UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

"Sistema de monitorización y telegestión remota basado en Arduino para Smart Buildings"

TRABAJO FINAL DE GRADO

Autor/a:
Julio Rubén Sánchez Torrecilla

Tutor/a:
Tomás Carlos Sogorb Devesa

GANDIA, 2014

Agradecimientos:

A todos mis amigos y familia por sus ánimos al volver a estudiar y no dejarme abandonar en los momentos más duros y en especial a mi primo Jorge por las largas conversaciones “frikis” tomando una cerveza.

A Tomás, mi director, por sus consejos sobre el proyecto y su tiempo hasta el último momento.

Y en especial, a Raquel, por estar siempre a mi lado apoyándome en todo y soportar que tenga “mis cacharros” por todas partes.

- Rubén Sánchez -

ÍNDICE

RESUMEN:

El continuo aumento de la demanda energética, el actual marco económico mundial y el fuerte empuje de las Tecnologías de la Información y la Comunicación (TIC) ha propiciado, en los últimos años, la aparición de soluciones tecnológicas que permiten la optimización del consumo energético. Dichos proyectos cobran una gran importancia a la hora de conseguir una reducción de costes y del impacto ambiental, gracias a sistemas cada vez más sofisticados y, por supuesto, económicos.

En el presente proyecto se plantea como objetivo crear un sistema basado en hardware libre con Arduino y un entorno web que sirva de herramienta para monitorizar y gestionar remotamente a través de internet parámetros de un edificio para convertirlo en un “edificio inteligente”.

A lo largo de la memoria se mostrarán las distintitas tecnologías que integran la solución desarrollada y se detallarán los elementos utilizados en el diseño e implementación de las distintas partes, tanto hardware como software, así como el funcionamiento del sistema completo.

Como se podrá apreciar, durante la realización del trabajo se han añadido funcionalidades adicionales a los objetivos originales que potenciaron la transparencia del uso por parte del usuario, así como su versatilidad. Todo ello se ha realizado para conseguir una aplicación comercializable y fácilmente ampliable para futuras aplicaciones.

Palabras clave: Arduino, monitorización, telegestión, edificio inteligente, Raspberry Pi

ABSTRACT:

The continuous increase of the energy requirement, the current global economic environment and the strong increase of the Information and Communication Technologies (ICT) have led, during the last years, to the development of technologic solutions that allows the optimization of energy consumption. These projects have a great relevance since it achieves a reduction in costs and environmental impact, because of the increasingly sophistication systems and, of course, more economical.

The objective of this project is to create a system based on open hardware Arduino and a web environment that serves as a tool to monitor and manage remotely via internet the parameters of a building, making it a "smart building".

This work presents the different technologies that integrate the developed solution and the used components for designing and implementing the several parts of the system, both hardware and software. The correct operation of the entire system will be also detailed.

As described in the project, new features were added to the original objectives, improving the user transparency and functionality, as well as its versatility. All this was implemented in order to make a marketable and easily expandable system or future applications.

Key words: Arduino, monitoring, remote management, smart building, Raspberry Pi

1. INTRODUCCIÓN

El control y monitorización de edificios puede suponer una serie de ventajas considerables tanto para usuarios, como para propietarios en aspectos de seguridad, confort, ahorro energético y reducción de costes de mantenimiento.

Dotando de las infraestructuras necesarias en materia de sistemas de automatización y comunicaciones al edificio y, con una adecuada integración de las mismas, se puede conseguir un “edificio inteligente” que nos permita obtener todos los beneficios mencionados.

En el presente trabajo se tratará, desde un punto de vista teórico-práctico, el proceso de diseño e implementación de un pequeño sistema que permita convertir a un edificio en “inteligente” mediante el uso de tecnologías de hardware libre y soluciones basadas en entorno web y comunicaciones de redes de datos.

1.1. OBJETIVOS Y MOTIVACIÓN

Tal como indica el título, el objetivo de este trabajo es implementar un sistema o plataforma que permita monitorizar y telegestionar de forma tanto local, como remota, equipos y sistemas de un edificio y lo dotarlo de cierta inteligencia.

Por mi trayectoria profesional, he tenido oportunidad de conocer diversos sistemas de control para grandes edificios desde una perspectiva más comercial. Es por ello, que se ha querido trabajar en el proyecto a más bajo nivel para poder aprender el funcionamiento e interconexión de diversas tecnologías y a su vez mejorar el aprendizaje en diversas disciplinas.

Se ha optado por un trabajo teórico-práctico, por lo que se va a diseñar el sistema de forma teórica, pero también se realizará la programación y montaje sobre un prototipo para poder comprobar su funcionamiento.

Más concretamente, los objetivos que se pretenden conseguir con este trabajo son:

- Objetivo principal: Desarrollar un sistema de adquisición de datos basado en la plataforma open-hardware Arduino que permita la integración de diversos sensores y su posterior visualización en remoto.

- Objetivos secundarios:
  1. Configurar un sistema basado en entorno web que cuente con una base de datos capaz de almacenar y mostrar los datos obtenidos a partir de una red de controladores en diversas localizaciones.
  2. Programación del sistema microcontrolador basado en Arduino.
  3. Adaptación de los diferentes sensores al sistema digital.
  4. Permitir accionamientos de relés para el control remoto de circuitos.
  5. Montaje del sistema y comprobación del dispositivo y su comunicación con la plataforma de almacenamiento de datos.

1.2. ORGANIZACIÓN DE LA MEMORIA

Este documento se ha organizado en nueve capítulos más otros apartados que incluyen los anexos, listados de referencias y la bibliografía.

El primer capítulo, en el que nos encontramos, muestra una introducción de la memoria y trata sobre los objetivos, metodología y etapas del proyecto.

El segundo capítulo hace un breve repaso sobre el estado del arte y aclara el porqué de este proyecto y la elección de una tecnología y no otra.

En el tercero se plantean los requisitos que deberá reunir el sistema para poder simular una plataforma de automatización y telegestión y poder cumplir con los objetivos del TFG. Todo ello, tras estudiar soluciones reales y bajo la experiencia del autor.

Entre el cuarto y quinto apartados podremos hacernos una idea de los distintos elementos (dispositivos, aplicaciones y tecnologías), tanto hardware, como software que han sido necesarios para la realización del proyecto, mientras que en el sexto nos centraremos en detalle en el diseño, montaje y programación de todo el conjunto hasta conseguir el sistema definitivo.

En el séptimo capítulo de este trabajo se describen las pruebas de funcionamiento que se han realizado al sistema, los problemas que han ido surgiendo durante el proceso y las soluciones que se han aportado.

Con los datos del capítulo anterior y en función de los objetivos marcados al comienzo de la memoria, se detallarán en el capítulo ocho las conclusiones obtenidas y posibles trabajos futuros.

Para finalizar se incluye un presupuesto del sistema implementado y unos anexos con listados de archivos adjuntos, imágenes y bibliografía incluidos.

1.3. METODOLOGÍA

Tras un análisis de las necesidades globales del sistema y la elección del dispositivo para su realización, se ha fragmentado en bloques funcionales o subsistemas y se ha definido un listado de objetivos. Posteriormente se ha ido programando y probando por separado pequeños fragmentos de código (sketch) para controlar un determinado dispositivo o funcionalidad. A continuación, dichos códigos se han ido incorporando al código general del proyecto.

Por último, se ha realizado la capa de interfaz y visualización y se ha montado un prototipo del sistema para probar el correcto funcionamiento y pulir detalles.

1.4. ETAPAS

El proyecto se ha ejecutado siguiendo el plan de trabajo que se propuso al comienzo del mismo. Durante el proceso, tras investigar más a fondo el estado del arte, se han incluido algunas mejoras, lo que ha llevado a añadir etapas adicionales.

A continuación se resumen los principales pasos:

1. Investigación, búsqueda bibliográfica y estudio del estado del arte de los sistemas Arduino.
2. Elección de elementos hardware (placas Arduino, sensores, shields…) y compra de los mismos.
3. Aprendizaje del lenguaje y forma de programación mediante el IDE de Arduino. Se realizan ejemplos propuestos por la web y consultas en foros.
4. Programación, montaje de circuitos y prueba de diversos subsistemas en Arduino por separado. Este paso se ha realizado varias veces hasta conseguir el correcto funcionamiento de cada subsistema (control de relés, lectura de temperatura y humedad, control de motores…)
5. Comparativa de distintas plataformas de IoT (Internet of Things) para la adquisición, almacenamiento y visualización de datos recogidos por Arduino. Elección de la plataforma Xively.
6. Aprendizaje y pruebas con Arduino y Xively.
7. Unión de los distintos fragmentos de código (sketch) de cada subsistema para que funcionen como un único sistema.
8. Aprendizaje de lenguajes de programación HTML5, JavaScript, CSS3 y Ajax para crear una web de control embebida en Arduino.
9. Mejora: se añade una segunda placa Arduino para añadir funcionalidades como el control de circuitos eléctricos y repartir procesos.
10. Pruebas de funcionamiento de un sistema formado por el Arduino más un shield actuando como web server. Como mejora, el dispositivo deberá publicar una página web propia que permita la visualización en tiempo real de datos recogidos por los sensores y actuar sobre elementos mediante un explorador web.
11. Mejora: comunicación entre las placas Arduino para el intercambio de información, simulando un sistema basado en bus.
12. Mejora: se creará una pequeña aplicación web alojada en un servidor local que de acceso a distintas visualizaciones y controles de los elementos.
13. Mejora: instalación de un servidor LAMP (Linux-Apache-MySQL-PHP) en un PC para alojar la aplicación web local.
14. Mejora: Empleo de una Raspberry Pi para la instalación del servidor LAMP. Este paso incluye el aprendizaje, instalación del sistema operativo (Linux) y diversos paquetes de software.
15. Mejora: Creación de una página web que muestre valores locales de temperatura y humedad relativa en tiempo real para ser visualizada desde una pantalla o monitor en locales de pública concurrencia en cumplimiento del Real Decreto 1826/2009 de noviembre.
16. Redacción de la memoria del proyecto.

2. ANTECEDENTES Y ESTADO DEL ARTE

En este apartado veremos estado actual del entorno tecnológico centrándonos en los puntos que más atañen al objetivo del TFG.

2.1. DOMÓTICA, INMÓTICA, URBÓTICA…

La automatización y telecontrol no es un concepto nuevo. Estas tecnologías se pueden aplicar a diversos ámbitos: si nos referimos a una vivienda, estaríamos hablando de domótica, si se aplica a edificios, se conoce como inmótica, y si abarca a toda una ciudad podría definirse como urbótica.

En este trabajo nos hemos centrado en edificios como podría ser una torre de oficinas, una biblioteca, un pequeño campus como el de Gandía… aunque las soluciones que se plantearán, también tendrían cabida en una gran vivienda e incluso, con una configuración adecuada de las comunicaciones, podría servir para una red de locales de negocios que se encuentren en localizaciones remotas.

En los años 70 ya existían soluciones basadas en protocolo X-10 que permitían la comunicación entre sensores y actuadores de una vivienda que se comunicaban a través de la línea eléctrica.

Con el paso de los años, han ido apareciendo protocolos y sistemas cada vez más complejos y a su vez robustos, que cuentan con unas características más adecuadas a cada campo. Por nombrar algunos, en automatización de edificios y viviendas, podemos encontrar KNX y Lonworks, y en el ámbito industrial podríamos destacar Profibus, CAN, Modbus y un largo etcétera.

No es el objeto de este proyecto ahondar en los procesos de automatización y los buses de campo pues en este trabajo, el objetivo es el uso de otras tecnologías como las redes de datos para implementar un sistema de gestión de edificios.

Los sistemas de gestión de edificios, conocidos como BMS (Building Management System) suelen estar formados por un conjunto de sensores, actuadores, controladores, pasarelas, buses de campo y una aplicación o programa que sirva de interfaz hombre-máquina (HMI) para facilitar la gestión.

Figura 1. Ejemplo sistema de control. Fuente: Iconics Inc.[8]

Cuando el edificio cuenta con unas infraestructuras de redes de datos y otros sistemas informáticos que permiten combinar y tratar la información de todos los subsistemas integrados (automatización iluminación, control HVAC, seguridad, control de accesos, consumos eléctricos…) con otros sistemas informáticos, como por ejemplo un ERP, estaríamos hablando de un “edifico inteligente” o “Smart Building”.

Como se ha comentado antes, esto solo es posible si partimos de un “edificio conectado” y apoyándonos en la tecnología IP.

2.2. SMART BUILDINGS Y EFICIENCIA ENERGÉTICA

Durante el discurso inaugural, *Smart Buildings now and in the future* (Edificios Inteligentes de ahora y del futuro), Jim Sinopoli aseguró que los edificios contabilizan un 41% de energía primaria consumida, más que los sectores del transporte y de la industria. Por ello, subrayó la importancia de potenciar el uso de las nuevas tecnologías para mejorar la eficiencia de los inmuebles [fuente: Casadomo.com. Artículo: “Smart Building Conference celebra su tercera edición en ISE 2014].

Haciendo uso de un sistema permita la monitorización y el control remoto es posible visualizar en tiempo real parámetros como consumos eléctricos y otras variables de la instalación (temperatura, estado de apertura de ventanas y compuertas, iluminación…) y a su vez actuar sobre los distintos sistemas controlados.

Analizando la información obtenida se pueden poner en marcha medidas de ahorro energético y comprobar, de forma inmediata, su efectividad.

2.3. IOT. INTERNET OF THINGS.

Los sistemas de automatización suelen estar formados por multitud de dispositivos interconectados con buses formando capas de control hasta llegar a un nivel en el que podemos encontrar pasarelas o interfaces que los conectan a internet para poder enviar y recibir información, pero si cada dispositivo contase con comunicación propia a internet, estaríamos hablando de un sistema en el que todos los dispositivos “hablan el mismo idioma” y nos encontraríamos con lo que se conoce como el IoT (Internet-of-Things) y el M2M (Machine-to-Machine).

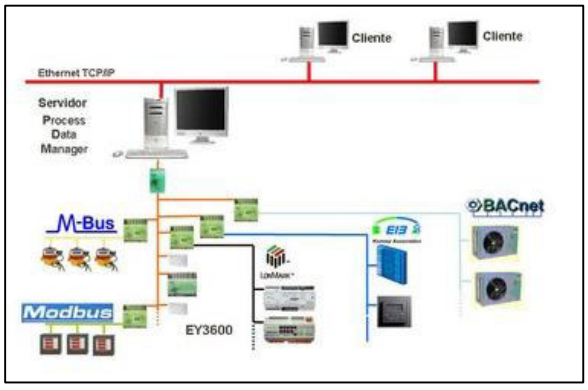

Figura 2. Sistema automatización. Fuente: Sauter Ibérica [14]

Gracias a la aparición de IPv6 (Internet Protocol version 6) se abre una puerta a la conexión de multitud de dispositivos directamente a internet pues el número de direcciones y la seguridad de esta versión han aumentado notablemente.

Es por eso, que están apareciendo multitud de dispositivos que no necesitan una pasarela para poder conectarse a internet, cosa antes impensable. Algunos ejemplos son los “wearables”, los termostatos con comunicación WiFi, las Google Glass, sistemas de tracking para vehículos….

Además de los fabricantes que han ingeniado esos productos, existe una comunidad de “makers” que están desarrollando dispositivos para controlar otros aparatos a partir de microcontroladores de código abierto como Arduino.

2.4. ELECCIÓN DEL HARWARE DE CONTROL: ARDUINO.

Para la realización del proyecto se podrían haber empleado como hardware, módulos domóticos o algún PLC (Programable Logic Controller) existente en el mercado, pero como se ha comentado anteriormente, uno de los objetivos es realizar desde cero un sistema que permita aprender cómo controlar dispositivos de un edificio y los distintos sistemas que lo componen.

Por ello, se ha optado por una placa Arduino que es, en esencia, un microcontrolador con muchas ventajas y que cumple con todas las expectativas, como veremos. También existen otras opciones parecidas como podría haber sido una Raspberry Pi, Beagle One, un barebone… pero cada una de ellas cuenta con unas características específicas que las podrían convertir en mejores candidatas por algún motivo, pero son penalizadas en otros.

Ventajas del modelo elegido:

- ✓ Precio. Existen multitud de modelos de placas originales, todos ellos de muy bajo coste, además de existir versiones de otros fabricantes.
- ✓ Sistema abierto. Existe la posibilidad de fabricar un modelo reducido más económico únicamente con los componentes necesarios y un micro Atmel pues contamos con toda la información y ficheros de diseño.
- ✓ Sistema muy didáctico. Debido a su configuración de hardware y la sencillez del lenguaje, unido a la cantidad de información y ejemplos existentes (compartidos por una gran comunidad), es posible empezar con pequeños proyectos de forma rápida y segura. Además, es compatible con multitud de módulos “plug&play” y el lenguaje *Scratch.*
- ✓ Entradas y salidas disponibles. En función de las necesidades del proyecto, podremos elegir entre las distintas placas, que cuentan con multitud de entradas y salidas digitales, entradas analógicas, así como puertos de comunicaciones.
- ✓ Amplia gama de versiones y accesorios compatibles.
- ✓ Muy extendido y estandarizado. Existen infinidad de librerías de libre distribución para poder comunicarse con hardware y software de terceros.

Desde el momento de la compra de los dispositivos han aparecido nuevos modelos de Arduino con más capacidades y mejoras, pero también suponen un sobrecoste y no aportaban grandes ventajas a este proyecto.

1.4.1. Arduino

Arduino es un una plataforma de hardware libre que se basa en un microcontrolador Atmel AVR y un entorno de desarrollo muy sencillo [1].

Además del microcontrolador, Arduino, en la mayoría de las versiones, cuenta con múltiples entradas/salidas, puerto USB por el que cargar la programación y alimentarlo y otros elementos electrónicos, todo ello montado sobre una placa con conectores de fácil acceso y conexionado.

En la siguiente tabla, extraída de la web de Arduino podemos ver los modelos oficiales disponibles con sus especificaciones. En las dos primeras posiciones, vemos remarcadas las dos placas que usaremos en este proyecto.

Model	Processor	Operating Voltage/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [KB]	SRAM [KB]	Flash [KB]	USB	UART
UNO	**ATmega328**	**5 V/7-12 V**	**16 Mhz**	**6/0**	**14/6**	**1**	**2**	**32**	**Regular**	**1**
Mega 2560	**ATmega2560**	**5 V/7-12 V**	**16 Mhz**	**16/0**	**54/15**	**4**	**8**	**256**	**Regular**	**4**
Due	AT91SAM3X8E	3.3 V/7-12 V	84 Mhz	12/2	54/12	-	96	512	2 Micro	4
Leonardo	ATmega32u4	5 V/7-12 V	16 Mhz	12/0	20/7	1	2.5	32	Micro	1
Mega ADK	ATmega2560	5 V/7-12 V	16 Mhz	16/0	54/15	4	8	256	Regular	4
Micro	ATmega32u4	5 V/7-12 V	16 Mhz	12/0	20/7	1	2.5	32	Micro	1
Mini	ATmega328	5 V/7-9 V	16 Mhz	8/0	14/6	1	2	32	-	-
Nano	ATmega168	5 V/7-9 V	16 Mhz	8/0	14/6	0.512	1	16	Mini-B	1
	ATmega328					1	2	32		
Ethernet	ATmega328	5 V/7-12 V	16 Mhz	6/0	14/4	1	2	32	Regular	-
Esplora	ATmega32u4	5 V/7-12 V	16 Mhz	-	-	1	2.5	32	Micro	-
ArduinoBT	ATmega328	5 V/2.5-12 V	16 Mhz	6/0	14/6	1	2	32	-	1
Fio	ATmega328P	3.3 V/3.7-7 V	8 Mhz	8/0	14/6	1	2	32	Mini	1
Pro (168)	ATmega168	3.3 V/3.35-12 V	8 Mhz	6/0	14/6	0.512	1	16	-	1
Pro (328)	ATmega328	5 V/5-12 V	16 Mhz	6/0	14/6	1	2	32	-	1
Pro Mini	ATmega168	3.3 V/3.35-12 V	8 Mhz	6/0	14/6	0.512	1	16	-	1
		5 V/5-12 V	16Mhz							
LilyPad	ATmega168V	2.7-5.5 V/2.7-5.5 V	8 Mhz	6/0	14/6	0.512	1	16	-	-
	ATmega328V									
LilyPad USB	ATmega32u4	3.3 V/3.8-5V	8 Mhz	4/0	09/4	1	2.5	32	Micro	-
LilyPad Simple	ATmega328	2.7-5.5 V/2.7-5.5 V	8 Mhz	4/0	09/4	1	2	32	-	-
LilyPad SimpleSnap	ATmega328	2.7-5.5 V/2.7-5.5 V	8 Mhz	4/0	09/4	1	2	32	-	-

Tabla 1. Comparativa modelos Arduino oficiales [1]

En los últimos días están apareciendo nuevos modelos como Arduino Yun y Galileo que incorporan procesadores más potentes y de otros fabricantes como Intel.

3. REQUISITOS DEL SISTEMA

Se pretende implementar un sistema que pueda actuar como "cerebro" o plataforma de control de un edificio dotándolo de cierta inteligencia, todo ello, a partir de un microcontrolador como es Arduino.

El objetivo no es monitorizar multitud de sensores, ni crear una gran aplicación tipo SCADA (Supervisory Control And Data Acquisition), sino un sistema de fácil uso, que pueda ser replicable y ampliable y sirva al autor en sus objetivos de aprendizaje.

La plataforma estará compuesta por dos partes:

- Hardware: placas de Arduino que actuarán como nodos de control. También existirán otros elementos auxiliares como sensores, relés, motores… que irán conectados a dichas placas y serán controlados por las mismas.
- Software: será el interfaz que permita interactuar al usuario con el sistema de forma sencilla.

Para el trabajo se van a emplear dos placas Arduino que simularán la red de nodos con una topología distribuida, en la que cada uno de ellos realice unas tareas determinas. De esta manera, se optimiza el funcionamiento de los nodos y se minimiza el "punto único de fallo", propio de topologías centralizadas. Para un gran edificio sería necesario un mayor número de nodos, pero para la realización de este proyecto, con los dos dispositivos elegidos sería más que suficiente para poder probar su funcionamiento.

A partir del estudio de los sistemas de automatización de edificios y de la experiencia personal del autor, se ha creado el siguiente listado de funcionalidades que se han creído necesarias y que deberá cumplir el sistema desarrollado:

Nº	Función
1	Encendido/apagado circuitos 230V
2	Medición de temperatura
3	Medición humedad relativa del aire
4	Medición niveles lumínicos
5	Detección de presencia
6	Ejecución de escenarios (accionamiento simultáneo de varios dispositivos con una única orden).
7	Visualización de parámetros recogidos desde un interfaz web
8	Almacenamiento de datos (data logging)
9	Control remoto mediante interfaz web
10	Control de motores
11	Detección de calidad del aire / sustancias volátiles nocivas.
12	Página web para alimentar un panel informativo de temperatura y humedad en cumplimiento del R.D. 1826/2009 de noviembre.
13	Visualización de gráficas con los históricos de datos registrados.
14	Comunicación entre nodos

Tabla 2. Listado de funcionalidades objetivo

4. HARDWARE EMPLEADO

A continuación se describen las principales características de los elementos hardware utilizados para la elaboración de este proyecto. En la carpeta de anexos se incluyen los datasheet con descripciones más detalladas de cada uno de ellos.

4.1. ARDUINO UNO

La placa Arduino UNO es de las más conocidas y utilizadas para introducirse en el aprendizaje de este dispositivo. En particular, la que se ha empleado es la versión R3 y es con la que se comenzaron a realizar los primeros pasos en este proyecto.

Figura 3. Vista frontal y trasera Arduino UNO Rev3 [1]

Sus características principales son:

Microcontrolador	ATmega328
Voltaje de operación	5V
Voltaje de entrada recomendado	7-12V
Pines digitales E/S	14 (6 disponen de salida PWM)
Pines de entrada analógica	6
Consumo por pin E/S	40mA
Consumo del pin 3.3V	50mA
Memoria flash	32kB (0.5kB empleados por el bootloader)
SRAM	2kB
EEPROM	1kB
Frecuencia de reloj	16MHz

Tabla 3. Resumen características Arduino UNO Rev3

Cuenta con puerto de comunicaciones UART (serie hardware), USB (puerto virtual), comunicación mediante I2C (TWI) y SPI.

El resto de características y el pinout de esta placa podrán encontrarse en la carpeta “Ficheros anexos/datasheet”.

4.2. ARDUINO MEGA

La Arduino Mega 2560 R3 es una placa electrónica basada en el microprocesador Atmega2560 que cuenta con 54 entradas/salidas digitales (de los cuales 14 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UARTs (puertos serie de hardware), un oscilador de cristal de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP y un botón de reset.

Es compatible con la mayoría de los shields diseñados para el Arduino Uno, Duemilanove o Diecimila.

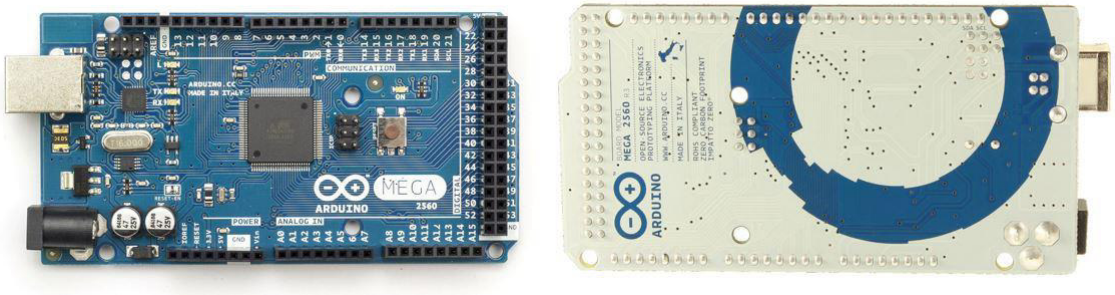

Figura 4. Vista frontal y trasera Arduino Mega Rev3 [1]

Microcontrolador	ATmega2560
Voltaje de operación	5V
Voltaje de entrada recomendado	7-12V
Voltaje de entrada (límites)	6-20V
Pines digitales E/S	54 (14 disponen de salida PWM)
Pines de entrada analógica	16
Consumo por pin E/S	40mA
Consumo del pin 3.3V	50mA
Memoria flash	256kB (8kB empleados por el bootloader)
SRAM	8kB
EEPROM	4kB
Frecuencia de reloj	16MHz

Tabla 4. Resumen características Arduino Mega Rev3

4.3. COMUNICACIONES

Las placas de Arduino, como se ha comentado con anterioridad, cuentan con diversas formas y puertos de comunicación como puede ser el puerto serie (mediante USB o pines), el puerto I2C/TWI, SPI pero en este proyecto se pretende también aprovechar la comunicación mediante tecnología TCP/IP porque está presente en cualquier edificio y es una tendencia creciente. Actualmente están apareciendo modelos nuevos de Arduino que cuentan con puerto Ethernet ya integrado (por ejemplo Arduino YUN), pero en este proyecto se optó con anterioridad por los modelos UNO y Mega para los que existe la opción de instalarles un “Shield” que lo dote de comunicaciones, además de otras ventajas.

4.3.1. Ethernet Shield

Con ésta placa y la ayuda de la librería proporcionada, podremos conectar nuestra placa Arduino a internet y también realizar tanto un pequeño servidor web, como un cliente. La configuración de red se realiza mediante software, por lo que podremos adaptar con facilidad la placa a nuestra red local.

Dispone de un zócalo para tarjetas de memoria micro-SD para poder almacenar ficheros o para utilizarlo como servidor web embedido. También incluye un controlador de reset automático para que el chip interno W5100 esté bien reiniciado y listo para utilizar al arranque (la antigua versión necesitaba ser reiniciada manualmente al inicio).

La placa Arduino se comunica con el módulo W5100 y la micro-SD utilizando el bus SPI (mediante el conector ICSP). Esto se encuentra en los pines digitales 11, 12 y 13 en el modelo UNO y en los pines 50, 51 y 52 del modelo Mega. En ambas placas, el pin 10 es utilizado para seleccionar el W5100 y el pin 4 para la micro-SD. Estos pines no pueden ser utilizados para otros fines mientras la Ethernet Shield esté conectada. En el MEGA, el pin SS (53) no es utilizado pero debe dejarse como salida para que el bus SPI funcione correctamente.

Hay que tener en cuenta que el W5100 y la micro-SD comparten el bus SPI, por lo que sólo uno de ellos puede ser utilizado a la vez. Si se desea utilizar ambos simultáneamente, hay que tenerlo en cuenta al escribir el código.

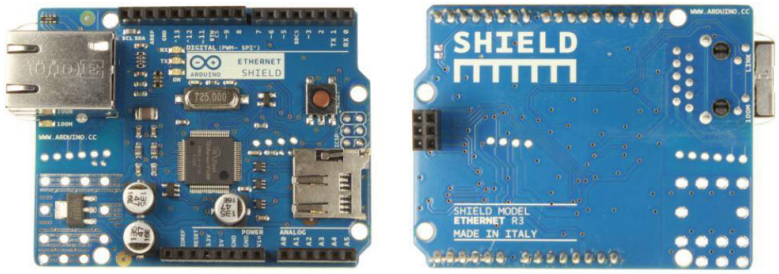

Figura 5. Vista frontal y trasera Ethernet Shield [1]

Esta versión de shield es compatible con tecnología PoE (Power over Ethernet) mediante el empleo de un accesorio.

La placa cuenta con los siguientes LED informativos:

- PWR: indica que la placa cuenta con alimentación
- LINK: indica presencia de una red y parpadea cuando se envía o reciben datos.
- FULLD: indica que la conexión de red es full dúplex.
- 100M: avis de la presencia de una conexión a 100Mb/s.
- RX: parpadea cuando se recibe datos.
- TX: parpadea cuando envía datos.
- COLL: parpadea cuando detecta colisiones en la red.

4.3.2. WiFi Shield

La placa Arduino WiFi Shield permite conectar a Arduino a internet de forma inalámbrica mediante protocolo 802.11b/g (WiFi).

A continuación se resumen sus características principales:

- Alimentación: 5V (proporcionado por Arduino)
- Red: 802.11b/g
- Encriptaciones soportadas: WEP y WPA2
- Conexión con Arduino por el puerto SPI
- Zócalo para tarjeta Micro SD incorporado
- Pines ICSP
- Conexión FTDI para debug
- Conexión Mini-USB para actualizaciones de Firmware

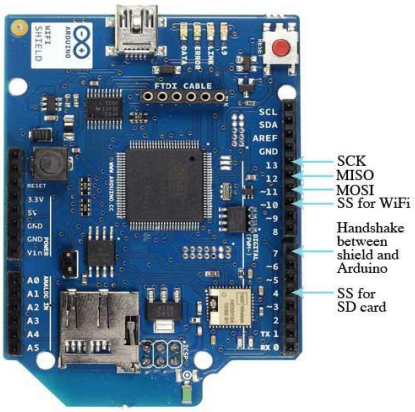

Figura 6. WiFi Shield

La placa cuenta con cuatro indicadores en forma de LEDs:

- L9 (amarillo): conectado al pin 9 de Arduino
- LINK (verde): indica la conexión a una red.
- ERROR (rojo): indica cuando hay un error de comunicación.
- DATA (azul): indica la transmisión/recepción de datos.

4.4. SENSORES

Los sensores utilizados nos permiten recibir información del entorno. Se han elegido buscando el compromiso entre las especificaciones y un bajo precio. Todos ellos cuentan con características suficientes para la realización del proyecto, sin llegar a las de dispositivos profesionales que dispararían su precio, pero que no aportarían nada al objetivo final de este trabajo.

4.4.1. Sensor digital de intensidad lumínica. (BH1750)

El BH1750, también conocido como GY-302, es un circuito integrado con sensor de luz ambiental y comunicación mediante bus I2C. Viene en una placa preparada para soldarle unos conectores y atornillarla para facilitar su instalación.

Características:

- Rango alimentación: 3 ~ 5V
- Rango de datos: 0-65535lx
- Convertidor de Iluminancia a digital incorporado y 16bitAD;
- Bajo consumo de corriente bajo función de “power down”,
- Posibilidad de seleccionar dos tipos de direcciones I2C esclavo
- Función especial de detección (min. 0.11 lx, max. 100000 lx)

Figura 7. Sensor BH1750

4.4.2. Sensor de temperatura y humedad. DHT22

El sensor DHT22 se presenta en un cuerpo de plástico ideal para montar sobre un panel o similar. El sensor de humedad es de tipo capacitivo, mientras que la medición de temperatura se efectúa mediante un termistor. Devuelve una señal digital con la medición (no se necesitan pines analógicos). Permite una lectura cada 2 segundos.

Usa un protocolo de un solo cable pero no es directamente compatible con Dallas One-Wire.

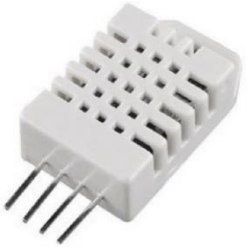

Figura 8. Sensor DHT22

Características**:**

- Sample rate: 0.5 Hz
- Tensión de operación: 3.3 a 6 VDC @1.5mA max. (50uA en stand-by)
- Rango de medición de humedad: 0-100% @±2% (max ±5%) de exactitud.
- Rango de medición de temperatura: -40 a +125ºC ±0.2 ºC de exactitud.

4.4.3. Sensor de intensidad lumínica LDR

Se ha incorporado un sensor LDR (Light Dependent Resistor) o fotorresistor para la medición de niveles lumínicos. Para realizar la medición la instalamos junto con otra resistencia y creamos un divisor de tensión.

En este caso, no importaba tanto la resolución como el precio y dimensiones, pues se instala en exterior de una caja estanca y su misión es encender o apagar la iluminación a partir de un nivel.

4.4.4. Sensor de presencia PIR

El sensor PIR “Passive Infra Red” es un dispositivo piroeléctrico que mide cambios en los niveles de radiación infrarroja emitida por los objetos a su alrededor. Cuando el sensor detecta movimiento, cambia el nivel lógico del pin de salida.

Características:

- Voltaje alimentación: 5Vdc.
- Rango medición: hasta 6m.
- Salida: estado de un pin TTL.
- Cuenta con reguladores para ajustar el retardo de detección y de restablecimiento.

Figura 9. Sensor PIR

4.4.5. Detector de gases MQ-2

El MQ-2 es un sensor de alta precisión sensible a LPG, Propano e Hidrógeno, pero también permite medir otros gases como Metano, humo y vapores de combustibles.

El sensor presenta una baja conductividad en aire limpio, que irá aumentando conforma crezca la concentración de gas.

El modelo elegido viene montado sobre una placa y cuenta con un circuito integrado y la circuitería que facilitan su conexionado, la lectura de valores mediante una entrada analógica de Arduino y una salida digital para accionar un indicador. También incorpora un potenciómetro para realizar un ajuste manual.

Características:

- Voltaje alimentación: +5V
- Resistencia de carga ajustable.
- Tiempo idóneo de precalentado: 48 horas
- Composición del sensor: SnO_2
- Concentraciones: 300-10000ppm

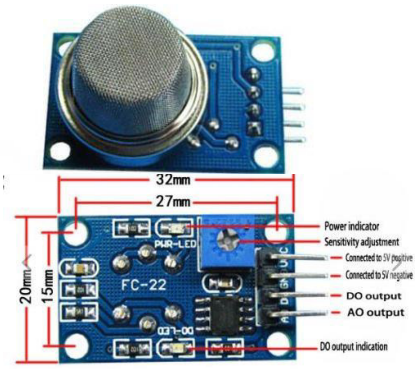

Figura 10. Sensor MQ-2

4.5. ACTUADORES

Los actuadores reciben una orden desde una salida del microcontrolador para convertirla en la activación de un proceso automatizado.

Existen multitud de actuadores que podríamos controlar con Arduino como podrían ser motores stepper, servos, buzzer… En nuestro caso hemos usado unos módulos de relés que permitan accionar circuitos eléctricos de a 230 voltios y también hemos realizado un pequeño montaje para activar un pequeño motor.

4.5.1. Módulo relés

Contamos con dos placas con relés para el accionamiento de circuitos a 230Vac. Una de las placas cuenta con 8 circuitos y la otra con 4, pero sus características son

idénticas. Las entradas son activas en modo LOW (hay que tenerlo en cuenta a la hora de programar el microcontrolador).

Características:

- Voltaje alimentación: +5V
- Voltaje accionamiento: +5V
- Canales opto-acoplados.
- Led de estado en cada canal.
- Consumo por canal: 80mA
- Relés: AC250V 10A ; DC30V 10A
- Dimensiones módulo 4CH: 75.09x54.91mm
- Dimensiones módulo 8CH: 137.91x53.34 mm

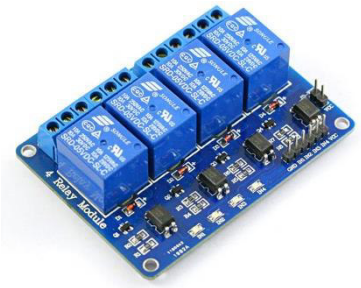

Figura 11. Módulo relés 4CH

La alimentación se puede realizar mediante los conectores +5V y GND a través de Arduino, pero para un aislamiento galvánico completo, cuenta con una entrada de alimentación (retirando el jumper JD-Vcc) a los que acoplar una fuente externa.

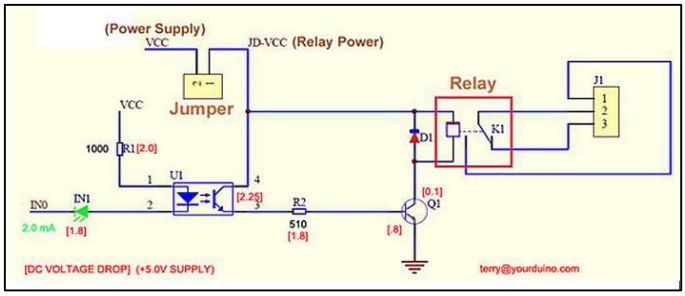

Figura 12 .Esquema de un canal y alimentación del módulo. Cortesía de “yourduino.com”

4.5.2. Control de motor 12V

Para la simulación del control de un motor de un sistema de ventilación o climatización se va a emplear una pequeña placa de diseño propio que accionará un motor de 12V de los usados para la refrigeración de PCs. Existen multitud de placas ya prefabricadas pero se ha optado por una solución económica.

En la siguiente imagen podemos ver el sencillo circuito realizado y el listado de componentes empleado:

- 1 transistor TIP120
- Diodo 1N4007
- Resistencia 1kΩ

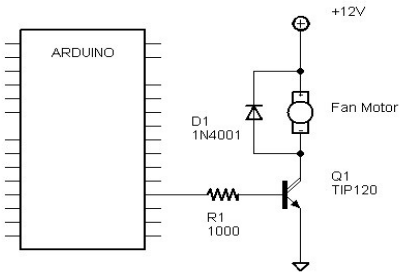

Figura 13. Montaje circuito control motor 12V

4.6. SERVIDOR – RASPBERRY PI

Aunque el controlador Arduino vaya a contar con un web server y publique su propia página, se ha querido incorporar un servidor local en el proyecto como encontraríamos en cualquier edificio automatizado. De esta manera, el usuario debería entrar en la aplicación alojada en el servidor y como alternativa (sobre todo para tareas de mantenimiento), le queda la opción de acceder directamente a la web de un nodo en particular.

Esto también ayuda a que el microcontrolador Arduino no esté sobrecargado si queremos añadir muchos efectos visuales y scripts. Además, cumple con el concepto de topología distribuida que comentábamos al principio.

Raspberry Pi es un ordenador del tamaño de una tarjeta de crédito, de bajo coste, desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

El diseño incluye un System-on-a-chip, que contiene un procesador central (CPU) ARM1176JZF-S a 700MHz, un procesador gráfico (GPU), y 256 MB de memoria RAM. No incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación ni carcasa.

La fundación (www.raspberrypi.org) [13] da soporte para las descargas de las distribuciones para arquitectura ARM, Raspbian (derivada de Debian), RISC OS 5, Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora); y promueve principalmente el aprendizaje del lenguaje de programación Python, y otros lenguajes como Tiny BASIC, C y Perl.

Figura 14: Vista Raspberry Pi (modelo B) [13]

4.7. OTROS ELEMENTOS

Para el montaje y correcto funcionamiento del sistema ha sido necesario el uso de elementos auxiliares. Sin entrar en detalles, podríamos nombrar:

- Fuente de alimentación 5Vdc, 1amp.
- Router con punto de acceso WiFi y switch.
- Protoboard.
- Cable UTP Cat5e.
- Placas prototipado y cableado variado.

5. ENTORNO DE PROGRAMACIÓN Y SOFTWARE UTILIZADO

Para la realización de este proyecto ha sido necesario el uso de diversos programas y herramientas para cada una de las fases y tareas.

A continuación se enumera el software utilizado con una breve descripción, así como las tecnologías o protocolos utilizados:

5.1. IDE ARDUINO

La programación de Arduino se realizado gracias al software gratuito que puede descargarse en su página oficial [1] (“Arduino IDE 1.05”), aunque es posible utilizar otras herramientas como un editor de texto y un programador de microcontroladores o incluso, mediante paquetes de programación como Eclipse y NetBeams (existen pluggins para todos ellos) y otras herramientas que permiten simulaciones y entornos visuales.

Para que nuestro Arduino pueda funcionar, primero se crea un programa, conocido como “sketch” en el editor de texto del IDE. Posteriormente, se compilará y volcará en la memoria del microcontrolador.

Además del editor, el IDE cuenta con otras herramientas que nos facilitan la programación, el uso de librerías, la detección de errores, selección de puerto y modelo de placa a programar… Una de las herramientas más utilizadas a la hora de depurar el funcionamiento del programa es el Monitor Serial, que nos permite comunicarnos mediante el USB de nuestro PC y ver mensajes de la ejecución que hayamos programado.

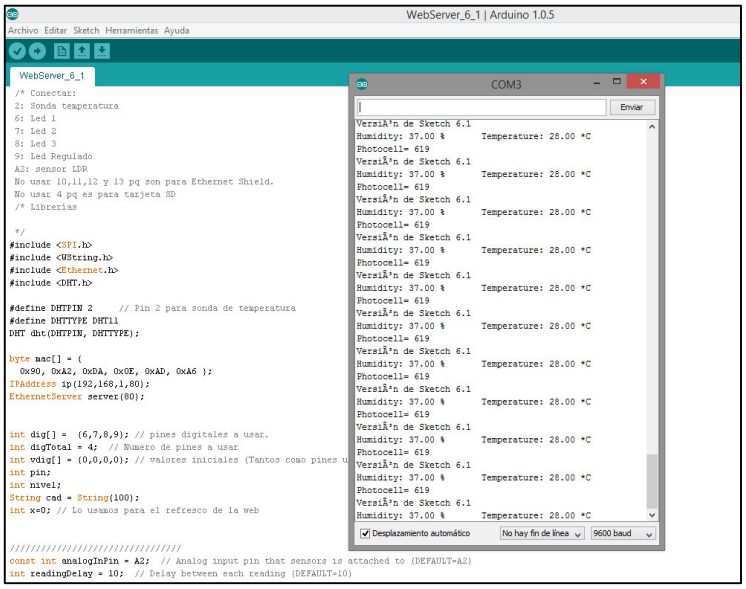

Figura 15. Vista IDE Arduino y Monitor Serie.

5.1.1. Lenguaje programación Arduino

El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectarlo al ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP). Acepta las estructuras de programación de C y la mayoría de C++.

5.1.2. Librerías

El entorno de programación puede extenderse mediante el uso de librerías que proporcionan funcionalidades extra trabajando con hardware y manipulando información. El IDE trae instaladas de serie unas cuantas genéricas que son muy utilizadas como puede ser la de la memoria (EEPROM), conexión con placas Ethernet, WiFi y GPS, uso de puertos…

En otros casos, deberemos descargarnos las específicas que nos proporcione el fabricante de un determinado hardware o software, e importarlas a nuestro IDE para que puedan estar disponibles cuando realicemos la llamada en nuestro sketch.

5.2. ATMEL´S DFU PROGRAMMER. FLIP

Para solventar un problema de incompatibilidades con la versión de firmware del WiFi shield y la versión de IDE de Arduino, es necesario realizar una actualización.

El software gratuito FLIP es un DFU programmer (Devide Firmware Update) de la marca Atmel que se ha usado para actualizar a la última versión de firmware tanto del microcontrolador AT32UC3A1256, encargado de gestionar la pila IP (TCP y UDP) y las APIs de las librerías WiFi, como del módulo HDG104, que proporciona el estándar de conectividad inalábrica IEEE 802.11 b/g.

5.3. XIVELY

Como objetivo para este proyecto se planteaba el registro de las lecturas de los sensores en una base de datos. Aunque se decidió contar con un servidor local que dispone de MySQL, el cual permitiría crear nuestra propia base de datos, se ha optado por usar una solución “Paas” (*Platform as a Service*) alojada en “la nube”.

Las ventajas de usar esta solución son muchas, entre ellas destacaría:

- ✓ Empleo de una tecnología en expansión utilizada por grandes empresas para desarrollo de nuevos productos de IoT.
- ✓ Alta disponibilidad de datos.
- ✓ Facilidad de intercambio de datos entre sistemas gracias las APIs.
- ✓ Se evita contar con servidores propios, direcciones IP estáticas o servicios DDNS para poder comunicar.
- ✓ Representación de la información en gráficas.

Existen multitud de soluciones de plataformas de IoT con características parecidas. A continuación se muestra una tabla comparativa de las más conocidas (Doukas, C., 2012. P59-60) [4].

Name	Features	Free Source	Open Source	URL
Xively	Visualize and store sensor data online	Yes	No	http://xively.com
Nimbits	Data Logging on the Cloud	Yes	Yes	http://www.nimbits.com
ThingSpeak	Visualize and store sensor data online	Yes	Yes	http://thingspeak.com
iDi	Device Cloud platform	Yes	No	http://idigi.com
SensorCloud	Visualize and store sensor data online	Yes	No	http://www.sensorcloud.com
Open.Sen.Se	Internet of Everything platform	Yes	No	http://open.sen.se
Exosite	Platform and Portals for Cloud-based data and device management	Yes	No	http://www.exosite.com
EVRYTHNG	Software Engine and Cloud Platform	Yes	No	http://evrythng.net
Paraimpu	Social tools for thins	Yes	No	http://paraimpu.crs4.it
Manybots	Collect and manage information from various	Yes	No	http://www.manybots.com
Lelylan	Focused on home automation and	Yes	--	http://lelylan.com

Tabla 5. Comparativa plataformas IoT. (Doukas, C., 2012. P59-60) [4]

Como se ha comentado, los servicios que ofrecen estas y las que van apareciendo todos los días, son similares y van adaptándose con el paso del tiempo.

Para nuestro proyecto se ha optado por Xively by LogMeIn [18] (anteriormente denominada COSM y Pachube) por comodidad al conocerla con anterioridad y por su fiabilidad demostrada.

Esta solución que cuenta con millones de dispositivos conectado, pionera en su sector, originalmente era gratuita, pero tras su gran éxito y posterior compra por parte de LogMeIn, pasó a ser de pago, aunque esto es para obtener unos servicios “premium” de asistencia, conexión de alto número de dispositivos y disponibilidad de comunicación y almacenamiento.

Si lo que queremos es conectar un pequeño número de dispositivos y no precisamos de altas tasas de transferencia, es una solución perfecta.

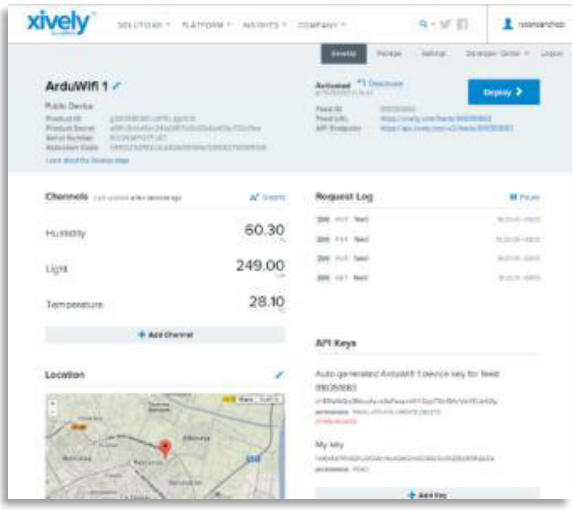

Figura 16. Vista pantalla control Xively [18]

5.4. DREAMWEAVER Y KOMPOZER

Para la creación del entorno web fue necesario el uso de un editor adecuado. Debido a los reducidos conocimientos en HTML y CSS con que se contaba al comienzo de este proyecto, se optó por la utilización de herramientas de edición visual de páginas web, lo que se conoce como editores WYSIWYG, acrónimo de *What You See Is What You Get.*

En primer lugar se optó por Kompozer por su sencillez, pero más adelante, conforme fueron aumentando los requisitos, así como los conocimientos en HTML5, CSS3 y la necesidad de uso de JavaScript, se optó por Dreamweaver que, aunque se trata también de un entorno visual, cuenta con mayores capacidades.

5.5. HTML5, JAVASCRIPT, CSS3, AJAX

Los lenguajes utilizados en el entorno web de creación propia son HTML5 para la estructura de la página, CSS3 para la edición de estilos y JavaScript para dar dinamismo a la página.

Éste último ha sido necesario, sobre todo, para intercambio de información entre páginas. Así, se han utilizado scripts de jQuery para la lectura de valores desde Xively a la web alojada en nuestro servidor local (Raspberry Pi). También se ha empleado AJAX (Asynchronous JavaScript and XML) para el intercambio dinámico de información entre la web alojada en nuestro Arduino y el navegador. Esto evita que la página deba ser recargada constantemente para que se actualice cada vez que un valor cambia o que se debe accionar un circuito desde un botón.

5.6. PUTTY

PuTTY, es un cliente de acceso remoto a máquinas informáticas de cualquier tipo mediante *SSH*, *Telnet* o *RLogin*, para plataformas Windows y UNIX.

Este programa se usa para realizar conexiones mediante SSH con nuestro servidor en la placa Raspberry Pi y poder instalar aplicaciones o actualizar información sin necesidad de un monitor, teclado o ratón.

5.7. TERMINAL V1.9B

Como se ha comentado anteriormente, el IDE de Arduino dispone de un Monitor Serie que nos permite visualizar mensajes, previamente programados, para realizar tareas de depuración del código y saber que todo está funcionando correctamente. Este proceso se realiza a través del cable USB y seleccionando el puerto COM en que se encuentre nuestro Arduino.

El problema del Monitor Serie del IDE es que solo nos permite visualizar un puerto, así que no es posible ver en tiempo real información de dos placas a la vez. Como esto era necesario, se optó por el uso de Terminal v1.9b que, además de permitirnos abrir varias conexiones, cuenta con herramientas adicionales como el registro de datos, grabación de macros, representaciones en binario y hexadecimal...

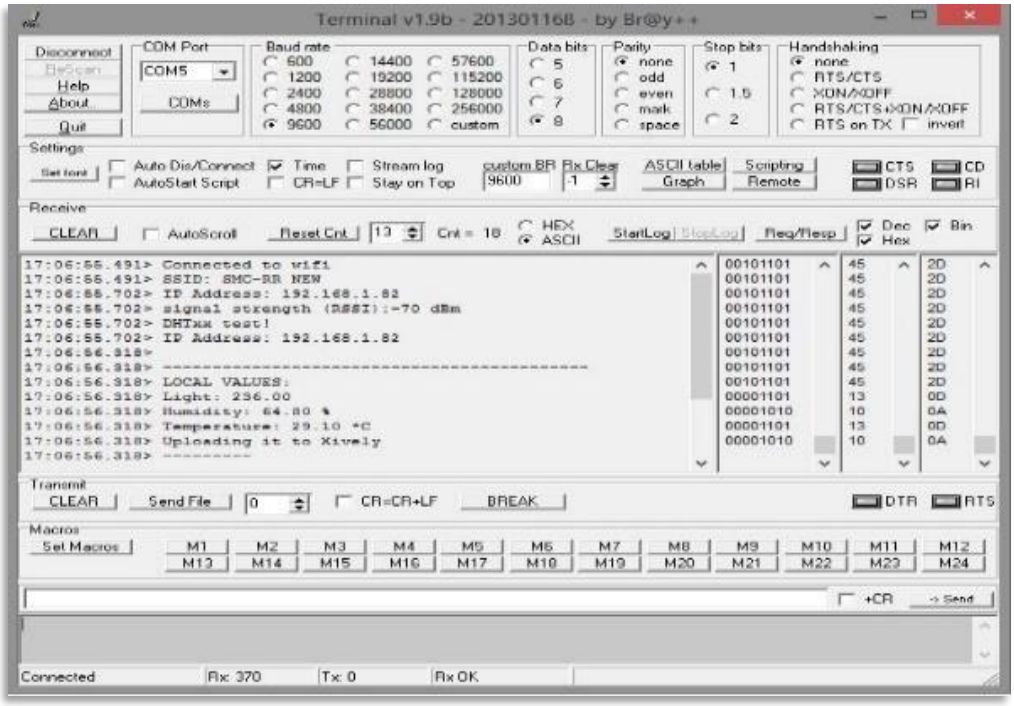

Figura 17. Pantalla control Terminal v1.9b con lecturas del puerto serie

6. DISEÑO, MONTAJE Y EJECUCIÓN

El diseño del sistema se ha realizado como si se tratase de una aplicación de gestión real para un pequeño “edificio inteligente”.

Basándonos en los requisitos que se proponían en el correspondiente apartado, se ha diseñado la solución del sistema.

Para no aumentar demasiado la envergadura del proyecto, no se han usado protocolos de bus propios de la automatización de edificios, como podrían ser KNX, Lonworks, Modbus, ProfiBus… En su lugar, se ha optado por una comunicación vía TCP/IP. En cualquier caso, existen librerías y multitud de ejemplos para fabricar interfaces sencillos que permiten la integración de dichos buses.

No solo se ha elegido esta última tecnología por razones prácticas del proyecto, sino también porque, como se comentaba al comienzo de esta memoria, el avance del “Internet de las Cosas” está sirviendo para que cada día más dispositivos y soluciones se desarrollen y funcionen interconectados, ya sea mediante comunicaciones inalámbricas, Ethernet o mediante Internet móvil (con una tarjeta GPRS/3G/4G), todo ello sobre tecnología IP.

En cualquier caso, a modo de ampliación o mejora, se ha incluido una comunicación cableada entre los dos nodos que podría simular un enlace mediante bus de los mismos.

Para la parte visual, se ha creado un sencillo interfaz basado en entorno web que permita visualizar y actuar sobre los elementos desde cualquier navegador. De esta forma se podrá interactuar mediante un PC, portátil, Tablet e incluso Smartphone.

6.1. ESTRUCTURA DEL SISTEMA

Como se comentó con anterioridad, la plataforma o sistema se compone de dos partes: hardware y software.

En el esquema que podremos observar en la siguiente página, existen dos placas Arduino que serán los nodos a los que irán conectados los distintos sensores y actuadores. También vemos el servidor local, instalado en la placa Raspberry Pi y la base de datos que nos proporciona Xively.

También se incluye la infraestructura de red, soportada por el router-switch con punto de acceso inalámbrico (WiFi), el cableado Ethernet y la conectividad a Internet.

Por último, se han incluido distintos sistemas de usuario para el acceso a la plataforma que pueden ser un PC, Tablet o Smartphone.

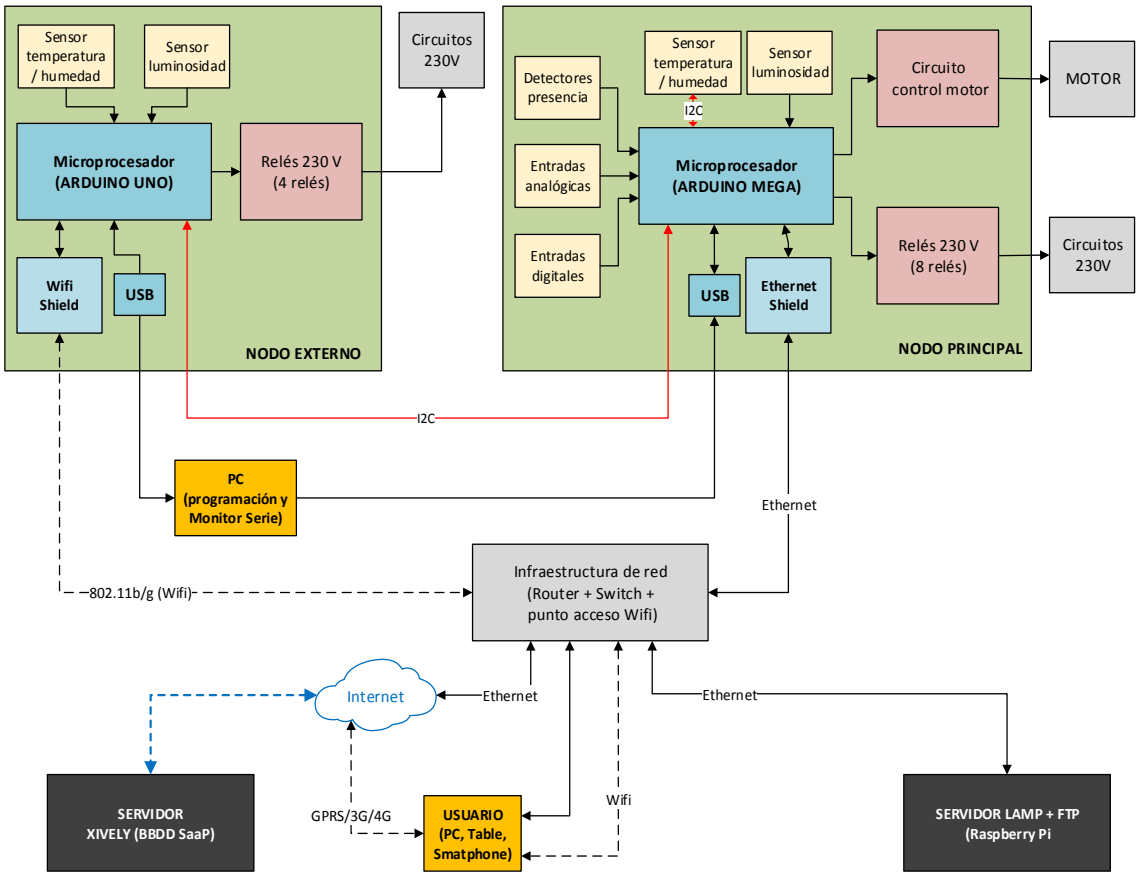

Figura 18. Diagrama bloques estructura sistema

6.1.1. Nodo principal.

Está compuesto por una placa Arduino Mega y un shield Ethernet. Dicho conjunto será el encargado de facilitarnos un web server en el que alojaremos la página de control del sistema que habremos de diseñar y programar.

Se instalará en el interior del edificio y se le ha llamado “Principal” por contar con el servidor y el máximo número de funciones.

También controlará y monitorizará:

- ✓ Encendido/apagado de ocho circuitos a 230V.
- ✓ Monitorización del sensor de temperatura y humedad.
- ✓ Lectura valores de luminosidad.
- ✓ Detección de presencia mediante sensores PIR en dos zonas.
- ✓ Dos entradas digitales de uso general
- ✓ Encendido, apagado y control de velocidad de un motor de 12V para simular un sistema de ventilación.
- ✓ Monitorización del sensor de gases.
- ✓ Dos entradas analógicas de uso general

En la siguiente imagen podemos apreciar el conexionado de la placa Arduino y el resto de elementos que rodean al nodo principal.

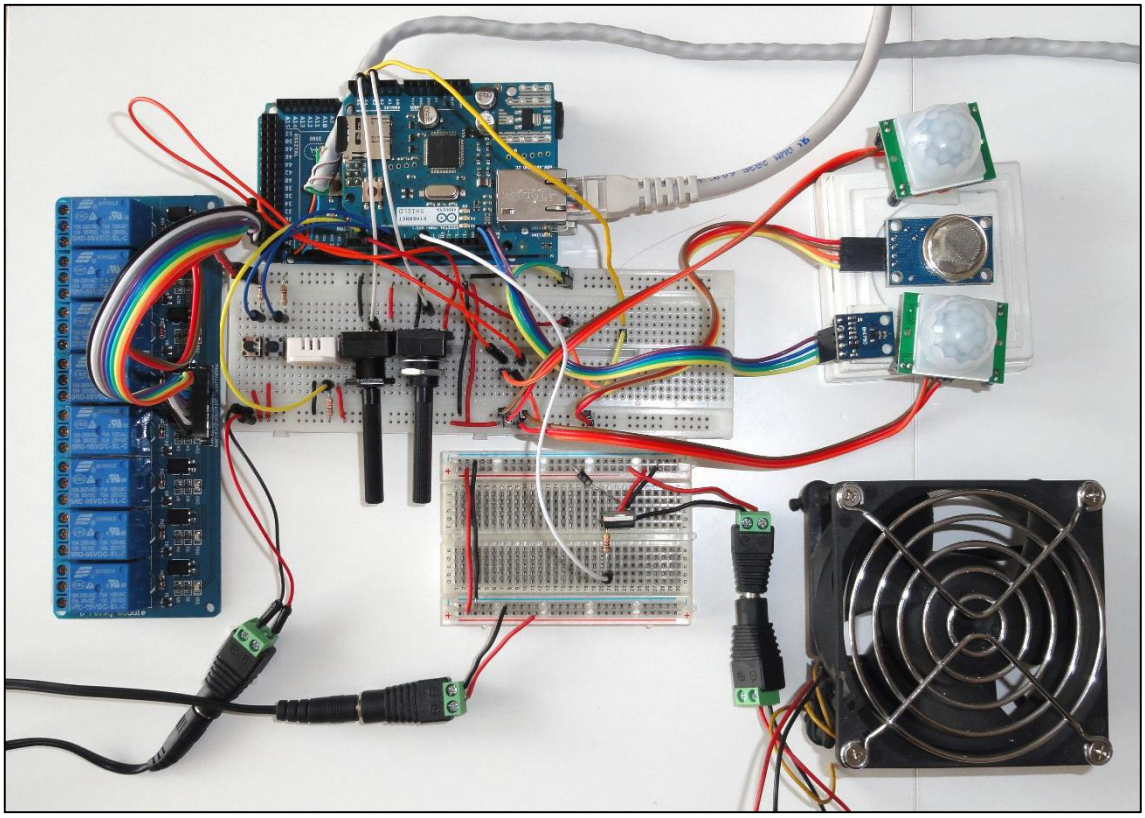

Figura 19. Vista nodo principal

Es posible alimentar la placa desde el propio puerto USB, mediante el pin Vin (unido al USB) por el que deberemos introducir 5 voltios y también es posible conectar una fuente de alimentación que suministre un voltaje en continua que cumpla con el rango de sus especificaciones. La alimentación de los sensores que lo precisen, se puede obtener de los pines de +5 y GND. Si el dispositivo presenta un alto consumo de corriente, puede ser recomendable usar una fuente externa. Del mismo modo, si nuestra placa cuenta con muchos dispositivos adheridos, es importante que la fuente de alimentación pueda suministra un nivel de corriente adecuado. Si no, se producirán fallos de funcionamiento. El motor requiere una fuente adicional de 12Vdc.

6.1.2. Nodo secundario.

Está formado por una placa de Arduino UNO y un shield WiFi al que se le ha incorporado un sensor de temperatura y humedad relativa además de un sensor LDR y el módulo de cuatro relés.

Este nodo se conecta a internet mediante su conexión vía WiFi y permite la subida de los datos registrados al servidor Xively.

Para asemejarse más a un caso real de un nodo aislado, el shield de comunicaciones podría haberse elegido con tecnología GPRS/3G/4G en lugar de WiFi, puesto que los datos a enviar son reducidos. Pero para esto, habría sido necesario la

compra de una tarjeta de alguna operadora con el coste que ello supondría y las ventajas tanto didácticas, como funcionales, serían prácticamente nulas pues la configuración es muy similar. Por otro lado, si el sistema se instalase en el Campus de la UPV, el acceso a la red inalámbrica está asegurado.

Para seguir con las semejanzas a la estructura de los sistemas de automatización de edificios se le ha querido dotar de una funcionalidad adicional, la de pasarela. Es por ello que se ha realizado una conexión mediante I2C con el nodo principal. Esto simula un bus de datos que nos permite intercambiar información entre ellos y así, también subir información recogida por otro nodo hasta Xively.

Igual que en el caso de la elección del shield, se podría haber optado por otros tipos de interconexión entre los nodos como podría ser la inalámbrica, mediante, por ejemplo, Bluetooth, Zigbee (Arduino dispone de módulos XBee) o cualquier otro radio enlace, pero se ha preferido simular un bus porque, como se ha comentado antes, en la automatización de edificios, el enlace entre los principales nodos o módulos suele hacerse mediante un sistema cableado.

Otra opción hubiese sido la comunicación seria entre los dispositivos, pero esto puede complicar la tarea de programación a la hora de depurar código porque la placa Arduino Uno solo cuenta con un único puerto serie (Arduino Mega cuenta con 4). Además es más tedioso a la hora de programar el microcontrolador pues, deberemos desconectar el bus para que el IDE pueda cargar el sketch a través del puerto USB.

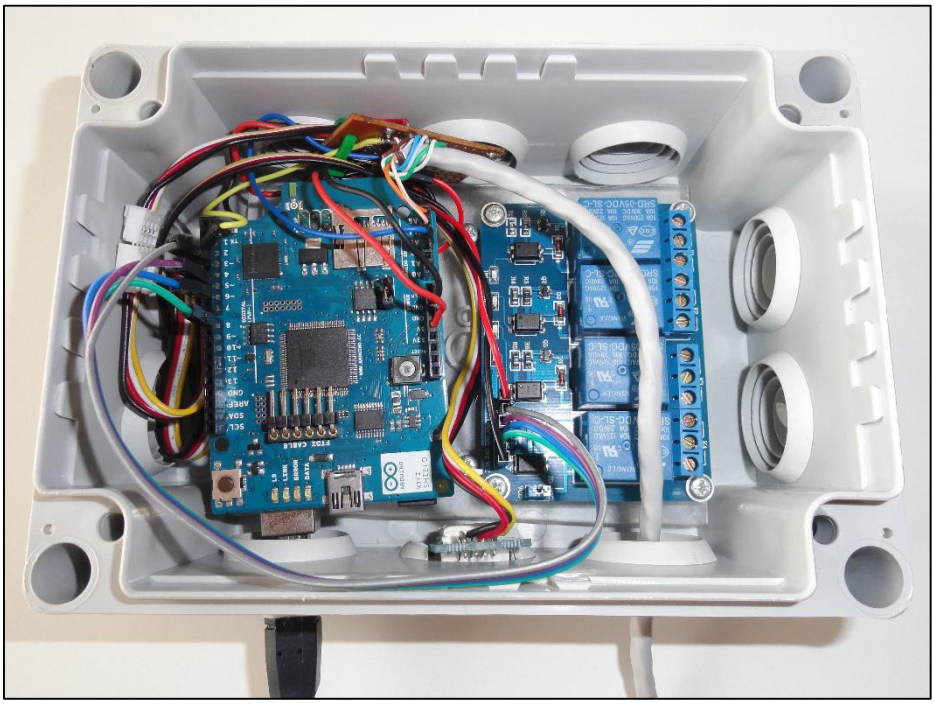

Figura 20. Vista nodo secundario en caja estanca preparado para exteriores

Como se comentaba en el apartado anterior, existen diversas formas suministrar alimentación a Arduino. En este caso, al contar con pocos dispositivos e ir instalado dentro de una caja estanca de conexiones, se ha optado por dejar el cable USB para permitir futuras reprogramaciones con solo desconectarlo del adaptador 230Vac / 5Vdc

y conectarlo al puerto USB de nuestro PC. Esto resultaría muy útil para instaladores y personal de mantenimiento.

6.1.3. Interfaz de control

Como se ha ido comentado en los apartados anteriores, aunque exista un hardware que realice las tareas de automatización y control, es necesario que exista un interfaz que permita al usuario manejar la información y actuar sobre los dispositivos de forma sencilla.

Para ello contamos con el entorno web que consta de tres partes bien diferenciadas, pero todas ellas interconectadas.

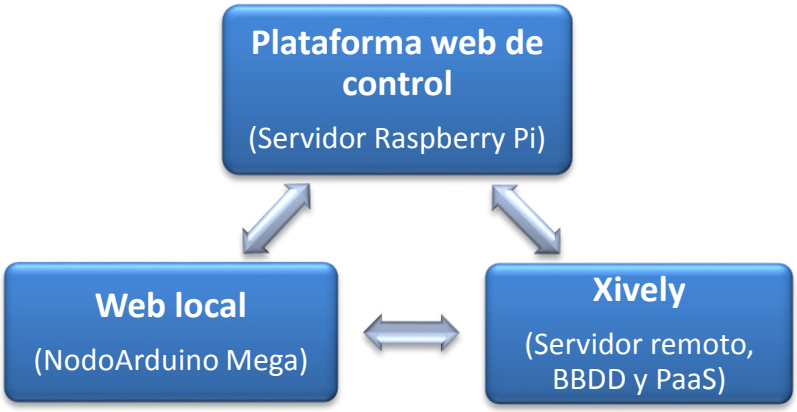

Figura 21. Esquema tres partes del interfaz de control

A. Plataforma web de control

Es la parte del interfaz web que se alberga en el servidor local. En nuestro caso, como se ha comentado antes, una Raspberry Pi con un servidor LAMP y FTP. A continuación se describen las pantallas y se muestra una imagen de ellas.

- ✓ Pantalla de acceso. Cuenta con un acceso sencillo que nos pide una contraseña para permitir el paso al resto de pantallas. Se ha fijado el “1234” por defecto. Tampoco se ha incluido una gran seguridad para la aplicación pues no entraba en los objetivos del trabajo.

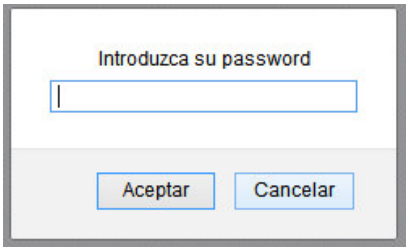

Figura 22. Pantalla acceso

✓ Pantalla principal (Home). En ella podremos ver un mapa de Google Maps centrado en nuestra instalación y que nos muestra la localización de los nodos.

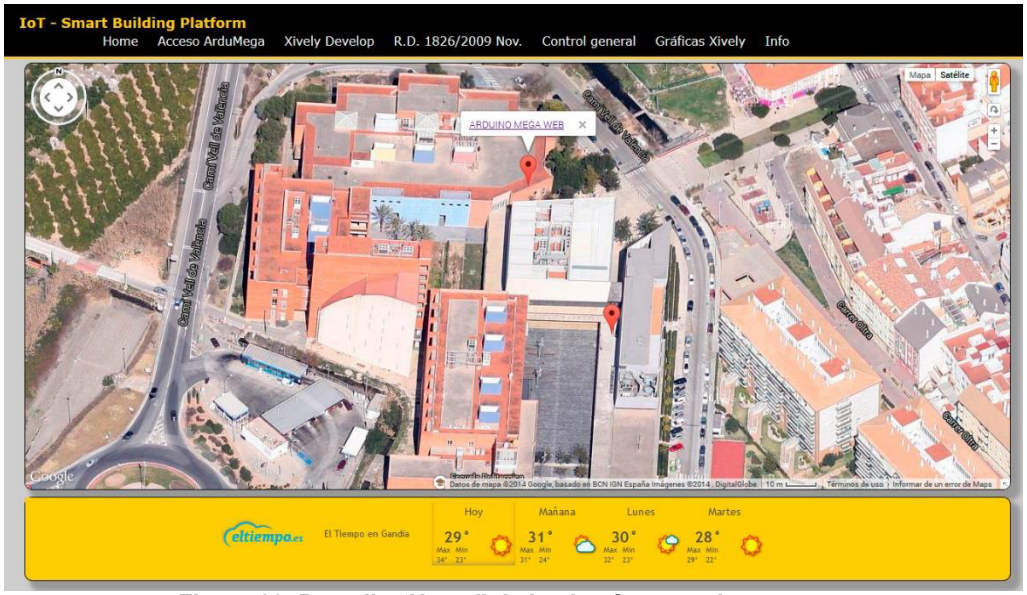

Figura 23. Pantalla "Home" de la plataforma web

Pinchando sobre cada uno de los "markers" podremos obtener información de cual se trata y si cuenta con link, nos abrirá su web local.

También encontramos un menú con los accesos a otras pantallas y un widget que nos informa del tiempo y la previsión a cuatro días.

✓ Pantalla "Acceso ArduMega". Nos lleva a la web que publica el servidor incorporado en el Arduino Mega. Desde esta pantalla podemos ejecutar acciones sobre circuitos, visualizar información en tiempo real de los sensores con que cuenta y ejecutar "escenarios" que consisten en lanzar una serie de acciones simultáneas con una sola llamada.

Veremos con más detalle el funcionamiento de esta pantalla en el siguiente apartado donde explicaremos cómo funciona el servidor alojado en el nodo formado por Arduino Mega y el Ethernet Shield.

✓ Pantalla "Xively Develop". Este link nos lanza a la pantalla de Xively en la que trabajaremos en la fase de desarrollo de nuestro proyecto.

En ella encontramos toda la información sobre la cuenta de usuario, los canales que hemos creado, gráficas en tiempo real, las API Keys para conectar dispositivos, un registro de los mensajes entrantes y salientes (Request Log) y otras funcionalidades como programar "*triggers*" que ejecuten una acción asociada a algún evento o lectura recibidos.

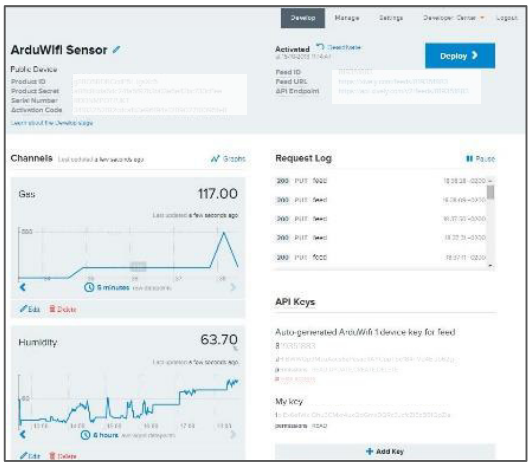

Figura 24. Pantalla “Develop” en Xively [18]

En esta pantalla también podemos definir “*Metadata”* asociada a nuestro dispositivo, incluir la localización en Google Maps y ver información detallada del contenido de los mensajes que se intercambian entre Xively y nuestro Arduino.

- ✓ Pantalla “R.D. 1826/2009 Nov.” En esta pantalla se muestran los valores de temperatura y humedad relativa del aire en cumplimiento del Real Decreto 1826/2009, del 27 de noviembre en el que se modifica el Reglamento de instalaciones técnicas de edificios (RITE) para la mejora de la eficiencia energética.

  En dicho documento [2], entre otros, se fijan los valores límite de temperatura y humedad en locales de uso administrativo, comercial y de pública concurrencia y en el apartado “*I.T. 3.8.3 Procedimiento de verificación”* se indica que dichos valores deberán ser monitorizados en todo momento y deberán ser visibles mediante un dispositivo de unas dimensiones (mínimo DIN A3) con una precisión de ± 0,5 ºC en locales de más de 1.000m^2.

  Es por ello que se incluye esta pantalla con la información recogida por nuestro sensor y que se podría mostrar en cualquier monitor o TV que cuente con acceso a la red, por ejemplo, en los monitores de los pasillos de la UPV.

Figura 25. **Pantalla “R.D. 1826/2009 Nov.”**

- ✓ Pantalla "Control General: Permite la lectura de los distintos parámetros de los distintos elementos del sistema. En el caso de contar con muchos nodos, se agruparía la información por categorías, pero para este proyecto se ha simplificado.

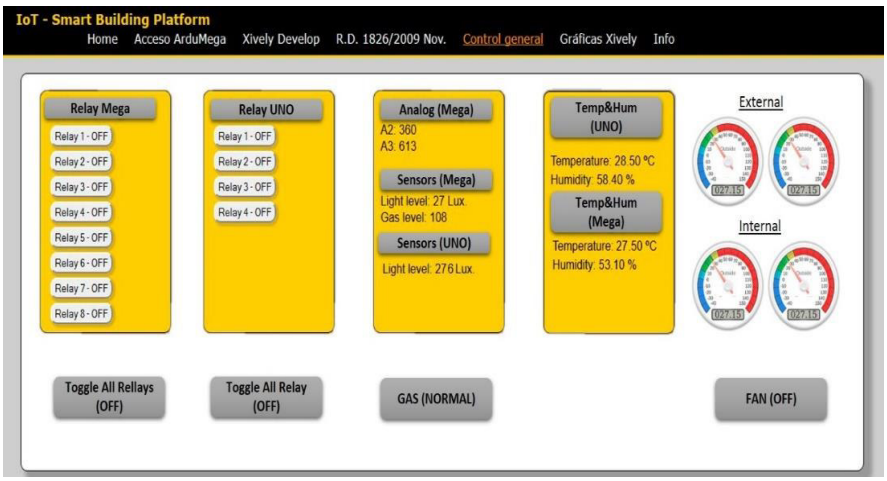

Figura 26. Pantalla control general del servidor

- ✓ Pantalla "Gráficas Xively". Nos muestra las gráficas con todos los datos almacenados en Xively. Podremos visualizarlos en un listado los valores en tiempo real o desplegar las gráficas.

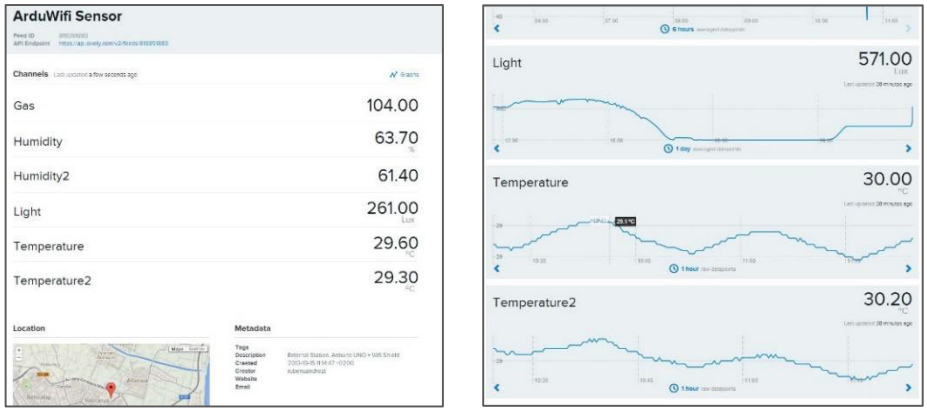

Figura 27. Vista de valores en tiempo real y gráficas en Xively

En las gráficas se mostrarán históricos de datos que se van actualizando en tiempo real. También es posible seleccionar la escala temporal desde los 5 minutos, hasta los 3 meses, según nuestras necesidades.

Esto puede ser muy útil para revisar la evolución de un determinado valor en función a otro parámetro, como por ejemplo, comparar el nivel de luminosidad exterior con la temperatura.

- ✓ Pantalla "Info". Este último acceso se ha dejado por si se quiere incluir información del proyecto, algún manual de uso, enlaces a datasheet…

B. Web local del nodo Arduino Mega

A la placa Arduino Mega se le ha acoplado un “*shield”* que lo dota de conectividad Ethernet y entre otros, nos permite configurarlo como servidor web. Esta funcionalidad es la que se ha utilizado para que el conjunto publique una página web que nos permita monitorizar y actuar sobre los dispositivos con que se comunica.

La página web se aloja en la tarjeta micro-SD que deberá ser introducida en el lector con que cuenta el Ethernet shield. En el apartado de “Programación” podremos ver las ventajas de haber usado esta técnica y no otra.

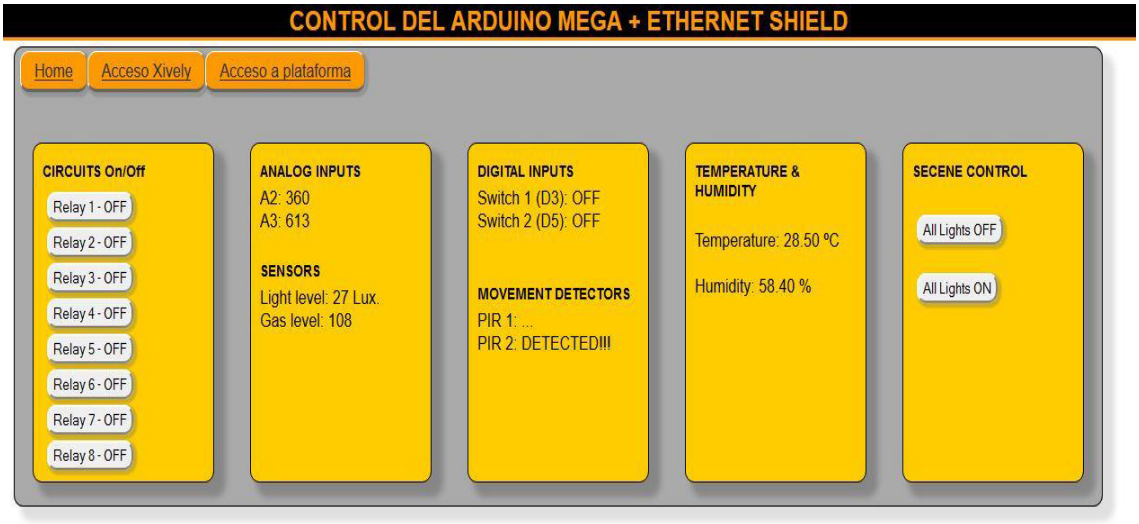

Figura 28. Vista de la web local publicada por el nodo principal

En la imagen anterior podemos observar la pantalla que veríamos con nuestro navegador y desde la controlaremos las funciones.

Se ha dividido en varios bloques en los que se ha dejado espacio para ir añadiendo más dispositivos. A continuación se describe su funcionamiento:

- ✓ Circuits On/Off: mediante la pulsación de los botones, accionamos en tiempo real los distintos relés que se han acoplado a la placa. Además, la información de estado se actualiza en tiempo real, por lo que si se pulsase desde otro punto (por ejemplo otro usuario desde otro navegador), la orden se ejecutaría y el estado cambiaría.

- ✓ Analog Inputs / Sensors: aquí visualizamos en tiempo real los valores de dos entradas analógicas genéricas y de los sensores de luminosidad y lo que hemos llamado “Gas” que es la lectura que nos aporta el sensor de humo y sustancias volátiles.

  Para no alargar más el TFG, no se han incluido los valores en unidades ppm (partículas por millón) porque para realizar ese ajuste, se debería realizar un programa más completo que incluya la calibración del dispositivo y separar entre

las distintas las distintas curvas de lectura según parámetros ambientales y sustancia a detectar.

En cualquier caso, con las lecturas que ahora realiza, si se acerca una fuente de gas, como un mechero, o un algodón impregnado en alcohol, podremos ver cómo el valor se dispara y podríamos asociarlo a una alerta.

- ✓ Digital Inputs: En este bloque se muestra el estado de dos entradas digitales accionadas por dos interruptores. Al igual que las entradas analógicas, su uso es genérico (en función de lo que conectemos) y, en lugar de interruptores podríamos estar monitorizando relés de algún dispositivo como las salidas digitales de una central de alarmas, el arranque de un grupo electrógeno, salidas de relé de un termostato y un largo etcétera de dispositivos que podríamos encontrar en un edificio. Dentro del bloque también se muestra el estado de los sensores PIR.

- ✓ Temperature & humidity: Muestra los valores de temperatura y humedad relativa del aire que registra el sensor DHT22 que lleva conectado el Arduino Mega. En este bloque se ha dejado más espacio pues la placa dispone de múltiples entradas libres a las que podríamos acoplar más sensores y monitorizar más zonas o salas.

- ✓ Scene Control: En este bloque se incluirán lo que se conoce como “escenas” o “escenarios” en el argot de la domótica e inmótica, que no es más que la agrupación de acciones sobre ciertos dispositivos para que se ejecuten de forma conjunta, si se da una determinada condición de entrada.

  Las combinaciones son infinitas pues podemos asociar el estado de uno o varios sensores con el accionamiento de determinados dispositivos.

  En nuestro caso, hemos programado, como ejemplo, dos muy comunes en sistemas de automatización: “All lights ON” y “All lights OFF”. Sea cual sea el estado en que se encuentren los relés, al accionar la primera, quedarán todos activados y del mismo modo, al pulsar sobre el botón “All lights OFF” se abrirán todos los relés dejando todos los circuitos desconectados.

6.2. MONTAJE

En este apartado veremos cómo se han conectado los diversos dispositivos para conseguir su funcionamiento. Como se comentaba al principio del documento, se ha querido probar todo lo que se ha programado, así que ha sido necesario contar con los equipos y conectarlos a una placa de pruebas.

Cada elemento, ya sea sensor, actuador, o interfaz cuenta con unas características que deberán ser tenidas en cuenta a la hora del cableado y su comunicación. En la carpeta de Anexos se incluyen datasheet e información detallada del hardware utilizado por lo que aquí solo vamos a centrarnos en el cómo y dónde conectamos cada elemento, y por supuesto, el porqué.

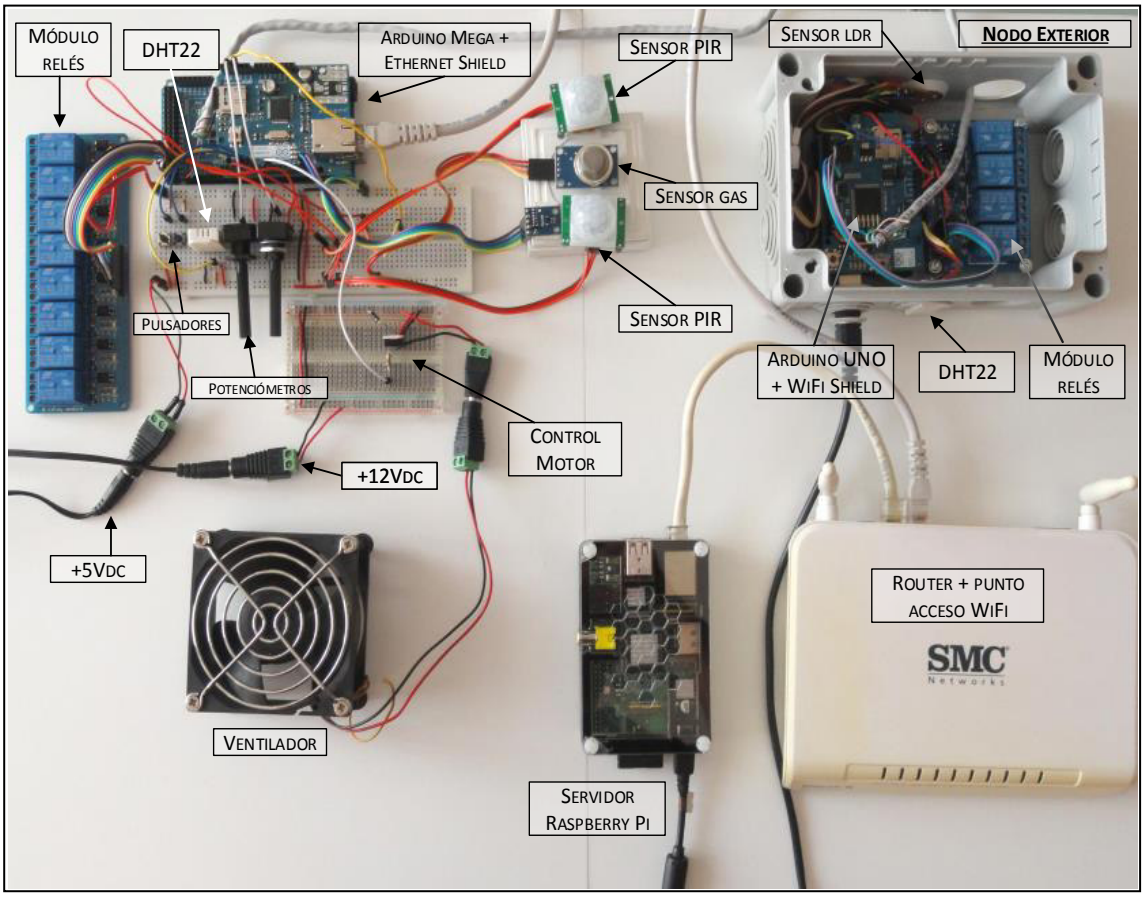

Figura 28. Montaje completo del sistema

6.2.1. Módulos relés.

La función de los módulos de relés es el accionamiento de circuitos a 230V. La placa que va conectada al Arduino Mega cuenta ocho relés que será accionada por ocho salidas digitales, mientras que Arduino UNO cuenta con una placa de 4 relés.

Esta placa cuenta con opto-acopladores lo que asegura que exista una separación galvánica entre la parte de control, que funciona a 5V y la de potencia (a 230V). Esto nos permite trabajar con más seguridad, evitando el paso de un voltaje elevado a nuestro Arduino, lo que supondría su avería.

Nº Pin	Nº Relé	Unidad Control	Nº pin Arduino	(Circuito asociado)
In1	1	Arduino Mega	22	Alumbrado 1
In2	2	Arduino Mega	23	Alumbrado 2
In3	3	Arduino Mega	24	Alumbrado 3
In4	4	Arduino Mega	25	Alumbrado 4
In5	5	Arduino Mega	26	Alumbrado 5
In6	6	Arduino Mega	27	Tomas corriente 1 (o fase R)
In7	7	Arduino Mega	28	Tomas corriente 2 (o fase S)
In8	8	Arduino Mega	29	Tomas corriente 3 (o fase T)

In1	1	Arduino UNO	3	Farolas (fase R)
In2	2	Arduino UNO	5	Farolas (fase S)
In3	3	Arduino UNO	6	Farolas (fase T)
In4	4	Arduino UNO	8	Focos exteriores
Vcc	Alimentación 5V de la placa de relés.			
GND	Masa placa relés.			

Tabla 6. Conexionado de módulos de relés a Arduino

En la tabla anterior podemos ver la relación de cada relé, con el pin que lo controla y el circuito asociado. Esto último es solo un ejemplo de la información que deberíamos mostrar si estuviésemos haciendo el proyecto de un edificio. De esta manera, tendríamos la información para facilitar el conexionado y las futuras revisiones a un instalador o personal de mantenimiento.

Del mismo modo, en los esquemas unifilares de los cuadros eléctricos y los esquemas de principio de climatización, debería aparecer esta asociación.

El haber usado los pines del 22 al 29 en la placa Arduino Mega ha respondido simplemente a una cuestión de comodidad para el conexionado, pues estos se encuentran en un extremo y así se evita cruzar demasiados cables.

En el Arduino UNO se ha saltado los pines 4 y 7, como veremos posteriormente, pues son utilizados por el WiFi Shield.

6.2.2. Sensor temperatura y humedad relativa (DHT22)

En cada placa de Arduino hemos conectado un sensor combinado de temperatura y humedad relativa del aire, en ambos casos se trata del modelo DHT22 (también conocido como AM2302). Para el Arduino Uno, hemos usado una solución que viene instalada sobre una pequeña placa y cuenta con conectores y resistencia de pull-up, mientras que para el Arduino Mega se ha montado un sensor suelto y la resistencia la añadimos nosotros.

En ambas placas de Arduino, el pin de entrada para la lectura de datos es D2. Se conectan a un pin digital pues estos sensores se comunican con un bus 1-wire específico. Este bus no es compatible con el “1-wire bus” de Maxim/Dallas.

Para la correcta lectura de valores existen librerías que permiten simplificar el proceso de captura de datos y arranque del dispositivo y nos ayudan a seleccionar el modelo de sensor elegido, pues existen otras opciones como el DHT11 que cuenta con menor precisión y no mide valores negativos, por lo que ha sido descartado.

El sensor funciona con ciclos de operación de 2s. En este tiempo, el microcontrolador externo y el microcontrolador que lleva integrado el sensor, se hablan entre sí de la siguiente manera:

- El microcontrolador (Arduino) inicia la comunicación.
- El sensor responde estableciendo un nivel bajo y otro alto de 80us.
- El sensor envía 5 bytes.
- Se produce el handshaking.

6.2.3. Sensores niveles de luminosidad

Para la lectura de valores de luminosidad hemos usado dos soluciones distintas en cada caso para realizar pruebas.

Al Arduino UNO le hemos acoplado un sensor LDR (con su correspondiente divisor de tensión) y las lecturas se realizan mediante la entrada analógica A2.

En cuanto al Arduino Mega, el dispositivo es el sensor BH1750 (GY-302) que va montado sobre su propia placa para conectar y cuenta con toda la electrónica necesaria y conectores. La comunicación con el microcontrolador se realiza mediante I2C y la conexión se ha realizado de la siguiente manera:

Pin Bh1750	Conectado a:
Vcc	5V en protoboard
GND	GND
SCL	SCL en Arduino Mega
SDA	SDA en Arduino Mega
ADDR	NC. Lo dejamos libre porque le dejamos la dirección por defecto.

Tabla 7. Conexionado sensor luminosidad

Este último sensor nos proporciona lecturas de mayor rango y además, está pre-calibrado y gracias a la librería, obtenemos los valores deseados sin necesidad de curvas de respuesta.

6.2.4. Detector MQ-2

Igual que en el caso anterior, se ha usado un sensor que viene instalado en una placa, lo que nos facilita su manipulación y el conexionado rápido.

Pin MQ-2	Conectado a:
Vcc	5V en protoboard
GND	GND en protoboard
D0	D3
A0	A1 en Arduino Mega

Tabla 8. Conexionado sensor MQ-2

6.2.5. Entradas analógicas

Para simular el accionamiento de dos entradas analógicas se ha decidido emplear dos potenciómetros en los pines analógicos A2 y A3 del Arduino Mega.

6.2.6. Entradas digitales

Dos pulsadores simulan el accionamiento de una entrada digital conectados a las entradas digitales D5 y D6 del Arduino Mega. Como puede apreciarse en el circuito, cuentan con sus correspondientes resistencias de pull-down.

6.2.7. Comunicación entre las dos placas Arduino

La comunicación para el intercambio de datos entre las placas Arduino UNO y Mega se ha realizado mediante el bus I2C por su sencillez y comodidad de montaje.

El bus I2C se creó como una forma de comunicación sencilla entre componentes que residen en la misma placa de circuito. Cuenta con una velocidad de transmisión de 100Kbits por segundo, aunque puede llegar hasta los 3,4Mhz. La metodología de comunicación de datos del bus I2C es en serie y sincrónica. Se emplean dos líneas, una de reloj (SCL) y otra para datos (SDA), las cuales se conectan a todos los dispositivos que pertenecen al bus [12].

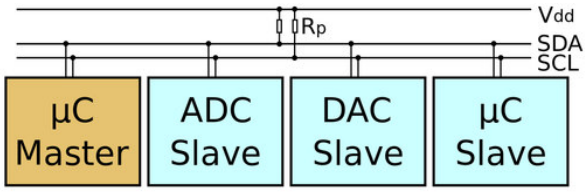

Figura 29. Conexión I2C. Fuente Wikipedia

Como se aprecia en la figura, y según las especificaciones del protocolo, se recomienda añadir resistencias pull-up, aunque también es posible la conexión directa uniendo los pines SDA y SCL de ambas placas, sobre todo, para distancias cortas. También debemos conectar las líneas GND.

A continuación vemos en qué pines deberemos conectar en cada placa.

	Arduino UNO	Arduino Mega
SDA	SDA y A4	SDA y pin nº 20
SCL	SCL y A5	SCL y pin nº 21

Tabla 9. Conexionado bus I2C entre placas Arduino

Como vemos, existen dos parejas de pines en cada placa. Esto es útil si queremos conectar más de dos dispositivos, pues iremos encadenándolos uno tras otro. En nuestro caso, para el Arduino Mega, hemos ocupado unos conectores para la comunicación con otro Arduino y la otra pareja, para el sensor de luminosidad.

Para el correcto funcionamiento del bus, es necesario el uso de la librería “Wire.h”, como veremos más adelante.

6.2.8. Shields (WiFi y Ethernet)

Una de las grandes ventajas de usar shields con Arduino es que no es necesario cablear, sino que se montan apilándolos aprovechando los pines de los conectores que

incorporan. Al contar con una forma similar a la placa Arduino, no surgen dudas a la hora de realizar la conexión pues encajan perfectamente uno sobre otro.

Tanto el shield Ethernet, como en el WiFi, se comunican con la placa Arduino mediante el bus SPI [1] (mediante el header ICSP que incorpora). Es importante tener en cuenta que no podremos usar ciertos pines cuando queramos cablear nuestros circuitos y realizar el programa.

En la siguiente tabla veremos la correspondencia:

Arduino UNO	Arduino Mega	Reservado
11	52	SCK
12	50	MISO
13	51	MOSI
10	10	Selecciona el chip W5100
4	4	Selecciona la tarjeta SD
	53	Reservado por SPI
7*	7*	Handshake WiFi Shield*

Tabla 10. Relación de pines reservados al usar Ethernet y WiFi Shield

* El pin nº 7 actúa de handshake entre el shield WiFi y Arduino, por lo que no deberemos dejarlo libre al usar esta placa.

Es posible apilar más shields o placas que diseñemos pero siempre deberemos tener en cuenta si son compatibles con los demás, respetando los pines de los que hacen uso.

6.2.9. Detectores de movimiento PIR

Se han incluido dos detectores de presencia que activan una entrada digital en caso de movimiento. El dispositivo cuenta con unos selectores giratorios que nos permitirán hacer un ajuste más fino de la sensibilidad de detección y del retardo una vez que los tengamos en su emplazamiento definitivo.

Cada detector cuenta con un pin Vcc por el que se alimenta con 5V y otro de GND para llevarlo a masa. El tercer pin, “OUT” irá conectado a lo pines 30 y 31 de nuestro Arduino Mega.

6.2.10. Control de motor

Para controlar el motor de 12V que simulará un sistema de ventilación y/o climatización, deberemos usar el circuito que veíamos en el apartado 4.5.2. y usaremos uno de los pines digitales de nuestro Arduino Mega configurándolo como salida.

Puesto que se quiere realizar un control regulado, debemos escoger un pin que lo permita, en nuestro caso es el pin digital que viene identificado como “~9” (PWM).

6.3. PROGRAMACIÓN

Como se comentó en el apartado “1.4 Etapas”, la programación de Arduino se ha realizado dividiéndola en bloques para poder entender y depurar cada subsistema con más facilidad. Normalmente, cada sketch se realizaba para controlar un único dispositivo, como pudiera ser el módulo de relés, aunque también se han realizado diversas subversiones de cada uno de ellos cuando se quería probar la interactuación entre dos de ellos o tratar de optimizar el código.

Tras juntar todas las partes, se ha obtenido un código para cada placa de Arduino que se cargará y ejecutará constantemente.

6.3.1. Sketch

Como sabemos, un sketch es el código que programaremos en nuestro microcontrolador para marcarle su funcionamiento.

Como norma, en cada sketch de Arduino, y como no podía ser de otra forma, en los de este trabajo, podremos encontrar estas partes:

- Comentarios. No afectan al código ni se cargan en el microcontrolador, por lo que no ocupan memoria, pero son recomendables para poder entender lo que se ha programado y añadir información útil como dónde se deberán conectar ciertos dispositivos.
- Llamada a librerías y declaración de variables.
- Setup, que se ejecuta una sola vez y es donde podemos inicializar variables, arrancar dispositivos y cualquier otra tarea que suponga una única configuración.
- Loop. En esta parte, el bucle, es donde se realiza la mayor parte de la programación pues es lo que realmente ejecutará el microcontrolador, una y otra vez.

Aunque el loop sea la parte donde se ejecuten las órdenes que definimos para el funcionamiento del sistema, es posible el uso de funciones como en la mayoría de lenguajes de programación. Éstas se crean fuera del bucle, pero al ser llamadas dentro del Loop se ejecutan también de forma ininterrumpida.

Como hemos indicado, cada placa Arduino cuenta con una programación específica pues cumple con una misión determinada del mismo modo que lo harían los distintos autómatas o módulos PLC de una instalación.

Los sketch de programación de cada placa se adjuntan completos (con comentarios) en la carpetas de “Ficheros anexos”. Por su extensión, se ha preferido no incluirlos en esta memoria pero sí se nombrarán las principales funciones y librerías utilizadas.

a) Sketch Ardunio UNO + WiFi Shield

El sketch de esta placa se puede encontrar en el archivo “ArduUno_Xively.ino” incluido en la carpeta “Ficheros anexos/código/Sketch” y su programación consigue que el microcontrolador realice las siguientes tareas:

- Activar la comunicación con el WiFi shield, el sensor de temperatura y configurar los pines como salida para los relés y como entrada, para el sensor LDR.
- Conectar el dispositivo a la red local mediante protocolo 802.11b/g (WiFi). Para ello se deben definir todos los parámetros de la red como el SSID, el password de la red, la dirección IP que queremos fijar (se podría hacer también por DHCP), el servidor DNS, la puerta de enlace y la subred.
- Recibir y tratar los datos enviados por la placa Arduino Mega a través del bus I2C.
- Crear un cliente web que realice peticiones a un servidor remoto.
- Establecer una comunicación con el servidor de Xively y subir los datos registrados por los sensores del controlador, así como los que llegan a través del bus desde el otro nodo.
- Accionar los circuitos de los relés para el encendido en caso de disminuir los niveles de iluminación por debajo de un umbral.

Para que podamos enviar y recibir datos del servidor de Xively, debemos configurar otros parámetros que nos facilita la página, como son la “API-key, el nº de “feed” y tener cuidado de nombrar las variables con el mismo identificador que configuremos los “datastreams” en la página. Todo esto se explicará más adelante.

Para el correcto funcionamiento del sistema es necesario el uso de librerías. A continuación se enumeran las utilizadas en este sketch y se deja para el apartado “Librerías” la explicación de cada una de ellas:

Figura 30. Imagen de librerías del sketch del nodo secundario

Además del uso de las funciones propias de cada librería, se han creado unas funciones para conseguir el efecto deseado que serán llamadas desde el loop:

- *getTemperature*: nos devuelve los valores de lectura de temperatura del sensor DHT22 en formato *float* y también los muestra en el Monitor Serial.
- *getHumidity:* igual que el anterior, pero en este caso devuelve humedad.
- *getLight:* nos muestra los valores de luminosidad registrados por el sensor LDR.
- *printWiFiStatus():* nos informa de los parámetros de la conexión WiFi.

- *receiveEvent():* permite extraer varios valores en formato *float* tras establecerse la recepción a través del bus I2C. Esta función se ha creado a partir de las recomendaciones del foro de Arduino [1].

Este sketch se ha programado partiendo de la información facilitada en los ejemplos de la web de Xively para poder transmitir información a su plataforma [18]. A partir de ahí, se han agregado elementos y configuraciones adicionales.

b) Sketch Arduino Mega + Ethernet Shield:

El código de este nodo se encuentra en la carpeta *“Ficheros anexos/código/Sketch”* bajo el nombre “*ArduMegaEthernet.ino*”.

Igual que en el caso anterior, a continuación se resumen las principales características de su funcionamiento:

- Activa la comunicación de la placa Ethernet Shield con la placa Arduino Mega. También arranca la transmisión mediante bus I2C, empleado por el sensor de luminosidad BH1750 y el intercambio de datos entre nodos.
- Activa comunicaciones con el sensor de temperatura DHT22 y configura los pines como salida para los ocho relés y el motor, y como entrada, para los dos detectores PIR, los pulsadores, las entradas analógicas y el sensor de gas.
- Conecta el dispositivo a la red local mediante su puerto Ethernet, previa activación de la placa. Para ello se introducen todos los parámetros de la red: dirección IP que queremos fijar (se podría hacer también por DHCP), MAC de la placa Ethernet y puerto del servidor (en nuestro caso, el 80).
- Transmite datos de los sensores a la placa Arduino UNO a través del bus I2C para que sean enviados a Xively.
- Acciona los circuitos de los relés para el encendido mediante la aplicación web.
- Arranca un servidor web que publica una página alojada en la tarjeta microSD que se inserta en el lector del shield.
- Activa el motor en caso de que los niveles de gas superen un umbral prefijado.
- Escucha peticiones de clientes web.

En la siguiente imagen podremos ver las librerías que se han empleado:

```
#include <SPI.h>
#include <Ethernet.h>
#include <SD.h>
#include <DHT.h>
#include <Wire.h>
#include <BH1750.h>
#include "I2C_Anything.h"
```

Figura 31. Imagen de librerías del sketch del nodo principal

Las funciones que se han programado para este sketch son:

- *getTemperature*: devuelve los valores de temperatura del sensor DHT22 en formato *float* y también los muestra en el Monitor Serial.
- *getHumidity:* igual que el anterior, pero en este caso devuelve humedad.
- *getLight:* nos muestra los valores de luminosidad registrados por el sensor Bh1750.
- *getGasValue:* devuelve el valor de los niveles de gas registrados por el sensor MQ-2.
- *listenForEthernetClients:* esta función escucha y si recibe una petición de un cliente a través de una petición HTTP, cargará la página en el navegador.
- *SetRelays:* actualiza valores de los estados de los relés.
- *XmL_response:* crea y envía el fichero XML con los valores de los relés, sensores, entradas analógicas y digitales. Esto es necesario por haber creado nuestra página web en la tarjeta SD.
- *StrClear:* limpia los arrays.
- *getPIR*: informa de la detección de presencia.
- *motorControl:* controla el funcionamiento del motor mediante una salida PWM cuando el valor de gas supera un umbral. La velocidad es controlada a partir del valor de una de las entradas analógicas.

Como se ha comentado con anterioridad, la página web se ha alojado en una tarjeta microSD y se produce un intercambio de información mediante AJAX a través de un archivo XML. Podemos ver información adicional sobre esta técnica más aclaraciones en el portal “Starting Electronics” [15] en el que se ha basado esta solución. Existe otra opción más sencilla que consiste en crear la página directamente desde el sketch que se cargará en Arduino.

La primera solución presenta una serie de ventajas como, por ejemplo, la posibilidad de almacenar mucha información en la tarjeta de memoria, incluyendo archivos de estilo CSS, fotos, otras páginas; mientras que en Arduino no hay una memoria que podamos usar a tal efecto, sino que para insertar cualquier etiqueta o campo HTML tendríamos que generar una línea de código con un “client.print” como veremos en la siguiente imagen:

```
cliente.println("<HR>");      // Líneas de separación
cliente.println("<FONT SIZE='+2' COLOR='black' FACE='Arial'> TEMPERATURA Y HUMEDAD:  </FONT>"); //Título
cliente.println("<BR>");
cliente.print("Temperatura: ");
cliente.print(t);
cliente.print(" C");
cliente.println("<BR>");          //Espacio
cliente.print("Humedad:  "); //Título
cliente.print(h);
cliente.print(" %");
```

Figura 32. Ejemplo programación con “client.print”

Vemos que no es una buena solución para una página web compleja, pues todo ese código solo sirve para mostrarnos los dos valores de temperatura y humedad y un título. Si quisiésemos que se muestre con un mejor estilo visual, tendríamos que añadir

muchas más líneas. Esto incrementaría notablemente el tamaño del sketch, pudiendo llegar a quedarnos sin memoria.

En cambio, guardando la página web en la tarjeta microSD podemos editar los archivos HTML, los estilos (CSS) desde nuestro editor habitual en nuestro PC y cuando el resultado es el deseado, solo tenemos que copiar los archivos en la tarjeta y arrancar el equipo. El sketch seguirá siendo el mismo, pero podremos cambiar el aspecto de la página.

Además, de esta forma, podemos programar funciones básicas al microcontrolador y luego, mediante aplicaciones o scripts, añadir funcionalidades amigables al usuario como pasaría con un PLC y un SCADA.

6.3.2. Librerías

Como se ha comentado en el apartado 5.3 las librerías nos permiten ampliar las capacidades del entorno Arduino para interactuar con otro hardware o manipular datos. Para conseguir los objetivos del proyecto ha sido necesario el uso de varias de ellas.

En la siguiente tabla se detalla el listado de las utilizadas y el motivo su elección. En la web oficial de Arduino [1] podremos encontrar y descargar la mayoría de ellas, a excepción de la última (I2C_Anything.h) que se creó ad-hoc y se incluirá en la carpeta de anexos. En el IDE de Arduino ya vienen instaladas muchas de ellas.

Librería	Aplicación	Arduino
DHT.h:	Integración con los sensores de temperatura y humedad DHT11 y DHT22.	UNO y Mega
Ethernet.h:	Conexión del shield Ethernet a internet. Permite aceptar conexiones entrantes y salientes.	Mega
HttpClient.h	Permite conectar con servidores. En nuestro caso, se emplea para conectar con Xively.	UNO
SD.h	Permite la lectura y escritura en la tarjeta SD.	Mega
SPI.h	Gestiona la comunicación con dispositivos SPI (Serial Periferical Interface). Se usan para la comunicación entre los shields y las placas Arduino.	UNO y Mega
WiFi.h	Conexión del shield WiFi a internet. Permite aceptar conexiones entrantes y salientes.	UNO
Xively.h	Facilita la comunicación con Xively	UNO
Wire.h:	Permite la comunicación con dispositivos I2C/TWI. En nuestro caso se emplea para la comunicación entre nodos y para la lectura de datos del sensor BH1750.	UNO y Mega
I2C_Anything.h	Se ha creado a partir de los consejos del foro de Arduino para permitir crear un “template” que nos ayude a enviar los datos a través del bus. Se realiza un procesado en origen y destino para salvar la limitación de tener que enviar datos del tipo *chart*.	UNO y Mega

Tabla 11. Relación de librerías empleadas y descripción

6.3.3. Entorno Web del servidor

En el apartado 6.1.4. “Interfaz de control” veíamos las distintas pantallas que conformaban el entorno web alojado en el servidor instalado de nuestra Raspberry Pi y su funcionamiento.

Para el correcto funcionamiento de la placa, primero hemos instalado el sistema operativo y activado la conexión SSH. Posteriormente se han configurado los diversos parámetros de red y se ha dispuesto un servidor LAMP que cuenta con Apache, MySQL y PHP (sobre Linux) y por último un servidor FTP en el que alojaremos nuestras páginas web y demás archivos.

La explicación detallada de este proceso no se incluye en esta memoria pues existen tutoriales y vídeos explicativos en la web de la Fundación Raspberry Pi [13], además de foros y documentación.

La web que hemos diseñado debe ser capaz de comunicarse con el servidor de Xively para solicitarle información y con la web que publica Arduino Mega. Como se ha comentado, la programación se ha realizado mediante HTML5, CSS3 y JavaScript.

Por ejemplo, para el caso de la pantalla “R.D. 1826/2009 Nov.” hacemos uso de unas librerías Javascript de Xively [19] y de un script que toma los datos que le solicitemos. También ha sido necesario el uso de las librerías jQuery para el correcto funcionamiento de los distintos scripts.

En este caso solo estamos utilizando dos valores para representarlos numéricamente pero el uso de la API de Xively nos permite solicitar y subir información de muchas maneras e incluso pedir históricos para representar nuestras propias gráficas.

En el caso de la pantalla “Home”, las librerías utilizadas para la localización son las “gmaps.js”, una versión reducida de las librerías de Google Maps para JavaScript que permiten un uso ligero y muy simplificado [7].

6.3.4. Configuración Xively

Aunque existe un tutorial que nos facilita Xively para conectar nuestro Arduino, vamos a repasar, de manera resumida, cómo se ha configurado en nuestro caso:

- En primer lugar, es necesario registrarse para obtener una cuenta personal en el servicio accediendo a la web de Xively en https://xively.com. Las cuentas personales son gratuitas y permiten multitud de aplicaciones. Las profesionales son de pago y cuentan con servicios avanzados, pero para el objetivo de este proyecto (e incluso mayores) es suficiente con la primera.

- Una vez tengamos nuestra cuenta, nos vamos al apartado “Develop” desde donde podremos gestionar nuestros prototipos. Aquí crearemos un dispositivo nuevo pulsando “Add device” donde le asignaremos un nombre y cierta información relevante.

- Una vez creado, ya podremos entrar en la ventana de nuestro dispositivo para realizar los ajustes y donde veremos las gráficas y otros datos.

- Aquí tendremos que buscar el “Feed Id” y la “API-key”, que son dos números o códigos que nos identifican para realizar la conexión entre el servidor y Arduino y que deberemos incluir en las correspondientes variables de nuestro sketch. Estos valores se eliminarán y serán sustituidos por “API-key here” o “feed-id here” en los archivos que se entreguen en los Ficheros anexos del CD.

- Pulsando sobre el botón “Add Channel” iremos añadiendo canales en los que recibir información. Al igual que con el Feed-id y la API-key, deberemos prestar cuidado de copiar los nombres tal cual los hemos declarado en nuestro sketch. En cualquier caso, si todo lo anterior se ha hecho correctamente y el Arduino es capaz de enviar la petición correctamente a Xively, a partir de los campos del archivo JSON, nos podrían crear los canales de forma automática.

En la página también podremos completar otra información como la localización del dispositivo por coordenadas en Google Maps, metadata con descripción del proyecto y su creador.

Existen tres apartados más que pueden resultarnos de mucha utilidad:

- Request Log, en el que podemos ver la llegada de las peticiones (Put, Get, Post) y pinchando sobre ellas, ver su contenido.

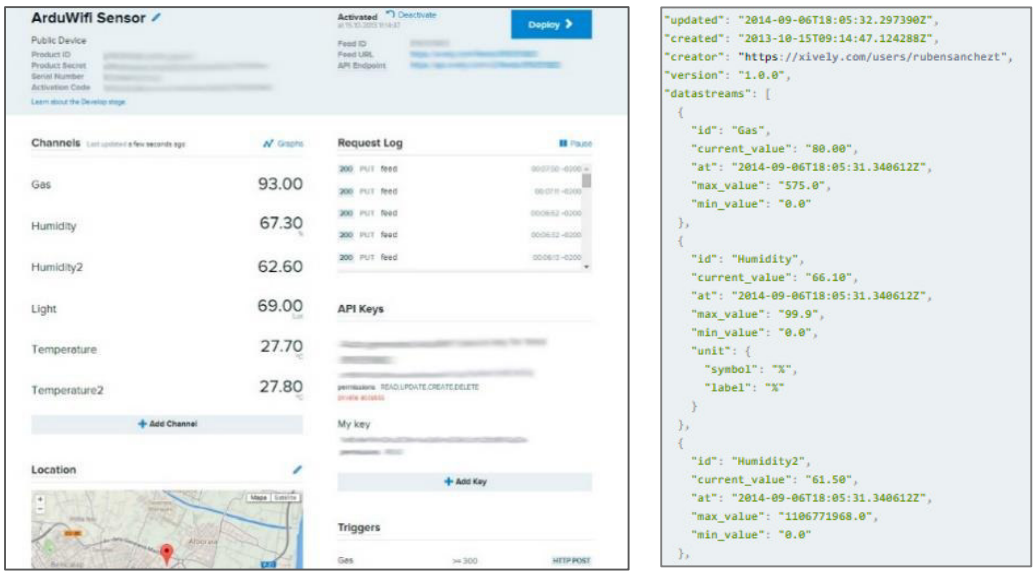

Figura 33.Vista pantalla Develop en Xively y estructura de datos JSON

En la anterior imagen podemos ver la pantalla de control en la que se observa el listado de peticiones y a la derecha, parte del cuerpo del “request” con información estructurada de los distintos *datastreams*.

- API Keys, donde podremos crear otras keys y administrar los permisos de lectura, escritura, actualización y borrado de las mismas. Podemos crear claves públicas y privadas.
- Triggers es un apartado de mucha utilidad pues, como su nombre indica, nos permite configurar el disparo de la ejecución de un proceso en función de multitud de parámetros de nuestros datos.

Esta última funcionalidad se ha empleado en el proyecto para generar un aviso en caso de que el nivel de gas supere un determinado valor. Esto es una aplicación muy común y demandada en los sistemas de automatización de edificios pues el responsable de seguridad o mantenimiento tiene que estar informado de las incidencias y tiene que quedar un registro.

Además, no siempre se puede estar ante el monitor comprobando el estado de los sistemas, por lo que es deseable que si se genera un evento, este se transmita de alguna manera.

En nuestro caso hemos hecho uso de la aplicación Zapier que permite enlazar los triggers de Xively con otras aplicaciones. Hemos optado por el envío inmediato de un email a una cuenta con información del desencadenante. También se podrían haber enviado avisos por SMS, Twitter o mensajes por Google Talk.

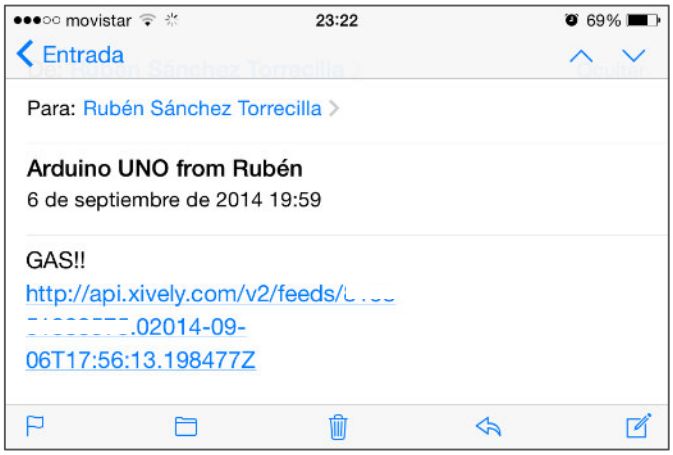

Figura 34. Mensaje de alerta generado por un trigger

En la anterior imagen podemos ver un email que se generó tras subir los niveles de gas al aproximar un mechero al sensor MQ-2.

Existen multitud de formas de configurar la respuesta de los triggers y también es posible configurar en Zapier la información que queremos recibir.

6.4. PUESTA EN MARCHA

Siguiendo los pasos de montaje y programación anteriores, el sistema estaría listo para funcionar con solo proporcionar la alimentación y añadir en cada sketch las claves de acceso necesarias, pues éstas han sido eliminadas de los archivos grabados en el CD.

Así, para el caso del sketch ArduUNO_Xively.ino necesitaremos configurar el SSID y contraseña de nuestra WiFi y por otro lado, el API-Key y número de feed proporcionado por Xively.

En ArduMegaEthernet.ino deberemos proporcionar la MAC que viene impresa en el Ethernet Shield que empleemos.

Para poder acceder de forma remota al servidor y así controlarlo desde cualquier equipo que se encuentre fuera de la red local, es necesario contar con una IP pública fija facilitada por nuestro proveedor de servicios de internet (ISP) y si esto no es posible (dado su elevado coste), podemos usar un servicio DDNS (Dynamic Domain Name System) como podría ser DynDNS, No-IP y otros muchos.

También será necesario realizar una correcta configuración de nuestro router para direccionar los puertos (“port fowarding”) y prestar atención al firewall por si bloquea las conexiones.

7. PRUEBAS, PROBLEMAS Y SOLUCIONES

A continuación veremos los diversos pasos y pruebas que se han ido realizando para comprobar que el sistema se comportaba de la forma que buscábamos.

Durante el proceso surgieron problemas a los que hubo que ir buscando su solución, como veremos.

7.1. PRUEBAS

Para la realización del proyecto ha habido que realizar multitud de pruebas pues la integración de tantos dispositivos y tecnologías aumentaban la probabilidad de fallo.

Cada vez que se pretendía añadir un sensor, actuador o funcionalidad, se ha procedido de la siguiente manera:

1. Montaje del dispositivo y conexionado a una placa Arduino independiente.
2. Escritura y carga de un sketch exclusivo para ese elemento o función concreta.
3. Pruebas de funcionamiento por separado viendo los resultados y ayudándonos del Monitor Serie y los mensajes que hemos configurado en puntos estratégicos para depurar errores.
4. Cuando el resultado era el esperado, se pasaba a conectarlo junto con el resto de los dispositivos.
5. Se modifica el sketch global añadiendo las partes necesarias del fragmento de código que acabamos de crear.
6. Se verifica visualmente que no se había alterado ninguno de los otros circuitos, ni existían incompatibilidad entre ellos.
7. Se revisa en el sketch que no estemos utilizando una variable ya existente y que se ha colocado en un punto correcto para su ejecución.
8. Se compila y carga el sketch y se pone en funcionamiento el dispositivo.
9. Se vuelve a chequear de forma, tanto visual, como por el Monitor el correcto funcionamiento.
10. Se deja funcionando cierto tiempo para ver si aparecen fallos.

En algunos casos existían interacciones entre varios elementos, como por ejemplo la comunicación entre las dos placas Arduino y el bus, el envío de un dato desde un Arduino a Xively pasando a través de otra placa, el control de los relés desde un entorno web… que han convertido el proceso en un poco más tedioso y ha supuesto ir con mucho cuidado, fijando más puntos de depuración, pues si no se complicaría mucho localizar el fallo a posteriori.

7.2. PROBLEMAS Y SOLUCIONES

Por su componente práctica, a lo largo de la realización del trabajo se ha ido encontrando algún escollo al poner en funcionamiento los distintos dispositivos que en un principio, de manera teórica, debían funcionar sin problemas.

En la mayoría de los casos, tras un periodo de análisis, se ha conseguido encontrar y solucionar el fallo, aunque en otras ocasiones se ha recurrido a la experiencia de la comunidad de “makers” a través del foro oficial y otros alternativos.

Debido a que estamos hablando de un microcontrolador que, aunque muy versátil, cuenta con limitaciones, en algunos casos, la solución no consistía en una aclaración de un experto, sino en buscar una alternativa. A continuación se detallan algunos problemas surgidos:

- ✓ Problemas de conexión del WiFi shield. Solucionado mediante la actualización del firmware. Para ello se actualizan los microcontroladores AT32UC3 y HDG104 mediante el uso del software Flip de la marca Atmel.

- ✓ A lo largo de toda la programación han surgido infinidad de comportamientos no esperados del sistema al unir los diversos fragmentos de código al global. Tras analizar con detenimiento, se observaron fallos al declarar y utilizar algunas variables o algún uso indebido de funciones. Se resolvió con facilidad realizando continuas revisiones de la programación y depurando código gracias a los puntos de *debbug* que se han ido incluyendo y que nos mostraban salidas en el Monitor Serial.

- ✓ El envío de datos mediante el enlace de los dos Arduinos se realiza mediante el tipo de dato “char”, pero las lecturas eran del tipo “int” o “float”. Para poder recibir los datos de manera ordenada y en el tipo correcto, se ha creado una librería con un “template” para tratar los datos.

- ✓ Aparecen problemas de memoria al usar la placa Arduino UNO pues el sketch sobrepasa la capacidad con que cuenta. Al principio se reducen funcionalidades pero luego se opta por añadir una segunda placa (Arduino Mega) que cuente con más capacidad y de paso, repartir la carga de las tareas.

- ✓ Las pruebas de encendido y apagado de circuitos se realizaron con Leds pero al cambiar a los módulos de relés, el funcionamiento era el inverso. Esto es porque vienen configuradas para funcionar en modo LOW, por lo que se solucionó cambiándolo en el sketch.

8. CONCLUSIONES Y TRABAJOS FUTUROS

La realización de este trabajo por su componente teórico-práctica me ha sido de mucha utilidad para poder aprender y consolidar el uso de diversas tecnologías.

Durante la investigación y las diversas pruebas, he podido comprobar otras funcionalidades e incluso llegar a sobrepasar las capacidades del sistema Arduino.

Como se pretendía, se ha creado un sistema basado en Arduino que permite la monitorización remota y en tiempo real de sensores y tras realizar las diversas pruebas, se ha podido comprobar su correcto funcionamiento que podría extrapolarse a un edificio real.

Todo el proceso se ha realizado siguiendo las etapas y cumpliendo los objetivos marcados al inicio del documento, además de haber añadido las funcionalidades y mejoras que se detallan a continuación:

- Añadida una segunda placa Arduino para incrementar las funcionalidades como el control de circuitos eléctricos y repartir procesos.

- Comunicación entre las placas Arduino para el intercambio de información, simulando un sistema basado en bus.

- Además de la página web que publica Arduino, se ha creado una pequeña aplicación web alojada en un servidor local que da acceso a distintas visualizaciones y controles de los elementos.

- Instalación de un servidor LAMP (Linux-Apache-MySQL-PHP) en un PC para alojar la aplicación web local.

- Empleo de una placa Raspberry Pi para la instalación del servidor LAMP. Este paso incluye el aprendizaje, instalación del sistema operativo (Linux) y diversos paquetes de software.

- Creación de una página web que muestra valores locales de temperatura y humedad relativa en tiempo real para ser visualizada desde una pantalla o monitor en locales de pública concurrencia en cumplimiento del Real Decreto 1826/2009 de noviembre.

- Se activa un servicio que permite el envío de avisos a través de un “trigger” cuando un valor de los registrados supera un umbral.

Es por ello, que se considera que se ha cumplido satisfactoriamente el objetivo global del trabajo, así como los objetivos secundarios llegándose a aportar muchas mejoras.

No se han querido incluir otras funcionalidades en la memoria que, aunque, actualmente funcionan de forma independiente, no están completamente incorporadas

a la solución global por producir algún fallo o comportamientos inesperados. A continuación se comentan dichas mejoras:

- Se ha añadido una tercera placa Arduino para simular un termostato. La placa es el modelo Nano que cuenta con unas dimensiones muy reducidas (1,85 x 4,39 cm), pero unas capacidades en cuanto a procesador, memoria y entradas/salidas, análogas al Arduino UNO, lo que lo convierten en el candidato ideal para introducirlo dentro de un aparato, dotándolo de conectividad.

  Una buena opción sería añadirle una placa que lo dote de WiFi o Ethernet, pero en nuestro caso se quiere comunicar por bus y pasar los valores a Arduino Mega para que sea este último el que muestre la información en un navegador vía web.

  Como se ha comentado antes, no se ha incorporado a la versión final porque la parte web que controla Arduino aún no está desarrollada pero podría servir para trabajos futuros.

- Se ha querido medir consumos eléctricos y almacenarlos en Xively. Se buscó hacer una solución real haciendo uso de un medidor de energía trifásica con comunicación Modbus con el que se contaba, pero era un préstamo y no se ha finalizado correctamente la parte de adquisición de datos.

  Mediante un circuito montado con un MAX485 (y resistencias de fin de línea) y el uso de las librerías SimpleModbusMaster, es posible realizar la lectura mediante el puerto serie de los distintos “coils”. No se llegó a tratar la información convenientemente para poderla enviar a Xively.

- Instalación de un módulo RTC que mantiene un calendario interno y nos sirve para ejecutar ciertas acciones programadas mediante un horario.

Se han investigado muchos otros dispositivos y aplicaciones que no se han incluido en esta memoria debido a que había que finalizar TFG en algún momento pues las posibilidades son tan grandes que se podría haber prolongado *ad eternum*.

Como trabajos futuros para ampliar el sistema, se propone:

- ✓ Una ampliación que me parece muy interesante es la incorporación de módulos XBee para la transmisión de datos entre placas Arduino e incluso crear una red de dispositivos inalámbricos compuesta por sensores y nodos.

- ✓ Construir un Arduino mínimo. Se puede construir una placa Arduino simplificada con un coste bajísimo, usando únicamente el microcontrolador y unos cuantos elementos electrónicos (resistencias, condensadores y cristal). Dicha placa podría emplearse para funciones muy específicas y así liberar de ellas a otro Arduino. Un ejemplo sería incorporarlo en viejos termostatos y dotarlos de comunicación.

- ✓ Ampliar la red de Arduinos comunicados por bus.

- ✓ Usar la placa Raspberry Pi para alojar los datos en la BBDD local. Para ello habrá que configurar la base de datos y crear las consultas adecuadas.
- ✓ Integrar un módulo RS-485 y Modbus para controlar termostatos y analizadores eléctricos con este protocolo.
- ✓ Emplear sensores de humedad y lluvia para controlar el riego de zonas ajardinadas de forma más eficiente.

ANEXO I - PRESUPUESTO

Se adjunta el presupuesto del material utilizado para el montaje del prototipo. No se añaden otros conceptos como horas de mano de obra o beneficio industrial por tratarse de un trabajo académico.

Item.	Unidades	Precio/ud.	Subtotales
Arduino UNO R3	1	20,00 €	20,00 €
Arduino Mega R3	1	41,00 €	41,00 €
Arduino Ethernet Shield R3	1	32,00 €	32,00 €
Arduino Wifi Shield R3	1	64,99 €	64,99 €
Módulo 8 relés	1	4,64 €	4,64 €
Módulo 4 relés	1	2,62 €	2,62 €
Mini sensor PIR	2	1,51 €	3,02 €
Sensor MQ-2	1	5,85 €	5,85 €
Sensor BH1750	1	1,99 €	1,99 €
DHT22	1	4,25 €	4,25 €
DHT22 (placa)	1	5,42 €	5,42 €
LDR	1	1,17 €	1,17 €
Pot. Lineal 10K ohmios	2	2,20 €	4,40 €
Pulsadores	2	0,19 €	0,38 €
Protoboard 1	1	4,55 €	4,55 €
Protoboard 2	1	13,30 €	13,30 €
Cable color conexiones	4	1,03 €	4,12 €
Resistencias distintos valores	1	1,15 €	1,15 €
Tira conectores macho/hembra	1	2,04 €	2,04 €
Transistor NPN Tip120	1	0,37 €	0,37 €
Cable UTP Cat5e	3	0,45 €	1,35 €
Tarjeta SD 8Gb	1	7,95 €	7,95 €
Raspberry Pi (model B)	1	26,05 €	26,05 €
Router TP-Link 3G/4G Wireless N	1	24,90 €	24,90 €
Ventilador PC 12V	1	4,95 €	4,95 €
	Total base imp.		282,46 €
	IVA	21%	59,32 €
	TOTAL MATERIALES		**341,78 €**

ANEXO II - LISTADO DE FICHEROS ANEXOS INCLUIDOS:

\Ficheros anexos\código\Librerías
- BH1750_master.rar
- Humidity_Temperature_Sensor
- I2C_Anything.rar
- XIVELY.rar

\Ficheros anexos\código\Sketch
\Ficheros anexos\Código\Sketch\ArduMegaEthernet\
- ArduMegaEthernet.ino

\Ficheros anexos\Código\Sketch\ArduUno_Xively\
- ArduUno_Xively.ino

\Ficheros anexos\Código\Web\BMS Rasp Pi
- control.html
- digital-7.ttf
- gmaps.js
- home.html
- index.html
- mimapa2.js
- RD1826_09.html
- style.css
- styleRD.css

\Ficheros anexos\código\Web\SD\
- index.htm

\Ficheros anexos\datasheet
- Arduino Ethernet Shield.pdf
- Arduino MEGA2560.pdf
- Arduino WiFi Shield.pdf
- ATmega2560.pdf
- ATMega328.pdf
- BH1750 (sin placa).pdf
- BH1750 SCH.pdf
- bh1750fvi-e.pdf
- DHT22.pdf
- Módulo 8 relés.pdf
- MQ-2 (sin placa).pdf
- MQ2.pdf
- PIR.pdf

\Ficheros anexos\datasheet_links
- Arduino – ArduinoEthernetShield.website
- Arduino - Ethernet Shield.
- Arduino - SD.website
- Arduino - SDCardNotes.website
- Arduino - WiFiShieldFirmware

\Ficheros anexos\varios\
- Firmware WiFi Shield.rar

LISTADO DE FIGURAS

LISTADO DE TABLAS

BIBLIOGRAFÍA Y REFERENCIAS

[1] ARDUINO. Arduino Oficial Website <http://www.arduino.cc/>

[2] BOE.es – Boletín Oficial del Estado <http://www.boe.es/buscar/doc.php?id=BOE-A-2009-19915>

[3] CASADOMO – Todo sobre Edificios Inteligentes <http://www.casadomo.com>

[4] DOUKAS, C. (2012). *Building Internet of Things with The Arduino: Arduino V.10 Ready!, Covers: communication with wired and wireless networks, android communication, cloud communication and more!*. S.L.: CreateSpace.

[5] ESEFICIENCIA– Todo sobre Eficiencia Energética <http://www.eseficiencia.es/>

[6] EVANS, B. W.(2011). *Arduino programming notebook, edición Española ver.1.2*. Ardumanía.

[7] GMAPS.JS – GOOGLE MAPS API WITH LESS PAIN AND MORE FUN. <http://hpneo.github.io/gmaps/>

[8] ICONICS – HMI/SCADA SOFTWARE. < http://www.iconics.com/Home.aspx>

[9] INTERNET OF EVERYTHING - <http://internetofeverything.cisco.com/es>

[10] JQUERY API DOCUMENTATION - <http://api.jquery.com/>

[11] KNX ASSOCIATION. OFFICIAL WEBSITE. <http://www.knx.org/knx-en/index.php>

[12] MARGOLIS, M. (2012). *Arduino Cookbook, Second Edition.* Sebastopol: O'Reilly Media, Inc.

[13] RASPBERRY PI FOUNDATION - <http://www.raspberrypi.org/>

[14] SAUTER IBERICA. Competencias básicas. Integración a nivel de gestión. <http://www.sauteriberica.com/es/competencias-basicas/sistemas/tecnologias/integracion-de-protocolos/nivel-de-gestion.html>

[15] STARTING ELECTRONICS. Electronics for Begginers and Beyond. <http://startingelectronics.com/>

[16] TREVENNOR, A. (2012). *Practical AVR Microcontrollers: games, gadgets, and home automation with the microcontroller used in Arduino.* New York: Springer Science+Business Media.

[17] UCKELMANN, D., HARRISON, M., MICHAHELLES, F. EDITORS (2011). *Architecting the Internet of Things with a foreword by Bernd Scholz-Reiter*. Berlin, Heidelberg: Springer-Verlag

[18] XIVELY BY LOG ME IN. <https://xively.com/>

[19] XIVELYJS - Tutorial - Xively Javascript Library - Pete Correia <http://xively.github.io/xively-js/tutorial/>