19-6-2014

|  |
| --- |
| Antonio Jesús Morales |



|  |  |
| --- | --- |
| I.E.S PICASSO | PROYECTO INTEGRADO: RASPTON “DOMÓTICA LOW COST” |

**ÍNDICE**

**0 – ÍNDICE Pág 2**

1. [**OBJETIVOS**](#objetivos) **Pág 3**
2. [**TEMPORALIZACIÓN/PLANIFICACIÓN**](#temporalizacion) **Pág 4**
3. [**RECURSOS NECESARIOS: HARDWARE/SOFTWARE**](#recursosnecesarios) **Pág 6**

[**Hardware:**](#recursosnecesarioshardware) **Pág 6**

[**Software:**](#recursosnecesariossoftware) **Pág 10**

[**Seguridad:**](#recursosnecesariosseguridad) **Pág 10**

1. [**ESTADO DE LA CUESTIÓN**](#estadocuestion) **Pág 11**

[**Recopilación de la información**](#estadocuestionrecopilacion) **Pág 11**

[**Identificación de las necesidades**](#estadocuestionidentificacion) **Pág 12**

[**Identificación de los aspectos que facilitan o dificultan el proyecto**](#estadocuestionidentificacion2) **Pág 12**

1. [**FUNCIONES QUE LLEVARÁ A CABO EL PROYECTO**](#funcionesquellevara) **Pág 15**

[**Página Web**](#funcionesquellevaraweb) **Pág 15**

[**Instalación y configuración del servidor**](#funcionesquellevarainstalacion) **Pág 17**

[**Infraestructura de red**](#funcionesquellevarainfraestructurea) **Pág 20**

1. [**PUESTA EN MARCHA DEL PROYECTO**](#puestaenmarcha) **Pág 22**

[**Servicios**](#puestaenmarcha)  **Pág 22**

[**Aplicación Web**](#puestaenmarcha) **Pág 22**

[**Pruebas de red**](#puestaenmarcha) **Pág 22**

1. [**ANEXO**](#anexo) **Pág 22**
2. [**BIBLIOGRAFÍA**](#bibliografia) **Pág 23**
3. **OBJETIVOS**

El objetivo principal del proyecto consiste en el aprendizaje de lenguajes de programación tales como: PHP, SHELL SCRIPTS, HTML, PROCESSING, PYTHON. Aprendizaje del funcionamiento de los servicios de los sistemas operativos basados en Linux, creación de programas informáticos mediante scripts de sistema. Aprendizaje de los conceptos básicos de electrónica y automatismo. Todo esto para aplicarlo a una maqueta de una casa que realizará las siguientes funciones:

* Abrir puerta de garaje
* Cerrar puerta de garaje
* Parar puerta de garaje
* Encender 3 luces diferentes
* Controlar la corriente eléctrica de un enchufe
* Controlar apagado y encendido de una alarma

Todos estos elementos se controlarán mediante interfaz web que estará disponible mediante una conexión wifi que emitirá el aparato.

Este proyecto se realizará enfocándolo más que nada en el aprendizaje de los procedimientos necesarios para informatizar los elementos de una casa, no será un proyecto con aplicaciones finales. Una vez terminado se poseerán conocimientos suficientes para aplicarlos en un caso real.

1. **TEMPORALIZACIÓN / PLANIFICACIÓN**

|  |  |  |
| --- | --- | --- |
| SEMANA | HORAS | TAREAS |
| 1 | 36 | Investigación sobre el material y software necesario. Planificación de las funcionalidades. |
| 2 | 36 | Investigación y primeras pruebas de instalación y funcionamiento de la RaspBerry Pi, servicios como ssh, hostapd (AP wifi). |
| 3 | 36 | Perfeccionamiento en la configuración de hostapd, realización de copias de seguridad, e instalación de vsftpd. |
| 4 | 36 | Pruebas con diferentes servidores web: lighthttp, apache. Utilizando finalmente el Apache. Configuración e Instalación de PHP5, MYSQL y varias extensiones imprescindibles. |
| 5 | 36 | Investigación de como ejecutar scripts sh como administrador mediante php. Instalación y configuración del servidor de nombres de dominio BIND9. |
| 6 | 36 | Instalación y configuración del CMS joomla 3.3.0. Investigación sobre los pines GPIO de la raspberry. Investigación de los elementos electrónicos necesarios. |
| 7 | 36 | Pruebas de control de pines GPIO mediante consola. Pruebas de control de pines GPIO mediante interfaz Web. |
| 8 | 36 | Planteamiento de las funcionalidades finales de domótica, incluirán: 3 luces y un enchufe en vez de 3 luces y un ventilador. Investigación de control de un motor bidireccional mediante pines gpio. |
| 9 | 36 | Construcción de los scripts necesarios para el control de los 3 leds y el enchufe. Construcción de página web PHP para ejecutar los scripts mediante web. |
| 10 | 36 | Primeras pruebas con Chip L293D usado para controlar el movimiento bidireccional del motor que impulsará el mecanismo de apertura y cierre de la puerta. Descubrimiento de la inviabilidad de utilizar los pines GPIO para mi propósito, investigación sobre la utilización de ARDUINO para ejecutar ordenes emitidas por la RaspBerry. Pruebas de comunicación por serial de RaspBerry y Arduino. |
| 11 | 36 | Construcción de los scripts necesarios para la arduino y para la Raspberry, que controlaran el movimiento bidireccional del motor. A la vista del funcionamiento tan bueno decido migrar el control de los 3 leds y el enchufe mediante serial con arduino. |
| 12 | 36 | Perfeccionamiento del control de todos los dispositivos mediante scripts sh que ejecutan a su vez uno python. Descubrimiento de lo mal que funciona Joomla en la Raspberry, he decidido crear una pequeña pagina web que controlará los elementos mediante php. |
| 13 | 36 | Construcción de la maqueta y el mecanismo de la puerta, investigación e implementación de un sensor de movimiento usado como alarma, esto formará parte del último elemento a utilizar en la maqueta. Construcción de los script necesarios y el reproductor omxplayer. |
| **Total H** | | **468** |

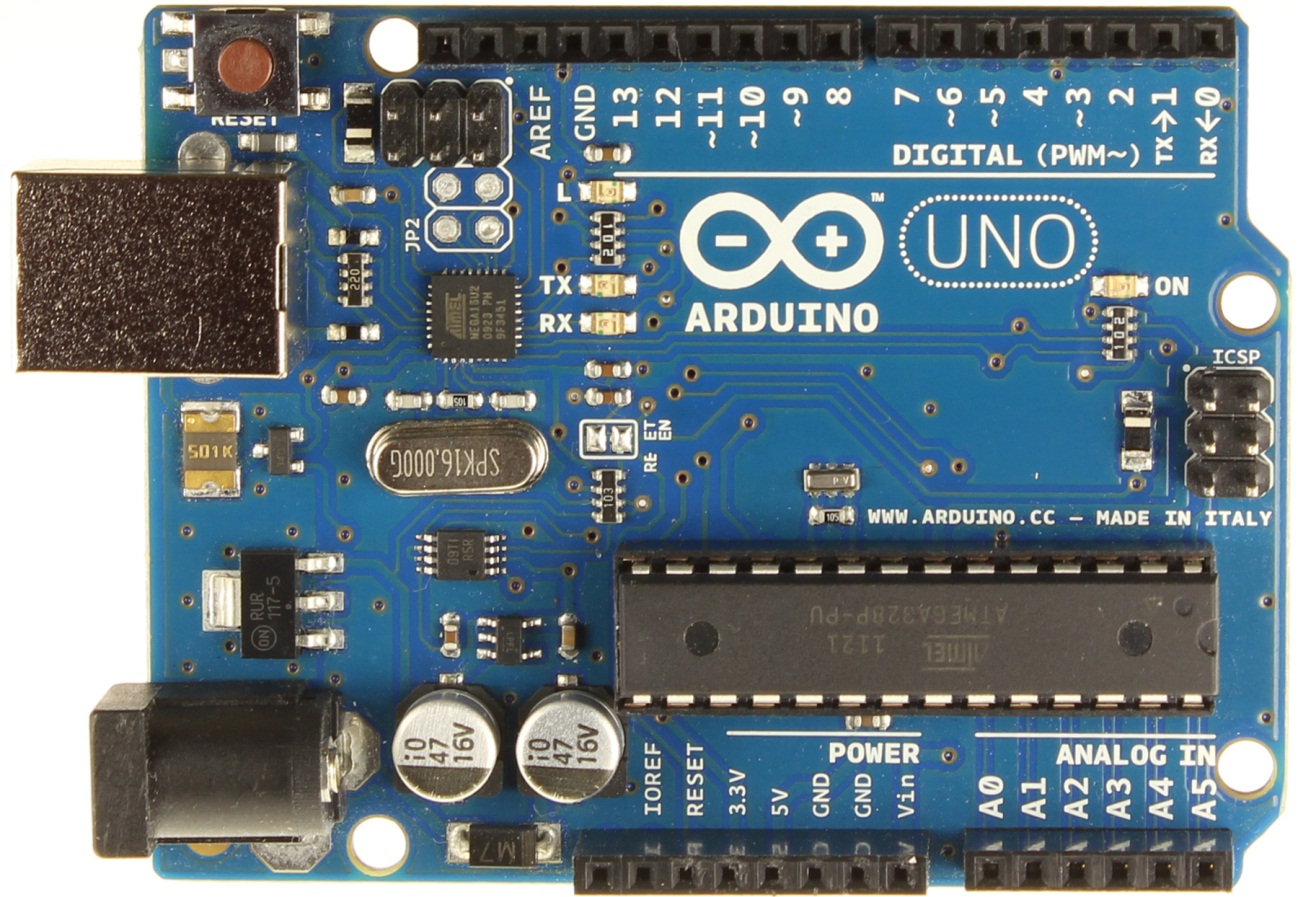
1. **RECURSOS NECESARIOS : HARDWARE Y SOFTWARE**

* **HARDWARE**

**RASPBERRY PI (COMPARACIÓN ENTRE MODELOS)**

|  | | [**Raspberry Pi**](http://www.raspberryshop.es/)**Model A** | [**Raspberry Pi**](http://www.raspberryshop.es/)**Model B** |
| --- | --- | --- | --- |
| **Placa**[**Raspberry Pi**](http://www.raspberryshop.es/) | | Raspberry Model A | Raspberry Model B |
| **Precio** | | 25€ | 45€ |
| **SoC** | | Broadcom BCM2835 (CPU + GPU + DSP + SDRAM + puerto USB) | |
| **CPU** | | ARM1176JZF-S a 700 MHz (familia ARM11) | |
| **GPU** | | Broadcom VideoCore IV, OpenGL ES 2.0, -2 y VC-1 (con licencia), 1080p30 H.264/MPEG-4 AVC | |
| **Memoria (SDRAM)** | | 256 MB (compartidos con la GPU) | 512 MB (compartidos con la GPU) desde el 15 de octubre de 2012 |
| **Puertos USB 2.0** | | 1 | 2 (vía hub USB integrado) |
| **Entradas de vídeo** | | Conector [MIPI] CSI que permite instalar un módulo de cámara desarrollado por la RPF | |
| **Salidas de vídeo** | | Conector RCA (PAL y NTSC), HDMI (rev1.3 y 1.4), Interfaz DSI para panel LCD | |
| **Salidas de áudio** | | Conector de 3.5 mm, HDMI | |
| **Almacenamiento integrado** | | SD / MMC / ranura para SDIO | |
| **Conectividad de red** | | Ninguna | 10/100 Ethernet (RJ-45) via hub USB |
| **Periféricos de bajo nivel** | | 8 x GPIO, SPI, IC, UART | |
| **Reloj en tiempo real** | | Ninguno | |
| **Consumo energético** | | 500 mA, (2.5 W) | 700 mA, (3.5 W) |
| **Fuente de alimentación** | | 5 V vía Micro USB o GPIO header | |
| **Dimensiones** | | 85.60mm x 53.98mm (3.370 x 2.125 inch) | |
| **sistemas Operativos soportados** | Debian (Raspbian), Fedora (Pidora), Arch Linux (Arch Linux ARM),Slackware Linux, RISC OS. | | |

**ARDUINO UNO**



La[Arduino Uno](http://www.olimex.cl/product_info.php?products_id=756) posee:

* 14 entrada/salida digitales, de los cuales 6 pueden se usados como salidas PWM
* Posee 6 entradas analógicas
* Los pin 0 y 1 pueden funcionar como RX y TX serial.
* Un oscilador de crystal de 16 MHz
* Conector USB
* Un jack de poder
* Una conector ICSP
* Botón de Reset

La [Arduino UNO](http://www.olimex.cl/product_info.php?products_id=756) posee todo lo que se necesita para manejar el controlador, simplemente se conecta a un computador por medio del cable USB o se puede alimentar utilizando una batería o un adaptador AC-DC. Si se conecta por USB, la alimentación externa no es necesaria.  
  
Para programar sólo necesita el IDE de Arduino, que se encuentra en la sección de descargas en el sitio web de www.arduino.cc   
  
Algunas características son:

|  |  |
| --- | --- |
| Microcontroller | ATmega328 |
| Operating Voltage | 5V |
| Input Voltage (recommended) | 7-12 V |
| Input Voltage (limits) | 6-20 V |
| Digital I/O Pins | 14 (of which 6 provide PWM output) |
| Analog Input Pins | 6 |
| DC Current per I/O Pin | 40 mA |
| DC Current for 3.3V Pin | 50 mA |
| Flash Memory | 32 KB (of which 0.5 KB used by bootloader) |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Clock Speed | 16 MHz |

Las Entradas analógicas son de 10 bits, por lo que entregan valores entre 0 y 1023. El rango de voltage está dado entre 0 y 5 volts, pero utilizando el pin AREF disponible, este rengo se puede variar a algún otro deseado.

**LISTA DE LA COMPRA**

|  |  |  |
| --- | --- | --- |
| DISPOSITIVO | **PRECIO** | **DESCRIPCIÓN** |
| RaspBerry Pi mod. B | 45 € |  |
| Arduino UNO | 14 € |  |
| Placa de prototipos 30x15 | 10 € | Placa con orificios utilizada para conectar componentes electrónicos. |
| Batería de 12500 Ma salida usb | 23 € | Batería con 2 salidas usb una de ellas ofrece 2 A, imprescindible para hacer funcionar el motor. |
| Array de relés | 8 € | Array de 8 relés de 5V que maneja voltajes alternos de 220V. |
| GearBox | 12 € | Conjunto de engranajes y motor que permiten una salida de rpm variable según necesidad. |
| Adaptador wifi | 8 € | Adaptador usb wifi. |
| Cables | 2€ | Gratis si los reciclas de algún cable. |
| Integrado L293D | 1,5€ | Chip controlador de motor. |
| 3 Resistencias 600 Ohm | 1,2€ |  |
| Enchufe | 4 € |  |
| 2 finales de carrera | 4 € |  |
| Hilo o cuerda finita |  |  |
| 3 Leds 3 W | 3,5€ | Funcionan a 3.3V y necesitan ser pegados a un trozo de hierro para disipar el calor generado |
| Marquetería | 4 € |  |
| Pintura acrílica | 1,5€ |  |
| Cable micro-USB | 1€ |  |
| Latiguillo RJ-45 | 1€ |  |
| TOTAL | **143.5 €** | |

* **SOFTWARE**

A continuación muestro una tabla descriptiva con el software que he utilizado en el proyecto. En el punto 5 B “Funciones que llevará a cabo la infraestructura hardware/software propuesta”, se exponen con más detalles.

|  |  |
| --- | --- |
| SOFTWARE | **DESCRIPCIÓN** |
| Raspbian | Sistema operativo base |
| Hostapd | Crea un AP wifi |
| Win32Diskimager | Permite montar imágenes en sd y hacer copias de seguridad |
| SSH | Este servicio te permite conectarte a la consola vía remoto |
| VSFTP | Este servicio permite navegar en los ficheros del servidor de forma remota |
| PILA LAMP | Constituido por los servicios: Apache2, php5, mysql. Sirven para crear un servidor web completo. |
| BIND9 | Permite crear un servidor de resolución de nombres de dominios |
| Omxplayer | Reproductor multimedia vía consola de comandos |
| DHCP | Servidor de direcciones ip dinámicas. Se encarga de proveer direcciones Ip a las máquinas que se conectan a la red wifi |

* **SEGURIDAD**

Respecto a la seguridad que se ha implementado en este proyecto, está el acceso a la red wifi. Este acceso está limitado por una contraseña WPA, este método de autentificación utiliza el cifrado RC4 y bastante seguro si la contraseña contiene caracteres mayúsculas, minúsculas y números con una longitud mínima de 8 caracteres. Esto permite que el control de los elementos de domótica solo pertenezca al poseedor de la contraseña.

1. **ESTADO DE LA CUESTIÓN**

Se sabe ya de aquí a unos años atrás que la inclusión de la tecnología en las casas estará cada vez más a la moda. Las casas del futuro tendrán personalidad propia, al poder controlarse y/o automatizar diferentes tareas, que harán cada vez más fácil la vida de sus habitantes. Este no es un tema fácil ni barato, ya que hoy día la tecnología que hace posible esto, es muy cara y requiere de personal con conocimientos muy específicos. En mi afán por encontrar soluciones “LOW COST” que poder aplicar en mi propia casa, se me ocurre construir mi propio sistema de domótica, o al menos aprender sobre la base de su funcionamiento.

Este proyecto que he realizado no se puede considerar un producto que se pueda vender ni comercializar, es más que nada una fuente de conocimiento e información, para personas que les guste el tema. Con una pequeña cantidad de dinero y haciéndolo uno mismo, se puede llegar a convertir una humilde casa en un hogar del futuro que responda a nuestras necesidades y que se pueda controlar con un Smartphone desde el sofá de tu casa. Obviamente mi contribución en este proyecto es más bien minúscula, comparado con la gran cantidad de información presente en internet, de la cual se ha visto nutrido mi proyecto, pero para gente que estén iniciándose en el tema debería bastar para alimentar las inquietudes de las personas que se estén iniciando en este interesante y novedoso mundo. Otra de las cosas que hace interesante a este proyecto, es la satisfacción que se recibe al ver nuestra propia creación dándole vida a nuestra casa.

* **Recopilación de la información:**

Este es con diferencia el punto más importante de un proyecto de este tipo, antes de nada hay que saber que queremos conseguir y una vez planteado el funcionamiento de nuestro sistema, tenemos que ir recopilando la información que esté en nuestra mano, tanto de los materiales que vamos a utilizar como de los factores que pueden intervenir a la hora de intentar llevarlo a cabo.

* **Identificación de necesidades:**

Claro está que cuando uno va a diseñar un sistema de este tipo, es porque va a ser enfocado a solventar una necesidad. Por ejemplo si vivimos en una comunidad de vecinos o en una urbanización que tiene una puerta de garaje común para todos los vecinos. Y por unas u otras razones se requiere que los que viven ahí puedan abrirla desde el Smartphone o pc, desde wifi o desde internet.

Ahí está la necesidad, que mediante el diseño de nuestro sistema se debe solventar. En mi caso concreto el diseño del proyecto está realizado de tal forma que se puedan saciar las necesidades de poder controlar las luces, enchufes, apertura de puertas y alarma de una casa.

* **Identificación de los aspectos que facilitan o dificultan el desarrollo del proyecto**

Desde la experiencia de ejecución de este proyecto digo que diseñar un sistema de este tipo no resulta cosa de una mañana. Es un largo proceso de prueba y error en el que se aprenden de los errores. A priori es muy difícil identificar los aspectos que nos costarán más trabajo desarrollas y los que nos resultarán más fáciles. Esto se descubre cuando te pones a trabajar en el proyecto y te vas encontrando con los diferentes errores que te imposibilitan desarrollar la aplicación para satisfacer las necesidades requeridas.

Mi proyecto no es diferente, en un principio investigué por internet las formulas y los trabajos que otras personas habían aplicado para solucionar problemas parecidos a los míos. La mayoría de las veces buscando en páginas en inglés, ya que casi toda esta información se encuentra en ese idioma, sin darse cuenta uno aprende inglés cuando motivado por la solucionar un problema se lee textos en inglés que no leería en otras circunstancias jaja.

Al empezar este proyecto uno de los problemas más grandes era el de crear la red wifi. Para ello probé diferentes configuraciones y adaptadores de red propuestos en una larga lista de páginas web que me ofrecía Google. Al final conseguí solucionar el problema y estabilizar la red wifi utilizando la recopilación de toda la información que vi y mezclándola de tal forma que se adaptarse a mi necesidad.

El siguiente problema fue el de poder ejecutar comandos desde el lenguaje para plataformas web PHP, por defecto la configuración de PHP no te permite ejecutar ciertas ordenes, como por ejemplo “**sudo halt –n”,** ya que el usuario que corre esas órdenes es el usuario de apache “www-data”. Este punto era imprescindible para poder seguir con el proyecto. Tras tardes y tardes de investigación, conseguí dar con una solución que fue la de cambiar el usuario activo de PHP, lo cambie por el usuario “pi” que tiene la mayoría de los permisos que iba a necesitar para ejecutar los scripts.

Otro de los problemas que casi me plantean abandonar el proyecto, fue el del control de los pines GPIO. Por los paso a paso que he redactado en el documento [8-Control pines GPIO.docx](8-Control%20pines%20GPIO.docx). Puede parecer fácil, pero sin conocimiento previo y buscando en internet todas las pruebas que realizaba fracasaban, hasta que el tiempo me dio la capacidad de discernir las cosas que estaba haciendo mal y así corregirlas, de esa forma y ayudado por otro gran listado de páginas webs visitadas logré hacer funcionar los pines GPIO de la RaspBerry. En este proceso también aprendí mucho sobre el funcionamiento de las señales eléctricas de entrada y salida a 3.3V, que me sirvieron para entender casi a la primera el funcionamiento de la arduino.

Otro problema que vino después fue la de soportar el ingente consumo de energía que demandaba los componentes, la raspberry no está diseñada para el propósito de alimentar ni siquiera un led de 3W ¿Cómo iba a hacer funcionar un motor? La solución vino de la mano de un microcontrolador llamado Arduino. Este si estaba preparado para satisfacer las necesidades eléctricas de los componentes utilizados. Por lo que en vez de conectar estos componentes directamente a los pines GPIO de la Raspberry, los conecte a varios pines de entrada de la arduino que gestionaría las señales provenientes activando las salidas correspondientes y haciendo de “intermediario”.

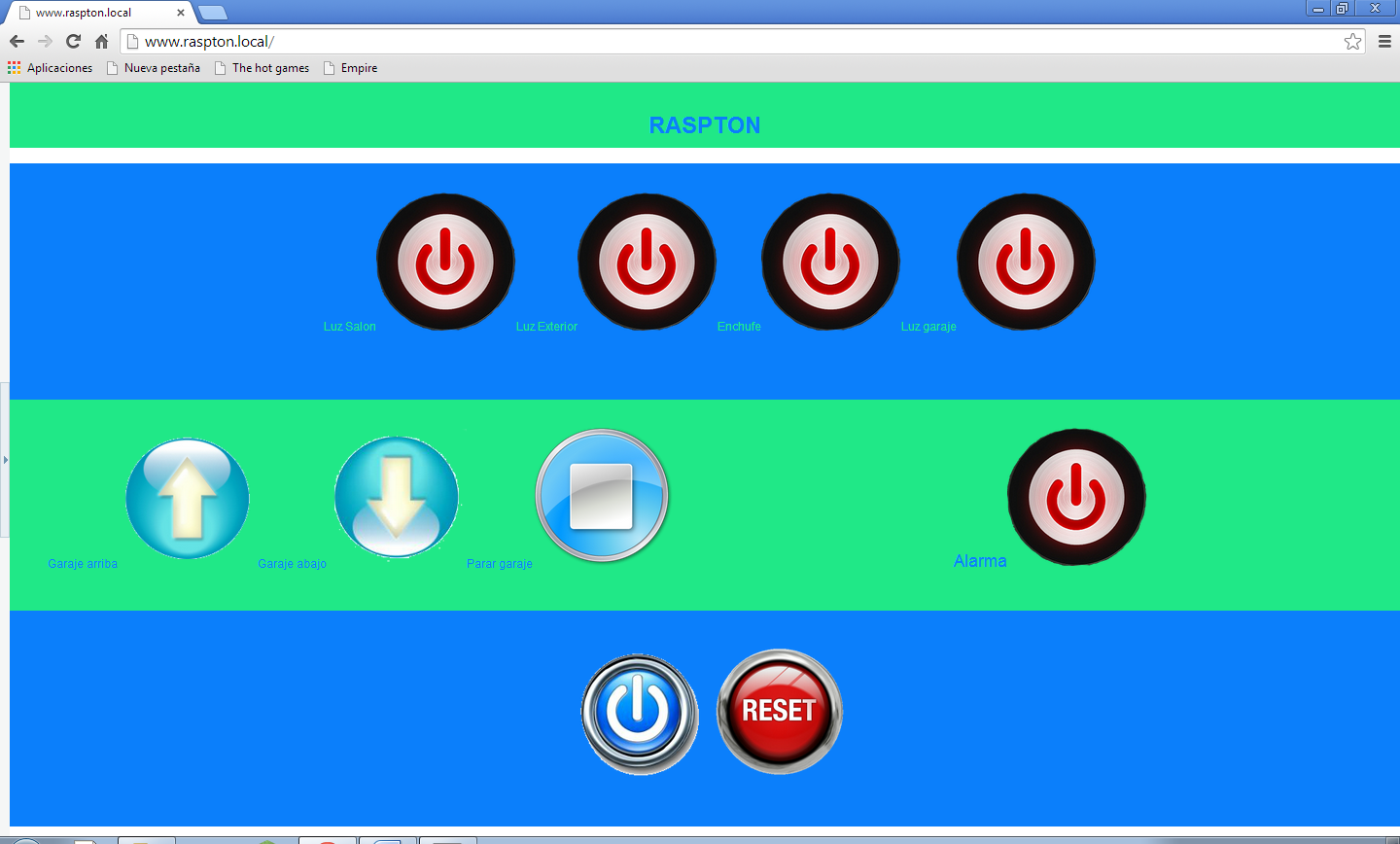
Pero sin lugar a dudas, el mayor reto al que me enfrenté fue el control bidireccional del motor que abriría y cerraba la puerta. Lo que hace girar un motor eléctrico de corriente discontinua a un lado o a otro es la posición del polo positivo y el polo negativo en una y otra patilla del motor. Una vez sabido esto intenté diseñar un circuito eléctrico mediante transistores, luego mediante relés, todo esto podría tener éxito pero cargaban de una gran complejidad al proyecto, necesitándose muchas horas para formarse en los conocimientos precisos para poder construirse. Debido a esto seguí investigando para encontrar otra solución, hasta que di con un integrado (chip), que simplificaba esta tarea de una forma increíble. Su nombre es L293D, y su funcionamiento se puede ver en [9.4-Control del motor mediante GPIO y el integrado L293D.docx](9.4-Control%20del%20motor%20mediante%20GPIO%20y%20el%20integrado%20L293D.docx).

Un problema que dio muchos quebraderos de cabeza fue el diseño del script Python y el de la comunicación por serial con la arduino. Había escuchado hablar mucho sobre la conexión serial pero nunca había realizado pruebas con ella, hasta que la necesidad de eliminar el extenso cableado que unía la raspberry con la arduino (cerca de 9 cables), me hizo indagar sobre el tema y diseñar un método alternativo de controlar la domótica. El diseño y correcto funcionamiento de la conexión serial simplifico bastante el diseño del proyecto. Dejando claro 2 módulos operativos. El módulo de la Raspberry que se podía quitar desenchufando el cable de red, el usb, el cargador y el gpio de la alarma. De la otra forma cada vez que se quisiera extraer la Raspberry para utilizarla para otra función había que remover todos los cables GPIO que no eran nada fáciles de volver a poner después.

Otro de los problemas presentados fue casi finalizando el plazo de entrega del proyecto, se quería incluir una alarma que se pudiese activar y desactivar, pero prueba tras prueba fallaba. Siendo este un problema menor se solucionó implementando más ficheros para que el control por Web fuese correcto. Aún se puede mejorar muchísimo más esta alarma, pero eso lo dejo para un futuro.

1. **FUNCIONES QUE LLEVARÁ A CABO LA INFRAESTRUCTURA HARDWARE/SOFTWARE PROPUESTA.**
2. Página Web

Nombre archivo: index.php.



* Información que se va a gestionar.

La aplicación web de Raspton gestionará 10 elementos:

· **Luz del salón** 🡪 Cuando se pulsa este botón de envío manda un valor a la misma página, este valor es evaluado por un switch case, que ejecuta exec("sh /home/pi/gpioscripts/luces/arduino\_salon.sh encender"), al mismo tiempo el script arduino\_salon.sh escribe en el archivo estado\_salon un 1. Este archivo es evaluado por un if antes de mostrar el botón, si este archivo es 1, se mostrará el botón encendido, si es 0 se mostrará el botón apagado como se muestra en la foto. Cuando el botón se encuentra encendido y lo pulsamos se ejecuta el siguiente comando : exec("sh /home/pi/gpioscripts/luces/arduino\_salon.sh apagar");

· **Luz exterior 🡪** Su funcionamiento es igual que el anterior pero el script que ejecuta es el **arduino\_exterior.sh encender/apagar**

**· Luz garaje 🡪** Más de lo mismo pero en este caso el script es **arduino\_garaje.sh encender/apagar**

**· Enchufe 🡪** Aunque ya no sean luces, el funcionamiento se rige de igual forma que los anteriores pero ejecutando el comando **arduino\_ventilacion.sh encender/apagar**

**· Garaje arriba 🡪** Este botón a diferencia de los anteriores no se modifica cuando se pulsa, se mantiene siempre el mismo botón, el comando que se ejecuta cuando se pulsa es: **exec("sh /home/pi/gpioscripts/motor/arduino\_motor.sh up");**

**· Garaje abajo 🡪** Igual que en el caso anterior pero ejecutando el comando : exec("sh /home/pi/gpioscripts/motor/arduino\_motor.sh down");

**· Parar garaje 🡪** Este botón sigue el mismo patrón que los dos anteriores pero parando el motor con el comando : exec("sh /home/pi/gpioscripts/motor/arduino\_motor.sh off");

**· Alarma 🡪** Este botón sigue el mismo funcionamiento que los cuatro primeros, modificando el aspecto del botón cuando este es pulsado. Cuando está apagado y se pulsa ejecuta los siguientes comandos: exec('sh /home/pi/alarma/start\_alarm.sh >/dev/null &');

El if que verifica si debe mostrar el botón encendido o apagado se basa en el contenido del fichero /home/pi/alarma/interruptor. Si este fichero se encuentra en 0 es que la alarma está apagada y por ende se muestra el botón apagado. Si por el contrario el fichero contiene un 1, se muestra el botón encendido que ejecuta el siguiente comando : exec('echo 0 > /home/pi/alarma/interruptor'). El script de sensor\_ir.sh modifica este fichero y lo pone en 1 a la hora de iniciarse la alarma.

· **Apagado 🡪** El botón de apagado simplemente ejecuta el comando : exec('sudo halt -n');

· **Reinicio 🡪**  Este botón ejecuta: exec('sudo reboot');

1. Instalación y configuración del servidor:
   * Este proyecto estará desarrollado sobre Raspbian, es una distribución del sistema operativo GNU/Linux y por lo tanto libre basado en [Debian Wheezy](http://es.wikipedia.org/wiki/Debian) (Debian 7.0) para la [placa computadora (SBC)](http://es.wikipedia.org/wiki/Placa_computadora) [Raspberry Pi](http://es.wikipedia.org/wiki/Raspberry_Pi), orientado a la enseñanza de [informática](http://es.wikipedia.org/wiki/Inform%C3%A1tica). El lanzamiento inicial fue en [junio](http://es.wikipedia.org/wiki/Junio) de 2012. **PODEMOS VER SU INSTALACIÓN EN** [**1-Instalación de Raspbian.docx**](1-Instalación%20de%20Raspbian.docx)**.**
   * Servidor UDHCP y HOSTAPD, estos 2 servicios se utilizan para ofrecer una red inalámbrica. El servidor UDHCP es un servidor dhcp muy liviano que configuramos para que ofrezca direcciones ip a la red WLAN0 que levantaremos con el servidor HOSTAPD. La red wifi que levantaremos se llamará Raspton y tendrá una contraseña segura WPA. **PODEMOS VER SU INSTALACIÓN EN** [**2-Instalación hostapd.docx**](2-Instalación%20hostapd.docx)**.**
   * Para las copias de seguridad utilizaremos Win32DISKIMAGER para Windows nos permite montar imágenes y hacer copias de seguridad en tarjetas sd, este mismo programa se utiliza para montar Raspbian en la SD. **PODEMOS VER SU USO EN** [**3-Copias de seg.docx**](3-Copias%20de%20seg.docx)
   * Para la conexión remota utilizaremos SSH que viene instalado y configurado por defecto en Raspbian.
   * Utilizaremos el servicio VSFTPD, que es un servidor ftp más liviano que el típico proftpd, este servidor nos proporcionará acceso remoto a los archivos ubicados en la Raspberry. **PODEMOS VER SU INSTALACIÓN EN** [**4-Instalación vsftpd.docx**](4-Instalación%20vsftpd.docx)
   * Instalación de la pila LAMP. La pila LAMP hace referencia a una serie de servicios que proporcionan una experiencia web muy amplia. Estos servicios son: APACHE WEB SERVER, PHP5, MYSQL. Mediante estos servicios proporcionaremos todo el interfaz web al proyecto, el usuario interactuará con ellos directamente para ejecutar las funciones propuestas. **PODEMOS VER SU INSTALACIÓN EN** [**5-Instalación Pila LAMP.docx**](5-Instalación%20Pila%20LAMP.docx) **Y UN ANEXO PARA PODER EJECUTAR COMANDOS MEDIANTE PHP EN LINUX** [**5.1-Ejecutar comandos linux desde php.doc**](5.1-Ejecutar%20comandos%20linux%20desde%20php.doc)**.**
   * También proveeremos al sistema de un servidor de nombres de dominio que trabajará para la red WLAN0, este servidor nos permitira crear una zona dns llamada raspton.local, gracias a esto poniendo en el navegador <http://www.raspton.local>, podremos acceder a la carpeta raiz del servidor web. El servidor que utilizaremos se llama BIND9, y **PODEMOS VER EL PASO A PASO** **EN EL FICHERO:** [**6-Instalación servicio DNS.docx**](6-Instalación%20servicio%20DNS.docx)
   * Me propuse a instalar el panel de control de la domótica en un Joomla, pero una vez hecho y probado, me he dado cuenta que no trabaja muy bien sobra la Raspberry, con que al final me he decantado por hacer la página a mano utilizando html, php y css. Con esto me aseguro que la ejecución de eventos por web sea más rápida y precisa. No obstante podemos ver el paso a paso de la instalación en el fichero : [7-Instalación CMS Joomla 3.3.0.docx](7-Instalación%20CMS%20Joomla%203.3.0.docx)
   * También se ha instalado un reproductor multimedia para consola de comandos, que me servirá para reproducir el sonido de la alarma cuando está sea activada, el servicio se llama omxplayer y podemos ver su instalación junto con la configuración de la alarma en el fichero: [9.7-Alarma utilizando sensor de movimiento.docx](9.7-Alarma%20utilizando%20sensor%20de%20movimiento.docx)
   * Se ha utilizado el software para Windows PUTTY para la conexión remota a la consola de comandos de la Raspton. Podemos ver su funcionamiento en una parte del fichero: [1-Instalación de Raspbian.docx](1-Instalación%20de%20Raspbian.docx)
   * Otro programa para Windows que se utilizará es el filezilla, que es un cliente ftp que nos permitirá conectarnos al servidor ftp de la máquina. El funcionamiento del mismo podemos verlo en el apartado: [4-Instalación vsftpd.docx](4-Instalación%20vsftpd.docx)
   * También utilizaremos el programa para Windows Arduino IDE, este software nos permitirá conectarnos a la arduino y poder compilar y subirle el código fuente que queramos.
   * El programa que se ejecuta en la arduino ha sido diseñado por mi, este recibe por serial una orden, la interpreta y activa en consecuencia el elemento correspondiente, ya puede ser un led, el motor a una dirección, o el enchufe. Este código fuente podemos verlo en : [scripts\skecth arduino\Raspton\_arduino v1.1.txt](scripts/skecth%20arduino/Raspton_arduino%20v1.1.txt)
   * Por otro lado, el programa que se encarga de mandar la orden a la arduino, está escrito en python, y también diseñado por mí. Este programa conecta por serial usb a la arduino y manda una orden que se pasa por parámetro a la hora de ejecutar el script. El código fuente podemos verlo en: [scripts\script python\arduino.py.txt](scripts/script%20python/arduino.py.txt).
   * Para el control de las luces de las 3 habitaciones, el motor, y el enchufe. En primera instancia se diseñaron scripts que hacían uso de los puertos GPIO de la propia RaspBerry, pero al encontrar una forma más óptima de hacerlo gracias a la conexión serial con la arduino, estos se quedaron sin utilizar, pero se pueden ver en las carpetas de scripts ubicada en el directorio raíz del CD entregado. Los script que hacen uso de los pines GPIO están nombrados como:

**salon.sh** 🡪 enciende y apaga la luz del salón.

**Exterior.sh** 🡪 enciende y apaga la luz del exterior.

**Garaje.sh** 🡪 enciende y apaga la luz del garaje.

**Motor.sh 🡪** Gestiona el giro del motor, se le puede pasar los parámetros de “der”, “izq” o “off”.

**Ventilador.sh** 🡪 En un principio en vez de utilizar un enchufe se pensaba utilizar un ventilador simulando el aire acondicionado de una casa, pero me pareció más interesante utilizar un enchufe. Debido a que el funcionamiento para ambas cosas es el mismo, al script no se le ha cambiado el nombre. Este programa se encarga de encender y apagar el enchufe.

Los programas que se están utilizando en la versión final de la Raspton para la ejecución de domótica son diferentes. Se encuentran en el mismo directorio que los otros, a diferencia de los scripts que hacen uso de los GPIOS, estos scripts hacen uso del programa principal arduino.py. Estos tienen el mismo nombre que los otros pero le anteceden la cadena arduino\_ , por ejemplo el script que gestiona el motor se llama **arduino\_motor.sh.** También se hay unos archivos que empiezan por la cadena **estado\_,** estos archivos registran el estado de encendido y apagado del dispositivo que va detrás del **\_** .

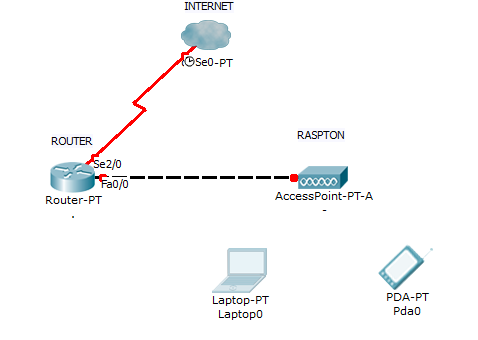
* Para el funcionamiento de la alarma se utiliza otro programa diseñado y adaptado por mí a este caso en concreto. Los archivos de este programa se encuentran en **scripts/alarma**, aquí podemos ver los archivos:
* **alarmadaemon** 🡪 Que es el demonio de la alarma, este archivo se coloca en el init.d y a él se le pueden pasar los parámetros “start”, “stop” y “restart”. Se encarga de ejecutar o cerrar el bucle de la alarma.
* **Sensor\_ir.sh 🡪** Es el bucle principal de la alarma, cuando este se activa se pone en escucha del pin GPIO de entrada del sensor de movimiento y si este se activa, corre el proceso de reproducción del sonido de la alarma.
* **Interruptor 🡪** Es un fichero creado con el propósito de cerrar la ejecución del bucle principal, mediante web. Cuando este está a 0, el bucle se acaba y se para la reproducción y detección. Cada vez que se inicia el bucle principal, este fichero se pone a 1.
* **Start\_alarm.sh 🡪** Encargado de activar la alarma, muy útil para hacerlo mediante web.
* **Stop\_alarm.sh 🡪** Encargado de apagar la alarma y parar la ejecución del sonido, muy útil para hacerlo mediante web.
* **Alarma.mp3 🡪** Este fichero será el que se reproducirá cuando la alarma detecte movimiento.
* Por último el panel de control web, que gestiona la domótica. Se llama **index.php** y está ubicado en **scripts/scripts Web/.** Está claro que ha sido diseñado y adaptado por mí, por lo que hace uso de todos los programas anteriormente mencionado para controlar la domótica mediante una simple página web.

1. INFRAESTRUCTURA DE RED

La infraestructura de red de este proyecto no es muy extensa, quizás en este apartado pueda incluir una utilidad extra que se ha incluido. He configurado una regla IPTABLES para que cuando la Raspberry se conecte por cable a un router, la red WLAN0 se comunique con la ETH0. Así cuando se conecte el cable y nosotros estemos conectados a la red wifi Raspton, tendremos internet. A esto se le llama funcionalidad AP, y es lo que hacen los routers wifi. Para hacer esto posible simplemente he incluido una línea en un script de inicio que creamos para el HOSTAPD que es la siguiente:

sudo iptables -t nat -I POSTROUTING -o eth0 -s 10.0.0.0/24 –j

Se puede obtener más información en detalle en el paso a paso de la instalación del hostapd en : [2-Instalación hostapd.docx](2-Instalación%20hostapd.docx)



1. **PUESTA EN MARCHA DEL PROYECTO: PRUEBAS**
2. Servicios

Las pruebas realizadas sobre el funcionamiento de los servicios que voy a utilizar los podemos encontrar en los ficheros paso a paso que he adjuntado en la raíz del Cd, en esos paso a paso se muestran pantallazos con las pruebas de funcionamiento tanto de los servicios instalados como de los scripts creados.

1. Aplicación web

Para la aplicación web la mejor prueba de su correcto funcionamiento es el video que adjunto en el cd, este video se encuentra en la carpeta videos ubicada en el directorio raíz.

1. Pruebas de red

Podemos Comprobar las pruebas de conexión wifi y enrutamiento desde el video correspondiente en la carpeta videos.

1. **ANEXOS**

En el disco suministrado podéis encontrar todos los paso a paso que se han llevado para la realización del proyecto, además se pueden ver los videos realizados, las fotos, los esquemas eléctricos, además los códigos fuentes completos de todos los programas utilizados.

1. **BIBLIOGRAFÍA**

<http://www.redirc.org/blog/creando-un-ap-en-linux-con-cifrado-wpa-y-dhcp/>

<http://www.alcancelibre.org/staticpages/index.php/09-como-vsftpd>

<http://dmolinap.blogspot.com.es/2009/06/montar-un-servidor-de-archivos-en-10.html>

<http://www.internetdelascosas.cl/2012/12/09/raspberry-pi-instalando-un-lamp-y-phpmyadmin/>

<http://sobrebits.com/montar-un-servidor-casero-con-raspberry-pi-parte-4-configurar-servidor-dns-cache/>

<http://www.raspberrypi-spy.co.uk/2012/09/raspberry-pi-p5-header/raspberry-pi-gpio-layout-revision-2/>

<http://geekytheory.com/arduino-raspberry-pi-raspduino/>

<http://pyserial.sourceforge.net/index.html>

<http://stackoverflow.com/questions/5697047/convert-serial-read-into-a-useable-string-using-arduino>

<http://www.comoprogramar.org/ejemplos-de-python-codigo/>

<http://arduino.cc/es/Reference/DigitalRead>

<http://arduino.cc/es/Tutorial/DigitalPins>

<http://www.fresymetal.com/como-controlar-2-motores-con-raspberry/>

<http://www.arduteka.com/2011/11/tutorial-0002-controlando-entradas-y-salidas-digitales/>

<http://skuarch.blogspot.com.es/2010/02/crear-un-servicio-en-ubuntu.html>