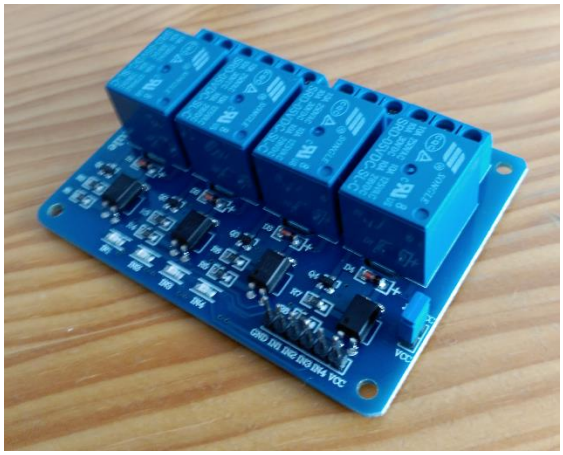


Figura 7.5 *Módulo de 4 relés empleado.*

Como podemos ver en la figura 7.6, nuestra placa Arduino Uno a través del pin que ofrece 5V y el pin digital de salida al que se conecta cada relé, que ofrece hasta 40mA nos servirán para alimentar al relé correspondiente en el módulo para encender o apagar el aparato del bloque de climatización correspondiente, mediante el envío de la señal de salida digital de encendido o apagado de nuestro Arduino al relé. En la figura 7.6, podemos ver como conectar cuatro salidas digitales de la placa Arduino Uno a la placa de cuatro relés.

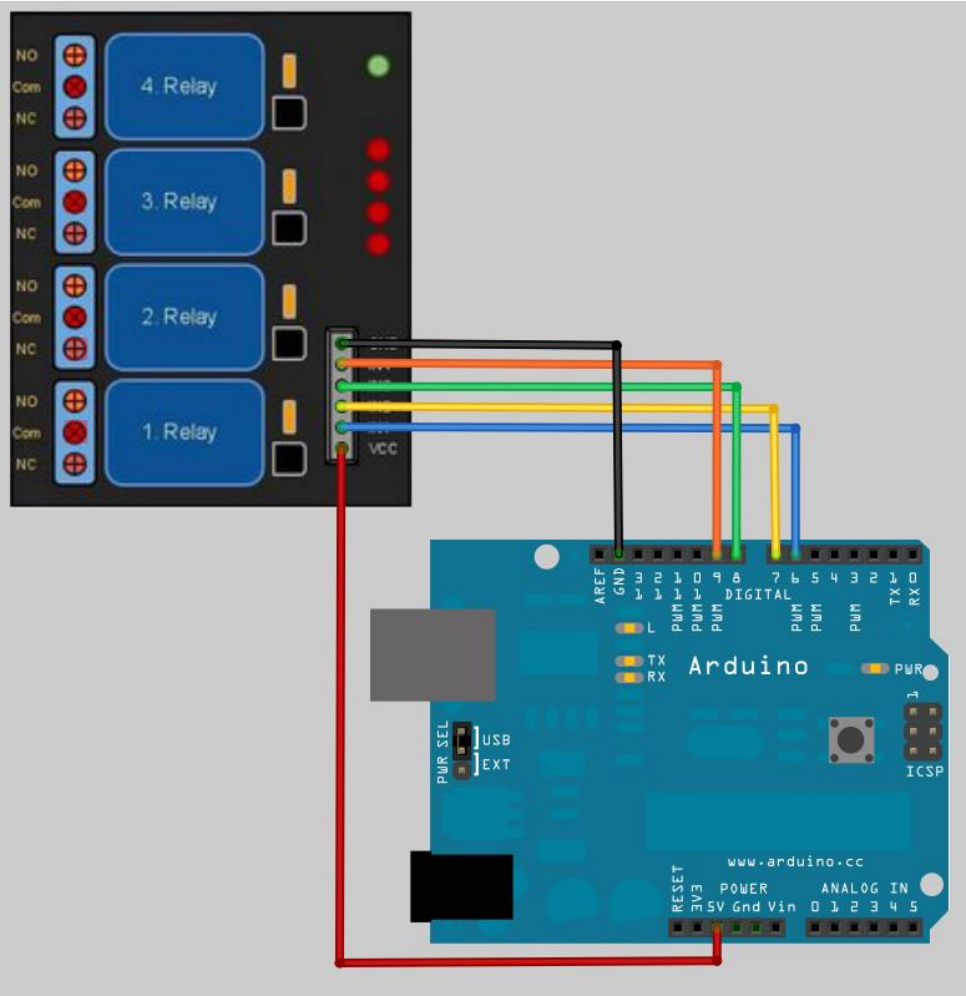


Figura 7.6 *Esquema conexión reles Arduino, fuente [51].*

Como podemos ver en la figura 7.6, cada relé dispone de tres salidas. La central COM, del inglés “Common Connection”, por la que circulará la corriente de salida del relé. La salida NO, del inglés “Normally Open, y la salida NC, del inglés “Normally Closed”. Por defecto, la salida COM está conectada a la salida NC, de tal forma que la salida digital desde nuestro Arduino tiene que tener un valor “HIGH”, es decir, el pin de entrada de los relés tendrá 5V, pero los aparatos conectados a los relés estarán desconectados. De esta forma, en caso de un fallo inesperado en nuestra placa Arduino, los aparatos se apagaran. Sin embargo, si la salida digital de nuestro Arduino a un relé tiene un valor “LOW”, es decir, el pin de entrada del relé tendrá 0V, la salida COM cambiará de posición y su corriente circulará a través de la salida NO, encendiendo el aparato conectado a dicho relé.

Por último, también emplearemos los relés como actuadores del nodo remoto del lugar que emplea el bloque de climatización con termostato de dos etapas. Como vimos en las estrategias de control, cuando el Arduino envíe una señal con valor “LOW” al relé de una etapa la activará. Mientras que si le envía una señal “HIGH” la mantendrá en reposo.

7.3.2 Servomotor

Como podemos ver en [2], un servo es un tipo de motor que en vez de rotar continuamente mueve su eje en diferentes posiciones. En concreto, estas posiciones son ángulos específicos en el que el servo mueve su eje dentro de un rango entre 0º y 180º. Para saber la posición que debe tomar el eje, el servo utiliza una señal de tipo "Pulse Coded Modulation", recibida desde un microcontrolador. La duración del pulso de la señal recibida determinará el ángulo de movimiento del eje del servo. En el mercado, existen diversos tipos de servomotores para nuestro Arduino que podemos emplear para controlar la posición de la válvula mezcladora de cada bloque de climatización. Entre ellos, están los servomotores analógicos, digitales, de rotación continua, de medio giro, etc. Por tanto, elegiremos un servomotor digital de medio giro (180º) que nos permita posicionar las entradas de las válvulas mezcladoras del bloque de climatización al que cada Arduino Uno regula en la posición adecuada, en función del comportamiento esperado de las válvulas que vimos durante el capítulo dos.

El servomotor que hemos escogido es el modelo estándar TowerPro SG-5010. Respecto a sus características técnicas, vemos que puede ser alimentado con un voltaje de 5V a 6V. De este modo, nos ofrece un torque entre 5 y 6 Kg/cm. Sin embargo, para las pruebas de funcionamiento emplearemos un micro servo 9G, menos potente que el seleccionado para integrar el sistema, pero más económico y suficiente para las pruebas.

Sin embargo, como veremos en el próximo capítulo, la librería que vamos a emplear para utilizar un control PID en nuestras placas Arduino Uno para determinar la posición del servomotor nos da una salida con un valor entre 0 y 255, normalmente empleados para actuadores que son dirigidos mediante señales PWM, del inglés "Pulse Width Modulation", en la que el valor del ciclo de trabajo determinará la actuación.

En nuestro caso, como veremos en el próximo capítulo, emplearemos la librería Servo.h [48] para dirigir la posición de nuestros servomotores. Esta librería emplea una función con un valor entre 0 y 180 para controlar la posición del eje del servo. Sin embargo, para cumplir con las estrategias de control vistas en el tercer capítulo, el servomotor va a girar su eje únicamente entre 0º y 90º, con un valor 0º mantendría el bloque en reposo y con un valor de 90º lo activaría a la máxima potencia disponible. Por tanto, con una expresión matemática en nuestro software convertiremos dicho valor PWM de salida del control PID en un valor proporcional entre 0 y 90 para dirigir la posición del eje de nuestro servomotor de manera más correcta.

7.4 Interfaz Auxiliar

Como dijimos durante el análisis de los requisitos, cada uno de los nodos remotos de un lugar del complejo con un determinado bloque de climatización va a disponer de una interfaz auxiliar en caso de fallo de la aplicación web. Para ello, vamos a dotar a cada una de nuestras placas Arduino Uno de tres elementos. Una pantalla LCD 16x2, un pulsador y un potenciómetro. En la figura 7.7 podemos ver el diseño propuesto para la interfaz auxiliar.

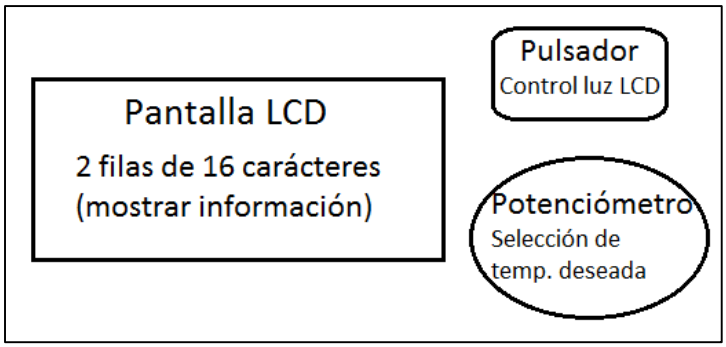


Figura 7.7 *Diseño Interfaz Auxiliar*

El pulsador lo conectaremos al pin 2 de cada una de nuestras placas Arduino Uno, para programar vía software una interrupción en el microcontrolador de la placa que será disparada cuando el usuario pulse el botón. Como veremos en el próximo capítulo, durante la interrupción la placa simplemente cambiará el valor de una variable que admite de dos posiciones, para representar con una que la luz de la pantalla LCD está activada y con la otra posición, que se encuentra apagada. De tal forma que al cambiar el valor de dicha variable, en el siguiente paso por el loop principal el Arduino apagará o encenderá la pantalla LCD en función del estado previo.

El potenciómetro lo emplearemos para seleccionar una temperatura de consigna en el bloque regulado por una placa Arduino Uno. El potenciómetro es una resistencia variable que en función del valor que tenga enviará una señal analógica al pin analógico al que se conecta con nuestro Arduino con un determinado valor de voltaje. Como vimos en el apartado anterior sobre la placa Arduino Uno, el conversor analógico digital está calibrado para ofrecer al microcontrolador un valor numérico entre 0, que representa 0V y 1024, que representa 5V. Por tanto, como veremos en el siguiente capítulo, mediante una fórmula matemática entre el valor ofrecido por el potenciómetro, el modo de funcionamiento (verano o invierno) y los límites inferior y superior de la temperatura permitidos en el bloque que regula el Arduino asignaremos un valor a la temperatura de consigna del bloque.

7.4.1 Pantalla LCD

Para mostrar la información completa del bloque de climatización regulado por cierto Arduino Uno vamos a disponer de una pantalla LCD 16x2, es decir, que dispone de dos líneas con 16 carácteres cada una para mostrar dicha información. En concreto, a través de la pantalla LCD mostraremos el estado de los elementos conectados a Arduino del bloque de control al que pertenece, para mostrar si están encendidos, averiados o apagados. Por otro lado, mostraremos la lectura de los sensores de temperatura de cada lugar o de temperatura y humedad, según corresponde. Además, mostraremos la temperatura de consigna asignada para cada bloque.

Como podemos observar en [31], a través de la librería “LiquidCrystal”, podemos controlar la pantalla LCD a través de nuestra placa Arduino Uno. Sin embargo, la pantalla LCD dispone de una interfaz paralela, por lo que el microcontrolador de nuestro Arduino debe ser capaz de manipular varios pines al mismo tiempo para controlar la pantalla LCD. Los pines requeridos para controlar la pantalla LCD son los siguientes:

- Pin de selección de registro (RS), para controlar el lugar en la memoria de la pantalla LCD donde escribimos.
- Pin de lectura y escritura (R/W), para seleccionar el modo lectura o escritura.
- Pin para permitir escribir en los registros de memoria de la pantalla LCD.
- Ocho pines de datos, correspondientes a los bits que escribimos durante el modo escritura o los bits que leemos durante el modo lectura.

Por otro lado, necesitaremos de varios pines adicionales. Por un lado, un pin para controlar la luz de fondo de la pantalla LCD, normalmente controlada por un potenciómetro. Un pin para alimentación, a la salida del Arduino que ofrece 5V, y otro a GND. Sin embargo, en nuestro caso no disponemos de tantos pines libres en cada una de nuestras placas Arduino Uno. Por tanto, vamos a emplear el protocolo I2C para reducir el número de pines necesario para controlar la pantalla LCD con nuestro Arduino.

El protocolo I2C es un protocolo que permite la comunicación entre múltiples esclavos conectados en el mismo bus con un dispositivo maestro, requiere solo de dos cables, SCL y SDA, para la comunicación. La ventaja que nos ofrece este protocolo es que podemos controlar todas los pines de la pantalla LCD empleando solo 4 pines de nuestra placa Arduino Uno, en los que dos de ellos son los pines para alimentar la pantalla y el dispositivo I2C conectado a ella. Los dos pines necesarios para controlarla mediante I2C son el pin analógico A4 para SDA y el pin analógico 5 para SCL. Además, el dispositivo I2C que conectamos con la pantalla LCD dispone de un potenciómetro para que podamos ajustar la luz de fondo de la pantalla LCD. En la figura 7.8, podemos observar la pantalla LCD con el dispositivo I2C conectado que hemos empleado. En la parte izquierda podemos observar la pantalla de frente y en la parte derecha por detrás.

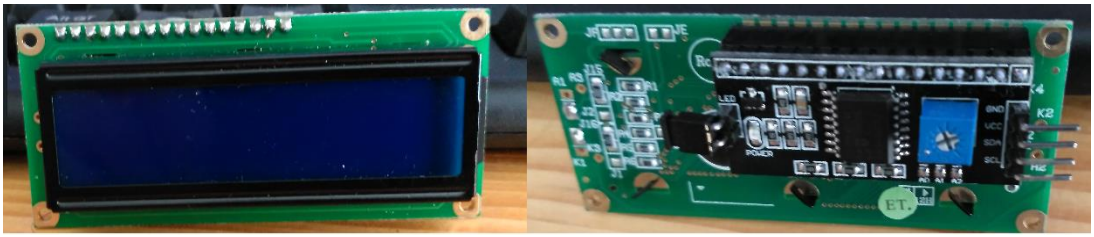


Figura 7.8 *Pantalla LCD con dispositivo I2C empleados.*

El formato de los mensajes mostrados por la pantalla LCD dependerán del estado en el que se encuentre el bloque de climatización regulado por la placa Arduino.

- Si el bloque de climatización tiene **orden central de apagado**, en el que todos sus elementos están desconectados el mensaje que aparecerá será: en la primera línea de la pantalla LCD será el nombre del lugar del nodo remoto, por ejemplo, “POLIDEPORTIVO”, “AULA”, “COMEDOR” o “PISCINA”. Mientras que en la segunda línea aparecerá el mensaje “Apagado”.

- Si el bloque de climatización tiene **orden central de encendido**, el mensaje que aparecerá tendrá el siguiente formato: en la primera línea aparecerán las primeras letras del lugar dónde está situado el nodo remoto, por ejemplo “Poli.”, “Comd.”

,“Pisc.” o “Aula”, seguido de las siglas para reconocer los aparatos conectados a dicho bloque de climatización, como “B” para bomba de circulación, “R” para recuperador y “Cl” para climatizador seguido del estado en el que están, es decir, “ON”, “OFF” o “ERR”. Sin embargo, para los nodos remotos que emplean el termostato de dos etapas, como las aulas, en la primera línea aparecerá el nombre del aula y el valor de la última medición de la humedad relativa de la sala. Por otro lado, en la segunda línea, aparecerá “Csg:” seguido de la temperatura deseada en el nodo remoto y “T:” seguido de la última lectura de la temperatura del lugar realizada por el sensor correspondiente. En caso de accionar el potenciómetro de consigna, en el nodo remoto de cualquier bloque de climatización la segunda línea seguirá de la misma manera, mientras que en la primera línea se sustituirá el tramo donde se podía visualizar la información de estado del elemento por los valores producidos por la posición del potenciómetro para seleccionar la temperatura deseada en el bloque.

7.5 Raspberry Pi B+

Como comentamos en el capítulo anterior, el nodo central de nuestra arquitectura será una Raspberry Pi B+. La Raspberry Pi será la encargada de coordinar, dirigir y conocer el estado de todos los nodos remotos de nuestra arquitectura. Los motivos principales por los que hemos escogido la Raspberry Pi frente a otros ordenadores de bajo coste que planteamos en el capítulo dos es que nos ofrece capacidad suficiente para la funcionalidad requerida, podremos alojar el software requerido para nuestro sistema de control en ella. Además, es el ordenador de bajo coste con menor precio de todos los que vimos y dispone de una comunidad de un tamaño considerable que ofrece muy buen soporte a los usuarios e información útil para trabajar con dicho dispositivo. Sin embargo, en caso de haber elegido cualquiera de los otros ordenadores de bajo coste vistos, también hubiéramos podido llevar a cabo nuestro proyecto, pero a un precio algo superior. En la figura 7.9 podemos visualizar la Raspberry Pi empleada y sus partes más importantes:

- Los **40 pines GPIO** (General Purpose Input/Output) permite a la Raspberry Pi interactuar con otros elementos. Por ejemplo, un pin puede ser una entrada correspondiente a un sensor, otro pin puede ser una entrada o una salida hacia otro dispositivo, como Arduino, y otro pin podría ser una salida hacia un actuador, por ejemplo un LED.

- Con el procesador ARM de 700 MHz, podemos instalar ciertas distribuciones Unix, como Raspbian o Pidora. Para ello, podemos descargar la distribución desde la web oficial de Raspberry Pi e introducirla en una tarjeta micro-SD para instalar la distribución en nuestra Raspberry. Sin embargo, también podemos comprar una tarjeta micro-SD con la distribución Raspbian preinstalada. Además, la tarjeta micro-SD ofrecerá capacidad de almacenamiento a nuestra Raspberry Pi.

- Disponemos de 4 puertos USB 2.0 con suficiente capacidad para alimentar un teclado, un ratón o cualquier otro dispositivo USB que necesitemos para nuestra Raspberry. Sin embargo, requiere una alimentación mínima de unos 2A para que funcione adecuadamente a través de la ranura micro-USB, para ello, podemos emplear un adaptador de corriente.

- Dispone de una entrada RJ-45 que soporta una conexión Ethernet de hasta 100 Mbps, lo que nos permite navegar por Internet e interactuar con otros dispositivos a través de la red.

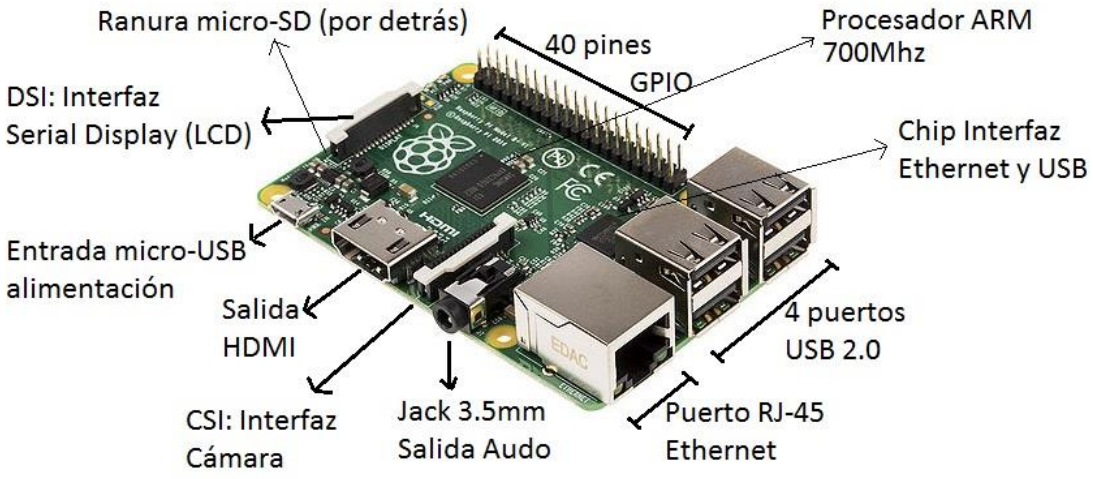


Figura 7.9 *Esquema Raspberry Pi B+.*

Para nuestro proyecto, hemos instalado la distribución Raspian en nuestra Raspberry Pi, ya que es la distribución más recomendada y dispone de un mayor soporte. Para ello, hemos adquirido una tarjeta micro-SD de 16Gb, sobre la que hemos instalado la imagen de la distribución, descargada desde la web oficial de Raspberry Pi de forma gratuita. Una vez instalada la imagen, introducimos la tarjeta micro-SD en la Raspberry Pi que al encenderla por primera procederemos a instalar el sistema operativo.

Por otro lado, como vimos durante el capítulo anterior, en la arquitectura de nuestro sistema de control, vamos a necesitar que nuestra Raspberry Pi aloje una serie de programas, para cumplir con sus **funciones** principales:

1. **Alojar la aplicación web Java** como interfaz central de usuario. Para ello, empleara el servidor web Apache Tomcat, que es un contenedor de Servlets de acceso libre y gratuito, descargable en su web oficial [36]. Un contenedor de Servlets es un software capaz de captar peticiones HTTP y redirigirlas al Servlet correspondiente. El servidor web Apache Tomcat será la base sobre la que se sustenta nuestra aplicación web local. Se encargará de recibir las peticiones HTTP desde el navegador web del Usuario, ejecutar el Servlet correspondiente a la petición, entre los que vimos en la arquitectura de la aplicación web en el capítulo seis, y mostrar la vista correspondiente en el navegador del usuario. De esta forma, podrá enviar órdenes y mostrar la información de cada uno de los nodos remotos

2. **Alojar un bróker local** que coordine la comunicación MQTT entre el nodo central y los remotos. Para ello, va a emplear el bróker Mosquitto es un bróker de acceso libre para el protocolo MQTT, descargable en su web oficial [37]. Como vimos en el capítulo anterior, el bróker es el encargado de distribuir los mensajes del protocolo MQTT entre un editor y sus suscriptores. Podemos instalar el software del bróker Mosquitto en

nuestra Raspberry Pi de forma gratuita para garantizar el funcionamiento correcto del protocolo MQTT en nuestra red local. De esta forma, podremos comunicar nuestra Raspberry Pi, a través de Mosquitto, con cada uno de las placas Arduino Uno de los nodos remotos.

3. **Alojar un servidor local MySQL**, que podemos descargar e instalar de forma gratuita desde su página web oficial [34]. En él, podemos almacenar los datos de cada uno de los nodos remotos en la tabla correspondiente, vista en el capítulo anterior, de la base de datos de nuestro sistema de control.

4. **Alojar cuatro** programas que actúen como **clientes MQTT** de cada uno de los bloques de climatización a los que pertenecen los nodos remotos, de tal forma que puedan recibir los informes de estado de cada uno de ellos e introducirlos en la tabla correspondiente de la base de datos para su almacenado.

Capítulo 8: Implementación software

En este capítulo vamos a describir la implementación software que hemos realizado en los nodos de nuestro sistema de control. Para ello, primero explicaremos la programación realizada en los cuatro tipos de nodos remotos de nuestro sistema de control. Después, explicaremos la programación realizada en el nodo central. En concreto, explicaremos la programación realizada para llevar a cabo nuestra aplicación web Java y la programación de los clientes MQTT de cada bloque de climatización.

8.1 Nodos Remotos

Como vimos anteriormente, las placas Arduino Uno serán el cerebro de cada uno de los nodos remotos, encargado de regular el comportamiento del bloque de climatización del lugar donde está situado el nodo remoto. En este apartado, veremos la programación que hemos implementado en los cuatro tipos de placa Arduino correspondiente a los cuatro diseños de nodo remoto, uno por cada bloque de climatización a controlar, vistos en el capítulo sexto, donde expusimos la arquitectura que vamos a realizar. Sin embargo, en primer lugar, vamos a describir brevemente la herramienta empleada para programar cada placa Arduino.

8.1.1 Arduino IDE

Arduino IDE es el entorno de programación empleado para desarrollar software para cualquier placa Arduino, se puede descargar desde la página web de la plataforma [35] de forma gratuita. Como comentamos en el capítulo dos, el entorno de programación es un entorno multiplataforma. Sin embargo, nosotros vamos a emplearlo en un ordenador portátil que incluye el sistema operativo Windows 8.1. En la figura 8.1 podemos ver el entorno de programación.

Como ya comentamos en el segundo capítulo, el lenguaje de programación empleado para Arduino está basado en los lenguajes de programación C/C++. Además, podremos emplear librerías adicionales para extender nuestro software y ofrecer una mayor funcionalidad a cada una de nuestras placas. En el próximo apartado, veremos las librerías que hemos empleado. Como podemos ver en la figura 8.1, la estructura de un programa de Arduino está formada por tres elementos:

1. **Declaración de variables**, que será el lugar donde incluiremos tanto las librerías adicionales como el conjunto de variables, pines y demás elementos que va a emplear nuestro código.
2. **Función setup**. Es la función que se ejecuta una única vez, cada vez que se reinicia o enciende la placa Arduino. En ella, se inicializa las variables empleadas por nuestro código y los periféricos conectados a nuestro Arduino Uno.
3. **Función loop**. Es la función que se ejecuta una y otra vez, ya que una vez finaliza se vuelve a repetir. En esta función se ejecutarán las instrucciones necesarias que debe llevar a cabo cada una de nuestras placas Arduino Uno.

4. Con el **botón 1** podemos verificar el programa escrito. Con el **botón 2**, podemos compilar y cargar el programa en la placa Arduino conectada con nuestro PC.

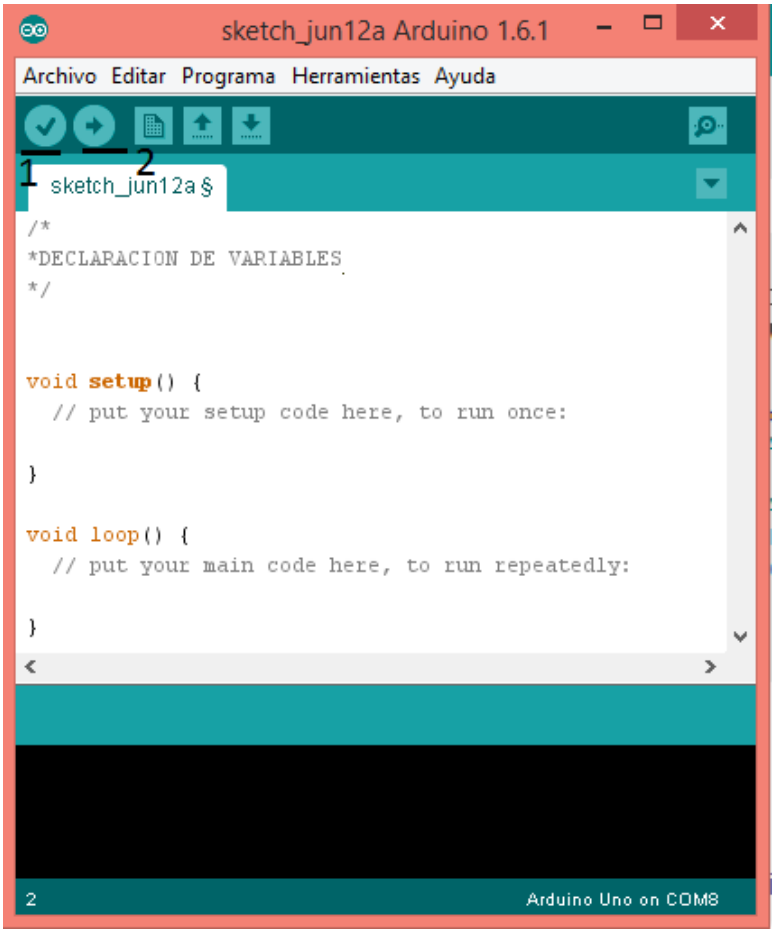


Figura 8.1 *Entorno de programación de Arduino*

8.1.2 Librerías empleadas

En este apartado, veremos las librerías empleadas en los programas que hemos desarrollado para nuestras placas Arduino. En la figura 8.2 podemos observar la declaración de las librerías en el código para incluirlas correctamente en el software.

```
#include <Wire.h>
#include <LCD.h>
#include <LiquidCrystal_I2C.h>
#include <OneWire.h>
#include <DallasTemperature.h>
#include <PID_v1.h>
#include <SPI.h>
#include <Ethernet.h>
#include <PubSubClient.h>
#include <ArduinoJson.h>
```

Figura 8.2 *Declaración de las librerías para Arduino empleadas.*

Librerías para interactuar con la pantalla LCD 16x2:

Librería Wire [45]. Es la librería que vamos a emplear para comunicar el microcontrolador de la placa Arduino Uno con otros dispositivos empleando el protocolo I2C. En nuestro caso, emplearemos la librería para comunicar el microcontrolador con la pantalla LCD a través de los pines A4 (SDA) y A5 (SCL).

Librería LCD [30]. Es la librería que nos permite interactuar entre nuestro microcontrolador y la pantalla LCD de 16x2. Entre otras cosas, nos permitirá escribir información en ella y encender o apagar la luz de fondo.

Librería LCD_I2C [44]. Esta librería va a emplear funciones de las dos librerías anteriores para controlar la pantalla LCD mediante el protocolo I2C que nos permite ahorrar pines de nuestra placa Arduino. Con ella cada placa Arduino Uno será capaz de mostrar la información necesaria, como vimos en los capítulos anteriores, para cada bloque de climatización. Además, ofrece un comportamiento superior a la librería LCD, a la cual extiende. Emplearemos diversas funciones:

- write(), para escribir datos en las dos líneas de 16 caracteres de la pantalla o
- backlight(), para encender la luz de fondo o apagarla.

En la figura 8.3, podemos ver un ejemplo de cómo crear un objeto LCD para que sea controlado por nuestra placa Arduino, para después inicializarlo e imprimir un mensaje en la primera línea y la lectura de un sensor en la segunda línea.

```
                        //addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C lcd(I2C_ADDR, 2, 1, 0, 4, 5, 6, 7, 3, NEGATIVE);
lcd.begin(16,2);
lcd.backlight();

lcd.setCursor(0,0);lcd.print("AULA");
lcd.setCursor(0,1);lcd.print("T: "); lcd.setCursor(3,1)lcd.print(temperatura,1);
```

Figura 8.3 *Código de ejemplo de control de pantalla LCD con Arduino empleado.*

Librerías para interactuar con el sensor digital DS18B20.

Librería 1-Wire [43]. La librería 1-Wire permite comunicar el microcontrolador de nuestra placa Arduino, empleando dicho protocolo, con otros dispositivos 1-Wirecomo los sensores de temperatura DS18B20.

Librería DallasTemperature [42]. La librería DallasTemperature será la empleada para recibir las lecturas de los sensores de temperatura digitales DS18B20 conectados a una placa Arduino Uno. Para ello, la librería empleará un objeto DallasTemperature para representar a un sensor DS18B20, identificado por su dirección de 64-bits única. Como dijimos, empleará la librería 1-Wire para la comunicación entre el sensor y la placa. Para ello, asignará al objeto

DallasTemperature un objeto de la librería 1-Wire que representa el bus de intercambio de datos de la comunicación. Emplearemos funciones como:

- RequestTemperatures(), para pedir el sensor que realice una medición y la convierta a un valor numérico.
- getTempC(), a partir del identificador del sensor, para obtener en la placa el valor numérico de dicha medición en grados celsius. Además, podremos ajustar parámetros del sensor, como la precisión requerida.

En la figura 8.4, podemos ver un ejemplo de uso de la librería Dallas Temperature, como declarar los objetos correspondientes al sensor para después inicializarlo y ajustarle la precisión para posteriormente realizar y recuperar una lectura de temperatura.

```
OneWire oneWire_sala(5); //Pin digital de entrada
DallasTemperature sensor_sala(&oneWire_sala);
double temp_sensor_sala;
DeviceAddress sensor_sala_addr;

sensor_sala.begin();
sensor_sala.setResolution(10); //precision 0.25°C

sensor_sala.requestTemperatures();
temp_sensor_sala = sensor_sala.getTempC(sensor_sala_addr);
```

Figura 8.4 *Código de ejemplo de librería para sensor DS18B20 empleado.*

Librería para implementar el controlador PID en la placa Arduino Uno.

Librería PID [46]. La librería PID no permite a cada placa Arduino Uno regular el bloque de climatización correspondiente mediante un control de tipo PID, aplicando el control visto en el capítulo dos para las estrategias de control de cada bloque vistas en el cuarto capítulo. Esta librería no se empleará en las placas Arduino Uno que regulen el tercer bloque de climatización, es decir, el termostato de dos etapas. Para ello, la librería nos proporcionará una serie de funciones que vamos a emplear:

- La función PID(), para crear el controlador PID vía software a partir de los siguientes parámetros: “input”, o variable de entrada que deseamos controlar, en este caso será la temperatura correspondiente a la última lectura del sensor de temperatura DS18B20 de la sala. “Output”, variable ajustada por el PID, será un valor entre 0 y 255 que ejecutará el actuador del bloque. “Setpoint”, que será la temperatura deseada o de consigna en el bloque. El valor de las constantes Ki, Kp y Kd del controlador y por último la dirección, que puede ser directa o inversa. La dirección será importante en bloques donde tengamos dos modos de funcionamiento, uno en invierno y uno en verano. En invierno, emplearemos la dirección directa, ya una mayor diferencia entre la temperatura deseada y la medida en la sala supone que la temperatura medida debe aumentar. Por ello, un valor más alto en la salida del

controlador supondrá un aumento de la temperatura. Por el otro lado, en modo verano emplearemos la dirección inversa, ya que una mayor diferencia entre la temperatura medida en la sala y la deseada supone que la temperatura medida debe reducirse. Por tanto, la dirección inversa logrará que una salida con un valor mayor del PID reduzca la diferencia.

- La función SetMode(), para activar el controlador PID en modo automático, ya que la librería permite desactivar el controlador PID en un determinado momento si se requiere.
- La función SetSampleTime(), para determinar el intervalo de tiempo sobre el que debe evaluarse el algoritmo PID del controlador.
- La función Compute(), para ejecutar el algoritmo PID visto en el capítulo dos y determinar la salida del controlador que debe ejecutar el actuador de la placa Arduino Uno.

En la figura 8.5 encontraremos un ejemplo de código del controlador PID que hemos usado en las placas Arduino Uno de determinados bloques. En ella, primero declaramos las variables necesarias para el controlador PID de los dos modos de funcionamiento. Después, ajustamos los parámetros necesarios para ejecutar, en función del modo funcionamiento detectado por la sonda del circuito, el controlador PID requerido.

```
double temp_sensor_sala;
double temp_sensor_circuito;
double temp_consigna_I;
double temp_consigna_V;
double salida_PID;

double Kp= 20;
double Ki = 1;
double Kd = 4;
PID PID_invierno(&temp_sensor_poli, &salida_PID, &temp_consigna_I, Kp, Ki, Kd, DIRECT);
PID PID_verano(&temp_sensor_poli, &salida_PID, &temp_consigna_V, Kp, Ki, Kd, REVERSE);

PID_invierno.SetMode(AUTOMATIC);
PID_invierno.SetSampleTime(30000); //30 segs en ms
PID_verano.SetMode(AUTOMATIC);
PID_verano.SetSampleTime(30000);

if( temp_sensor_circuito >= cambio_IV )
   PID_calor.Compute();
}else{
   PID_frio.Compute();
}
```

Figura 8.5 *Código de ejemplo de la librería PID empleado.*

Librería para la generación y recepción de mensajes.

Librería ArduinoJson [38]. La librería ArduinoJson permitirá que cada placa Arduino Uno que empleemos cree los mensajes de estado del bloque que le corresponde e informar al nodo central. Además, la librería le permitirá crear dichos formatos en correspondencia con el formato diseñado para los mensajes de estado de cada bloque, vistos en la tabla 6.1 del capítulo seis. Por otro lado, nos permitirá que una vez recibamos un mensaje desde el nodo central, es decir, una orden, cada placa Arduino Uno pueda comprender el mensaje e identificar sus puntos más importantes como vimos en el formato de dichos mensajes en la tabla 6.2 del capítulo seis. Emplearemos las siguientes funciones y objetos de la librería:

- Un objeto StaticJsonBuffer<N>, para reservar espacio en la memoria tanto para el mensaje JSON de informe de estado que enviemos al nodo central como para el mensaje JSON que podamos recibir como una orden desde el nodo central.
- Un objecto JsonObject& creado por cada placa Arduino Uno para enviar un mensaje de estado con createObject() o a partir de un mensaje recibido con parseObject().
- Para acceder y modificar a los elementos de un JSON, utilizaremos la misma notación que para los elementos de un array, donde el índice del elemento al que queremos acceder será la clave que le asignamos al crearlo.

En la figura 8.6, podemos visualizar un ejemplo de código que hemos empleado para crear el mensaje de estado con una placa Arduino Uno de uno de los nodos remotos para informar del estado de su bloque de climatización. En concreto, pertenece a la creación de un mensaje de estado placa Arduino Uno del nodo remoto de la piscina grande. En primer lugar, reservamos espacio para el mensaje, lo creamos, rellenamos con los datos correspondientes e imprimimos.

```
StaticJsonBuffer<200> jsonBuffer;

JsonObject& estado = jsonBuffer.createObject();
estado["arduino"] = "17";
estado["bomba_cprimario"] = "on";
estado["bomba_csecundario"] = "on";
estado["sonda_cprimario"] = "on";
estado["temp_piscina"] = temp_sonda_piscina;
estado["consigna"] = temp_consigna;

char buffer[256];
estado.printTo(buffer, sizeof(buffer));
Serial.println(buffer);
```

Resultado en monitor Serial:

```
{"arduino":"17","bomba_cprimario":"on","bomba_csecundario":"on",
"sonda_cprimario":"on","temp_piscina":23.20,"consigna":23.00}
```

Figura 8.6 *Código de ejemplo para crear un mensaje JSON informativo empleado.*

En la figura 8.8, en el método Callback del cliente MQTT podemos ver un ejemplo de la decodificación de una orden del nodo central a la placa Uno de un nodo remoto empleando la librería ArduinoJson.

Librerías para obtener conexión a Internet.

La **librería Ethernet** [39], permite a una placa Arduino Uno que emplea un shield Ethernet con un chip Wiznet, como vimos en el capítulo anterior, se conecte a Internet. Esta librería permite emplear Arduino como un servidor, que reciba peticiones, o como un cliente, que envíe peticiones, permitiendo hasta cuatro conexiones simultáneas.

Como vimos en el capítulo anterior, la librería Ethernet va a requerir el uso de la **librería SPI** [40]. La librería SPI permite al microcontrolador de la placa Arduino Uno comunicarse mediante SPI, empleando los pines 10, 11, 12 y 13, con el chip Wiznet del shield Ethernet.

En nuestro caso, emplearemos la librería Ethernet como cliente a través de un objeto EthernetClient, que será empleado como podemos ver en la figura 8.7 por la librería MQTT. Además, le asignaremos una dirección MAC de 48 bits a nuestro Arduino, teniendo especial cuidado de que no se repita la misma dirección con ningún otro dispositivo de la red local Ethernet del complejo, para que mediante DHCP sea el propio router local el que nos asigne una dirección IP de la red local empleando la función begin().

Librería para controlar un servomotor desde la placa Arduino Uno.

La **librería Servo** [48], nos permite controlar la posición del eje de hasta dos servomotores con una placa Arduino Uno empleando los pines 9 y 10 de la placa. En nuestro caso, emplearemos un único servomotor, conectado al pin 9 en cualquiera de nuestras placas de los nodos para el primer, segundo y cuarto bloque de climatización. Como vimos en el capítulo anterior, el eje girará entre 0º y 180º, mientras que nosotros sólo necesitamos un rango de 0º a 90º. Además, la salida del controlador PID es un valor entero entre 0 y 255 que tendremos que convertir a un valor entero entre 0 y 90 para que la placa Arduino mande la orden sobre el servo.

En la figura 8.8, podemos ver un ejemplo de código que hemos empleado para controlar un servomotor. El objeto Servo, representa el servomotor conectado al pin 9 de la placa Arduino Uno y con la función write(), le enviamos la orden para que posicione el eje entre 0º y 90º.

```
Servo myservo;
int posicion_valvula;
myservo.attach(9);

posicion_valvula = (int)(salida_PID*90/255);
myservo.write(posicion_valvula);
```

Figura 8.7 *Código de ejemplo de control de Servomotor empleado.*

Librería para comunicación MQTT entre cada placa Arduino Uno y el nodo central.

La **librería MQTT** [41], permite que nuestro Arduino trabaje como un cliente MQTT para que pueda comunicarse con el nodo central mediante dicho protocolo. Sin embargo, como Arduino tiene una capacidad más limitada que otros dispositivos, la librería nos ofrece casi todas las funcionalidades que permite el protocolo MQTT que vimos durante el capítulo seis. La librería MQTT empleará las librerías vistas más arriba para emplear el protocolo a través de Internet.

En concreto, sólo permitirá una calidad de servicio de nivel 0, es decir, no garantizará la recepción de mensajes. En cualquier caso, como ya vimos en el capítulo seis, no supone un grave problema el perder uno de muchos mensajes enviados de estado del nodo remoto al nodo central. Por otro lado, sí que nos proporciona otras funcionalidades, como usuario y contraseña, último testamento y publicar mensajes con retención. Emplearemos funciones como:

- La función PubSubClient(), para crear un cliente MQTT que represente a la placa Arduino Uno proporcionando la dirección IP o nombre del bróker y su puerto junto a un objeto que represente un cliente Ethernet para emplear la red local en la comunicación.
- La función connect(), para conectar el cliente MQTT que representa la placa con el bróker local situado en el nodo central. Proporcionaremos el identificador del nodo remoto como id del cliente MQTT junto a otras funcionalidades que queramos, como usuario y contraseña o último testamento.
- La función publish(), para publicar un mensaje de estado del bloque empleado por el nodo remoto, que reciba uno de los clientes MQTT en el nodo central a través del tema elegido para dicho bloque de climatización.
- El método loop() del cliente MQTT, que llamaremos en cada iteración del propio loop de la placa Arduino, para comprobar si recibimos un mensaje desde el nodo central.
- El método callback(), para procesar los mensajes recibidos desde el nodo central.

En la figura 8.8, podemos ver un ejemplo de código que hemos empleado para que las placas Arduino actúen como clientes MQTT. En ella, primero definimos todos los parámetros necesarios para crear un cliente MQTT que represente a la placa Arduino del nodo remoto con identificador 1. Durante el método setup() de la placa Arduino, inicializamos la conexión Ethernet y con el bróker local en el nodo central situado en la dirección y puerto proporcionados.

Para crear el mensaje último testamento emplearemos la función ArduinoJson, como en el ejemplo que vimos en la figura 8.6, pero indicando que los elementos del primer bloque de climatización al que pertenece el nodo remoto están desconectados. El mensaje de estado del bloque que publicamos también se hará como en la figura 8.6. Por último, llamamos al método loop() del cliente MQTT, para que en caso de recibir una orden desde el nodo central, la procesemos decodificando el JSON recibido correctamente.

```
byte orden;
double temp_consigna;
byte activado;
byte mac[] = {  0x1E, 0xE1, 0x1A, 0xA1, 0xF1, 0x1F };
byte nodo_central[] = { 192, 168, 1, 52 };


// Callback function header
void callback(char* topic, byte* payload, unsigned int length);
EthernetClient ethClient;
PubSubClient client(nodo_central, 1883, callback, ethClient);


// Callback function
void callback(char* topic, byte* payload, unsigned int length) {

  payload[length]='\0';
  char* orden_central = (char*)payload;
  StaticJsonBuffer<50> buffer_Received;

  JsonObject& received = buffer Received.parseObject(orden central);
  if(received["arduino"]==1){
    orden = received["orden"];
    if(orden == 0){
      temp_consigna = received["consigna"];
    }
    if(orden == 1){
      activado= received["estado"];
    }
  }

}

void setup()
{
  Ethernet.begin(mac);
  if (client.connect("remoto_1", "Arduino_1", 1, 0, testamento)) {
    client.subscribe("Arduino_1");
  }
}

void loop()
{
  client.publish("Bloque_1", estado);

  client.loop();
}
```

Figura 8.8 *Código de ejemplo de librería MQTT empleado.*

Librería para interactuar con el sensor digital de temperatura y humedad DHT22.

La **librería DHT** [47], nos permite recibir las lecturas de temperatura del aire de la sala y de su nivel de la humedad relativa en cada una de las placas Arduino Uno de los nodos remotos que emplean el tercer bloque de climatización.

```
#define DHTPIN 8
#define DHTTYPE DHT22

DHT dht(DHTPIN, DHTTYPE);
float temp_sala;
float humedad_sala;

dht.begin();

humedad_sala = dht.readHumidity();
humedad_sala= dht.readTemperature();
```

Figura 8.9 *Código ejemplo de sensor DHT22 empleado.*

En la figura 8.9, podemos observar un ejemplo de código empleado para obtener dichas lecturas. Como podemos comprobar, el código para interactuar con el sensor es muy sencillo. Para crear un objeto asociado al sensor, emplearemos una instancia de DHT, proporcionándole el tipo de sensor y el pin al que se encuentra conectado. Después, con la función begin() lo podemos inicializar. Por último, con las funciones readHumidity() y readTemperature() podemos recuperar en nuestra placa Arduino la última lectura del nivel de humedad relativa y de la temperatura, en Celsius, realizadas por el sensor.

8.1.3 Software de control.

En este apartado, vamos a terminar de describir la lógica del software de control que hemos programado para las placas Arduino Uno de los cuatro bloques de climatización a controlar por los Arduinos de cada nodo remoto. Para ello, describiremos en primer lugar el diagrama de flujo que siguen cada uno de los cuatro tipos de nodo remoto para concluir especificando las peculiaridades de cada uno.

Diagrama de flujo general.

En la figura 8.10, podemos visualizar el diagrama de flujo que vamos a seguir los cuatro programas de control que hemos realizado para las placas Arduino Uno de cada bloque de climatización. La descripción de dicho diagrama es la siguiente:

1. **Setup**, común a todos los programas. Es la función que se ejecuta al arrancar la placa, e inicializa todas las variables y componentes necesarios para el funcionamiento del nodo remoto. Inicializará actuadores, sensores, cliente MQTT, cliente Ethernet, controlador PID u ON/OFF, pantalla LCD, interruptor, primera lectura potenciómetro de consigna.

2. **Visualización de información en pantalla LCD**. Esta parte es común a los cuatro programas. En ella controlaremos si el usuario quiere encender o apagar la pantalla LCD y actualizaremos la información a mostrar.

3. **Cambio en potenciómetro**, común a todos los programas. La última posición del potenciómetro de consigna se conservará en una variable global para comprobar si el usuario la ha cambiado. En caso de cambio, se llamará a una función para asignar la nueva consigna a partir del potenciómetro.

Hemos decidido realizar el control sobre el bloque de climatización del nodo remoto cada 30 segundos ya que como vimos en el capítulo tres, en un sistema de climatización no se notan los cambios hasta que haya pasado unos pocos minutos ante un cambio en la temperatura deseada. Cada 30 segundos realizaremos en cada Arduino los pasos 4,5 y 6:

4. **Lectura de sondas**, común para los bloques uno, dos y cuatro, que disponen de dos sondas de temperatura DS18B20. Una para detectar la temperatura del agua en el circuito, para que los bloques uno y dos detecten si deben funcionar en modo invierno o verano y otra para que medir la temperatura a controlar. En el bloque tres, disponemos de una sonda de temperatura y humedad DHT22 para la sala a controlar y una sonda de temperatura DS18B20 para medir la temperatura del circuito y detectar el funcionamiento del nodo en modo verano o invierno. En cualquier caso, la placa podrá detectar una avería en cualquier sensor.

5. **Control del bloque**, único para cada bloque de climatización. En esta función se llevará a cabo la estrategia de control propuesta para dicho bloque en el capítulo cuatro. En ella, la placa Arduino leerá las entradas digitales de los elementos del bloque para detectar si están funcionando correctamente o ha ocurrido una avería. Después, empleará las lecturas realizadas en la función anterior para ejecutar un control PID u ON/OFF para determinar la señal de salida a los actuadores de las válvulas de dicho bloque, cumpliendo con las estrategias.

6. **Publicar mensaje de estado**. Con esta función, la placa Arduino Uno creará un mensaje JSON como el ejemplo visto en la figura 8.6 con los parámetros del bloque de climatización del nodo remoto, con el formato diseñado en el capítulo seis. Para publicarlo a través del protocolo MQTT para que lo reciba el nodo central.

7. **Callback**. Con esta función la placa Arduino Uno va a procesar un mensaje recibido desde el nodo central siguiendo el protocolo MQTT. En la figura 8.8 podemos visualizar un ejemplo del método callback empleado, para decodificar el mensaje JSON recibido como una orden desde el nodo central y como es procesado. Si recibimos una orden con identificador 0, significa que el usuario desea cambiar la temperatura deseada o de consigna para el bloque, por lo que asignaremos la nueva consigna recibida a la del bloque. Si recibimos una orden con identificador 1, significa que el usuario quiere conectar o desconectar los elementos conectados al bloque. En caso de recibir un 0, en la siguiente iteración que realicemos un control sobre la instalación, la placa Arduino Uno conectará todos los elementos del bloque. En caso de recibir un 1, en la siguiente iteración que realicemos un control sobre la instalación, la placa Arduino Uno desconectará todos los elementos del bloque.

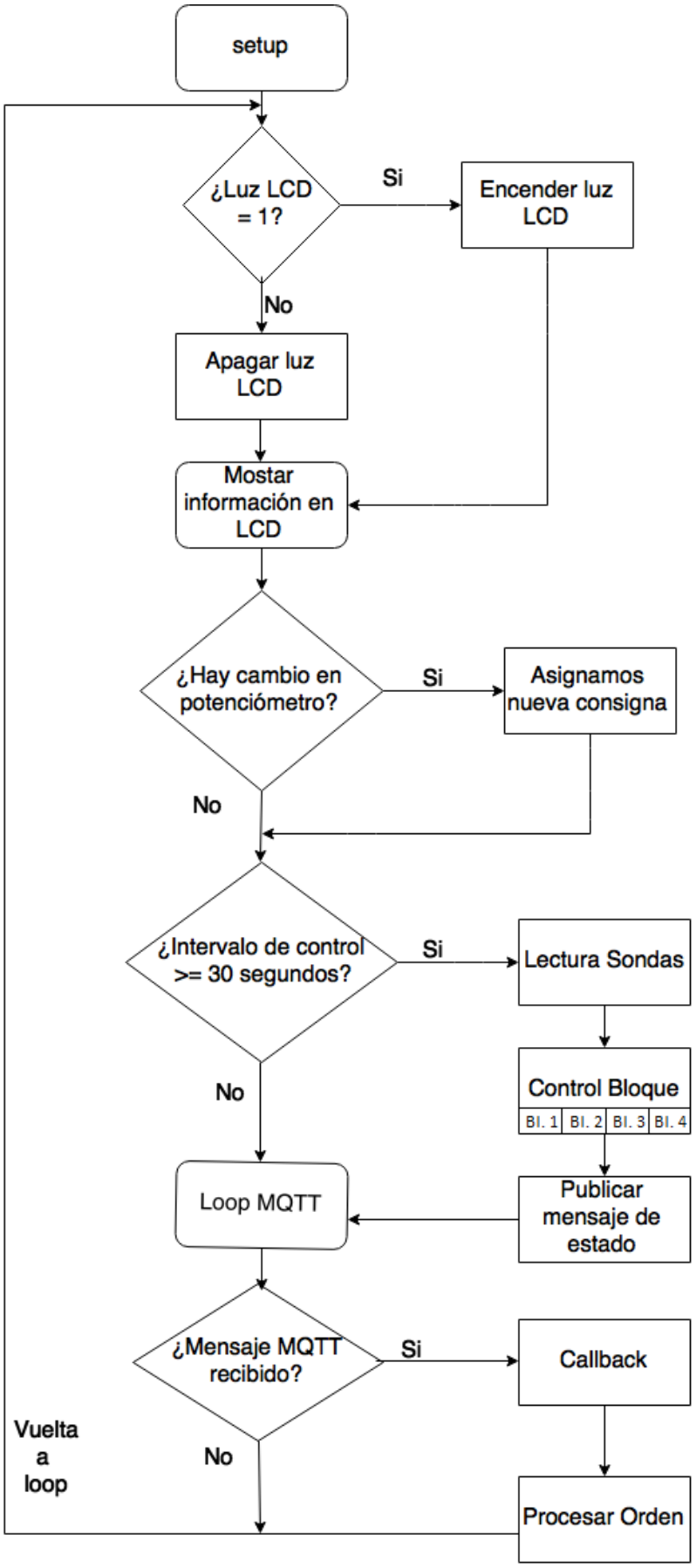


Figura 8.10 *Diagrama de flujo de software de control empleado.*

8.2 Nodo Central

En este apartado vamos a describir la implementación que hemos realizado en el nodo central, es decir, en nuestra Raspberry Pi. En primer lugar, tras realizar la instalación del sistema operativo Raspbian, instalamos tres componentes clave para nuestro sistema de control:

1. **Servidor local MySQL** [34], como vimos en el capítulo seis, nos permitirá crear una base de datos MySQL, en la que crearemos las cuatro tablas para cada uno de los bloques con el formato que definimos en la arquitectura. Estas tablas serán accesibles por cada cliente MQTT encargado de recopilar la información desde cada nodo remoto e introducirlas en la tabla correspondiente para su almacenado, y por la aplicación web.
2. **Bróker Mosquitto** [37], como ya vimos en el capítulo seis, será el bróker que alojaremos en la Raspberry Pi de forma gratuita, al ser de acceso libre. La elección además vino determinado ya que disfruta de un soporte ofrecido por la fundación Eclipse. Se encargará de coordinar la comunicación entre los clientes MQTT del nodo central y de cada nodo remoto.
3. **Servidor web Apache Tomcat** [36], también elegido al ser de acceso libre. Se encargará de recoger las peticiones destinadas a nuestra aplicación web, procesarlas y mostrar las vistas adecuadas en el navegador del cliente.

Una vez completado la instalación y configuración de los tres componentes que acabamos de ver, procedimos a la implementación del software de los clientes MQTT del nodo central y de la aplicación web. En los próximos apartados, vamos a ver la implementación que hemos realizado.

8.2.1 Cliente MQTT

Para la implementación del software de los cuatro clientes MQTT, uno por bloque de climatización, hemos empleado el cliente MQTT Java desarrollado por Eclipse en un proyecto denominado Paho. En [54], podemos ver la documentación oficial del cliente MQTT que hemos empleado. El objetivo del proyecto Paho es el de ofrecer clientes MQTT en diversos lenguajes de programación de acceso libre. En nuestro caso, hemos desarrollado cuatro clases Java que representan a los cuatro primeros bloques de climatización entre los que hay nodos remotos. La función principal de estas clases es la de recibir los mensajes de estado de cada uno de los nodos remotos de nuestro sistema de control e insertar los datos en la tabla correspondiente. Las cuatro clases que hemos desarrollado implementan la clase MQTTCallback que podemos ver en [54]. El motivo por el que hemos escogido dicha clase es porque nos ofrece tres métodos que podemos desarrollar y que se ejecutan en un hilo diferente del hilo principal de ejecución cuando suceden tres eventos. De estos tres eventos el que vamos a usar nos ofrece una función messageArrived(topic, message), que nos permite procesar un mensaje que nos acaba de llegar gracias al bróker de un tema al que estamos suscritos. En la implementación del método messageArrived será dónde se diferencien nuestras cuatro clases. En la figura 8.11, podemos ver el esquema de ejecución que comparten las cuatro clases que hemos desarrollado.

```
private void onRun() {
    while (true) {
        try {
            client = new MqttClient(broker, clientId);

            MqttConnectOptions connOpts = new MqttConnectOptions();
            connOpts.setKeepAliveInterval(60);
            connOpts.setCleanSession(true);
            client.connect(connOpts);

            client.setCallback(this);
            client.subscribe(topic);
            connected = true;

            while (connected) {
                try {
                    Thread.sleep(5000);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        } catch(MqttException e) {
            e.printStackTrace();
            try {
                Thread.sleep(5000);
            } catch(Exception e2) {
                e2.printStackTrace();
            }
        }
    }
}

public static void main(String[] args) {
    cliente_Bloque1 cl = new cliente_Bloque1();
    cl.onRun();
}
```

Figura 8.11 *Esquema de ejecución de cliente MQTT desarrollado.*

La estructura que siguen los cuatro clientes MQTT es sencilla. En primer lugar, cada una define una instancia de la clase MQTTClient, sobre la que van a ajustar los parámetros de la conexión con el bróker Mosquitto y se va a realizar una conexión con él. Tras esto, el cliente estará a la espera de recibir nuevos mensajes de los editores de su bloque de climatización, es decir, ciertos nodos remotos. Cada vez que le llega un nuevo mensaje de estado de uno de sus nodos remotos realizará lo siguiente en la función messageArrived de MQTTCallback:

1. En primer lugar, recuperará el mensaje recibido como un objeto JSON empleando la librería para Java que podemos encontrar en la página oficial con la documentación de JSON para Java [53]. Gracias a esta librería, podremos recuperar las claves y su valor asociado de los elementos del JSON de dicho bloque ya que sigue el formato que definimos en el capítulo seis. El JSON que maneja el cliente MQTT de cada bloque tendrá unos parámetros diferentes, en función del formato definido para cada bloque. En la figura 8.12, podemos ver como recuperamos los elementos de un JSON en uno de los bloques.

```java
public void messageArrived(String topic, MqttMessage message) {

    String toJSON = new String(message.getPayload());


    try{
        JSONObject json = new JSONObject(toJSON);
        int arduino = json.getInt("arduino");
        String estado = json.getString("estado");
        String climatizador = json.getString("climatizador");
        String recuperador = json.getString("recuperador");
        String bomba = json.getString("bomba");
        String sonda_circuito = json.getString("sonda_circuito");
        Double temperatura = json.getDouble("temperatura");
        Double consigna = json.getDouble("consigna");
```

Figura 8.12 *Lectura de JSON en cliente MQTT desarrollado.*

2. En segundo lugar, procederemos a introducir los datos en la tabla de la base de datos correspondiente. Para ello, primero realizaremos una conexión con la base datos con un driver JDBC para realizar una consulta en la que insertar los datos recogidos del mensaje. Al igual que en el punto anterior, cada cliente insertará los datos en una tabla, correspondiente con los datos recogidos y los parámetros que definimos para la tabla de dicho bloque en el capítulo seis.

```java
String query = "INSERT INTO Bloque_1
(arduino,estado,climatizador,recuperador,bomba,sonda_circuito,temperatura,consigna)
VALUES (?,?,?,?,?,?,?,?)";


PreparedStatement preparedStmt = conn.prepareStatement(query);
preparedStmt.setInt(1, arduino);
preparedStmt.setString(2, estado);
preparedStmt.setString(3, climatizador);
preparedStmt.setString(4, recuperador);
preparedStmt.setString(5, bomba);
preparedStmt.setString(6, sonda_circuito);
preparedStmt.setDouble(7, temperatura);
preparedStmt.setDouble(8, consigna);
```

Figura 8.13 *Consulta de inserción de datos en tabla por cliente MQTT desarrollado.*

8.2.2 Aplicación web Java

Como vimos en capítulo seis, vamos a desarrollar una aplicación web Java con el objetivo de ofrecer una interfaz para el usuario en la que pueda acceder a la información de cualquier nodo remoto en un determinado momento. Además, que permita el envío de órdenes a cada uno de ellos. En este apartado, vamos a describir las vistas que forman nuestra aplicación web Java y detallaremos las funciones principales de los Servlets que la forman.

La primera vista que podemos visualizar según entremos en la aplicación web es la vista de la figura 8.11. Como dijimos en el capítulo seis, sólo el usuario encargado del sistema de control podrá visualizar los datos y el resto de vistas de la aplicación web. En esta pantalla, el usuario podrá autenticarse, para acceder a la siguiente vista. Para autenticarse, como podemos ver en la figura 8.11, disponemos de un formulario en la parte superior para que el usuario introduzca su nombre y contraseña. Una vez que confirme el envío de dicho

formulario, un Servlet de login se encargará de procesar dicho formulario, comprobando que los datos introducidos se corresponden con el usuario existente en la tabla Usuarios de nuestra base de datos. En caso de autenticarse correctamente, se avanzará a la próxima vista.

En caso de autenticarnos correctamente, avanzaremos a la pantalla principal de la aplicación web. Como podemos ver en la figura 8.12, disponemos de 5 columnas que organizan los nodos remotos que emplean un determinado bloque de climatización. A través de las filas de dichas columnas, podemos avanzar a la vista de un determinado nodo remoto. Si pulsamos sobre cualquiera de las filas de una de las cinco columnas, nuestro navegador enviará una petición GET con un parámetro indicando el nodo remoto que se ha seleccionado, que identificará el elemento que queremos visualizar. Un Servlet se encargará de procesar dicha petición GET, y recoger el parámetro que identifica al nodo remoto que queremos visualizar. A partir de dicho identificador, el Servlet recuperará de la tabla de la base de datos a la que pertenece el nodo el último informe de estado recibido. Para ello empleará una clase JDBC, como driver para conectar con la base de datos alojada en nuestro servidor local MySQL y recuperar dicha información. Una vez que se ha recuperado la información del nodo correspondiente, se almacenará en una clase Java Bean con los atributos de dicho nodo, como vimos en el capítulo seis, que enviará a la vista correspondiente del bloque al que pertenece el nodo para visualizar la información.

En las figuras 8.14, 8.15, 8.16, 8.17, 8.18 y 8.19 podemos ver las cuatro vistas que disponemos para los elementos de cada bloque de climatización. En ella, podemos ver los elementos que lo forman y podemos enviar dos tipos de órdenes. La primera orden, será la de cambio de consigna y la segunda la de encendido/apagado del nodo remoto. Además, como podemos ver, disponemos de un botón de Log Out con el que el usuario podrá cerrar la sesión. Un Servlet se encargará de procesar la petición e invalidar la sesión.

Un Servlet se encargará de procesar la petición POST tras confirmar el envío de cualquiera de los dos formularios de envío de orden. Dicho formulario, tendrá dos inputs de tipo hidden, uno para identificar la orden de la que se trata y otro para identificar al nodo al que debe enviarse la orden. Tras identificar la orden de la que se trata, el Servlet recuperará los datos necesarios del formulario, es decir, la nueva temperatura de consigna o el nuevo estado que se quiere asignar al nodo remoto. Tras esto, comprobará que los datos del formulario son correctos, en concreto, que la nueva temperatura de consigna se encuentra entre los límites establecidos para dicho bloque de climatización, como vimos en el tercer capítulo, o que el Arduino del nodo remoto se encuentre conectado. En caso de que la nueva consigna no sea válida, no se enviará la orden, y se mostrará al usuario un cuadro de diálogo informativo, como podemos ver en la figura 8.20. En caso de ser correcto, también se mostrará al usuario un cuadro informativo. Sin embargo, el Servlet tendrá que enviar la orden al nodo remoto correspondiente. Para enviar la orden, tras obtener los parámetros necesarios desde el formulario de una u otra orden, el Servlet creará un mensaje JSON, con el formato que definimos en el capítulo seis, para enviárselo a dicho nodo remoto. Para enviarle dicho mensaje, el Servlet creará una instancia de una clase MQTTClient [54], en la que realizará una conexión con el bróker local Mosquitto, ajustando los parámetros necesarios y publicará en el tema correspondiente a la placa Arduino del Nodo remoto destino, con calidad de servicio 1, para garantizar que dicho mensaje llegue a la placa. Tras publicar dicho mensaje, el bróker se encargará de entregárselo al Arduino correspondiente suscrito al tema de destino de la orden y el Servlet cerrará la conexión con el bróker para volver a cargar a la vista del nodo remoto.

Figura 8.14 *Vista Inicial de la aplicación web desarrollada.*

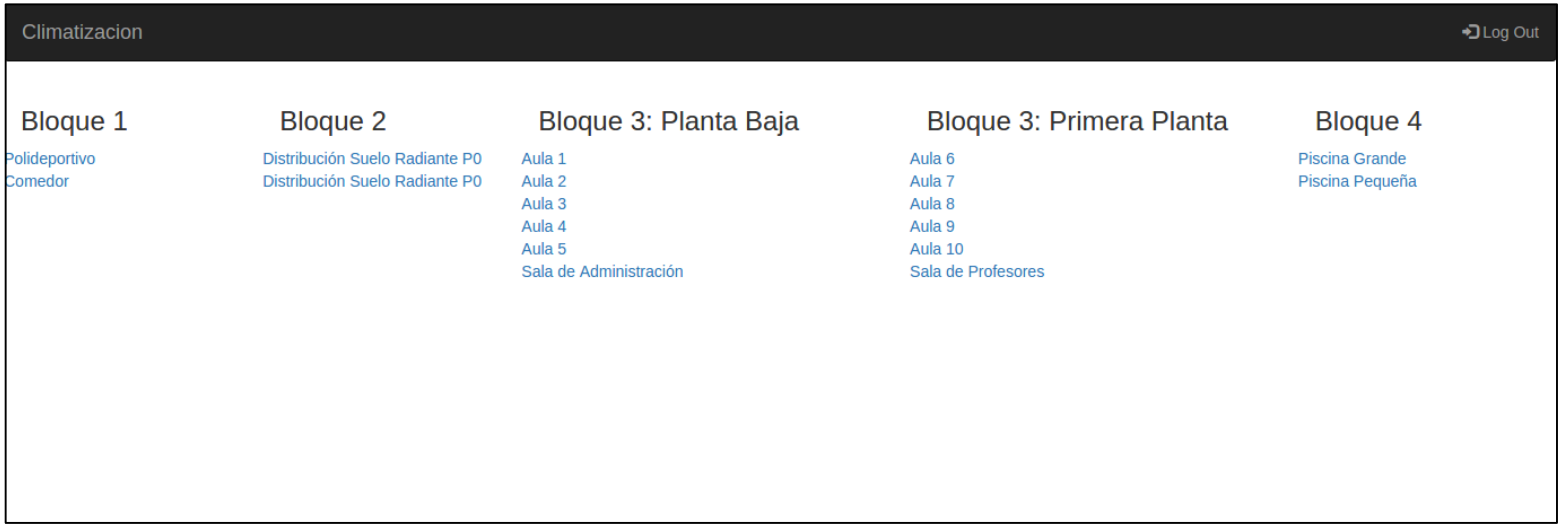


Figura 8.15 *Vista principal de la aplicación web desarrollada.*

Figura 8.16 *Vista nodo remoto bloque 1 de la aplicación web desarrollada.*

Figura 8.17 *Vista nodo remoto bloque 2 de la aplicación web desarrollada.*

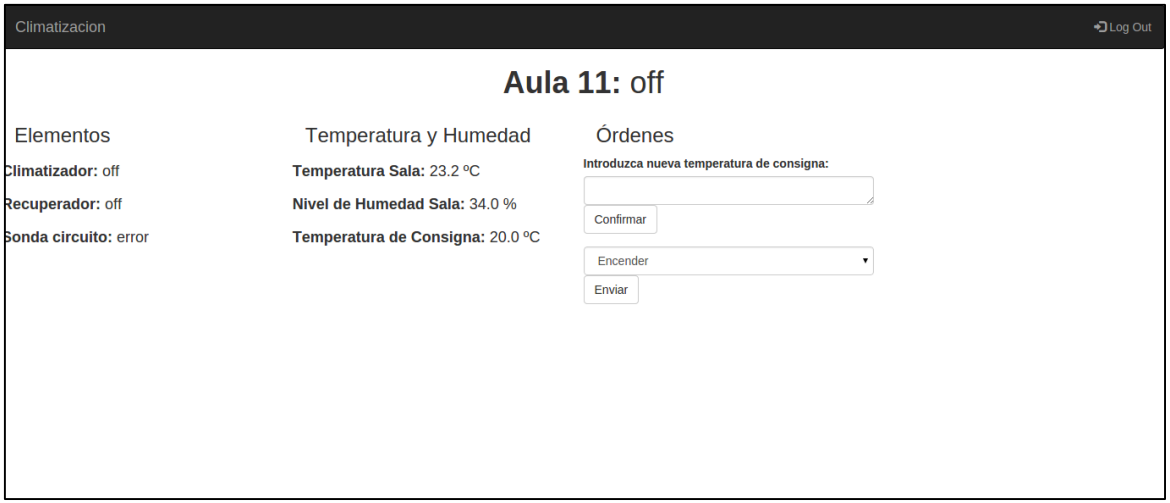


Figura 8.18 *Vista nodo remoto bloque 3 de la aplicación web desarrollada.*

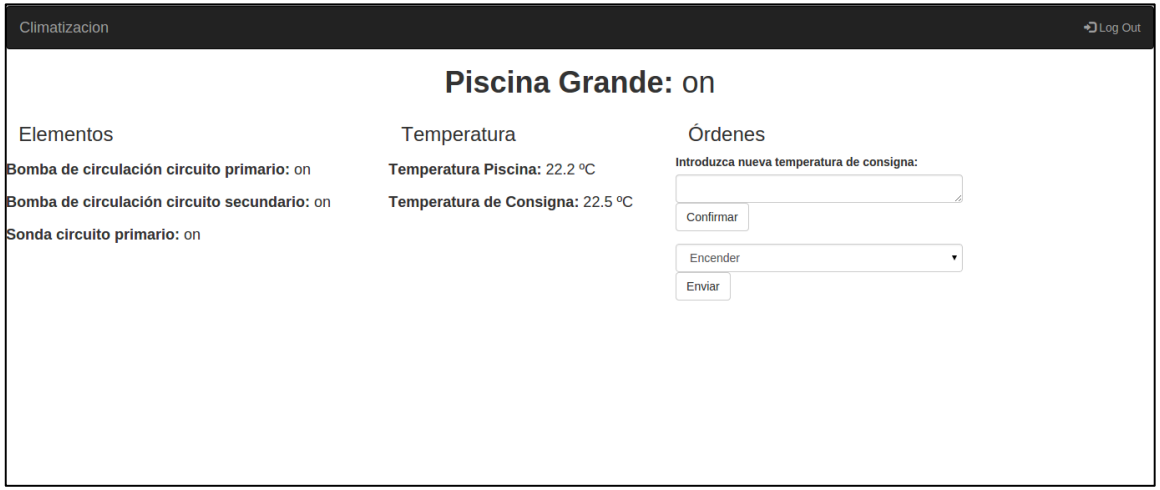


Figura 8.19 *Vista nodo remoto bloque 4 de la aplicación web desarrollada.*

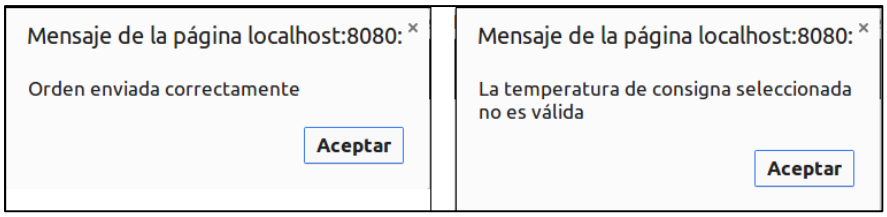


Figura 8.20 *Mensajes de confirmación o error de orden central.*

Capítulo 9: Pruebas de funcionamiento

En este capítulo vamos a describir una serie de pruebas que hemos realizado sobre el sistema domótico que hemos realizado. En concreto, hemos realizado tres tipos de prueba. El primer tipo han sido pruebas sobre los nodos remotos, para determinar que el funcionamiento de los sensores fuera el correcto y que ejecutasen la orden correspondiente sobre el actuador requerido. El segundo tipo de pruebas que hemos realizado ha sido sobre la aplicación web Java desarrollada. El tercer tipo de pruebas que hemos realizado han sido pruebas respecto a la comunicación entre un nodo remoto y el nodo central, para comprobar el funcionamiento general del sistema domótico desarrollado.

9.1 Pruebas en nodo remoto

En este apartado vamos a describir una serie de pruebas que hemos realizado para determinar el funcionamiento de los nodos remotos. En este apartado hemos tratado de determinar en primer lugar el correcto funcionamiento de los sensores tanto de temperatura, como de temperatura y humedad que hemos empleado. Además, hemos tratado de verificar que los nodos de cada bloque ejecuten la orden sobre sus actuadores siguiendo las estrategias de control determinadas en el capítulo cuatro. Por último, verificaremos el comportamiento de la interfaz auxiliar controlada por el nodo remoto.

Para simplificar la explicación de las pruebas que hemos realizado, pondremos el caso de dos nodos remotos. El nodo remoto del bloque uno, cuyos actuadores y sensores son iguales que para los nodos remotos encargados de controlar los bloques dos y cuatro de climatización. Y el nodo remoto del bloque tres, que dispone de unos sensores y actuadores distintos al resto de bloques. En la figura 9.1 podemos ver una “protoboard” que hemos empleado para simular el nodo remoto del bloque del primer bloque de climatización. Los elementos que hemos conectado a la placa Arduino Uno (con shield Ethernet), a los pines de la placa vistos en el capítulo siete, empezando por la izquierda, son:

- **Círculo amarillo: sondas de temperatura DS18B20**, una para medir la temperatura del agua del circuito del bloque y otra para medir la temperatura de la sala.
- **Círculo negro: LEDS**. El primer LED por la izquierda, servirá para simular el relé salida a climatizador. El LED del centro es un led auxiliar que encenderemos cuando la placa tenga orden remota de encendido y apagaremos cuando tenga orden remota de apagado. El LED de la derecha servirá para simular el relé salida a recuperador.
- **Círculo azul: pulsador** interfaz auxiliar, servirá para controlar la luz de la pantalla LCD.
- **Círculo rojo: pulsadores**, para simular las entradas de estado de la bomba de circulación de agua, climatizador y recuperador.
- **Círculo gris: servomotor**, para simular el servomotor que será el actuador encargado de dirigir la posición de la válvula mezcladora.
- **Círculo blanco: potenciómetro**, para seleccionar manualmente una temperatura de consigna. Y **pantalla LCD** 16x2, interfaz auxiliar.

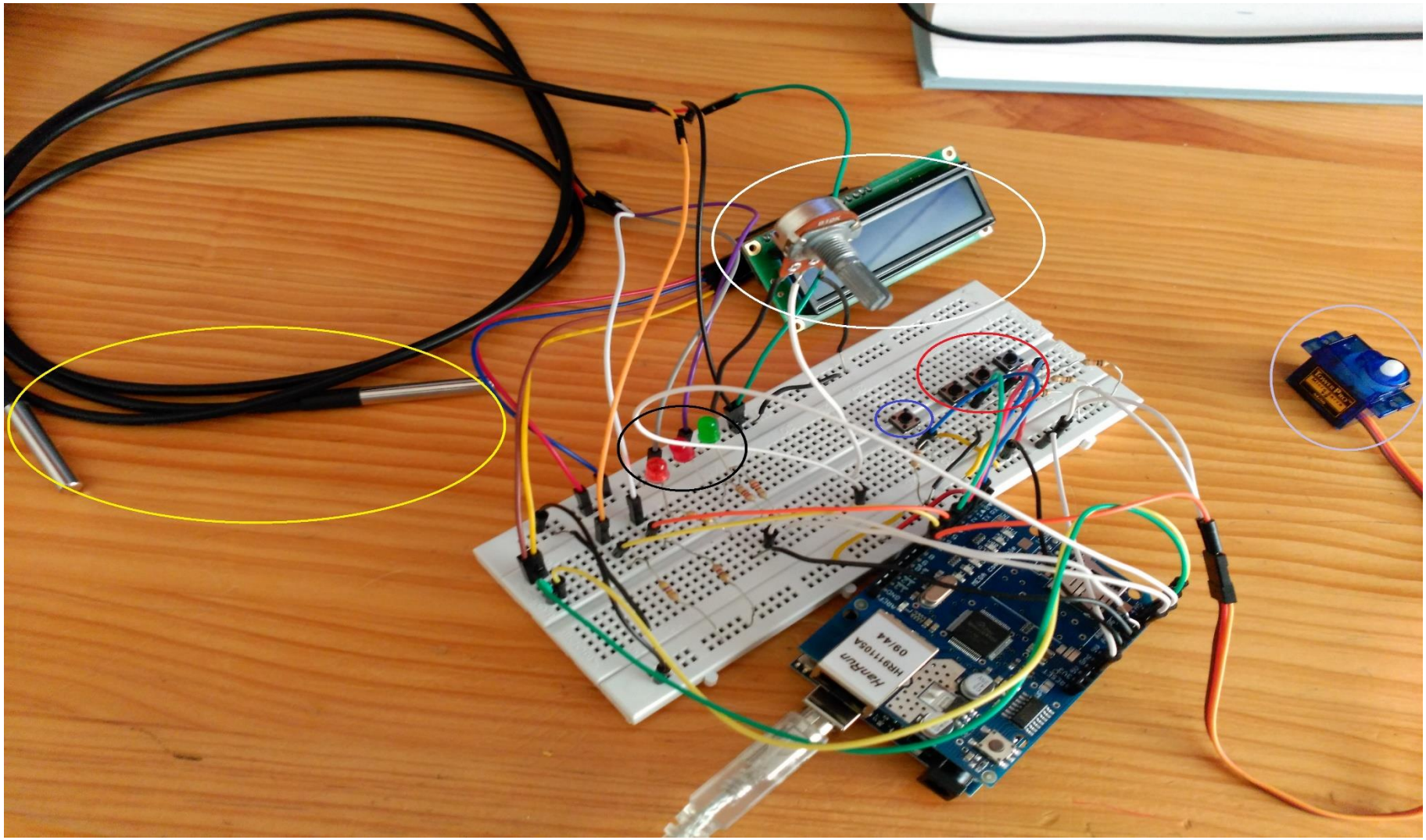


Figura 9.1 *Protoboard empleada para nodo remoto bloque 1.*

En primer lugar, hemos comprobado de forma satisfactoria que los sensores realicen las lecturas de temperatura de forma adecuada tanto en el aire, como en agua. Para medir la temperatura del agua con el sensor DS18B20 lo hemos sumergido en un vaso de agua. Después, hemos comprobado que el sensor de la temperatura del agua del circuito detectará si estaba funcionando en modo invierno o en modo verano, para ejecutar el control PID del modo correspondiente y enviase una orden de actuación al servomotor de la válvula mezcladora. En la figura 9.2, podemos visualizar el código de control empleado y el resultado obtenido por el monitor serial.

```
if( temp_sensor_bomba >= cambio_IV ){
  modo_IV = 0;
  PID_calor.Compute();
  Serial.println("Modo Invierno detectado");
  Serial.print("Consigna: ");
  Serial.println(temp_consigna_I);
}else{
  modo_IV = 1; //Verano
  PID_frio.Compute();
  Serial.println("Modo Verano detectado");
  Serial.print("Consigna: ");
  Serial.println(temp_consigna_V);
}
int salida_servo = salida_PID*90/255;
servo_valvula.write(salida_servo);
digitalWrite(pin_marcha_paro, HIGH);
Serial.print("Salida PWM PID: ");
Serial.println(salida_PID);
Serial.print("Salida a servomotor: ");
Serial.println(salida_servo);
```

Resultado en monitor serial

```
Sensor sala: 22.75
Sensor circuito: 19.25
Modo Verano detectado
Consigna: 21.00
Salida PWM PID: 68.80
Salida a servomotor: 24
```

Figura 9.2 *Código y resultado de simulación todo on y ejecutar control PID.*

En segundo lugar, hemos comprobado el funcionamiento según la orden remota y el estado recibido por los elementos del bloque. Para ello, hemos empleado los pulsadores del círculo rojo de la figura 9.1 en modo pull-up. De esta forma, cuando no estén pulsados recibimos un estado del elemento de dicho pulsador con valor lógico 1, es decir, que se encuentra funcionando. Cuando los pulsemos, obtendremos un 0, cuyo estado es que el elemento en cuestión se encuentra apagado. Para comprobar los resultados, podemos ver que LEDS del círculo negro se encuentran encendidos y cuáles no. Además, podemos comprobar el resultado en la interfaz auxiliar (Pantalla LCD).

- En la **figura 9.3**, podemos visualizar el resultado de tener orden remota de encendido, con todos los LEDS encendidos y pulsadores con valor 1.
- En la **figura 9.4**, podemos visualizar el resultado de orden remota de apagado, con todos los LEDS apagados.
- En la **figura 9.5**, podemos visualizar el resultado de tener orden remota de encendido. Sin embargo, hemos pulsado el pulsador del climatizador, obteniendo un 0 en dicha entrada en nuestra placa Arduino Uno. Al tener orden remota de encendido, esto supone una avería, como vemos reflejado en la pantalla LCD. Si tuviésemos orden

remota de apagado, en vez de “ER”, tras “CL” aparecería “OF”. En cualquier caso, el LED izquierdo podemos ver que se encuentra apagado.

- En la **figura 9.6** podemos visualizar el resultado de tener orden remota de encendido. Sin embargo, hemos pulsado el pulsador del recuperador, obteniendo un 0 en dicha entrada en nuestra placa Arduino Uno. Al tener orden remota de encendido, esto supone una avería, como vemos reflejado en la pantalla LCD. Si tuviésemos orden remota de apagado, en vez de “ER”, tras “R” aparecería “OF”. En cualquier caso, el LED derecho podemos ver que se encuentra apagado.
- En la **figura 9.7** podemos visualizar el resultado de tener orden remota de encendido. Sin embargo, hemos pulsado el pulsador de la bomba de circulación, obteniendo un 0 en dicha entrada en nuestra placa Arduino Uno. Al tener orden remota de encendido, esto supone una avería, como vemos reflejado en la pantalla LCD. Si tuviésemos orden remota de apagado, en vez de “ER”, tras “B5” aparecería “OF”.
- En la **figura 9.8** podemos ver la prueba realizada, de forma satisfactoria, para comprobar la modificación del punto de consigna desde la interfaz auxiliar de usuario en cada nodo remoto.

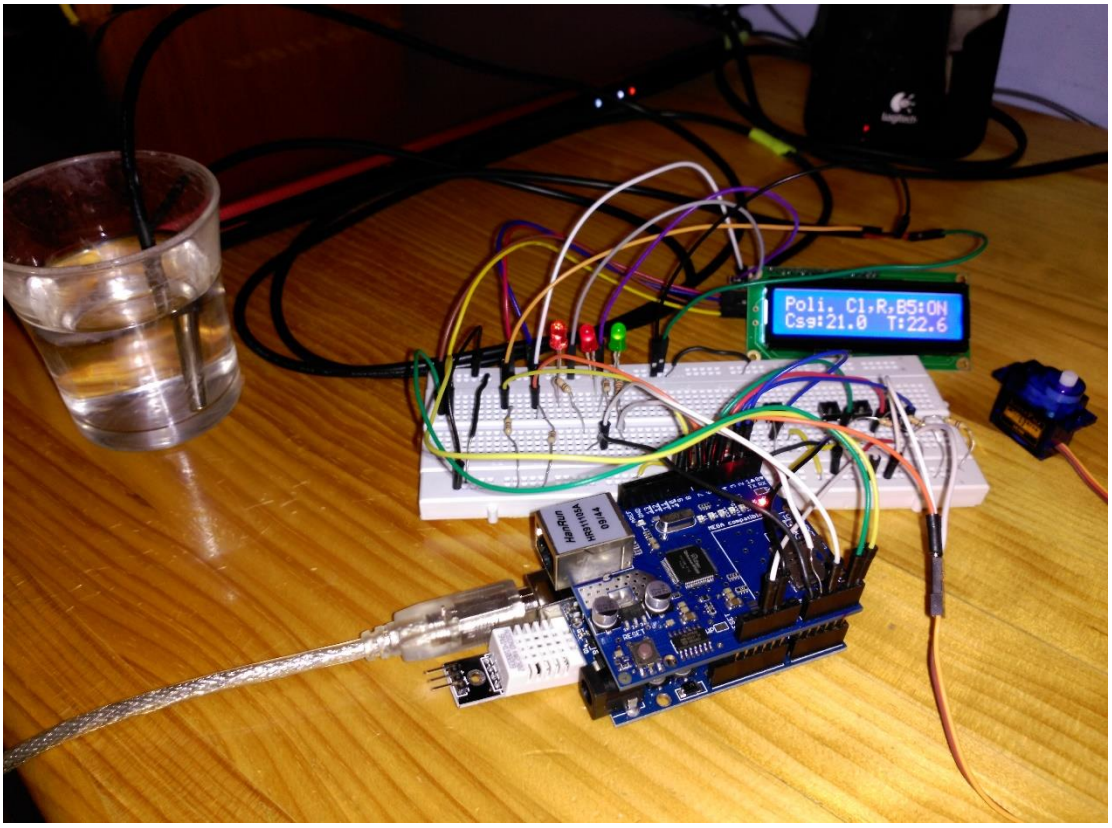


Figura 9.3 *Prueba con todos los elementos ON y orden remota de encendido realizada.*

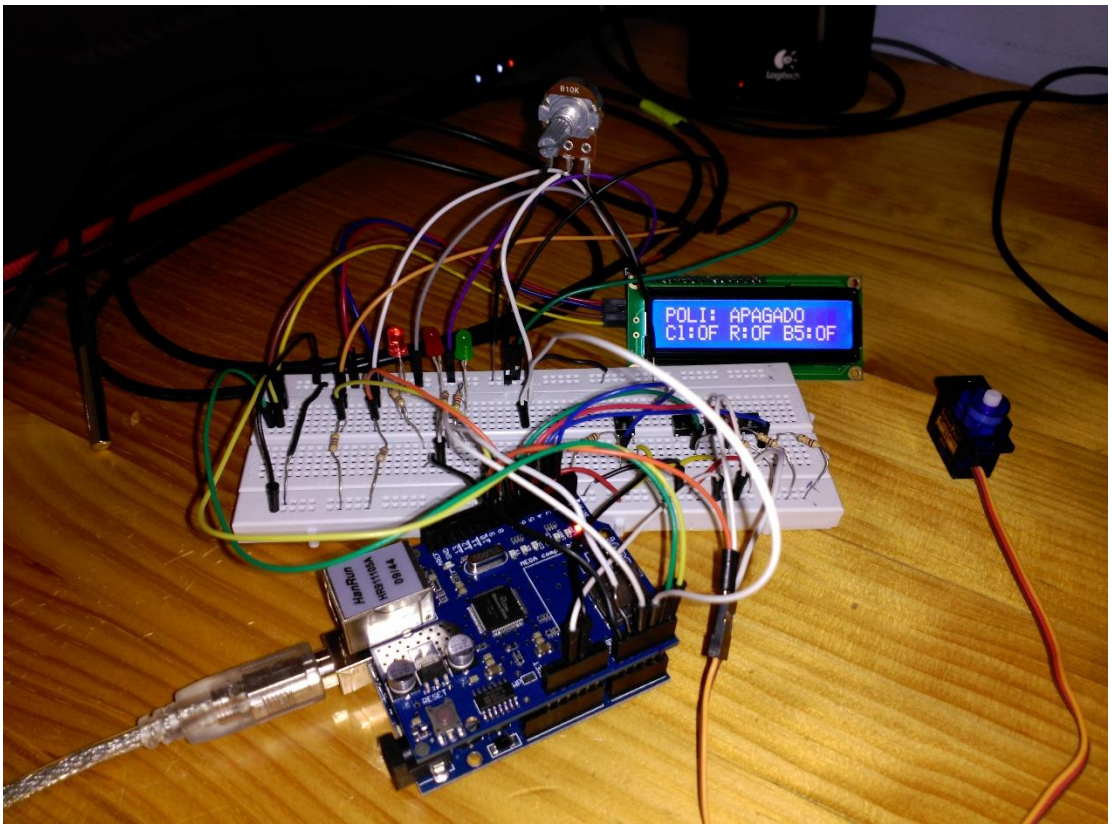


Figura 9.4 *Prueba con todos los elementos OFF y orden remota de apagado realizada.*

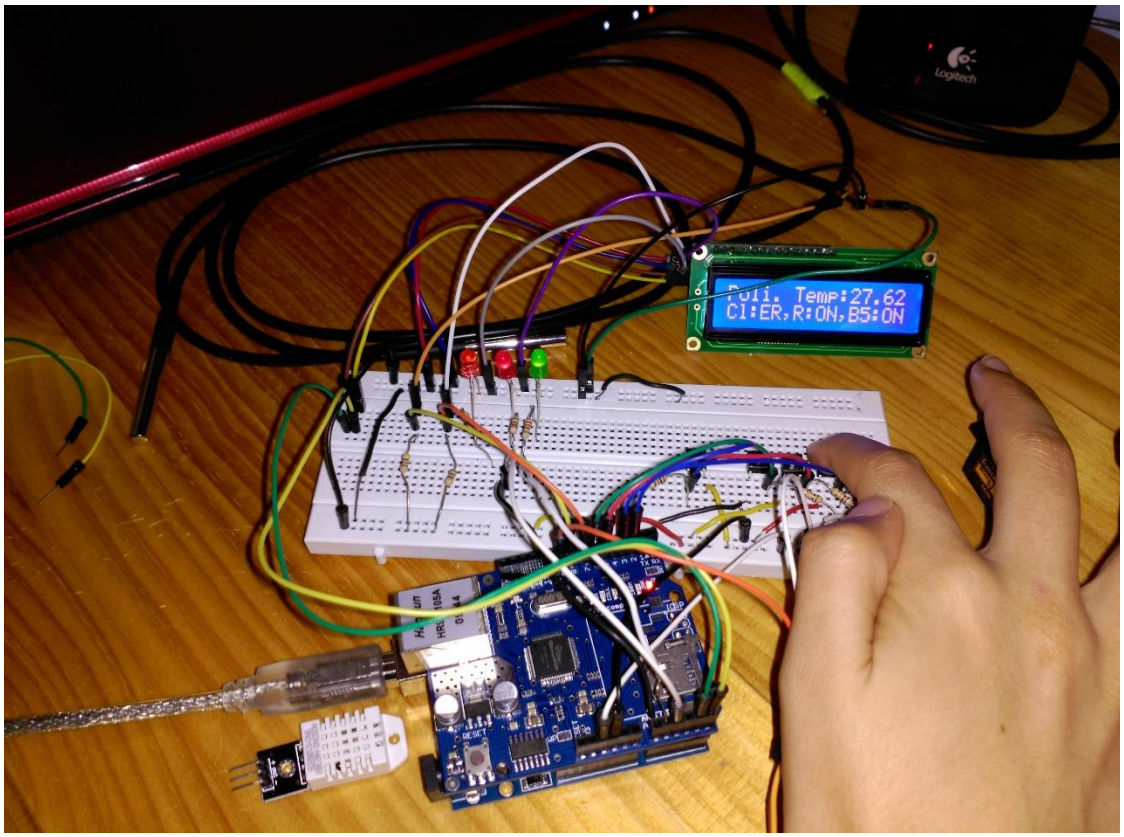


Figura 9.5 *Prueba con error en climatizador y orden remota de encendido realizada.*

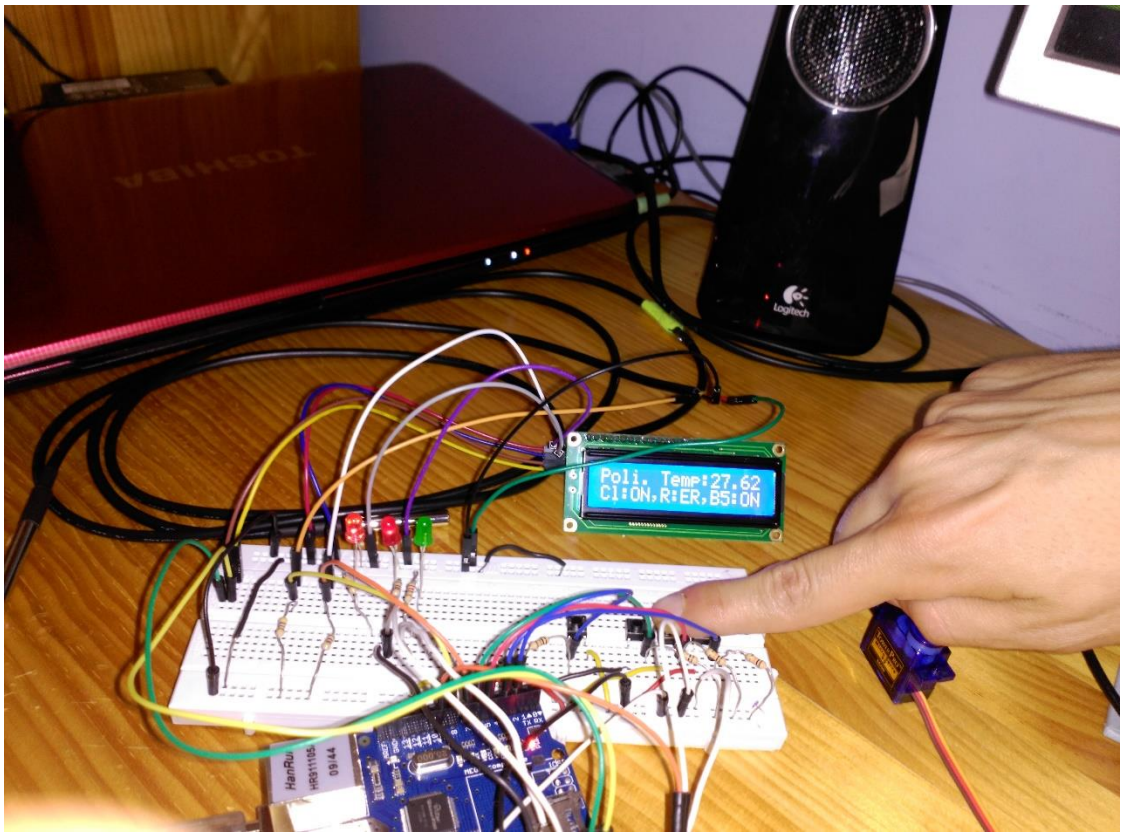


Figura 9.6 *Prueba con error en recuperador y orden remota de encendido realizada.*

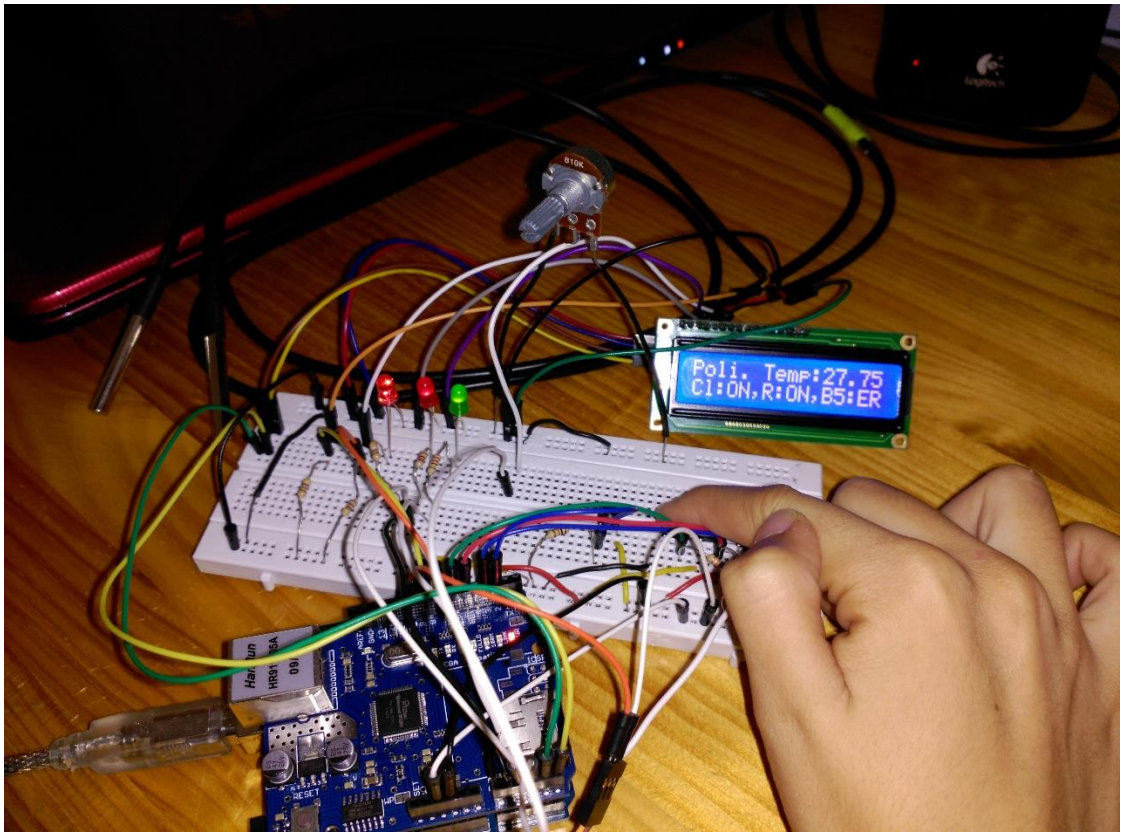


Figura 9.7 *Prueba con error en bomba de circulación y orden remota de encendido realizada.*

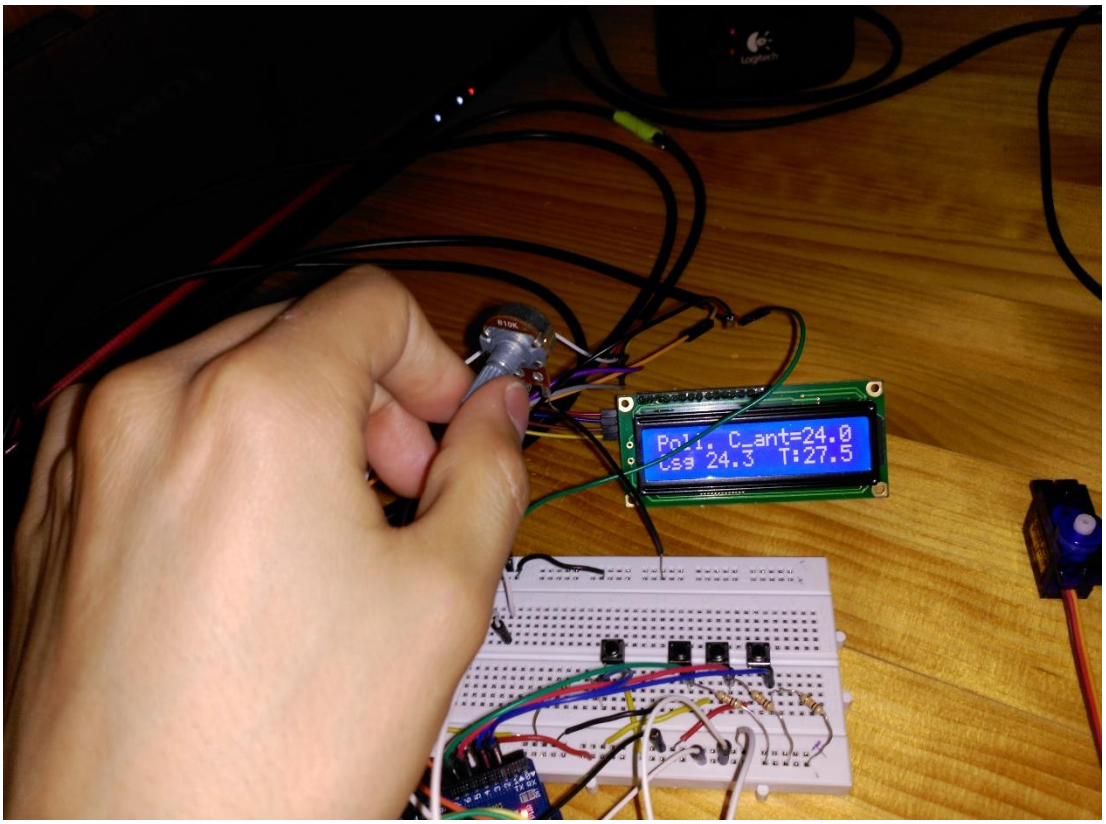


Figura 9.8 *Prueba de selección de temperatura de consigna realizada.*

En la figura 9.9 podemos ver una "protoboard" que hemos empleado para simular el nodo remoto del bloque del tercer bloque de climatización. Los elementos que hemos conectado a la placa Arduino Uno (con shield Ethernet), correspondientes con los pines de la placa seleccionados en el capítulo siete, empezando por la izquierda, son:

- **Círculo amarillo: sonda de temperatura DS18B20**, para medir la temperatura del agua del circuito del bloque.
- **Círculo negro: LEDS**. El primer LED por la izquierda, es un led auxiliar que encenderemos cuando la placa tenga orden remota de encendido y apagaremos cuando tenga orden remota de apagado. El segundo LED por la izquierda, servirá para simular el relé actuador para la válvula de la segunda etapa. El LED del centro, servirá para simular el relé actuador para la válvula de la primera etapa. El segundo LED por la derecha servirá para simular el relé salida a climatizador. El primer LED por la derecha servirá para simular el relé salida a climatizador.
- **Círculo azul: sensor de temperatura y humedad DHT22**, para medir la temperatura y humedad relativa del aire de la sala.
- **Círculo rojo: pulsador** interfaz auxiliar, servirá para controlar la luz de la pantalla LCD.
- **Círculo morado: pulsadores**, para simular las entradas de estado de la bomba del climatizador y recuperador.
- **Círculo blanco: potenciómetro**, para seleccionar manualmente una nueva temperatura de consigna. Y **pantalla LCD** 16x2, interfaz auxiliar.

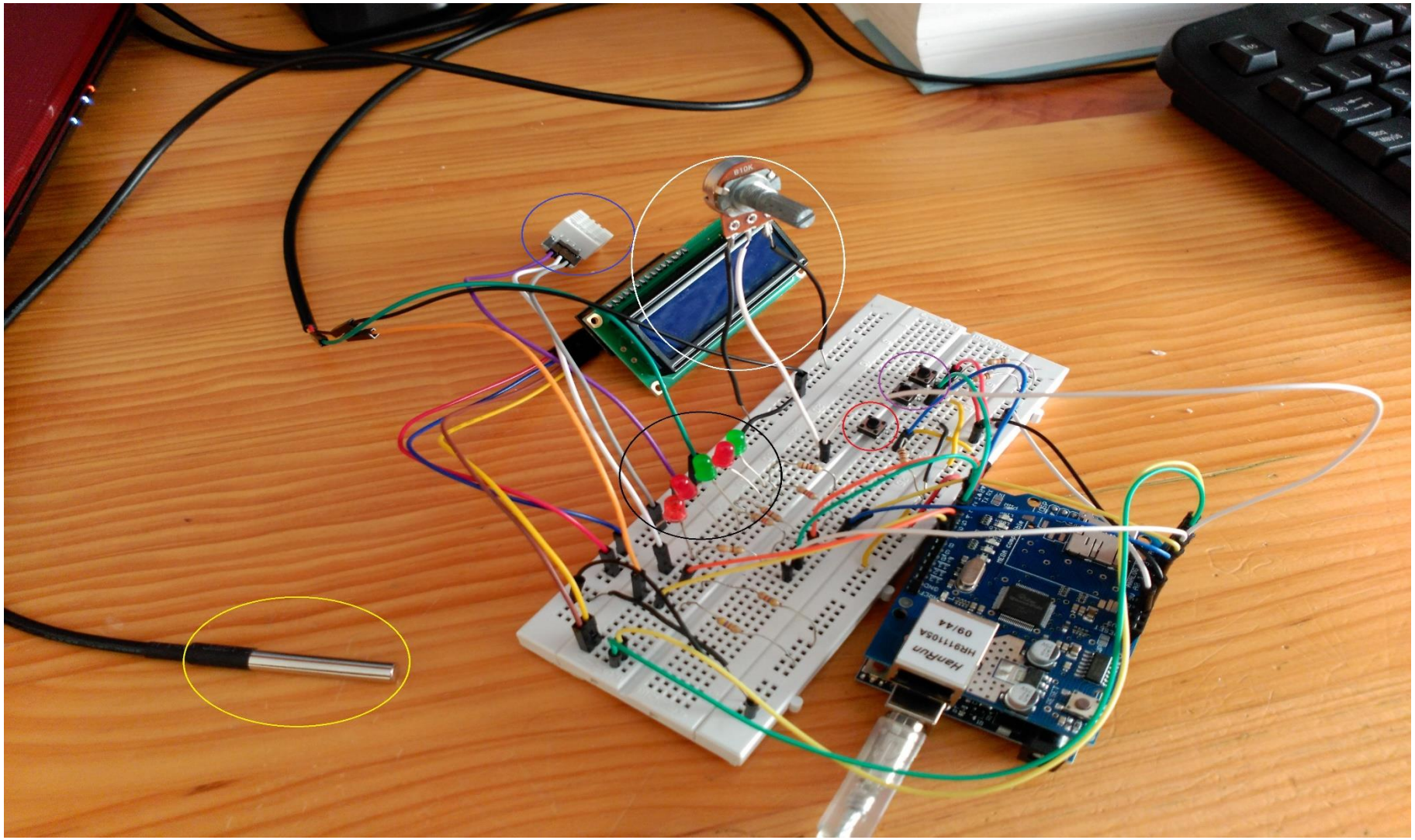


Figura 9.9 *Protoboard empleada para nodo remoto bloque 3.*

En primer lugar, hemos comprobado al igual que en el caso anterior con el bloque uno, los diferentes resultados en función del estado del climatizador y recuperador, obteniendo el mismo resultado satisfactorio respecto a los LEDS que enciende, como en el caso anterior.

En segundo lugar, hemos comprobado de forma satisfactoria que el sensor DS18B20 mida correctamente la temperatura del agua del circuito, para ello, lo hemos sumergido en un vaso de agua. También hemos comprobado de forma satisfactoria que el sensor DHT22 mida la temperatura del aire de la sala y su nivel de humedad relativa. Por último, hemos comprobado que la placa Arduino Uno realice el control ON/OFF sobre el bloque siguiendo la estrategia de control planteada para el bloque en el capítulo cuatro. En la figura 9.10, podemos visualizar el código de control empleado y el resultado para los diferentes casos.

```
if( temp_sensor_bomba >= cambio_IV ){
  modo_IV = 0;
  Serial.println("Modo Invierno");
  Serial.print("Temperatura consigna: ");
  Serial.println(temp_consigna_I);
  if(temp_sensor_DHT > temp_consigna_I){
    Serial.println("Desactivamos ambas etapas");
    digitalWrite(pin_valvula_1, LOW);
    digitalWrite(pin_valvula_2, LOW);
  }else{
    digitalWrite(pin_valvula_1, HIGH);
    Serial.println("Activada primera etapa auxiliar");
    if(temp_sensor_DHT < (temp_consigna_I - 2) ){
      Serial.println("Activada segunda etapa auxiliar");
       digitalWrite(pin_valvula_2, HIGH);
     }
  }
}else{
    modo_IV = 1; //Verano
    Serial.println("Modo Verano");
    Serial.print("Temperatura consigna: ");
    Serial.println(temp_consigna_V);
    if(temp_sensor_DHT < temp_consigna_V){
      Serial.println("Desactivamos ambas etapas");
      digitalWrite(pin_valvula_1, LOW);
      digitalWrite(pin_valvula_2, LOW);
    }else{
      digitalWrite(pin_valvula_1, HIGH);
      Serial.println("Activada primera etapa auxiliar");
      if(temp_sensor_DHT > (temp_consigna_V + 2) ){
         Serial.println("Activada segunda etapa auxiliar");
         digitalWrite(pin_valvula_2, HIGH);
       }
    }
    if(humedad_sensor_DHT > humedad_critica){
        Serial.println("Activada segunda etapa condensacion");
        digitalWrite(pin_valvula_2, HIGH);
      }
  }
  Serial.print("Nivel Humedad critica: ");
  Serial.println(humedad_critica);
  digitalWrite(pin_marcha_paro, HIGH);
```

Resultado monitor serial caso 1

```
Temperatura Sala: 28.80
Humedad Sala: 23.60
Modo Verano
Temperatura consigna: 28.00
Activada primera etapa auxiliar
Nivel Humedad critica: 80
```

Resultado monitor serial caso 2

```
Temperatura Sala: 28.80
Humedad Sala: 23.60
Modo Verano
Temperatura consigna: 21.00
Activada primera etapa auxiliar
Activada segunda etapa auxiliar
Nivel Humedad critica: 80
```

Resultado monitor serial caso 3

```
Temperatura Sala: 28.70
Humedad Sala: 24.20
Modo Invierno
Temperatura consigna: 25.00
Desactivamos ambas etapas
Nivel Humedad critica: 80
```

Resultado monitor serial caso 4

```
Temperatura Sala: 28.70
Humedad Sala: 24.60
Modo Verano
Temperatura consigna: 29.50
Desactivamos ambas etapas
Activada segunda etapa condensacion
Nivel Humedad critica: 20
```

Figura 9.10 *Código y resultados de controlador ON/OFF para bloque 3 realizado.*

- En el **primer caso**, la temperatura de la sala es mayor que la temperatura deseada, en modo verano, pero sólo es necesario activar la primera etapa. Como podemos comprobar en la figura 9.11, dentro del círculo amarillo, sólo encendemos el LED

derecho (primera etapa), mientras que el LED izquierdo (segunda etapa), se encuentra desconectado. Además, en la figura 9.10, podemos visualizar la salida del monitor serial obtenida para el primer caso.

- En el **segundo caso**, en modo verano, la temperatura de la sala es mayor que la temperatura deseada, pero es necesario activar ambas etapas. Como podemos comprobar en la figura 9.12, dentro del círculo amarillo, encendemos tanto el LED derecho (primera etapa) como el LED izquierdo (segunda etapa). Además, en la figura 9.10, podemos visualizar la salida del monitor serial obtenida para el segundo caso.

- En el **tercer caso**, la temperatura de la sala, en modo invierno, es mayor que la temperatura de consigna, por tanto ambas etapas están desconectadas. Como podemos comprobar en la figura 9.13, dentro del círculo amarillo, apagamos tanto el LED derecho (primera etapa) como el LED izquierdo (segunda etapa). Además, en la figura 9.10, podemos visualizar la salida del monitor serial obtenida para el tercer caso.

- En el **cuarto caso**, la temperatura de la sala es menor que la temperatura de consigna, pero el nivel relativo de humedad supera el límite establecido por riesgo de condensación. Como podemos comprobar en la figura 9.11, dentro del círculo amarillo, sólo encendemos el LED izquierdo (segunda etapa), mientras que el LED derecho (primera etapa), se encuentra desconectado. Además, en la figura 9.10, podemos visualizar la salida del monitor serial obtenida para el cuarto caso. (nota: en las estrategias de control vimos que el nivel de humedad crítico era del 80%. Sin embargo, para la prueba hemos ajustado el nivel crítico para que sea más bajo que el medido por el sensor DHT22 y asegurar que la placa active la segunda etapa). (nota: para simular el modo verano hemos empleado agua fría en el vaso, menos de 20ºC y para simular el modo invierno agua del tiempo, mayor a 20ºC).

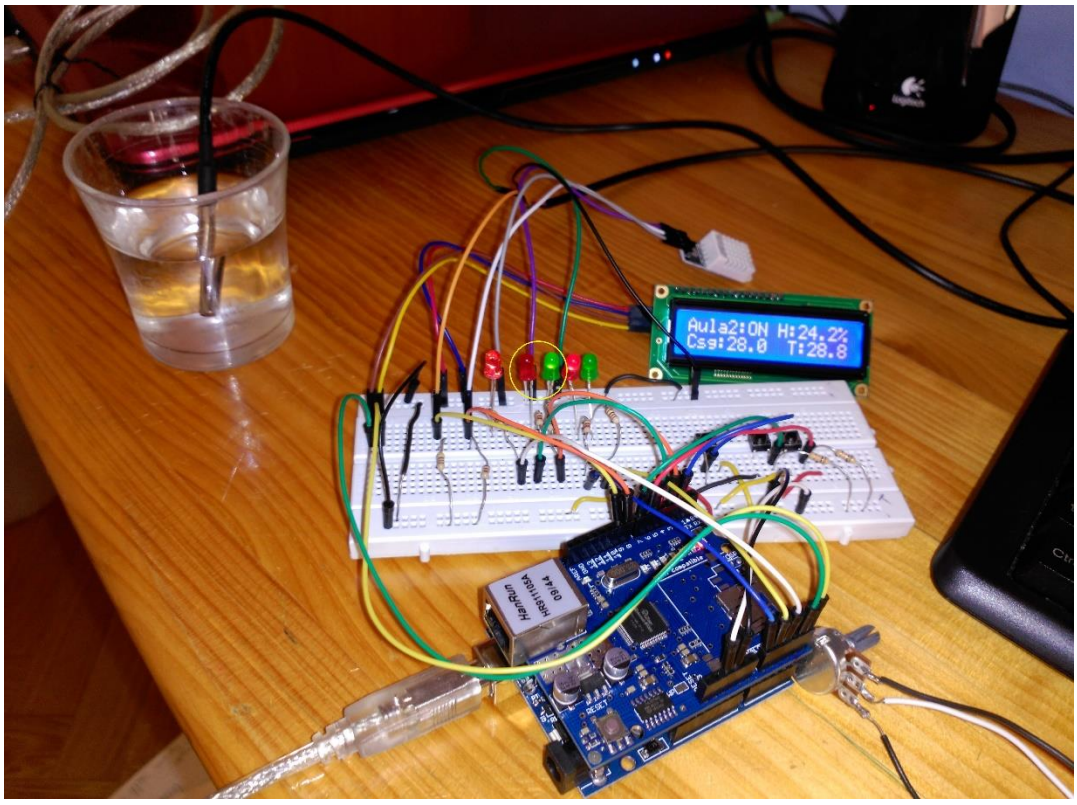


Figura 9.11 *Resultado obtenido en prueba caso uno de protoboard bloque 3.*

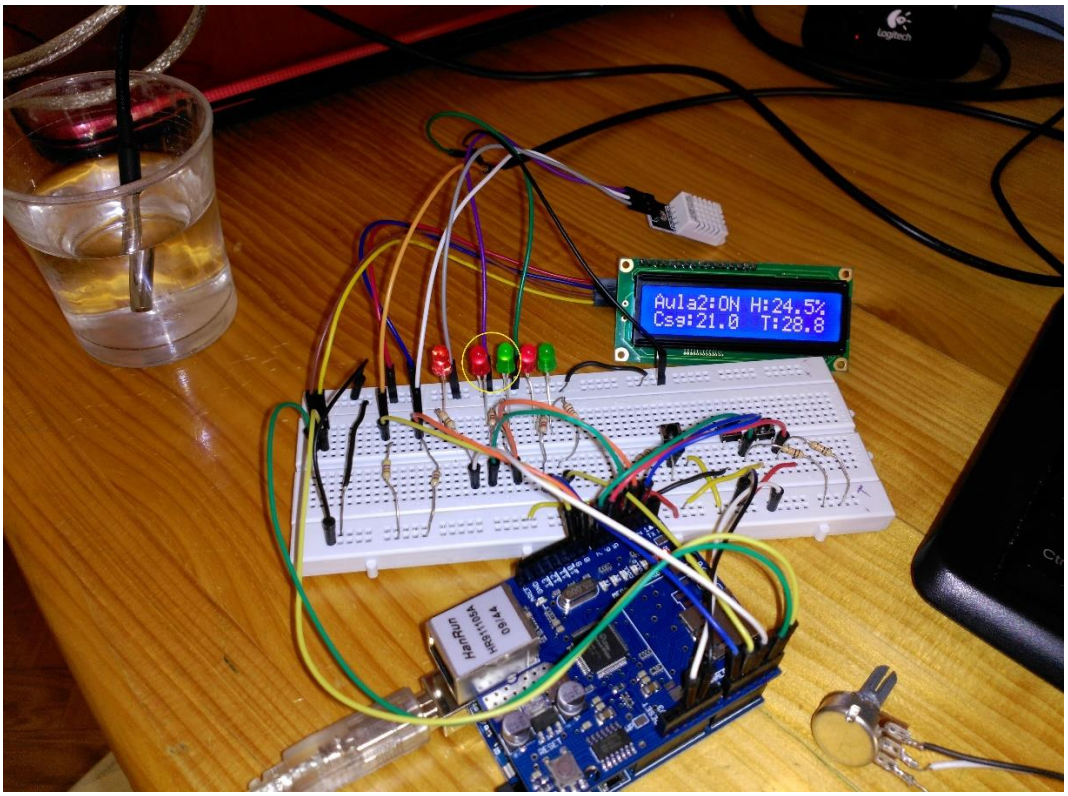


Figura 9.12 *Resultado obtenido en prueba caso dos de protoboard bloque 3.*

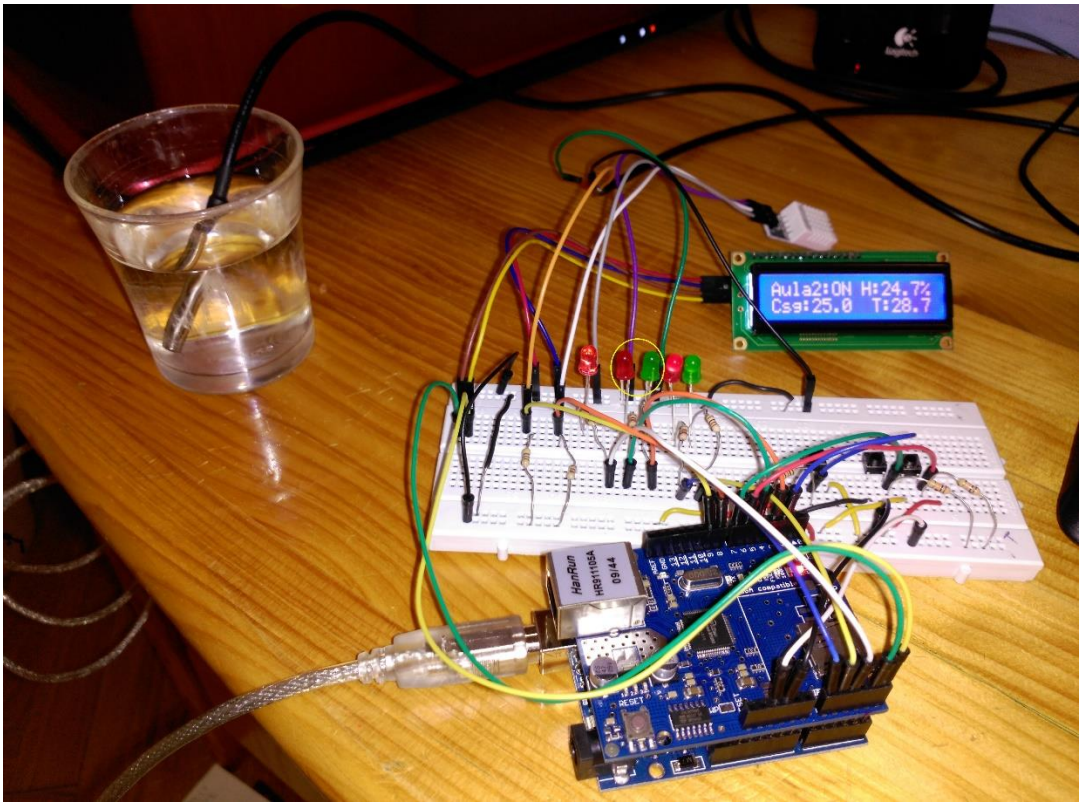


Figura 9.13 *Resultado obtenido en prueba caso tres de protoboard bloque 3.*

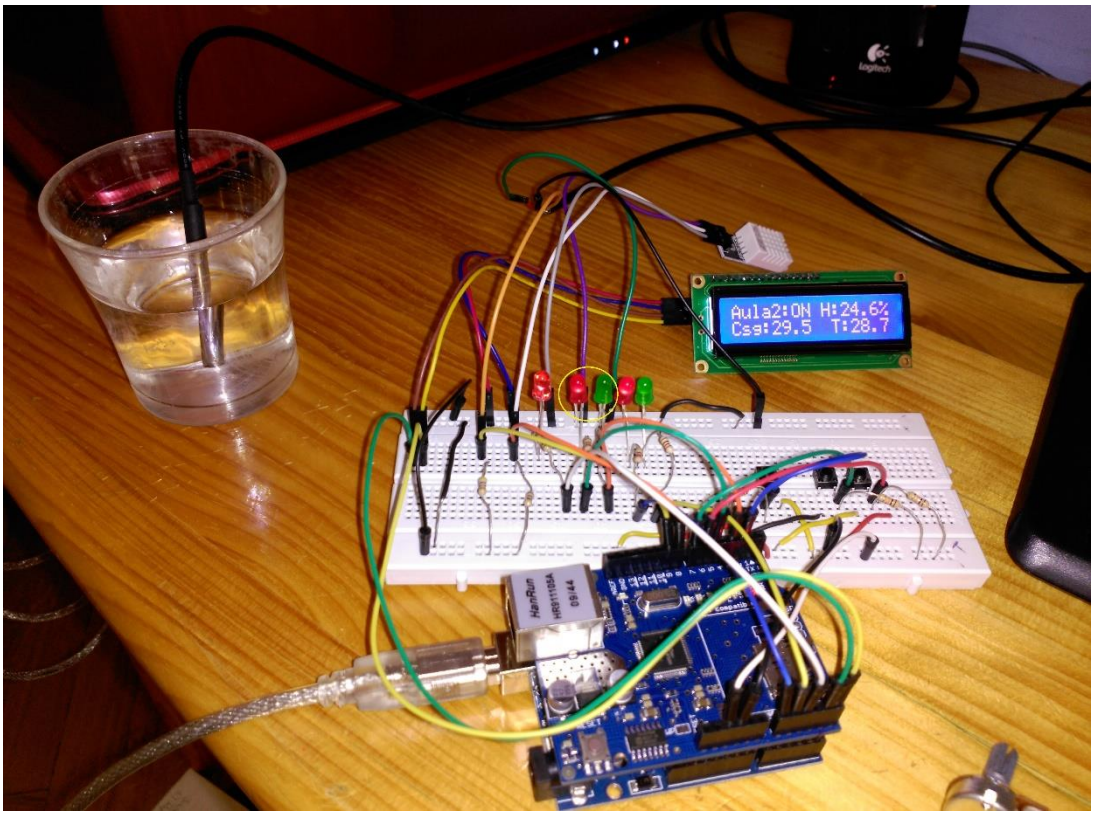


Figura 9.14 *Resultado obtenido en prueba caso cuatro de protoboard bloque 3.*

9.2 Pruebas aplicación web Java.

En este apartado hemos realizado una serie de pruebas sobre las vistas de la aplicación web que ya vimos en el capítulo ocho. En estas pruebas, hemos confirmado de forma satisfactoria, que la transición entre vistas se realice correctamente y que cada vista muestre lo que debe mostrar en función del diseño que hicimos para la interfaz web central en el capítulo seis. Para realizar las pruebas hemos encendido el servidor web local en la Raspberry Pi, y desde el navegador web del portátil Toshiba, en la misma red local Ethernet que la Raspberry Pi, hemos hecho una serie de comprobaciones.

Por otro lado, en la vista de nodo remoto hemos confirmado que recupera los últimos datos disponibles en la tabla del bloque al que pertenece dicho nodo remoto para mostrar. Además, hemos comprobado que aparezcan los mensajes de error y confirmación correctos en función de la orden que introducimos. Si introducimos una orden de cambio de consigna cuyo valor no puede ser asignado al bloque, aparece un mensaje de error y la orden no se procesa. En caso de enviar una orden de manera correcta, se procesa y el Servlet la envía al nodo remoto correspondiente.

Por último, hemos comprobado que sólo tengamos acceso a la interfaz web central con el usuario autenticado y que en caso de no estar autenticado que no muestre ninguna vista.

9.3 Pruebas de la comunicación

En este apartado, hemos comprobado que la comunicación se realice de manera adecuada entre la placa Arduino Uno de un nodo remoto y la Raspberry Pi (nodo central). Además, esto nos permite determinar el comportamiento general del sistema domótico.

En primer lugar, hemos probado a enviar un mensaje de estado desde un nodo remoto del bloque uno, empleando la protoboard de la figura 9.1. El resultado ha sido satisfactorio. Para enviar el mensaje de estado, hemos empleado el software desarrollado que explicamos durante el capítulo ocho, en concreto, en esta prueba participarán las librerías ArduinoJson, para crear el mensaje de estado según el formato definido en el capítulo seis, y la librería PubSubClient, para permitir que nuestra placa Arduino sea un cliente MQTT. En la figura 9.16, podemos visualizar el mensaje de estado recibido por el cliente MQTT del bloque uno en la Raspberry Pi. Además, en la figura 9.17, podemos ver la última entrada de dicha tabla, que se corresponde con el mensaje de estado enviado por la placa Arduino.

En segundo lugar, hemos probado a enviar desde la interfaz web remota las dos órdenes remotas posibles al nodo remoto de la protoboard de la figura 9.1. El resultado también ha sido satisfactorio, el mensaje recibido por el Servlet ha sido procesado correctamente, como podemos ver en la figura 9.15. El resultado de la primera prueba de cambio de consigna hemos obtenido un resultado como el de la figura 9.3, pero en la segunda línea aparecía “Csg:26”. El resultado de la primera prueba ha sido como el de la figura 9.3, con todos los elementos del bloque conectados y la orden correspondiente enviada al servomotor.

Por último, en la figura 9.18 podemos ver los clientes MQTT participes de la comunicación de nuestro sistema domótico. Con estas últimas pruebas hemos comprobado que el funcionamiento del sistema satisface lo esperado y deseado.

```
payload[length]='\0';
char* orden = (char*)payload;
StaticJsonBuffer<20> buffer_Received;
JsonObject& received = buffer_Received.parseObject(orden);
int arduino = received["arduino"];
if(id_nodo == arduino){
  Serial.print("Orden recibida en Arduino: ");
  Serial.println(id_nodo);
  if(received["orden"] ==0){
    Serial.println("Orden de cambio de consigna recibida");
    temp_consigna = received["valor"];
    Serial.print("Nueva consigna: ");
    Serial.println(temp_consigna);
  }
  else{
    Serial.println("Orden de encendido/apagado recibida");
    MP_remoto = received["estado"];
    Serial.print("Nuevo estado: ");
    if(MP_remoto == 1){
      Serial.println("Encendido");
    }else{
      Serial.println("Apagado");
    }
  }
}
```

Resultado orden cambio de consigna

```
Orden recibida en Arduino: 1
Orden de cambio de consigna recibida
Nueva consigna: 26
```

Resultado de orden encendido/apagado

```
Orden recibida en Arduino: 1
Orden de encendido/apagado recibida
Nuevo estado: Encendido
```

Figura 9.15 *Código y resultado de recepción de mensajes en protoboard.*

```
^Cpi@raspberrypi ~/MQTT $ java -cp lib/org.eclipse.paho.client.mqttv3-1.0.2.jar:lib/java-json.jar:lib/mysql-connector-java-5.1.35-bin:. cliente_Bloque1
Topic: Bloque_1, Message: {"climatizador":"on","consigna":26,"estado":"on","arduino":1,"bomba":"on","recuperador":"on","sonda_circuito":"on","temperatura
":27.2}
```

Figura 9.16 *Mensaje recibido en cliente MQTT bloque 1 de nodo central.*

```
mysql> SELECT * FROM Bloque_1 WHERE arduino = 1 ORDER BY fecha DESC LIMIT 1
    -> ;
+---------+--------+--------------+-------------+-------+----------------+-------------+----------+---------------------+
| arduino | estado | climatizador | recuperador | bomba | sonda_circuito | temperatura | consigna | fecha               |
+---------+--------+--------------+-------------+-------+----------------+-------------+----------+---------------------+
|       1 | on     | on           | on          | on    | on             |       27.20 |    26.00 | 2015-06-21 12:44:54 |
+---------+--------+--------------+-------------+-------+----------------+-------------+----------+---------------------+
1 row in set (0.01 sec)
```

Figura 9.17 *Última inserción de mensaje de estado realizada en la tabla del bloque uno.*

```
pi@raspberrypi ~ $ mosquitto
1434884459: mosquitto version 1.3.5 (build date 2014-10-18 21:06:44+0100) starting
1434884459: Using default config.
1434884459: Opening ipv4 listen socket on port 1883.
1434884459: Opening ipv6 listen socket on port 1883.
1434884459: Warning: Address family not supported by protocol
1434884464: New connection from 192.168.1.45 on port 1883.
1434884464: New client connected from 192.168.1.45 as cliente_bloque_1 (c1, k60).
1434884511: Socket error on client cliente_bloque_1, disconnecting.
1434884520: New connection from 192.168.1.45 on port 1883.
1434884520: New client connected from 192.168.1.45 as cliente_bloque_1 (c1, k60).
1434884585: New connection from 192.168.1.39 on port 1883.
1434884585: New client connected from 192.168.1.39 as cliente_arduino_1 (c1, k30).
1434884631: New connection from 192.168.1.45 on port 1883.
1434884631: New client connected from 192.168.1.45 as cliente_interfaz_web (c1, k30).
```

Figura 9.18 *Terminal bróker local Mosquitto*

Capítulo 10: Presupuesto

En este capítulo vamos a ofrecer una estimación de los costes que conllevarían la realización de este trabajo fin de grado. Para ello, vamos a organizar los costes en diferentes apartados para facilitar su comprensión. En primer lugar, veremos los costes de personal.

10.1 Costes de personal

En este apartado, veremos los costes derivados del personal involucrado en la realización del proyecto, en función del precio por hora y de las horas realizadas. En la tabla 10.1, podemos visualizar dichos costes.

Código	Descripción	Unidad	Cantidad	Precio	Precio Total
1.1	**Ingeniero Telemático**, con conocimientos básicos de programación de microcontroladores, web y Java. Carlos Gómez Garrido	Mes / media jornada	280	30€	8400€
1.2	**Ingeniero Industrial Mecánico**. Tutor, con conocimientos amplios de sistemas de control y de climatización. Higinio Rubio Alonso	Consultas/ 1 hora	20	60€	1200€
				COSTE TOTAL:	9600€

Tabla 10.1 *Costes de personal para la realización del proyecto.*

10.2 Costes de hardware y software

En este apartado, veremos los costes a partir del material requerido para llevar a cabo el sistema de control propuesto a partir de los componentes hardware requeridos y el software necesario para su ejecución y funcionamiento. En la tabla 10.2, podemos visualizar los gastos del software que hemos empleado, como podemos comprobar, la mayoría no supone ningún coste adicional, ya que como hemos ido viendo en los capítulos anteriores, se trata de software libre.

Código	Descripción	Precio	Amortización	Precio Total
2.1	**Microsoft Office 2013**. Microsoft Word 2013 empleado para el desarrollo de la memoria del trabajo fin de grado.	70€	1/4	17.50€
			COSTE TOTAL:	17.50€

Tabla 10.2 *Costes de licencias Software para la realización del proyecto.*

Por último, en la tabla 10.3, podremos ver el coste total de todos los nodos remotos por bloque más el coste del nodo central, para obtener el coste total de los componentes hardware.

Código	Descripción	Precio	Cantidad	Precio Total
3.1	**Arduino Uno**. Placa de la plataforma Arduino empleada como unidad de control para el bloque con climatizador: polideportivo y comedor, distribución suelo radiante, aulas y piscinas.	20€	18	360€
3.2	**Ethernet W5100 Shield**. Shield Ethernet para ofrecer conexión a Internet a la placa Arduino.	13.50€	18	243€
3.3	**Sensor DS18B20**. Sensor de temperatura digital.	4€	24	96€
3.4	**Sensor DHT22**. Sensor de digital de temperatura y humedad.	6€	12	72€
3.5	**Módulo de 4 relés MSP430**. Actuador para varios elementos del bloque de climatización.	6€	18	108€
3.6	**TowerPro SG-5010**. Servomotor digital estándar empleado como actuador.	10€	6	60€
3.7	**Potenciómetro.** Potenciómetro para ajustar la temperatura de consigna desde interfaz auxiliar.	1€	18	18€
3.8	**Pulsador.** Pulsador para apagar o encender la pantalla LCD de la interfaz auxiliar.	0.20€	18	3.60€
3.9	**Pantalla LCD 16x2**. Pantalla LCD para mostrar la información al usuario en la interfaz auxiliar.	7€	18	126€
3.10	**Módulo I2C.** Módulo I2C para controlar la pantalla LCD 16x2 con la placa Arduino Uno.	3€	18	54€
3.7	**Adaptador de corriente.** Adaptador de corriente AC-DC de 9V y 1000mA para alimentar la placa Arduino Uno.	5€	18	90€
3.8	**Raspberry Pi modelo B+.** Modelo de Raspberry Pi empleado como nodo central para nuestro proyecto.	30€	1	30€
3.9	**Cargador micro-USB.** Cargador micro-USB de 5V y 2000mA para alimentar la Raspberry Pi.	10€	1	10€
3.10	**Tarjeta micro-SD de 16Gb**. Tarjeta micro-SD para proporcionar almacenamiento a la Raspberry Pi.	6€	1	6€
			COSTE TOTAL:	1276.60€

Tabla 10.3 *Costes de componentes hardware de nodos remotos para la realización del proyecto.*

10.3 Costes Indirectos

En este apartado, vamos a describir los costes indirectos de la realización del proyecto, que incluyen aquellos gastos de las herramientas y equipos empleados. En la tabla 10.4, podemos visualizar dichos costes.

Código	Descripción	Precio	Amortización	Precio Total
4.1	**Ordenador Toshiba Satellite.** Ordenador portátil empleado realizar diversas tareas del proyecto.	350€	1/5	70€
4.2	**Monitor ASUS 19'' HDMI.** Monitor empleado para configurar y trabajar con la Raspberry Pi.	80€	1/6	13.35€
4.3	**Teclado GENIUS USB.** Teclado empleado para configurar y trabajar con la Raspberry Pi.	7€	1/6	1.20€
4.4	**Ratón GENIUS USB.** Ratón empleado para configurar y trabajar con la Raspberry Pi.	4€	1/6	0.7€
4.5	**Gastos de luz.**			25€
4.6	**Gastos de conexión a Internet local.**			35€
			COSTE TOTAL:	145.25€

Tabla 10.4 *Costes indirectos relacionados con la realización del proyecto.*

10.4 Coste total

Por último, en la tabla 9.5 podemos ver los costes totales de la realización del proyecto. El total de los costes totales ascienden a la cantidad de **trece mil trescientos cincuenta y siete euros con treinta y siete céntimos**.

	Coste	Precio
1	Costes de personal	9600€
2	Costes software	17.50€
3	Costes hardware	1276.40€
4	Costes indirectos	145.25€
	Total	11039.15€
	IVA (21%)	2318.22€
	COSTE TOTAL	**13357.37€**

Tabla 10.5 *Costes totales para la realización del proyecto.*

Capítulo 11: Conclusions and Future Work

In this chapter, we are going to expose some conclusions that we have found after finishing our project. Secondly, we are going to describe some future improvements that can be implemented in order to expand our designed automation system.

11.1 Conclusions

At this point, we can say that we have satisfied our main objective, which was to design and develop a low-cost automation system, based on the real characteristics of a heating and air cooling system, already in place in a building. An automation system, whose components belong to what it's known as open hardware and software, communicating over a local Ethernet network. A design and implementation that could be integrated in a building in the future.

In order to complete our main task, first we have analysed the characteristics of the real heating and air cooling system to control in a building complex. After this analysis we were able to develop and idea for controlling that system. This idea provided us the control strategies which our automation system should include, which also allowed us to determine the requirements of our design. Secondly, we have analysed some software development models, for determining the methodology to develop our project and planificate its phases according to it. Thirdly, we were able to design the arquitecture for our automation system, following its requirements. We were able to include MQTT as the communication protocol in our architecture. Moreover, we chose an Arduino board and a Raspberry Pi together with some sensors and actuators as the main hardware components of our automation system, which allowed us to reduce its total cost. Finally, we developed the automation software according to the design we made in those hardware componentes. In addition, after some tests we were able to determine that our complete implementation was a success.

The Arduino platform has demonstrated to be a very useful platform in order to develop automation projects. It has many extension libraries, already tested and ready to use, which can simplify a lot the difficulties when implementing a software with many requirements, if you know what to use. However, Arduino boards have a limited memory amount, so it will be a good idea to use the board with the highest memory capacity for achieving higher results. In any case, we were able not just to learn about the Arduino platform but more open electronics prototyping platform, which can be used as control unit in an automation system. In addition, the existing Arduino libraries for extending its software allowed us to interact with different temperature sensors and actuators in our control system.

Low-cost computers have been a great advantage for achieving a low-cost and low energy consuming automation system. It is true that their capacity is limited, if we compare it with our regular computer. However, if we know what are the uses for what a low-cost computer is needed, and its capacity is enough to do them, we should take advantage of them. In our case, it was enough to coordinate our centralised automated system and perform what it was designed to. The Raspberry Pi satisfied one of our secondary goals for achieving a

centralised architecture, where we could host a local web application and store the data from each component of our automation system.

On the other hand, the Raspberry Pi foundation started selling in February, 2015 a new model of the Raspberry Pi, which has a double capacity than the one we are using, which was out around half 2014. So, if in less than a year they have been capable of doubling the capacity of their devices, we will probably see really good hardware characteristics in low-cost computers in the recent future.

The communication protocol MQTT that we have used has allowed us to reach not only the flexible arquitecture, according to the characteristics of the system to control, but to get a reliable and low computing communication. Its publish/subscribe model is really suitable for automation systems, where the devices that are being used have a limited capacity. It is a protocol that offers great future opportunities to automation systems.

Nowadays, almost every building has at least one computer and a local Ethernet network in order to access to the Internet. In this situation, it is essential to being able to see what is happening in our automation system using both technologies and we were able to do it using a local web application and offering our Arduino boards Ethernet connection. Moreover, giving the user the chance to interact with the control elements of the automation system, from their web browser offers a great experience. Then, we were able to satisfy this secondary goal by developing a Java web application as central user interface.

Thanks to this project, we have been able to design and implement a low-cost automation system to control a heating and air-cooling system, according to its characteristics, in a building complex higher than a home. An automation system whose architecture is in accordance with the real characteristics of a regular system existing in a building. It can serve as a guideline to anyone, with programming and electronics basic knowledge, interested in control a regular system not just in their homes, but in any building, using low-cost and low energy consuming components. We are not only proposing the methodology to develop an automation system. Our guideline starts from the analysis of the system to control and continues until the design of its architecture and hardware components. Finishing with its software implementation and testing, to get a complete example about how to develop an automation system project.

Finally, after taking a closer look to the budget for developing this project, we have discovered that the total cost of developing it is close to thirteen thousand euros. If we want to integrate it in a building complex with the characteristics of the one we have presented here, the cost will raise for installing what is necessary to integrate our automation system. However, in a building complex whose cost can be around several millions of euros, integrating an automation system with a cost between ten and twenty thousand euros is affordable. As we have seen, this recent open hardware and software technologies will give automation a bright future.

11.2 Future Work

One of the advantages of developing a project inside home automation is that it results will last for a long time. In the future, according to the results after integrating our automation system, we can add more functionalities or modify the ones we already have. Meanwhile,

there are many functionalities that we can think about and include into our automation system for improving the user´s comfort. In this section, we will see some improvements that can be added to our automation system in a future development.

11.2.1 Increase web application scope

One of the few things which can be done to our system, is to offer the user the opportunity to expand the scope of the web application. For example, to visualize more information about the behaviour of the heating and air cooling system. What we can do is adding in the web application a few charts that describe its behaviour. For example, in every remote node view, where we can see all the information related to that node in a specific time and interact with that node, we can add a chart that shows the different values of the temperature from the place, where the remote node is located, whose heating system is being controlled by the node. In this way, by showing the range of temperatures over that place in a limited period of time selected by us, for example, an hour, a day or a month, we can describe the behaviour of the place controlled by any remote node of our automation system.

On the other hand, we can include another section in every remote view in our web application where the user can configure automatised actions. For example, to order an specific node to turn on or turn off its heating and air cooling system, or maybe to select a desired temperature in a block at a specific time of the day. In order to do this, we will probably need to add a new Servlet to our web application to handle those new orders POST requests, but, thanks to MQTT we can define a new topic and a new message format with JSON where any remote node will listen and understand to those new orders.

11.2.2 Integrate with more devices and systems

One of the things that we have discovered after finishing our project is Arduino is very useful for automation systems, since it has many expansion libraries to complement our software. However, Arduino boards have some limitations because of their limited capacity and low memory space. Although Arduino Uno boards could do the work for the automation system we have designed, it won't support more functionalities, since we have found that with the board we are using, its capacity and memory is almost full. As a future work, we think that will be very convenient to try the same design and implementation but in a more powered Arduino board, for example, an Arduino Mega board.

On the one hand, using an Arduino Mega would increase but just a little bit our automation system cost. However, this extra cost compared with the personal cost as we saw in chapter ten would be very small.

On the other hand, using an Arduino Mega board would support also the functionalities required by our designed automation system. In addition, we can add any other functionality that we can think to increase the user's comfort when using our automation system to control their heating and cooling system.

Finally, using an Arduino Mega board can also integrate more systems that are already installed in our building in our automation system. For example, the lighting or smoke detection system. In this way, we can expand our design not only for the heating and air

cooling system of our building complex. We can adapt the architecture already designed to the requirements of those systems in order to get a more complete automation system, which can integrate many systems already installed in our building complex, to achive a greater level of comfort, security and lower energy consumption to the user.

11.2.3 Mobile application

Nowadays, the use of Smart Phones has increased a lot, almost everyone has one. So in order to make the automation system more accesible to the user, it would be a great advantage to develop a mobile application in the future. For example, we could develop an Android or IOS application, since both of them are the operating systems more used for mobile phones recently.

There are two things that will help the development of the mobile application. The first one is that there is already a web server working in our automation system for the web application. This means that a mobile application can take advantage of it and retrieve the same information about every remote node in the database as the web application.

The second one is that a mobile phone can also use the local network to communicate with any remote node, and for example, there is a MQTT Paho client already developed for Android, which we can use in order to communicate the application with the other nodes.

11.2.4 Integrate in our building complex

The next future task to perform is integrating our automation system in the building complex described in chapter three. As we saw previously, it cost will increase once we integrate it since the complex need to include what is necessary for our automation system to start working, like Ethernet connection or electric power. In addition, it will also be needed to adjust the devices of our automation system following the regulation where they apply. Every device must be homologated before installed in our building complex.

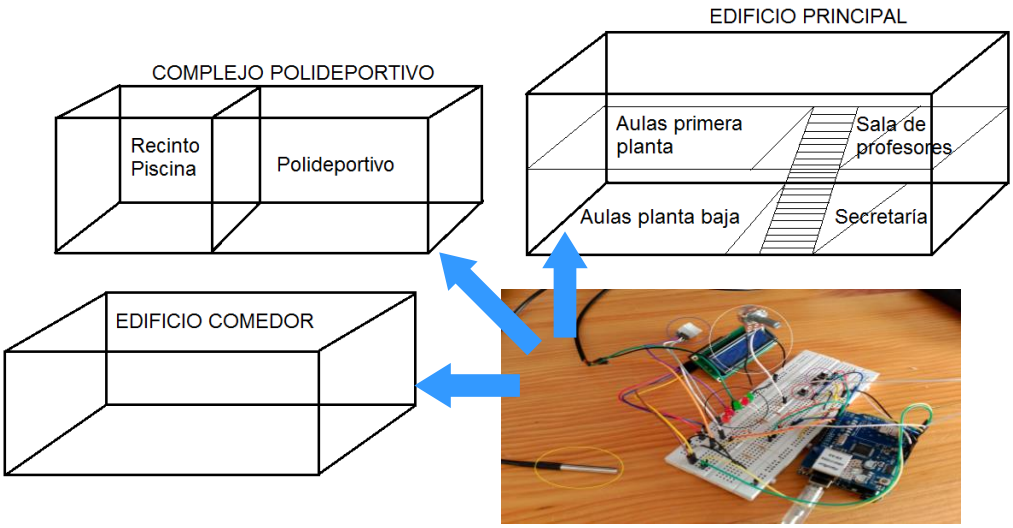


Figura 11.1 *Integración de sistema de control diseñado en complejo.*

Capítulo 12: Bibliografía

12.1 Bibliografía

[1] Torrente, Oscar. *Arduino, curso práctico de formación*. 1ª ed, 2013.

[2] Doukas, Charalampos. *Building Internet of Things with the Arduino*. 1ª ed, 2012.

[3] S. Pressman, Roger. *Ingeniería del software: un enfoque práctico*. 7ª ed, 2010.

[4] Murach, Joel; Steelman, Andrea. *Murach's Java Servlets and JSP*. 2ª ed, 2008.

[5] Arias Fisteus, Jesús. *Apuntes de la asignatura: "Computación Web"*. Grado en Ingeniería Telemática. Universidad Carlos III de Madrid. 2015.

12.2 Referencias Web

[6] Página de la tienda oficial del fabricante de ARDUINO:
http://store.arduino.cc/category/11
(Último acceso: 06-Junio-2015)

[7] Página del hardware oficial del fabricante de WIRING:
http://wiring.org.co/hardware/
(Último acceso: 06-Junio-2015)

[8] Página del hardware oficial del fabricante de NETDUINO:
http://www.netduino.com/hardware/
(Último acceso: 06-Junio-2015)

[9] Página de la placa oficial netduino 2:
http://www.netduino.com/netduino2/
(Último acceso: 06-Junio-2015)

[10] Hoja de datos del sensor digital de temperatura DS18B20:
http://www.adafruit.com/datasheets/DS18B20.pdf
(Último acceso: 08-Junio-2015)

[11] Hoja de datos del sensor analógico de temperatura TMP36:
http://www.analog.com/media/en/technical-documentation/datasheets/TMP35_36_37.pdf
(Último acceso: 08-Junio-2015)

[12] Hoja de datos del sensor termistor NTC de temperatura:
http://cdn.sparkfun.com/datasheets/Sensors/Temp/ntcle100.pdf
(Último acceso: 08-Junio-2015)

[13] Hoja de datos del sensor digital DHT22 de temperatura y humedad:
https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf
(Último acceso: 08-Junio-2015)

[14] Hoja de datos del sensor digital DHT11 de temperatura y humedad:
http://www.micropik.com/PDF/dht11.pdf
(Último acceso: 08-Junio-2015)

[15] Especificaciones del protocolo 1-Wire:
http://www.maximintegrated.com/en/app-notes/index.mvp/id/1796
(Último acceso: 08-Junio-2015)

[16] Página oficial de la plataforma RASPBERRY PI:
https://www.raspberrypi.org/help/what-is-a-raspberry-pi/
(Último acceso: 11-Junio-2015)

[17] Página oficial del modelo RASPBERRY PI A+:
https://www.raspberrypi.org/products/model-a-plus/
(Último acceso: 11-Junio-2015)

[18] Página oficial del modelo RASPBERRY PI B+:
https://www.raspberrypi.org/products/model-b-plus/
(Último acceso: 11-Junio-2015)

[19] Página oficial del modelo RASPBERRY PI 2:
https://www.raspberrypi.org/products/raspberry-pi-2-model-b/
(Último acceso: 11-Junio-2015)

[20] Figura sensor analógico TMP36:
https://learn.adafruit.com/tmp36-temperature-sensor/overview
(Último acceso: 08-Junio-2015)

[21] Figura sensores de temperatura y humedad DHT11 y DHT22:
http://sysmagazine.com/posts/167459/
(Último acceso: 08-Junio-2015)

[22] Página tienda de distribuidor oficial de productos RASPBERRY PI:
http://uk.rs-online.com/
(Último acceso: 11-Junio-2015)

[23] Página oficial fundación BEAGLEBOARD:
http://beagleboard.org/about
(Último acceso: 11-Junio-2015)

[24] Página oficial del producto BEAGLEBONE BLACK:
http://beagleboard.org/black
(Último acceso: 11-Junio-2015)

[25] Página oficial de productos PCDUINO:
http://www.linksprite.com/?page_id=782
(Último acceso: 11-Junio-2015)

[26] Página de la tienda oficial de PCDUINO LITE:
https://www.sparkfun.com/products/12077
(Último acceso: 11-Junio-2015)

[27] Página de la tienda oficial de PCDUINO 2:
https://www.sparkfun.com/products/12749
(Último acceso: 11-Junio-2015)

[28] Página de la tienda oficial de PCDUINO 3:
https://www.sparkfun.com/products/12856
(Último acceso: 11-Junio-2015)

[29] Página fuente de imagen Arduino Uno detallado:
http://www.zenbike.co.uk/arduino/uno_components/index.html
(Último acceso: 07-Junio-2015)

[30] Página librería LiquidCrystal para controlar LCD desde ARDUINO:
http://www.arduino.cc/en/Tutorial/LiquidCrystal
(Último acceso: 07-Junio-2015)

[31] Página recopilación de publicaciones esenciales del protocolo MQTT por la empresa HIVEMQ:
http://www.hivemq.com/mqtt-essentials-wrap-up/
(Último acceso: 05-Junio-2015)

[32] Página oficial de protocolo MQTT:
http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html
(Último acceso: 05-Junio-2015)

[33] Página oficial de mensajes JSON:
http://json.org/
(Último acceso: 05-Junio-2015)

[34] Página oficial de MySQL:
http://www.mysql.com
(Último acceso: 05-Junio-2015)

[35] Página oficial de plataforma ARDUINO:
http://www.arduino.cc
(Último acceso: 05-Junio-2015)

[36] Página oficial de servidor web APACHE TOMCAT:
http://tomcat.apache.org/
(Último acceso: 05-Junio-2015)

[37] Página oficial de bróker MOSQUITTO:
http://mosquitto.org/
(Último acceso: 05-Junio-2015)

[38] Página de la librería JSON para Arduino:
https://github.com/bblanchon/ArduinoJson/wiki
(Último acceso: 12-Junio-2015)

[39] Página de la librería Ethernet para Arduino:
http://www.arduino.cc/en/Reference/Ethernet
(Último acceso: 12-Junio-2015)

[40] Página de la librería para comunicación SPI para Arduino:
http://www.arduino.cc/en/Reference/SPI
(Último acceso: 12-Junio-2015)

[41] Página de la librería MQTT para Arduino:
http://knolleary.net/arduino-client-for-mqtt/api/
(Último acceso: 12-Junio-2015)

[42] Página de la librería DallasTemperature para sensor DS18B20 y Arduino :
https://milesburton.com/Dallas_Temperature_Control_Library
(Último acceso: 12-Junio-2015)

[43] Página de la librería 1-Wire para sensor DS18B20 y Arduino :
http://www.pjrc.com/teensy/td_libs_OneWire.html
(Último acceso: 12-Junio-2015)

[44] Página de la librería LiquidCrystal para controlar pantalla LCD con Arduino e I2C:
https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home
(Último acceso: 12-Junio-2015)

[45] Página de la librería Wire para comunicación I2C y Arduino :
http://www.arduino.cc/en/Reference/Wire
(Último acceso: 12-Junio-2015)

[46] Página de la librería PID para Arduino :
http://playground.arduino.cc/Code/PIDLibrary
(Último acceso: 12-Junio-2015)

[47] Página de la librería DHT para el sensor DHT22 y Arduino :
https://github.com/adafruit/DHT-sensor-library
(Último acceso: 12-Junio-2015)

[48] Página de la librería Servo para controlar servomotores con Arduino :
http://www.arduino.cc/en/Reference/Servo
(Último acceso: 12-Junio-2015)

[49] Página oficial de la placa Arduino Uno:
http://www.arduino.cc/en/Main/ArduinoBoardUno
(Último acceso: 08-Junio-2015)

[50] Página oficial del shield Ethernet para la placa Arduino Uno:
http://www.arduino.cc/en/Main/ArduinoEthernetShield
(Último acceso: 08-Junio-2015)

[51] Página fuente de la imagen de conexionado de módulo 4 relés y placa Arduino:
http://www.sainsmart.com/4-channel-5v-relay-module-for-pic-arm-avr-dsp-arduino-msp430-ttl-logic.html
(Último acceso: 09-Junio-2015)

[52] Página controladores ON/OFF y PID:
http://www.tecnoficio.com/electricidad/instrumentacion_industrial4.php
(Último acceso: 01-Junio-2015)

[53] Página documentación oficial JSON Java:
http://www.json.org/java/index.html
(Último acceso: 14-Junio-2015)

[54] Página documentación oficial cliente MQTT Paho Java:
http://www.eclipse.org/paho/files/javadoc/index.html
(Último acceso: 14-Junio-2015)

[55] Página control de temperatura:
http://www.jmi.com.mx/documento_literatura/Dispositivos-control-temperatura.pdf
(Último acceso: 01-Junio-2015)

[56] Página sintonización PID:
https://sites.google.com/site/picuino/ziegler-nichols#TOC-Referencias
(Último acceso: 01-Junio-2015)

[57] Página fuente de imagen con los modos de conexión del sensor DS18B20:
http://ciberdroide.com/AcuBioMed/cao-9-sensor-digital-de-temperatura-ds18x20/
(Último acceso: 07-Junio-2015)

Anexos a la memoria:

Anexo 1: English Summary

Chapter 1: Introduction.

As a motivation to our project we have found that the appearance of some open hardware and software technologies, like the Arduino platform or low-cost computers (Raspberry Pi), have been able to reduce automation systems cost. We would like to extend automation systems scope to any kind of building where the total cost to implement that system can be reduced. Moreover, another motivation for this project is to create a guide for other people that want to include an automation system for increasing their comfort, in any type of building.

The main goal of this project is to design and develop a low-cost automation system, based on the real characteristics of a heating and air cooling system, already in place in a building. An automation system based whose components, belong to what it's known as open hardware and software, communicating over a local Ethernet network. As well as the main goal, we have some secondary goals to achieve during the completion of our project. In chapter one we can find a more detailed explanation about this secondary goals.

Finally, in this chapter we can see a detailed explanation of the structure of this document, where we are specifying what we are going to see in each chapter.

Chapter 2: State of technology.

In this chapter we are going to explore the context where our project is located. We are going to explain almost everything required to the reader for understanding the following chapters of the project and the context where our project is located.

On the one hand, we will talk about automation systems and the architecture that they usually have. We will describe what is a centralised architecture, descentralised architecture, distributed and mixed architectures in automation systems. Secondly, we will describe what a control system is, its main components and the two most common control systems used in automation. Specifically, we will talk about on and off control systems and proportional, integral and derivative control (PID).

On the other hand, we will describe some of the hardware components that have appeared recently and can be used in automation system. At first, we will describe what is an open electronics plaftorm, based in a microcontroller, and three platforms that are being used in automation systems, like Arduino, Wiring and Netduino. Secondly, we will explain some basic concepts of sensors in order to introduce some of the temperature sensors, available to use with an open electronics platform, as well as some of the temperature and humidity sensors which we can use. Finally, we will talk about low-cost computers, which can be used as the central element in an automation system and we will introduce as examples the Raspberry Pi, BeagleBone Black and pcDuino.

Chapter 3: Building complex description.

In this chapter we will see how are the buildings from the building complex where our automation system is going to be located. In this case, we have three buildings, one where the classes are located, called the main building. A second one, where the sports centre and the swimming pools are located. And the last one, where the dining hall is placed. Those are the buildings where we have different heating and air cooling systems, each one will be a different block.

Firstly, we will describe the heating and air cooling block one, which will be used in the sports centre and the dining hall. This block one is composed by a climate controller that receives water from a circuit and uses it to heat or chill the air inside the sports centre and the dining hall, we can find a more detailed explanation in chapter three.

Secondly, we will describe the heating and air cooling block two, which will be used to distribute water at the right temperature to the floor heating systems located in the two floors at the main building. This block two is composed by a heat and cold water generation pump that sends its water to a water circulation pump that boosts the water on to the heating and cooling floor circuits in each floor, we can find a more detailed explanation in chapter three.

Thirdly, we will describe the heating and air cooling block three, which will be used to climate the classes in the main building using a two stages thermostat. This block three is composed by an auxiliar heating controller (stage two) and the previous block heating and cooling floor circuits (stage one) to climate each class, we can find a more detailed explanation in chapter three.

Forthly, we will describe the heating and air cooling block four, which will be used to climate the swimming pools in the sports centre. This block three is composed by a heat exchanger to climate each swimming pool, we can find a more detailed explanation in chapter three.

Finally, we will describe the fifth block, which will feed the hot or cold water to the other blocks in order to work as the are suppose to do. In addition, we will provide some useful figures along with the description of each block to understand it properly, but more information about it can be found in chapter three.

Chapter 4: Control strategies analysis.

During this chapter we will describe what are the control strategies which we are going to follow in order to design and implement our low-cost automation system, according to the characteristics of the heating and air cooling system located in the building complex seen in chapter three.

In order to describe the control strategies that we are going to follow, we need to understand how each heating and air cooling block works first, but thanks to chapter three we know it. However, there will be only a control strategy for blocks one, two, three and four, since as we saw in chapter three, the user does not require a control system in the fifth block. After this, we should include one of the two control systems that we saw in chapter two in every heating and air cooling block. In our designed automation system, according to the user

preferences, we are going to use a PID control in every place that uses blocks one, two and four while we will use an ON/OFF control in every place that uses block three.

Finally, we will describe the elements: control unit, sensors and actuators that will be present in each block in order to fulfil the control strategy, in order to understand how will be the control process done according to the heating and air cooling block performance, seen in chapter three. In addition, we will provide some useful figures along with the description of each control strategy per block to understand it properly, but more information about it can be found in chapter three.

Chapter 5: Methodology

During this chapter we are going to describe some details about the methodology that we are going to follow in order to develop our project. Before to chapter five, we have already seen the different technologies that involve our automation system and we also know how the heating and air cooling system works and how to control it. Then, at this point we can determine a methodology that will suit in our project and design a planning to develop our automation system. In addition, we can also understand what are the requirements for our automation system.

At first in this chapter, we are going to see what a software development is, for understanding how a project with our characteristics can be planned. Secondly, we will go through three different software development models which can define the planning for designing and developing our automation system. The three models which we are going to explain are waterfall model, incremental model and evolutive model. Then, we will define which one is more suitable for us and start the planning for our project.

As we can see in chapter five, our methodology is following a waterfall model with six phases, which are:

1. Analysis of the characteristics of the heating and air cooling system.
2. Control strategies and requirements modelling.
3. Automation system architecture design.
4. Hardware components election.
5. Software development.
6. Behaviour tests.

More information about what we are going to do in each phase can be seen in chapter five. We can also find there a detailed explanation about the time distribution of each phase and its activity thanks to a Gantt diagram and a table explaining those activites and the time they took.

Before finishing this chapter, we can also find the requirements of our automation system design, useful for the next chapters. We will see a detailed explanation of the user's requirements, control unit requirements, central element requirements and communication requirements in a table describing each of them.

Chapter 6: Design of the automation system architecture.

In this chapter we are going to describe the arquitecture which we have designed for our automation system.

The first thing that we are going to explain is the arquitecture of the automation system with respect to how its nodes are connected to each other. As we can see in detail in this chapter, we are going to have two different nodes. Several remote nodes, which are in charge of controlling the heating and air cooling system that is being used in single place in our building complex. A central node, which controls and coordinates the total set of remote nodes. In conclusion, our architecture is going to have two levels. A first centralised level, between every remote node and the central node, following a centralised arquitecture like the one we explained in chapter two. The second level will follow a distributed arquitecture, like the one we explained in chapter two, between every remote node and the heating and air cooling system which is controlling. We can find a more detailed explanation of each node in chapter six.

Secondly, we will describe the communication architecture that is being used by the remote and central nodes previously seen. In this section, we will explain the communication protocol which we are using, called MQTT, from “Message Qeueing Telemetry Transport”. As we can see it is an open source, lightweight protocol that follows a publish/subscribe model instead of the typical client-server model, making it very useful for automation systems where some nodes can have a limited capacity. More information about the protocol can be found in chapter six. After understanding the basics concepts of MQTT, we will explain the communication architecture followed between every remote node and the central node. In addition, we are going to describe the format of the messages exchanged during the communication, whose format is a JSON string. We can find more information at its section in chapter six.

Finally, we will describe how are we going to store the information about the performance of each remote node, to keep track of what is happening in our automation system. We are going to explain why we are using MySQL as a database manager and the format of each of the tables in the database to store the information. After this, we will describe the basic concepts about a Java web application, which is the one that we are using as central user interface, in order to introduce the user the architecture of the web application that we are developing. What is more, we will explain the auxiliar user interface which we have placed in every remote node in case that the central interface is not working. As we have seen in chapter six, the interaction between the user and this auxiliar interface is more limited than with the central interface, because to this auxiliar interface more people can get access but to the central interface only the person in charge of the automation system can access.

Chapter 7: Hardware components.

In this chapter we are going to explain the main hardware components which we are going to use in our automation system, we will extend the technologies seen in chapter two. In addition, we will explain the one among each type of technology which we have selected and we will give some alternatives that could have been selected among the ones presented in chapter two. We will try to explain the user what are the limitations of each for understanding

why we have chosen one among the others, according to the requirements seen in chapter five and the architecture which we have already designed.

Firstly, we are going to describe the Arduino Uno board, which we have selected to be the control unit from each remote node to control the performance of a heating and air cooling system at a place in our building complex. We are going to explain the different parts of the board and the numbers and type of each of its pins. Also, we can find the distribution of each pin in an Arduino Uno board that we have selected to develop our automation system at the end of the chapter. Later, we will explain what will be their main functions in the automation system which we have designed. Afterwards, we will talk about the Ethernet shield which we are going to use together with every Arduino Uno board. This shield is offering every board connection to the Internet. It will give us the opportunity to connect every remote node to the local Ethernet network of the building complex and communicate every remote node with the central node. More information about the Ethernet shield which we are using can be found in chapter seven. Moreover, we will present some alternatives from the ones seen in chapter two that could have seen selected instead of the Arduino Uno board.

Secondly, we will present the sensors that we are actually using in our automation system. Specifically, we will describe the digital temperature sensor DS18B20 and extend the knowledge about it already presented in chapter two. We will explain in more detail also the digital temperature and humidity DHT22 sensor which we have selected to use in remote nodes controlling places with block three heating and air cooling system. We will see the alternatives that could have been selected among the sensors introduced in chapter two. We are explaining also, according to the requirements seen in chapter five, why we have selected one sensor among the ones described in chapter two.

Thirdly, we will introduce the actuators that we are going to use to execute the control orders from each Arduino Uno board to control the performance of each heating and air cooling block. We will explain the relay module which we are going to use, describing how to use it and its main purposes. We will also explain the basics concepts of a servo motor that we are going to use also as an actuator, describing how to use it and its main purposes.

Forthly, we will describe the auxiliar user interface that we are placing along with every remote node, we will explain each of its components and it purposes. We will present the basics to understand a LCD display and how we are controlling it using the I2C protocol with our Arduino Uno board, and why are we using also a switch and a potentiometer.

Finally, we will present the low-cost computer which we have selected as the central node in our automation system. We will describe the elements which compose the Raspberry Pi model B+. In addition, we will describe the different tasks which need to be performed by the Raspberry Pi in our automation system. The Raspberry Pi will be hosting the web server for our Java web application, a MySQL local database server to store the database and its tables with the information about the performance of every remote node, a local broker, called Mosquitto, which coordinates the communication using MQTT and four MQTT clients to receive information about each block which we are controlling in our automation system, and store that information at its corresponding table in our database. We can find more information about it on its section at chapter seven.

Chapter 8: Software development.

During this chapter we are going to explain the software development that we have done to implement our automation system.

On the one hand, we are going to explain the software which we have developed in every remote node. In this case, the programming task was to develop the control software programmed in the Arduino Uno board. As we saw in chapter two, one of the advantages of the Arduino platform was that no external harware is needed to program the board's microcontroller. Instead, thanks to what is called "Bootloader", we can directly program any Arduino board from the Arduino IDE running in our computer, which we are going to explain first. Later, we will describe the extension libraries which we have used to develop the control software of each Arduino Uno board to control the heating and air cooling system block of the place where it is located. The libraries which we are going to use are:

1. **Wire, LCD and LCDI_2C libraries** to control and interact with the LCD display from the auxiliar user interface from the Arduino Uno board.
2. **1-Wire and DallasTemperature libraries** to control and interact with the digital temperature sensor DS18B20 from the Arduino Uno board.
3. **PID library** to implement a PID controller in the Arduino Uno board to control the corresponding heating and air cooling block.
4. **ArduinoJson library** to send and receive the JSON messages, whose format was designed in chapter six, between every Arduino Uno board and the central node.
5. **SPI and Ethernet libraries** to use the shield Ethernet in every Arduino Uno board to be connected to the local Ethernet network in our building complex.
6. **Servo library** to interact with a servo motor using our an Arduino Uno board.
7. **MQTT client library** to communicate every Arduino Uno board with the central node using the MQTT protocol.
8. **DHT library** to control and interact with the digital temperature and humidity sensor DHT22 from the Arduino Uno board.

We can also find a detailed explanation and an example of use from each of the eight libraries at their corresponding section in chapter eight. We will conclude with the remote node software development section after presenting the flowchart followed by the software in every remote node. This flowchart will give us a complete understanding of how every remote node behaves. As we can see in chapter eight, there are four types of remote nodes, one with an Arduino board controlling one of the first four heating and air cooling blocks seen in chapter two. The main difference in the control software of each of them is in the system control function, where they will execute a different action to their corresponding actuators following the control strategy, analysed in chapter four, for each of the four heating and air cooling blocks, controlled by a remote node in our automation system.

On the other hand, we will describe the software development completed in the central node. First, we will describe the logic of the four MQTT clients developed in the central node, which each one is subscribed to the topic belonging to one of the four heating and air cooling blocks where its remote nodes will publish periodically information about their

performance. This MQTT clients will receive those messages thanks to a Java library, called MQTT Paho, developed by the Eclipse Foundation, and insert them in their corresponding table in our database. More information about the development of this clients can be seen in chapter eight.

Finally, we will explain the Servlets and JSP pages that shape our Java web application. We will show the main views of our Java web application and how the user can interact with it. We will also introduce some knowledge about how the Servlets are processing every user request to our Java web application. In conclusion, we will explain everything related to the central user interface which we have developed in our automation system.

Chapter 9: Performance Tests

During this chapter we have performed some tests in the automation system which we have designed and developed, in order to determine if the performance of the system is the desired and expected.

At first, we performed some tests about the control performance by two remote nodes to their heating and air cooling blocks. During these tests we were able to determine the behaviour between the sensors and actuators attached to the Arduino Uno board from a remote node. In addition, we were able to see the results over the auxiliar interface.

Secondly, we performed some tests in our Java web application to determine if the surfing between the views was done as expected and without error.

Finally, we performed two tests to check the overall system behaviour. On the first check we tested the result of sending a message from an Arduino Uno board to the central node. On the second test, we tested the result of receving an order from the web interface to an Arduino Uno board. In order to perform these tests, we use one of the protoboards which we used also in the first part at this chapter. After concluding these two tests, we were able to determine that the overall performance of the automation system was the desired and expected.

Chapter 10: Budget.

In this chapter we will give an estimation of the total cost regarding the development of this project. As we can see, we have divided the costs in three sections. The first section presents the personal costs, related to the cost derived from the people participating in the project. The second section will present the costs derived from the software licenses needed to perform the project and the hardware components that are necessary to implement our automation system. The third section will describe the indirect costs which have appeared while developing our project.

Finally, we can see the total price combining the three sections. As we can see in chapter ten, the total price of our project is thirteen thousand three hundred fifty seven euros with thirty seven cents.

Chapter 11: Conclusions

During this chapter we will present the conclusions found after developing our automation system and some future developments which could be interesting to perform.

First of all, we will present some conclusions regarding the low-cost obtained in the project and some about the Arduino platform. Later, we will describe our conclusions about the use of a low-cost computer in this project and the protocol MQTT. Finally, we would present what the development of our project can be used for other people and a summary of what we have done and achieved.

As we can see in chapter eleven, our last conclusion about the project is that we have been able to design and implement a low-cost automation system to control a heating and air-cooling system, according to its characteristics, in a building complex higher than a home. An automation system whose architecture is in accordance with the real characteristics of a regular system existing in a building. It can serve as a guideline to anyone, with programming and electronics basic knowledge, interested in control a regular system not just in their homes using low-cost and low energy consuming components.

Secondly, we will describe four future developments that could be useful to perform in the future to increase the scope and functionalities of our automation system. The first future task that we can perform is to increase the web application functionalities. We propose to add two new sections in every remote node view. We can add a chart to show the user the perfomance of that specific remote node by showing the temperatures measured on its heating and air cooling system in an hour, a day or a mont. We can also add another section where the user can automate some actions in a remote node, for example, to turn on or off at a specific time of the day.

The second task that we propose to perform is to try the same automation system but instead of using an Arduino Uno board, to use a board with more capacity, like the Arduino Mega board. By doing this we can offer more functionalities to the automation system controlling the heating and air cooling system. What's more, we can also integrate in our automation system other systems to control, like the lighting system or the smoke detection system, thanks to the bigger computing capacity and memory amount that this other board can offer.

Nowadays, the use of Smart Phones has increased a lot, almost everyone has one. Then, the third future task which we propose is to develop a mobile application that can used also as a central user interface in our automation system.

Finally, the last task we propose is to integrate our automation system in the building complex we presented in chapter three. But before integrating it, we need to homologate every device according to the proper regulations which affect the building and prepare the building so our system works.