ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Trabajo Fin de Grado

Plataforma para el control energético en aplicaciones domóticas

(Platform for energetic controlling over home automation applications)

Para acceder al Título de

Graduado en Ingeniería de Tecnologías de Telecomunicación

Autor: Pablo Gómez Ortiz

10 - 2017

- Puede variarse el ancho del lomo con sólo aumentar o disminuir el tamaño de letra. En la muestra el tamaño de letra es Univers 10.

TÍTULO	**Plataforma para el control energético en escenarios domóticos**			
AUTOR	**Pablo Gómez Ortiz**			
DIRECTOR	**Alberto Eloy Garcia Gutierrez**			
TITULACIÓN	**GRADUADO EN INGENIERIA DE TECNOLOGIAS DE TELECOMUNICACIÓN**	**FECHA**	10-2017	**TOMO I DE I**

Tabla de contenido

Lista de figuras

Lista de tablas:

Resumen

El uso de la tecnología en el hogar permite ayudar, en la medida de lo posible, a mejorar el confort, reducir del gasto, aumentar la eficiencia, etc. La idea de automatizar los procesos más comunes de nuestro hogar no es nueva, pero este trabajo plantea una solución basada en dispositivos comerciales, sin acudir a soluciones propietarias. El sistema implementado se caracteriza por su sencillez y por utilizar elementos que pueden encontrarse fácilmente, y que, comparados con los sistemas que se comercializan actualmente, presenta gran número de ventajas. Las tecnologías utilizadas, además, son parte de las denominadas open source, porque su filosofía permite una rápida adaptación a los cambios gracias la comunidad que tienen detrás ellas, como es el caso de Arduino, Raspberry y Bluetooth.

El uso combinado de tecnologías open-source y de componentes de uso generalizado, el diseño propuesto se caracteriza por reducir los costes de instalación, pero manteniendo las capacidades de adquisición de información desde la sensórica instalada, para su tratamiento y gestión de acuerdo con las necesidades del hogar. Todo ello en base a una mínima intervención en las infraestructuras de la vivienda, mediante el uso de comunicaciones inalámbricas Bluetooth, centralizadas para el procesado y gestión en la parte central del sistema, la Raspberry Pi, la cual es la encargada de almacenar esta información en una base de datos, para su posterior exposición a través de en un entorno web que permita la interpretación de los datos de una forma más sencilla y cómoda para el usuario final.

Abstract

The use of technology in the home allow to help, as much as possible, to improve comfort, reduce expenditure, increase efficiency, etc. The idea of automating the most common processes in our home is not new, but this work poses a solution based on commercial devices, without resorting to proprietary solutions. The implemented system is characterized by its simplicity and by using elements that can be easily found, and that, compared to the systems that are commercialized now, displays great number of advantages. The technologies used, in addition, are part of the so-called open source, because its philosophy allows a rapid adaptation to the changes thanks to the community behind them, such as Arduino, Raspberry and Bluetooth.

The combined use of open-source and widely used components, the proposed design is characterized by reducing installation costs but maintaining the capacity of acquiring information from the installed sensor, for its treatment and management according to the needs of the home. All this based on a minimal intervention in the infrastructure of home, using Bluetooth wireless communications, centralized for processing and management in the central part of the system, the Raspberry Pi, which is responsible for storing this information in a database, for later exposure through a web environment that allows the interpretation of data in a simple and convenient way for the end user.

1. Introducción:

La idea de poder controlar y monitorizar la mayoría de las acciones cotidianas que se realizan en el día a día de un hogar no es nueva, pero nunca se ha estado tan cerca de soluciones realistas como ahora, el momento en el que la era digital ha llegado a los hogares.

Una de las acciones más comunes, sobre todo en las últimas horas del día, cuando el usuario se recoge en la tranquilidad de su hogar, es el encendido y apagado de luces, seguida por el encendido y apagado de diferentes electrodomésticos, relacionados precisamente con el confort, como por ejemplo calefacciones y aires acondicionados. Así, la gestión de dichas acciones, no solo en su ejecución, sino en la monitorización en el tiempo de las mismas empieza a ser fundamental. Quizá, a corto plazo, no sea un dato interesante el conocer qué estancias son las más utilizadas y cuáles son los patrones de funcionamiento de los dispositivos conectados. Sin embargo, una vez analizada la información, recopilada en históricos de la vivienda, permitirá decidir, por ejemplo, qué tipo de tecnología usar en cada una de ellas, dependiendo del uso y la frecuencia, con el fin de permitir un máximo ahorro de energía.

Puesto que la iluminación no es la principal consumidora de la electricidad en el hogar, y que la mayor parte de los dispositivos utilizados en los hogares todavía no están "conectados", el uso de sensores especializados es fundamental, en principio, como medio de cuantificación de períodos de actividad y consumos energéticos asociados. la información recopilada por los sensores debe ser filtrada y analizada para así poder tomar decisiones inteligentes y eficientes. Este es el típico ejemplo esgrimido por las energéticas cuando quieren hablar de ahorro en sus tarifas, que sea el propio usuario el que decida el encendido de los electrodomésticos en horas en la cuales el coste energético sea menor.

Sin embargo, el despliegue de sensores dentro de la vivienda no debe venir solamente justificado por el ahorro energético, ya que, como se indicó anteriormente, el tiempo que se pasa en el hogar debe caracterizarse por un determinado grado de confortabilidad. Es ahí donde el control de parámetros como la temperatura, la humedad, índices de CO2, etc, resultan no solo recomendables, sino necesarios. Mediante historiales de medidas, particularizados para cada estancia individual, podrán tomarse decisiones acerca de necesidades de ventilación/calefacción, apagado automático de dispositivos, generación de alarmas, etc De esta forma, no solo se podrá conseguir una confortabilidad óptima con un consumo energético adaptado, sino que se añade la proactividad de los sistemas de medida como elementos de prevención y corrección.

1.1. Motivación y objetivos:

Hablar de domótica actualmente, pese a la proliferación de más y más soluciones y dispositivos, es todavía sinónimo de "alto precio" y "obras para su instalación".

El principal objetivo de este proyecto es demostrar cómo se puede llevar el mundo domótico al hogar, de una manera sencilla y barata.

De este objetivo primario se desprenden dos condiciones asociadas de por sí:

- El sistema debe ser barato: Si bien la palabra gratis sería la deseable, razonable es más realista. Se debe traducir en el uso dispositivos comunes, fáciles de adquirir y manejar, así como de tecnologías de programación basadas en software libre.
- El sistema debe de tener una instalación sencilla. Lo mismo que antes, la idea fundamental sería "sin obras", cuando la realidad nos lleva hacia soluciones muy particulares, lejos de esta idea. Se debe buscar una solución cercana al "Plug&Play", esto es, enchufar y utilizar. El uso de tecnologías inalámbricas parece una de las posibles soluciones, permitiendo intercomunicar estancias del hogar sin la necesidad de cableado adicional o modificar el cableado existente.

Con todo ello, el principal resultado que se espera obtener es reducir los costes del producto final, evitar el uso de productos licenciados o propietario y minimizar las intervenciones en la infraestructura del hogar. Para ello se propone el desarrollo de una prueba de concepto en la cual se demuestre el uso de una plataforma domótica que permita gestionar al menos el encendido de dispositivos, los consumos energéticos y el monitorizado de parámetros de confortabilidad (temperatura y humedad).

1.2. Organización del documento:

Este documento se organiza de la siguiente forma. Tras este apartado de introducción, en el capítulo 2 se define lo que se entiende por un sistema domótico y se describen las tecnologías más usadas en la actualidad, ofreciendo algunos ejemplos de ellas, para a continuación, en el capítulo 3, definir las tecnologías que finalmente se utilizan en el desarrollo del proyecto.

En el capítulo 4, se da una lista detallada de los diferentes elementos de los que se compone el proyecto, así como el diseño sobre el que se basará el sistema demótico resultante. Precisamente, en el capítulo 5 se explica con detalle la implementación del sistema elegido, y en el capítulo 6 se describe la prueba de concepto desarrollada para la demostración práctica de los resultados obtenidos en este Trabajo.

Por último, en el capítulo 7 se exponen las conclusiones obtenidas una vez finalizado el proyecto, dando detalle tanto de las ventajas como de los inconvenientes que conlleva este proyecto, así como las líneas que quedan abiertas para futuros desarrollos.

2. Estado del arte:

El término domótica viene de la unión de las palabras domus (que significa casa en latín) y tica (de automática, palabra en griego, 'que funciona por sí sola'). Sin embargo, actualmente se denomina Domótica al conjunto de sistemas que cuentan con la capacidad de automatizar una vivienda o edificación de cualquier índole. Mediante el uso de estos sistemas se posibilita así, por ejemplo, gestionar la energía, integrar sistemas de seguridad, influir en el bienestar y/o combinarse con sistemas de comunicación.

Estos sistemas pueden ser agregados por medio de redes de comunicación, tanto en interiores como en exteriores, que a su vez pueden utilizar medios cableados o inalámbricos. Además, este tipo de sistemas pueden ser controlados tanto desde el interior del hogar como desde el exterior, dotando de mayor accesibilidad al sistema. No es de extrañar que, por ejemplo, en Wikipedia se les defina como *"la integración de la tecnología en el diseño inteligente de un recinto cerrado"* (Wikipedia, s.f.)

La historia de la domótica tiene sus comienzos en la década de los 70, con la aparición de los primeros dispositivos para la automatización de edificios, aunque siempre en forma de pequeños proyectos pilotos mediante los cuales descubrir las ventajas de este tipo de tecnologías. De hecho, no fue hasta la década de los 80, cuando estos sistemas integrados se empezaron a comercializar, para más tarde adaptarse al ámbito doméstico en las ciudades.

Básicamente los desarrollos clásicos integran en un mismo sistema el ámbito eléctrico y electrónico, y han sido liderados por países como Alemania, Estados Unidos y Japón, especialmente tras el desarrollo de la informática. De hecho, el primer programa que utilizó la domótica fue el Save, desarrollado en Estados Unidos en 1984, que permitía lograr eficiencia y bajo consumo de energía en los sistemas de control del edificio. Las comunicaciones se realizaban a través del sistema X-10, protocolo de comunicaciones para la transmisión de datos sobre líneas de baja tensión, desarrollado en 1978 por Pico Electronics (Escocia), y que sigue siendo una de las tecnologías más utilizada dentro de la domótica.

Con la llegada de la estructuración del cableado en los edificios, la integración de estos sistemas resulta mucho más sencilla, dando lugar a los conocidos edificios inteligentes. Estas soluciones se basan en un alto grado de automatización, con el único objetivo de facilitar la vida a las personas que habitan este tipo de edificios.

En la actualidad son muchos y muy variados los sistemas domótico que podemos encontrar en el mercado, aunque pueden resumirse en dos grandes grupos: sistemas propietarios y sistemas abiertos.

Sistemas propietarios

El primero de los grupos son sistemas cerrados que requieren instalaciones muy complejas y con configuraciones cerradas, incluyendo la modificación de la infraestructura original. Esto suele conllevar un elevado sobrecoste para el cliente, tanto en su implementación como en su mantenimiento, así como la manipulación por parte de un instalador certificado por la marca.

Algunas de las marcas más conocidas son las siguientes:

Simon Vita

Este fabricante propone soluciones muy variadas y eficaces para cualquier tipo de escenario posible, todo con tecnología propia a un elevado coste. Tal y como se indica en su página web, sus principales ventajas son:

- Escalable, ya que permite la adaptación a las necesidades que requiere cada usuario, ofreciendo tanto soluciones locales para el control de pequeños escenarios en el hogar, como la creación de sistemas mucho mayores y más globales.
- Ampliable, permite la ampliación o modificación de sus sistemas según las necesidades específicas de la instalación sobre la que se implanta el sistema.
- Integración, permite el uso de varios sistemas interrelacionados, facilitando de este modo la gestión y la posibilidad de obtener una mayor eficiencia en el control energético.
- Seguro, cada módulo del sistema consta de memoria independiente, permitiendo de este modo que un fallo en uno de estos módulos, no afecte la operatividad del resto de la instalación.
- Visualización, aporta una gran variedad de soluciones para la visualización de sus funciones, pudiéndose adaptar a cualquier instalación. Tal y como se puede observar en la Fig. 1: Partes del sistema Simon Vita.

La tecnología aplicada se denomina LonWorks, siendo utilizada por muchas empresas en todo el mundo. Esta tecnología diseña soluciones para el transporte, la industria y la robotización de edificios y viviendas entre otros sectores.

Fig. 1: Partes del sistema Simon Vita

KNX

El estándar KNX basa su experiencia en más de 24 años en el mercado y en los sistemas predecesores de KNX: EIB (Bus de Instalación Europeo), EHS (European Home Systems Protocol) y BatiBUS. El medio de transmisión de KNX puede ser par trenzado, radiofrecuencia, línea de fuerza o IP/Ethernet. Los dispositivos conectados al bus pueden ser tanto sensores como actuadores y son utilizados para el control y la gestión de edificios de acuerdo con las aplicaciones que el mismo ofrece, tal y como se muestra en la Fig. 2, como son la iluminación, sistemas de seguridad, gestión energética, calefacción, sistemas de ventilación, sistemas de supervisión y señalización, interfaces a servicios y sistemas de control de edificios, etc. Todas estas funciones pueden ser controladas, supervisadas y señalizadas utilizando un sistema uniforme sin la necesidad de centros de control adicionales.

Fig. 2: Áreas de influencia de KNX

Es un sistema domótico modular, que puede ser ampliable de forma sencilla, pero con la necesidad de cambiar su programación. Además, proporciona el ahorro de tiempo y de costes, ya que está diseñado como un sistema de cableado muy sencillo y estructurado que facilita su instalación, tal y como se puede apreciar en la Fig. 3. No obstante, al tratarse de un sistema ya prediseñado, hace uso de una aplicación que permite realizar el proyecto. Al ser todo diseñado a medida, el coste asociado es elevado.

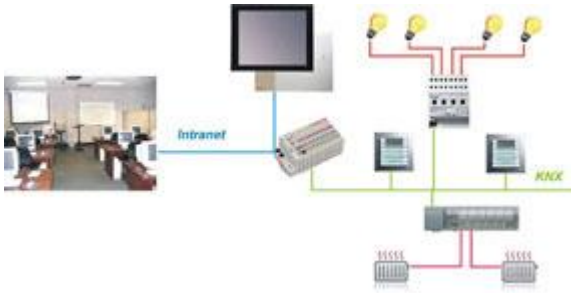


Fig. 3: Ejemplo de instalación KNX

X10

La tecnología X10 permite el control a distancia de dispositivos eléctricos que utilizan la línea eléctrica ya existente para transmitir señales de control. De esta forma es posible gestionar diferentes módulos conectados directamente a la instalación eléctrica del hogar. Los dispositivos X10 actualmente comercializados, algunos de los cuales se muestran en la Fig. 4, son para uso individual y en entornos domésticos con coberturas de hasta 250 m², debido a sus limitaciones en cuanto al ancho de banda y al número máximo de dispositivos a controlar, con un máximo de 256 dispositivos.

Fig. 4: Módulos de X10

No obstante, existen elementos de última generación que incorporan, entre otros, los protocolos X-10 extendidos, para proveer nuevas funciones a la comunicación, como la bidireccionalidad, solicitud de estados y comprobación de la correcta transmisión de las tramas.

Esta tecnología dispone de la gran ventaja de transmitirse en un medio que cualquier hogar ya tiene instalado, el cableado eléctrico, y donde no se pueda, a través de radiofrecuencia. Puesto que la instalación resulta muy sencilla, sin más que conectar a un enchufe, resulta una tecnología relativamente barata y apta para la gran mayoría del público. Cada módulo para poder activar o desactivar un enchufe esta alrededor de unos 30€ y el módulo encargado de encender y apagar las luces esta alrededor de unos 35€, por lo que la disminución en el coste de esta tecnología está en la facilidad a la hora de su instalación.

Sistemas abiertos

El segundo de los grupos son dispositivos individuales que realizan tareas muy concretas y, en su mayoría, el control lo lleva un dispositivo móvil. Este tipo de tecnologías puede gestionar ciertas áreas de nuestro sistema, pero la interconexión de todas ellas para conseguir un sistema domótico único y compacto es complicado tanto por la gestión de muchos elementos sencillos como por el coste de adquisición de todos los dispositivos necesarios.

Xiaomi

Es uno de los casos más significativos de este grupo y que más auge está teniendo en estos momentos. Son productos independientes que llevan a cabo el control de zonas muy específicas del hogar y que se han instalado en el mercado pese a pertenecer a una empresa muy reciente, en comparación con sus rivales, fundada el 6 de abril del 2010. Esta empresa se dedica al diseño, desarrollo y venta de un gran abanico de productos electrónicos, como por ejemplo móviles, ordenadores, etc.

El primer dispositivo que lanzó al mercado, fue un teléfono inteligente (Xiaomi mi 1), el cual fue puesto a la venta en agosto de 2011, desde entonces la marca ha ganado una importante cuota de mercado en China y se ha expandido a otros mercados. Xiaomi es la palabra china para "mijo". En el 2011, su CEO Lei Jun declaro que su significado no es único, relacionó la parte "Xiao" con el concepto budista de "un solo grano de arroz de un budista es tan grande como una montaña," dando a entender que Xiaomi tiene intención de trabajar desde pequeñas cosas, en vez de comenzar por la búsqueda de la perfección, mientras "mi" es un acrónimo para Mobile Internet (Internet móvil en español) y además para Misión Imposible, haciendo referencia a los obstáculos encontrados en los comienzos de la compañía. En 2012 Lei Jun dijo que el nombre es sobre revolución y ser capaz de llevar la innovación a una era.

En lo referente a la domótica, Xiaomi ha sacado unos cuantos productos siendo el más popular el kit "*Xiaomi Smart Home"*, este kit trae una serie de sensores y actuadores, como se puede ver en la Fig. 5: kit domótico de Xiaomi ,que se conectan al wifi del hogar y permiten realizar una serie de acciones con la ayuda de nuestro celular.

Estos elementos están enfocados para usos muy específicos, por ejemplo, en el caso del control de iluminación solo ofrecen una posibilidad cambiar todas las bombillas por sus propias bombillas, esto produce que solo se pueda colocar este tipo de bombilla en el sistema domótico. Es por eso que este sistema no está pensado para adaptarse a cualquier hogar, se permite cierto tipo de controles, pero de la forma predeterminada por el fabricante. Además, estos módulos no resultan del todo baratos, ya que el precio oscila entre los 10€ y 30€, pero con el inconveniente que no siempre ofrecerá todas las características que se desea en el sistema, debido al limitado abanico de sensores y actuadores, con un uso muy específico.

Fig. 5: kit domótico de Xiaomi

Con todo ello, este proyecto busca una solución hibrida, en la que el sistema pueda ser instalado de forma sencilla y a la vez reduzca los costes asociados en comparación con soluciones integrales como las existentes actualmente en el mercado.

3. Conceptos teóricos:

A continuación, se va a proceder a explicar las principales tecnologías que se han utilizado para la implementación del sistema propuesto. En primer lugar, el medio de comunicación, la tecnología Bluetooth, explicando su origen y sus mayores ventajas/desventajas. En segundo lugar, los sistemas de control, basados en la tecnología de placas programables Arduino. En último lugar y no por ello de menor importancia en este proyecto, los sistemas de gestión, basados en la tecnología de microcomputadores de Raspberry Pi, la cual será el cerebro de este desarrollo.

3.1. Bluetooth

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal (WPAN) que hace posible la transmisión de voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda ISM de los 2.4 GHz.

Fig. 6: El icono de Bluetooth

Los principales objetivos que reúne esta tecnología son los siguientes:

- Facilitar las comunicaciones entre equipos móviles.
- Deshacerse de los cables y conectores entre estos.
- Ofrecer la posibilidad de crear pequeñas redes inalámbricas y facilitar la sincronización de datos.

El nombre procede del rey danés y noruego Harald Blåtand, cuya traducción se derivó al inglés como Harald Bluetooth. Conocido por unificar las tribus noruegas, suecas-danesas y convertirlas al cristianismo. Este nombre fue propuesto por Jim Kardach, que desarrolló un sistema que permitiría a los teléfonos móviles comunicarse con los ordenadores y unificar la comunicación inalámbrica.

El logo de Bluetooth tal y como se observa en la Fig. 6: El icono de Bluetooth, combina las runas Hagall (ᚼ) y Berkana (ᛒ), que corresponden a las iniciales de Harald Bluetooth.

Entre sus principales ventajas podemos indicar la versatilidad al tratarse de una tecnología inalámbrica, la amplia utilización en todo el mundo de la misma, su utilización es muy sencilla y su control, ya que a pesar de ser capaz de intercambiar datos entre dispositivos permite controlar la seguridad en la transmisión gracias a un pin usado en la transferencia.

Entre sus desventajas se encuentra el uso extra de batería, una velocidad lenta de transmisión (comparado a otras tecnologías inalámbricas como el WiFi) y el limitado alcance (el rango oscila entre 1 m para dispositivos de clase 3 y 100 metros para los de clase 1).

Esta tecnología se encuentra actualmente en su versión 5, también denominada BLE (Bluetooth Low-Energy) y liberada recientemente, que incluye grandes mejoras tanto en velocidad, fiabilidad, cobertura y capacidad, pero especialmente en cuanto menores consumos energéticos, tal como indica ya el propio acrónimo.

Pese a que el nuevo estándar incluye nuevos mecanismos de comunicación basados en la capacidad BLE, en este desarrollo se ha decidido mantener el uso de los protocolos tradicionales, compatibilizando así el uso de dispositivos bluetooth de generaciones anteriores. Este es el caso del protocolo RFCOMM (Radio Frequency Communications) el cual es un protocolo serie que ofrece transporte de datos binarios y emula las señales de control de EIA-232 a través de la capa de banda base de Bluetooth. Este protocolo ofrece un flujo fiable de datos y sencillo para el usuario, similar a TCP.

Muchas aplicaciones Bluetooth utilizan RFCOMM debido a su amplio soporte y la posibilidad de encontrar API públicas en la mayoría de sistemas operativos. Además, toda aplicación que use el puerto serie para comunicarse, podrá ser portada a RFCOMM fácilmente.

Es por ello que, a pesar de ser un protocolo muy sencillo, es el elegido para el proyecto ya que nos ofrece la posibilidad de comunicar dispositivos sin usar protocolos de capas más altas.

3.2. Arduino

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de la electrónica en proyectos multidisciplinares, cuyo logo es el que se muestra en Fig. 7: El logo de Arduino.

Fig. 7: El logo de Arduino

Arduino define tanto el software como el hardware, siendo su principal diferencia con respecto a otras placas y microcontroladores. Esto significa que los entornos de desarrollo, y lenguaje de programación de Arduino y las placas en las que se ejecutan han sido desarrollados conjuntamente, por lo que se asegura tanto la compatibilidad como la sencillez de desarrollo sobre ellas. Además, esto permite que toda la plataforma, incluyendo sus componentes de hardware (esquemáticos) y Software, sean liberados con licencia de código abierto, esto es, permite total libertad de acceso a ellos.

El hardware de Arduino es básicamente una placa con un microcontrolador. Un microcontrolador es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. Un microcontrolador incluye en su interior tres principales unidades funcionales: unidad central de procesamiento, memoria y periféricos de entrada/salida, estas unidades también se incluyen en otros dispositivos como un ordenador. Asimismo, posee un puerto de conexión USB desde donde se puede alimentar la placa y establecer comunicación con el ordenador.

Por otro lado, el software de Arduino es un IDE, entorno de desarrollo integrado (siglas en inglés de Integrated Development Environment), esto es, un programa informático compuesto por un conjunto de herramientas de programación. De hecho, el IDE de Arduino es un entorno de programación que ha sido empaquetado como una aplicación; el cual, consiste en un editor de código, un compilador, un depurador con su propia interfaz gráfica. Además, incluye las herramientas para cargar, una vez compilado el programa, en la memoria flash del hardware.

La primera placa Arduino fue introducida en 2005, ofreciendo un bajo costo y facilidad de uso para todos los públicos. Con el fin de desarrollar proyectos interactivos con su entorno gracias al uso de actuadores y sensores. A partir de octubre de 2012, se incorporaron nuevos modelos de placas de desarrollo que usan microcontroladores ARM de 32 bits, que coexisten con los originales modelos que integran microcontroladores AVR de 8 bits. ARM y AVR no son plataformas compatibles en cuanto a su arquitectura y por lo que tampoco lo es su set de instrucciones, pero se pueden programar y compilar bajo el IDE predeterminado de Arduino sin ningún cambio, una de las grandes ventajas de esta tecnología.

Las placas Arduino están disponibles de dos formas: ensambladas o en forma de kits "Hazlo tú mismo" (por sus siglas en inglés "DIY"). Los esquemas de diseño del Hardware están disponibles bajo licencia Libre, con lo que se permite que cualquier persona pueda crear su propia placa Arduino sin necesidad de comprar una prefabricada. Adafruit Industries estimó a mediados del año 2011 que, alrededor de 300 000 placas Arduino habían sido producidas comercialmente y en el año 2013 estimó que alrededor de 700 000 placas oficiales de la empresa Arduino estaban en manos de los usuarios.

Arduino se puede utilizar para desarrollar objetos interactivos autónomos o puede ser conectado a software tal como Adobe Flash, Processing, Max/MSP, Pure Data, etc. Una tendencia tecnológica es utilizar Arduino como tarjeta de adquisición de datos desarrollando interfaces en software como JAVA, Visual Basic y LabVIEW.6 Las placas se pueden montar a mano o adquirirse. El entorno de desarrollo integrado libre se puede descargar gratuitamente.

3.3. Raspberry Pi

Raspberry PI es un ordenador de placa reducida (en inglés: Single Board Computer o SBC) de bajo coste, una especie de ordenador de tamaño reducido, del orden de una tarjeta de crédito, desarrollado en el Reino Unido por la Fundación Raspberry PI (Universidad de Cambridge) en 2011, con el fin de estimular la enseñanza de la informática en las escuelas, aunque su comercialización no empezó hasta el año 2012.

Básicamente la idea que hay detrás de este proyecto es la de llevar hasta los usuarios un ordenador básico., esto es, libre de todos los accesorios que se pueden eliminar sin ver afectado su funcionamiento básico.

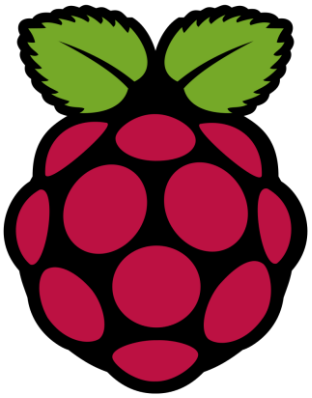


Fig. 8: El logo de Raspberry Pi

El nombre de Raspberry surge debido a las empresas tecnológicas ya existentes, cuyos nombres provienen de frutas. Por ejemplo, en Reino Unido están Apricot (Albaricoque) y Tangerine (Mandarina) y como una de las marcas por excelencia está la estadounidense, Apple. Raspberry (Frambuesa) un nombre de fruta que aún no estaba cogido en aquellos momentos y, además, tal y como comentaba Eben Upton, fundador de *Raspberry Pi fundation* una entrevista para Make, hace honor a la frase "blowing a raspberry", algo así como hacer burla con la lengua, de ahí el actual logo, mostrado en la Fig. 8: El logo de Raspberry Pi.

Pese a no indicarse expresamente si es hardware libre o con derechos de marca, en su web oficial explican que disponen de contratos de distribución y venta con dos empresas, pero al mismo tiempo cualquiera puede convertirse en revendedor o redistribuidor de las tarjetas Raspberry Pi, por lo que se entiende que es un producto con propiedad registrada, manteniendo el control de la plataforma, pero permitiendo su uso libre tanto a nivel educativo como particular.

En cambio, el software sí es open source, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian, aunque permite usar otros sistemas operativos, incluido una versión de Windows 10. En todas sus versiones incluye un procesador Broadcom, una memoria RAM, una GPU, puertos USB, HDMI, Ethernet (El primer modelo no lo tenía), 40 pines GPIO y un conector para cámara. Ninguna de sus ediciones incluye memoria, siendo esta en su primera versión una tarjeta SD y en ediciones posteriores una tarjeta MicroSD.

Una diferencia importante entre la Raspberry Pi y el PC de escritorio o portátiles, aparte de su tamaño y su coste, es el sistema operativo (el software que permite controlar el ordenador) que utiliza.

La mayoría de los PC's y portátiles actualmente en el mercado, funcionan con dos de los sistemas operativos más comunes en la actualidad: Microsoft Windows o Apple OS. Ambas plataformas son de código cerrado, creados bajo un ambiente reservado

utilizando técnicas patentadas. Los usuarios pueden obtener el software final, pero nunca podrán saber los entresijos de este software.

La Raspberry Pi, por el contrario, está diseñada en sus orígenes para ejecutar el sistema operativo GNU/Linux. A diferencia de Windows o Apple OS, Linux es de código abierto. Lo que implica el poder descargar el código fuente del sistema operativo por completo y hacer los cambios que uno vea oportunos. Este espíritu de desarrollo de código abierto permite a Linux ser modificado con celeridad para poder ejecutarse sobre la Raspberry Pio cualquier otra tecnología que salga en un futuro, un proceso conocido como portabilidad.

La fundación 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. Otros lenguajes también soportados son Tiny BASIC, C, Perl y Ruby.

Actualmente existen todos los siguientes S.O. compatibles, ya sean S.O. completos o versiones modificadas/portadas, que funcionan perfectamente en el dispositivo Raspberry Pi:

Tabla 1: Sistemas operativos para Raspberry Pi

Sistemas operativos completos	
Linux	Android97
	Arch Linux ARM
	Debian Whezzy Soft-Float
	Firefox OS
	Gentoo Linux98
	Google Chromium OS
	Kali Linux
	Open webOS99
	PiBang Linux100
	Pidora
	QtonPi
	Raspbian102
	Slackware ARM
	Plan 9 from Bell Labs103 104
Risc OS 52	
Unix	FreeBSD105
	NetBSD106 107

Distribuciones ligeras multipropósito:

- Moebius, distribución ligera ARM HF basada en Debian que usa el repositorio de Raspbian y que cabe en una tarjeta SD de 1GB, usa pocos servicios y está optimizada para usar poca memoria.
- Squeezed Arm Puppy, una versión de Puppy Linux (Puppi) para ARMv6 (sap6) específicamente para Raspberry Pi108
- Minibian, distribución ligera basada en Raspbian.

Distribuciones ligeras de único propósito:

- Instant WebKiosk, sistema operativo con solo un navegador
- IPFire
- Micro Elastix, solución de código abierto para comunicaciones unificadas109
- OpenELEC (Multimedia)
- Raspbmc (Multimedia)
- Xbian (Multimedia)

4. Conceptos prácticos:

Este proyecto tiene sus orígenes en la propia curiosidad en las nuevas tecnologías de programación. lo que llevó a descubrir la plataforma Raspberry, ya en sus primeros estadios, allá en 2011, 2012. A partir de ahí, irían apareciendo otras facilidades, como es el caso de Arduino y nuevas formas de acceder a la electrónica fundamental, en forma de sensores y actuadores de bajo coste.

¿Por qué Raspberry Pi?

Inicialmente asumida como plataforma perfecta para implementar un entorno de aprendizaje ideal, resulta la solución perfecta para el despliegue de aplicaciones web. Como en cualquier otro sistema operativo tradicional, una vez instalado el famoso paquete AMP, o en nuestro caso más concreto **LAMP** al tratarse de **Linux,** el sistema permite la instalación de las herramientas fundamentales para obtener un servidor de servicios **A**pache, **M**ySQL y **P**HP.

Unas breves definiciones extraídas de la enciclopedia virtual, Wikipedia:

"El servidor HTTP Apache es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras, que implementa el protocolo HTTP/1.1 y la noción de sitio virtual. Cuando comenzó su desarrollo en 1995 se basó inicialmente en código del popular NCSA HTTPd 1.3, pero más tarde fue reescrito por completo."

"MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo,1 y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web."

"PHP es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos." PHP es reemplazado a veces por Perl o Python, y el acrónimo se mantiene.

Estas herramientas ofrecen tanto la posibilidad de albergar archivos, creando así un servidor en el que albergar contenido tanto de forma local como hacia la red, gracias a Apache. Por otra parte, MySQL permite la gestión de grandes cantidades de información gracias a la gestión de bases de datos. Y por último PHP nos permite la creación de contenido de una forma dinámica.

¿Por qué Arduino?

Al tiempo que se comprobaba in situ el potencial de RasPi, la casualidad nos llevó a conocer al siguiente elemento, Arduino. Su aparición coincide con un momento en el que ya sonaba el concepto de "Domótica en casa". De hecho, fue cuestión de semanas el juntar ambas tecnologías, dando lugar a la primera versión de la comunicación vía cable de la Raspberry Pi y un Arduino, de forma que la primera se encarga de ofrecer

servicios basados en web y el segundo de la actuación con relés, permitiendo interactuar con una bombilla, como simulando previa de la mayoría de elementos a gestionar en un hogar.

Sin embargo, con el tiempo y la experiencia adquirida, surge la pregunta acerca de la redundancia que provoca usar una Raspberry Pi y un Arduino como dispositivos controladores, ya que en ambos casos presentan puertos GPIO's, que son los que permiten interactuar con todo tipo de señores y actuadores. La respuesta es sencilla, realmente no es necesario ambos dispositivos para el mero control de sensores. Sin embargo, el problema surge por la naturaleza de la comunicación entre elementos. Por defecto, toda la comunicación se realiza por medios cableados, lo que complica el despliegue de los sensores y actuadores. Esto implica que siempre es deseable colocar el Arduino o la RasPi en las proximidades de los sensores/actuadores. Esto no siempre es posible, y cuando lo es, requiere hacer uso de varios elementos de control que a su vez debería estar cableados entre sí. Como solución se plantea la idea de usar bluetooth como sustituto del cable de interconexión. De esa forma el Arduino, que resulta bastante más económico, puede ser colocado, por ejemplo, uno por estancia, Por su parte, dada la limitación en la capacidad de cómputo de los Arduino, la gestión del flujo de información entre los sensores, actuadores y el Arduino, debería de ser centralizada, para su posterior procesamiento, por lo que una Raspberry Pi puede actuar como elemento gestor, estableciendo sendos enlaces bluetooth con cada uno de los Arduinos controlados.

Con todo ello, el sistema propuesto, como solución viable, es el uso de una Raspberry Pi como gestor del sistema domótico, que permite interconectar varias estancias, realizando las tareas de Master. Como tal, se encarga de gestionar los demás dispositivos llevando un control unidireccional mediante la realización de una serie de peticiones (polling) que los Slaves reciben antes de generar la respuesta. Por supuesto, los Slaves son implementados en los Arduinos, no estando cerrado al uso de otros modelos similares. Los slaves tienen instalados una serie de sensores y actuadores que permiten recoger, por ejemplo, la temperatura y humedad del ambiente, las intensidades que circulan por la instalación e interactuar con actuadores para controlar distintos dispositivos del hogar. Una vez que el Arduino recopila esta información, es mandada mediante RFCOMM a la Raspberry Pi, para almacenar todos los datos recibidos en la base de datos del sistema de gestión.

Una vez se puede gestionar la mayoría de los elementos, el reto es poder gestionar toda la información generada, ya que, como ya es conocido, es que la información normalmente se acumula llegando a ser, en muchos casos hasta un problema, al no poderse obtener ninguna utilidad. Como primera aproximación, la representación de los datos recogidos en forma de gráficas, permite al usuario final poder obtener información de una forma sencilla y rápida, como paso previo a un posterior análisis de los resultados.

4.1. Diseño

En este apartado se va a tratar todo lo relacionado con la parte del Arduino. Primero se explica de que partes consta su diseño, dando una breve explicación de cada uno. En segundo lugar, veremos los esquemas de conexión de todos los elementos anteriormente mencionados y como son sus interconexiones. En tercer lugar, se presentará el código usado para que el Arduino recopile información y permita responder a las peticiones que le hagamos. Por último, se expondrá el protocolo diseñado para estandarizar estas peticiones y así en cierto modo poder tener recopiladas todas las funcionalidades del mismo. El protocolo se ha diseñado de una forma muy completa, ya que se documentan una serie de posibles peticiones que no van a ser usadas en este proyecto.

El diseño de los "Slaves" siempre por norma general va a contener elementos comunes en todas sus posibles configuraciones, en primer lugar, el elemento principal un Arduino, bien sea en su modelo Arduino Uno R3 o Arduino Micro, en segundo lugar, otro de los elementos comunes es un módulo bluetooth HC-05 y por ultimo un sensor de temperatura DHT11.

Además, dependiendo del caso se puede añadir tanto, sensores de intensidad para ver los consumos en las líneas o electrodomésticos que deseemos como relés digitales para actuar sobre diversos elementos del hogar.

La lista de los elementos usados en el desarrollo:

Raspberry Pi 2 Modelo B:

Este modelo fue lanzado en 2014, fue el primer modelo que no tenía instalado el mismo procesador usado en los tres anteriores: se sustituyó por el modelo BCM2836. Lo que conllevo a pasar de un núcleo a cuatro, y de 700MHz a 900MHz. No obstante, emplea la misma gráfica, la VideoCore IV. Además, dobla la cantidad de memoria RAM, pasando de 512MB a 1GB, esta memoria está compartida con la gráfica. Incluye 40 pines GPIO, y mantiene los cuatro puertos USB como el modelo anterior, pero suprime la conexión RCA, tal y como se muestra en la Fig. 9: Raspberry Pi Versión 2 Modelo B.

Fig. 9: Raspberry Pi Versión 2 Modelo B

Arduino Uno R3:

Arduino Uno es un microcontrolador basado en el ATmega328P (hoja de datos). Tiene 14 pines digitales de entrada / salida (de los cuales 6 se pueden utilizar como salidas PWM), 6 entradas analógicas, un cristal de cuarzo de 16 MHz, una conexión USB, una conexión de alimentación, una cabecera ICSP y un botón de reinicio, dispuesto tal y como se puede apreciar en la Fig. 10: Imagen del Arduino Uno R3. Las características de este elemento las enumeramos en la Tabla 2: Características de Arduino Uno R3.

Fig. 10: Imagen del Arduino Uno R3

Tabla 2: Características de Arduino Uno R3

Microcontrolador	ATmega328P
Tensión de funcionamiento	5V
Tensión de entrada (recomendada)	7-12V
Tensión de entrada (limite)	6-20V
Pines digitales de E / S	14 (6 salidas PWM)
Canales PWM	6
Canales de entrada analógicos	6
Corriente DC por pin E/S	20 mA
Corriente DC por pin 3.3V	50 mA
Memoria Flash	32 KB (ATmega328P) 0.5 KB por bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Velocidad de reloj	16 MHz
LED_BUILTIN	13
Longitud	68.6 mm
Ancho	53.4 mm
Peso	25 g

Arduino Micro:

El Micro es un microcontrolador basado en el ATmega32U4 (hoja de datos), desarrollado conjuntamente con Adafruit. Cuenta con 20 pines digitales de entrada/salida (de los cuales 7 se pueden utilizar como salidas PWM y 12 como entradas analógicas), un oscilador de cristal de 16 MHz, una conexión micro USB, una cabecera ICSP y un botón de reinicio como se muestra en la Fig. 11: Imagen de Arduino Micro. Las características de elemento son las que se describen en la Tabla 3: Características de Arduino Micro

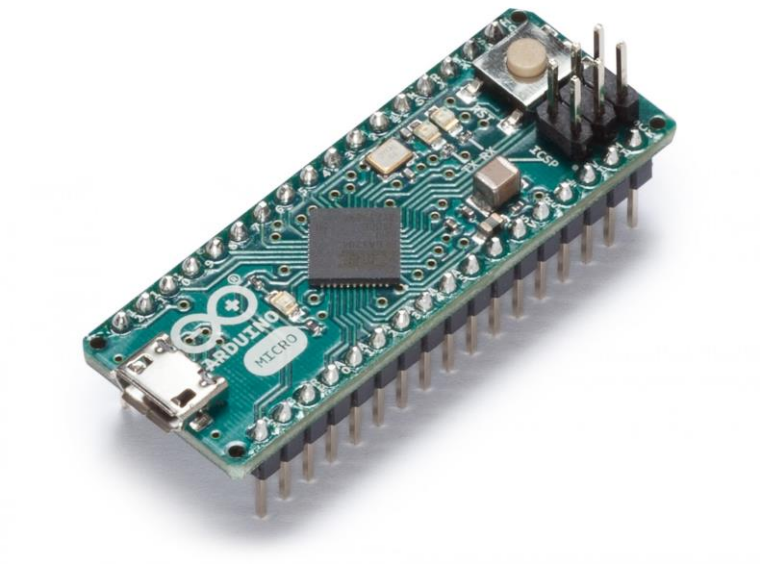


Fig. 11: Imagen de Arduino Micro

Tabla 3: Características de Arduino Micro

Microcontrolador	ATmega32U4
Tensión de funcionamiento	5V
Tensión de entrada (recomendada)	7-12V
Tensión de entrada (limite)	6-20V
Pines digitales de E / S	20
Canales PWM	7
Canales de entrada analógicos	12
Corriente DC por pin E/S	20 mA
Corriente DC por pin 3.3V	50 mA
Memoria Flash	32 KB (ATmega32U4) ,4 KB por bootloader
SRAM	2.5 KB (ATmega32U4)
EEPROM	1 KB (ATmega32U4)
Velocidad de reloj	16 MHz
LED_BUILTIN	13
Longitud	48 mm
Ancho	18 mm
Peso	13 g

En el caso del Arduino Micro, su coste en la página oficial, este entorno a los 18 euros. Pero se puede encontrar por precios inferiores en otras tiendas online.

Módulo HC-05:

El módulo Bluetooth HC-05 con capacidad para gestionar el modo master y el modo slave por configuración. Es un módulo sencillo, tal y como se muestra en la Fig. 12: Modulo HC-05, que permite dotar a nuestro sistema de una interfaz bluetooth con la que comunicarnos con otros dispositivos.

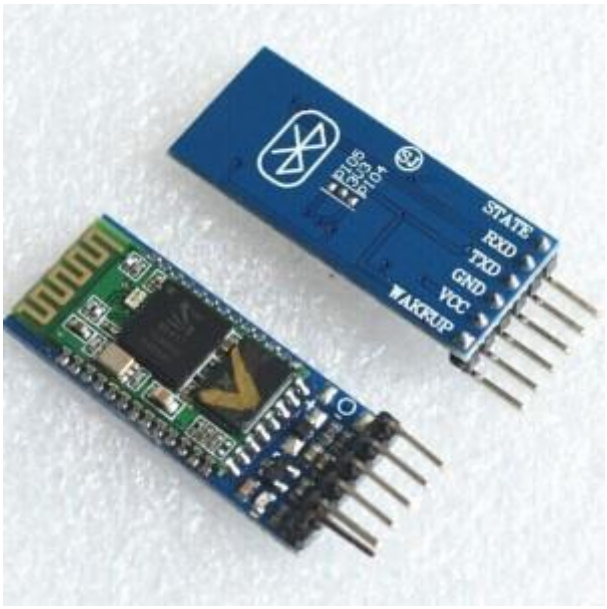


Fig. 12: Modulo HC-05

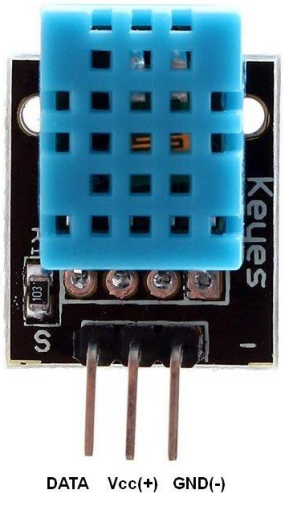
Este módulo es configurable a través de comandos AT, esta configuración será explicada más adelante.

Este módulo se trata de un módulo con Bluetooth v2. Pero con el tiempo han ido apareciendo módulos que soportan el protocolo Bluetooth V4.0 o Low Energy, tales como los modelos HC-08 y HC-10.

El nuevo Bluetooth 4.0 es un nuevo protocolo diseñado pensando en disminuir todo lo posible las necesidades de energía de los dispositivos que lo usan, y sobre todo de la propia comunicación que tradicionalmente ha sido de consumo insaciable.

Se le suele llamar también BLE por Bluetooth Low Energy, o simplemente Bluetooth LE. Mejora bastante el consumo previo, pero la distancia también disminuye. Es por esta razón y ya que este módulo estará conectado a la red garantizando un suministro ilimitado que se ha optado por una versión inferior, que pese a suponer un mayor coste energético, nos ofrece una mayor distancia de funcionamiento.

DTH-11:



Este sensor se caracteriza por tener la señal digital calibrada por lo que asegura una alta calidad y una fiabilidad a lo largo del tiempo, ya que contiene un microcontrolador de 8 bits integrado. Está constituido por dos sensores resistivos (NTC y humedad). Tiene una excelente calidad y una respuesta rápida en las medidas. Puede medir la humedad entre el rango 20% – aprox. 95% y la temperatura entre el rango 0ºC – 50ºC.

El protocolo de comunicación es a través de un único hilo (protocolo 1-wire), tal y como se observa en la Fig. 13: Módulo DTH-11, identificándose como el pin DATA. Por lo tanto, hace que la integración de este sensor en los proyectos sea rápida y sencilla. Además, presenta un tamaño reducido, un bajo consumo y la capacidad de transmitir la señal hasta 20 metros de distancia.

Fig. 13: Módulo DTH-11

Relé:

Un relé es un interruptor que se puede activar mediante una señal eléctrica. En su versión más simple es un pequeño electro-imán que cuando se excita mueve la posición de un contacto eléctrico de conectado a desconectado o viceversa.

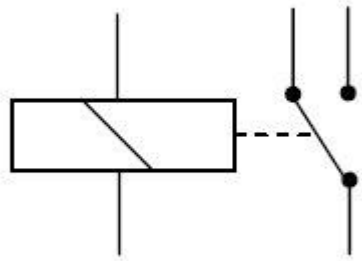


Fig. 14: Esquema simple de un relé

El símbolo del relé, mostrado en la Fig. 14: Esquema simple de un relé, muestra la bobina y en este caso, un accionador que conmuta entre dos contactos, pero también existen relés de múltiples contactos. Mediante una señal de control de poca intensidad que excite la bobina podemos conmutar grandes tensiones o intensidades.

En el caso de este proyecto usamos, el siguiente dispositivo.

Este módulo de relé permite una amplia gama de microcontroladores como Arduino, AVR, PIC, ARM con salidas digitales para controlar cargas más grandes y dispositivos como motores de CA o CC, electroimanes, solenoides y bombillas incandescentes.

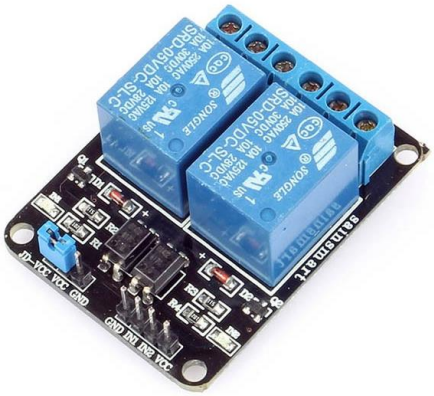


Fig. 15: Imagen de un relé

Este módulo está diseñado para ser capaz de controlar 2 relés. El dispositivo utiliza un QIANJI JQC-3F relé de alta calidad con carga nominal 7A / 240VAC, 10A / 125VAC, 10A / 28VDC.El estado del relé es indicado individualmente por led.

Fig. 16: Disposición de los pines

'COM'- Perno común

'NC'- Normalmente cerrado, en cuyo caso 'NC' está conectado con COM cuando INT1 está a nivel bajo de tensión y desconectado cuando INT1 está a nivel alto de tensión.

‘NO’- Normalmente Abierto, en cuyo caso ‘NO’ se desconecta de COM1 cuando INT1 está a nivel bajo de tensión y conectado cuando INT1 está a nivel alto de tensión.

El terminal 2 es similar al terminal 1, excepto que el puerto de control es INT2, tal y como se puede apreciar en la Fig. 16: Disposición de los pines.

Sensor corriente:

EL módulo sensor de corriente ACS712 con 50 kHz de ancho de banda. cuenta con un tamaño compacto y estable.

Fig. 17: Sensor de corriente ACS712

Este módulo puede leer tanto tensiones positivas como negativas en VDC y VAC, es un sensor de corriente tanto alterna como continua, que permite medir la intensidad eléctrica que atraviesa un conductor. Podemos emplear el ACS712 junto con un procesador como Arduino para medir la intensidad o potencia consumida por una carga.

El ACS712 contiene en su interior un sensor hall de precisión y bajo offset junto con un canal de conducción localizado cerca de la superficie del integrado. Cuando la corriente fluye por el canal de cobre genera un campo magnético que es detectado por el sensor Hall y es convertido en una tensión.

La salida del sensor indicada como “OUT” en la Fig. 17: Sensor de corriente ACS712, es una tensión proporcional a la corriente, y altamente independiente de la temperatura.

La mayor desventaja del ACS712 es que es un sensor intrusivo, es decir, es necesario insertarlo en un conductor lo cual puede suponer que tengamos que cortar un cable, pero ya que en la mayoría de casos vamos a querer controlar la activación o desactivación de este cable no supone una gran desventaja en nuestro proyecto. Por eso se ha elegido este sensor y no otros del tipo no intrusivo.

Dependiendo del tipo de medición que se quiera realizar se podrá optar por unos de sus 3 modelos, solo difieren en la intensidad máxima que soportan.

Tabla 4: Modelos de ACS712

Max Intensidad	Sensibilidad	Tensión salida	Resolución
±5A	185 mV/A	1,575V a 3,425V	26mA

±20A	100 mV/A	0,5V a 4,5V	49mA
±30A	66 mV/A	0,52V a 4,48V	74mA

La tensión en la salida del ACS712 es proporcional a la intensidad que atraviesa el sensor. Se incluye un offset de 2.5V de forma que la referencia está centrada, lo que permite medir intensidades positivas y negativas.

5. Implementación:

Una vez identificados todos los elementos que van a formar parte de nuestro proyecto, vamos a proceder a identificar como son las conexiones que se deben realizar en cada caso. Pese a estar en una versión final, todos los elementos conectados a la vez, primero se explica cada instalación por separado y en último lugar se mostrará una foto real con el resultado. No se explica todo el proceso con esa foto ya que resultaría más difícil de comprender.

5.1. Esquemas

En primer lugar, el punto de salida será el esquema del Arduino Uno R3, se usa este modelo como referencia, debido a que resulta más amplia la esquematización.

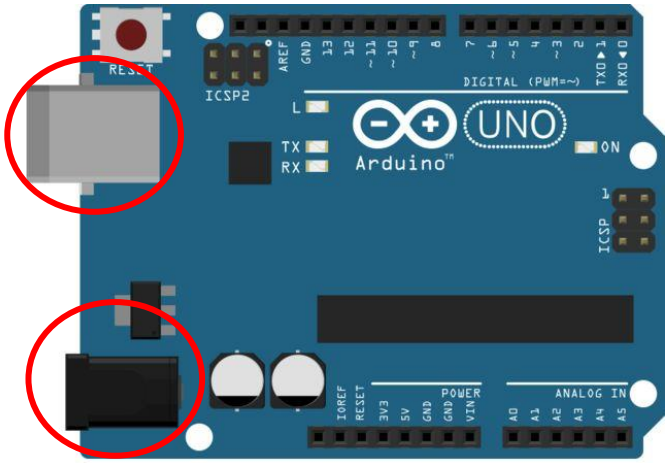


Fig. 18: Entradas de alimentación de Arduino Uno R3

El Arduino solo necesita alimentación y se puede suministrar por cualquiera de los dos conectores que están resaltados en la Fig. 18: Entradas de alimentación de Arduino Uno R3.

En primer lugar, se explicará la instalación de modulo bluetooth HC-05. Para su puesta en marcha, primero deberemos configurar el módulo por comandos AT. Para que el HC-05 entre en modo comandos AT, requiere que cuando se encienda el módulo, el pin KEY este a tensión alta. Por eso se conecta la tensión Vcc del módulo Bluetooth al pin 8 del Arduino, tal y como aparece indicado en la Fig. 19.

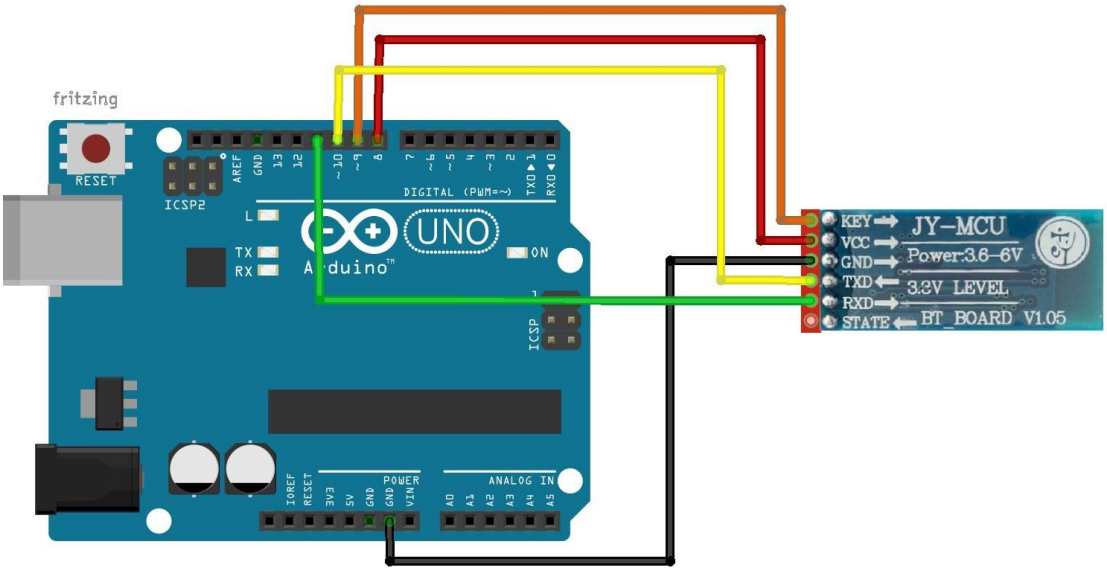


Fig. 19: Conexión de HC-05 con Arduino

El consumo del módulo es mínimo y el Arduino es capaz de alimentarlo sin problemas, por eso el módulo se encenderá cuando pongamos HIGH (tensión alta) en el pin 9. Esto permitirá poner en HIGH el pin digital 8, al iniciar nuestro programa y después levantar el pin 8, de este modo cuando arranque entrará sin más en el modo de comandos AT.

Una vez se encuentra en este modo, en el Arduino deberá correr el siguiente programa:

```
#include <SoftwareSerial.h>

SoftwareSerial BT1(10, 11); // RX | TX

void setup()

   { pinMode(8, OUTPUT);         // Al poner en HIGH forzaremos el modo AT

     pinMode(9, OUTPUT);

     digitalWrite(9, HIGH);

     delay (500);               // Un tiempo de espera prudencial

     Serial.begin(9600);

     Serial.println("Iniciando el módulo HC-05");

         // Cometarios para ayudar a seguir el programa

     digitalWrite (8, HIGH);    //Enciende el modulo

     Serial.println("Comando AT:");

     BT1.begin(9600); // Puede que se deba cambiar hasta dar con la correcta

   }

void loop()
```

```
{  if (BT1.available())

          Serial.write(BT1.read());

   if (Serial.available())

          BT1.write(Serial.read());

}
```

Para empezar, escribiendo en mayúsculas AT e Intro, se debería recibir una respuesta de OK en la consola. En este caso es que se ha conectado con éxito.

Algunos de los comandos que se pueden usar:

- AT+VERSION, Requiere la versión del Firmware.
- AT+NAME, Informa del nombre que tiene asignado el módulo. Debería devolver un mensaje del tipo NAME=HC-05, indicando que se llama HC-05.

- AT+NAMEXXXX, configura el nombre que se quiere presentar cuando alguien nos localice:

- AT+BAUD, permite solicitar la velocidad a la que está programado el módulo para hablar con Arduino, y AT+BAUDX, Fija la velocidad de comunicación entre el módulo y la consola de acuerdo a la siguiente tabla:

  1 => 1200bps
  2 => 2400bps
  3 => 4800bps
  4 => 9600bps (Default)
  5 => 19200bps
  6 => 38400bps
  7 => 57600bps
  8 => 115200bps

- AT+PIN, Solicita el PIN actual y en la consola se verá: PIN=1234 o similar.

- AT+PINXXXX, configura el número de identificación personal, que se requerirá para establecer la vinculación

  El PIN es el número de identificación personal, que se usa al conectar el módulo. El PIN es de 4 dígitos siempre

- AT+ROLE Informa de si está configurado como Maestro 1, o como esclavo 0.

- AT+ROLE1 Configura el módulo como Master.

- AT+ROLE0 Configura el módulo como Slave.

Estos son los comandos más usados.

Una vez realizada esta primera configuración, el módulo ya almacena esta configuración. No es necesario seguir conectando el pin Key, pero el resto de las conexiones se mantienen así. Como se observará en el sketch completo ese código mantiene la mayoría de las líneas usadas, salvo que ahora ya este módulo permitirá comunicarnos con la Raspberry Pi.

El segundo módulo es el DHT-11, su conexión es de las más sencillas, tal y como se muestra en la Fig. 20: Conexión de DHT-11 con Arduino. Además, si se utiliza su librería la obtención de los datos se realiza con un par de instrucciones.

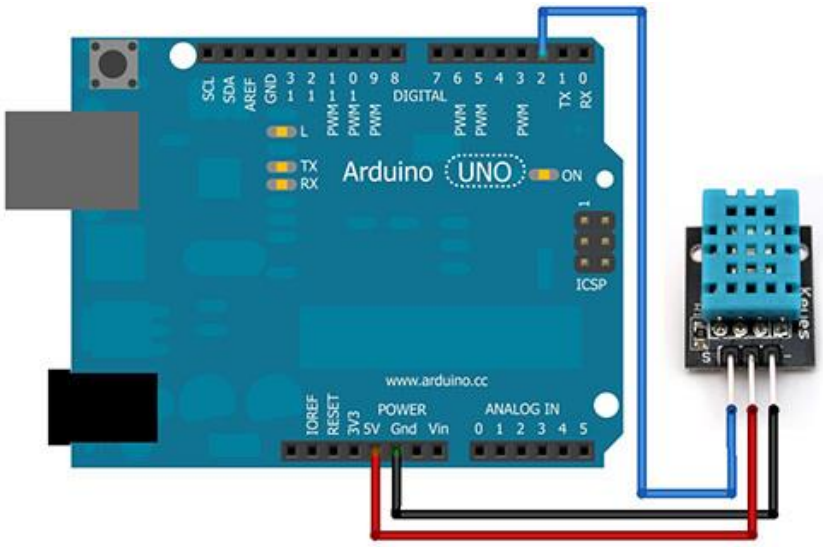


Fig. 20: Conexión de DHT-11 con Arduino

En primer lugar, tenemos que descargar una librería para manejarlos cómodamente e importarla.

Importamos la librería DHT que nos pondrá:

```
#include <DHT.h>
```

Se define una instancia del sensor en la cual se declara el pin al que el módulo está conectado y que modulo es.

```
#define DHTPIN 2     // Pin sensor de temperatura y humedad

#define DHTTYPE DHT11   // El módulo que es

DHT dht(DHTPIN, DHTTYPE);
```

Luego en la inicialización, se inicializa el sensor:

```
void setup(){

  dht.begin();

}
```

Ahora para leerlo solo necesitamos las siguientes instrucciónes:

```
temp1 = dht.readTemperature();

humi1 = dht.readHumidity();
```

El tercer modulo es el relé, el cual se debe alimentar dando tensión al Vcc y poniendo a tierra el pin GND. Una vez hecho esto solo queda conectar un E/S del Arduino al pin del relé que se quiere controlar. El resultado sería tal y como se muestra en la siguiente figura. (Fig. 21)

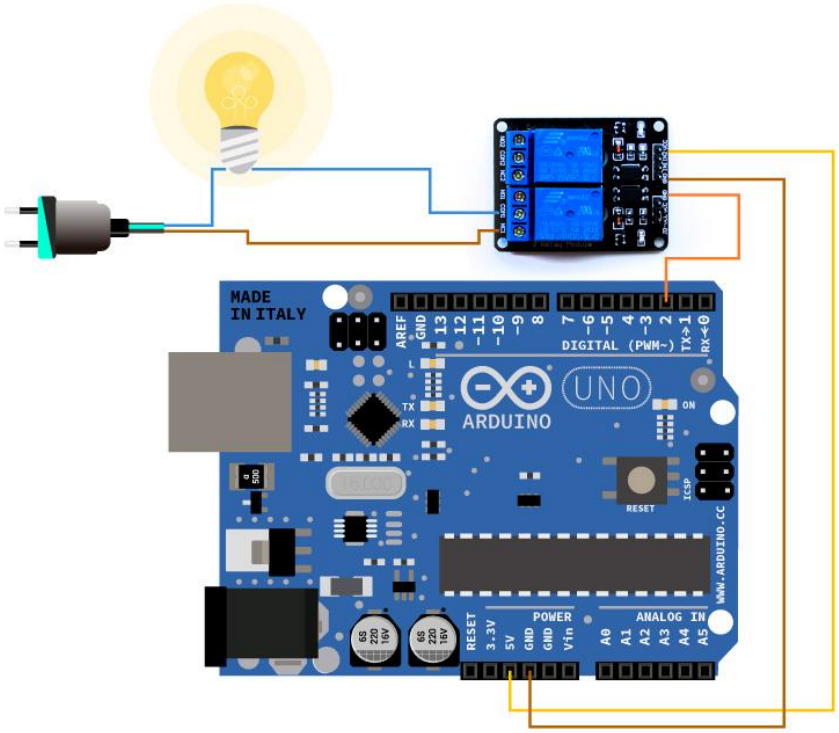


Fig. 21: Conexión de Relé con Arduino

Una vez hecho esto, solo quedar declarar el pin al cual se ha conectado el relé:

```
const int relePin1 = 2;
```

Luego se inicializa el sensor con la siguiente instrucción:

```
void setup(){

  pinMode(relePin1, OUTPUT);

}
```

Por último para actuar sobre el sensor solo se deben utilizar los siguientes comandos:

```
digitalWrite(relePin1, HIGH);       //Activa el relé
```

```
digitalWrite(relePin1, LOW);        //Desactiva el relé
```

El cuarto modulo, el sensor de intensidad ACS712 solo consta de tres pines como se puede apreciar en la Fig. 22, dos para la alimentación y el tercero saca la tensión, la cual se convertirá en intensidad con una sencilla transformación.

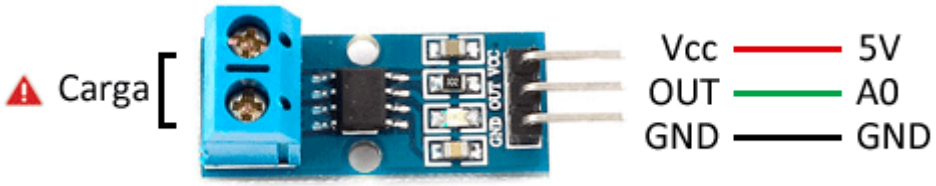


Fig. 22: Conexiones de ACS712

Por lo tanto la conexión con el Arduino resulta de la siguiente manera, tal y como se puede observar en la Fig. 23

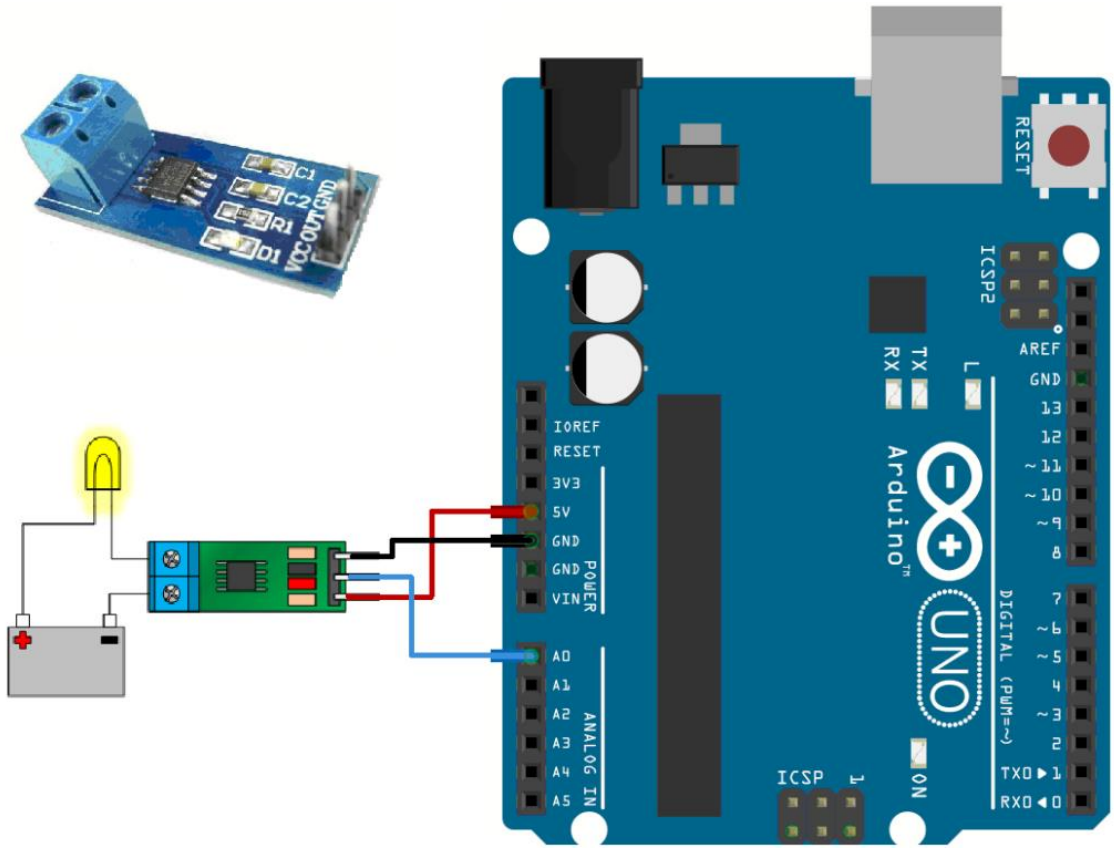


Fig. 23: Conexión de ACS712 con Arduino

Una vez se ha conectado de esta forma, se debe aplicar esta transformación:

$$V = 2.5 + K \cdot I \Rightarrow I = (V - 2.5)/K$$

En el código se realiza de la siguiente forma, primero se debe definir esa sensibilidad (K), la cual se obtiene dependiendo del modelo. En este caso la sensibilidad de 0.185 mV/A, en este caso se pone en (mV) porque se va trabajar en esta unidad.

```
float Sensibilidad=185; //sensibilidad en Voltios/Amperio para sensor de 5A.
```

Una vez definida esta variable, se procede a obtener el valor de la entrada analógica A0.

```
voltajeSensor= (analogRead(A0)/ 1023.0)*5000; //lectura del sensor  en mV

I=((voltajeSensor-2492.6)/Sensibilidad); //Ecuación  para obtener la corriente

Sensor += I;    //Se suma para promediar las muestras.

if (i >= 128) {          //Se lleva a cabo un promediado cada 128 muestras.

  SensorPro = Sensor/i; //Se divide el total entre el nº de muestras

  if(SensorPro < 0){SensorPro=SensorPro*(-1);} //Valor absoluto

  Sensor = 0;   //Se inicializn de nuevo contadores

  i=0;

}

i++;
```

Una vez hecho esto, en la variable SensorPro se obtendrá un valor promediado de las lecturas realizadas por el sensor y este dato se podrá pasar a la Raspberry Pi.

El resultado final con todas las conexiones anteriormente mencionadas y constituyendo un ejemplo de prueba, el cual tiene el módulo bluetooth, el sensor de temperatura, el sensor de intensidad y dos relés conectados resultaría una cosa así:

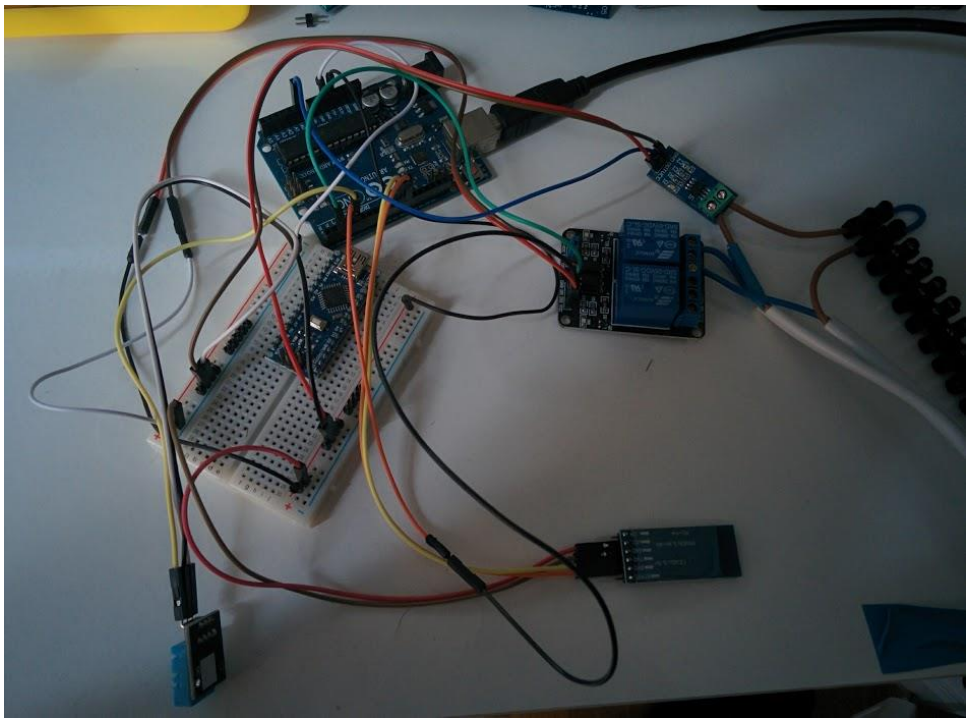


Fig. 24: Imagen del montaje completo

5.2. Sketch

Una vez se han enumerado los distintos componentes, de los que va a constar la instalación y se han explicado sus configuraciones. El programa completo el cual estaría compilado es el Arduino sería el siguiente:

```
#include <SoftwareSerial.h>
#include "DHT.h"

#define DHTPIN 4     // what pin we're connected to
#define DHTTYPE DHT11   // DHT 11

SoftwareSerial BT1(10, 11); // RX | TX

float Sensibilidad=185; //sensibilidad en Voltios/Amperio para sensor de 5A
const int relePin1 = 2; //El pin para el Relé 1
const int relePin2 = 3; //El pin para el Relé 2
char incomingByte[6];      // La variable en la que se almacenan la peticion
char Serial_Arduino = 'A';     // La letra que determina el modulo
int estado;
int i = 0;
int index = 0;
int P_Log = 0;
int j = 0;
float Sensor = 0;
float SensorPro = 0;
float voltajeSensor=0;
float I=0;
float temp1 = -99;
float humi1 = -99;


DHT dht(DHTPIN, DHTTYPE);

void setup()
{
  BT1.begin(9600);
  pinMode(relePin1, OUTPUT);
  dht.begin();
  Serial.begin(9600);
  Serial.print("Iniciando servicio ...\n");
  digitalWrite(relePin1, HIGH);
  estado = 1;
}

void loop()
{
  voltajeSensor= (analogRead(A0)/ 1023.0)*5000; //lectura del sensor  en mV
  I=((voltajeSensor-2492.6)/Sensibilidad); //Ecuación  para obtener la corriente
  Sensor += I;

  if (i >= 128) {
    SensorPro = Sensor/i;
    if(SensorPro < 0){SensorPro=SensorPro*(-1);}
    Sensor = 0;
    i=0;
  }

  temp1 = dht.readTemperature();

  humi1 = dht.readHumidity();

  if (BT1.available() > 0) {
    index = 0;
    memset(incomingByte, 0, sizeof(incomingByte));

    while(BT1.available() > 0){
```

```
    char aChar = BT1.read();

    incomingByte[index] = aChar; // Se añade el carácter a la variable
    index++;                     // Sumamos uno al puntero
    incomingByte[index] = '\0';  // NULL terminación para la variable
  }

  if(sizeof(incomingByte)>=3 && incomingByte[0] == Serial_Arduino ){

    switch (incomingByte[1]) {

      case 'A':          //Actuador Relé 1
        switch(incomingByte[2]){
          case '1':
            digitalWrite(relePin1, HIGH);
            BT1.println("OK");
            estado = 1;
            break;
          case '0':
            digitalWrite(relePin1, LOW);
            BT1.println("OK");
            estado = 0;
            break;
         case 'C':
          if(estado == 0){
            digitalWrite(relePin1, HIGH);
            estado = 1;
          }else{
            digitalWrite(relePin1, LOW);
            estado = 0;
          }
            BT1.println("OK");

            break;
        }
        break;

      case 'T':                   //Petición de sensor de Temperatura
        switch(incomingByte[2]){
          case '1':
            BT1.println(temp1);
            break;
          case '2': //Para futuros sensores
            BT1.println("N/D");
            break;
        }
        break;

      case 'H':                   //Petición de sensor de Humedad
         switch(incomingByte[2]){
          case '1':
            BT1.println(humi1);
            break;
          case '2': //Para futuros sensores
            BT1.println("N/D");
            break;
        }
        break;

      case 'W':                   //Petición de sensor de Intensidad
        switch(incomingByte[2]){
          case '1':
            BT1.println(SensorPro,3);
            break;
          case '2': //Para futuros sensores
            BT1.println("N/D");
            break;
        }
        break;
    }

  }else if(incomingByte[0] != Serial_Arduino){
```

```
      if(incomingByte[0]=='L' && incomingByte[1]=='O' && incomingByte[2]=='G'){
       if(incomingByte[3] == '1'){
         P_Log = 1;
         BT1.println("Log Activado");
       }else if(incomingByte[3] == '0'){
         P_Log = 0;
         BT1.println("Log Desactivado");
       }
      }
       //Funcionalidad de desarrollador para obtener ciertos datos de interés.
   }
  }

  if (j >= 100 && P_Log == 1) {  //Obtención de los valores obtenidos
    Serial.print("\n \n");
    Serial.print(i);
    Serial.print("\nLa temperatura es de: \n");
    Serial.print(temp1);
    Serial.print("\nLa humedad es de: \n");
    Serial.print(humi1);
    Serial.print("\nLa intensidad es de: \n");
    Serial.print(I);
    Serial.print("\n");
    Serial.print(SensorPro,3);
    j = 0;
  }
  i = i + 1;
  j = j + 1;
  delay(10);
}
```

Como se puede observar solo se lee del módulo bluetooth en caso de estar disponible, esto se comprueba con la siguiente instrucción:

```
if (BT1.available() > 0) {}
```

En este caso, si está disponible se borra la variable en la que se almacena el valor del comando recibido, se adquiere en su totalidad el comando y en primer caso comprobamos si el comando tiene un tamaño mayor o igual que tres caracteres, ya que como veremos en el protocolo es el tamaño mínimo de trama y en segundo lugar si el primer carácter del comando coincide con la variable Serial_Arduino, es una simple comprobación para evitar el recibir tramas para otros módulos Arduinos.

```
     index = 0;
    memset(incomingByte, 0, sizeof(incomingByte));

    while(BT1.available() > 0){
       char aChar = BT1.read();

       incomingByte[index] = aChar; // Se añade el carácter a la variable
       index++;                     // Sumamos uno al puntero
       incomingByte[index] = '\0';  // NULL terminación para la variable
    }

    if(sizeof(incomingByte)>=3 && incomingByte[0] == Serial_Arduino ){
```

Una vez se han pasado estas comprobaciones, se entra en un switch el cual a su vez contiene otros switch. Este primer switch mira el segundo carácter recibido y dependiendo en que caso se encuentre, se accede al segundo swicth en el caso de tratarse de sensores, el switch mira el tercer y último carácter y determina el número de sensor sobre el que se debe extraer la información y mandarla por el módulo bluetooth. En el caso de tratarse de actuadores, el switch mira el tercer y último carácter y determina si se ha de encender (1), apagar (0) o conmutar (C) ese actuador.

```
      switch (incomingByte[1]) {

        case 'A':          //Actuador Relé 1
          switch(incomingByte[2]){
            case '1':
              digitalWrite(relePin1, HIGH);
              BT1.println("OK");
              estado = 1;
              break;
            case '0':
              digitalWrite(relePin1, LOW);
              BT1.println("OK");
              estado = 0;
              break;
           case 'C':
            if(estado == 0){
              digitalWrite(relePin1, HIGH);
              estado = 1;
            }else{
              digitalWrite(relePin1, LOW);
              estado = 0;
            }
              BT1.println("OK");

              break;
          }
          break;

        case 'T':                   //Petición de sensor de Temperatura
          switch(incomingByte[2]){
            case '1':
              BT1.println(temp1);
              break;
            case '2': //Para futuros sensores
              BT1.println("N/D");
              break;
          }
          break;

        case 'H':                   //Petición de sensor de Humedad
           switch(incomingByte[2]){
            case '1':
              BT1.println(humi1);
              break;
            case '2': //Para futuros sensores
              BT1.println("N/D");
              break;
          }
          break;

        case 'W':                   //Petición de sensor de Intensidad
          switch(incomingByte[2]){
            case '1':
              BT1.println(SensorPro,3);
              break;
            case '2': //Para futuros sensores
              BT1.println("N/D");
              break;
          }
          break;
      }
```

Por ultimo y como funcionalidad para el debuggeado del programa de puede habilitar log, esta funcionalidad se activa mandando **LOGX**, el ultimo carácter indica activo o desactivo.

```
}else if(incomingByte[0] != Serial_Arduino){
       if(incomingByte[0]=='L' && incomingByte[1]=='O' && incomingByte[2]=='G'){
        if(incomingByte[3] == '1'){
```

```
          P_Log = 1;
          BT1.println("Log Activado");
        }else if(incomingByte[3] == '0'){
          P_Log = 0;
          BT1.println("Log Desactivado");
        }
      }
       //Funcionalidad de desarrollador para obtener ciertos datos de interés.
   }
```

5.3. Protocolo

Este sencillo protocolo define comandos de tres caracteres, en el que el primer carácter define el módulo Arduino al cual va dirigida la instrucción y los otros dos caracteres definen el sensor o actuador sobre el que recae la instrucción.

El planteamiento es el siguiente:

Si en el segundo carácter aparecen las letras de la (A) a la (G) indican 7 actuadores sobre los que se actuaran, el carácter que venga después será el que determine si encendido en caso de venir a "1", apagado si recibe un "0" o en el caso de recibir una "C" conmutara el estado de ese actuador.

> **AA1** enciende el actuador 1 del módulo A que se haya declarado en el código.
>
> **AA0** apaga el actuador 1 del módulo A que se haya declarado en el código.
>
> **AAC** conmuta el actuador 1 del módulo A que se haya declarado en el código.

Si en el segundo carácter aparece una "T", indica que se requiere el dato de temperatura, y el carácter que venga después determina el número de sonda de temperatura que dará ese valor.

> **AT1** pide el dato de la sonda de temperatura 1 del módulo A que se haya declarado en el código.

Si en el segundo carácter aparece una "H", indica que se requiere el dato de humedad y el carácter que venga después determina el número de sonda de humedad que dará ese valor.

> **AH1** pide dato sonda de humedad 1 del módulo A que se haya declarado en el código.

Si en el segundo carácter aparece una "W", indica que se requiere el dato de intensidad, y el número que le sigue determina el número de sonda de intensidad que dará ese valor.

> **AW1** pide dato sonda de intensidad 1 del módulo A que se haya declarado en el código.

Para comprobar el correcto funcionamiento del módulo se define la trama "LOGX", la cual activa o desactiva el log que nos permitirá comprobar la correcta adquisición de los datos por parte del Arduino.

5.4. Configuración

En este apartado se va a tratar todo lo relacionado con la parte de la Raspberry Pi, describiéndose toda la configuración inicial que se ha usado para permitir al dispositivo ser el cerebro de este proyecto. Tanto la configuración que permita obtener todas las herramientas adecuadas para convertir al dispositivo en un servidor capaz de alojar las páginas web y almacenar datos en la base de datos, la cual se procederá a describir en el siguiente apartado.

Con lo que respecta a la configuración de la Raspberry Pi, primero se debe puntuar que se ha usado como sistema operativo Raspbian, sobre el que se ha instalado un servidor web gracias a Apache, MySQL, PHP y Python. Estas herramientas van a permitir la gestión el sistema, gracias a Python se utiliza una librería para la gestión de bases de datos y otra para la comunicación mediante un socket con dispositivos bluetooth, en este caso mediante el protocolo rfcomm, el cual se ha comentado en el apartado Bluetooth, permitiendo la conexión directa con los arduinos y de esta forma pudiendo mandar comandos para interactuar con él.

Primero, se van a comentar las primeras configuraciones a realizar en la Raspberry Pi para poder trabajar con ella de una forma cómoda. Antes que nada, debemos tener una microSD con la imagen del sistema operativo Raspbian para poder funcionar con el dispositivo. Esto se puede hacer simplemente, bajando la imagen de internet y montando la tarjeta con la imagen del sistema operativo o por el contrario nos podemos ayudar de NOOBS que permite instalar más de un sistema operativo en la tarjeta y conectando a internet el dispositivo, se descarga el sistema operativo elegido y lo instala.

Una vez hecho esto, se deberá configurar tanto el idioma, como el tipo de teclado a utilizar, una vez se inicie el sistema operativo por primera vez nos aparecerá una pantalla como la que se muestra en la Fig. 25: Ventana inicial de Raspbian. En ella se permite el cambio de la configuración del teclado en el cuarto ítem, el cual se pedirá el tipo de teclado, el idioma a usar con el teclado y una serie de opciones como la ubicación de la tecla “Alt”, para más información sobre esta configuración y las demás opciones de la Fig. 25 se pueden consultar en la (manuti, 2013).

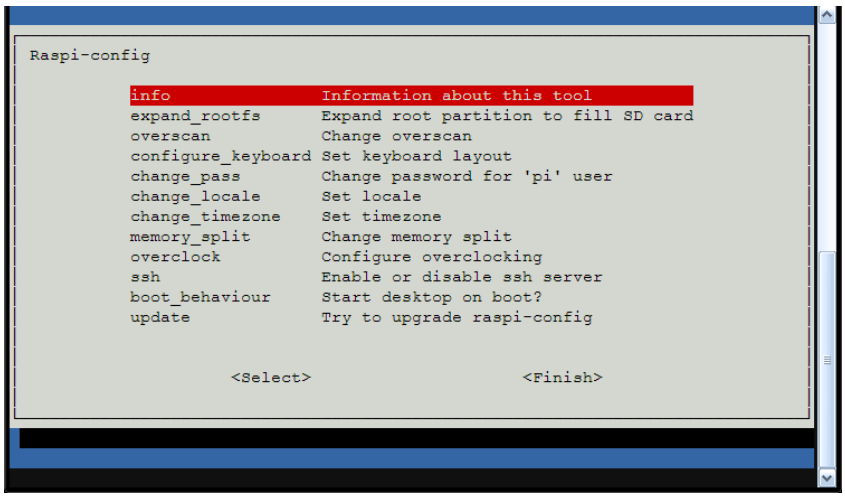


Fig. 25: Ventana inicial de Raspbian

Una vez se ha realizado esta configuración y se comprueba el correcto uso del ratón y el teclado, se puede optar por continuar trabajando de forma directa gracias al ratón y el teclado por lo que se debe tener acceso al dispositivo y además disponer de un monitor o televisor donde poder trabajar o por el contrario y la opción que se tomó para este proyecto es trabajar a través del protocolo SSH.

De esta forma podemos acceder a la consola, la principal herramienta desde la que se puede trabajar con este dispositivo desde cualquier otro dispositivo. De esta forma permite dejar al dispositivo, la Raspberry Pi, situado en un emplazamiento fijo y solo necesitando una fuente de alimentación y el acceso a internet bien sea por medio cableado o inalámbrico. Además, este modo de uso permite trabajar con facilidad desde un dispositivo más conocido y permitiendo explorar las funcionalidades del dispositivo con la ayuda de internet.

Una vez se puede acceder al dispositivo a través, en este caso del programa Putty, el cual se ha elegido por su sencillez y su liviano peso. Como se puede observar en la Fig. 26, pese al principio aparentar cierta complejidad, para el uso más sencillo solo se necesita saber la ip que tiene la raspberry, la cual se puede obtener con el comando "ifconfig" en la propia terminal del dispositivo o pudiéndola obtener del propio router al que se conecta. Una vez se conoce esa ip, se introduce en el primer recuadro rojo remarcado en la Fig. 26, con los dos recuadros siguientes se puede guardar esta sesión para facilitar los futuros accesos y solo queda de pulsar sobre "*open"* para que se abra la consola, solicite usuario y contraseña y podemos interactuar con nuestro dispositivo.

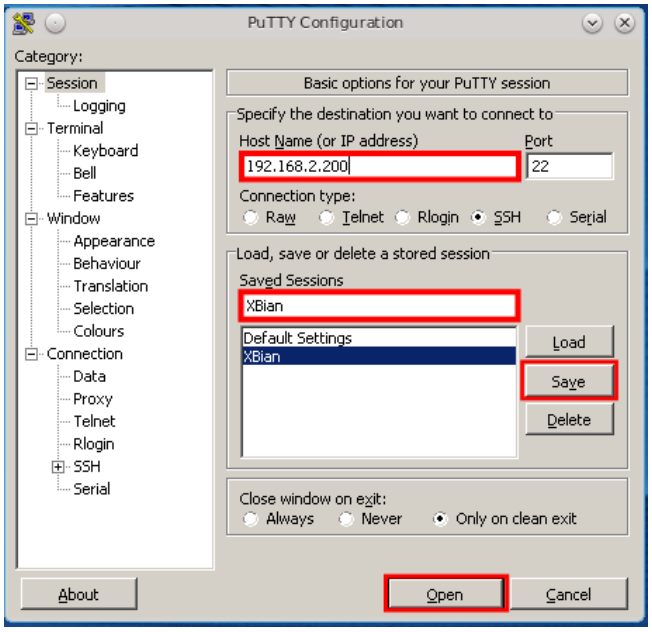


Fig. 26: Apariencia programa Putty

Con el acceso remoto a través de este programa ya es posible ejecutar diversos comandos sobre el dispositivo. Los más elementales y para actualizar el dispositivo, sus fuentes y sus librerías son *sudo apt-get update* y *sudo apt-get upgrade*. Después de ejecutar estos comandos y dejar el debido tiempo para que finalicen y de este modo poder escribir sobre la consola, se podrá empezar con la instalación de las diversas herramientas.

Lo primero se creará y se concederán permisos al grupo que usa apache por defecto.

```
sudo addgroup www-data

sudo usermod -a -G www-data www-data
```

Después se instalará el servidor HTTP Apache, mediante el siguiente comando:

```
sudo apt-get install apache2
```

Reiniciamos Apache, una vez se haya concluido la instalación anterior con el siguiente comando:

```
sudo /etc/init.d/apache2 restart
```

Se podrá comprobar el correcto funcionamiento accediendo a localhost o 127.0.0.1 desde un navegador en la Raspberry o desde la IP local que ésta tenga desde algún otro dispositivo en la red.

Para saber cuál es la IP local que se le asignó, desde una terminal podemos usar:

```
ifconfig
```

Por regla general las IP's locales tienen un número similar a 192.168.0.xx o 192.168.1.xx, así que buscamos la línea que contenga la ip del dispositivo, el cual nos servirá de acceso y el cual podremos reenviar en el router para poder acceder por fuera de la red local.

Accediendo a la IP desde el navegador se puede ver la página por defecto de apache, la cual tendrá una apariencia como se muestra en la Fig. 27: Pagina inicial de Apache.

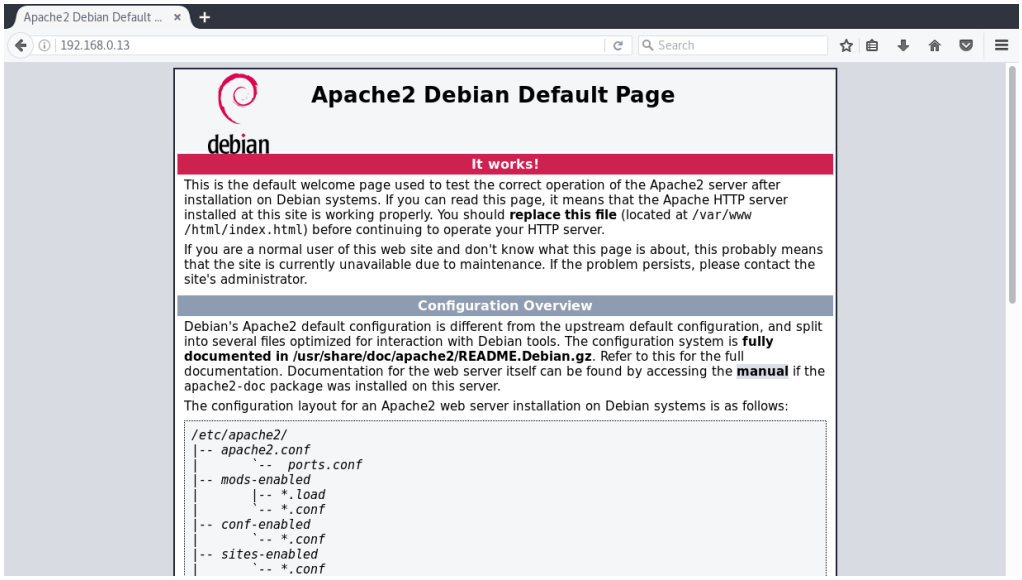


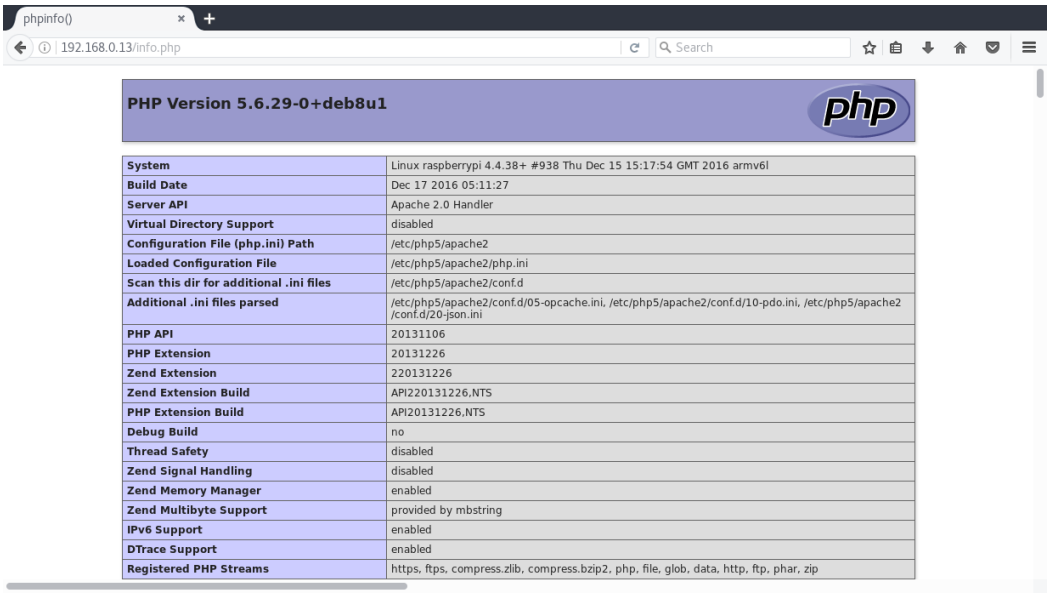
Fig. 27: Pagina inicial de Apache

Ahora se va a proceder a instalar PHP y su respectivo módulo de Apache, lo cual es el complemento perfecto para el desarrollo de webs dinámicas, se instala con la siguiente instrucción:

```
sudo apt-get install php5 libapache2-mod-php5
```

Se puede comprobar el correcto funcionamiento del siguiente modo, se crea un archivo con el nombre `info.php` con el contenido `<?php phpinfo(); ?>` en el directorio `/var/www/html`. Esta es una función de PHP que mostrará una página de información con respecto a la instalación local.

Una vez hecho esto, se puede acceder `localhost/info.php` desde el navegador y, si se puede ver la página de información, la cual se muestra en la Fig. 28: Información sobre PHP , solo quedaría instalar MySQL.



System	Linux raspberrypi 4.4.38+ #938 Thu Dec 15 15:17:54 GMT 2016 armv6l
Build Date	Dec 17 2016 05:11:27
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/apache2/conf.d
Additional .ini files parsed	/etc/php5/apache2/conf.d/05-opcache.ini, /etc/php5/apache2/conf.d/10-pdo.ini, /etc/php5/apache2/conf.d/20-json.ini
PHP API	20131106
PHP Extension	20131226
Zend Extension	220131226
Zend Extension Build	API220131226,NTS
PHP Extension Build	API20131226,NTS
Debug Build	no
Thread Safety	disabled
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	enabled
Registered PHP Streams	https, ftps, compress.zlib, compress.bzip2, php, file, glob, data, http, ftp, phar, zip

Fig. 28: Información sobre PHP

Para finalizar con las herramientas necesarias para el desarrollo web, se procederá a instalar MySQL con su módulo de PHP. Durante el progreso de la instalación, se solicitará que se cree una contraseña para el usuario "***root"*** de MySQL. Este usuario no es el mismo que el usuario root de Linux, pero cumple una función de administrador dentro de la instalación de MySQL. Es recomendable usar una contraseña segura, ya que este usuario tendrá los permisos administrativos sobre todas las bases de datos y los usuarios.

```
sudo apt-get install mysql-server php5-mysql
```

Una vez que se ha llevado a cabo la instalación de forma correcta, se ejecutara la siguiente instrucción:

```
mysql_sercure_instalation
```

Este es un script que permite configurar cuestiones respecto a la seguridad de MySQL, ya que, por defecto, muchas opciones están activadas y cuya finalidad es hacer pruebas,

pero que no son útiles en el normal funcionamiento del servidor y pueden llegar a comprometer la integridad de la base de datos:

1. En primer lugar, se pide que introduzcamos la contraseña del usuario "***root"*** (de MySQL), y luego se solicita una serie de acciones:

2. Si se desea cambiar la contraseña o si fue creada ninguna durante la instalación, respondiendo que sí ésta se puede cambiar o crear.

3. Por defecto, se crea un usuario anónimo que permite acceder a la base de datos sin la creación de otro usuario. Respondiendo que **sí**, este usuario será borrado.

4. El usuario "***root***" solo se debería poder acceder desde la IP local donde está instalada la base de datos. Respondiendo que **sí**, se desactivara el acceso remoto.

5. Se crea una base de datos *test* que cualquiera puede acceder. Respondiendo que **sí**, para borrar esta base de datos.

6. Finalmente, se pide recargar la tabla de privilegios para que los cambios tengan efecto de inmediato. Respondiendo que **sí**, se recarga y finaliza el script.

Ahora se puede reiniciar el servicio de Apache con la siguiente instrucción y todo debería estar funcionado correctamente:

```
sudo /etc/init.d/apache2 restart
```

Como un ayuda adicional, se puede instalar phpMyAdmin para poder manipular las bases de datos de MySQL con una interfaz web, ya que por el momento solo se acceder a éstas a través de la línea de comandos o utilizando PHP. Para instalar este paquete, solo se debe escribir el siguiente comando:

```
sudo apt-get install phpmyadmin
```

Durante la instalación, se puede elegir usar Apache como servidor web, seleccionando la opción con la barra espaciadora y luego aceptar. Una vez hecho esto, se solicita si se quiere configurar la base de datos con dbconfig-common, se responde que si y se solicita la contraseña del usuario "***root"*** de MySQL. Ahora se pide crear una contraseña para que phpMyAdmin se registre en MySQL.

Para comprar que la instalación ha sido satisfactoria, se debe acceder a localhost/phpymadmin desde el navegador para poder registrarse, con cualquier usuario de MySQL, tal y como se muestra en la Fig. 29: Página inicio de phpMyAdmin

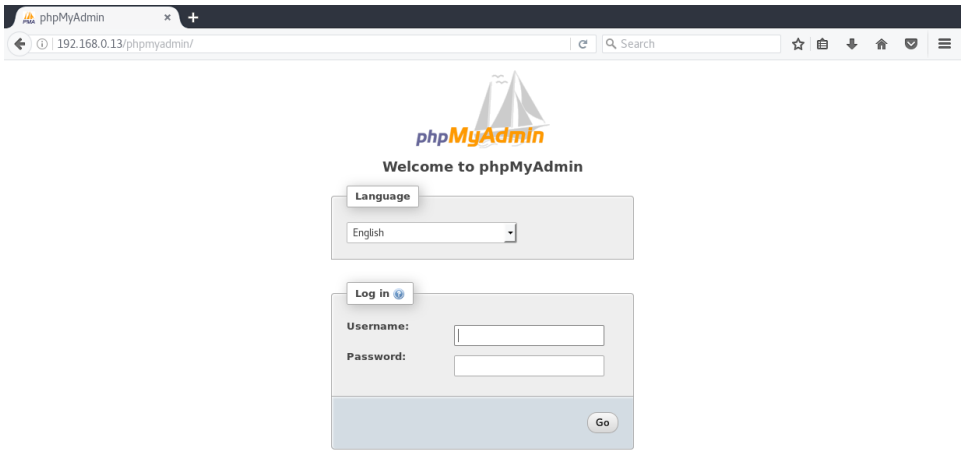


Fig. 29: Página inicio de phpMyAdmin

Por último, es necesario instalar la librería que permite a Python interactuar con MySQL y la librería que permita crear el socket por rfcomm para poder comunicarse con Arduino.

Para comunicarse con MySQL se usara la librería MySQLdb, la cual se instala con la siguiente instrucción:

```
sudo apt-get install python-mysqldb
```

Como se explica en el apartado de Scripts usados, esta librería hace uso de una serie de instrucciones que permite conectarse a la base de datos y realizar la lectura y escritura de datos, a continuación se muestra un resumen con las instrucciones más comunes.

```
#!/usr/bin/python
import MySQLdb

db = MySQLdb.connect(host="localhost",              # Host
                     user="user",                   # Usuario
                     passwd="1234",    # Contraseña
                     db="test")                     # Nombre de la base de datos

# Se debe crear un objeto Cursor, el cual permita ejecutar todas las instrucciones que se necesitan

cur = db.cursor()

# Ejemplo de SELECT

cur.execute("SELECT * FROM TABLA")

# Imprimir la primera columna de todos los registros

for row in cur.fetchall():
   print row[0]
```

La mayoría de los adaptadores bluetooth se basan en USB y pueden ser configurados con las utilidades HCI.

Para instalar los paquetes de software necesarios en Raspbian, se ejecutan los comandos que se muestran a continuación:

```
sudo apt-get install bluetooth bluez bluez-tools rfkill rfcomm
```

A continuación, se inicia el servicio bluetooth, con la siguiente instrucción:

```
sudo service bluetooth start
```

Para la exploración de dispositivos bluetooth, primero se debe asegurar que el dispositivo bluetooth esté encendido y desbloqueado, esto se puede comprobar con el comando rfkill.

```
sudo rfkill list
```

Si el dispositivo bluetooth está bloqueado (bloqueo blando o duro), se desbloquea con el siguiente comando:

```
sudo rfkill unblock bluetooth
```

Se activa el dispositivo bluetooth con el comando hciconfig y se comienza con el escaneo, una vez se ha podido comprobar que el dispositivo esta encendido y es visible.

```
sudo hciconfig hci0 up

hcitool scan
```

Se creara una lista con los dispositivos que estén visibles, como el siguiente ejemplo:

```
83:23:26:15:54:46 Mi-Nokia
```

Para descubrir los servicios y ya que se dispone de la dirección MAC del dispositivo bluetooth de destino, se utiliza el comando sdptool para saber qué servicios (como están disponibles en el dispositivo.

```
sdptool browse 83:23:26:15:54:46
```

Se puede filtrar el resultado con el comando grep, de la salida del anterior comando.

```
sdptool browse 83:23:26:15:54:46 | grep 'Service Name:'
```

También puede utilizar la herramienta interactiva bluetoothctl para este propósito.

```
$ bluetoothctl [bluetooth]# info 83:23:26:15:54:46
Device 83:23:26:15:54:46
Name:&amp;nbsp;My-Nokia
Alias: Nokia
```

```
Class: 0x580204
Icon: phone
Paired: yes
Trusted: yes
Blocked: no
Connected: no
LegacyPairing: no
UUID: OBEX Object Push (00001105-0000-1000-8000-00805f9b34fb)
UUID: Audio Source (0000110a-0000-1000-8000-00805f9b34fb)
UUID: A/V Remote Control Target (0000110c-0000-1000-8000-00805f9b34fb)
UUID: Handsfree Audio Gateway (0000111f-0000-1000-8000-00805f9b34fb)
```

Para comprobar la conexión con el dispositivo, se puede realizar un ping al dispositivo bluetooth. Si el dispositivo de destino esta activo, se puede hacer ping con el comando l2ping, requiere privilegio de root.

```
sudo l2ping 83:23:26:15:54:46 Ejemplo
```

Una vez se ha comprobado la conexión, se puede realizar el emparejamiento del dispositivo, conectando al dispositivo bluetooth con rfcomm. Este comando requiere privilegio de root.

```
sudo rfcomm connect
```

Un ejemplo de conexión del dispositivo hci0 a un dispositivo cliente en el canal 1:

```
sudo rfcomm connect hci0 83:23:26:15:54:46 1
```

El dispositivo solicitara que se acepte esta solicitud de conexión. Ahora el dispositivo cliente bluetooth estará disponible como /dev/rfcomm0, este proceso de emparejamiento se deberá realizar por cada uno de los Arduinos que se deban conectar al proyecto.

5.5. Base de Datos

En el segundo apartado se describirá la estructura de la base de datos, describiendo las distintas tablas que se usan para gestionar las configuraciones y almacenar los datos que se obtienen. Esta configuración se verá reflejada en la base de datos, la cual deberá ser actualizada con cualquier cambio que se produzca en los diferentes módulos. De forma manual mediante el uso de inserts y updates sobre la base de datos o mediante una página web de configuración para permitir el dar de alta o de baja sensores y módulos de forma más sencilla. Esta última opción requiere de un cierto conocimiento por parte del cliente por lo que se podrá activar o desactivar en función del perfil del cliente.

Para la definición de la base de datos, primero se describirán sus tablas, haciendo hincapié en el uso que se le da a cada campo de la tabla.

La primera tabla, *Modulos_Arduinos* es un historial de los módulos dados de alta en el sistema tanto si están activos en el momento, con su campo *Fecha_Baja* a valor null o por el contario si este campo tiene una fecha, el dispositivo se habrá dado de baja en la fecha que indique el campo. Todos los módulos deberán tener una fecha en la que fueron dados de alta, la cual se registra en el campo *Fecha_Alta* además del usuario que lo registro, el nombre aparecerá en el campo *Usuario*. Los campos *Modulo* y *Direccion*, tendrán la letra que tiene asignada ese modulo en el caso del primer campo y la dirección MAC del dispositivo bluetooth en el caso del segundo campo, tal y como se describe en la Tabla 5: Descripción de Modulos_Arduinos. Esto permite guardar la relación entre la letra asignada a cada dispositivo con su dirección física.

Tabla 5: Descripción de Modulos_Arduinos

Campo	Tipo
Modulo	varchar(1)
Direccion	varchar(20)
Fecha_Alta	datetime
Fecha_Baja	datetime
Usuario	varchar(20)

La segunda tabla, *Sensores_Activos* comparte la filosofía de la primera tabla. En este caso esta tabla guarda la relación de todos los sensores con el módulo al que pertenecen. El primer campo se almacena el *Modulo*, llave primaria que relaciona la tabla 1 con la tabla 2. El segundo campo *TipoSensor*, recoge la letra que se define en el Protocolo, definiendo el tipo de sensor o en el caso de los actuadores el número de actuador. El tercer campo *NSensor*, recoge el número que se define en el Protocolo, definiendo el número de sensor, en el caso de los actuadores el valor de este campo no tiene transcendencia. Por último, los campos *Fecha_Alta* y *Fecha_Baja* recogen el estado de los sensores y actuadores si se encuentra dados de alta en el sistema o no, y en este caso cuando fueron dados de baja, tal y como se observa en la Tabla 6: Descripción de Sensores_Activos.

Tabla 6: Descripción de Sensores_Activos

Campo	Tipo
Modulo	varchar(1)
TipoSensor	varchar(1)
NSensor	varchar(1)
Fecha_Alta	datetime
Fecha_Baja	datetime

La segunda tabla *SensoresHistorico*, almacena todas las acciones que se han producido en el proyecto es la tabla de mayor importancia. Se registra el módulo con el campo 1, el tipo de sensor o el número de actuador con el campo 2, el número de sensor con el campo 3, acompañado de un valor en el caso de los sensores con el campo 4 y la fecha en que se produjo la acción con el campo 5, tal y como se describe en la Tabla 7: Descripción de SensoresHistorico. Dependiendo del periodo de ejecución del script que se encargara de adquirir los datos, el volumen de datos que albergara esta tabla puede variar. En el caso de ser un periodo muy bajo, la tabla recibirá muchos datos y con el paso del tiempo el manejo de tablas grandes puede ralentizar la extracción de datos. Es por ese motivo que una línea de mejora deberá ir por ese camino, la cual se explica en el apartado 7.2. De momento la extracción de datos se resuelve con una sencilla *select*, tipo de instrucción que permite la extracción de datos de tablas en MySQL, la cual deberá ser modificada en los parámetros del *where* para extraer los datos que se precisen y entre las fechas que se solicite o bien los últimos n resultados.

Tabla 7: Descripción de SensoresHistorico

Campo	Tipo
Modulo	varchar(1)
TipoSensor	varchar(1)
NSensor	varchar(1)
Valor	float(6,3)
Fecha_Alta	datetime

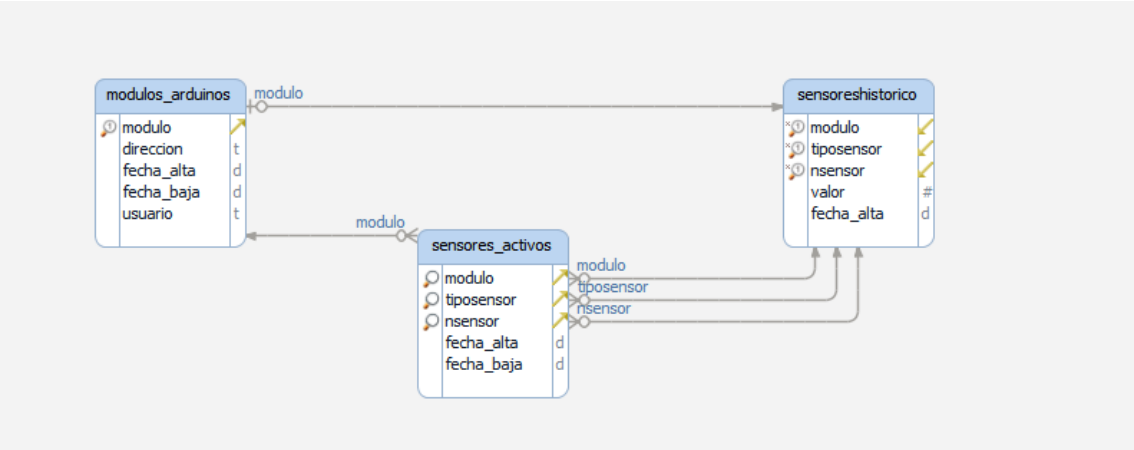


Fig. 30: Esquema de la base de datos

Para facilitar el uso de la herramienta por parte del usuario se crea una cuarta tabla la cual relación los campos modulo, tipo de sensor y numero de sensor en el caso de

tratarse de un actuador y un alias, el cual permite al usuario saber de que elemento se trata por un nombre conocido y evitar el tener que recordar a que sensores pertenece cada conjunto de caracteres. Esta tabla denominada *NombresElementos*, al igual que las dos primeras tablas contiene los campos *Fecha_Alta* y *Fecha_Baja*, llevando así un historial de los nombres que se han ido implementando y un campo Usuario para identificar quien realizo estos cambios. Por último, los campos que guardarían la relación serian: el *Modulo*, el *TipoSensor*, el *NSensor* y el *Alias*, este último con un límite de 20 caracteres, tal y como se indica en la Tabla 8: Relación de nombres y elementos.

Tabla 8: Relación de nombres y elementos

Campo	Tipo
Modulo	varchar(1)
TipoSensor	varchar(1)
NSensor	varchar(1)
Alias	varchar(20)
Fecha_Alta	datetime
Fecha_Baja	datetime
Usuario	varchar(20)

5.6. Scripts usados

En el tercer apartado se mostrarán y explicaran los dos scripts que se usan para llevar a cabo tanto la ejecución de ordenes sencillas sobre los actuadores y sensores como el script que se ejecutara de forma periódica para la recolección de datos, los cuales se obtendrán de los sensores que se tengan configurados en los distintitos módulos Arduino que se hayan instalado.

Primero, se explicará el script comando.py, ya que al realizar una sola tarea es mas sencillo de explicar.

```
#!/usr/bin/env python
import bluetooth        #Libreria Bluetooth
import sys                    #Libreria Inputs
import MySQLdb                #Libreria SQL

DB_HOST = 'localhost'   #Host de BBDD
DB_USER = 'root'        #Usuario de BBDD
DB_PASS = 'XXXX'        #Contraseña de BBDD
DB_NAME = 'ProyectoDomotico'
#Nombre de BBDD

bd_addr = sys.argv[2]   #Direccion Bluetooth
port = 1                      #Puerto Bluetooth

sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
#Creación del socket
sock.connect((bd_addr, port))
#Conexión al socket
```

```
rec = ""          #Variable temporal
recP = ""         #Variable resultado final
response = None #Inicializar variable While

comando = sys.argv[1]
#Recepción del comando a enviar

sock.send(comando)      #Envió de comando
try:                    #Recepción de Datos
    while(response is None):
      (rec, adress) = sock.recvfrom(1024)
      recP += rec
      if(recP.find("\n") != -1):    #Resultado completo
            response=recP[0:len(recP)-1] #Deshecha \n
            sock.close()                 #Cierra el socket
except KeyboardInterrupt:           #Interrupción
      sock.close()

if(response[0:2]=='OK'):            #En el caso de Actuador
      valor=1                             #Se sustituye por 1
else:
      valor=response                      #En el caso de Sensor

datos = [DB_HOST, DB_USER, DB_PASS, DB_NAME]
#Configuracion para acceder a la base de datos

query = 'INSERT INTO SensoresHistorico (Modulo, TipoSensor, NSensor,
Valor, Fecha_Alta) VALUES (\''+comando[0]+"\' ,\' "+comando[1]+"\' ,
"+comando[2]+" , " +str(valor)+" ,now() " +")"
#Instruccion INSERT para ingresar los datos en la BBDD

conn = MySQLdb.connect(*datos) # Conectar a la base de datos
cursor = conn.cursor()         # Crear un cursor
cursor.execute(query)          # Ejecutar una consulta
conn.commit()              # Hacer efectiva la escritura de datos
```

Como se puede observar las tres primeras líneas son para importar las librerías que se van a necesitar, y ya se han explicado en apartados anteriores. A continuación, las 4 líneas siguientes, aparecen los datos necesarios para poder acceder a la base datos.

```
bd_addr = sys.argv[2]   #Direccion Bluetooth
port = 1                      #Puerto Bluetooth
```

Estas dos líneas son para la conexión Bluetooth, el primero de ellos se recibe como argumento al llamar al script y el segundo es el puerto para el acceso al módulo Bluetooth.

```
sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )
#Creación del socket
sock.connect((bd_addr, port))
#Conexión al socket
```

A continuación, se crea el socket y se realiza la conexión. Las tres líneas posteriores son variables que se van a usar para la recepción de la información.

```
comando = sys.argv[1]
#Recepción del comando a enviar
```

Esta variable alberga el parámetro con el parámetro que se pretende enviar.

```
sock.send(comando)      #Envió de comando
try:                    #Recepción de Datos
    while(response is None):
      (rec, adress) = sock.recvfrom(1024)
      recP += rec
      if(recP.find("\n") != -1):    #Resultado completo
            response=recP[0:len(recP)-1] #Deshecha \n
            sock.close()                  #Cierra el socket
except KeyboardInterrupt:           #Interrupción
      sock.close()

if(response[0:2]=='OK'):            #En el caso de Actuador
      valor=1                             #Se sustituye por 1
else:
      valor=response                      #En el caso de Sensor
```

Esta parte del script, manda el comando y espera la contestación por parte del Arduino. Una vez se recibe la información, detectando el final de la información con un salto de línea (“\n”), se procesa y se prepara para ser registrada en la base de datos.

```
datos = [DB_HOST, DB_USER, DB_PASS, DB_NAME]
#Configuracion para acceder a la base de datos

query = 'INSERT INTO SensoresHistorico (Modulo, TipoSensor, NSensor,
Valor, Fecha_Alta) VALUES (\''+comando[0]+"\' ,\' "+comando[1]+"\' ,
"+comando[2]+" , " +str(valor)+" ,now() " +")"
#Instruccion INSERT para ingresar los datos en la BBDD

conn = MySQLdb.connect(*datos) # Conectar a la base de datos
cursor = conn.cursor()         # Crear un cursor
cursor.execute(query)          # Ejecutar una consulta
conn.commit()               # Hacer efectiva la escritura de datos
```

Por último, se preparan los datos de acceso a la base de datos, se monta la sentencia para insertar los datos recopilados, se crea la conexión y se ejecuta la instrucción.

El siguiente script es algo más complejo de entender, pero se puede comprender extrapolando del anterior script. Ya que tienen ciertas partes comunes, como lo son las variables iniciales, pero en este caso para obtener las direcciones de los distintitos módulos bluetooth y los comandos a realizar compuestos por Modulo,TipoSensor, NSensor, se realiza una petición a la base de datos para obtener la lista de módulos y sensores disponibles en el momento de la ejecución del script, lo cual se comprueba con el valor de la fecha baja, el cual debe estar a valor null y además con la clausura *TipoSensor not in ("A","B","C","D","E","F",G)*, se evita extraer los actuadores.

Una vez se obtiene esa *select*, de la base de datos. Se crea un ciclo *For* para recorrer los distintos módulos y comandos, en ese momento por cada sensor se lanza una petición al módulo bluetooth, se recopila y se trata el valor. Por último, se inserta esa información en la base de datos. Todo este proceso se conecta y desconecta el socket para dejar libre el canal de información el mayor tiempo posible, evitando así posibles interferencias.

```
#!/usr/bin/env python
import bluetooth    #Libreria Bluetooth
import sys              #Libreria Inputs
```

```
import MySQLdb              #Libreria SQL

DB_HOST = 'localhost'       #Host de BBDD
DB_USER = 'root'            #Usuario de BBDD
DB_PASS = 'G8934m03'        #Contraseña de BBDD
DB_NAME = 'ProyectoDomotico'
#Nombre de BBDD


datos = [DB_HOST, DB_USER, DB_PASS, DB_NAME]

query  = 'select sa.Modulo,sa.TipoSensor,sa.NSensor,ma.Direccion '
query += 'from  Modulos_Arduinos ma '
query += ' inner join Sensores_Activos sa on sa.Modulo = ma.Modulo '
query += 'where ma.Fecha_Baja is null and sa.Fecha_Baja is null'
query += 'where sa.TipoSensor not in ("A","B","C","D","E","F",G)'

# Select a ejecutar

conn = MySQLdb.connect(*datos) # Conectar a la base de datos
cursor = conn.cursor()         # Crear un cursor
cursor.execute(query)          # Ejecutar una consulta
sock=bluetooth.BluetoothSocket( bluetooth.RFCOMM )

for (Modulo,TipoSensor, NSensor, Direccion) in cursor:

       bd_addr = Direccion
       port = 1
       sock.connect((bd_addr, port))

       rec = ""
       recP = ""
       response = None
       comando = Modulo+TipoSensor+NSensor

       sock.send(comando)

       try:
              while(response is None):
                     (rec, adress) = sock.recvfrom(1024)
              recP += rec
              if(recP.find("\n") != -1):
                     response=recP[0:len(recP)-1]
                     sock.close()
       except KeyboardInterrupt:
              sock.close()

       if(len(response) <= 6):
              if(response[0:2]=='OK'):
                     valor=1
              else:
                     valor=response
       else:
              valor = -1

       query2  = 'INSERT INTO SensoresHistorico '
       query2 += '(Modulo, TipoSensor, NSensor, Valor, Fecha_Alta)'
       query2 += ' VALUES (\''+comando[0]+"\' ,\'"+comando[1]
       query2 += "\' ,\'"+comando[2]+"\' ," +str(valor)+" ,now())"

       conn = MySQLdb.connect(*datos)
       cursor = conn.cursor()
       cursor.execute(query2)
       conn.commit()

cursor.close()
```

Este script se puede ejecutar de forma periódica gracias a CRONTAB.

El comando crontab se puede utilizar en sistemas UNIX para programar la ejecución de otros comandos, con otras palabras, para la automatización de tareas. Se pueden ver los crontabs programados y también editarlos.

Para ver los programados, se utiliza este comando:

```
sudo crontab -l
```

Para editar:

```
sudo crontab -e
```

Las tareas cron siguen una determinada sintaxis. Tienen 5 asteriscos seguidos del comando a ejecutar.

```
* * * * * /bin/ejecutar/script.sh
```

Los 5 asteriscos, de izquierda a derecha, representan:

Minutos: de 0 a 59.
Horas: de 0 a 23.
Día del mes: de 1 a 31.
Mes: de 1 a 12.
Día de la semana: de 0 a 6, siendo 0 el domingo.

Si se deja un asterisco en los cinco valores, quiere decir "cada" minuto, hora, día de mes, mes o día de la semana.
Por ejemplo:

```
* * * * * /bin/ejecutar/script.sh
```

Se ejecutaría el script.sh cada minuto durante todo el tiempo.

En el caso de este proyecto, al querer que se produzca cada X minuto se puede crear la siguiente tarea:

```
*/5 * * * * command
```

De esta forma, la Raspberry Pi ya ejecutaría consultas de forma periódica y las registrara en la base de datos. Por este motivo el sistema ya se comparta de forma autónoma, a la espera de recibir comandos por parte de la web y realizando la tarea de recolectar datos.

5.7. Interfaz Web

Por último, para esta sección Implementación:, se va a describir la parte más visual del proyecto que es la página web que permite controlar todo nuestro proyecto domótico además de poder visualizar las diferentes informaciones registradas en la base de datos.

En primer lugar, se mostrará una ventana “Controles de Casa”, tal y como se muestra en la Fig. 31: Pagina web de controles, en la cual se puede apreciar como los diferentes módulos activos que estén registrados en la base de datos irán apareciendo por orden de modulo. En cada uno de estos apartados se podrá observar los diferentes sensores y actuadores que tengan los distintos módulos. Como se puede apreciar en la Fig. 31, el módulo A consta de un actuador “A1” el cual podremos encender, apagar o conmutar. También consta de un sensor de humedad, del cual podremos observar su grafica pulsando sobre el respectivo botón. Lo mismo ocurrirá para los casos de humedad y potencia. Estas graficas mostraran las últimas horas pudiéndose configurar este periodo para poder ver distintos datos.

Además, para poder mostrar la verdadera funcionalidad de este proyecto se mostrarán diversas graficas conjuntas en las que se podrán mostrar toda la información de un módulo, tanto sus actuadores como sus sensores, pudiendo de este modo observar la información más rápido y de una mejor forma. Ya que con las distintas graficas se puede observar, por ejemplo, como al encender un actuador, si se dispone de un sensor de potencia que detecte esa línea, se verá incrementado el consumo en esa línea. O el caso contrario si teniendo los actuadores apagados y se detecta un consumo considerable en esa línea, es que la línea está teniendo un consumo residuo y podríamos detectar una derivación o algún fallo.

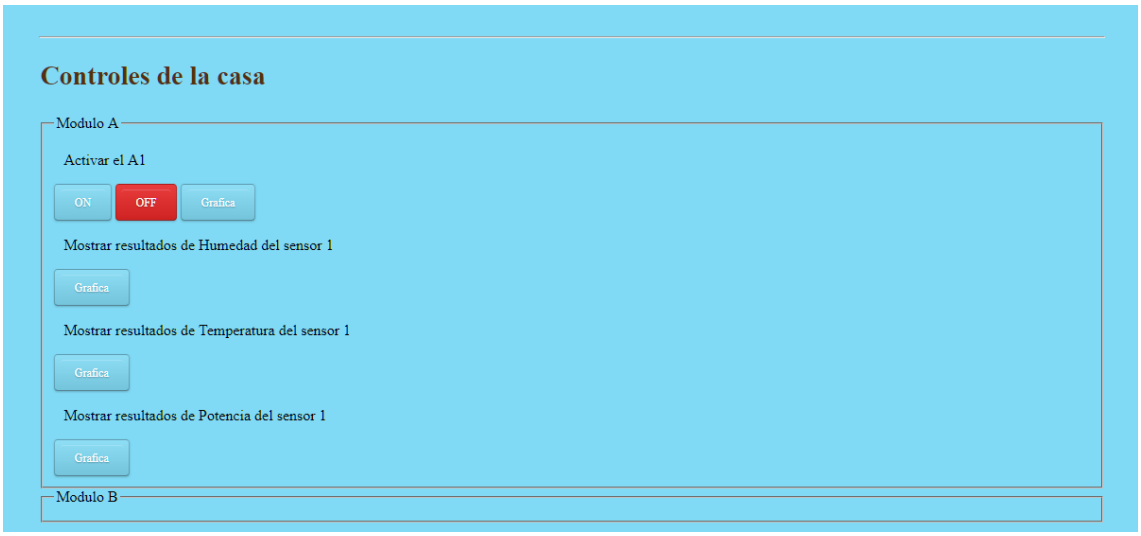


Fig. 31: Pagina web de controles

Algunas graficas que se podrían mostrar en la web para mostrar resúmenes diarios:

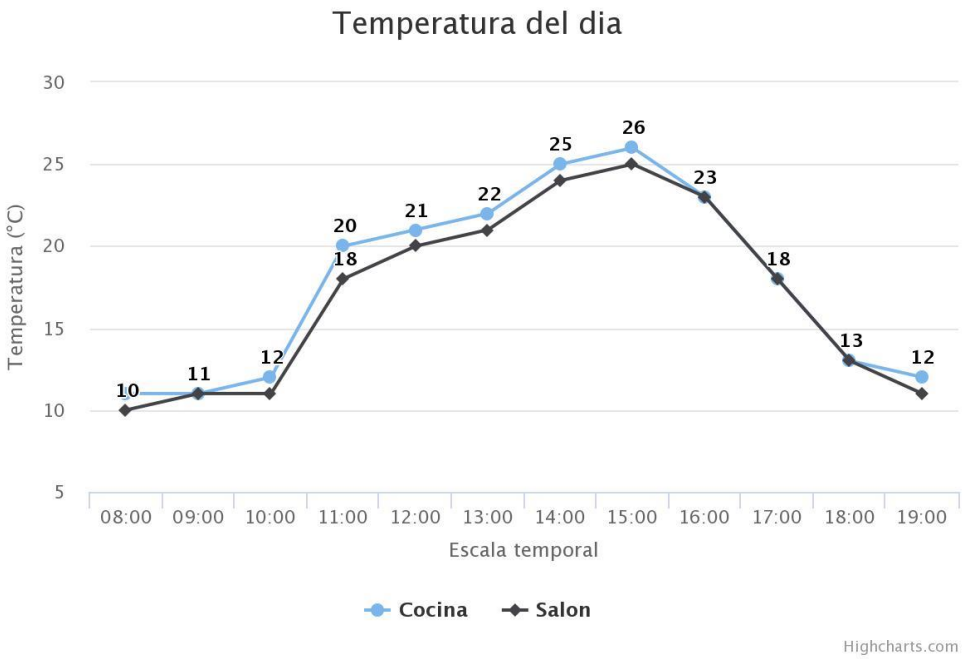


Fig. 32: Grafica temperatura diaria

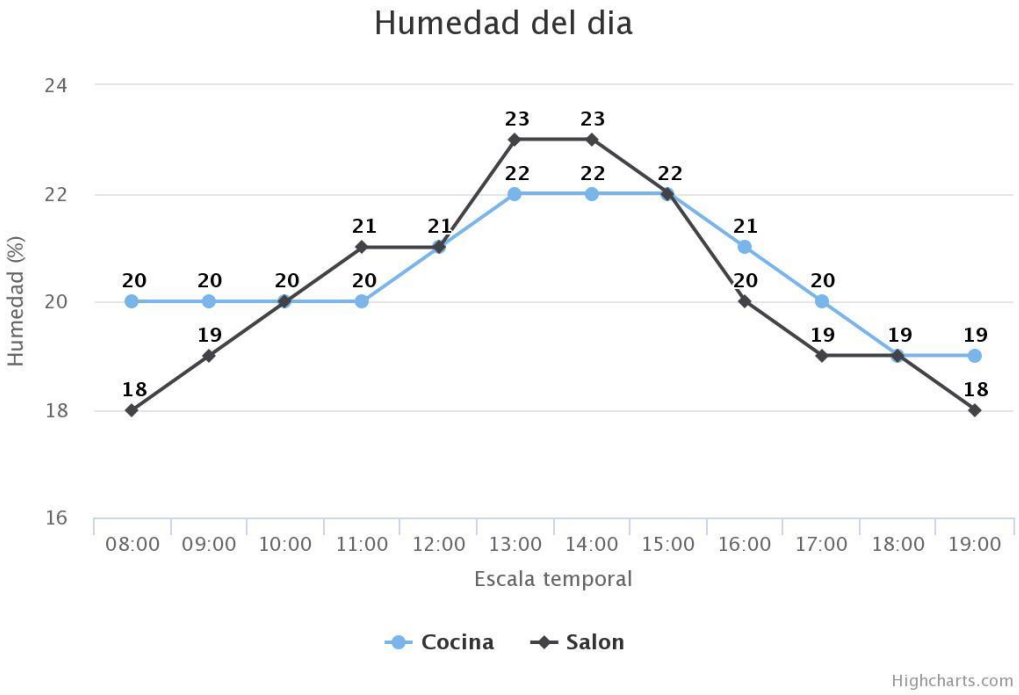


Fig. 33: Grafica humedad del dia

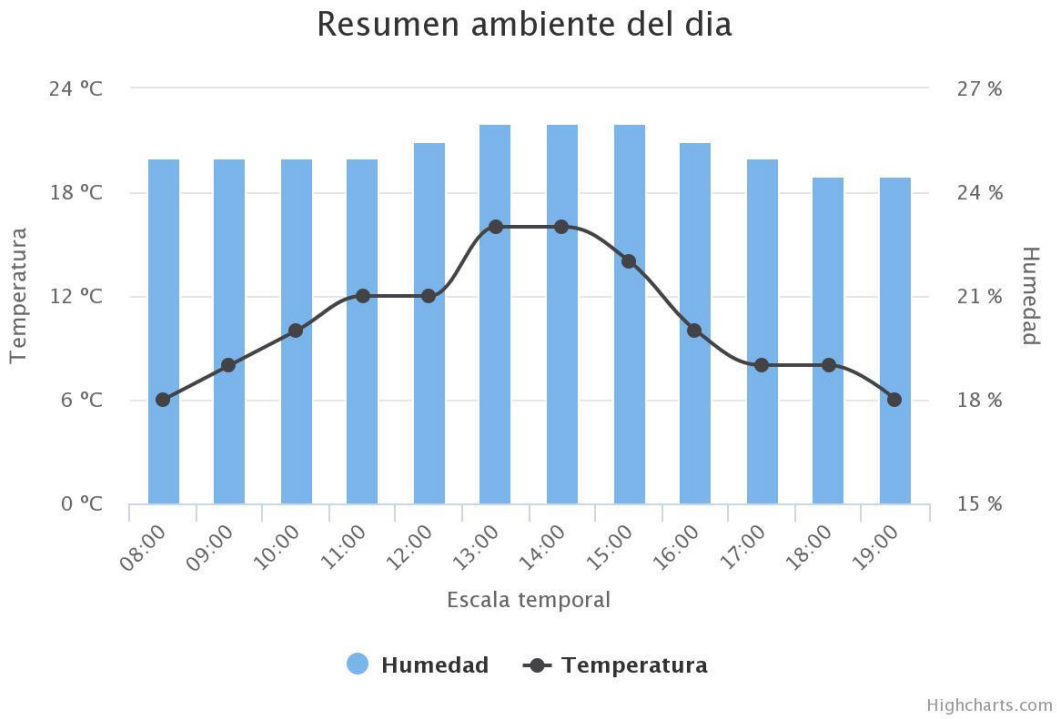


Fig. 34: Grafica resumen ambiente

6. Prueba de concepto:

En este apartado se intenta plasmar, lo anteriormente mencionado en un caso concreto de un posible caso real. Esta vivienda de una sola planta, la cual consta de baño, cocina, dos habitaciones y salón, tiene distribuidos como se puede ver en la Fig. 35: Plano vivienda, una serie de interruptores, conmutadores, puntos de luz y enchufes.

En este caso concreto sería necesario la instalación de 6 módulos arduinos con una raspberry pi para poder controlar todos los elementos de una forma sencilla y ordenada. Colocando los arduinos en cada una de las estancias de la vivienda, incluido el vestíbulo.

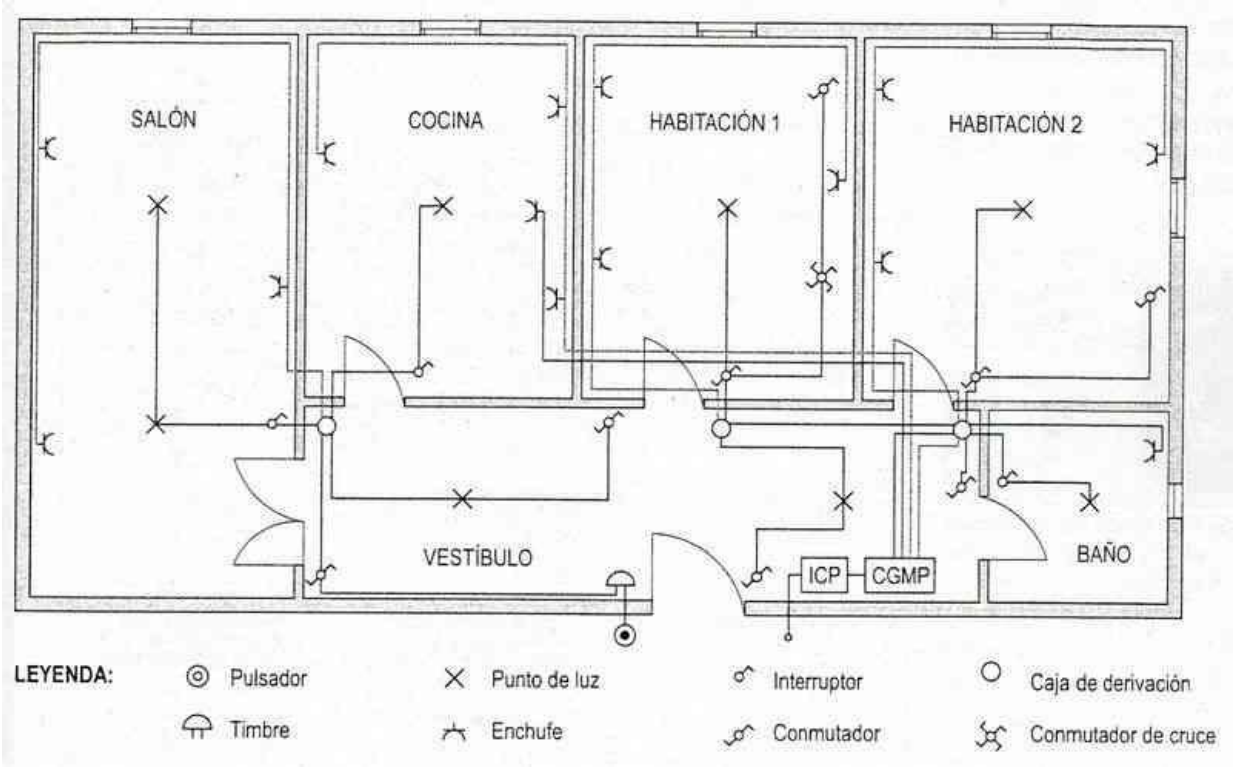


Fig. 35: Plano vivienda

En el caso de la cocina se necesitaría además del Arduino el cual cogería la alimentación de la propia red con un pequeño transformador. Un relé para poder apagar y encender las luces, además sustituyendo el interruptor sencillo por un conmutador se podría conmutar con el relé digital permitiendo el uso del interruptor físico.

En la práctica es mucho más beneficioso comprar relés dobles, por lo que el segundo relé podría ser utilizado para activar o desactivar el enchufe de la pared izquierda, el cual se trata de un enchufe en el que irían conectados pequeños electrodomésticos como plancha, tostadora, microondas, etc. Estos electrodomésticos en muchos casos traen consigo un pequeño led para identificar el modo en espera o stand-by, permitiendo un ahorro en el consumo apagando ese led, cuando sea conveniente.

Por otra parte, se instalaría un sensor de temperatura para tener registrada la temperatura y humedad de esta estancia. Además de 5 sensores de intensidad, para monitorizar el

gasto energético que producirán las luces de la cocina, el enchufe el cual se puede activar y desactivar, y los tres electrodomésticos de la cocina, la lavadora, el frigorífico y la placa vitrocerámica.

En el caso del salón se necesitaría, teniendo en cuenta el Arduino elemento fijo en todas las estancias. Al igual que en el caso de la cocina, se sustituye el interruptor físico por un conmutador, permitiendo el uso dual del conmutador físico y el relé digital. Además de poder controlar uno de los enchufes de la habitación, en este caso sería el enchufe situado en la pared derecha de la habitación. Este enchufe tendrá conectado todos los aparatos multimedia, televisión, video, equipo de sonido, etc. En el apartado sensores se añadirían, uno de temperatura y humedad para controlar esta estancia, y tres sensores de intensidad. El primero para monitorizar el gasto energético de las luces, el segundo para el enchufe, el cual se puede controlar y el tercer sensor para controlar el gasto en los otros dos enchufes, los cuales tienen un uso esporádico.

En el caso de la habitación 1, será necesario la utilización de los dos relés conjuntos ya que, al tratarse de un circuito conmutado con tres accionadores, se debe añadir un conmutador de cruce más para permitir el correcto funcionamiento de las demás llaves. En el apartado sensores, se añadiría el sensor de temperatura y humedad, junto con 2 sensores de intensidad para monitorizar el consumo de las luces y el consumo de los demás enchufes de la habitación.

En el caso de la habitación 2, será necesario la utilización de los dos relés conjuntos ya que, al tratarse de un circuito conmutado con dos accionadores, se debe añadir un conmutador de cruce para permitir el correcto funcionamiento de las demás llaves, muy parecido al caso de la habitación 1. En el apartado sensores, se añadiría el sensor de temperatura y humedad, junto con 2 sensores de intensidad para monitorizar el consumo de las luces y el consumo de los demás enchufes de la habitación.

En el caso del baño, se sustituye el interruptor sencillo por un conmutador, de esta forma al incluir el relé digital se puede crear un circuito conmutado para que funcionen tanto el conmutador físico como el domótico. Como en el caso del salón y de la cocina se utilizará el otro relé para activar o desactivar el enchufe del baño. En el apartado sensores, se añadiría el sensor de temperatura y humedad, junto con 2 sensores de intensidad para monitorizar el consumo de las luces y el consumo del enchufe, el cual podemos controlar.

En el caso del vestíbulo, al tener en los dos circuitos de luces, unos conmutadores ya instalados solo se debe añadir los relés digitales para crear los circuitos conmutados y permitir así el funcionamiento dual. En el apartado sensores, se añadiría el sensor de temperatura y humedad, junto con 2 sensores de intensidad para monitorizar el consumo de las luces, de forma individual.

De esta forma ya estaría toda la vivienda monitorizada y controlada, para poder obtener una vista más general se enumerarán los distintos elementos que se van utilizar y el precio total de todos los elementos necesarios para poder crear el sistema domótico.

Tabla 9: Presupuesto para la instalación

Descripción	Cantidad	Precio/Unidad (€)	Precio Total (€)
Raspberry Pi 2 Modelo B	1	34,00	34,00
Kit Cargador Carcasa Raspberry	1	10,99	10,99
Dongle Usb Bluetooth	1	7,16	7,16
Tarjeta Micro SD 16 Gb Clase 10	1	8,90	8,90
Arduino Uno R3	1	9,99	9,99
Juego de 5 pcs Arduino Micro	1	14,99	14,99
Fuente AC/DC 5V 600mA	6	2.21	13,26
Módulo de 2 Canales relé	6	2,00	12,00
Módulo sensor DHT-11	6	2,18	13,08
5PCS 30A Módulo ACS712	16	2,32	37,12
Cableado para conexiones Kit	2	7,99	15,98
Conmutador	3	6,86	20,58
Mano de obra (horas)	6	30	180
		Total	378,05

Como se puede apreciar, el mayor número de elementos en la lista son los sensores de intensidad, esto es debido a que una de las características más importantes para este sistema domótico recae en la monitorización de los consumos ya que es en este ámbito donde mayor repercusión tiene el sistema, favoreciendo el ahorro en el consumo eléctrico.

7. Conclusiones y Líneas Futuras

En este apartado, se expondrán las conclusiones a las que se ha llegado después de todo el proceso que ha ocurrido durante el desarrollo de este proyecto. Este desarrollo es el paso inicial para un abanico de múltiples posibilidades, usando lo desarrollado como base, estas líneas futuras se expondrán en el apartado 7.2 en detalle.

7.1. Conclusión:

En primer lugar, al usar tecnología muy conocida y de "*open source*", la comunidad que avala estos desarrollos permite resolver dudas muy comunes de forma muy sencilla evitando posibles retrasos con problemas que pueden aparecer al desarrollar un proyecto de estas características. Este motivo también beneficia a la alta capacidad de adaptación con la que se cuenta con esta tecnología, con la aparición de nuevos métodos y productos de este ámbito al ser una comunidad tan amplia y diversa aparecen todo tipo de ayudas que permitan afrontar nuevos proyectos de una forma más sencilla, resultando mucho más fructífero el comienzo de los mismos.

En lo que respecta al proyecto domótico, permite el gestionar la mayoría de las acciones cotidianas de una forma barata y sencilla. Incluyendo además el control del consumo del hogar, a simple vista pudiera parecer solo un mero registro de datos, pero expuestos en gráficas y contrastados con los distintos datos que también se registran. Se puede obtener mucha información desconocida para el habitante del hogar. Por ejemplo, en primer lugar y con una gráfica de consumos totales se podría observar el rango de tiempo, en el cual se produce un mayor consumo. Comparándolo con los precios de la electricidad permite descubrir cuando se pueden producir mayor consumo ya que el precio es menor.

Otra posibilidad, al visualizar una gráfica con el consumo de un actuador y el de la línea del proviene, así como las activaciones y desactivaciones del mismo permite descubrir consumos residuales producidos por electrodomésticos en modo stand-by, el cual pese a parecer un consumo muy ínfimo permite ahorrar dinero ya que este consumo se prolonga por periodos muy largos de tiempo.

Por estos y muchos otros escenarios que se irán presentando con el uso de la plataforma, es cuando se descubre la gran utilidad de este proyecto. Debido que, al mantener un registro de muchos de los datos del hogar, aparece la posibilidad de monitorizar y controlar los consumos. Por así decirlo, saber en qué se ha destinado cada uno de los euros gastados por la factura eléctrica y de esta forma saber cómo es posible reducir esta cantidad gracias a pequeños gestos que antes no se sabía que se debían llevar a cabo. El ejemplo más claro al apagar ciertos aparatos que no se dan uso, pero se dejan conectados a la red eléctrica, se consigue un ahorro considerable.

Por todas estas razones se puede concluir que el proyecto se trata de un proyecto adaptable que persigue ayudar con el control del hogar, de una forma lo más limpia posible repercutiendo lo menos posible en la estética del hogar, gracias a su principal ventaja el estar diseñado como módulos independientes que no exigen ni de una instalación eléctrico en concreto, como sería el caso si se hubiera desarrollado sobre la

tecnología PLC, la cual debe tener una instalación eléctrica no muy antigua, una continuidad en los cables ya que son estos cables los que se usan como medio para la transmisión de los datos. En definitiva, un desarrollo que desvela una infinidad de escenarios posibles que sean capaces de satisfacer la gran mayoría de las necesidades.

7.1.1. Ventajas:

Entre las ventajas más relevantes sobre el hardware se encuentran la total libertad para usar estas tecnologías ya que se tratan de tecnologías de uso libre, la facilidad de conseguir este tipo de productos en el mercado debido a su implantación en mayoría de los mercados globales, la facilidad con la que se puede a adaptar a las futuras tecnologías que puedan ir surgiendo y el bajo precio debido a la gran producción que se lleva a cabo por todo el mundo, sobre todo en China.

Por otro lado, sobre el beneficio que aporta el uso de este proyecto cabe destacar la capacidad de control sobre la iluminación del hogar, sobre la monitorización de los datos ambientales del mismo, como se ha explicado antes los grandes beneficios que aporta sobre el consumo energético, la libertad de poder cambiar muchas de las cosas al tratarse de un diseño propio el cual se puede adaptar con el paso del tiempo y la acumulación de la experiencia.

7.1.2. Desventajas:

Entre las desventajas más relevantes sobre el hardware cabe destacar que al tratarse de hardware genérico y el cual puede ser usado para muchas otras actividades, no es un hardware diseñado en el que los recursos son los específicos para esta tarea, de esta manera podremos tener casos de desaprovechamiento de los recursos del hardware.

Por otro lado, sobre el software cabe destacar que al tratarse de un desarrollo inicial la eficiencia del software puede no ser la esperada y se pueda mejorar reduciendo el tiempo y los recursos que se utilizan.

7.2. Líneas futuras:

En lo referente a las posibilidades que surgen después de este desarrollo, las líneas de actuación han ido surgiendo a medida que se avanzaba en el desarrollo.

La primera idea y que puede ser la más interesante, surge al tener ahora la posibilidad de encender y apagar luces y medir su consumo. En esta línea apareció la idea de combinar la domótica con el control de iluminación ya existente en el hogar, al tratar el relé digital como si fuese parte de un circuito conmutado, el cual estaría compuesto de las llaves que estuviesen ya instaladas podríamos interactuar tanto de la forma digital como la analógica, incluyendo el relé como un conmutador o interruptor cruzado. De esta forma se podría controlar la luz desde un interruptor físico o desde uno virtual, además permitiendo la comunicación directa empezada por el Arduino sobre la

Raspberry, cambiando el modo "*polling"* que se usa en este proyecto, podríamos detectar los dos tipos de encendido gracias al sensor de intensidad e ir registrando estos datos, de esta forma el cambio al sistema domótico sería un cambio más sencillo.

La segunda de las líneas por las que debería continuar este proyecto, sería la incorporación de un gran número de actuadores. En primer lugar, un regulador que en vez de como permite el relé digital, encender y apagar luces, permita modificar la intensidad para poder graduar las luces. Por otra parte, y junto al cambio de permitir el comienzo de la comunicación por parte del Arduino, se pueden incorporar sensores que avisen de eventos específicos, detectores de humo, de gases, de agua de ventanas abiertas. Este tipo de sensores que su funcionamiento es muy sencillo, pero precisan de una comunicación creada por el Arduino, permitirían el almacenar un mayor volumen de información que permitiría tener una mayor perspectiva del hogar y un mayor control de lo que ocurre en él, con beneficios en la seguridad y el ahorro energético.

La tercera línea, la cual ayudaría a la seguridad en el hogar consiste en simular la estancia en el hogar durante un periodo de vacaciones, pero no simulando una secuencia de acciones que podrían resultar siendo predecibles, si no debido a que se guarda un historial de todas las acciones en el hogar, se podría repetir un cierto periodo estacional resultando una secuencia aleatoria difícil de descubrir la simulación.

La cuarta vía podría ser la creación de multitramas, por las que el sistema sería capaz de mandar múltiples valores de sensores con una solo petición. De esta forma la recogida de datos seria solo mandar un comando, recibir toda la información en una trama y que el sistema, en este caso la raspberry sea el encargado de procesar la trama y realizar la inserción de datos conforme se hayan recibido los datos en la multitrama.

La quinta línea, debido a la diferencia entre los distintos datos que se obtienen. En el caso de la temperatura es un dato que puede recogerse cada más tiempo que en el caso de la intensidad de una línea, puesto que la temperatura no es tan variante en cortos periodos de tiempo. Por el contrario, la intensidad es algo variante en un corto plazo. Por ese motivo, se podría controlar el tiempo en los distintos arduinos, con el envío de tramas para poder tener un tiempo sincronizado en todos los elementos e ir almacenando las distintas medidas en los arduinos y mandar conjuntos de esas medidas con sus respectivos tiempos. Esto supondría un ahorro en el número de tramas enviadas, pero obteniendo un mayor número de muestras, haciendo más precisa la monitorización de las intensidades por parte del sistema.

Una de las líneas que resulta más atractiva, pero conlleva mayor complejidad es el diseño de sensores independientes que lleven a cabo funciones como puede ser detectar el estado de una ventana, pero que su funcionamiento no depende de un suministro continuo de luz, que puedan ser instalados en cualquier parte. Con este propósito se ha estudiado la idea de empezar a probar elementos como pequeñas placas solares, placas peltier, etc, para que ayudando a una pila y procurando un consumo muy bajo puede ser elementos autónomos.

8. Referencias

1. AMPPS. (2017). *AMPPS*. Recuperado el 16 de 09 de 2017, de http://www.ampps.com/
2. Aprediendo Arduino. (13 de 11 de 2016). *Wordpress*. Recuperado el 19 de 09 de 2017, de https://aprendiendoarduino.wordpress.com/tag/hc-05/
3. Arduino. (2017). *Arduino*. Recuperado el 08 de 09 de 2017, de https://www.arduino.cc/
4. Arduino. (2017). *Arduino*. Recuperado el 18 de 09 de 2017, de https://store.arduino.cc/arduino-uno-rev3
5. Caffelli, P. (28 de 09 de 2010). *eTecnologia*. Recuperado el 04 de 09 de 2017, de http://etecnologia.com/gadgets/funcionamiento-bluetooth
6. Cobo, J. G. (03 de 11 de 2014). *Hardware Libre*. Recuperado el 11 de 09 de 2017, de https://www.hwlibre.com/que-es-una-placa-sbc/
7. Crespo, E. (2017). *Wordpress*. Recuperado el 08 de 09 de 2017, de https://aprendiendoarduino.wordpress.com/2017/01/21/que-es-arduino-3/
8. Doutel, F. (25 de 01 de 2017). *Xataka*. Recuperado el 08 de 09 de 2017, de https://www.xataka.com/especiales/guia-del-arduinomaniaco-todo-lo-que-necesitas-saber-sobre-arduino
9. floston8400. (2017). *scribd*. Recuperado el 04 de 09 de 2017, de https://es.scribd.com/doc/75955999/Ventajas-y-Desventajas-Del-Bluetooth
10. Geeetech. (05 de 01 de 2015). *Geeetech*. Recuperado el 27 de 09 de 2017, de http://www.geeetech.com/wiki/index.php/2-Channel_Relay_module
11. Holbrook, S. (28 de 02 de 2014). *Make*. Recuperado el 27 de 09 de 2017, de https://makezine.com/2014/02/28/10-questions-for-raspberry-pis-eben-upton/
12. Llamas, L. (18 de 01 de 2017). *Luisllamas*. Recuperado el 28 de 09 de 2017, de https://www.luisllamas.es/arduino-intensidad-consumo-electrico-acs712/
13. manuti. (2 de Abril de 2013). *raspberryparatorpes*. Obtenido de https://raspberryparatorpes.net/empezando/raspi-config-configuracion-inicial-de-raspbian/
14. MCI Electronics. (2017). *Arduino.cl*. Recuperado el 19 de 09 de 2017, de http://arduino.cl/arduino-uno/
15. Museo Informática. (18 de 12 de 2013). *Blog Historia de la Informática*. Recuperado el 11 de 09 de 2017, de http://histinf.blogs.upv.es/2013/12/18/raspberry-pi/
16. Penalva, J. (22 de 12 de 2015). *Xataka*. Recuperado el 11 de 09 de 2017, de https://www.xataka.com/tag/raspberry-pi-a-fondo
17. Prometec. (2017). *Prometec*. Recuperado el 21 de 09 de 2017, de https://www.prometec.net/bt-hc05/
18. Prometec. (2017). *Prometec*. Recuperado el 22 de 09 de 2017, de https://www.prometec.net/sensores-dht11/
19. Prometec. (2017). *Prometec*. Recuperado el 21 de 06 de 2017, de https://www.prometec.net/reles/

20. Prometec. (2017). *Prometec*. Recuperado el 25 de 09 de 2017, de https://www.prometec.net/producto/sensor-corriente-20a/
21. Raspberry Pi Foundation. (2017). *Raspberry*. Recuperado el 11 de 09 de 2017, de https://www.raspberrypi.org/
22. Regata. (24 de 12 de 2012). Recuperado el 25 de 09 de 2017, de https://tallerarduino.com/2012/12/24/sensor-dht11-humedad-y-temperatura-con-arduino/
23. Wikipedia. (01 de 09 de 2017). *Wikipedia*. Recuperado el 04 de 09 de 2017, de https://es.wikipedia.org/wiki/Bluetooth
24. Wikipedia. (04 de 09 de 2017). *Wikipedia*. Recuperado el 05 de 09 de 2017, de https://es.wikipedia.org/wiki/Arduino
25. Wikipedia. (01 de 09 de 2017). *Wikipedia*. Recuperado el 11 de 09 de 2017, de https://es.wikipedia.org/wiki/Raspberry_Pi
26. Wikipedia. (10 de 09 de 2017). *Wikipedia*. Recuperado el 16 de 06 de 2017, de https://es.wikipedia.org/wiki/PHP
27. Wikipedia. (13 de 09 de 2017). *Wikipedia*. Recuperado el 16 de 09 de 2017, de https://es.wikipedia.org/wiki/LAMP
28. Wikipedia. (15 de 09 de 2017). *WIkipedia*. Recuperado el 16 de 09 de 2017, de https://es.wikipedia.org/wiki/MySQL
29. WIkipedia. (10 de 09 de 2017). *WIkipedia*. Recuperado el 16 de 09 de 2017, de https://es.wikipedia.org/wiki/Servidor_HTTP_Apache
30. WIkipedia. (15 de 09 de 2017). *WIkipedia*. Recuperado el 16 de 09 de 2017, de https://en.wikipedia.org/wiki/Master/slave_(technology)
31. Wikipedia. (s.f.). *Wikipedia*. Recuperado el 02 de 10 de 2017, de https://es.wikipedia.org/wiki/Dom%C3%B3tica