

# **CPD inteligente y autónomo de la Biblioteca Virtual del Sistema Sanitario Público de Andalucía (BV-SSPA). Domotización con hardware abierto: Arduino & Shields**

José M Carrión-Pérez<sup>1</sup>; Antonio Correa Romero<sup>1</sup>; Luis J Borrego-Lopez<sup>1</sup>; Verónica Juan-Quilis<sup>1</sup>

1. Biblioteca Virtual del Sistema Sanitario Público de Andalucía. Consejería de Salud de la Junta de Andalucía.

[jmcarrion@bvsspa.es](mailto:jmcarrion@bvsspa.es); [acorrea@bvsspa.es](mailto:acorrea@bvsspa.es) ;[ljbtorrego@bvsspa.es](mailto:ljbtorrego@bvsspa.es)

## 1. INTRODUCCIÓN

El control y monitorización de un CPD puede suponer una serie de ventajas considerables para el personal encargado de su mantenimiento y vigilancia 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 CPD y, con una adecuada integración de las mismas, se puede conseguir un armario informático 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 CPD 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 del CPD de la Biblioteca Virtual del Sistema Sanitario Público de Andalucía y dotarlo de cierta inteligencia.

Por nuestra trayectoria profesional, hemos tenido oportunidad de conocer diversos sistemas de control para grandes edificios. 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, aplicándolo a un CPD.

Hemos realizado un trabajo teórico-práctico, por lo que se va a diseñar el sistema de forma teórica, con el fin de documentar el proyecto, pero también se realizará la programación y montaje sobre un prototipo para poder comprobar su funcionamiento, previa simulación con **Electronics WorkBench Multisim** (<http://www.ni.com/multisim/whatis/esa/>) y el programa fritzing<sup>1</sup> (<http://fritzing.org/home/>), para pasar a placa los circuitos hechos sobre protoboard.

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

#### 1.1.1 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.

---

<sup>1</sup> **Fritzing** es un programa de automatización de diseño electrónico libre que busca ayudar a diseñadores para que puedan pasar de prototipos (usando, por ejemplo, placas de pruebas) a productos finales.

Fritzing fue creado bajo los principios de [Processing](#) y [Arduino](#), y permite a investigadores y aficionados documentar sus prototipos basados en Arduino y crear esquemas de circuitos impresos para su posterior fabricación. Además, cuenta con un sitio web complementario que ayuda a compartir experiencias y a reducir los costos de fabricación.

### 1.1.2 Objetivos secundarios

- 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(1).
- Programación del sistema microcontrolador basado en Arduino.
- Adaptación de los diferentes sensores al entorno del CPD.
- Permitir accionamientos de relés para el control remoto de circuitos.
- Montaje del sistema y comprobación del dispositivo y su comunicación con la plataforma de almacenamiento de datos.

El proyecto se ha ejecutado siguiendo un plan de trabajo, en función de los problemas a resolver, por el nuevo e inadecuado emplazamiento del CPD de la Biblioteca Virtual del Sistema Sanitario de Andalucía (en adelante BV-SSPA) . 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 Búsqueda bibliográfica y estudio del estado del arte de los sistemas Arduino.
2. Elección del hardware (placas Arduino, sensores, shields).
3. Aprendizaje del lenguaje y forma de programación con el IDE de Arduino.
4. Programación, montaje de circuitos y prueba de los mismos en Arduino. Este paso se ha simulado (Electronic Workbench) para garantizar el correcto funcionamiento de cada circuito o shields (control de relés, lectura de temperatura y humedad,...), antes de pasarlo a placa.
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 circuito para que funcionen en un único programa.
8. Control de circuitos eléctricos.

### 1.1.3 Mejoras en via de desarrollo

1. 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.
2. Crear una pequeña aplicación web alojada en un servidor local que de acceso a distintas visualizaciones y diseño de interfaz de usuario para controles de los elementos. (Matlab).

3. 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.

## 1.2 Antecedentes y estado de la cuestión

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 el CPD de la BV-SSPA ... aunque las soluciones que se plantearán, también tendrían cabida en una vivienda, local o negocio, etc.

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.

Los sistemas de gestión automatizados y de telecontrol, 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.

Y en este proyecto, basado todo en Hardware Abierto, se pretende conseguir el equivalente a un BMS (Building Management System), formado con sus sensores, actuadores, pasarelas y programa (interfaz hombre-máquina, HMI), en versión web y para Smartphone, que facilite la gestión remota del CPD(2)



Figura 1.2 Interfaz Hombre Maquina. Fuente: <http://2.bp.blogspot.com/-RZF3VwQGZ2o/Vci-9rCofFI/AAAAAAAAAs/FDRNqfqgYjg/s1600/3.jpg>

### 1.3 El internet de las cosas

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).

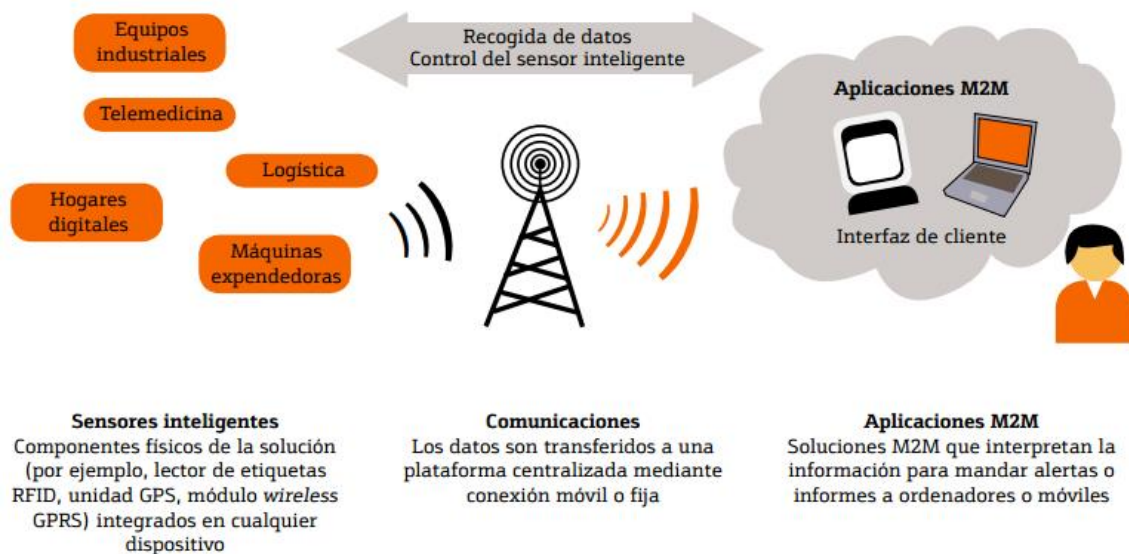
La llegada de IPv6 supone un factor clave en el desarrollo del concepto IoT. Gracias a este nuevo protocolo (diseñado para reemplazar a IPv4) se evitará que el crecimiento de Internet quede restringido, y hará posible la gestión de direccionamiento de innumerables dispositivos.

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....

Por otro lado, ya son muchas las aplicaciones móviles y servicios en la nube que nos permiten la conexión a todos estos dispositivos, y proporcionan una vía para el tratamiento de una inmensa cantidad de datos en tiempo real (sistemas ‘Big Data’), facilitando la integración de infinidad de sensores aplicables prácticamente a cualquier tipo de necesidad. De hecho, ya han surgido incluso redes sociales de sensores, como la plataforma ‘Xively’ [3], donde los usuarios comparten datos en tiempo real procedentes de distintos sensores.

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.

También son muchas las empresas que ofrecen, o están interesadas, en soluciones IoT. La gestión de recursos y la eficiencia energética son las aplicaciones más solicitadas. Tecnologías inalámbricas como las redes móviles, WIFI, Zigbee, Bluetooth, etc., permiten optimizar posibles soluciones y facilitan su despliegue.



Arduino ha marcado un punto de inflexión en este sentido, convirtiéndose en la herramienta ideal para llevar a cabo multitud de prototipos y obtener nuevos usos de uso.

Por su bajo coste, sencillez y a la variedad de modelos que podemos encontrar, resulta una herramienta de gran ayuda a la hora de implementar ideas y soluciones de ámbito doméstico y profesional. Existen multitud de sensores y actuadores compatibles con esta plataforma, mediante los cuales podemos recopilar datos de nuestro entorno, analizarlos, y actuar en consecuencia, incluso conectar con otros dispositivos a través de las distintas tecnologías de comunicación (GSM/GPRS, 3G, Bluetooth, RFID, etc.) aprovechando toda una variedad de shields de expansión.

#### **1.4. Elección del hardware 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:

- 1. Precio.** Existen multitud de modelos de placas originales, todos ellos de muy bajo coste, además de existir versiones de otros fabricantes, mucho más baratas.
- 2. 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.
- 3. Sistema muy didáctico.** 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 pequeños proyectos de forma rápida y segura. Además, es compatible con multitud de módulos "plug&play, muchos de ellos sensores y otros "shields", que amplían el campo de actuación de Arduino.
- 4. 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.
- 5. Muy extendido y estandarizado.** Existen infinidad de librerías de libre distribución para poder comunicarse con hardware y software de terceros.
- 6. Lenguaje programación asequible.** Arduino se programa con su propio lenguaje de programación, fácil de manejar y muy bien documentado en su web y en infinidad de comunidades de Arduino.

## 2. LA PLATAFORMA ARDUINO

### 2.1 ¿Qué es Arduino?

Arduino es una plataforma electrónica de hardware libre basada en una placa con un microcontrolador. Con software y hardware flexibles y fáciles de utilizar, Arduino ha sido diseñado para adaptarse a las necesidades de todo tipo de público, desde aficionados, hasta expertos en robótica o equipos electrónicos.

Ante todo y sobre todo es un microcontrolador. Es decir, un ordenador completo integrado en un chip, con su CPU, memoria de programa, memoria de datos y circuitos para el control de periféricos.

El microcontrolador necesita para su correcto funcionamiento, de algunos circuitos auxiliares y complementos tales como:

- La entrada de alimentación
- El oscilador de trabajo
- Circuito de RESET
- La conexión USB
- Los accesos a las líneas de entrada y salida, etc.

También consta de un simple, pero completo, entorno de desarrollo, que nos permite interactuar con la plataforma de manera muy sencilla. Se puede definir por tanto como una sencilla herramienta de contribución a la creación de prototipos, entornos, u objetos interactivos destinados a proyectos multidisciplinares y multitecnología[4]. En la figura 2.1 podemos ver una de sus placas más vendidas a nivel mundial, se trata del modelo Arduino UNO.

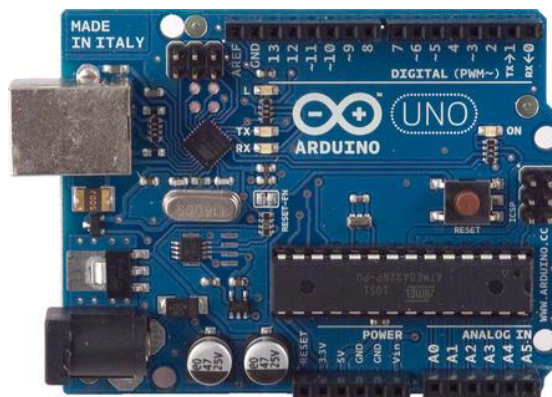


Figura 2.1 Fotografía de la placa Arduino UNO

A través de Arduino podemos recopilar multitud de información del entorno sin excesiva complejidad. Gracias a sus pines de entrada, nos permite jugar con toda una gama de sensores (temperatura, luminosidad, presión, presencia, humedad, gases, etc.) que nos brindan la capacidad de controlar o actuar sobre ciertos factores del entorno que le rodean, como por ejemplo: controlando luces, accionando motores, activando alarmas...y muchos otros actuadores.

Gracias a que la plataforma es de hardware libre, las placas Arduino[5] pueden ser hechas a mano por cualquier aficionado o compradas ya montadas de fábrica.

Su rotundo éxito se debe básicamente a los siguientes aspectos:

- **Económico.** Las tarjetas Arduino son bastante asequibles
- **Sencillez.** El lenguaje de programación es un lenguaje de alto nivel, fácil de aprender y de utilizar.
- **Hardware libre.** Los esquemas de las diferentes tarjetas de Arduino son fáciles de conseguir[6].
- **Código abierto.** Las herramientas de software y el lenguaje de programación son libres.
- **Corporativo.** Existen gran cantidad de colaboradores que día a día diseñan nuevas tarjetas, mejoran el entorno de trabajo, amplían el lenguaje de programación y proporcionan nuevas librerías y ejemplos.
- **Documentado.** La página web de Arduino está muy documentada, en lo relacionado al hardware y software.

Una de los principales motivos por el cual resulta muy interesante la utilización de la plataforma Arduino para determinados proyectos, se basa en su independencia respecto a tener que mantenerse conectado a un PC. Arduino es perfectamente capaz de trabajar en modo “*standalone*”, solo es necesario asegurarnos de haber cargado previamente el programa que deseamos que mantenga en ejecución. Si bien, todo esto no le priva de poder operar manteniendo en todo momento la conexión con el PC.

## 2.2 Tipos de tarjetas Arduino

### 2.2.1 Arduino UNO

Es la última versión de la placa básica de Arduino (fig. 2.2). Ha sido la evolución de la Duemilanove. Se conecta al ordenador mediante un cable USB estándar y contiene todo lo necesario para comenzar a programar y utilizar la placa. Sus funcionalidades se pueden ver incrementadas gracias a que existen multitud de Shields perfectamente compatibles con este modelo. Integra un nuevo chip USB-serie y cuenta con un nuevo diseño de etiquetado, facilitando la identificación de las distintas entradas y salidas de la placa. Con un precio aproximado de 24€, se trata quizás de la placa Arduino más interesante a la hora de buscar la mejor relación precio-prestaciones, por lo que será uno de los modelos a tener muy en cuenta para el desarrollo del proyecto.

El microcontrolador está insertado en la tarjeta mediante un **zócalo**. En caso de avería, no es frecuente pero puede ocurrir, se puede cambiar fácilmente por otro nuevo, sin necesidad de estar desoldando y soldando.

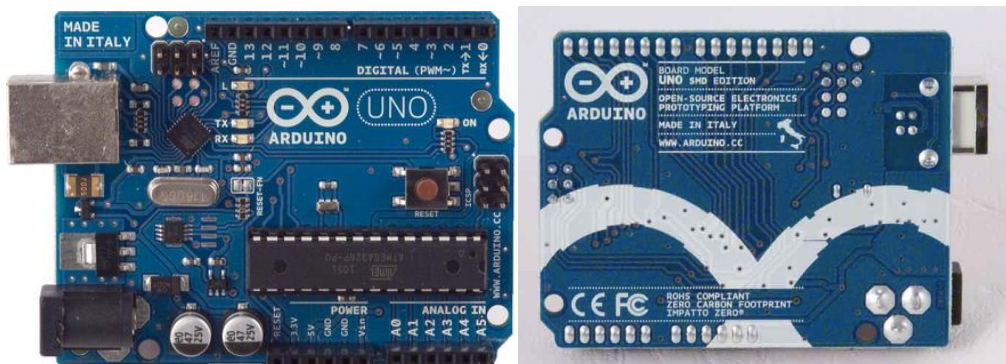


Figura 2.2 Arduino UNO



### 2.2.2 Arduino Leonardo

Muy similar a Arduino UNO, tanto en tamaño como en prestaciones, se trata de una evolución del mismo, con un precio similar. También es un modelo que puede resultar bastante interesante para el desarrollo de pequeños proyectos. Precio: 20€, algo más económica que la Arduino UNO, ya que se ha simplificado la electrónica, pues el propio microcontrolador integra el interface USB para la conexión con el PC.

El microcontrolador está soldado directamente en la tarjeta (fig. 2.3), lo que hace muy difícil su sustitución en caso de avería. Hay que contar con estación de soldadura/desoldadura para SMD y destreza con el manejo de este tipo de componentes.

Sus dimensiones son 69x53.3 mm



Figura 2.3 Arduino UNO

### 2.2.3 Mega 2560

El Arduino Mega es probablemente la placa con mayores prestaciones de la familia Arduino. Cuenta con 54 pines digitales, que funcionan como entrada/salida, además de sus 16 entradas analógicas. Es la placa más grande y potente de Arduino. Es totalmente compatible con los shields Arduino UNO, y cuenta con una memoria que duplica su capacidad en comparación con el resto de placas.

Arduino MEGA es por tanto la opción más adecuada para aquellos proyectos en los que se requiera un gran número de entradas y salidas disponibles, o para soportar la carga de códigos de programación pesados que no pueden almacenarse en las memorias de menor capacidad que ofrecen otras placas, como por ejemplo el modelo UNO.

Quizá por este motivo, resulte interesante contar con este modelo de placa para la realización del proyecto, ya que se pretenden utilizar bastantes entradas/salidas a las que conectar la Shield GSM/GPRS y los distintos sensores/actuadores para el desarrollo del proyecto. Además, el código puede llegar a ser bastante extenso cuando intentemos centralizar todas las funciones en una sola plataforma: control de temperatura, envío y recepción de SMS, activación de alarmas, etc. Su precio ronda los 45€, aunque podemos encontrar placas no oficiales por unos 25€.

El microcontrolador está soldado directamente en la tarjeta (fig. 2.4), lo que hace muy difícil su sustitución en caso de avería. Sus dimensiones son 102x53.3 mm

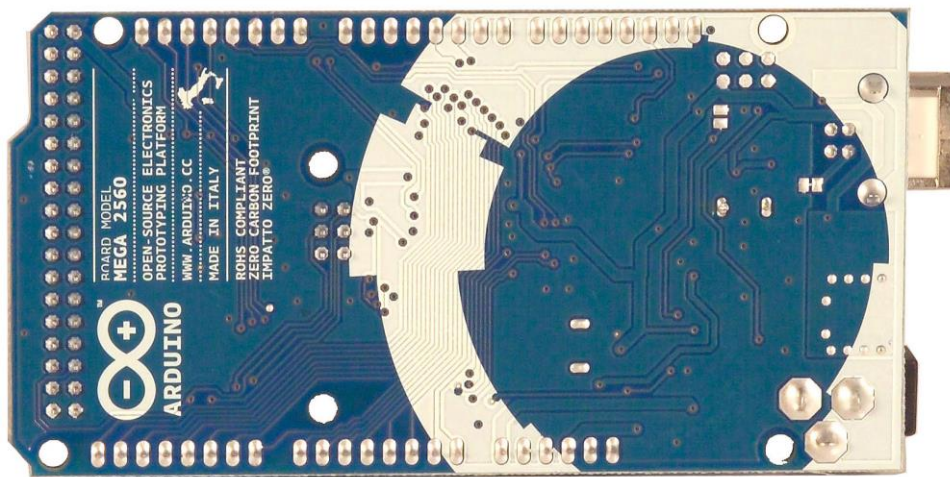
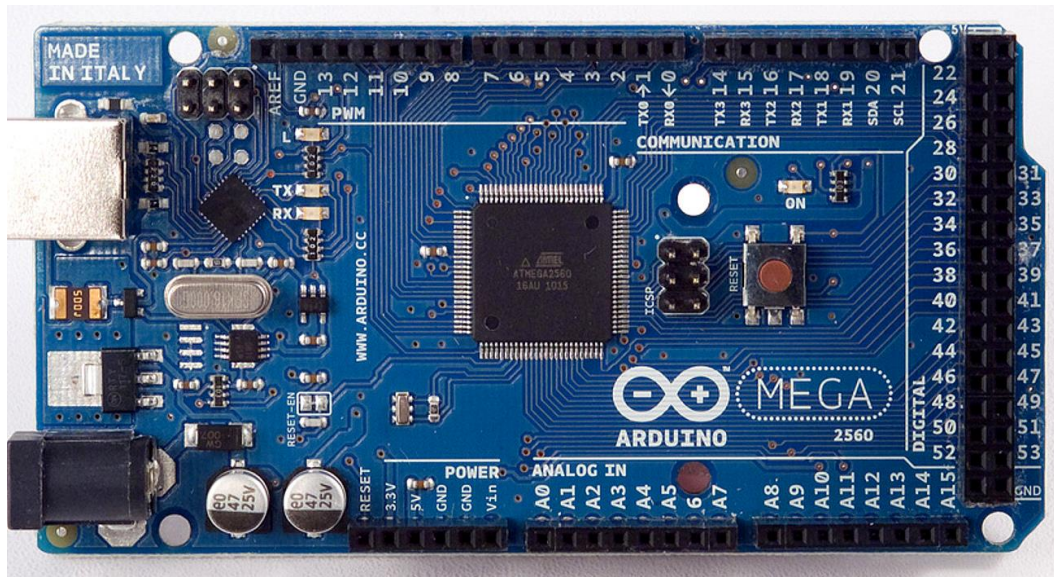


Figura 2.4 Arduino Mega 2560

## 2.2.4 DUE

Es la tarjeta Arduino mas potente que existe (fig. 2.5). El tamaño es idéntica a la Mega-2560. La diferencia se encuentra en el controlador que lleva. Es el mas rápido y con mas capacidad, de todos los montados en la familia Arduino.

El chip cuadrado, grande y negro en el centro de la placa, es el microcontrolador. Esta soldado directamente y el numero de patillas es muy elevado, lo que hace muy difícil su sustitución en caso de avería. Hay que contar con estación de soldadura/desoldadura para SMD y destreza con el manejo de este tipo de componentes.

Sus dimensiones son de 102x53.3 mm



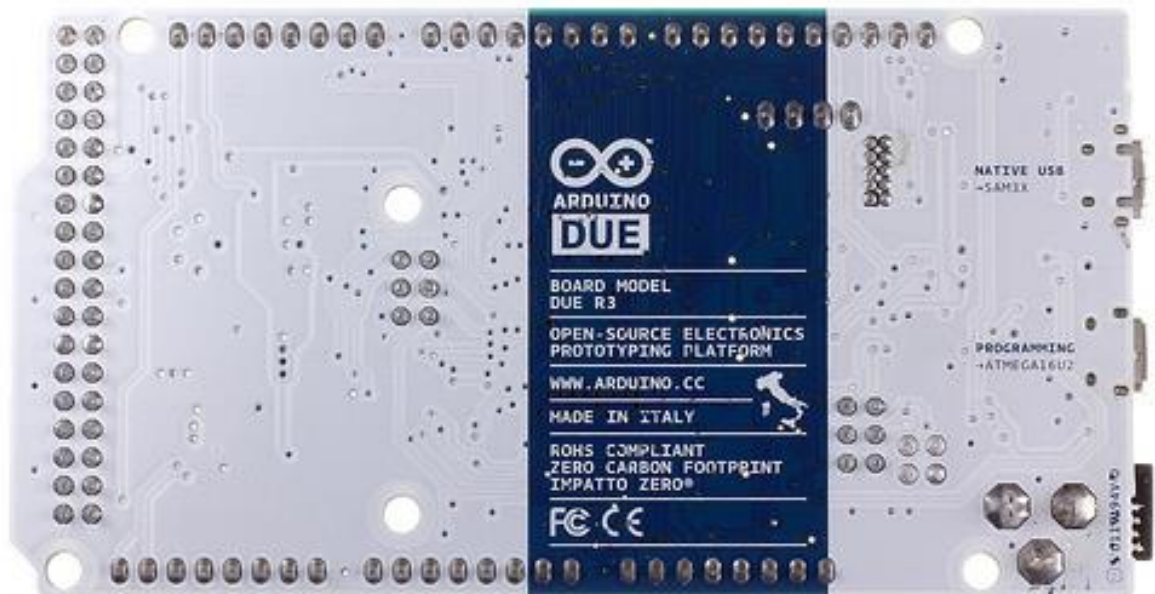
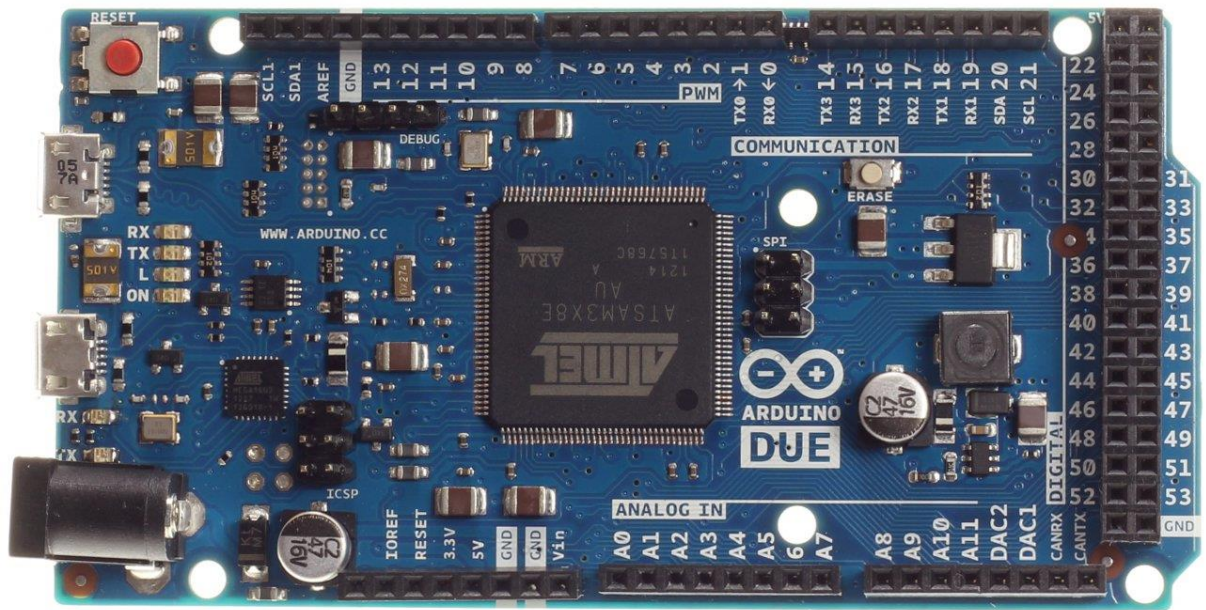


Figura 2.5 Arduino DUE

### 2.2.5 Arduino Nano

Una placa compacta diseñada para usar directamente en placas de desarrollo. Su conexión con el PC debe ser a través de un cable Mini-B USB(fig. 2.6).

Es de las tarjetas Arduino mas pequeñas que existen. Sus dimensiones son 19x43 mm. Se emplea para alojarlas en las propias máquinas que se van a controlar. Este tipo de tarjetas de Arduino se emplean para la producción de equipos. Precio: 40€.

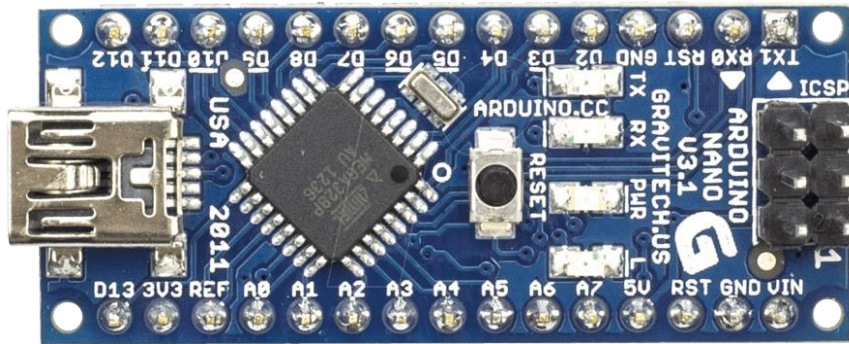


Figura 2.6 Arduino Nano

### 2.2.6 Arduino Mini

Es la placa Arduino más pequeña(fig. 2.7). Diseñada especialmente para aquellas aplicaciones en las que el espacio es primordial. Permite la conexión con un ordenador a través del adaptador Mini USB.

No incorpora interface ni conector USB. Es necesario por lo tanto emplear un adaptador especial que conecte la tarjeta con el puerto USB del PC y poder así grabar nuestros programas. Este adaptador se inserta por la fila de 6 orificios que hay en la derecha de la imagen. Una vez grabada la tarjeta se retira.

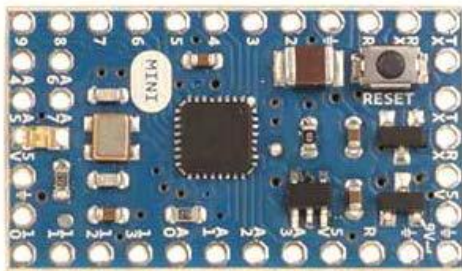


Figura 2.7 Arduino Mini

### 2.2.7 Arduino Pro Mini

La Pro Mini está diseñada para usuarios avanzados que buscan reducir los costes el máximo posible(fig. 2.8). Su tamaño es muy reducido. Precio: 23€. Sus dimensiones son de 18x33 mm. No incorpora interface ni conector USB. Es necesario por lo tanto emplear un adaptador especial que conecte la tarjeta con el puerto USB del PC y poder así grabar nuestros programas. Este adaptador se inserta por la fila de 6 orificios que hay en la derecha de la imagen. Una vez grabada la tarjeta se retira.



Figura 2.8 Arduino Pro Mini



## 2.2.8 Arduino Bluetooth

El Arduino BT contiene un módulo bluetooth (fig. 2.9), que permite comunicarse y programarse sin necesidad de cableado.

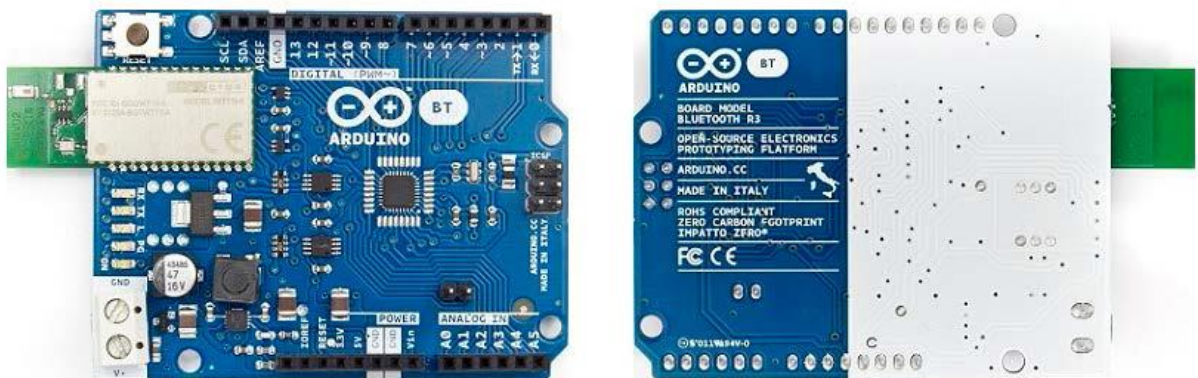


Figura 2.9 Arduino Pro Mini

## 2.2.9 Arduino Ethernet

Simplemente se trata de una placa Arduino, del estilo a Leonardo o Arduino UNO, pero dotada con un puerto Ethernet (fig. 2.10), para su conexión a Internet. Es una solución interesante para aquellos proyectos en los que se necesite estar conectado permanentemente a Internet. Precio: 35€.

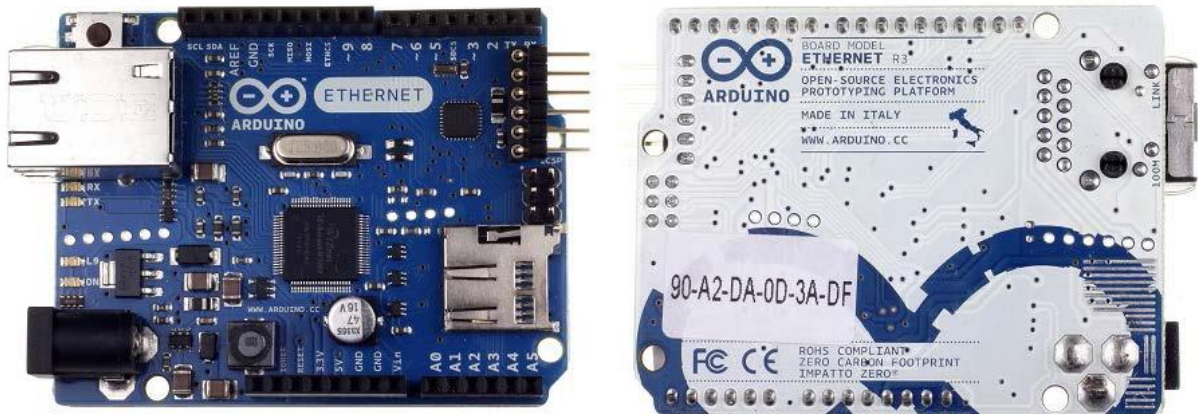
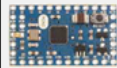








Figura 2.10 Arduino Pro Mini

Existen mas tipo de tarjetas Arduino, tal es la Lilypad, Arduino Fio, Arduino Pro, Arduino ADK, Arduino Robot, etc, no relevantes en este proyecto y de las cuales obviaremos sus detalles técnicos y usos para las que han sido diseñadas.

							
<b>Fabricante</b>	Arduino	Arduino	Arduino	Arduino	Arduino	Arduino	Arduino
<b>Modelo</b>	Pro Mini	Nano	Uno	Mega / Mega 2560	Leonardo	Micro	Due
<b>Microcontrolador</b>	AVR Atmega 168 ó 328 8bits	AVR ATmega 168 ó 328 8bits	AVR ATmega 328 8bits	AVR ATmega2560 8bits	AVR ATmega 32u4 8bits	AVR ATmega 32u4 8bits	ARM SAM3X8E Cortex-M3 32bits
<b>Frecuencia</b>	16Mhz	16Mhz	16Mhz	16Mhz	16Mhz	16Mhz	84Mhz
<b>Memoria RAM</b>	2KiB	2KiB	2KiB	8KiB	2.5KiB	2.5KiB	96KiB (64+32KiB)
<b>Memoria EEPROM</b>	1KiB	1KiB	1KiB	4KiB	1KiB	1KiB	0
<b>Memoria FLASH</b>	16 ó 32KiB	16 ó 32KiB	32KiB	128 ó 256KiB	32KiB	32KiB	512KiB
<b>Pines digitales entradas/salidas</b>	14/14	14/14	14/14	54/54	20/20	20/20	54/54
<b>Tensión/corriente pines digitales</b>	3.3v ó 5v 40mA	5v 40mA	5v 40mA	5v 40mA	5v 40mA	5v 40mA	3.3v 3~15mA (130mA entre todos)
<b>Pines analógicos entradas/salidas</b>	6/0	8/0	6/0	16/0	12/0	12/0	12/2
<b>Tensión/resolución pines analógicos</b>	3.3v ó 5v 10bits (1024 valores)	5v 10bits (1024 valores)	5v 10bits (1024 valores)	5v 10bits (1024 valores)	5v 10bits (1024 valores)	5v 10bits (1024 valores)	3.3v 12bits (4096 valores)
<b>Pines con interrupción externa</b>	2	2	2	6	2	2	-
<b>Pines PWM</b>	6	6	6	15	7	7	12
<b>Conexiones Serial / UART</b>	1	1	1	4	1	1	4
<b>Conexiones I2C / TWI</b>	1	1	1	1	1	1	2
<b>Conexiones ISP / ICSP</b>	1	1	1	1	1	1	1
<b>Conexión USB</b>	No (necesita adaptador externo)	Si	Si, USB-B	Si, USB-B	Si, Nativa, MicroUSB	Si, Nativa, MicroUSB	Si, Nativa, MicroUSB
<b>Conexión USB de depuración</b>	No	No	No	No	No	No	Si, MicroUSB
<b>Conexión Bluetooth</b>	No	No	No	No	No	No	No
<b>Conexión WiFi</b>	No	No	No	No	No	No	No
<b>Conexión Ethernet</b>	No	No	No	No	No	No	No
<b>Conexión USB Host</b>	No	No	No	No	No	No	Si
<b>Almacenamiento por SD</b>	No	No	No	No	No	No	No
<b>Corriente en el pin de 5v</b>	-	500mA	500~800mA	500~800mA	500~800mA	500mA	800mA
<b>Corriente en el pin de 3.3v</b>	-	50mA	50mA	50mA	50mA	50mA	800mA
<b>Voltaje de alimentación por el USB</b>	3.3v ó 5v (sin usb)	5v	5v	5v	5v	5v	5v
<b>Voltaje de alimentación recomendado por el Jack</b>	3.35 -12 V (modelo 3.3V) ó 5 - 12 V (modelo 5V)	7~12v	7~12v	7~12v	7~12v	7~12v	7~12v
<b>Voltaje de alimentación limite por el Jack</b>	-	6~20v	6~20v	6~20v	6~20v	6~20v	6~20v
<b>Precio oficial</b>	15+gi	-	20€+gi	40€+gi	18€+gi	18€+gi	39€+gi
<b>Precio BBB</b>	~4€	~9€	~10€	~12€	11€~	~16€	~38€

En la siguiente tabla se habla brevemente de cada una de las partes de la Arduino UNO:

CARACTERÍSTICAS	DESCRIPCIÓN
<b>Modelo de controlador</b>	Las tarjetas Arduino están construidas en torno a diferentes modelos de microcontroladores. Son estos precisamente lo que determinan el resto de características que definen a cada tarjeta
<b>Tensión de alimentación</b>	Indica el valor de la tensión externa que debemos aplicar a la tarjeta para su correcto funcionamiento. Según el modelo, la tensión puede variar de 3.8 v a 5 v, de 5 v a 12 v o de 7 v a 12 v. si aplicas una tensión superior a estos rangos, es posible que se dañe el microcontrolador.
<b>Tensión interna de trabajo</b>	Cada tarjeta Arduino incorpora una electrónica con la que se obtiene la tensión estable y fija a la que van a trabajar el resto de componentes, entre ellos el microcontrolador. La mayor parte de ellas trabajan con 5v. Únicamente DUE y Lilypad trabajan con 3.3v
<b>Velocidad de trabajo</b>	Representa en millones de ciclos por segundos (Mhz) la velocidad a la que trabaja el microcontrolador. Aproximadamente la cuarta parte de esa velocidad la emplea el micro para ejecutar instrucciones básicas. Por ejemplo, un micro trabajando a 16 Mhz ejecuta unos 4 millones de instrucciones por segundo.
<b>Patillas de E/S digitales</b>	Son patillas a las que podemos conectar los periféricos que vamos a controlar. A través de ellas el controlador puede leer el estado de los periféricos de entrada (pulsadores, sensores, teclados, etc.), o activar/desactivar los periféricos de salida (bombillas, relés, motores, pantallas, etc.). Cuantas más patillas tenga una tarjeta Arduino, más periféricos podrá controlar.
<b>Corriente de cada patilla</b>	Por las patillas de E/S se aplican u obtienen señales eléctricas hacia/desde los periféricos. Hay periféricos que funcionan con intensidades pequeñas (bombilla) y otros necesitan grandes corrientes (motor). Lo máximo que puede proporcionar Arduino por cada patilla es de 40 mA. Exigirle una corriente mayor, puede averiar la tarjeta Arduino.
<b>Entradas analógicas</b>	Se tratan de unas patillas de entrada a las que podemos conectar periféricos que proporcionan señales eléctricas de valor variable. Por ejemplo, un sensor de temperatura proporciona una señal cuya tensión es proporcional a la temperatura ambiente.
<b>Salidas PWM</b>	Son unas patillas que actúan como salidas hacia los periféricos. Las salidas que generan pueden ser moduladas y variables en el tiempo.
<b>Memoria FLASH de programa</b>	Es la memoria del micro, donde queda almacenado nuestro programa cuando lo grabamos desde el PC. Es una MEMORIA FIJA. Su contenido no varía nunca aunque se desconecte la alimentación. Solo se puede modificar cuando se graba en ella un nuevo programa.
<b>Ocupado por el Bootloader</b>	El Bootloader es un pequeño programa que viene grabado de fábrica en la memoria FLASH del micro. Este programa es el que se encarga de establecer la comunicación entre Arduino y el PC.
<b>Memoria RAM de datos</b>	Es la memoria del micro destinada a guardar datos temporales que posteriormente podrán ser rescatados y utilizados. Se puede grabar y leer datos tantas veces como se quiera. Es volátil.
<b>Memoria EPROM de datos</b>	Es otro tipo de memoria que tiene el micro y que está a medio camino entre la FLASH y la RAM. La memoria EPROM también se utiliza para guardar datos y luego poder rescatarlos. Puedes escribir en ella un número prácticamente ilimitados de veces y es no volátil.

## 2.3 LA TARJETA ARDUINO UNO

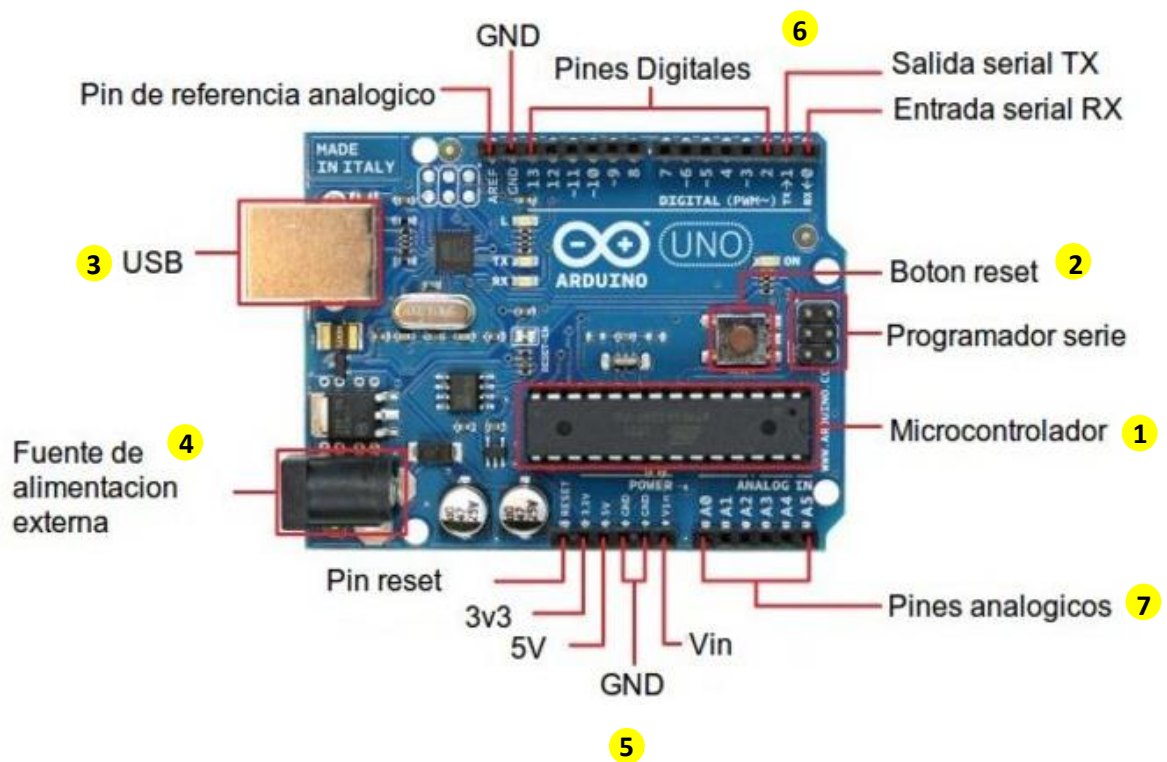
### 2.3.1 Secciones y descripción de Arduino uno

Es la tarjeta que vamos a usar en este proyecto de domotización del CPD de la Biblioteca Virtual del Sistema Sanitario Público de Andalucía (BV-SSPA).

Trasladar este proyecto a otra tarjeta de la familia Arduino es posible y se trata de una tarea rápida, siempre que respetemos la conexión de los periféricos a las mismas E/S digitales o salidas PWM. Así pues, podemos en un futuro reemplazar la Arduino uno por la Mega 2560, para aprovecharnos de su mayor número de E/S digitales.

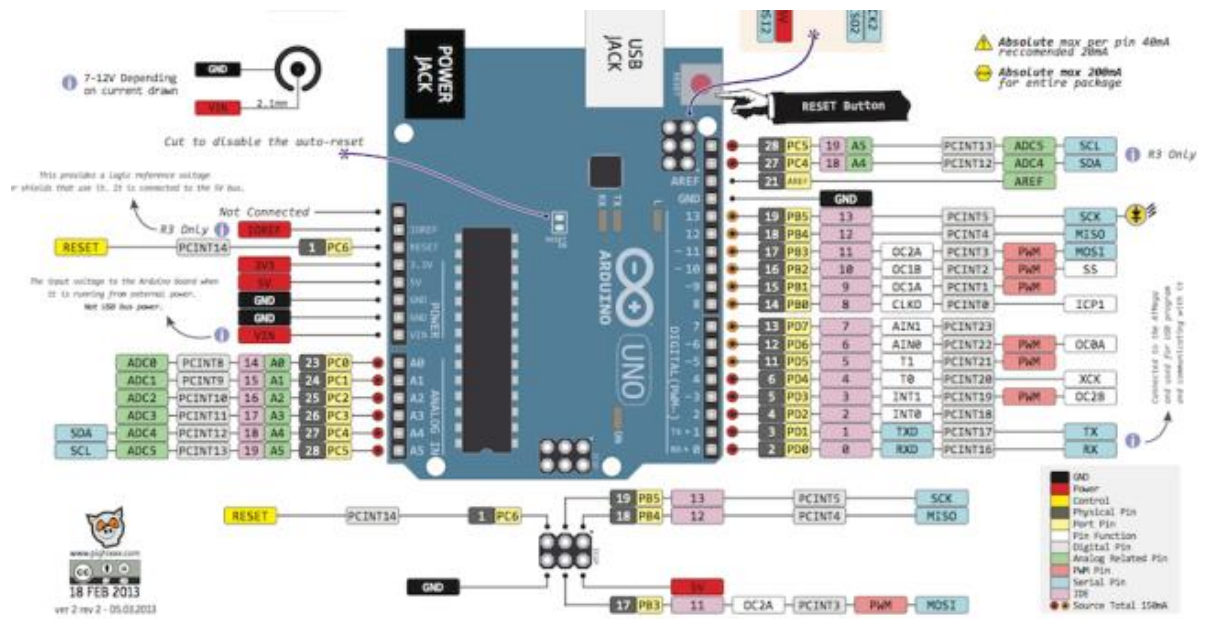
Lo único que tendremos que hacer es conectar la electrónica periférica a las mismas patillas, y cargar el programa en la nueva tarjeta Arduino que hemos reemplazado.

En la siguiente figura hay una imagen de Arduino UNO. En ella se han resaltado las secciones más importantes y que analizaremos a continuación.





SECCIÓN	DESCRIPCIÓN
1	Es el <b>microcontrolador</b> y “corazón” del sistema. En él se conecta los periféricos, se graban los programas y se ejecutan
2	Pulsador <b>RESET</b> . Sirve para iniciar el sistema, como si le conectaras la alimentación. Se usa con cierta frecuencia cuando el sistema se bloquea, reiniciando así la ejecución del programa.
3	Conector <b>USB</b> . Conecta Arduino con el PC. Una vez grabado el programa se puede retirar el cable USB. Arduino ejecutará el programa independientemente del PC.
4	<b>Clavija de alimentación</b> . Por ella debes conectar la tensión de alimentación procedente del adaptador AC o fuente de alimentación, o una pila/batería. En el caso de Arduino Uno la tensión estará comprendida entre 7v y 12v. Arduino queda alimentado también cuando está conectado al PC mediante el cable de USB.
5	<b>Power</b> . Se trata de un conector con 8 pines donde se puede insertar otros cables con los que realizar conexiones. La serigrafía de la placa Arduino corresponde a una señal determinada. Una mala conexión en estos pines puede estropear la tarjeta. <b>IOREF</b> : salida de frecuencia con la tensión a la que trabaja la tarjeta <b>3.3V</b> : Salida de tensión estabilizado a 3.3V <b>5v</b> : salida de tensión estabilizada a 5v. Alimenta a otros accesorios. <b>(2) GND</b> : conexión de tierra o 0v <b>Vin</b> : entrada de alimentación entre 7v y 12v
6	<b>Digital (PWM)</b> : Conector de 18 pines que dan acceso a las señales digitales de entrada y salida del controlador. En ellas conectarás los periféricos que vas a controlar: pulsadores, teclas, leds, motores, etc. Hay un total de 14 líneas numeradas del 0 al 13. Las señales 0 y 1 también se usan como líneas de recepción (Rx) y transmisión (Tx) serie de datos. Las líneas <b>3, 5, 6, 9, 10 y 11</b> precedidas del signo ~ pueden actuar como salida de señal PWM. El resto de las líneas <b>2, 4, 7, 8, 12 y 13</b> , son líneas digitales de entrada/salida.  De los 18 pines nos quedan cuatro señales.  <b>GND</b> : conexión a tierra o 0v <b>AREF</b> : entrada de tensión de referencia para el convertidor ADC <b>SDA</b> : señal de datos para el protocolo I2C de comunicaciones <b>SCL</b> : señal de reloj para el protocolo I2C de comunicaciones
7	<b>Analogic IN</b> : Conector de 6 pines que permiten conectar hasta 6 periféricos analógicos de entrada ( <b>A0 – A5</b> ).



### 3. ENTORNO DE DESARROLLO EN ARDUINO

#### 3.1. Descarga del entorno de trabajo

El entorno de desarrollo en Arduino (IDE) es el encargado de la gestión de la conexión entre el PC y el hardware de Arduino con el fin de establecer una comunicación entre ellos por medio de la carga de programas. Como podemos ver en la figura 3.16, el IDE de Arduino se compone de[7]:

- Un **editor de texto**.- donde se escribe el código del programa.
- Un **área de mensajes**.- a través del cual el usuario tendrá constancia en todo momento de los procesos que se encuentren en ejecución, errores en el código, problemas de comunicación, etc.
- Una **consola de texto**.- mediante la que podremos comunicarnos con el hardware Arduino y viceversa.
- Una **barra de herramientas**.- donde podremos acceder a una serie de menús y a los botones con acceso directo a las principales funcionalidades de Arduino.

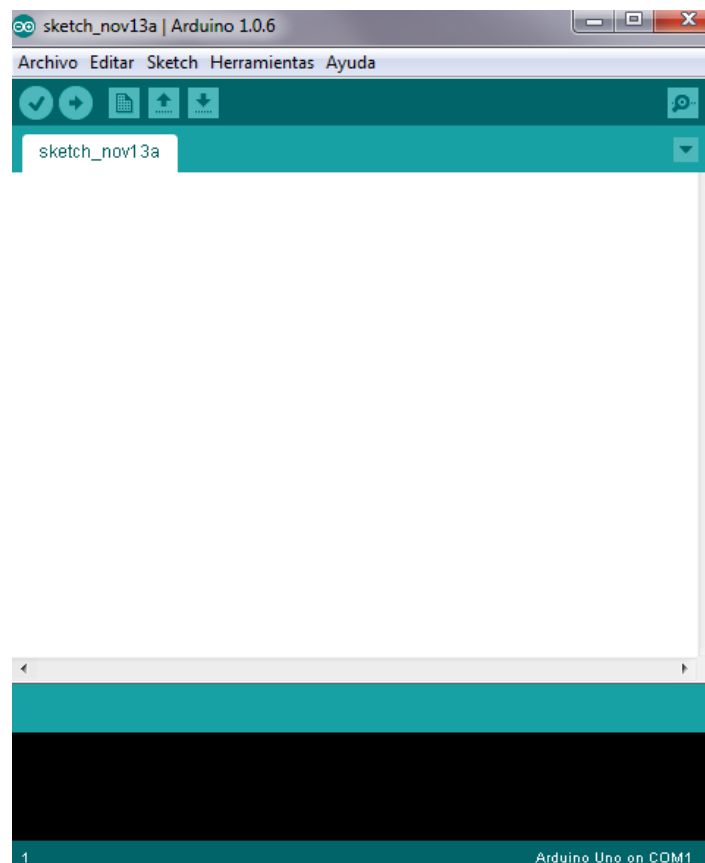


Figura 3.1 Entorno de desarrollo en Arduino


A través de la IDE de Arduino, podemos escribir el código del programa software y crear lo que se conoce por "sketch" (programa). ¿Por qué lo llamamos sketch y no programa? Pues porque el IDE de Arduino viene de Processing, y en este lenguaje de programación enfocado al mundo gráfico, cada código es considerado un boceto, en inglés "sketch".


El sketch permite la comunicación con la placa Arduino. Estos programas son escritos en el editor de texto, el cual admite las posibilidades de cortar, pegar, buscar y remplazar texto. En el área de mensajes se muestra, tanto la información mientras se cargan los programas, como los posibles errores que tengamos a la hora de compilar, ya sea por problemas en el código del sketch, por fallo en la detección de nuestro Arduino en el puerto USB, o por cualquier otro problema que sea detectado.


La consola muestra el texto de salida para el entorno de Arduino incluyendo los mensajes de error completos y otras informaciones.


Desde la barra de herramientas tenemos acceso directo a las principales funcionalidades que ofrece el IDE de Arduino, como por ejemplo: verificar el proceso de carga, crear un nuevo sketch, abrir un sketch ya existente, guardar los programas, abrir el Monitor Serial, etc.


A continuación pasamos a describir la utilidad de cada uno de los iconos que aparecen en la pantalla principal del entorno de desarrollo de Arduino:


 **“Verificar”**.- Esta funcionalidad se encarga de verificar el código del sketch en busca de posibles errores. A través del área de mensajes se le notificará al usuario el resultado de dicha verificación. En el caso de que se detecten errores en el código, éstos se detallarán junto con el número de línea en la que han sido detectados. Sólo cuando la comprobación resulta libre de errores podremos proceder a la carga del código en nuestra placa Arduino.

 **“Cargar”**.- Permite compilar el código del sketch y lo carga en Arduino. Cuando la carga a terminado se informa al usuario a través del área de mensajes, y podremos proceder a la apertura del monitor serial.

 **“Nuevo”**.- Para la creación de un nuevo sketch. Abre una nueva hoja de texto donde escribiremos el código correspondiente al sketch.

 **“Abrir”**.- Permite abrir un sketch ya existente que ha sido previamente guardado. También puedes abrir cualquiera de los sketches que trae instalados por defecto el IDE de Arduino.

 **“Guardar”**.- Esta funcionalidad nos permite almacenar el sketch que estemos desarrollando en ese momento. Te permite elegir la ruta en la que quieres guardarlo, y te crea automáticamente una carpeta con el mismo nombre que le des al sketch, guardando éste dentro de la misma.

 **“Monitor Serial”**.- Al pinchar sobre este icono, el entorno de desarrollo de Arduino abre una nueva ventana a través de la cual podemos ver la comunicación establecida por el puerto serie entre la placa Arduino y el PC durante la ejecución del programa. Contiene una barra de escritura mediante la que podemos comunicarnos con Arduino a través de su puerto serie, por ejemplo, para seleccionar distintas opciones que contemple un posible menú creado por el usuario dentro de un código, o para enviar directamente comandos AT a una shield GPRS/GSM que tengamos montada sobre el Arduino. También contempla la opción de seleccionar el envío de algunos caracteres junto con el texto que introduzcamos en la barra de entrada del mismo, como el carácter de nueva línea, retorno de carro, o los dos. En la figura 3.2 podemos ver la pantalla correspondiente al Monitor Serial y la pestaña desplegable en la que podemos seleccionar las distintas opciones referentes a los caracteres de fin de línea.

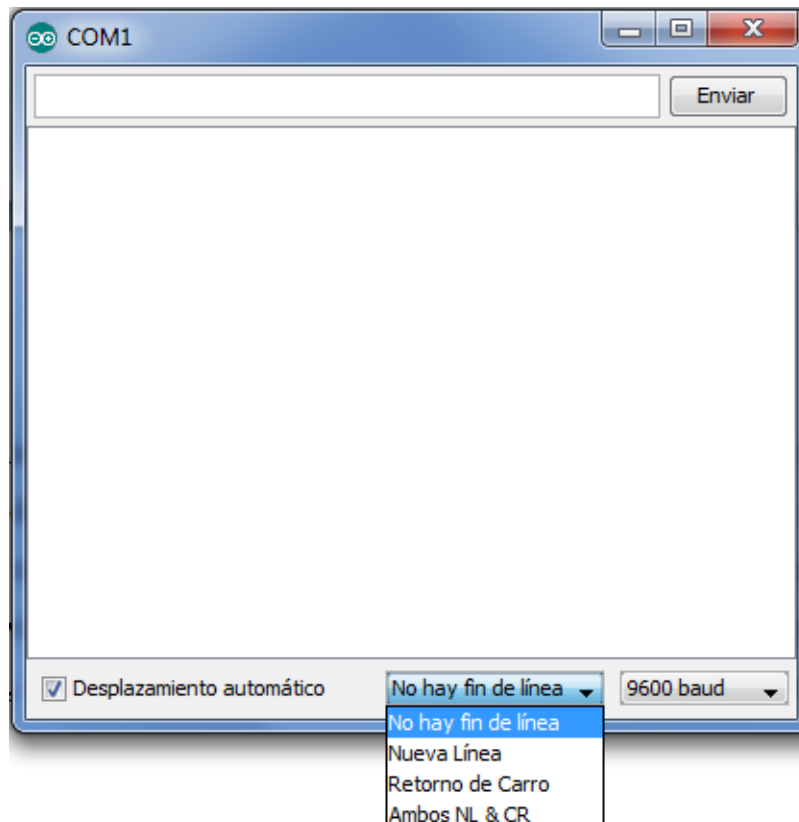


Figura 3.2 Selección de caracteres de fin de línea en la ventana "Monitor Serial"

Dentro del Monitor Serial disponemos de otra pestaña para establecer la tasa de baudios (Baudrate), que marca el número de unidades de señal transmitidas por segundo. Este valor ha de estar sincronizado con el baudrate en el que esté trabajando el Arduino, el cual puede ser establecido en el código del sketch mediante el comando `Serial.begin("valor delbaudrate")`, o de no ser así, se establecerá un valor por defecto. Si Monitor Serial y Arduino no están sincronizados con la misma tasa de baudios, la información que aparezca en la ventana será completamente ilegible. En la figura 3.3 aparece desplegada la pestaña para la selección de los distintos valores de baudrate disponibles.

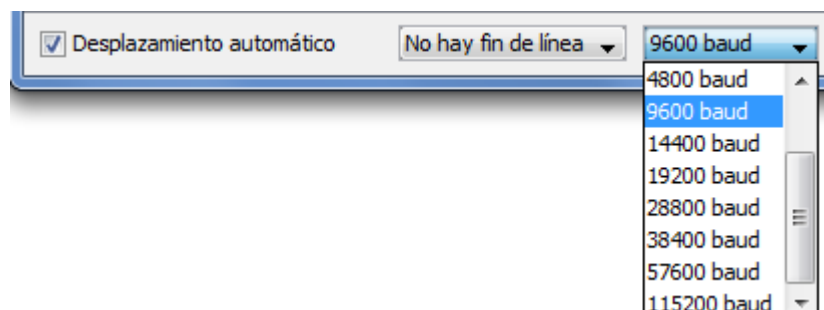


Figura 3.3 Selección del valor de 'baudrate' en la ventana "Monitor Serial"

Dentro de los menús, cabe mencionar la existencia de librerías, que pueden proporcionar funcionalidades adicionales para la utilización en sketches, por ejemplo para trabajar con hardware o manipular datos. Para utilizar una librería dentro de un sketch, debemos declararla

previamente. Para ello nos iremos al menú “*sketch*”, y seleccionaremos la opción importar librerías. Dentro buscaremos la librería que sea de nuestro interés y la importaremos al *sketch*, insertando una sentencia de tipo *#include* al comienzo del mismo. Se debe tener en cuenta que al cargar un código que incluya librerías, éstas también se vuelcan en la placa junto con el resto del *sketch*, incrementando la ocupación del programa y reduciendo el espacio disponible en la memoria de Arduino.

## 4. SHIELDS EMPLEADOS EN LA DOMOTIZACION DEL CPD DE LA BV-SSPA

### 4.1 Los Shields

Si queremos ampliar las funcionalidades de nuestra plataforma Arduino, siempre podemos recurrir a una gran variedad de shields compatibles prácticamente con cualquiera de sus modelos. De este modo, podemos dotar al dispositivo de funciones adicionales dedicadas específicamente a ofrecer algún tipo de servicio concreto.

Un shield es un módulo de expansión en forma de placa impresa que se puede conectar a la parte superior de la placa Arduino para ampliar sus capacidades, permitiendo además ser apiladas unas encima de otras manteniendo un diseño modular, tal como podemos ver en la Figura 4.1

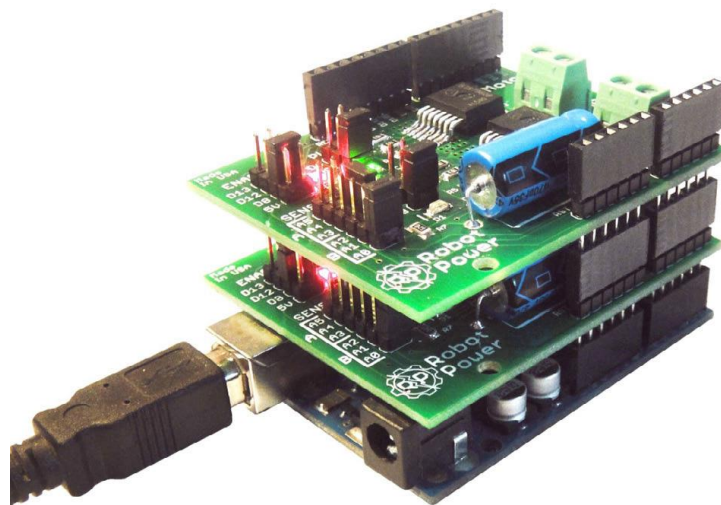


Figura 4.1: Ejemplo de estructura modular con varias shields para Arduino

Para nuestro proyecto del CPD de la BV-SSPA, buscamos una shield que nos permita utilizar los sistemas de comunicaciones *móviles e internet*, para poder interactuar a distancia con nuestra plataforma. Podemos encontrar varios shields que han sido diseñadas específicamente para ofrecer servicios a través de los sistemas GSM, GPRS, 3G, o una combinación de los mismos. Y shields para dotar de servicios de internet a nuestro sistema Arduino.[8]

Algunos Shiels más comunes:



Figura 4.1 Shield de Ethernet



Figura 4.2 Shield de pantalla táctil



Figura 4.3 Shield de Relés



Figura 4.4 Shield de pantalla





Figura 4.5 Shield de Relés

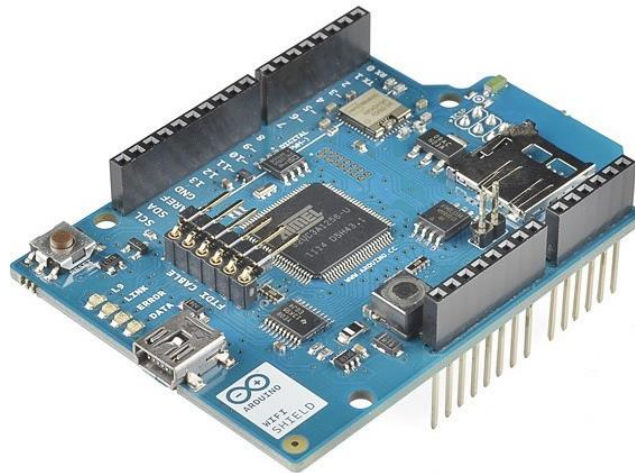


Figura 4.6 Shield de Wifi



Figura 4.7 Shield GSM – GPRS



Figura 4.8 Shield Xbee

#### 4.2 Esquema teorico del proyecto de domozitacion y televigilancia del CPD de la BV-SSPA

Antes de empezar a describir cada uno de los sensores, asociado a su circuito electrónico, cuando sea el caso, cada uno de los Shields empleados en el proyecto, y la conexión de todos estos elementos a la placa Arduino, es necesario dar una imagen visual de todo el sistema electrónico.

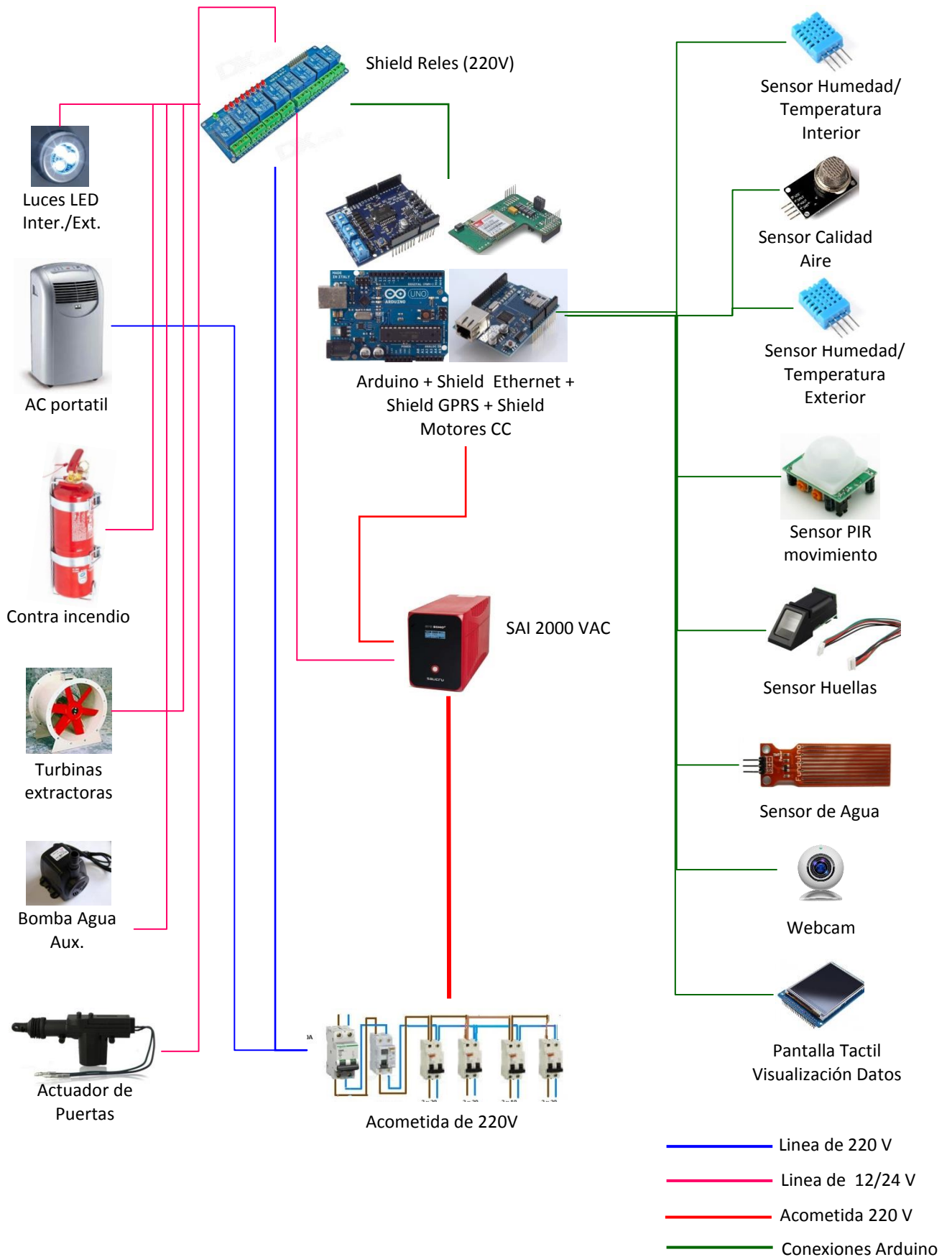
Esta imagen dará una visión de todo el proyecto, y las primeras pinceladas de todo su entorno de trabajo.

De forma lógica se irá explicando cada una de las partes del proyecto, previa visión global del mismo.

En esa explicación modular se seguirá el mismo esquema:

- Sensor/Shield empleado
- Circuito electrónico asociado y modo de conexión a la placa Arduino
- Programa que lo controla con el IDE de Arduino
- Simulación de circuito con Electronics workbench Multisim
- Circuito impreso, realizado con el programa Fritzing

# DIAGRAMA PRÁCTICO DE LA DOMOTIZACION/TELECONTROL DEL CPD DE LA BV-SSPA



## 4.2 Módulo GPRS/GSM quadband para Arduino

Esta shield puede convertir nuestra placa Arduino en un plataforma capaz de ofrecer conectividad GPRS/GSM. Integra un módulo SIM900 que nos permite establecer llamadas con otros dispositivos móviles, enviar SMS, incluso la comunicación de datos a través de los protocolos TCP, UDP, HTTP o FTP.



Figura 4.9 Módulo GPRS/GSM Quadband para Arduino

A continuación se adjunta el esquema de diseño correspondiente a dicho módulo:

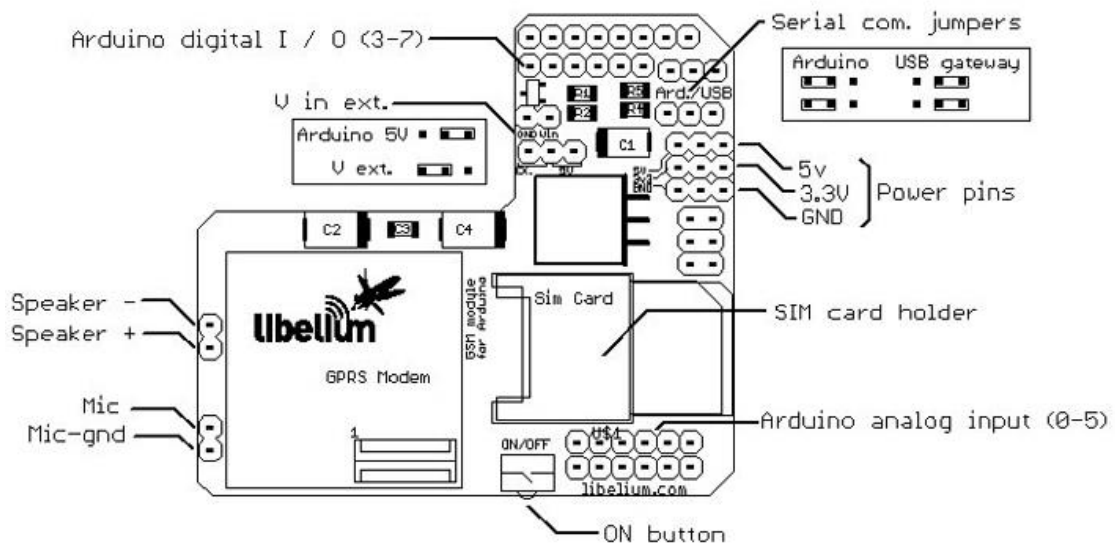


Figura 4.10 Diagrama de conexiones Módulo GPRS/GSM

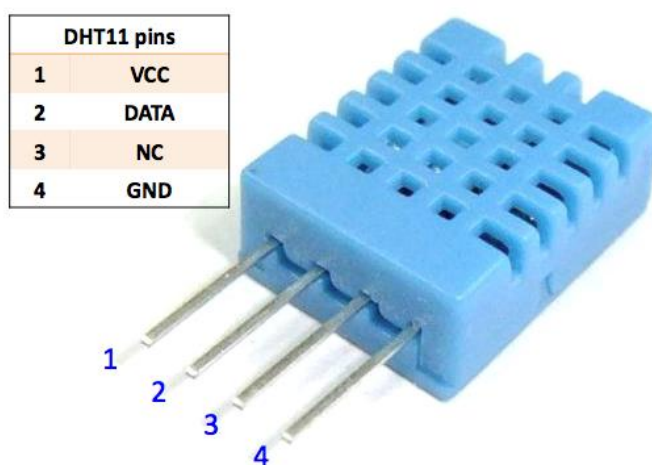
Este Shield, lo usaremos en nuestro proyecto, para que Arduino pueda enviar al personal de la televigilancia del CPD de la BV-SSPA. SMS de alerta de temperatura. Este servicio es redundante,

pues el sistema de domotización de Arduino, al mismo tiempo que envía el SMS de alerta, ejecuta un algoritmo para paliar el exceso de temperatura que hay dentro del armario informático, y que se expondrá mas adelante.

El servicio de SMS es para tener en alerta al personal de la custodia del CPD, y operar, si es necesario, con el interfaz remoto (BMS, Building Management System) via web o Smartphone.

### 4.3 Sensor de temperatura

Para este proyecto, entre los diferentes tipos de sensores que hay, analógicos / digitales, nos hemos decantado por el DHT11, de temperatura y humedad.



4.11 Sensor DHT11

El DHT11 es un sensor que proporciona una salida de datos digital. Entre sus ventajas podemos mencionar el bajo coste y el despliegue de datos digitales. Esto supone una gran ventaja frente a los sensores del tipo análogo, como el LM335 por ejemplo, en los cuales las fluctuaciones en el voltaje alteran la lectura de datos.

Entre las desventajas pues, el DHT11 solo lee enteros, no podemos leer temperaturas con decimales por lo que tenemos que pensarlo muy bien a la hora de utilizar este sensor para trabajos en los que se requieran lecturas precisas de temperatura y/o humedad.

Si se desea utilizar un sensor de humedad/temperatura de alta precisión recomendamos el *DHT22*.

La siguiente tabla muestra las diferencias mas significativas entre ambos sensores.

Modelo	DHT11	DHT22
Rango de medición de humedad	20-90 % HR	0-100 % HR
Rango de medición de temperatura	0 hasta 50 °C	-40 hasta 80 °C
Precisión de temperatura	±2 °C	±0.5 °C
Precisión de humedad	±5 % HR	±2 % HR



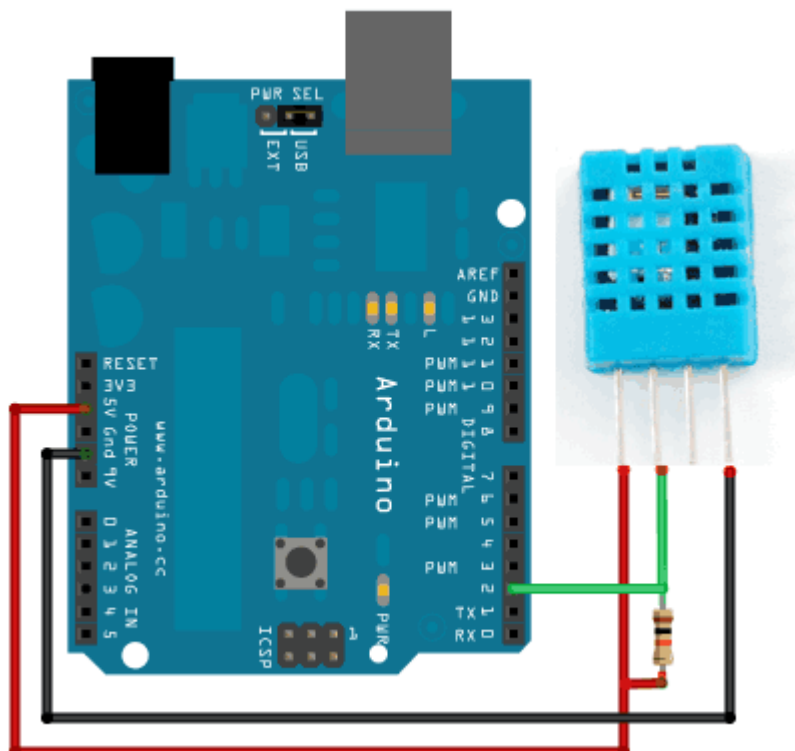
El sensor de temperatura se usara ene l CPD de la BV-SSPA para las siguientes situaciones:

- *Mostrar datos via web (Xively), para monitorizar en tiempo real la temperatura interior/ exterior del armario y el % de humedad.*
- Para disparar los ventiladores de extracción de aire caliente, ubicados en tres toberas de 110 mm en la parte superior del armario. El disparo se hara fijando una temperatura critica, para la salvaguarda de todos los equipos informáticos que aloja el CPD.
- El mismo sensor se usara para el disparo de las aperturas de puertas, cuando la temperatura sea tan alta, en este caso las maquinas de AC no están cumpliendo su cometido o la principal, o la auxiliar han dejado de funcionar, y arduino por medio de este sensor, dará un pulso de CC a través de la tarjeta de relés para desbloquear el cierre magnetido de las puertas, a continuación los actuadores eléctricos, abrirán las puertas, y finalmente las turbinas extractoras de aire también entrarían en funcionamiento.

Con este algoritmo lo que se pretende es la libre circulación del aire, para desalojar el mismo del interior del armario y proteger los tres servidores y resto de equipo informático.

Si se restablece la temperatura interior (fijada con la programación del IDE de Arduino), por seguridad, se pondrá en marcha un proceso inverso al descrito. Los actuadores cerrarán las puertas, y los extractores dejarán de funcionar.

En todo momento este proceso puede intervenirse manualmente, pues las maniobras que se han descritos se podrán ejecutar desde interfaz hombre-máquina (HMI), en versión web y para Smartphone, que facilite la gestión remota del CPD.



4.12 Montaje del sensor DHT11 sobre Arduino

La anterior ilustración muestra el esquema practico del montaje del sensor DHT11 con Arduino.

El código mas simple para funcionar con Arduino el sensor DHT11 se muestra a continuación:

```
/*Sensor de Temperatura y Humedad DHT11<br>Instrucciones:  
Recuerda descargar la libreria DHT para poder utilizar este sensor  
Conectaremos el Sensor DHT11 a 5v y el pin de señal a la entrada digital  
7  
*/  
#include "DHT.h"  
#define DHTPIN 7  
#define DHTTYPE DHT11  
DHT dht(DHTPIN, DHTTYPE);  
void setup() {  
  Serial.begin(9600);  
  dht.begin();  
}  
void loop() {  
  int h = dht.readHumidity();// Lee la humedad  
  int t= dht.readTemperature();//Lee la temperatura  
  ////////////////////////////////////////////////////////////////////Humedad  
  Serial.print("Humedad Relativa: ");  
  Serial.print(h);//Escribe la humedad  
  Serial.println(" %");  
  delay (2500);  
  ////////////////////////////////////////////////////////////////////Temperatura  
  Serial.print("Temperatura: ");  
  Serial.print(t);//Escribe la temperatura  
  Serial.println(" C");  
  delay (2500);  
  ////////////////////////////////////////////////////////////////////  
  Serial.println("BV-SSPA");  
  delay (3000);  
  Serial.println ();  
}  
//BV-SSPA
```

En el CPD de la BV-SSPA, hemos incluido una pantalla táctil, que visualice los valores que devuelven los sensores. El objetivo de la pantalla, es doble. Por un lado, estar accesible para operar con ella (función táctil) y por otro, como medio de visualización de los valores.

Esta pantalla estará ubicada, justo debajo de la consola original del CPD.

En la BV-SSPA, hemos desarrollado el sensor de temperatura y humedad DHT11 junto con un sensor de luz exterior, que nos dará lúmenes, que interpretaremos como posibilidad de día caluroso, en el inapropiado emplazamiento del armario.

El conexionado sería:



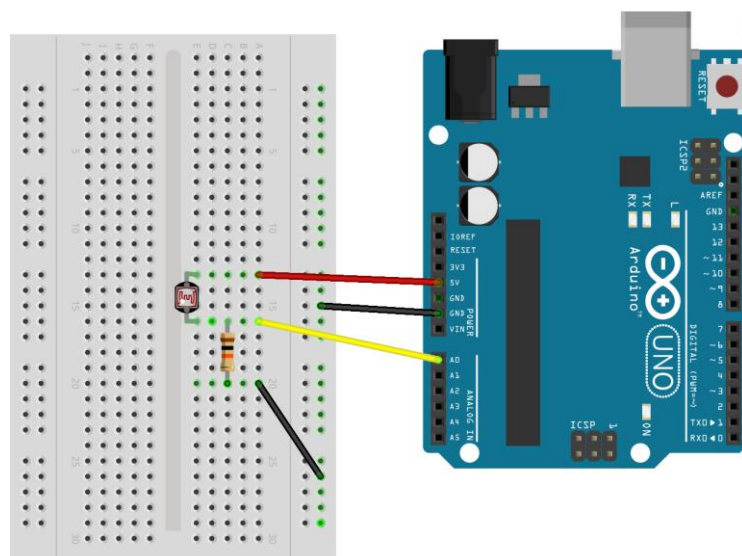
4.13 Pantalla TFT

Arduino: 5v      TFT DISPLAY   VCC

Gnd	GND
Digital pin 4	SCL
Digital pin 5	SDA
Digital pin 6	CS
Digital pin 7	DC
Digital pin 8	RES

Necesitamos instalar la librería de la pantalla de TFT ST7735.

El sensor de Luz, lo hemos hecho con una LDR y una resistencia de 10K ohmios.

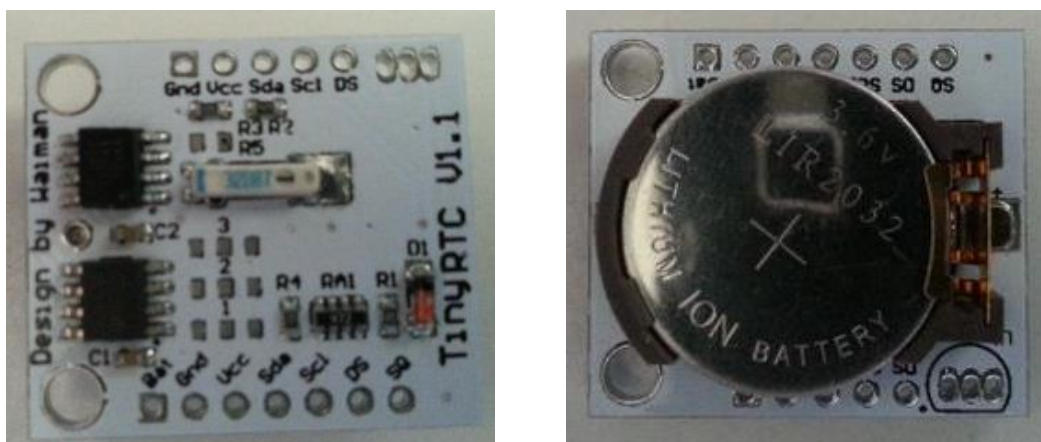


4.14 Conexionado del sensor de luz



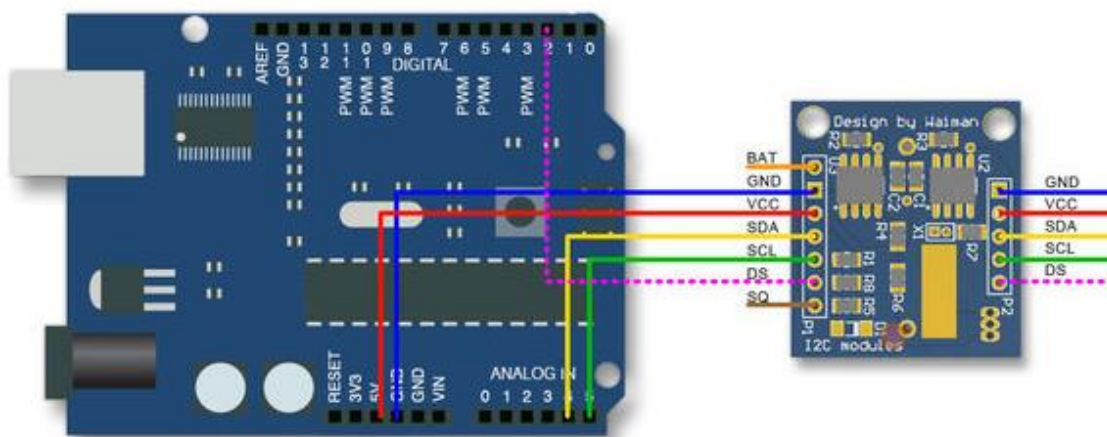
Hemos instalado con el sensor de temperature/humedad y luz un sensor, para tener fecha y hora en nuestro interfaz remoto (BMS, Building Management System).

Hemos usado el modulo TinyRTC:



4.15 Shield reloj

Hay que instalar la librería RTC y Wire. El conexionado a Arduino sería:



4.16 Conexionado del reloj a Arduino

Y el Sketch quedaría:

```
#define sclk 4  
  
#define mosi 5  
  
#define cs 6  
  
#define dc 7  
  
#define rst 8
```

```
#define ANALOG_IN 3 // for cds light sensor

#include <Adafruit_GFX.h> // Core graphics library

#include <Adafruit_ST7735.h> // Hardware-specific library

#include <SPI.h>

#include <Wire.h> // library needed for RTC

#include "RTClib.h" //RTC Library

#include <dht11.h> // DHT11 temp humidity sensor library

RTC_DS1307 RTC;

dht11 DHT11;

Adafruit_ST7735 tft = Adafruit_ST7735(cs, dc, mosi, sclk, rst);

void setup(void) {

DHT11.attach(2); // set digital port 2 to sense DHT11 input

Wire.begin();

RTC.begin();

tft.initR(INITR_BLACKTAB); // initialize a ST7735S chip, black tab

tft.fillScreen(ST7735_BLACK);

//tftPrintTest(); //Initial introduction text, uncomment to view on screen

tft.fillScreen(ST7735_BLACK); // Clear screen

// *** Printing static Items on display in the setup void in order to speed up the loop void****

tft.drawLine(0, 50, tft.width()-1, 50, ST7735_WHITE);

tft.setCursor(0, 60);

tft.setTextColor(ST7735_YELLOW);

tft.println("Temperature (C): ");

tft.drawLine(0, 110, tft.width()-1, 110, ST7735_WHITE);
```

```
tft.setTextColor(ST7735_WHITE);

tft.println("Humidity  (%): ");

tft.setTextColor(ST7735_YELLOW);

tft.println("Temperature (F): ");

tft.setTextColor(ST7735_YELLOW);

tft.println("Temperature (K): ");

tft.setTextColor(ST7735_WHITE);

tft.println("Dew Point  (C): ");

tft.setTextColor(ST7735_WHITE);

tft.println("DewPointFast(C): ");

tft.setCursor(0,115);

tft.setTextColor(ST7735_YELLOW);

tft.print("Light intensity ");

}

void loop() {

tft.setCursor(10,0);

tft.setTextColor(ST7735_WHITE);

tft.setTextSize(1);

tft.println("INSTRUCTABLES.COM");

tft.setTextColor(ST7735_YELLOW,ST7735_BLACK);

tft.setTextSize(1);

tft.setCursor(30,10);

DateTime now = RTC.now();

tft.print(now.year(), DEC);

tft.print('/');

tft.print(now.month(), DEC);
```

```
tft.print('/');

tft.print(now.day(), DEC);

tft.println(' ');

tft.setCursor(15,25);

tft.setTextSize(2);

tft.setTextColor(ST7735_BLUE,ST7735_BLACK);

tft.print(now.hour(), DEC);

tft.print(':');

tft.print(now.minute(), DEC);

tft.print(':');

tft.print(now.second(), DEC);

tft.println(" ");

tft.setTextSize(1); //set text size for all data coming from DHT11

tft.setCursor(98, 60);

tft.setTextColor(ST7735_GREEN, ST7735_BLACK); // set color for all data coming from DHT11

tft.print((float)DHT11.temperature,2);

tft.setCursor(98, 68);

tft.print((float)DHT11.humidity,2);

tft.setCursor(98, 76);

tft.print(DHT11.fahrenheit(), 2);

tft.setCursor(98, 84);

tft.print(DHT11.kelvin(), 1);

tft.setCursor(98, 92);

tft.print(DHT11.dewPoint(), 2);

tft.setCursor(98,100);

tft.print(DHT11.dewPointFast(), 2);
```

```
tft.setCursor(98,115);

int val = analogRead(ANALOG_IN); // READ LIGHT SENSOR VALUE

tft.setTextColor(ST7735_YELLOW, ST7735_BLACK); //set light sensor data text color

tft.print(val, 1); // PRINT LIGHT SENSOR VALUE

tft.print(" ");

}

void tftPrintTest() {

  tft.setTextWrap(false);

  tft.fillScreen(ST7735_BLACK);

  tft.setCursor(0, 10);

  tft.setTextColor(ST7735_WHITE);

  tft.setTextSize(1);

  tft.println("INSTRUCTABLES.COM");

  delay(500);

  tft.setCursor(0, 60);

  tft.setTextColor(ST7735_RED);

  tft.setTextSize(2);

  tft.println("temperature");

  tft.setTextColor(ST7735_YELLOW);

  tft.setTextSize(2);

  tft.println("humidity");

  tft.setTextColor(ST7735_GREEN);

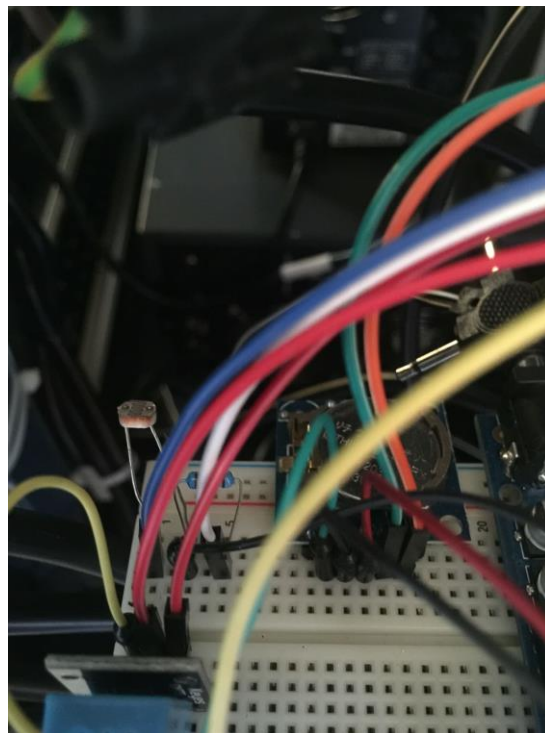
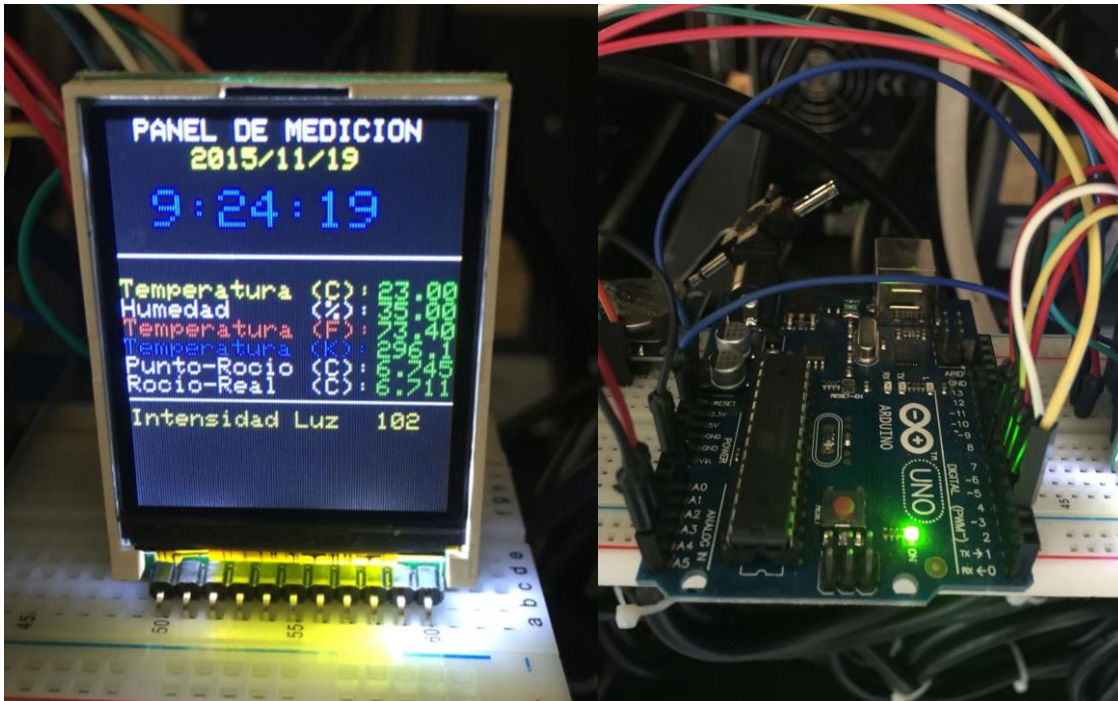
  tft.setTextSize(2);

  tft.println("monitor");

  tft.setTextColor(ST7735_BLUE);

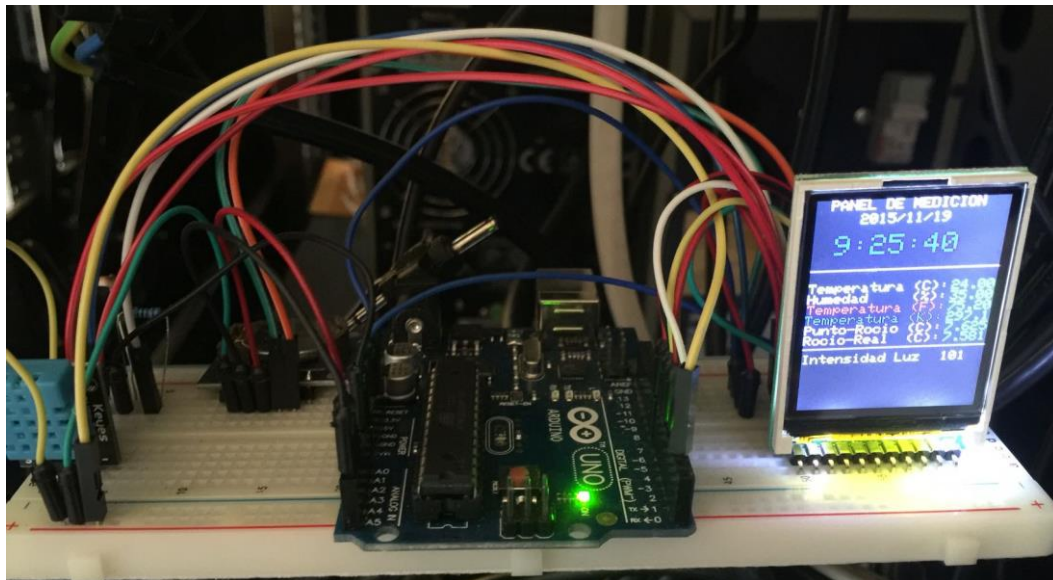
  delay(50); }
```

Algunas fotos de esta sección del Proyecto:



4.18 Prototipo del sensor de Temperatura

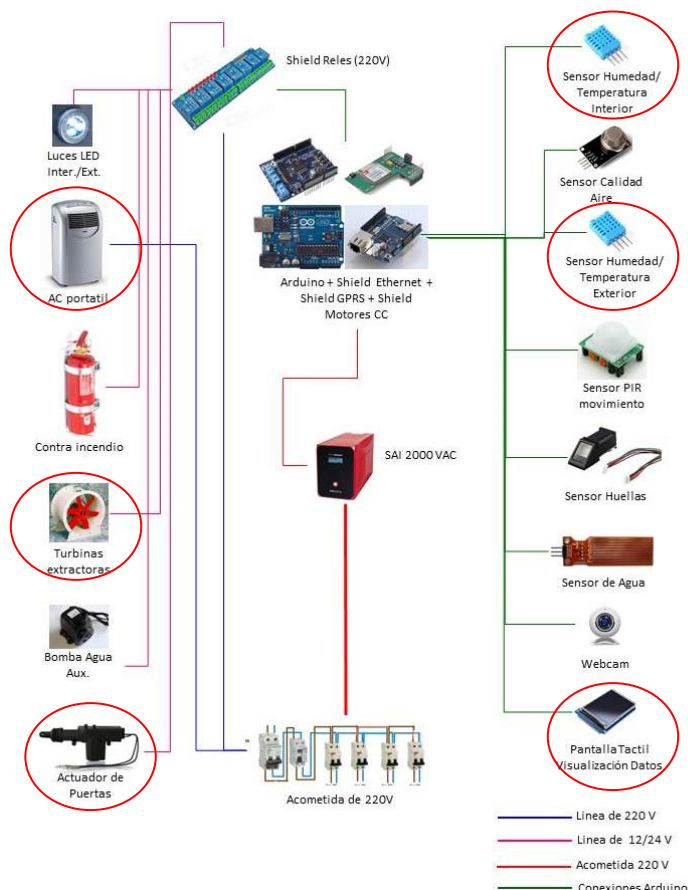




4.19 Sensor de temperatura, luz, reloj y pantalla de TFT

El sensor de temperatura exterior, tiene fijado unos valores de alta/baja. Dado que el emplazamiento del CPD está en un lugar muy azotado por el calor en verano, el sensor exterior será una medida preventiva para preparar al armario frente a una subida de temperaturas, disparando el AC auxiliar, y en la medida de los posible evitar el anterior algoritmo:

“... disparo de los actuadores que abren las puertas y ponen en marcha los ventiladores extractores, para cuando se restablezca una situación controlada, invertir todo este proceso ...”

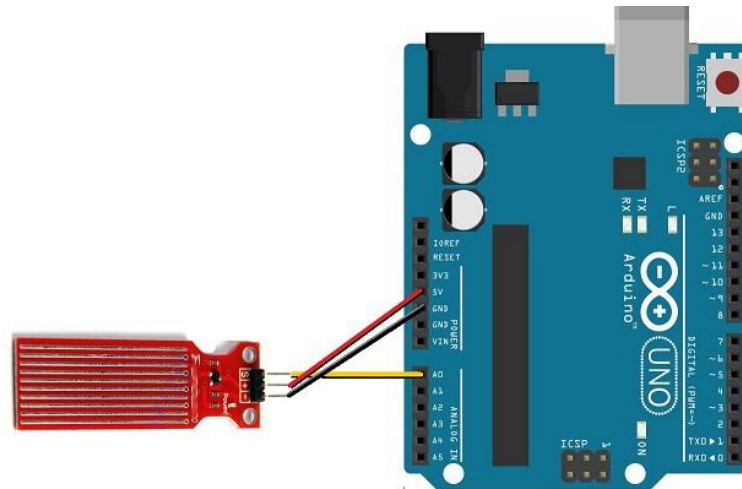


Hasta este momento han entrado en juego los elementos señalados del diseño de domotización que vamos planteando.

Estos elementos, son susceptibles, como es obvio, de que vuelvan a actuar en otro escenario que requiera otro algoritmo diferente de ejecución.

#### 4.4 Sensor de agua

El conexionado seria el de la siguiente ilustración:



4.20 Conexionado del sensor de agua

El CPD de la BV-SSPA, como hemos venido hablando, esta emplazado en un lugar físico que no es el mas adecuado, no solamente para el control de temperatura y consecuente protección de los equipos informáticos albergados en él, sino también afecta a la instalación de los AC.

Dada su ubicación, no hay desagüe en el edificio donde conectar el agua residual de los AC. Así pues, nos vimos obligados llevarla al exterior del edificio, con el uso de una bomba principal, y una secundaria, para crear redundancia, en caso de avería de la principal.

La bomba secundaria la pone en marcha Arduino y el sistema de domotización del CPD.

El sensor estará alojado dentro del receptáculo donde va cayendo el agua del AC principal. En caso de que esta bomba, por el motivo que sea, deja de funcionar, el sensor de agua lo detectara antes de que rebose, y pondrá en funcionamiento, mediante el shield de relés, la segunda bomba.

De esta manera evitaremos el rebose de la bomba principal y posible corto circuitos debido al agua que empezaría a resbalar por la pared del armario, donde esta ubicada la bomba principal de evacuación.

El sketch es muy sencillo:

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  Serial.println(analogRead(A0));
}
```



Faltarían las líneas que mostrase el valor de humedad en la pantalla (TFT) de nuestro sistema de domotización. Estas líneas son similares a la que muestra los valores de temperatura, humedad y lúmenes, del sensor de luz, por lo que obviaremos estas líneas.



4.21 Bomba de agua del AC principal

#### 4.5 Sensor de huellas

El armario se abre haciendo uso del teclado que viene incorporado, donde además podemos configurar el mismo (teclado), silenciar las alarmas y ver una breve información (temperatura/humedad) en una pantalla de cristal líquido.

Estas medidas de temperatura/humedad tienen un gran error de lectura. Por eso que se ha dotado de medición de temperatura/humedad, al sistema de domotización del CPD.

Para activar el teclado, es necesario hacerlo con una llave de seguridad, que opera con una cerradura electrónica. Muchas veces la llave no se lleva consigo y obliga desplazamientos para localizarla.

Se ha implementado el detector de huellas, y dadas de alta todas las huellas del personal de la BV-SSPA que tiene acceso al armario informático. De esta manera resulta mas cómodo activar la consola de manejo del CPD.



4.22 Consola del CPD de la BV-SSPA

En círculo rojo, la llave electrónica para activar el teclado y poder empezar a operar con la consola.

Debajo del botón de disparo manual de contraincendios (en amarillo), se instalara el detector de huellas.

La salida de este detector estará conectada en paralelo a la cerradura electrónica, de manera que enviara un pulso "high" (1), cuando la lectura de huella sea correcta y acto seguido activara la consola original del CPD, de la que no se trata de prescindir.

Futuras mejoras, entre ellas, pantalla táctil de 8", creara físicamente un (interfaz hombre-máquina, HMI), instalado, por encima de la consola original del CPD, que nos permitirá hacer las

actuales funciones del mismo, mas las que incorpora nuestro sistema de domotización. Esta versión física, tendrá su versión web, por medio de Xively y una aplicación web.

La librería de este sensor está en la siguiente página:

<https://github.com/adafruit/Adafruit-Fingerprint-Sensor-Library>

#### 4.6 Sensor de calidad del aire

El sensor de calidad del aire, estará ubicado dentro del armario informático, con la misión de medir la presencia de humos.

Su misión será disparar los sistemas contraincendios en caso de un nivel de humos crítico, que se fijara en el sketch de Arduino. A través de Shields de reles, se activara la electroválvula de las bombonas de extinción de fuego, alojadas dentro del CPD.

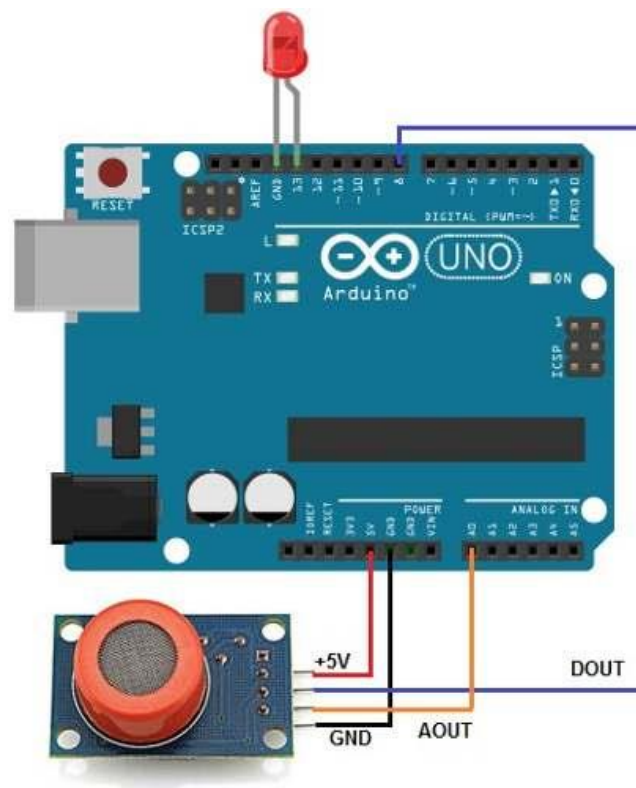


4.23 Medidor de calidad del aire

Su ubicación dentro del CPD, debe de ser estratégica para evitar que se dispare sin necesidad o lo haga demasiado tarde.

El CPD de la BV-SSPA tiene dos módulos, por lo que se ha optado usar uno en cada uno.

El conexionado de este sensor a Arduino, no es nada complejo. La siguiente ilustración muestra como hacerlo:



4.24 Conexión del sensor de calidad de aire

## 4.7 Sensor PIR y webcam

El sensor, es un detector de movimiento.

Los sensores infrarrojos pasivos (PIR) son dispositivos para la detección de movimiento. Son baratos, pequeños, de baja potencia, y fáciles de usar. Por esta razón son frecuentemente usados en juguetes, aplicaciones domóticas o sistemas de seguridad.

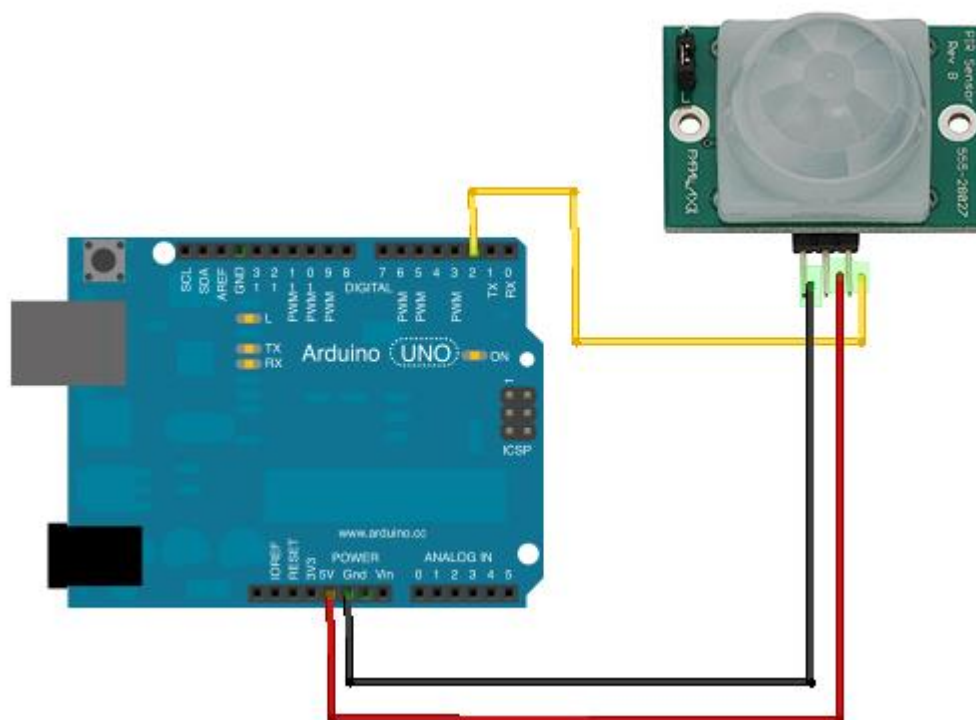
Los sensores PIR **se basan en la medición de la radiación infrarroja**. Todos los cuerpos (vivos o no) emiten una cierta cantidad de energía infrarroja, mayor cuanto mayor es su temperatura. Los dispositivos PIR disponen de un sensor piezo eléctrico capaz de captar esta radiación y convertirla en una señal eléctrica.

En realidad **cada sensor está dividido en dos campos** y se dispone de un circuito eléctrico que compensa ambas mediciones. Si ambos campos reciben la misma cantidad de infrarrojos la señal eléctrica resultante es nula. Por el contrario, si los dos campos realizan una medición diferente, se genera una señal eléctrica.

De esta forma, si un objeto atraviesa uno de los campos se genera una señal eléctrica diferencial, que es captada por el sensor, y se emite una señal digital.

El otro elemento restante para que todo funcione es **la óptica del sensor**. Básicamente es una cúpula de plástico formada por lentes de fresnel, que divide el espacio en zonas, y enfoca la radiación infrarroja a cada uno de los campos del PIR.

El conexionado tampoco es complejo:



4.25 Conexión del sensor Pir a Arduino

El código necesario para realizar la lectura es simple. Simplemente leemos la salida del PIR, y hacemos funcionar la webcam mientras la señal esté activa.

```
1  const int WebcamPin= 13;
2  const int PIRPin= 2;
3
4  void setup()
5  {
6    pinMode(WebcamPin, OUTPUT);
7    pinMode(PIRPin, INPUT);
8  }
9
10 void loop()
11 {
12   int value= digitalRead(PIRPin);
13   if (value == HIGH)
14   {
15     digitalWrite(WebcamPin, HIGH);
16     delay(50);
17     digitalWrite(WebcamPin, LOW);
18     delay(50);
19   }
20   else
21   {
22     digitalWrite(WebcamPin, LOW);
23   }
24 }
```

Este sensor empezará a estar operativo después de las 18:30 horas, cuando la oficina queda vacía de personal.

Será un sistema de alarma, para tener control de las personas que acceden al lugar físico donde está ubicado el CPD. En caso de ser de noche, este sensor Pir trabaja con el sensor de luminosidad exterior, que a unos determinados niveles de lúmenes, activará los focos de leds, para iluminar la zona de grabación.

El módulo de webcam de Arduino, trae una bahía para instalar una tarjeta micro SD. Es capaz de direccionar hasta 32GB, memoria suficiente para varios días, antes de proceder a su descarga.

El conjunto de:

- luces LEDS
- Sensor Pir
- Webcam
- Sensor de luminosidad



Podrá ser activado a voluntad del personal encargado del CPD. Tal situación puede ser necesaria cuando ante la evidencia que arroja un sensor o conjunto de sensores, se hace necesaria una conexión en tiempo real, para poder ver que está sucediendo en el CPD y su entorno.

Esto se lograría con el interfaz hombre-máquina, HMI, remoto que se implementará via web y Smartphone.

## 5. CONEXIÓN DE ARDUINO A LA PLATAFORMA XIVELY

### 5.1 ¿Qué es Xively?

**Xively** (originalmente llamada Pachube, más tarde Cosm y finalmente Xively, debido a una marca previamente registrada) consiste en un **servicio online** desarrollado específicamente para el **Internet de las Cosas** (Internet Of Things – IOT).

La plataforma permite **publicar** los **datos** recogidos por distintos sensores (como pueden ser sensores de humedad, temperatura, gases, luminosidad, radiación, etc.) mediante gráficas en tiempo real y widgets.

### 5.2 Registro en la plataforma XIVELY

Lo primero que deberemos hacer será registrarnos en la plataforma. Para ello creamos una cuenta desde el siguiente enlace: <https://xively.com/signup>. A continuación, verificaremos la cuenta accediendo al enlace facilitado por el correo electrónico que recibamos.

## Sign Up

For a free Developer Account

[Looking for Commercial Service?](#)

**Username**  
only letters, numbers and underscores

  
**Email**

**Password**

Tell us a little bit about yourself

**What describes you best?**

**Full Name**

**Organization** optional

**Country**

**ZIP Code / Postcode**

**Time zone**

**Areas of Interest**  
pick one or more

- Commercial
- Government
- Personal
- Education

**Communication Settings**  
 Xively may contact me directly

 Sign Up

By signing up you agree to the [Terms of Use](#)

Una vez registrados se nos presentará la página “**Test Drive**”. Pulsamos sobre “**Skip the Test Drive**”.

## Welcome to Xively

### Take the Test Drive

A 5 minute tutorial to get familiar with all the basics from connecting one device to building interconnected systems and apps.



### What you'll learn

- ✓ How to setup a **development device** through the Xively workbench.
- ✓ How to make your phone a **connected device**.
- ✓ See Xively **bi-directional communication** in action, controlling your devices and communicating between devices and apps.

Try Xively today, all you need is a smartphone and a computer!

Take Test Drive >

[Skip the Test Drive](#)

Already familiar with Xively?

Se nos presentará por tanto, la página “**Development Devices**”. Aquí es donde añadiremos y controlaremos los datos de nuestra Raspberry Pi, pero para ello necesitaremos registrar primero el dispositivo, por lo que pulsaremos sobre “**+ Add Device**” para acceder a la siguiente página.

## <> Add Device

The Xively Developer Workbench will help you to get your devices, applications and services talking to each other through Xively. The first step is to create a development device. Begin by providing some basic information:

**Device Name**

**Device Description** optional

**Privacy** You own your data, we help you share it. [more info](#)

**Private Device**  
You use API keys to choose if and how you share a device's data.

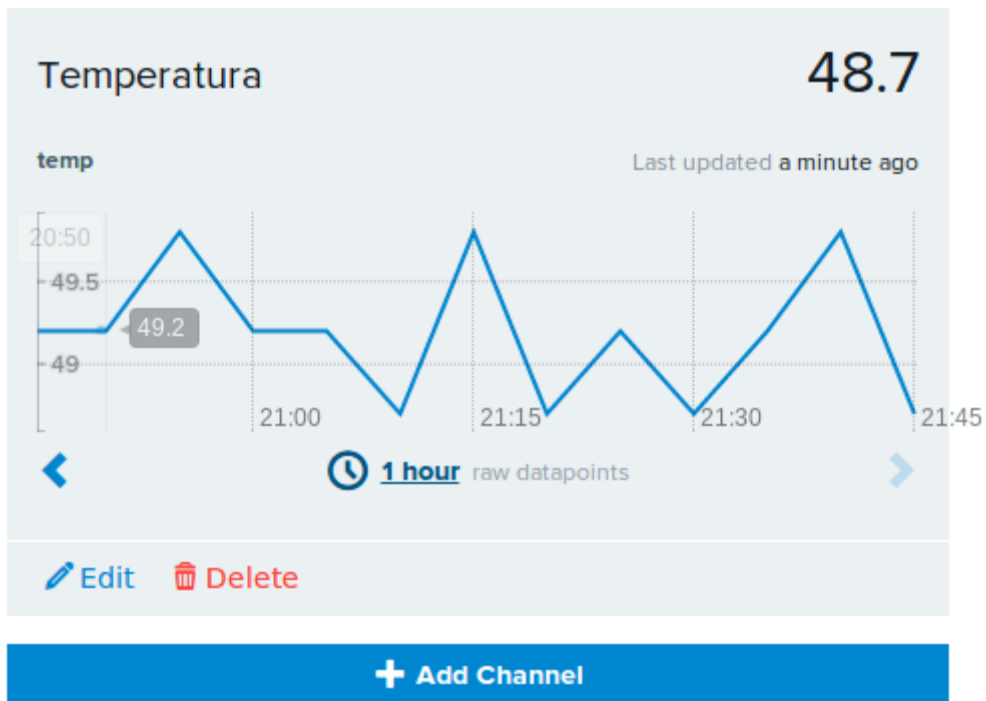
**Public Device**  
You agree to share a device's data under the [CC0 1.0 Universal license](#). The Device's data is indexed by major search engines, and its Feed page is publicly viewable.

Escribiremos un **nombre para el dispositivo** y opcionalmente una descripción del mismo. Seleccionaremos la privacidad del dispositivo como **“Private Device”** y finalmente pulsaremos sobre **“Add Device”**.

Se nos presentará la **página principal de nuestro dispositivo**. Aquí es donde se registrarán y mostrarán los distintos canales de datos recogidos por los sensores.

**¡IMPORTANTE!** Tomad nota de los valores que aparecen asociados a **FEED ID** y **API KEYS**, ya que son los datos necesarios para conectar el script Python con la plataforma Xively.





Hay que escribir un código que muestra las bases de cómo obtener datos desde arduino y enviarlos a Xively mediante la librería Python que ofrecen. Desde aquí las oportunidades son infinitas. Por ejemplo:

- Se podrían conectar varios sensores reales a ARduino mediante los pines E/S y enviar los datos recogidos a Xively.
- Configurar tu script Python como un servicio para que se ejecute justo cuando Arduino se encienda.
- Usar tu Arduino para suscribirte a un FEED existente a través del protocolo MQTT y tener el control de activar “cosas” en respuesta a los cambios que se produzcan en ese FEED o manualmente.

## 6. MEJORAS A DESARROLLAR

### 6.1 Mejoras en via de desarrollo

1. 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.
2. Crear una pequeña aplicación web alojada en un servidor local que de acceso a distintas visualizaciones y diseño de interfaz de usuario para controles de los elementos. (Matlab).
3. 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.
4. Creación de una página web que muestre valores locales de todos los sensores en tiempo real para ser visualizada desde una pantalla o monitor. Así mismo dicha página web recogerá la monitorización de todos los actuadores y componentes electrónicos conectados al sistema de domotización de Arduino, conociendo en todo momento el estado de los mismo y pudiendo operar sobre ellos a voluntad.
5. Migración de la placa Arduino Uno a la Mega 2560 o a la DUE, ambas de mayor capacidad en cuanto a E/S analógicas y digitales y memoria junto a la velocidad del microprocesador.
6. Comunicación entre las placas Arduino para el intercambio de información. Para ampliar el proyecto y dejarles puertas abiertas para su crecimiento y con el fin de no sobrecargar una placa Arduino, se baraja la conexión de placas Arduino simulando un sistema basado en bus.
7. Almacenamiento de datos (data logging).

## Bibliografía

1. Sanchez Torrecilla, Julio Ruben; Sogorb Devesa TC. Sistema de monitorización y telegestión remota basado en Arduino para Smart Buildings. Valencia; 2014.
2. Castro Dominguez A. Sistema de control de temperatura a traves de Arduino y la tecnologia GPRS/GSM. 2012.
3. <https://xively.com/> (última consulta Julio 2015)
4. <http://www.arduino.cc/es/> (última consulta Agosto 2015)
5. <http://arduino.cc/es/Main/Software> (última consulta Agosto 2015)
6. <http://botscience.wordpress.com/2012/06/05/historia-de-arduino-y-su-nacimiento/> (última consulta Agosto 2015)
7. <http://arduino.cc/es/Guide/Environment> (última consulta Octubre 2015)
8. <http://www.cooking-hacks.com/index.php/catalogsearch/result/?q=gprs> (última consulta Octubre 2015)
9. <http://arduino.cc/es/Main/Software> (última consulta Octubre 2015)
10. <http://arduino.cc/es/Tutorial/Blink?from=Tutorial.BlinkingLED> (última consulta Octubre 2015)