FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA

SISTEMA DE DOMOTICA MEDIANTE RASPBERRY PI E INTERNET

Trabajo presentado por

Omar García Giorgini

Matricula: 9395

Para obtener el Grado de

INGENIERO ELECTRÓNICO

Director de la Tesis: Ing. Juan Carlos Bonadero.

Mar del Plata 12 de Diciembre de 2017

Índice General

1. Resumen

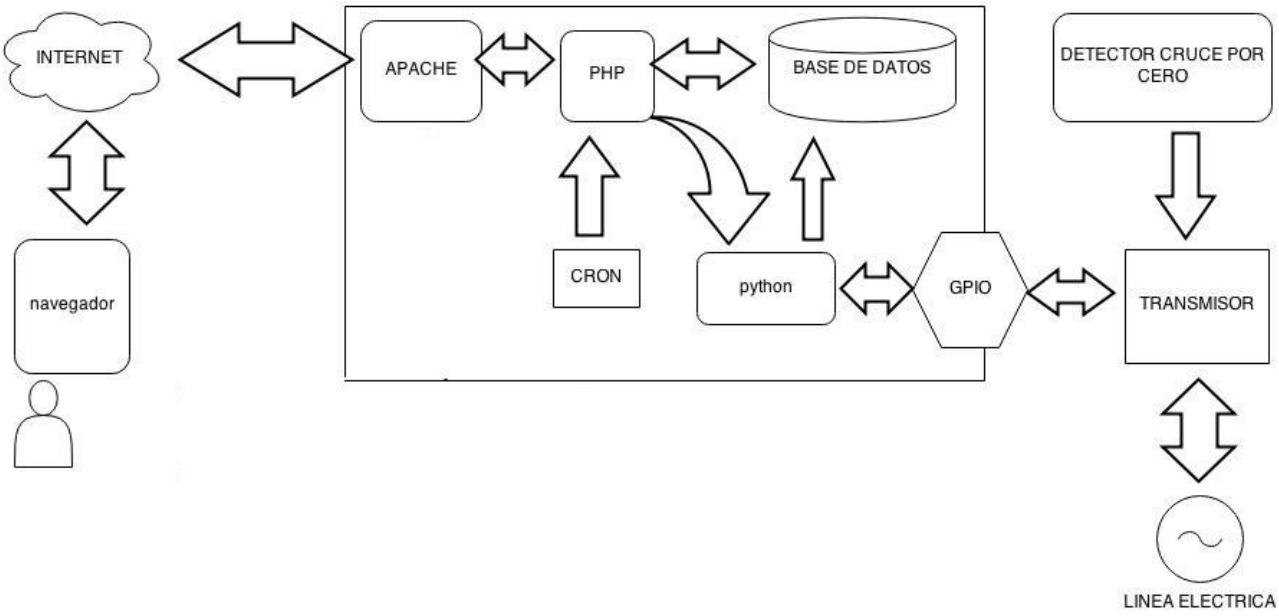
Mediante el uso de una computadora Raspberry Pi, en la cual funciona un servidor de páginas web dinámicas y un motor de bases de datos, se pretende controlar los dispositivos de la casa, empleando para ello el protocolo X-10.
El x10 envía y recibe datos a través de la línea eléctrica hogareña y placas electrónicas especialmente diseñadas para el proyecto .

Una interfaz web se diseñó para enviar los comandos que controlan los dispositivos. En dicha página se implementa un sistema de autenticación, con usuario y contraseña, para evitar el ingreso no autorizado a la interfaz de control. Se usa para ello una base de datos MySQL. El servidor chequea en la base si los datos ingresados son correctos o no.
Una vez recibidos los datos en el servidor, el mismo se comunica con programas escritos en el lenguaje Python, los cuales se encargan de enviar las señales adecuadas a través de los pines/puertos que tiene la Raspberry Pi.
Una placa transceptora, cuyo componente más importante es un microcontrolador, recibe las señales de los puertos GPIO de la Raspberry Pi y las inyecta a la red eléctrica, sincronizando el mensaje con los cruces por cero de la línea.

Asimismo la interfaz permite agendar, mediante el programa cron, una tarea para determinado día y horario, o periódicamente de acuerdo a un calendario especificado.

Con el uso de un servicio de DNS dinámico y una adecuada configuración del router hogareño se logro hacer accesible la pagina web desde cualquier computadora conectada a internet .

El presente trabajo se realizo con el fin de explorar la comunicación PLC (Power Line Comunication) y no con fines comerciales , dado que el protocolo X10 esta superado por otras tecnologías al momento presente.
Al final del informe se consideran posibles mejoras para incrementar la tasa de transferencia y la fiabilidad del sistema , pero algunas de esas mejoras sugeridas se salen del protocolo x10.



2 Introducción

2.1 Que es la domótica.

La domótica es la automatización de una vivienda por medio de diversas técnicas, con el fin de mejorar la calidad de vida de los habitantes del inmueble, proveyendo mayor confort, seguridad y ahorro energético. También contribuye a facilitar el manejo de las tareas cotidianas a las personas mayores o con disminución de las capacidades físicas disminuidas.

Elementos que componen un sistema domótico

Cualquier sistema domótica está compuesto de los siguientes elementos:

- Controladores. Son los que permiten actuar sobre el sistema, bien de una forma automática por decisión tomada por centrales domóticas previamente programadas (que incluso puede ser un PC), pulsadores, teclados, pantallas táctiles o no, mandos a distancia por infrarrojos IR (locales), por radiofrecuencia RF (hasta 50 metros), por teléfono, SMS o por PC (de forma local e incluso a través de Internet). Estos elementos emiten órdenes que necesitan un medio de transmisión

- Medio de transmisión. Según la tecnología aplicada existen distintos medios, fibra óptica, bus dedicado, red eléctrica, línea telefónica, TCP/IP, por el aire.

- Actuadores reciben las órdenes y las transforman en señales de aviso, regulación o conmutación. Los actuadores ejercen acciones sobre los elementos a controlar en el hogar.

- Sensores. Son los "ojos del sistema", o "la adquisición de datos" del sistema, pueden ser todo lo sofisticados que queramos, lo necesario es que lo pueda entender el sistema. Estos datos pueden ser órdenes directas a los Actuadores o pueden ir previamente a una central domótica, en función de la programación en ella introducida saldrá la orden final al Actuador correspondiente. Ejemplos de sensores son los detectores de fuga de agua, de gas, de humo y/o fuego, de concentración de CO, de movimiento o intrusión, los termostatos.

- Elementos externos. Los elementos y/o sistemas instalados en el hogar que son controlados por el sistema domótico.

Entre las ventajas de la domótica se puede mencionar las siguientes:

*Ahorro energético:

Mediante la adecuada gestión de los dispositivos y aparatos controlados por el sistema de domótica, se logra un uso adecuado del consumo.
Se puede llevar un registro del uso y consumo de los electrodomésticos y a partir del análisis de esa información gestionar mejor el uso de ellos en días y horarios

Por ejemplo por medio de la desconexión de equipos de uso no prioritario en función del consumo eléctrico en un momento dado. Y derivando el funcionamiento de algunos aparatos a horas de tarifa reducida.

*seguridad

El sistema brinda seguridad controlando el ingreso de personas a través de las cámaras y las alarmas que se disparan ante una intrusión en alguna puerta o ventana .Ante lo cual se puede programar el cierre y bloqueo de puertas en toda la casa o parte de la misma.

También ofrece seguridad detectando a tiempo la presencia de potenciales siniestros, como incendios, fugas de gas o agua. Lo cual se logra mediante sensores de humo, para activar los aspersores y la alarma en caso de incendio.

Ante viajes o ausencias prolongadas, se puede utilizar la simulación de presencia, mediante la programación de la activación de luces, persiana, sistema de riego, etc.

*confort

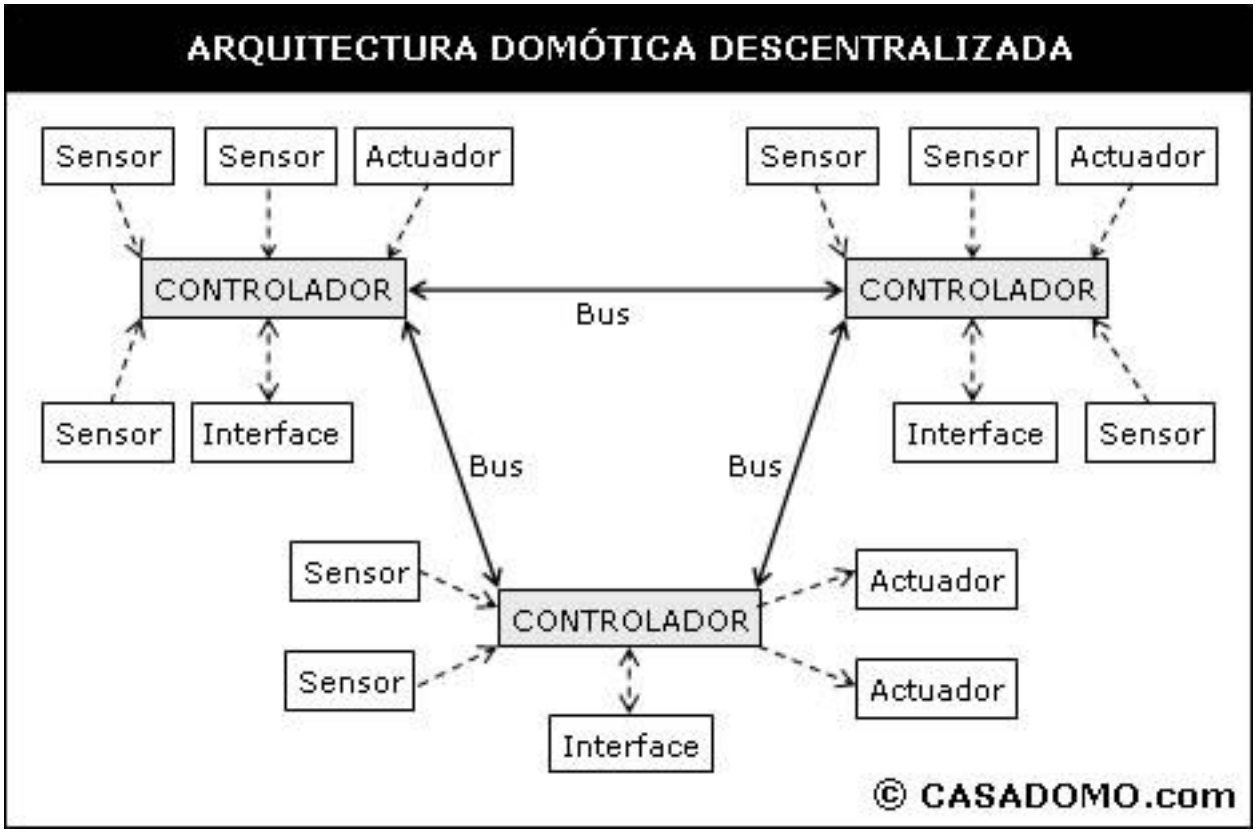
Facilita el manejo de la calefacción, iluminación, persianas, riego, ventilación, aire acondicionado

La comunicación entre las partes del sistemas y la conexión a internet u otra red de datos, tal como la de telefonía celular, hacen posible el monitoreo en vivo de personas mayores con problemas de salud, como hipertensión o anomalías cardiacas. Ante la presencia de cualquier eventualidad se dispara un mensaje de alerta al celular del o los parientes a cargo de la persona de salud delicada.

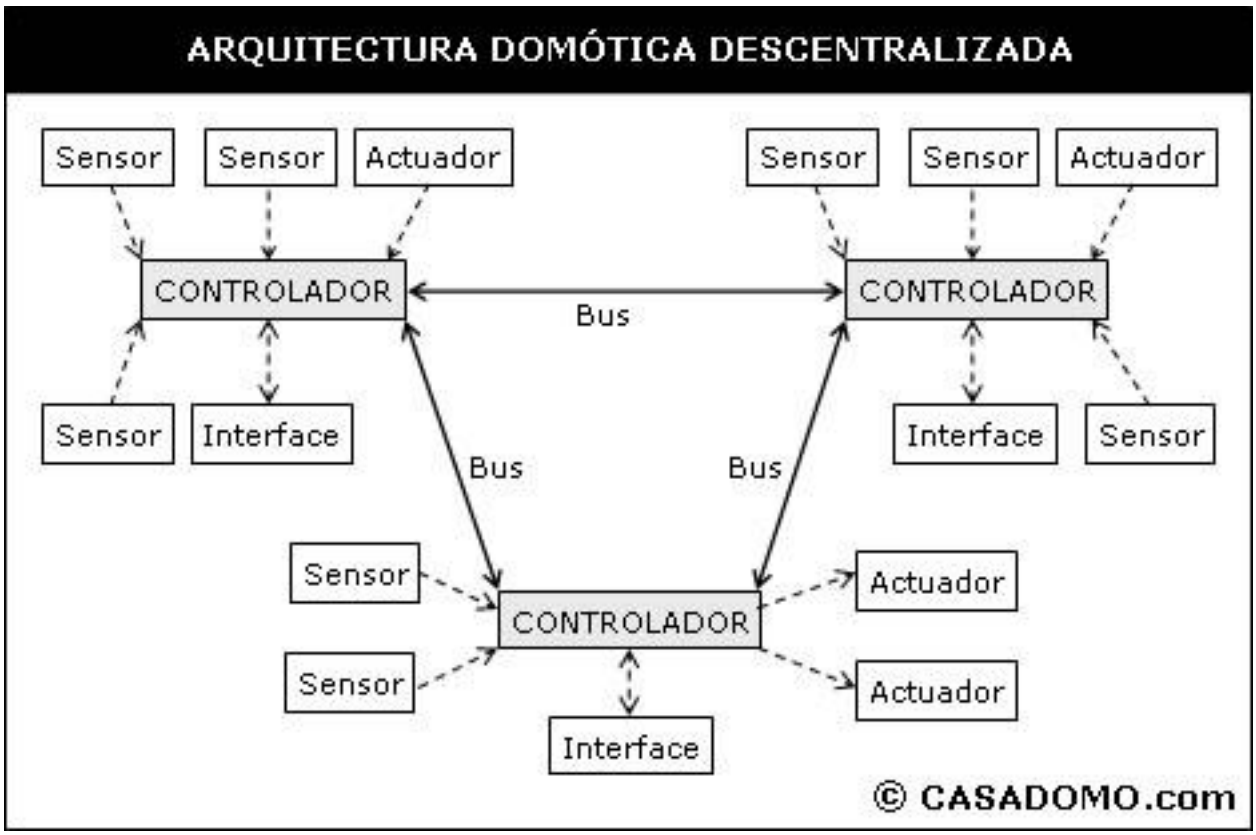
2.2- Revisión de arquitecturas disponibles

Según la distribución de los dispositivos que integran la red domótica y los controladores de estos, se distinguen tres tipos de arquitecturas:

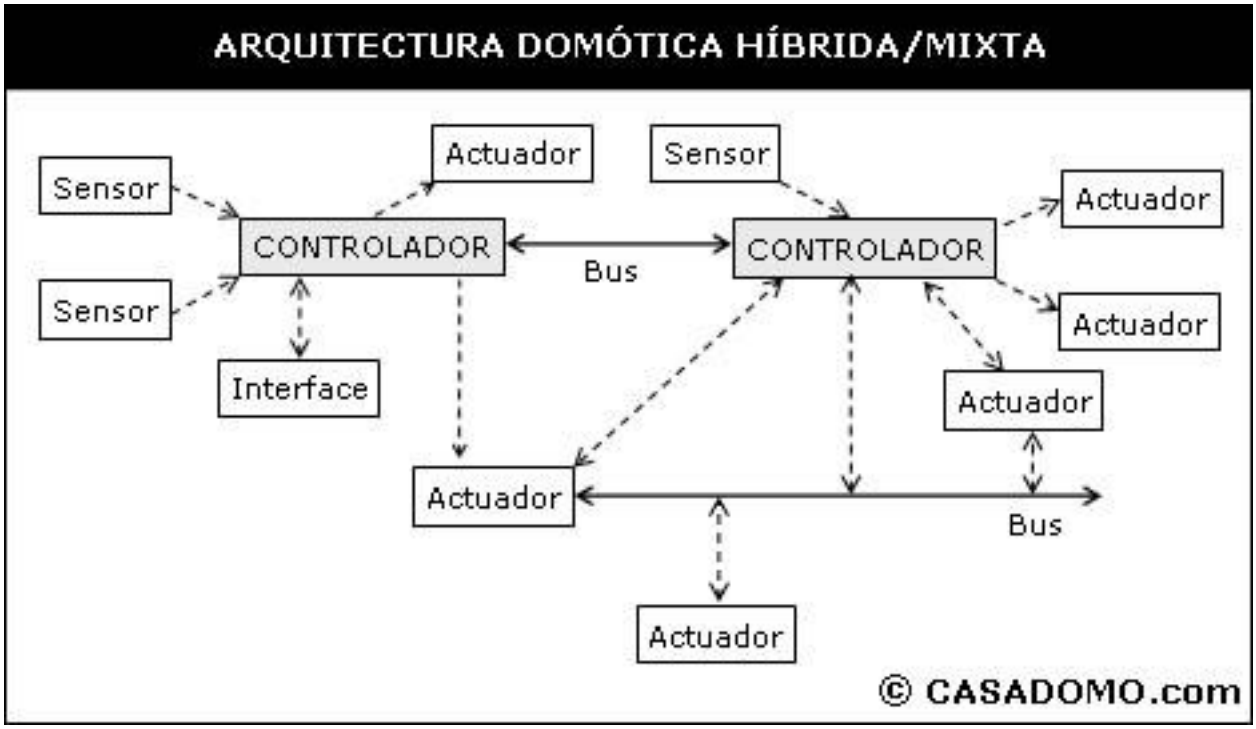
* *Arquitectura centralizada*: Un único dispositivo controlador recibe la información de todos los sensores y procesa los datos para decidir qué acciones tomar. Los sensores y actuadores se comunican solamente con el controlador y no entre sí. El controlador es el cerebro de la vivienda.



**Arquitectura distribuida*: Cada elemento es autónomo, tiene su propio controlador y es capaz de realizar las órdenes oportunas sobre los actuadores del sistema. Las instalaciones con este tipo de arquitectura tienen gran escalabilidad. El riesgo está distribuido, esto es, si un elemento deja de funcionar, el resto del sistema sigue operando.
La desventaja es un incremento en el costo al implementar esta arquitectura.



* *Arquitectura mixta*: Se puede disponer de un controlador central o varios controladores descentralizados. Los sensores y actuadores pueden también ser controladores como en un sistema con arquitectura distribuida. Existen sistemas que utilizan una arquitectura descentralizada pero que poseen módulos o islas de control centralizadas.



2.3-Estándares de mercado

Estándares propietarios o cerrados: Son protocolos específicos de una marca en particular y que solo son usados por dicha marca. Pueden ser variantes de Protocolos Estándares.

Son protocolos cerrados de manera que solo el fabricante puede realizar mejoras y fabricar dispositivos que "hablen" el mismo idioma.

Esto protege los derechos del fabricante, pero limita la aparición de continuas evoluciones en los sistemas domóticos, con lo que, a medida que los sistemas con protocolo estándar se van desarrollando, van ganando cuota de mercado a los sistemas de protocolo propietario.

Otro problema que tienen es la vida útil del sistema domótico, ya que un sistema propietario que depende en gran medida de la vida de la empresa y de la política que siga, si la empresa desaparece, el sistema desaparece y las instalaciones se quedan sin soporte ni recambios.

Estándares abiertos: Son protocolos definidos entre varias compañías con el fin de unificar criterios.

Son abiertos (open systems), es decir, que no existen patentes sobre el protocolo, de manera que cualquier fabricante puede desarrollar aplicaciones y productos que lleven implícito el protocolo de comunicación.

En un sistema estándar, si una empresa desaparece o deja de sacar productos al mercado, no afecta demasiado ya que hay otros productos en el mercado que cubren ese hueco.

Los protocolos estándar para aplicaciones domóticas más extendidos en la actualidad son: KNX, Lonworks y X10.
Desde los inicios de la domótica hubo una carrera constante por parte de los fabricantes y agrupaciones de empresas del sector por establecer estándares de fabricación, en la actualidad solo dos lograron permanecer en el tiempo e imponerse a nivel mundial, los cuales son, el KNX de Konnex Association y el LonWorks de LonMark Association.

A continuación se presenta una breve reseña de los estándares de domótica que fueron apareciendo con el correr del tiempo.

EHS

El estándar EHS (European Home System) fue otro de los intentos que la industria europea (año 1984) procuró, auspiciada por la Comisión Europea, al crear una tecnología que permitiera la implantación de la domótica en el mercado residencial de forma masiva. El resultado fue la especificación del EHS en el año 1992. Estuvo basada en una topología de niveles OSI (Open Standard Interconnection), y se especificaron los niveles: físico, de enlace de datos, de red y de aplicación.
Desde su inicio estuvieron involucrados los fabricantes europeos más importantes de electrodomésticos de línea marrón y blanca, las empresas eléctricas, las operadoras de telecomunicaciones y los fabricantes de equipamiento eléctrico. La idea… crear un protocolo abierto que permitiera cubrir las necesidades de interconexión de los productos de todos estos fabricantes y proveedores de servicios.

Tal y como fue pensado, el objetivo de la EHS fue cubrir las necesidades de automatización de la mayoría de las viviendas europeas cuyos propietarios no podían permitirse el lujo de usar sistemas más potentes pero también más caros como LonWorks, EIB o BatiBUS, debido fundamentalmente a la mano de obra especializada que exigía su instalación.
El EHS viene a cubrir, por prestaciones y objetivos, la parcela que tienen el CEBus norteamericano y el HBS japonés y rebasa las prestaciones del X-10 que tanta difusión ha conseguido en EEUU.

EHSA

La EHS Association (EHSA) fue la encargada de emprender y llevar a cabo diversas iniciativas para aumentar el uso de esta tecnología en las viviendas europeas. Además se ocupó de la evolución y mejora tecnológica del EHS y de asegurar la compatibilidad total entre fabricantes de productos con interface EHS.

BatiBUS

Este protocolo de domótica está totalmente abierto, esto es, al contrario de los que sucede con el protocolo LonTalk de la tecnología LonWorks, el protocolo del BatiBUS lo puede implementar cualquier empresa interesada en introducirlo en su cartera de productos.
A nivel de acceso, este protocolo usa la técnica CSMA-CA, (Carrier Sense Multiple Access with Collision Avoidance) similar a Ethernet pero con resolución positiva de las colisiones. Esto es, si dos dispositivos intentan acceder al mismo tiempo al bus ambos detectan que se está produciendo una colisión, pero sólo el que tiene más prioridad continua transmitiendo, el otro deja de poner señal en el bus. Esta técnica es muy similar a la usada en el bus europeo EIB y también en el bus del sector del automóvil llamado CAN (Controller Area Network).
La filosofía es que todos los dispositivos BatiBUS escuchen lo que ha enviado cualquier otro, todos procesan la información recibida, pero sólo aquellos que hayan sido programados para ello, filtrarán la trama y la subirán a la aplicación empotrada en cada dispositivo.
Al igual que los dispositivos X-10, todos los dispositivos BatiBUS disponen de unos micro interruptores circulares o mini teclados que permiten asignar una dirección física y lógica que identifican unívocamente a cada dispositivo conectado al bus.

EIB

El European Installation Bus o EIB es un sistema domótico desarrollado bajo los auspicios de la Unión Europea con el objetivo de contrarrestar las importaciones de productos similares que se estaban produciendo desde el mercado japonés y el norteamericano, países donde estas tecnologías se encontraban desarrolladas.
El objetivo era crear un estándar europeo, con el suficiente número de fabricantes, instaladores y usuarios, que permita comunicar a todos los dispositivos de una instalación eléctrica como: contadores, equipos de climatización, de custodia y seguridad, de gestión energética y electrodomésticos.
El EIB está basado en la estructura de niveles OSI y tiene una arquitectura descentralizada. Este estándar europeo define una relación extremo a extremo entre dispositivos que permite distribuir la inteligencia entre los sensores y los actuadores instalados en la vivienda.

EIBA

La EIB Association (EIBA) es una agrupación de 113 empresas europeas, líderes en el mercado

eléctrico, que se unieron en 1990 para impulsar el uso e implantación del sistema domótico EIB. La EIBA tiene su sede en Bruselas y todos sus miembros cubren el 80 % de la demanda de equipamiento eléctrico en Europa. Las tareas principales de esta asociación son:

• Fijar las directrices técnicas para el sistema y los productos EIB, así como establecer los procedimientos de ensayo y certificación de calidad.
• Distribuye el conocimiento y las experiencias de las empresas que trabajan sobre el EIB.
• Encarga a laboratorios de ensayo las pertinentes pruebas de calidad.
• Concede a los productos EIB y a los fabricantes de estos una licencia de marca EIB con la que se pueden distribuir los productos.
• Colabora activamente con otros organismos europeos o internacionales en todas las fases de la normalización y adapta el sistema EIB a las normas vigentes.
• Lidera el proceso de convergencia (ver Konnex Association en este mismo apartado) de los tres buses europeos de más amplia difusión como son el propio EIB, el BatiBUS y el EHS.

Según la EIBA hay unos 10 millones de dispositivos EIB instalados por todo el mundo, unas 70.000 instalaciones, una gama de 4.500 productos diferentes, 113 empresas asociadas a la EIBA, y 70.000 instaladores cualificados.

KNX

Konnex es la iniciativa de tres asociaciones europeas unidas para crear un único estándar europeo para la automatización de las viviendas y oficinas:

• EIBA, (European Installation Bus Association).
• BatiBUS Club International.
• EHSA (European Home System Association).

Los objetivos de esta iniciativa, con el nombre de "Convergencia", son:

• Crear un único estándar para la domótica e inmótica que cubra todas las necesidades y requisitos de las instalaciones profesionales y residenciales del ámbito europeo.
• Aumentar la presencia de estos bus domóticos en áreas como la climatización o HVAC (heating, ventilation, and air conditioning).
• Mejorar las prestaciones de los diversos medios físicos de comunicación sobre todo en la tecnología de radiofrecuencia.
• Introducir nuevos modos de funcionamiento que permitan aplicar una filosofía Plug&Play a muchos de los dispositivos típicos de una vivienda.
• Contactar con empresas proveedoras de servicios como las empresas de telecomunicaciones y las empresas eléctricas, con el objeto de potenciar las instalaciones de tele gestión técnica de las viviendas.

En resumen, se trata de, partiendo de los sistemas EIB, EHS y BatiBUS, crear un único estándar europeo que sea capaz de competir en calidad, prestaciones y precios con otros sistemas norteamericanos como el LonWorks o CEBus
Actualmente la asociación Konnex concluyó las especificaciones del nuevo estándar el cual es compatible con los productos EIB instalados. Se puede afirmar que el nuevo estándar tiene lo mejor del EIB, del EHS y del BatiBUS, lo que aumenta considerablemente la oferta de productos para el mercado residencial el cual ha sido, hasta la fecha, la asignatura pendiente de este tipo de tecnologías.

La versión 1.0 contempla tres modos de funcionamiento:

1. S.mode (System mode): en esta configuración el sistema usa la misma filosofía que el EIB actual, esto es, los diversos dispositivos o nodos de la red son instalados y configurados por profesionales con ayuda del software de aplicación especialmente diseñada para este propósito.
2. E.mode (Easy mode): en esta configuración sencilla los dispositivos son programados en fábrica para realizar una función concreta. Aun así deben ser configurados algunos detalles en la instalación, ya sea con el uso de un controlador central (como una pasarela residencial o similar) o mediante unos microinterruptores alojados en el mismo dispositivo (similar a muchos dispositivos X-10 que hay en el mercado).
3. A.mode (Automatic mode): en esta configuración automática, con una filosofía Plug&Play, ni el instalador, ni el usuario final tienen que configurar el dispositivo. Este modo está especialmente indicado para ser usado en electrodomésticos, equipos de entretenimiento (consolas, set-top boxes, HiFi, etc.) y proveedores de servicios.

De forma coloquial ¿qué nos aportan estos tres modos?

• S.mode: está especialmente pensado para su uso en instalaciones como oficinas, industrias, hoteles, etc. Sólo los instaladores profesionales tendrán acceso a este tipo de material y a las herramientas de desarrollo. Los dispositivos S.mode sólo podrán ser comprados a través de distribuidores eléctricos especializados.

• E.mode: cualquier electricista sin formación en manejo de herramientas informáticas o cualquier usuario final autodidacta, podrán conseguir dispositivos E.mode en ferreterías o almacenes de productos eléctricos. Aunque la funcionalidad de estos productos está limitada (viene establecida de fábrica), la ventaja de este modo es que se configuran en un instante seleccionando, en unos micro interruptores, las opciones ofrecidas con una pequeña guía de usuario. Para los que conozcan el popular X-10 de amplio uso en EE.UU, los dispositivos E.mode aplican la misma filosofía.

• A.mode: es el objetivo al que tienden muchos productos informáticos y de uso cotidiano. Con la filosofía Plug&Play, el usuario final no tiene que preocuparse de leer complicados manuales de instalación o perderse en un mar de referencias o especificaciones. Tan pronto como conecte un dispositivo A.mode a la red este se registrará en las bases de datos de todos los dispositivos activos en ese momento en la instalación o vivienda y pondrá a disposición de los demás sus recursos (procesador, memoria, entradas/salidas, etc.). Es la misma filosofía que la iniciativa de Sun Microsystems con el Jini o de Microsoft con el Universal Plug&Play. Son los fabricantes de electrodomésticos y de pasarelas residenciales, así como los proveedores de servicios (empresas de telecomunicaciones, eléctricas, ISPs), los más interesados en este tipo de productos ya que permitirán ofrecer nuevos servicios a sus clientes de forma rápida y sin necesidad de complicadas instalaciones.

Respecto al nivel físico el nuevo estándar funcionar sobre:

• Par trenzado (TP1): aprovechando la norma EIB equivalente.
• Par trenzado (TP0): aprovechando la norma BatiBUS equivalente (actualmente en desuso).
• Ondas Portadoras (PL100): aprovechando la norma EIB equivalente.
• Ondas Portadoras (PL132): aprovechando la norma EHS equivalente (actualmente en desuso).
• Ethernet: aprovechando la norma EIB.net.
• Radiofrecuencia: aprovechando la norma EIB.RF

X10

X10 es un protocolo de comunicaciones para el control remoto de dispositivos eléctricos, que utiliza la línea eléctrica (220V o 110V) preexistente, para transmitir señales de control entre equipos de automatización del hogar (domótica) en formato digital. Los dispositivos X10 que se comercializan son solo para uso individual y en entornos domésticos de hasta 250 m2, dada su limitación en ancho de banda y en el número máximo de dispositivos a controlar (256). No obstante existen elementos de última generación que incorporan, entre otros, los protocolos X-10 extendidos, para dar funcionalidad a soluciones de comunicación como la bidireccionalidad, solicitud de estados y comprobación de la correcta transmisión de las tramas.

X10 fue desarrollada en 1978 por Pico Electronics of Glenrothes, Escocia, para permitir el control remoto de los dispositivos domésticos. Fue la primera tecnología domótica en aparecer y sigue siendo la más ampliamente disponible, principalmente por su característica de autoinstalable, sin necesidad de cableado adicional.

Las señales de control de X10 se basan en la transmisión de ráfagas de pulsos de RF (120 kHz) que representan información digital. Estos pulsos se sincronizan en el cruce por cero de la señal de red (50 Hz ó 60 Hz). Con la presencia de un pulso en un semi-ciclo y la ausencia del mismo en el semi-ciclo siguiente se representa un '1' lógico y a la inversa se representa un '0'. A su vez, cada orden se transmite 2 veces, con lo cual toda la información transmitida tiene cuádruple redundancia. Cada orden involucra 11 ciclos de red (220 ms para 50 Hz y 183,33, para 60Hz).

Primero se transmite una orden con el Código de Casa y el Número de Módulo que direccionan el módulo en cuestión. Luego se transmite otro orden con el código de función a realizar (Function Code). Hay 256 direcciones soportadas por el protocolo.

Se han propuesto distintas alternativas con más banda, incluyendo protocolos como European Home Systems, Lonworks, XD2, CEBus, aunque sigue siendo el más extendido.

Protocolo y descripción del sistema

El protocolo X10 consta de bits de «direcciones» y de «órdenes». Por ejemplo, permite decir «lámpara #3», «¡enciéndete!» y el sistema procederá a ejecutar dicho mandato. Se puede direccionar varias unidades antes de dar la orden: «lámpara #3, lámpara #12», «¡enciendan!». Como puede verse en la lista de más abajo, existen múltiples instrucciones utilizadas por el protocolo entre las cuales destacamos: ON, OFF, All Lights ON, All off, DIM, BRIGHT.

Los dispositivos están generalmente enchufados en módulos X10 (receptores). X10 distingue entre módulos de lámparas y módulos de dispositivos. Los módulos de lámpara proporcionan energía y aceptan órdenes X-10. Los módulos de dispositivos son capaces de gestionar cargas grandes (ejemplo: máquinas de café, calentadores, motores, etc), simplemente encendiéndolos y apagándolos.

Si desea controlar luces vía mandatos X-10, debería conectar la luz en un módulo de luz en la red y, a continuación, asignarle una dirección (A1, por ejemplo). Así, cuando envíe la orden «A1 encendido» a través de los cables de la red eléctrica, la luz se debería encender. Cabe destacar que los módulos de lámparas no pueden soportar grandes cargas y que todo el sistema es muy sensible a los ruidos eléctricos por lo que es considerado como un sistema para el "hazlo tú mismo".

Actualmente X10 es un protocolo que está presente en el mercado mundial, sobre todo en Norteamérica y Europa (España, Holanda, Portugal y Gran Bretaña fundamentalmente).

Los nuevos protocolos de comunicación en la red eléctrica ocupan una señal más fuerte e inmune al ruido eléctrico, uno de estos protocolos es el llamado UPB (Universal Powerline Bus)

Lista de comandos X10

Código	Función	Descripción	Unidireccional	Bidireccional
0 0 0 0	All units off	Apaga todos los dispositivos con el código de casa indicado en el mensaje	X	
0 0 0 1	All lights on	Enciende todas las luces (con la posibilidad de controlar el brillo)	X	
0 1 1 0	All lights off	Apaga todas las luces	X	
0 0 1 0	On	Enciende un aparato	X	
0 0 1 1	Off	Apaga un aparato	X	
0 1 0 0	Dim	Atenúa la intensidad de la luz	X	
0 1 0 1	Bright	Incrementa la intensidad de la luz	X	

0 1 1 1	Extended code	Código de extensión		X
1 0 0 0	Hail request (solicitud de saludo)	Solicita una respuesta del dispositivo(s) con el código de casa indicado en el mensaje		X
1 0 0 1	Hail acknowledge (confirmación de saludo)	Respuesta al comando anterior		X
1 0 1 0	Pre-set dim	Permite la selección de dos niveles predefinidos de intensidad de luz		X
1 1 0 1	Status is on	Respuesta a la Solicitud de Estado indicando que el dispositivo está encendido		X
1 1 1 0	Status is off	Respuesta indicando que el dispositivo está apagado		X
1 1 1 1	Status request	Solicitud pidiendo el estado de un dispositivo		X

2.4 Porque X-10

Se eligió el protocolo x10, por ser este un estándar no propietario, fácil de implementar, que no requiere instalar cableado nuevo, sino que usa el ya instalado, ahorrando así considerable dinero. Existe una gran cantidad de módulos en el mercado, lo que no requieren para su uso más que ser enchufados a la red, y que se les asigne un código de casa.

2.5 Porque una Raspberry Pi

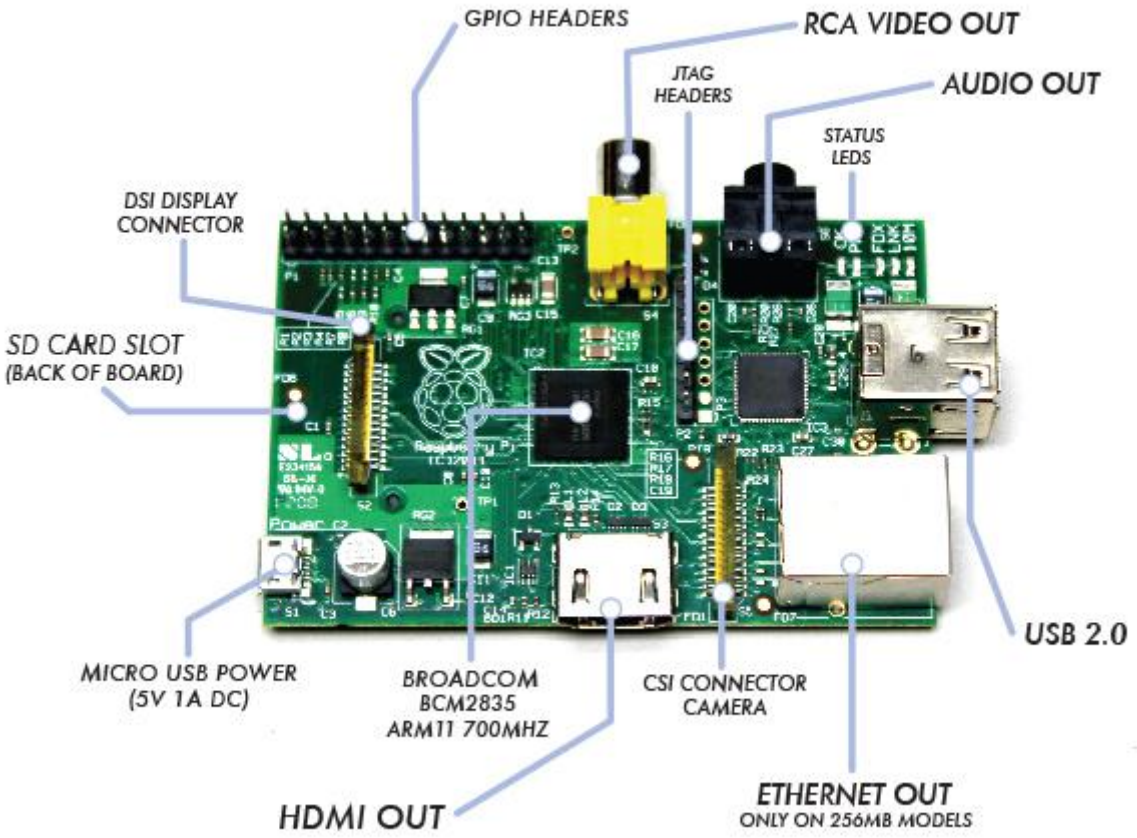
La raspberry pi es una computadora de tamaño apenas mayor a una tarjeta de crédito, (85.60mm × 53.98mm) con un procesador ARM de 700 Mhz y 512 M de RAM , conexión ethernet , 2 puertos USB ,1 puerto HDMI , puerto RCA ,8 pines GPIO ,SPI ,I2C y UART , con consumo de 700 mA y 3,5 W.
El diseño no incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación ni carcasa.

Se alimenta con una fuente de 5 volt y 1 ampere con puerto mini USB. Puede correr varios sistemas operativos, la mayoría de ellos basados en núcleos Linux, entre ellos:
GNU/Linux:Debian(Raspbian),Fedora(Pidora),Arch Linux(Arch Linux ARM),Slackware Linux.
RISC OS2

El Raspberry Pi no viene con reloj en tiempo real, por lo que el sistema operativo debe usar un servidor de hora en red, o pedir al usuario la hora en el momento de arrancar el ordenador. Sin embargo se podría añadir un reloj en tiempo real (como el DS1307) con una batería mediante el uso de la interface I²C.

Se la eligió para realizar el proyecto dado que es más potente que un micro controlador, permitiendo correr un servidor web, bases de datos y un servidor de vídeo dentro de un sistema operativo robusto como los GNU/Linux , siendo fácilmente configurable y accesible , mediante escritorio remoto o protocolo SSH.

Es equivalente a una computadora normal de uso doméstico, pero con la ventaja de disponer de un buen número de GPIO, pines configurables de entrada/salida, a través de los cuales se interactúa con el mundo físico.



Detalle de los conectores y los principales circuitos integrados en el PCB

3.3V	1	2	5V
I2C0 SDA	3	4	DNC
I2C0 SCL	5	6	GROUND
GPIO4	7	8	UART TXD
DNC	9	10	UART RXD
GPIO 17	11	12	GPIO 18
GPIO 21	13	14	DNC
GPIO 22	15	16	GPIO 23
DNC	17	18	GPIO 24
SP10 MOSI	19	20	DNC
SP10 MISO	21	22	GPIO 25
SP10 SCLK	23	24	SP10 CE0 N
DNC	25	26	SP10 CE1 N

Detalle de los pines de la Raspberry Pi y sus funciones.

3-Implementación de la interfaz con el usuario

3.1Configuracion de la Raspberry Pi

Para empezar a usar la Raspberry Pi hay primero que elegir un sistema operativo e instalarlo en la tarjeta de memoria SD.

Para ello se precisó de un cable HDMI, para conectar a la pantalla del televisor, un teclado y un mouse USB, además de una fuente de 5 volts y la tarjeta SD, en este caso de 8 Gb.

Hay varios SO disponibles, cuyas imágenes se pueden descargar desde:

http://www.raspberrypi.org/downloads/

Hemos elegido NOOBS (New Out Of the Box Software), por ser la forma recomendada como mas facil para los principiantes o gente que no gusta de las complicaciones innecesarias.

NOOBS hace innecesario el acceso a Internet durante la instalación, y tan solo tendremos que descargar NOOBS y descomprimirlo en una tarjeta SD de al menos 4 GB de capacidad. Al hacerlo se nos dará la opción de instalar soluciones como Raspbian, Arch Linux, RaspBMC, el recién salido Pidora u OpenELEC sin problemas.
La instalación permitirá más tarde iniciar nuestro Raspberry Pi de forma normal con esa nueva distribución, pero NOOBS quedará residente en memoria y podremos acceder a esta aplicación siempre que queramos mediante la pulsación de la tecla Shift durante el proceso de arranque. Eso permitirá cambiar a otro sistema operativo si lo necesitamos, pero también da acceso a la edición rápida del fichero *config.txt* para la distribución instalada en la que se establecen parámetros que afectan por ejemplo a la resolución de pantalla y el overscan aplicado para ajustar los bordes de la

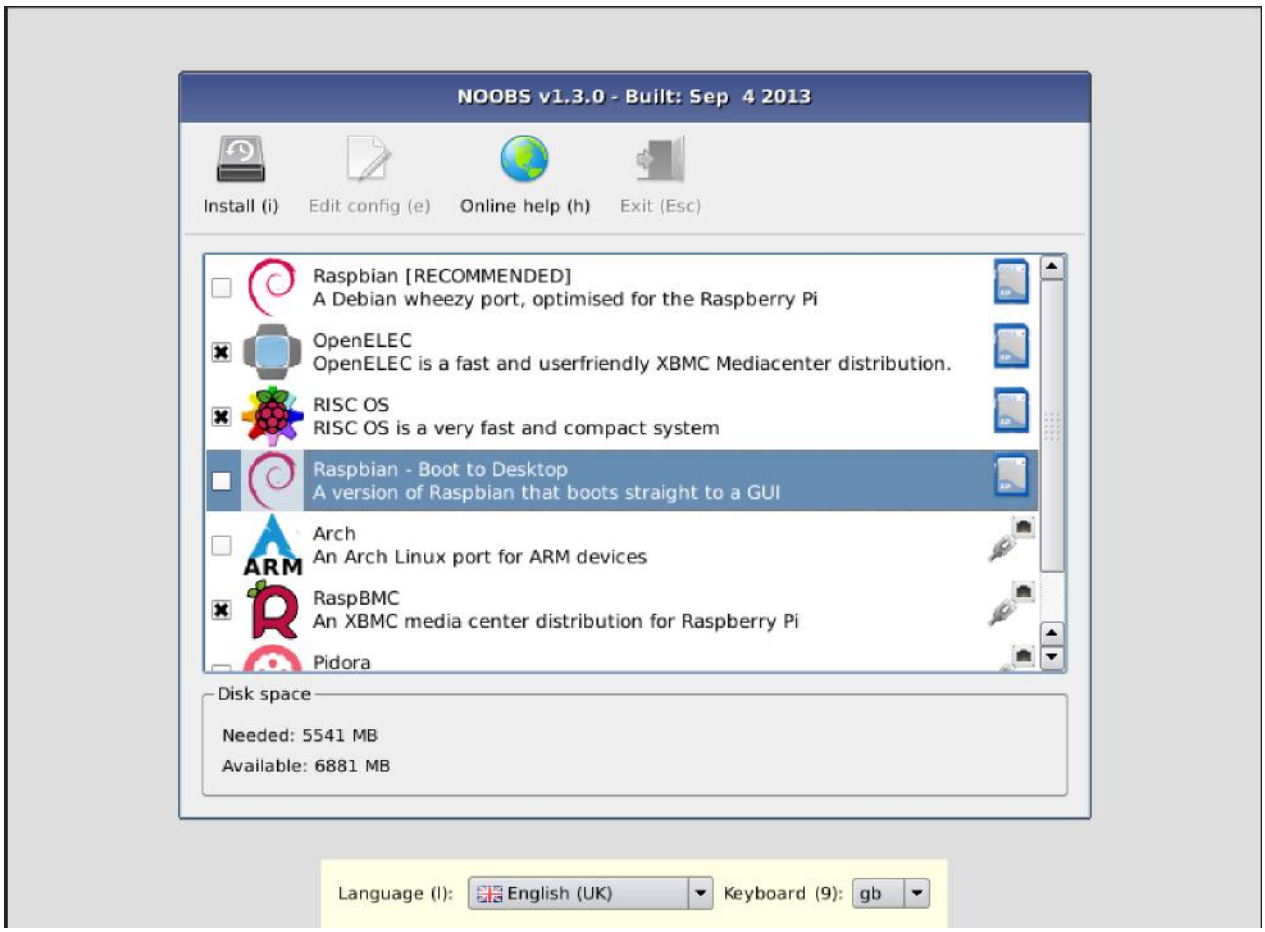
interfaz a los de la pantalla o televisor. Incluso podremos tener acceso a un navegador de emergencia (Arora, un pequeño desarrollo basado en QtWebKit) para consultar información durante esa edición del fichero —o de cualquier otro apartado— en ese menú de recuperación.

Una vez obtenida la imagen, es necesario formatear la tarjeta SD, para hacerlo en windows se recomienda usar la herramienta de formateo disponible en https://www.sdcard.org/downloads/formatter_4/

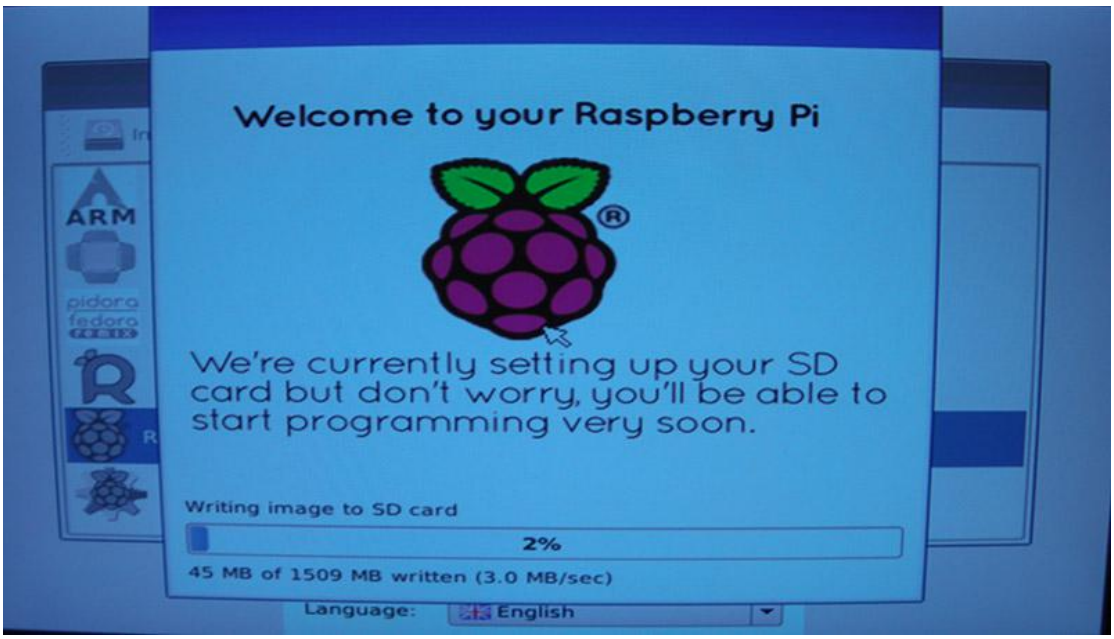
Para GNU/Linux se recomienda gparted.

Luego de formatear, se descomprime el archivo de NOOBS descargado y se copian los archivos en las tarjeta SD, de manera tal de hacer de ella el directorio raíz.

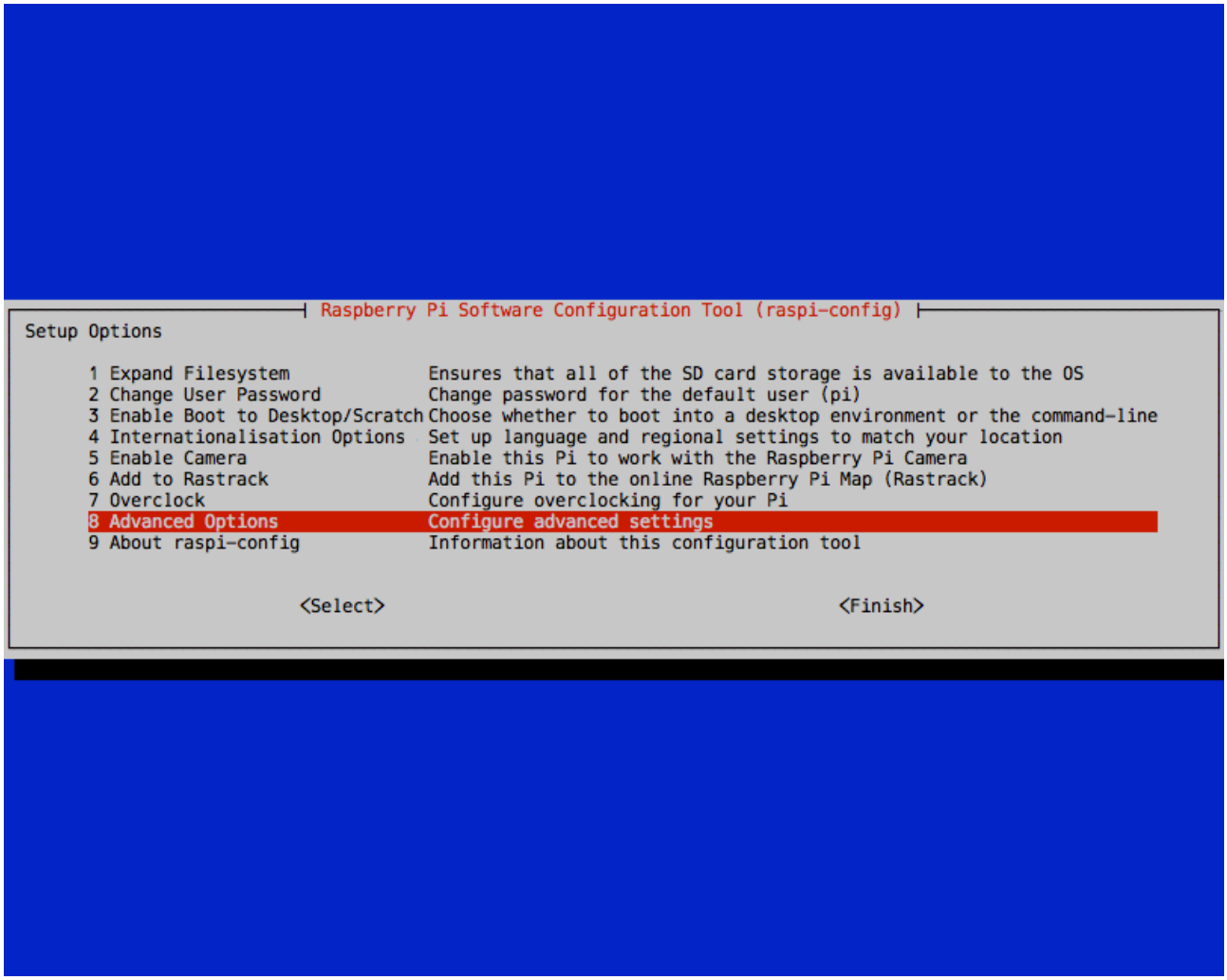
Una vez hecho esto, se inserta la tarjeta en la ranura correspondiente de la raspberry, se enchufa el cable HDMI al televisor encendido, habiendo verificado que la entrada de vídeo seleccionada sea la correcta; y se prende el sistema conectando la fuente de 5v. Aparece la siguiente pantalla:



Se pueden seleccionar más de un operativo a instalar, en nuestro caso se instaló solo uno: Raspbian.



Al iniciarse por primera vez Raspbian se ve el programa de configuración raspi-config:

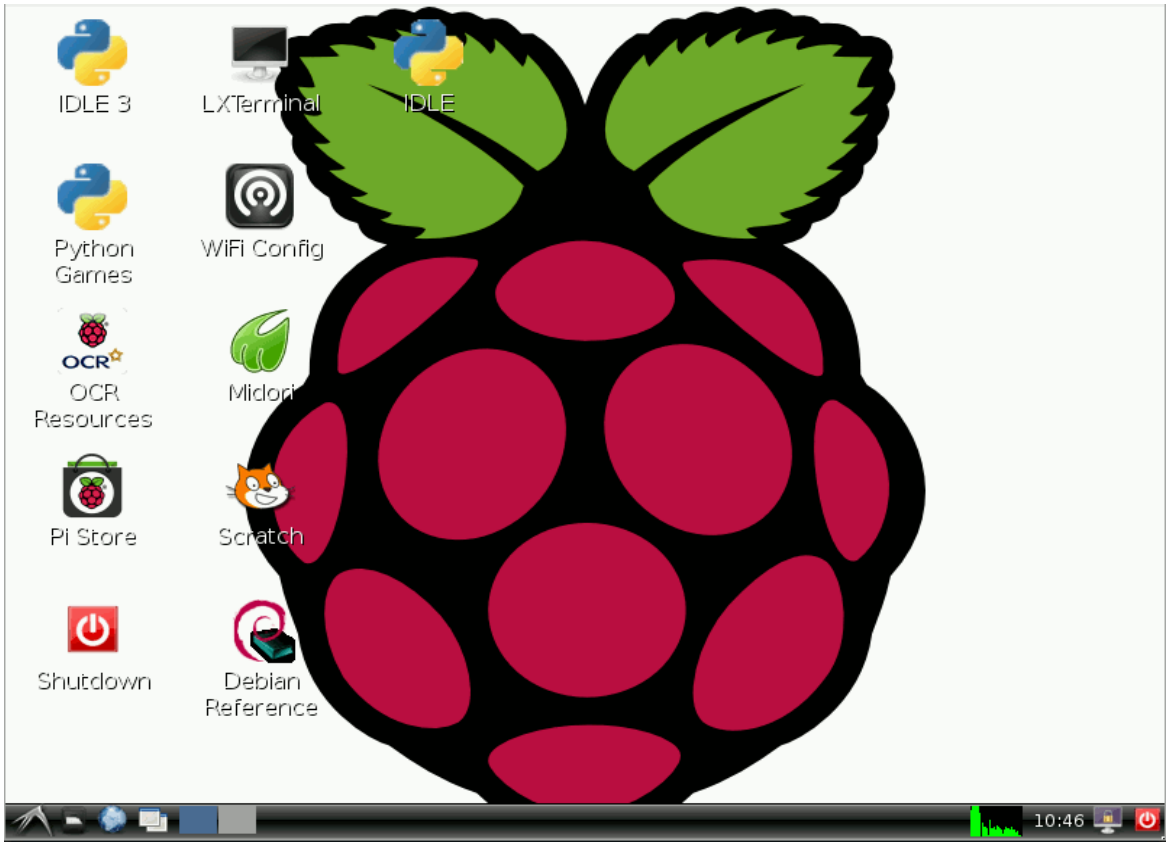


El usuario por defecto se llama pi y su contraseña es raspbian, la cual se puede cambiar por motivos de seguridad, desde la opción 2 del menú.
También se configura el lenguaje y la zona en la que uno se halla.
En configuraciones avanzadas se establece el acceso vía SSH.
Se puede establecer si al bootear , se carga el entorno gráfico o la línea de comandos, si se habilita el overclocking , etc.

Luego de configurar lo necesario se presiona finish y se entonces aparece un prompt , para cargar el entorno gráfico se escribe startx

pi@raspberrypi:~$ startx

y se presiona enter , aparece entonces esta pantalla :



Finalmente instalado el sistema, falta actualizarlo, en una terminal se escribe:

sudo apt-get dist-upgrade

sudo rpi-update (para tener el ultimo firmware)

sudo apt-get update (actualiza la lista de repositorios , desde los cuales se descargan los paquetes y dependencias de los programas instalables)

sudo apt-get upgrade (actualiza los paquetes y dependencias de los programas instalados)

e instalar un escritorio remoto

sudo apt-get install xrdp

El sistema ya está totalmente listo para ser utilizado.

3.2 Interfaz web

La interfaz web , permite al usuario ver las cámaras de la casa que monitorizan posibles actividades sospechosas , así como también enviar comandos a los dispositivos caseros , así como programar su encendido y apagado en horarios y días determinados.
Se puede programar el riego del jardín en vacaciones o hacer simulación de presencia.
Desde la página se solicita un nombre de usuario y contraseña para acceder a la pantalla de configuración. Evitando de tal forma la intromisión en el control de cualquier individuo que conozca la URL del sitio web.
En la pantalla de configuración se puede ver el estado actual de los dispositivos, tal información se encuentra en una base de datos que se actualiza dinámicamente al ser enviados los comandos por el usuario.

Para realizar el sitio web se eligió usar PHP, APACHE Y MySQL

3.2.1 PHP

Es un lenguaje de programación multiplataforma e interpretado, de código libre y abierto.
Pensado inicialmente para crear páginas web con contenido dinámico, luego se extendió para permitir crear aplicaciones de escritorio.
Corre del lado del servidor, lo que lo hace más seguro, pues su código no puede ser modificado del lado del cliente .Permite ser incorporado directamente en el documento HTML, en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página Web resultante.

Para instalar php en una consola ejecutamos

sudo apt-get install php5

3.2.2-Servidor web Apache, instalación y configuración

Es un servidor web HTTP de código abierto, para plataformas Unix (BSD, GNU/Linux, etc.), Microsoft Windows, Macintosh y otras.

Presenta entre otras características altamente configurables, bases de datos de autenticación y negociado de contenido.
Apache tiene amplia aceptación en la red: desde 1996, Apache, es el servidor HTTP más usado. Alcanzó su máxima cuota de mercado en 2005 siendo el servidor empleado en el 70% de los sitios web en el mundo, sin embargo ha sufrido un descenso en su cuota de mercado en los últimos años. (Estadísticas históricas y de uso diario proporcionadas por Netcraft3).
La mayor parte de la configuración se realiza en el fichero apache2.conf o httpd.conf, según el sistema donde esté corriendo. Cualquier cambio en este archivo requiere reiniciar el servidor, o forzar la lectura de los archivos de configuración nuevamente.

Instalación

Para tener el servidor andando en la raspberry pi se lo instala con los siguientes comandos, siempre desde una terminal, e introduciendo la contraseña que será pedida al presionar enter

sudo apt-get install apache2 php5 php5-mysql mysql-server

De ahora en más el servidor se puede parar y reanudar, mediante:

sudo service apache2 restart/start/stop

o sino de esta otra manera, igualmente valida .

sudo /etc/init.d/apache2 restart/start/stop

Ahora introduce la dirección IP de tu Raspberry Pi en tu navegador web y verás una simple página que dice "It Works!"

Las páginas creadas deben colocarse en el directorio

/var/www/

Finalmente para ser vistas desde el navegador es necesario cambiar permisos en el directorio de las páginas web

chmod 775 -R /var/www/abba/

abba/ es el directorio donde pongo todos los archivos pertenecientes a este sitio web.

3.2.3-MySQL

MySQL es un sistema de administración de bases de datos (*Database Management System, DBMS*) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL fue escrito en C y C++ y destaca por su gran adaptación a diferentes entornos de desarrollo, permitiéndole interactuar con los lenguajes de programación más utilizados como PHP, Perl y Java y su integración en distintos sistemas operativos.

También es muy destacable, la condición de open source de MySQL, que hace que su utilización sea gratuita e incluso se pueda modificar con total libertad, pudiendo descargar su código fuente. Esto ha favorecido muy positivamente en su desarrollo y continuas actualizaciones, para hacer de MySQL una de las herramientas más utilizadas por los programadores orientados a Internet.

MySQL es un sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. MySQLAB desde enero de 2008 una subsidiaria de Sun Microsystems, y ésta a su vez de Oracle Corporation desde abril de 2009— desarrolla MySQL como software libreen un esquema de licenciamiento dual.

Por un lado se ofrece bajo la GNU GPL para cualquier uso compatible con esta licencia, pero para aquellas empresas que quieran incorporarlo en productos privativos deben comprar a la empresa una licencia específica que les permita este uso. Está desarrollado en su mayor parte en ANSI C.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de licenciamiento anteriormente mencionado. Además de la venta de licencias privativas, la compañía ofrece soporte y servicios. Para sus operaciones contratan trabajadores alrededor del mundo que colaboran vía Internet. MySQLAB fue fundado por David Axmark,Allan Larssony y Michael Widenius.

MySQL es usado por muchos sitios web grandes y populares, como Wikipedia, Google(aunque no para búsquedas), Facebook, Twitter, Flickr, y YouTube

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

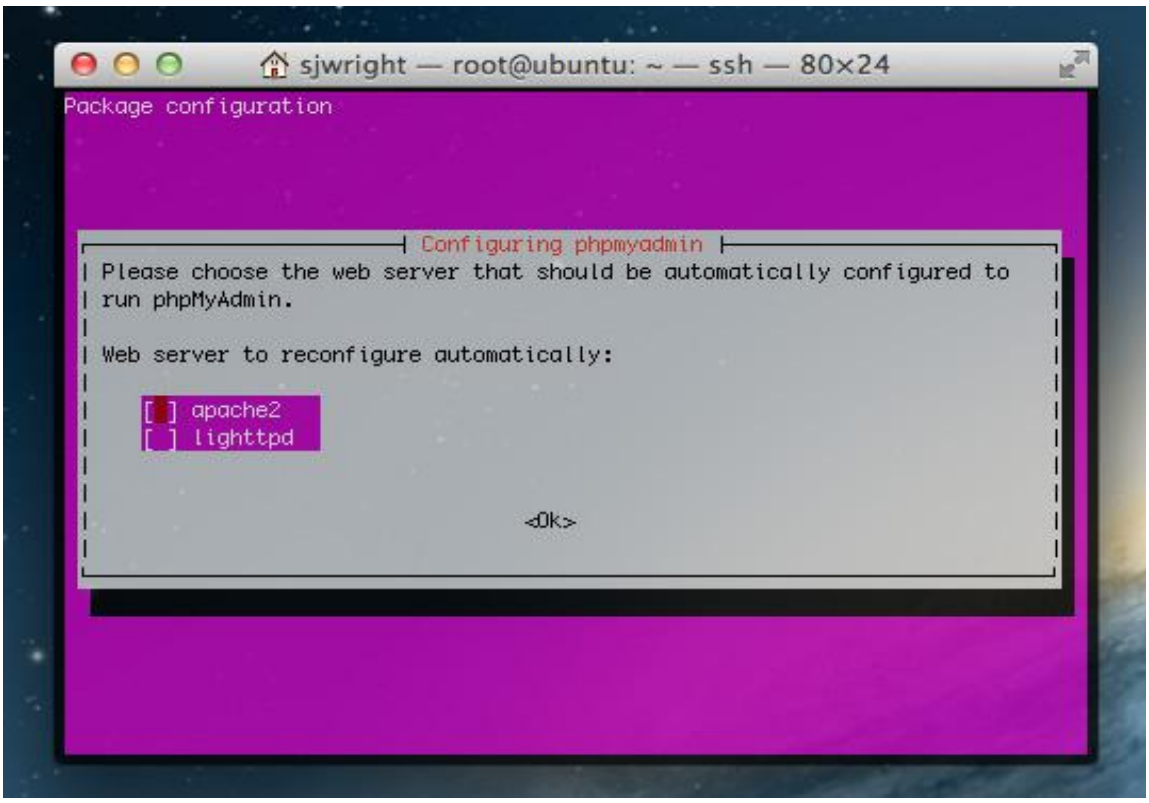
3.2.4 Phpmyadmin

Ahora instalaremos **phpMyAdmin** para manejar fácilmente las bases de datos de MySQL. Para ello ejecutaremos el siguiente comando:

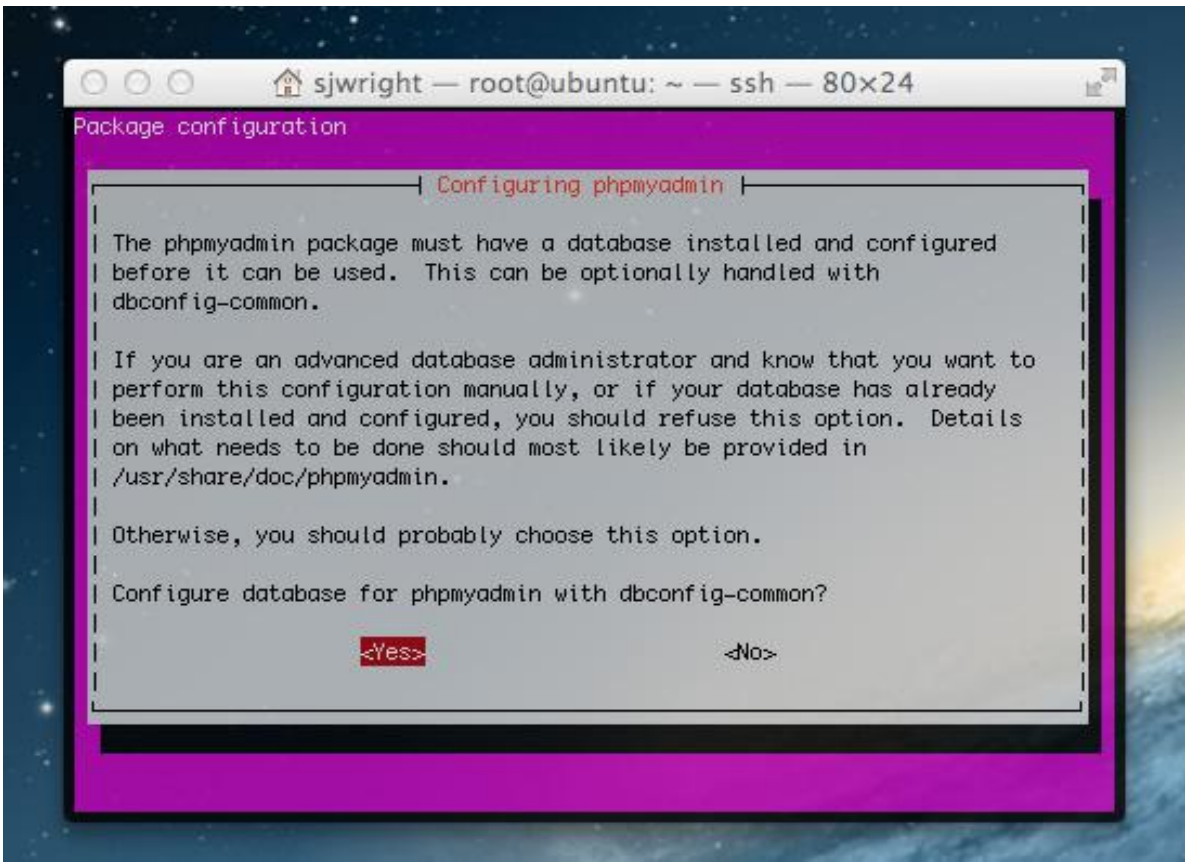
sudo apt-get install phpmyadmin

Durante la instalación nos preguntará qué tipo de servidor tenemos; marcaremos Apache y

continuaremos.



Después nos preguntará si queremos configurar una base de datos; le diremos que sí y nos pedirá que introduzcamos la contraseña de MySQL y nos pedirá de nuevo que introduzcamos una contraseña para phpMyAdmin.



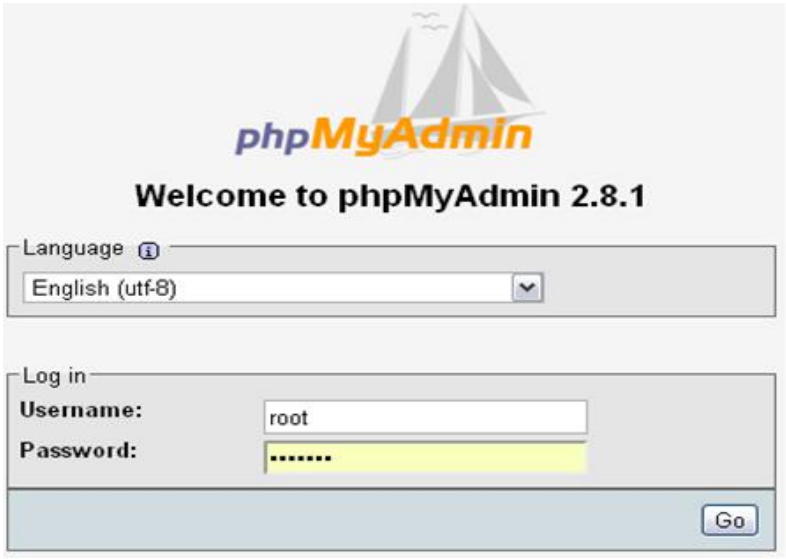
Necesitamos crear un enlace simbólico para acceder a la página de administración de phpmyadmin desde apache

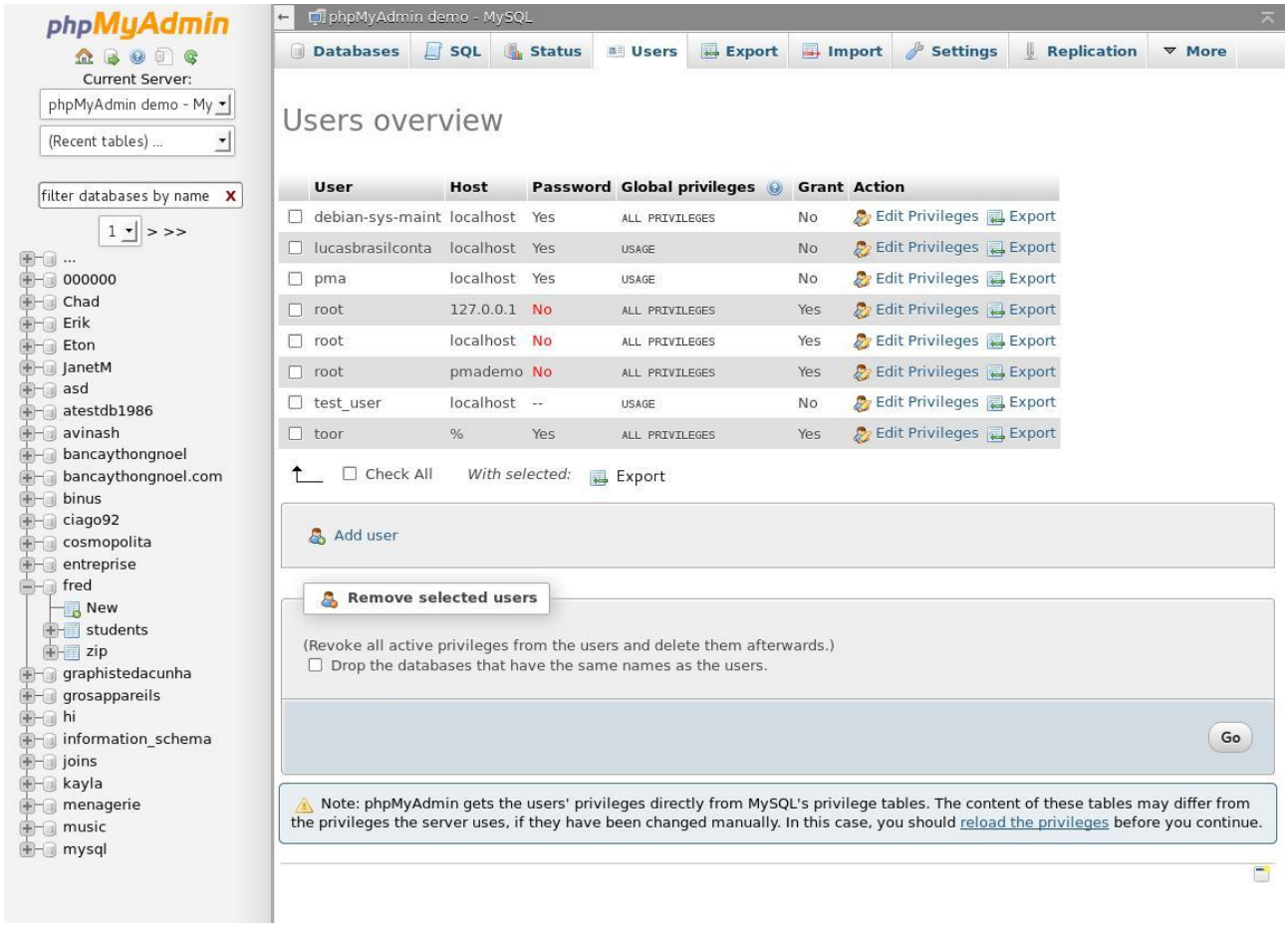
sudo ln -s /usr/share/phpmyadmin/ /var/www/phpmyadmin

Reiniciamos apache, mediante la terminal, ejecutando el comando:

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

Ahora podemos manejar las bases de datos, crearlas, modificarlas y eliminarlas desde un navegador web.


phpMyAdmin
Welcome to phpMyAdmin 2.8.1
Language
English (utf-8)
Log in
Username:
root
Password:
Go


phpMyAdmin
Current Server:
phpMyAdmin demo - My
(Recent tables) ...
filter databases by name
1 > >>
...
000000
Chad
Erik
Eton
JanetM
asd
atestdb1986
avinash
bancaythongnoel
bancaythongnoel.com
binus
ciago92
cosmopolita
entreprise
fred
New
students
zip
graphistedacunha
grosappareils
hi
information_schema
joins
kayla
menagerie
music
mysql
phpMyAdmin demo - MySQL
Databases
SQL
Status
Users
Export
Import
Settings
Replication
More
Users overview
User
Host
Password
Global privileges
Grant
Action
debian-sys-maint localhost Yes ALL PRIVILEGES No Edit Privileges Export
lucasbrasilconta localhost Yes USAGE No Edit Privileges Export
pma localhost Yes USAGE No Edit Privileges Export
root 127.0.0.1 No ALL PRIVILEGES Yes Edit Privileges Export
root localhost No ALL PRIVILEGES Yes Edit Privileges Export
root pmademo No ALL PRIVILEGES Yes Edit Privileges Export
test_user localhost -- USAGE No Edit Privileges Export
toor % Yes ALL PRIVILEGES Yes Edit Privileges Export
Check All
With selected:
Export
Add user
Remove selected users
(Revoke all active privileges from the users and delete them afterwards.)
Drop the databases that have the same names as the users.
Go
Note: phpMyAdmin gets the users' privileges directly from MySQL's privilege tables. The content of these tables may differ from the privileges the server uses, if they have been changed manually. In this case, you should reload the privileges before you continue.

3.2.5 Bases de datos utilizadas

La base de datos se llama “domótica” y consta de 6 tablas relacionadas y normalizadas.

ambientes
- id
- nom

cod_casa
- id
- nombre
- codigo

direccion
- id
- nom
- cod_direccion
- usada

dispositivos
- id
- nom
- id_amb
- id_est
- id_direccion
- id_cod_casa

estado
- id
- nom
- cod

usuarios
- usr
- psw

Conocer los métodos de normalización del diseño de las tablas en tu sistema de BD relacional ayuda a programar un código PHP más fácil de comprender, ampliar, y en determinados casos, incluso hacer tu aplicación más rápida.
Básicamente, las reglas de Normalización están encaminadas a eliminar redundancias e inconsistencias de dependencia en el diseño de las tablas.
Existen cinco pasos progresivos para normalizar

Primer nivel de Formalización/Normalización. (F/N)

1. Eliminar los grupos repetitivos de las tablas individuales.
2. Crear una tabla separada por cada grupo de datos relacionados.
3. Identificar cada grupo de datos relacionados con una clave primaria

Segundo nivel de F/N

1. Crear tablas separadas para aquellos grupos de datos que se aplican a varios registros.
2. Relacionar estas tablas mediante una clave externa.

Tercer nivel de F/N.

1. Eliminar aquellos campos que no dependan de la clave.

3.2.6 Programación de tareas con CRON

Para crear una agenda de tareas a ejecutarse en determinada fecha y hora, desde la página web, mediante un formulario se crea (si no existe, sino solo se agregan datos con cierto formato) un archivo de texto en el servidor, donde se especifican las tareas a ser ejecutadas y en que tiempo (fecha, hora, periodicidad).

Ese archivo es leído y validado por un script en PHP, el cual compara la fecha agendada con la actual y luego, si es que hubo coincidencia, compara las horas agendadas y de sistema. Si también hubo coincidencia se ejecuta la tarea listada en el archivo.

El script PHP se ejecuta periódicamente gracias a cron.

En los sistemas GNU/Linux **cron** es un administrador regular de procesos en segundo plano (*demonio*) que ejecuta procesos a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab. El nombre *cron* viene del griego *chronos* (χρόνος) que significa "tiempo".

Para agregar, quitar o modificar tareas, hay que editar el crontab. Esto se hace con la orden *crontab -e*, que abrirá el editor definido en la variable de entorno **EDITOR** y cargará el fichero **crontab** correspondiente al usuario que está logueado.

El formato del archivo crontab es el siguiente:

```
.---------------- minuto (0 - 59)
|  .------------- hora (0 - 23)
|  |  .---------- día del mes (1 - 31)
|  |  |  .------- mes (1 - 12)
|  |  |  |  .---- día de la semana (0 - 6) (Domingo=0 ó 7)
|  |  |  |  |
*  *  *  *  *  comando para ser ejecutado
```

Se agrega como tarea con frecuencia de cada minuto, el script en PHP, que lee un archivo de texto donde se definen las tareas a ejecutarse y la fecha y hora determinadas por el usuario desde la página web. Para lo cual se agrega al crontab la línea:

m h dom mon dow   command

* * * * * /var/www/domus/ejecutatareas.php &> /home/pi/fichero.log

La anterior línea indica que cada minuto de cada día, indiferentemente del mes o del día de la semana (lunes, martes,..) se ejecutará el script que está en /var/www/domus/

A fines de tener una herramienta de debug en caso de problemas, se agrega un registro mediante:

&> /home/pi/fichero.log

El archivo de texto que funciona como agenda de tareas se llama bastante lógicamente; agenda.txt

En él se guardan con el siguiente formato, análogo del de crontab, las tareas

min :hora :dia :mes :dia_de_la_semana:ambiente :nombre_dispositivo:funcion: codigo_a_enviar

Se muestra a continuación unas líneas del archivo para ilustrar mejor su formato

53:00:*:*:4::LuzPared:on: 1 3 1 1 1 1 1 1 1 1 1 0 0 1 0 1
56:00:*:*:4::LuzPared:off: 2 3 1 1 1 1 1 1 1 1 1 0 0 1 1 1
29:01:23:11:*::LuzPrincipal:off: 2 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1

Por ejemplo ;la última línea agenda que a la 1 y 29 del día 23 de noviembre , siendo indiferente el día de la semana , la luz principal se pondrá en estado OFF , enviando el código : 2 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 , el cual se detallará más adelante.

A continuación se muestra el código de ejecutatareas.php y luego se lo explicará.
El script de PHP se ejecuta mediante línea de comandos, de ahí que se agregue la ruta del ejecutable en la primera línea.

```
#!/usr/bin/php -q
<?php
        $fecha = getdate(time());
        $dia = $fecha['mday'];
        $mes = $fecha['mon'];
        $ano = $fecha['year'];
        $hora=$fecha['hours'];
        $min=$fecha['minutes'];
        $dow=$fecha['wday'];

        $file = fopen("/var/www/domus/agenda.txt", "r+") or exit("Unable to open file!");
        $j=0;
        while(!feof($file))
```

```
	{
		$g[$j]=fgets($file);
		$j++;
	}
	fclose($file);

	foreach ($g as $value)
	{
		list($Min,$Hora,$Dom,$Mes,$Dow,$amb,$disp,$est,$cod) = explode(":",$value);

	if (($Dow == '*') ||($Dom == $dia && $Mes == $mes) || ($Dow == $dow )
					||($Dom == '*' && $Mes == '*') )
				{	if ($hora==$Hora && $min==$Min)
					{
						exec("/var/www/domus/python/codigo7.py ".$cod);

						}
				}
	}
?>
```

Cada vez que se ejecuta, obtiene el día del mes, mes, hora, minutos y día de la semana, con esa información, convenientemente separada se prosigue.
Se abre para lectura el archivo donde está agendado las tareas a realizar y se lo lee línea por línea. Guardando esa información obtenida en la variable vector $g.

Se recorre luego esa variable, separando (explode) según el separador : (dos puntos), y guardando cada parte separada en variables de minutos, horas, día del mes, mes, día de la semana, nombre del dispositivo a comandar, nombre del estado a establecer y el código a enviar .

El código a enviar consta de la suma de:
id del comando + id del dispositivo + código casa + código de dirección + código de estado.

Se compara entonces, para determinar si corresponde ejecutar el código, en el instante que fue invocado el script.

Si se cumple alguna de las condiciones del if, se invoca el script de python llamado codigo7.py, al que se le pasan los argumentos del código a enviar.

Las condiciones son si, el día de la semana es indiferente (*) , o el día del mes y el mes se corresponden con el actual , o el día de la semana fuera el presente, o si el día del mes y el mes agendado sean indiferentes (*).
Cumplida alguna de esas condiciones, finalmente hay otro if anidado, que verifica sea la hora y los minutos agendados para ejecutar el envío de la trama x10.

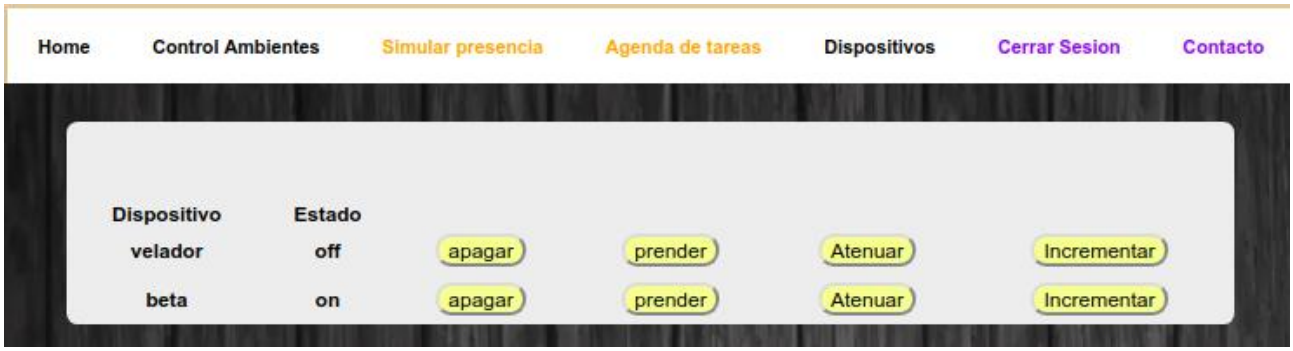
3.2.7- Página Web

La página web a través de la cual se gestiona los dispositivos de la casa, cuenta con acceso mediante usuario y contraseña para brindar cierta seguridad al control del hogar.

Una vez correctamente ingresados, se puede ver un menú con distintas opciones:

'Home' permite cambiar la contraseña.
'Control de ambientes' despliega un submenú con todos los ambientes de la casa.
'Simular presencia', permite agendar tareas a ser ejecutadas automáticamente, ya sea por única vez, o con periodicidad diaria o semanal.
'Agenda de tareas' muestra la lista agendada según el día seleccionado en el calendario.
'Dispositivos' permite gestionar los dispositivos de la casa, agregando nuevos, quitando aquellos que ya no se usen, o modificar sus características, como la dirección, nombre, código casa o ambiente asignado.
'Cerrar Sesión' finaliza la sesión de usuario.
'Contacto' permite enviar un mail al servicio técnico.

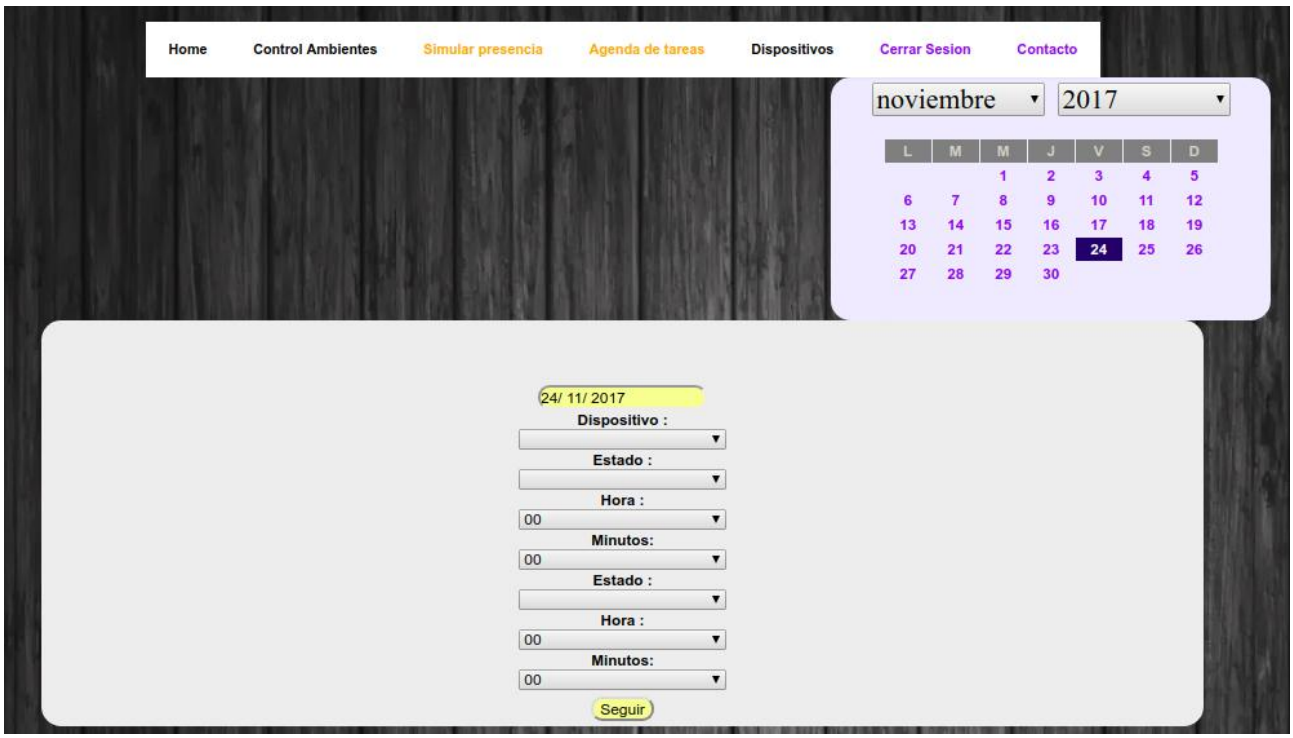
El submenú de ambientes desplegado.



Arriba: Vista de los dispositivos de un ambiente con sus respectivos estados actuales y los botones de comando disponibles para ejecutar.

Abajo: vista de la pantalla ‘simular presencia’, primer paso para agendar un evento.

Abajo: Siguiente paso en el agendamiento de una tarea, tras haber escogido la periodicidad de la misma. Primero se selecciona un día en el calendario, luego se elige el dispositivo a comandar, el estado, la hora y los minutos.
Opcionalmente se puede elegir un segundo comando en distinto horario para el mismo dispositivo.



Arriba: vista de ‘Agenda de tareas’. Allí se visualizan las tareas a ser realizadas automáticamente en el día seleccionado en el calendario. Así también se puede eliminar unas tareas seleccionando en el checkbox y presionando el botón ‘eliminar’.

3.2.8 Código de la Página web explicado

Es posible distinguir entre las **páginas web estáticas** (cuyos contenidos son predeterminados) y las **páginas web dinámicas** (que generan contenidos al momento de solicitar información a un servidor de web a través de lenguajes interpretados como **JavaScript**).Un conjunto de páginas web, por lo tanto, forman un sitio web.

Nuestra página de domótica, por supuesto, es una de tipo dinámica, pues se envía y muestra distinta información según convenga.

Una página web se escribe en lenguaje HTML (Lenguaje de Marcas de Hipertexto, por sus siglas en ingles), y se interpreta en el navegador del usuario.
La parte de procesamiento de datos e interacción con la base de datos se ejecuta en el servidor, y es escrita en otro lenguaje; PHP.
El código PHP, es el que primero se ejecuta y devuelve una respuesta, el cual se incorpora al código HTML, en los lugares donde se encuentra intercalado.

Mediante unos elementos HTML llamados formularios (FORM), se hacen peticiones y se envían variables desde el lado del usuario hacia el servidor, este envío puede hacerse mediante dos métodos, el POST y el GET.
La diferencia entre los métodos GET y POST radica en la forma de enviar los datos a la página cuando se pulsa el botón “Enviar”. Mientras que el método GET envía los datos usando la URL, el método POST los envía de forma que no podemos verlos (en un segundo plano u "ocultos" al usuario).

A continuación se comenta y muestra el código de las principales partes de la página, en cuanto a control de los dispositivos.

Dispo.php

Tras escoger el ambiente a manejar, desde el menú correspondiente, se llama y ejecuta el código HTML con PHP incrustado, expuesto más abajo.
En el mismo , luego de incluir el menú y obtener mediante el método GET la variable con la id del ambiente a controlar, se procede a ejecutar una consulta SQL a la base de datos para conocer todos los dispositivos de dicho ambiente con sus respectivos estados actuales .
El conjunto de datos recabado como respuesta a la consulta, se expone en una tabla HTML, donde cada fila se compone de varios botones, cada uno dentro de un formulario, además del nombre del dispositivo y su estado actual.

```
<form action='cambia_estado.php?id_dis=".$row['id']."&est=2&id_amb=".$id_amb."' method=get
 id=form>
                                                <input type=submit value= 'apagar' >
        </form>
```

Como se aprecia arriba, cada formulario tiene un botón de tipo ‘submit’, mediante el cual, al presionarlo, se lo envía al destino establecido en ‘action’. En el ‘action’ también se define el método de envío, GET, así como también se definen las variables que se mandan al servidor;
El id del dispositivo a controlar, el id del estado a establecer y el id del ambiente al que pertenece.

```
<!DOCTYPE HTML> <HTML>
   <HEAD>
                              <TITLE> Menú Principal </TITLE>
                              <link rel="stylesheet" href="css/style.css">

<?php
session_start();

if (isset($_SESSION['usr_valido']))
          {
                    include('menu.inc.php') ;

          if (isset($_GET['id_amb']))
                    {
                    $id_amb=$_GET['id_amb'];
                    $_SESSION['amb'] =$id_amb ;

                    ?>
</head>
          <body>
          <div id="contenedor">
                    <?php
                    include_once('conecta.inc.php') ;
                    $link=conectar();

                    $sql="SELECT D.nom as nom , E.nom as est , D.id as id  FROM dispositivo D, estado E
WHERE D.id_amb =".$id_amb." AND E.id=D.id_est";

                    if(!($res=mysql_query($sql,$link))){die ( "error en la consulta SQL ".mysql_error());};

                    if (mysql_num_rows($res)>0){
                    echo "
                    <table >
                    <tr>
                    <td>Dispositivo</td>
                    <td>Estado    </td>
                    </tr>" ;
                    }else{echo "<p> No hay dispositivos registrados en este ambiente  </p>";}

while ( $row=mysql_fetch_array($res))
                              {
                              echo "
                              <tr>

                                        <td>".$row['nom']."</td>
                                        <td>".$row['est']."</td>
                                        <td>
<form action='cambia_estado.php?id_dis=".$row['id']."&est=2&id_amb=".$id_amb."' method=post
id=form>
                                                  <input type=submit value= 'apagar' >
          </form>
                                        </td>
```

```
                                        <td>
<form action='cambia_estado.php?id_dis=".$row['id']."&est=1&id_amb=".$id_amb."' method=post
id=form>
                                                                        <input type=submit value= 'prender' >
        </form>
                                        </td>

                                        <td>
<form action='cambia_estado.php?id_dis=".$row['id']."&est=6&id_amb=".$id_amb."' method=post
id=form>
                                                                        <input type=submit value= 'Atenuar' >
                                        </form>
                                         </td>
                                        <td>
<form action='cambia_estado.php?id_dis=".$row['id']."&est=7&id_amb=".$id_amb."' method=post
id=form>
                                                                        <input type=submit value= 'Incrementar' >
        </form>
                                         </td>
                                </tr>
                                ";
                                }
echo "</table>";

        mysql_close($link);
                } // cierra if (isset($_GET['id_amb']))

}
else {
                 header("Location:index.php");

        }

 ?>
```

cambiaestado.php

Este es el script encargado de recolectar los códigos de casa, dispositivo y función de la base de datos y pasárselos al script de python que invoca.

Primero se recogen los id de ambiente, estado y dispositivo enviados por el metodo GET, desde alguno de los formularios de dispo.php.
Luego se arman consultas SQL, de la siguiente estructura:

SELECT dato_a_recoger FROM (desde) tabla_a_consultar WHERE (donde) id=dato_del _formulario

Consultadas las distintas tablas,se formatean las respuestas, insertando un espacio entre caracteres, con str_split (), y luego se las aúna en una variable, que se pasa como argumento al script de Python.
El código en Python se describirá más adelante.

```
<?php  // cambia el estado del dispositivo mandando una orden a través de python

if (isset($_GET['id_dis']))
{	//recogo los datos necesarios enviados
	$id_amb=$_GET['id_amb'];
	$est=$_GET['est'];
	$id_dis=$_GET['id_dis'];
	//me conecto a la DB
	include "conecta.inc.php";
	$link=conectar();

		$sql="SELECT cod,id  FROM estado WHERE id ='".$est."'";

		if(!($res=mysql_query($sql,$link))){die ( "error en la consulta SQL 1".mysql_error());};
		$row=mysql_fetch_array($res);
		$codEstado=$row['cod'];
		$c1=str_split($codEstado);

		$sql2="SELECT cod_direccion  FROM dispositivo as d join direccion as z
				on z.id =d.id_direccion where d.id='".$id_dis."'";

		if(!($res2=mysql_query($sql2,$link))){die ( "error en la consulta SQL 2 ".mysql_error());};
		$row2=mysql_fetch_array($res2);
		$codDispositivo=$row2['cod_direccion'];
		$c2=str_split($codDispositivo);

		$sql3="SELECT codigo  FROM cod_casa as c join dispositivo as d
				on c.id =d.id_cod_casa where d.id='".$id_dis."'";

		if(!($res3=mysql_query($sql3,$link))){die ( "error en la consulta SQL 3 ".mysql_error());};
		$row3=mysql_fetch_array($res3);
		$codDispositivo=$row3['codigo'];
		$c3=str_split($codDispositivo);
		$c=array_merge($c3,$c2,$c1) ;
		$arg=" ".$row['id']." ".$id_dis." ";
		for ($i=0 ; $i<14 ;$i++)
			{
			$arg=$arg." ".$c[$i];
			}
		$cmd="sudo python ./python/codigo7.py".$arg;

		exec($cmd);

	mysql_close($link);
	header('Location:dispo.php?id_amb='.$id_amb);

} else {
	echo "houston we have a problem..";
	}


?>
```

3.3- Configuración del router y del DNS. Acceso desde fuera de la red local.

Para acceder al servidor web y poder controlar los dispositivos de forma remota, es necesario dos cosas; primero un DNS dinámico, y segundo, realizar un port forwarding en el router domiciliario. El DNS, o sistema de nombres de dominio, es el sistema que vincula un nombre dado a una página web, con la IP de la maquina donde corre el servidor que la aloja. Es mucho más fácil de recordar un nombre como www.google.com que una dirección como 201.216.255.255

Al conectarnos al proveedor de internet, este nos otorga una dirección IP distinta en cada ocasión. Por lo que si queremos disponer de un servidor casero accesible desde fuera de la casa, no sería posible, dado que la vinculación nombre-de-pagina-ip registrada en el DNS, seria rápidamente obsoleta. Para mantenerla actualizada es que se usan servicios como noip.com el cual ofrece mantener al día la IP que corresponda a nuestro nombre de dominio (como se llama nuestra página)

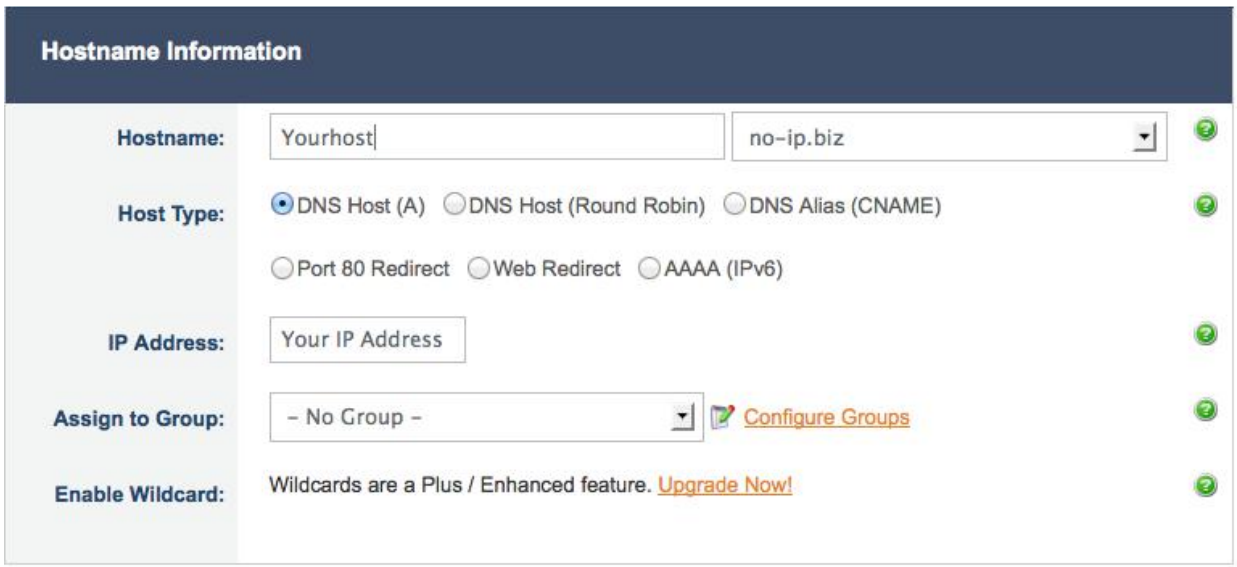
Por otra parte el port forwarding es una forma de NAT (network address translation), donde se redirecciona la información dirigida a un IP1 hacia una IP2.
Desde cualquier lugar, ya con el asunto DNS dinámico resuelto de la forma que se explicará más adelante, se puede solicitar acceso a nuestra página domiciliaria, y esta petición llegará hasta nuestro router, el cual tiene una IP pública dada por el ISP (proveedor de internet).
Si bien suele haber varios dispositivos conectados a la red casera, aunque hubiese uno solo, el servidor que nos interesa, es necesario redireccionar la petición con un cambio de IP, de la pública a la privada que corresponda.

DNS Dinámico

Se utilizó el servicio que brinda la página www.noip.com. Hay que crear una cuenta y luego de registrase, se pueden administrar los nombres de dominio (host) que se asocien a la cuenta.

En la versión gratuita, que se utilizó en este caso, el servicio se brinda con el nombre de dominio que uno elija, pero al que se le adiciona .no-ip.com, para no tener ese agregado hay que pagar.

Abajo, en la figura, como se agrega un host a la cuenta



.

.

• Elección del tipo de host. La opción predeterminada, DNS Host A, suele ser la correcta.

Según aclara la información dada por no-ip, no se debe elegir ninguno de los otros tipos de host a menos que se intente resolver un problema particular que un registro de DNS A no admitirá.

Si el ISP bloquea el puerto 80, por ejemplo, y se está tratando de ejecutar un servidor web u otro servicio en el puerto 80, puede elegir Port 80 Redirect (en ese momento se le pedirá que especifique el puerto que se utilizará para la redirección) Si no está ejecutando un servidor web, entonces no elija esta opción, ya que probablemente le impida conectarse correctamente.

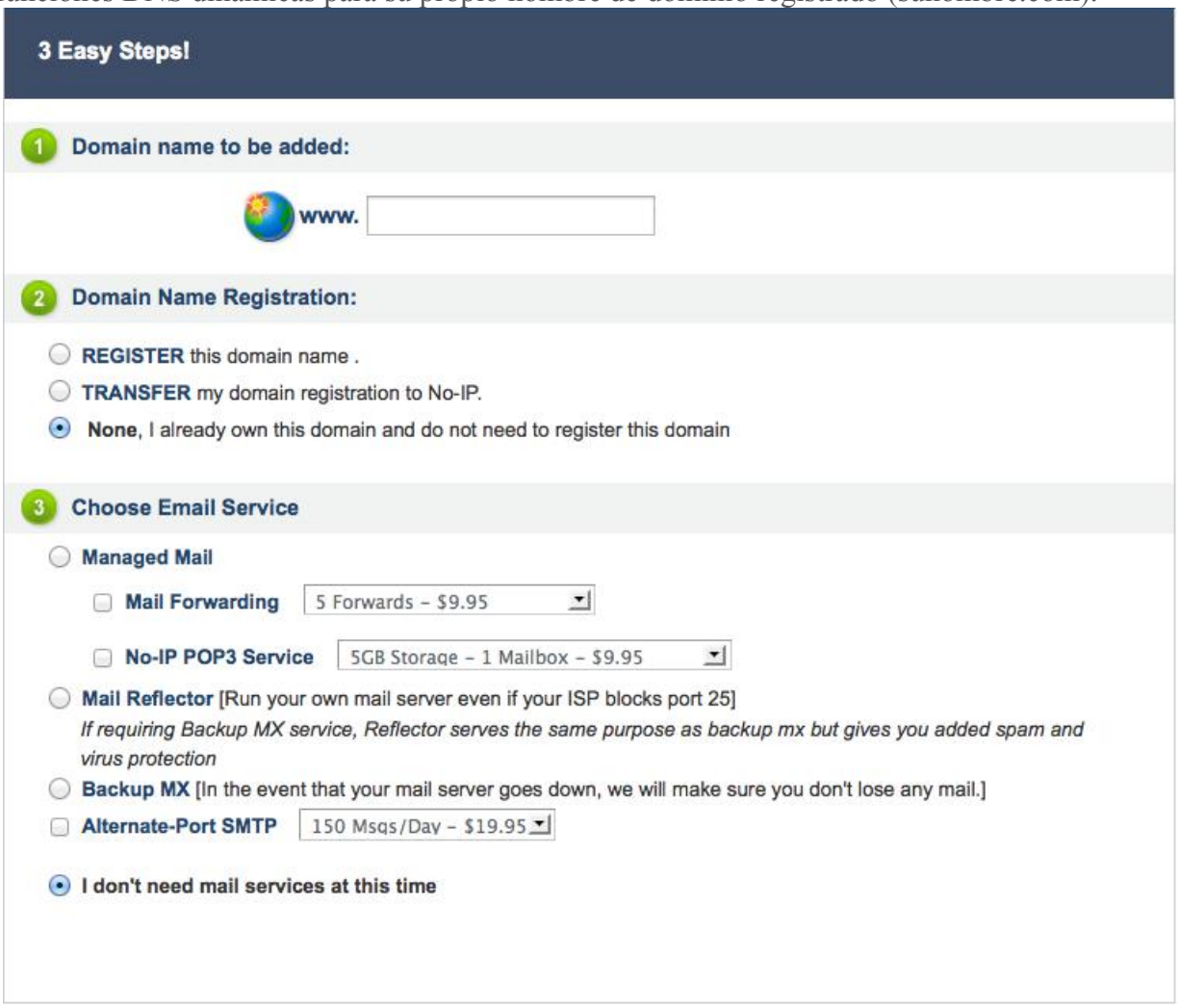
• En el campo marcado Dirección IP, se ve la dirección IP actual desde donde se accede.

Se puede configurar la dirección IP del servidor en un lugar diferente.

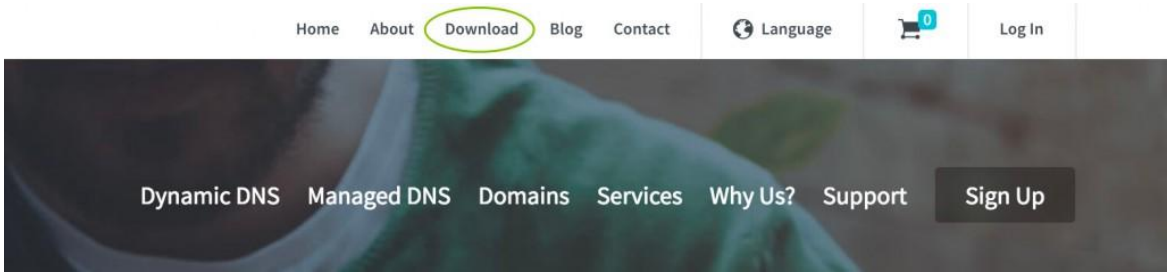
• Las siguientes dos opciones no se utilizan para una configuración de cuenta básica, por lo que no nos ocuparemos de ellas.

• Luego de completada la información para el nuevo nombre de host, se hace clic en el botón "Agregar host" en la parte inferior de la página para guardarlo.

Como fue dicho anteriormente se puede utilizar No-IP para administrar hosts y / o realizar funciones DNS dinámicas para su propio nombre de dominio registrado (sunombre.com).



Finalmente, se instala un programa en la raspberry pi que envía la información de la nueva IP cada vez que esta cambia al sitio no-ip para mantener actualizado el DNS.



El programa se llama Dynamic Update Client (DUC). Es un software gratuito disponible en la sección de descargas del sitio.

Como se usa el sistema operativo Raspbian, un GNU/Linux derivado de Debian, se siguieron los pasos que se detallan a continuación.

Información extraída desde: https://www.noip.com/support/knowledgebase/installing-the-linux-dynamic-update-client/

Instalación del cliente

Los siguientes comandos se deben ejecutar desde una ventana de terminal (símbolo del sistema) después de iniciar sesión como el usuario "raíz". Puede convertirse en el usuario root desde la línea de comandos mediante la introducción de "sudo su -" seguido de la contraseña de root en su máquina.

Nota: *Si no tiene privilegios en la máquina en la que está conectado, puede agregar el comando "sudo" delante de los últimos dos pasos.*

- cd /usr/local/src
- wget http://www.no-ip.com/client/linux/noip-duc-linux.tar.gz
- tar xzf noip-duc-linux.tar.gz
- cd no-ip-2.1.9
- make
- make install

Si se hubiera obtenido "make not found" o "missing gcc", se debería haber instalado las herramientas del compilador gcc .

Configuración del cliente

Como root nuevamente (o con sudo) se ejecuta el siguiente comando:

- / usr / local / bin / noip2 -C *(guion C mayúscula, esto creará el archivo de configuración por defecto)*

Luego se le ingresa el nombre de usuario y contraseña de cuenta de No-IP, así como también los nombres de host que se desea actualizar, si fuera el caso.
Ahora que el cliente está instalado y configurado, solo necesita iniciarlo.
Para iniciar el cliente en segundo plano:

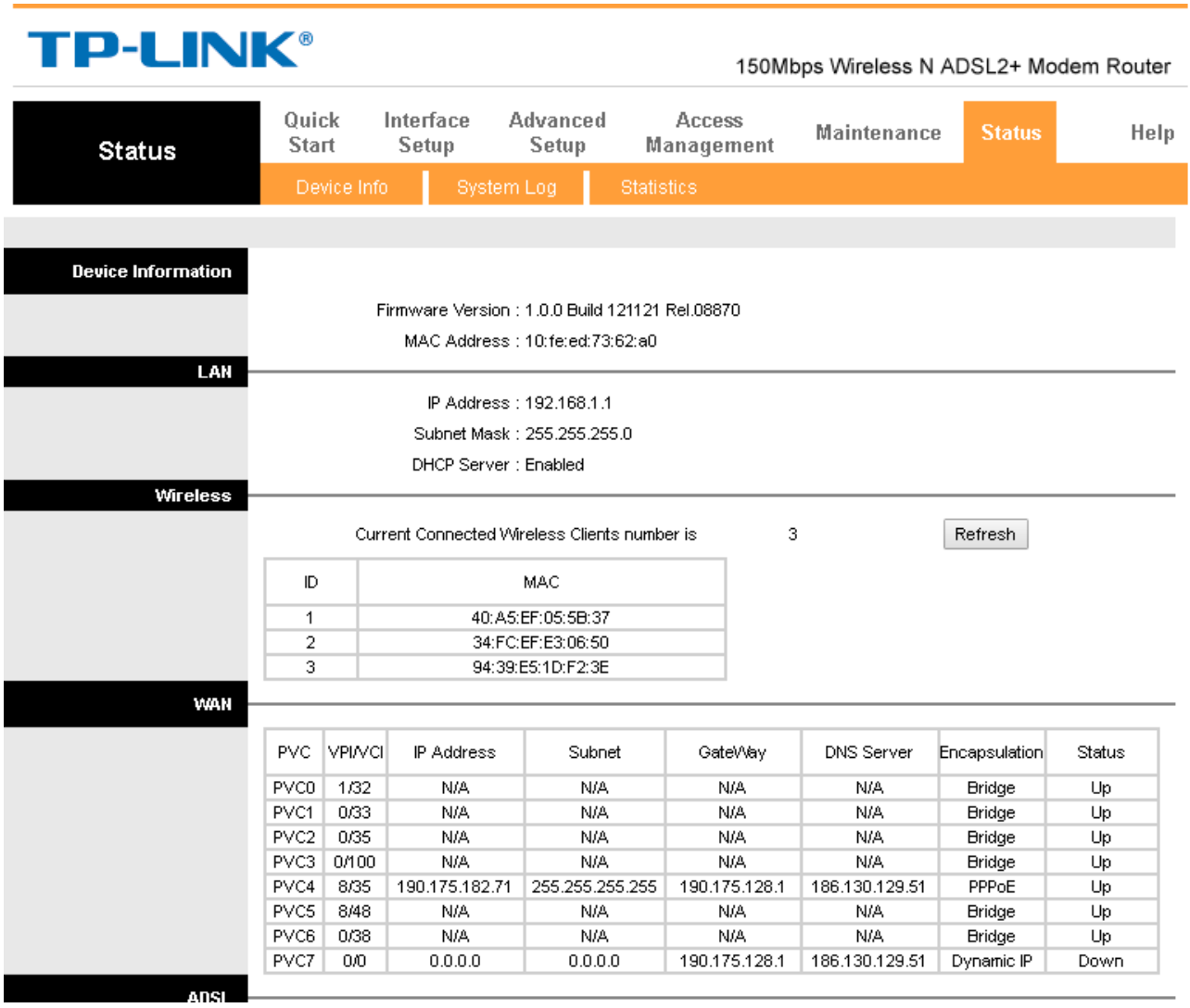
- / usr / local / bin / noip2

Port Forwarding

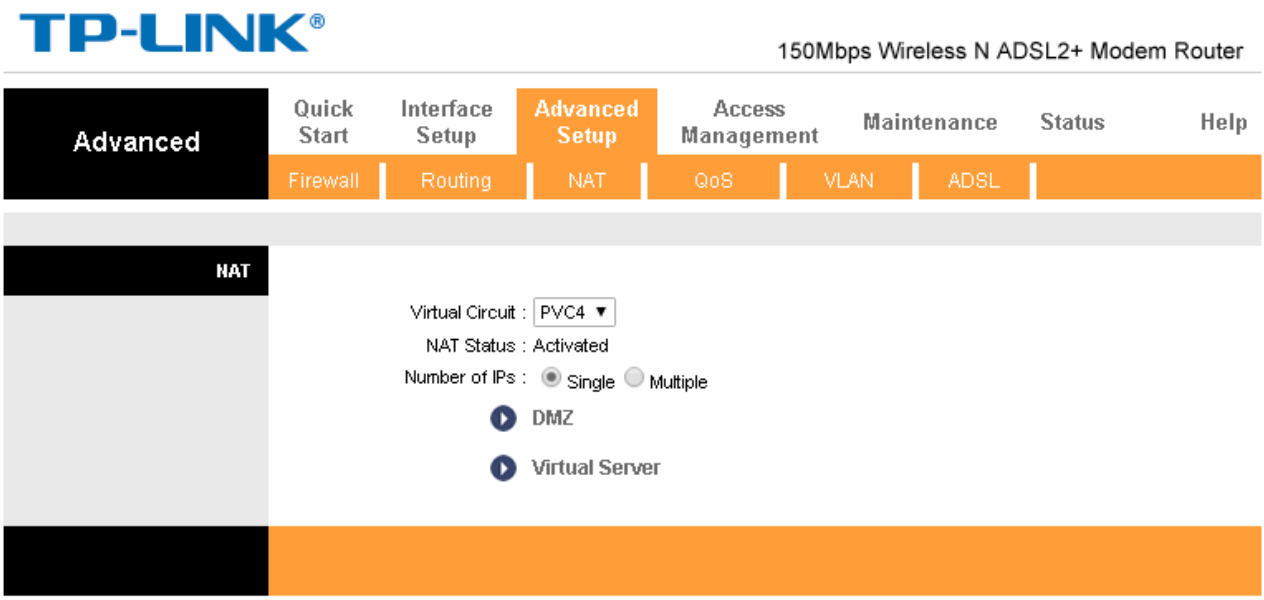
La información viaja por internet en paquetes con la ip de destino y el puerto, el cual identifica el programa o servicio dentro de la computadora al que está destinado. Por supuesto el asunto es más complejo, pero el detallarlo va más allá del alcance de este informe.
El port forwarding o re direccionamiento de puertos consiste en reenviar los paquetes con IP pública según el puerto de destino a la IP privada que corresponda.
En este caso se re envían los paquetes destinados al puerto 80, que es el correspondiente al protocolo de servidor HTTP, a la ip que ostenta la Raspberry Pi.

Para esto se debe ingresar a la configuración del router, ingresando en el navegador la ip 192.168.1.1 . Generalmente es esa, pero si no se puede averiguar desde una consola, la IP del gateway utilizado (o sea el router), mediante el comando route en linux.

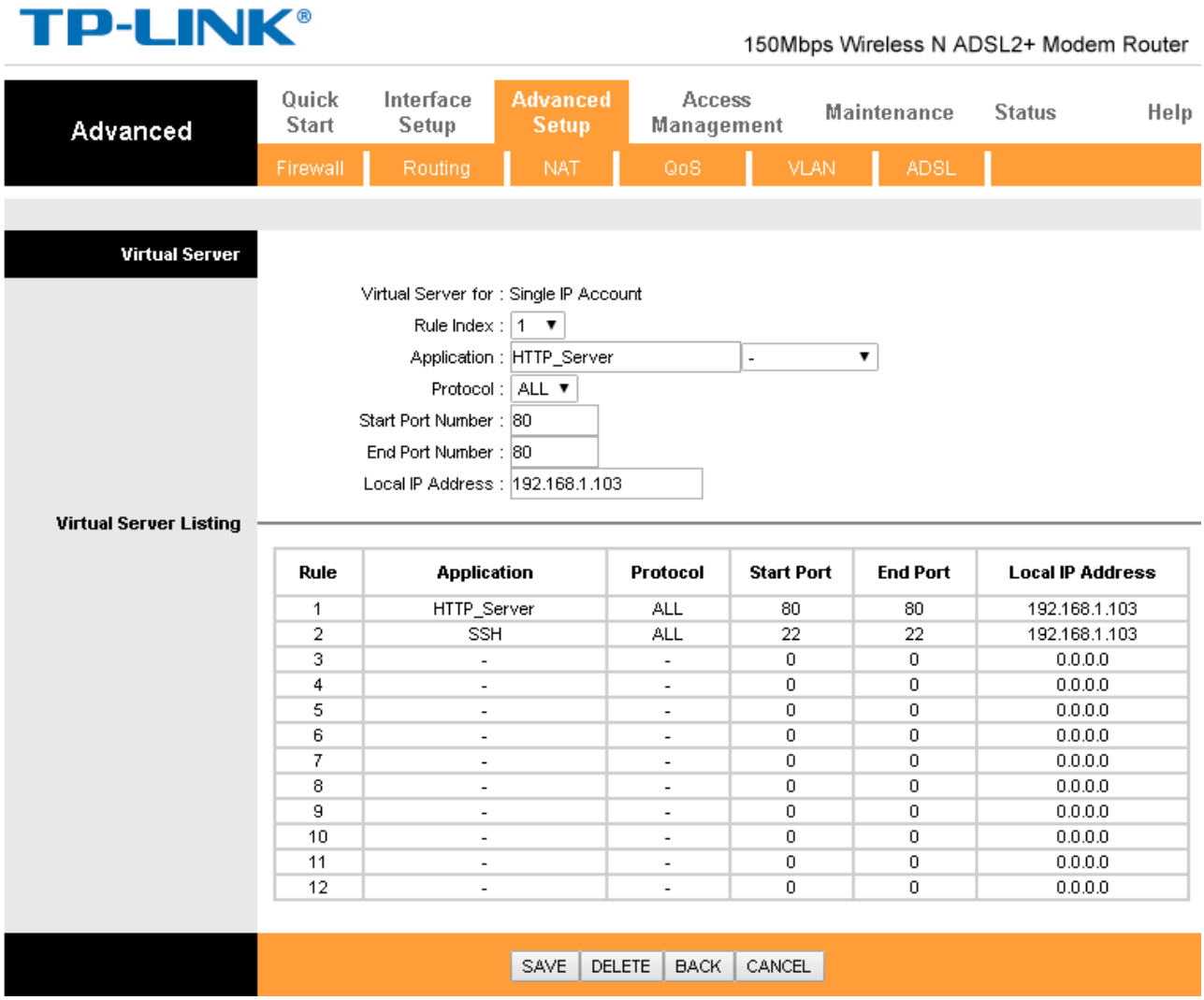
Se ilustra el procedimiento con el route TP-link, es muy similar para otras marcas.
Una vez cargada la página de administración del router, nos pide nombre de usuario y contraseña.
Se los ingresa y se accede a la siguiente pantalla:



Yendo a la pestaña “Advances Setup” →”Virtual Server”



Desde allí se pueden agregar las reglas de redireccionamiento, especificando el protocolo, los puertos que debe usar, y la dirección IP local a la que se redirige.



Rule	Application	Protocol	Start Port	End Port	Local IP Address
1	HTTP_Server	ALL	80	80	192.168.1.103
2	SSH	ALL	22	22	192.168.1.103
3	-	-	0	0	0.0.0.0
4	-	-	0	0	0.0.0.0
5	-	-	0	0	0.0.0.0
6	-	-	0	0	0.0.0.0
7	-	-	0	0	0.0.0.0
8	-	-	0	0	0.0.0.0
9	-	-	0	0	0.0.0.0
10	-	-	0	0	0.0.0.0
11	-	-	0	0	0.0.0.0
12	-	-	0	0	0.0.0.0

4 Modulo Transceptor

Introducción

El modulo transceptor cumple con tres funciones:

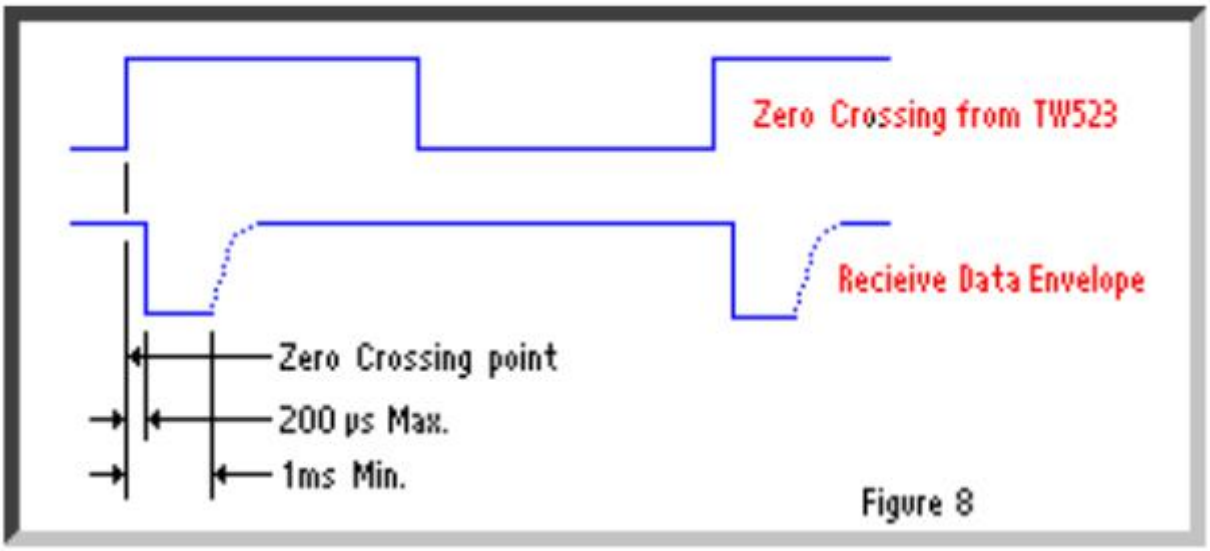
1 Proveer cierta seguridad y aislamiento de la red.
2 Comunicarse con el servidor y la base de datos de la Raspberry Pi.
3 Generar, enviar y recibir las tramas del protocolo X10 por la red eléctrica, sincronizados con los cruces por cero.

En esta sección primero se hará una descripción del funcionamiento del protocolo x10, tras los cual se detallara la comunicación entre el módulo transceptor y la raspberry mediante el uso de los GPIO (General Purpouse Input Ouptut)y el lenguaje Python .Para finalmente adentrarnos en el hardware y el software diseñados

4.1-Descripción protocolo X10

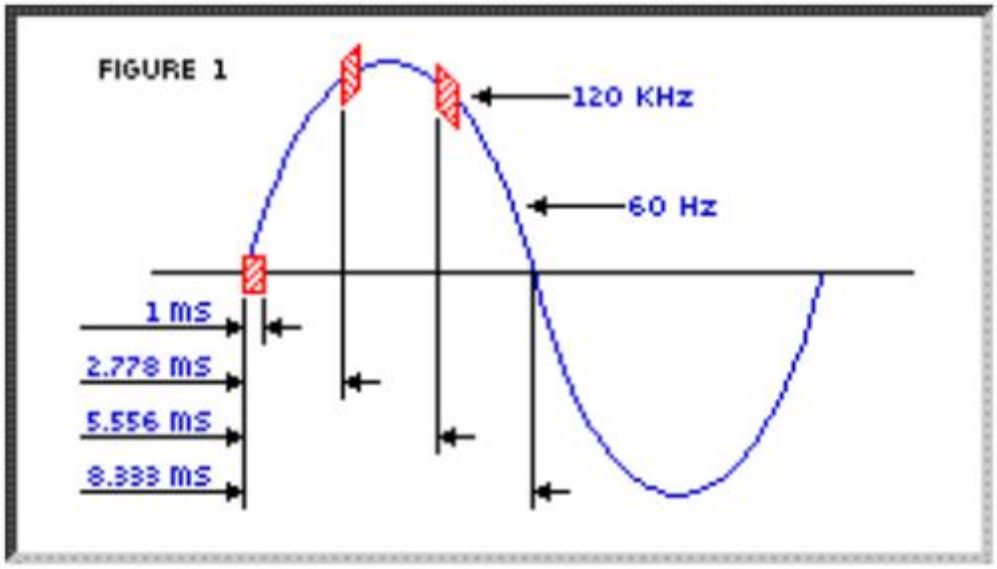
FUNCIONAMIENTO GENERAL DEL PROTOCOLO X-10

Ya se indicó que el protocolo de comunicación X-10 es un sistema de transmisión de datos digitales que usa las instalaciones eléctricas de hogares y oficinas. La versión original de este protocolo permitía realizar seis funciones: encender, apagar, “dim” que significa atenuar, “bright” que significa iluminar, encender todo y apagar todo. Las versiones actuales permiten ejecutar otras nueve funciones además de las tradicionales. Entre las nueve funciones adicionales se encuentran dos que permiten que el protocolo pueda realizar una gran cantidad de tareas nuevas. Estos comandos permiten enviar códigos extendidos (normalmente hasta 256 códigos) que a su vez permiten enviar datos extendidos, normalmente hasta dos bytes adicionales, aunque, dependiendo de la tecnología usada, pueden ser ilimitados si no hay conflictos entre los dispositivos instalados. Para realizar la transmisión de datos se utilizan señales de radiofrecuencia que se inyectan a la red eléctrica, sincronizándolas con los cruces por cero de la señal de poder (60 Hz). Esta técnica es llamada control por corriente portadora (“carrier current” control) 1. Por ejemplo, para transmitir un uno lógico es necesario inyectar señales de radiofrecuencia de 120kHz, dentro de los 200 microsegundos posteriores al cruce por cero de la señal de poder. La Figura muestra un diagrama de los tiempos máximos y mínimos que se deben respetar para la transmisión de un código X-10.



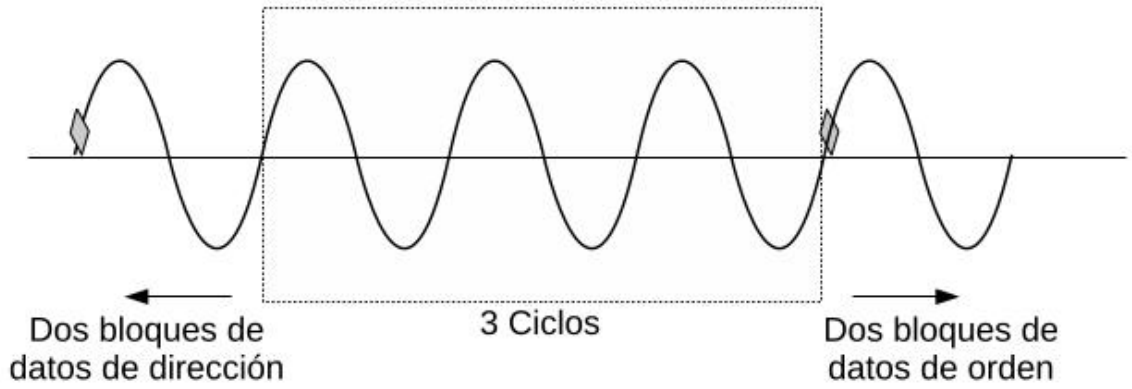
La presencia de las señales de radiofrecuencia en la red debe ser de 1 milisegundo para que el uno lógico sea válido. Un cero lógico es representado por la ausencia de las señales de radiofrecuencia.

Otra consideración importante es que en un sistema trifásico los dispositivos X-10 que se encuentren conectados en fases distintas no pueden comunicarse, a menos de que se coloque un puente de señal que permita que la señal X-10 viaje por toda la red sin importar la fase. Por este motivo es necesario que los pulsos de radiofrecuencia se retransmitan al 1/3 y a los 2/3 del semiperiodo. La ilustración de la transmisión de un uno lógico en X-10 para un sistema trifásico de 60Hz se puede ver en la Figura siguiente.



Un paquete de datos X-10 o byte se transmite utilizando 11 ciclos de la señal de poder. El paquete consta en primer lugar de un encabezado, el cual es transmitido durante dos ciclos; en segundo lugar un código denominado de casa, para el cual se necesitan cuatro ciclos; y finalmente el Código Clave, el cual puede ser un Código de Dirección o un Código de Función. El Código Clave requiere de cinco ciclos para ser transmitido
El Código de Casa junto con el código de dirección sirve para identificar a cualquiera de los 256 dispositivos X-10 admitidos dentro de una red X-10.
El Código de Casa fue pensado originalmente para no tener interferencias entre vecinos que utilicen este protocolo en la automatización de sus hogares. Cada vecino tomaría uno de los 16 códigos de casa y dentro de su hogar distribuiría los 16 códigos de dirección que corresponden a su Código de Casa asignado.
Las interferencias con los hogares vecinos se pueden evitar instalando un filtro a la entrada del cable de alimentación eléctrica del hogar.

Para el envío de un código X-10 completo es necesario transmitir dos bytes con un intervalo entre ellos de tres ciclos completos de la línea de poder. En el primer byte se transmite el Encabezado, el Código de Casa y el Código de Dirección. En el segundo byte se transmite el Encabezado, el Código de Casa y el Código de Función.
La repetición del Código de Casa en los dos paquetes y el envío del complemento de todos los bits, a excepción de los bits del encabezado, da cierta seguridad a la transmisión; es decir, la única manera de verificación de errores que posee este sistema es el envío duplicado de información. De lo indicado se puede deducir que un código X-10 completo se transmite en 25 ciclos de la línea de poder.
Debido a que la tasa de transferencia en un sistema a 60 Hz es de 60 bps, la transmisión del código completo tarda 417 milisegundos. Por esta razón, este protocolo solo es usado para tareas en las que la velocidad de transmisión de datos no es vital.
Adicionalmente, se debe considerar que para mantener la compatibilidad hacia atrás es necesario dejar una pausa de tres ciclos de línea entre el fin del primer comando completo enviado y el siguiente. Esto es necesario ya que los dispositivos análogos que se encuentran en el mercado están diseñados para esperar tres ciclos de línea antes de comenzar a leer el siguiente paquete.



TRAMA X-10

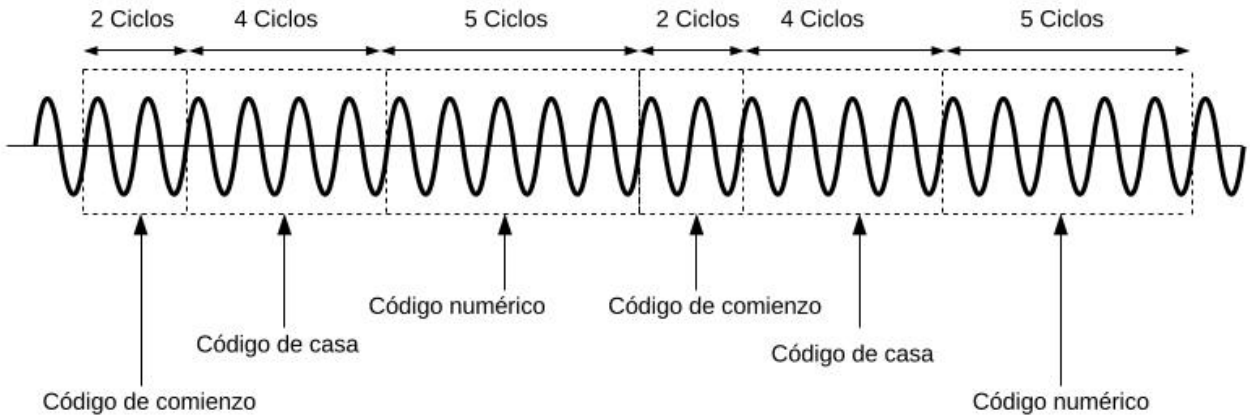
Encabezado

El encabezado siempre es el código “1110”. Los bits se transmiten cada cruce por cero, es decir, un bit por cada semi ciclo de la línea de poder.

Código de casa

El Código de Casa permite 16 diferentes combinaciones, las cuales son identificadas por las letras de la “A” a la “P”.

Para transmitir el Código de Casa se debe transmitir un bit durante el primer semi-ciclo y su complemento lógico durante el segundo semi-ciclo, por lo que la transmisión de cada bit se la realiza en un ciclo completo de la línea de poder. Por ejemplo, para transmitir el Código de Casa “A” (0110) se debe transmitir la secuencia:
0 1 1 0 1 0 0 1
Lo que implica ocupar 4 ciclos de la línea de poder.



Código Clave

Este código permite las combinaciones siguientes. Las 16 primeras corresponden a la dirección del dispositivo y las otras 16 corresponden a una función a ser ejecutada. Las direcciones de los dispositivos son identificadas por los números del 1 al 16. Existen 15 comandos diferentes dentro del protocolo X-10 Pro.

Consideraciones para enviar códigos X-10

No todos los comandos se envían de la misma manera, las excepciones y consideraciones especiales son las siguientes.
En primer lugar, se mencionó que se debe dejar una pausa de tres ciclos de la línea de poder entre un comando X-10 completo y otro para poder mantener la compatibilidad hacia atrás con los módulos antiguos de X-10 PRO.
En el caso de que el comando sea para toda una casa (es decir, todos los dispositivos tienen el mismo Código de Casa) es suficiente con el envío de un solo byte, el cual contendrá: encabezado,

Código de Casa y Código Clave. Los comandos que se aplican a todos los dispositivos con el mismo Código de Casa son: encender todas las luces, apagar todas las luces y apagar todos los dispositivos.
Cuando se ejecuta un código de Dim o Brigth se aumenta o disminuye el voltaje de alimentación en un paso. Si se desea variar más que un solo paso se envían más comandos de Dim o Bright. Para ello se envía la dirección en el primer byte y10 el primer código de Dim en el segundo byte. Los dos primeros Bytes se pueden transmitir normalmente pero los siguientes comandos de Dim o Bright se envían consecutivamente, sin ninguna pausa entre los bytes. Generalmente son 20 el número de pasos de voltaje que se pueden dar para apagar o encender completamente un módulo.
Para el envío de una instrucción de Código Extendido se envían normalmente los dos primeros bytes y, sin ninguna pausa, se envía un byte de ocho bits el cual representa un comando distinto de los 15 comandos estándar del protocolo X-10.
El código "Pedir Saludo" sirve para buscar otros dispositivos capaces de enviar códigos X-10 PRO dentro del rango de comunicación del dispositivo que lo envía.
Esta petición de saludo será respondida por cualquier otro transmisor/receptor de señales X-10 que reciba el comando "Pedir Saludo" y tenga el Código de Casa recibido en dicho comando.
La petición de saludo se responde con un comando de "Contestar Saludo". Estos comandos permiten evitar que dos transmisores X-10 tengan la misma Dirección de Casa e impedir que algún dispositivo de otro propietario de controladores X-10 interfiera con el funcionamiento del sistema a instalarse.
Lo más aconsejable para evitar estas interferencias es colocar un filtro contra radiofrecuencia en el cable de alimentación del hogar donde se van a instalar dispositivos X-10.

Para una instrucción de Dim conteniendo un valor pre seteado, el bit D8 es el bit más significativo del nivel de luminosidad y los bits H1, H2, H4 y H8 son los menos significativos de los 5 bits que contienen la información del "set point" de luminosidad. La combinación de estos 5 bits permite 32 combinaciones distintas, pero por lo general los actuadores solo permiten valores entre 0 y 20. Para enviar esta instrucción el contenido del segundo byte es el único que se debe alterar.
El código de Dato Extendido se usa para recibir información de cualquier tipo, como por ejemplo enviar pequeños archivos de eventos ocurridos durante el día desde un sensor hasta el Módulo de Control. La transmisión es normal para los dos primeros bytes (como en el caso de Dim o bright) pero los siguientes bytes11 serán de 8 bits y sin pausas entre ellos. Se recomienda que el primer byte adicional contenga el número de bytes de 8 bits que se van a transmitir, aunque eso depende del protocolo fijado por el programador para el envío de datos extendidos, porque el protocolo X-10 no ha normalizado esto.
Los comandos para petición de estado y reporte de estado se usan para dispositivos con capacidad de transmisión bidireccional y sirven para monitorizar el funcionamiento de los módulos X-10.

4.2 Comunicación entre la Raspberry Pi y el módulo transceptor

Manejo de los puertos GPIO de la Raspberry Pi

En la Raspberry Pi podremos encontrar pines GPIO los cuales comúnmente se utilizan para los sistemas embebidos. Dado que el GPIO puede ser de entrada y/o salida (un ejemplo de entrada es cuando se pulsa un botón y envía una señal y la de salida cuando se indica a la Raspberry pi que encienda un LED) se podrán conectar accesorios con la correcta configuración como un teclado o altavoces.
También hay que tomar en cuenta que los componentes electrónicos cuentan con dos tipos de señales; señal digital y analógica.

- La **señal digital** es cuando pulsamos un botón, el ordenador no sabe con qué fuerza lo has pulsado, simplemente representa su estado.
- La **señal analógica** cuando varía de forma continua a lo largo del tiempo. La mayoría de las señales que representan una magnitud física (temperatura, luminosidad, humedad, etc.) son señales analógicas. Las señales analógicas pueden tomar todos los valores posibles de un intervalo. En la Raspberry Pi si queremos introducir señales analógicas habrá que convertirlas a digitales.
-

Hay que recordar que la Raspberry Pi GPIO de entrada y salida, dicho puerto se encuentra en la esquina superior derecha, viendo de frente al puerto ethernet.
El número de pines ofrecidos es de 26 (13×2) y pueden ser usados para SPI, I2C, transmisión serie UART, alimentación 3v3 y 5V, GND, PWM y demás E/S.
Algo importante es que dichas conexiones no son “plug and play” por lo cual se debe tener cuidado al conectar para evitar conexiones incorrectas, así mismo, los pines usan 3V3 y no toleran 5V y si no hay protección contra el alto-voltaje simplemente quemarían la Raspberry conectando un pin a 5V.

- Como los pines pueden ser configurables según lo que se necesite, si no se hace uso de SPI, I2C y UART se podrían deshabilitar estas funciones y obtener 17 pines GPIO disponibles (8 pines GPIO + 5 pines SPI + 2 pines I2C + 2 pines UART = 17).
- Se debe usar la corriente mínima posible para cada pin porque puede haber salidas simultáneas de conmutación y esto causar interferencias.
- Cada pin GPIO puede entregar hasta 16 mA, pero el total de todos los pines no debe superar los 50 mA. Por lo cual si se quiere usar los 17 pines, ninguno tendría que consumir más de 3 mA.
- Si hacen falta más pines GPIO se puede expandir el número de pines usando circuitos expansores de pines GPIO que les hay para interfaces i2c o SPI.

RPi.GPIO es una librería de Python que simplifica el uso de los pines GPIO, una vez instalada un simple led puede ser comandado con 3 líneas de Python. Instalar la librería es muy simple, usando la consola se escribe y ejecuta lo siguiente:

$ wget http://pypi.python.org/packages/source/R/RPi.GPIes O/RPi.GPIO-0.1.0.tar.gz
$ tar zxf RPi.GPIO-0.1.0.tar.gz
$ cd RPi.GPIO-0.1.0

$ sudo python setup.py install

Ahora para manejar los pines basta escribir en un archivo script.py, al que le damos los permisos correspondientes, lo que se ve a continuación:

```
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(18, GPIO.OUT)
GPIO.output(18,False)
```

Se importa la librería, se configura el uso de la numeración de pines de la placa, y luego de establecer el pin 18 como salida, se la establece en cero volt, nivel lógico de "false".

Igualmente se la puede estableciendo en alto, fácilmente con

```
GPIO.output(18,True)
```

Lo cual nos permite, incluyendo la librería time, que agrega el uso de retardos, generar una sucesión de pulsos.

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
GPIO.setup(7, GPIO.OUT)

numTimes=25
speed=0.2
for i in range(0,numTimes)
        GPIO.output(7,True)
        time.sleep(speed)
        GPIO.output(7,False)
        time.sleep(speed)

GPIO.cleanup()
```

En el código arriba mostrado, la salida número 7, se enciende y apaga 25 veces, a un ritmo de 0.2 segundos.

Ahora se pasa a explicar el código utilizado en el sistema.

```
#!/usr/bin/python2.7
import RPi.GPIO as gpio
import time
import sys
import MySQLdb
```

```
gpio.setmode(gpio.BOARD)
gpio.setwarnings(False)

gpio.setup(13,gpio.IN,pull_up_down=gpio.PUD_DOWN)
gpio.setup(21,gpio.OUT)
gpio.setup(23,gpio.OUT)

global i
global buffer
global id_disp
global id_est
```

Tras la definición de variables e inclusión de librerías se definen las funciones que luego serán llamadas y usadas en el resto del código.
La función ‘guardar’, se conecta con la base de datos y ejecutara una consulta, mediante la que se guarda una actualización del estado de un dispositivo. Tanto el id del dispositivo, como el id del estado, se pasan como argumentos a la función.

```
def guardar(a,b):
        try:
                db = MySQLdb.connect("localhost","root","omar4072","domotica_2")
                cursor = db.cursor()
                query = "UPDATE dispositivo SET id_est ='%i' WHERE id = %i" % (int(a), int(b))
                cursor.execute(query)
                db.commit()
                cursor.close()
                except db.Error ,e:
                print "Error %d: %s" % (e.args[0],e.args[1])
                sys.exit(1)
        finally:
                if db:
                        db.close
                sys.exit()
```

Acá abajo se puede ver la parte específica que se encarga de enviar la información a la placa transceptora.
Los datos pasados como argumentos al invocar el script de python, se recuperan dentro del script, mediante la variable vector sys.argv[].
Primero se guardan en variables, los id de estado y de dispositivo. Luego se pasa a recorrer el resto del vector. Se usa el pin 21 como clock, al ritmo impuesto por el retardo definido en la variable ‘ret’; y el pin 23 se utiliza para definir la información a enviar.
El flanco descendente del clock , indica al microcontrolador el instante donde leer el dato enviado.
Si bien existe una librería para el uso del protocolo SPI, se creyó innecesario su uso, dado la simplicidad de la transmisión, donde solo hay dos dispositivos implicados.

```
buffer=[]
ack=[1,1,0,1]
nack=[1,0,1,0,1]
i = 0
flag = 0
```

```
ret = 0.1
id_est = sys.argv[1]
id_disp = sys.argv[2]

for x in sys.argv[3:]:
        if  '1' in x :
                gpio.output(21,True)
                gpio.output(23,True)
                time.sleep(ret)
                gpio.output(21,False)
                gpio.output(23,True)
                time.sleep(ret)

        if '0' in x :
                gpio.output(21,True)
                gpio.output(23,False)
                time.sleep(ret)
                gpio.output(21,False)
                gpio.output(23,False)
                time.sleep(ret)
```

Luego de enviar al módulo transceptor los datos, se agrega una detección de evento. Con ello se pone en alerta de que ocurra un flanco ascendente en el pin 13 de la Raspberry, cuando ello ocurra, se llamará a la función especificada en ‘callback’.

```
gpio.add_event_detect(13,gpio.RISING,callback=guardar(id_est,id_disp))
time.sleep(9);
gpio.cleanup()
sys.exit()
```

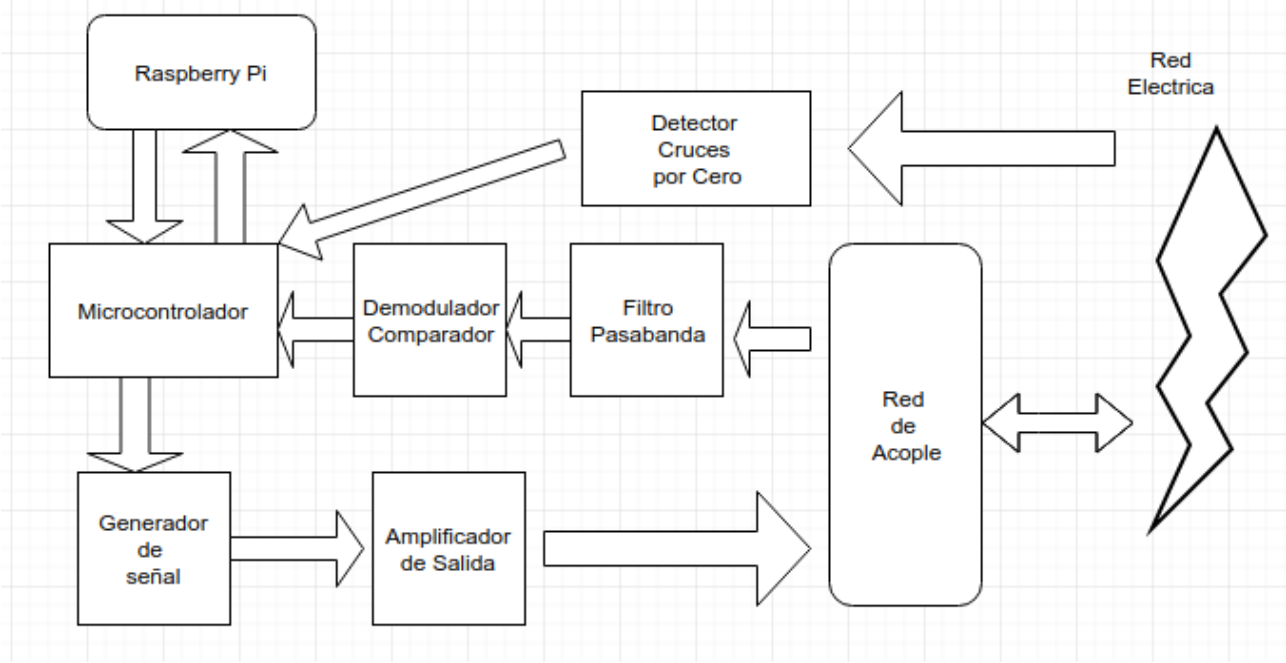
Se especifica una pausa de 9 segundos, antes de limpiar las entradas y salida, y concluir el programar .Ese tiempo es necesario para dar lugar al envío y posible re envío de la trama x10 entre dispositivos. Los 9 segundos son excesivos, antes se recibirá el ACK, confirmando el envío, y lanzando el llamado a la función ‘guardar’.
Si no llega, el estado no cambia.

4.3- Hardware del módulo

Este es el modulo que permite enviar y recibir a través del cableado de energía domiciliario las tramas de mensaje del protocolo x-10. Tiene también la funcionalidad de poder comunicarse con la raspberrry pi, de donde recibe el código del dispositivo a controlar y el código de función a ejecutar, y al cual envía la confirmación o el aviso de fallo.
La misma placa, con el mismo hardware, puede ser empleada como transmisor con conexión con el controlador principal (Raspberry pi), o como receptor, adjunto a los dispositivos manejados remotamente.

Se creyó conveniente implementar circuitalmente la posibilidad de re programar el microcontrolador, que es el cerebro de la placa, mediante una tira de pines, por las que se puede conectar el PicKit 3

El siguiente diagrama de bloques presenta las principales partes del módulo:



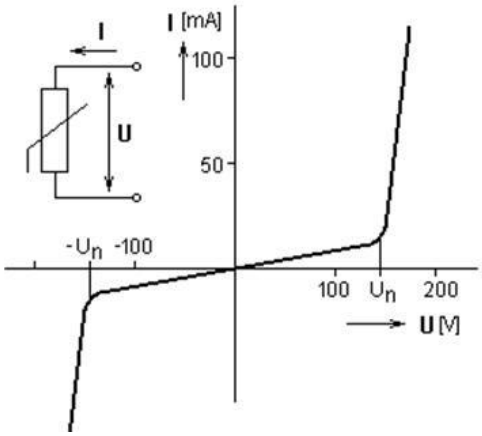
Red de acople a la red eléctrica

La red de acople cumple la función de aislar y proteger los circuitos de la placa, de la tensión de línea, tanto como de ser de medio de inyección y recepción de señal, hacia y desde el cableado de red eléctrica.
Para el aislamiento se usó un transformador con núcleo de ferrita, en relación 1:1.
Se escogió está vía por ser simple, efectiva y con muy poca perdida de potencia de las señales que lo atraviesan.
La otra opción considerada era un opto acoplador, pero este no brinda tanto aislamiento, pues requiere de una fuente de alimentación del lado de la red, complicando además el hardware necesario para la implementación.

Como la señal a inyectar es una ráfaga de 120Khz de portadora, se hizo necesario usar un núcleo de ferrita. A diferencia de los transformadores de fuerza que poseen un núcleo de acero al silicio, los enrollados de los transformadores para uso con corrientes de RF frecuencia se colocan en un núcleo de ferrita, material que se obtiene por pulvimetalurgia (o metalurgia de polvos) sometiéndolo a un proceso de sinterización. La ferrita tiene la propiedad de ofrecer muy buena respuesta a la inducción electromagnética generada por las corrientes de alta frecuencia.

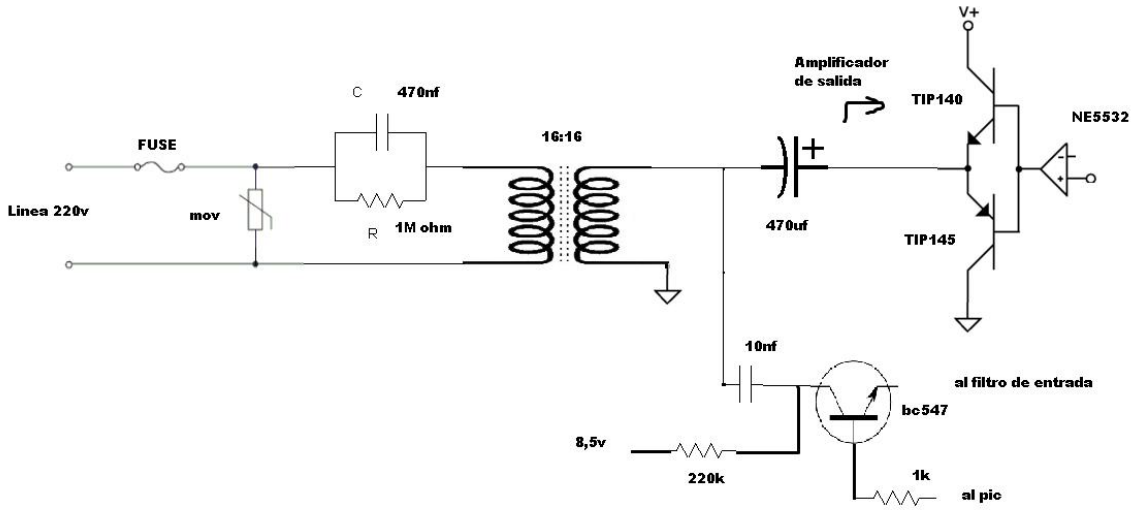
Para incrementar la seguridad, se incluyó un fusible y un varistor a la entrada, para proteger ante sobre tensiones, tanto esporádicas, como de duración prologada.

El varistor es un elemento que tiene la propiedad de presentar una resistencia variable en sus bornes, dependiendo de la tensión a la que se someta. Idealmente se comporta como un circuito abierto para la tensión nominal de la red, pero se cortocircuita para valores de tensión superiores a el.



La figura de la derecha ilustra la relación tensión-corriente del varistor.
Hasta la tensión nominal, tiene una corriente muy baja que lo atraviesa, superado ese umbral, puede ser atravesado por una gran corriente.

El par fusible-varistor funciona de la siguiente manera; al existir una sobre tensión en la línea, el varistor cambia su resistencia, desviando el exceso de corriente y protegiendo el resto del circuito. Si la sobre tensión dura en el tiempo, el fusible se quema, cortando el paso de corriente desde la red a la placa.

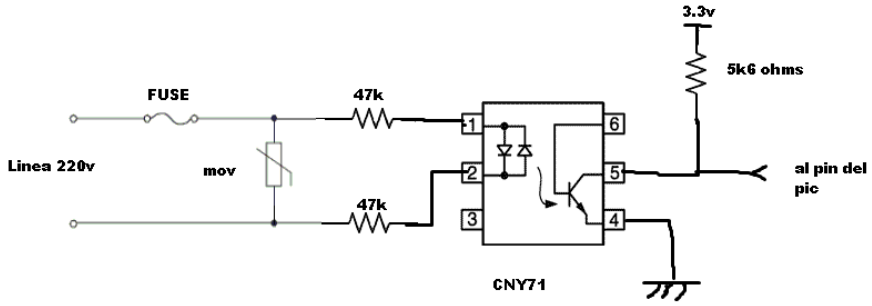


Un filtro RC evita el paso de los 220v, al menos en su mayor parte, y permite el paso de las ráfagas de 120Khz, desde y hacia la red eléctrica.

Por ultimo un transistor BC547 sirve de llave para separar el circuito de entrada cuando se está transmitiendo hacia la red, evitando así el desvío de la potencia.
El transistor se controla desde la lógica del microcontrolador, energizando su base o no según la ocasión.

Detector de cruce por ceros

El sincronismo para el envío y recepción de los bits de la trama x-10, está dado por este circuito. Se implementó mediante un opto acoplador de dos fotodiodos, el CNY 71.

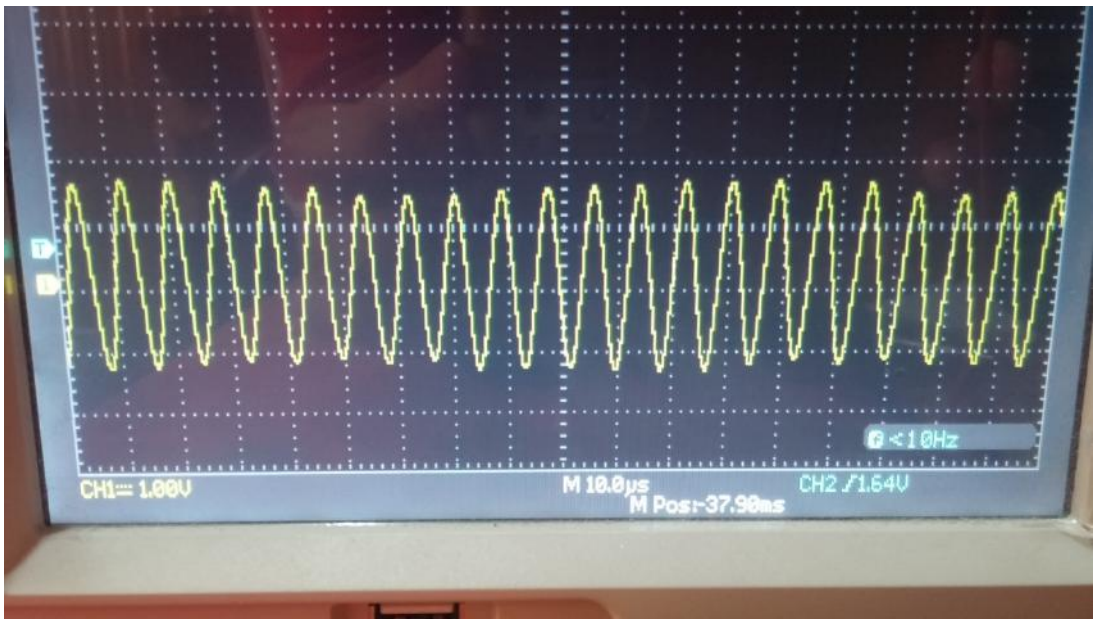


El opto acoplador brinda seguridad al separar eléctricamente las partes del circuito. La base del transistor se activa por la luz emitida por los fotodiodos al ser atravesados por la corriente.
Al actuar como llave que abre (apaga) y cierra (satura), al ritmo de los cruces por cero, da una sucesión de pulsos de sincronismo que van al microcontrolador.

En la figura se puede apreciar el tren de pulsos vistos en el osciloscopio .Están separados cada 10ms, con cada cruce por cero de la red eléctrica de 50Hz.

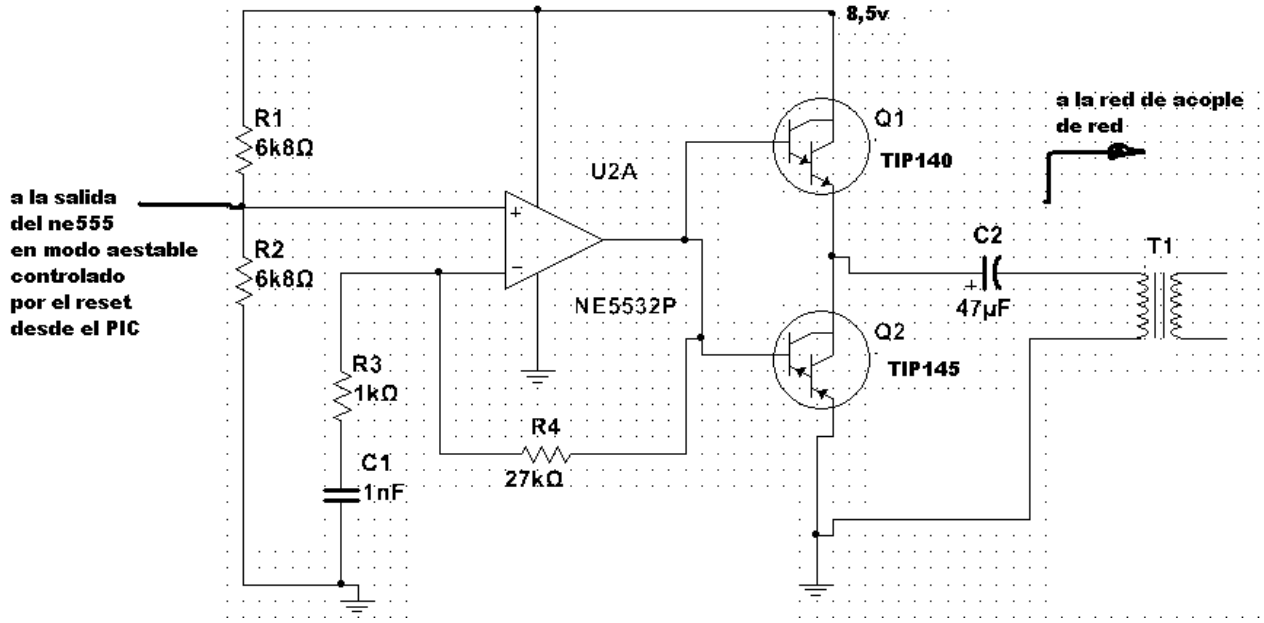
Circuito transmisor

La portadora de 120KHz se genera mediante un simple NE555, el cual se controla por su pin de reset desde el microcontrolador.
El integrado tiene la ventaja de ser simple, barato y dedicado, al no depender más que de los valores de las resistencias y capacitores empleados.
Se dispuso de una resistencia variable, del tipo multivuelta, para mejor ajuste de la frecuencia generada.

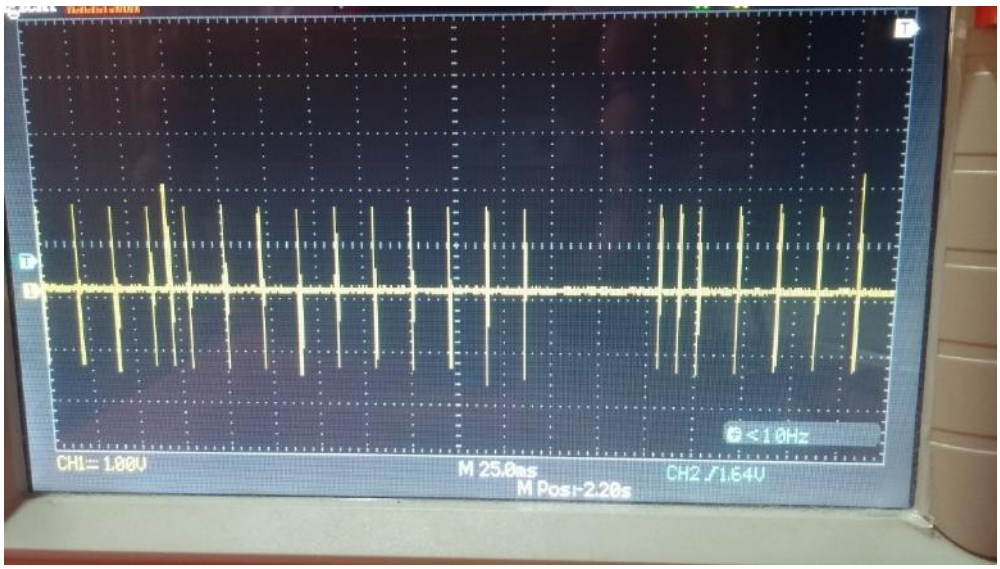


En la figura se puede observar la señal de 120KHZ en el osciloscopio.

La señal así obtenida se amplifica antes de ser inyectada en la red, esto se logra por el uso de un amplificador clase AB, en el cual la parte del driver es ejecutada por un amplificador operacional NE5532.

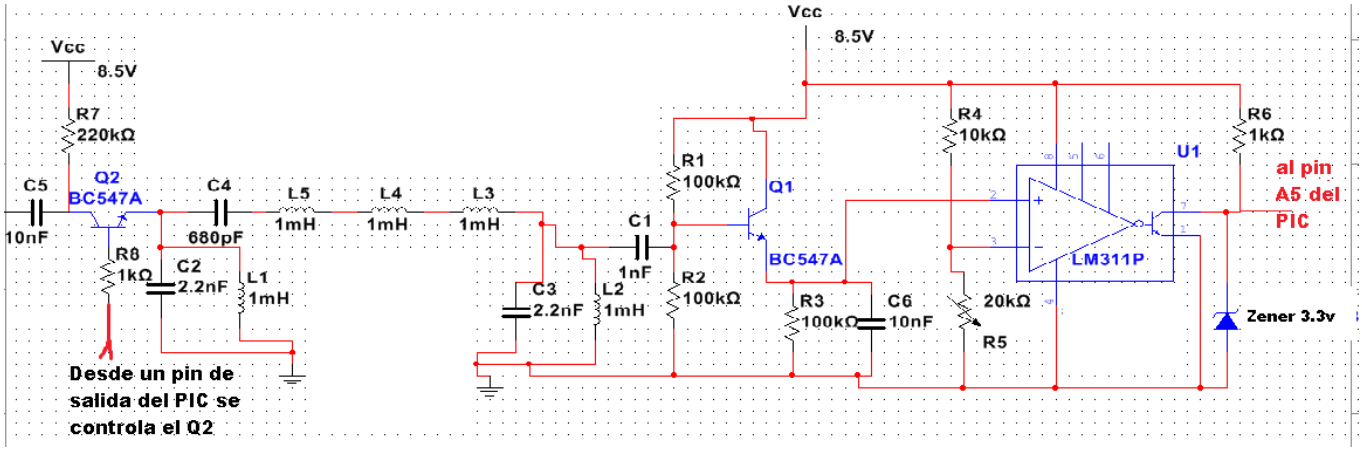


La salida clase B, está compuesta de un par de transistores de tipo Darlington, para asegurar la potencia deseada, la cual es relativamente alta, dada la baja impedancia de la línea eléctrica.



Arriba, en la figura, una vista de la trama x10 enviada, vista en la pantalla del osciloscopio.

Circuito receptor



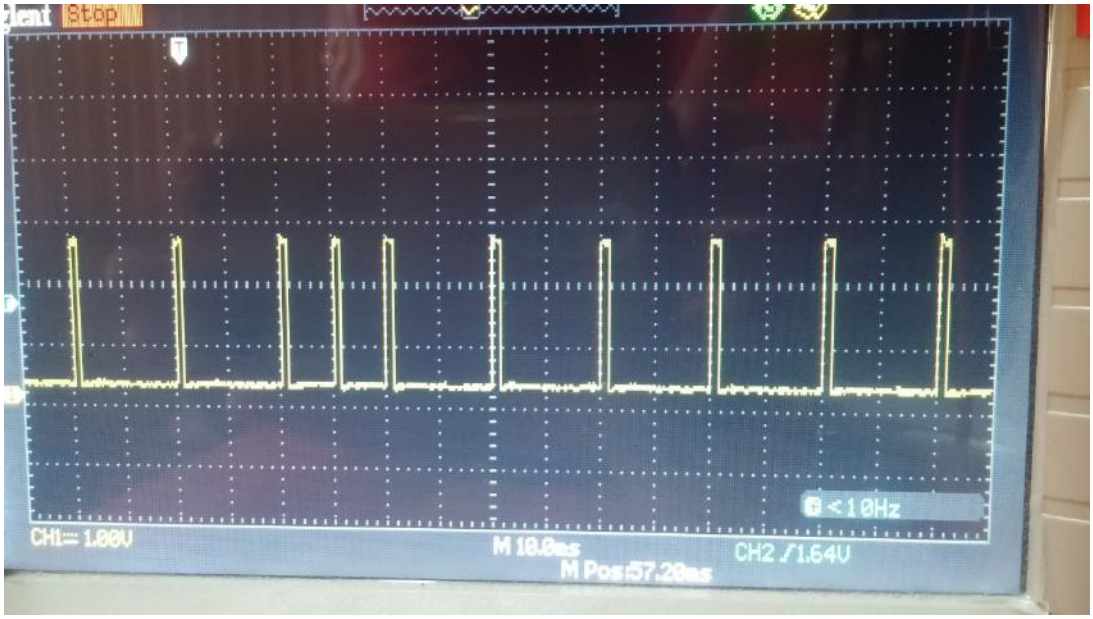
El circuito receptor consta de un filtro pasa banda para limpiar, en la mayor medida posible, de ruido la señal de interés, y un demodulador OOK (On Off Keying).
El demodulador comprende un RC en paralelo, para rescatar la envolvente, y un comparador LM311 para obtener una señal limpia.

Luego de atravesar la red de acople, mediante el transistor Q2, ingresa la señal al filtro pasa banda, el cual está centrado en la frecuencia de la portadora (120Khz) y tiene un ancho de banda de 70kHz Más adelante se detalla el diseño del filtro.
A continuación del filtro hay un seguidor de emisor, para presentar una impedancia de carga alta, mayor a la que ofrece el demodulador solamente.
Las resistencias de base R1 y R2 se calcularon de forma tal de obtener la mitad de la tensión de alimentación en la base del transistor; para lo cual la corriente que atraviesa R2 debe ser al menos 10 veces mayor que la corriente que ingresa por la base del Q2.

Teniendo en cuenta que el hfe típico del BC547 es de 110 y la resistencia de emisor en continua es 100k ohm, el valor de R2 se eligió de 100k, al igual que R1, para obtener la premisa inicial de la mitad del valor de alimentación en la base del transistor.

El demodulador de envolvente RC, se calculó teniendo en cuenta que la duración de la ráfaga es de 1ms.
Entonces Tau = RC = 1ms => si R=100k ohms => C=10nF

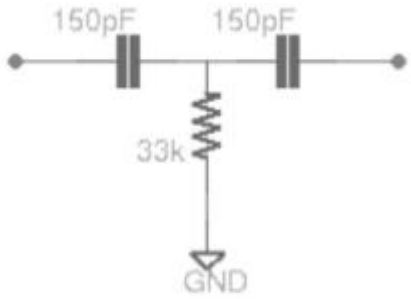
Como la señal es ASK binario, y tiene sólo dos niveles de tensión a considerar; es conveniente el uso de un comparador LM 311, muy popular, fácil de conseguir y factible de ser alimentado con fuente simple.
El LM311 funciona con valores de 5 hasta 30 volts, con alimentación simple, o con +- 15 con fuente partida.



En la figura, se puede apreciar la vista del osciloscopio de la señal recibida, a la salida del comparador LM311.

Filtro pasa banda

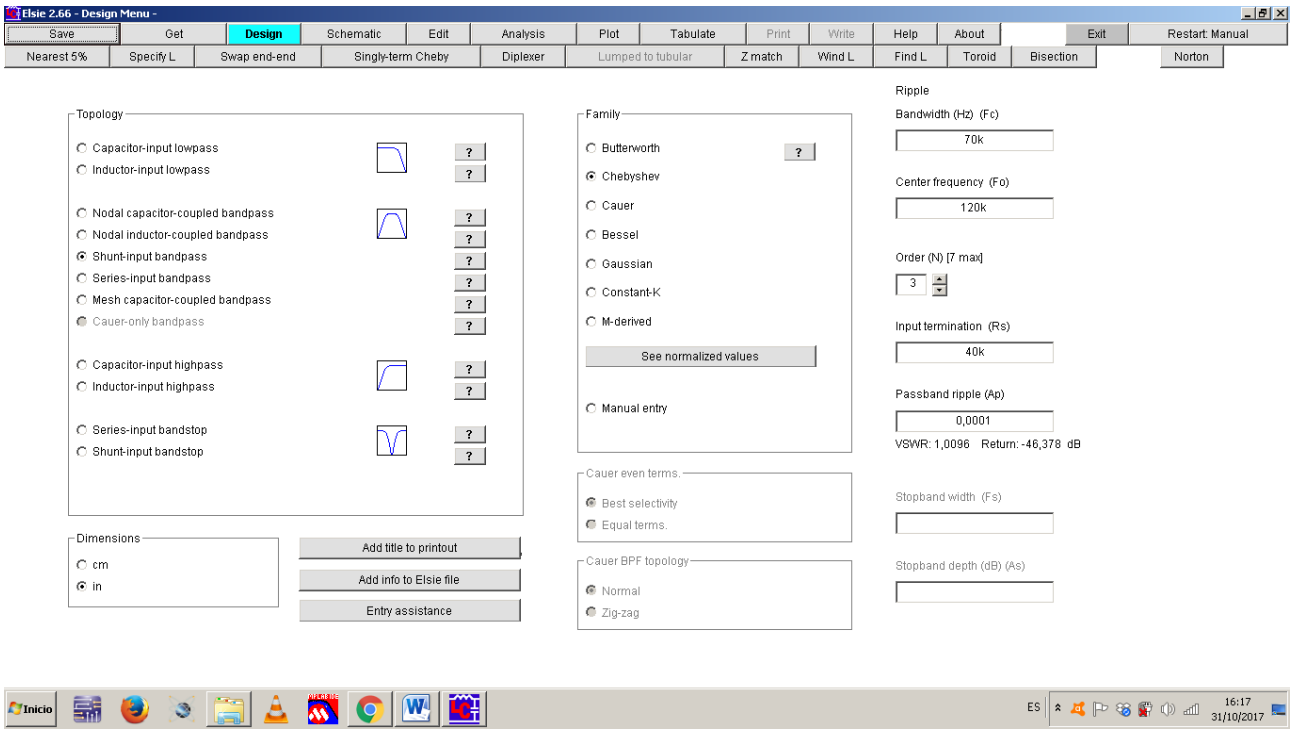
Debido al ruido presente en la línea eléctrica, se debe filtrar la señal recibida.
En la nota de aplicación AN236 de Microchip, se utilizó un filtro pasa altos de segundo orden con una frecuencia de corte de 30KHz como el que se muestra en la Figura



Filtro pasa altos de segundo orden

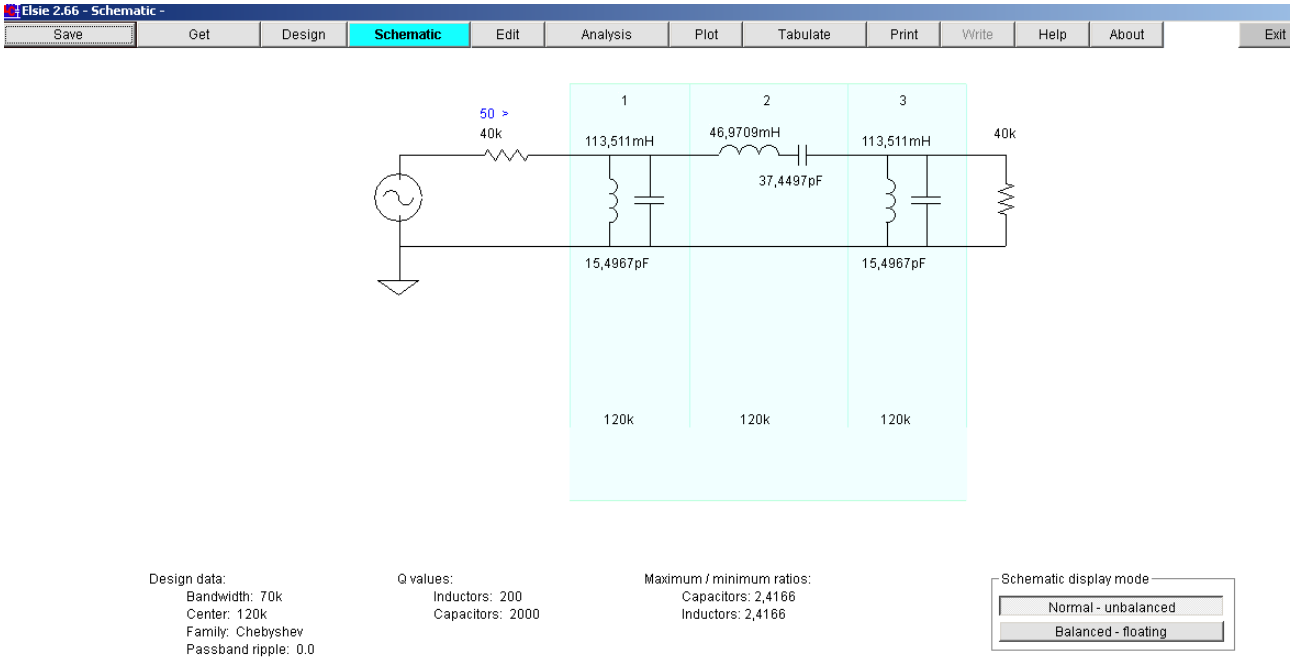
Después de realizar pruebas se llegó a la conclusión de que este filtro no era apropiado debido a que en la red de 220V existen muchas componentes de ruido de frecuencias mayores a 30KHz.
Cuando se midió la salida de dicho filtro con el osciloscopio se observó que muchas de estas componentes pasaban el filtro ingresando en las etapas posteriores al mismo, lo que originaba una deformación en el techo de los pulsos recibidos.
A su vez esta deformación causaba problemas en la decodificación del mensaje.
Por este motivo se modificó el circuito y se procedió a diseñar un filtro más selectivo.
Se estudiaron varias opciones, y se concluyó que un pasa bandas del tipo Chebyshev era lo más adecuado.
Con este tipo de filtros se puede lograr la respuesta en frecuencia deseada con un orden relativamente bajo, y el comportamiento del sistema no se ve afectado por el ripple en la banda de paso.
El filtro debía tener una frecuencia central de 120KHz y un ancho de banda lo suficientemente grande para que los pulsos pudieran pasar sin sufrir deformaciones en sus flancos. En este caso se comenzó con un filtro de tercer orden con un ancho de banda de 3dB de aproximadamente 70KHz.
En pruebas posteriores se pudo observar que cumplía perfectamente con el objetivo planteado.

Para el cálculo de los componentes del circuito se utilizó la versión para estudiantes de “Elsie”, un software para el diseño de filtros pasivos. Su uso es muy intuitivo, ya que se colocan parámetros tales como: topología, ancho de banda, orden, etc. y luego se puede visualizar tanto el circuito resultante como el gráfico de respuesta en frecuencia. Además, se pueden modificar los componentes del circuito manualmente para hacer coincidir sus valores con los disponibles comercialmente y ver que efecto causa esto en la respuesta del filtro. En la Figura 2.17 se muestra la pantalla de configuración de “Elsie”, en donde se colocan los parámetros del filtro que se desea diseñar.



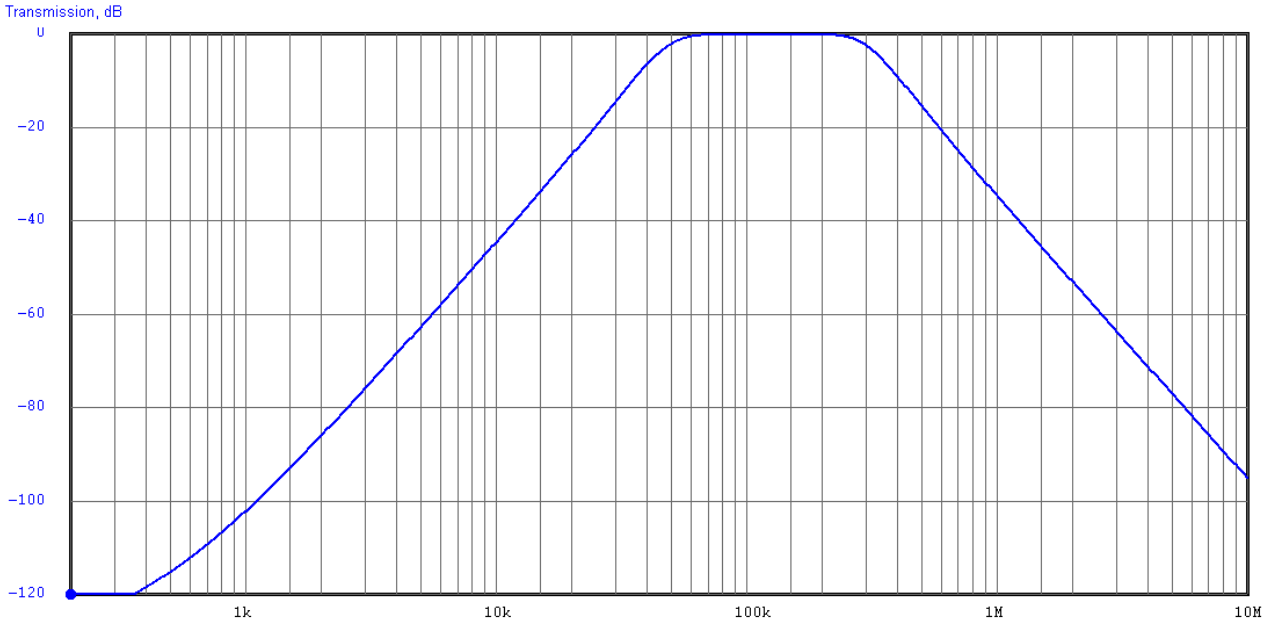
Pantalla de configuración de “Elsie”

Luego de ingresar las características del filtro en el programa se puede acceder a la solapa “schematic” que se muestra en la Figura.

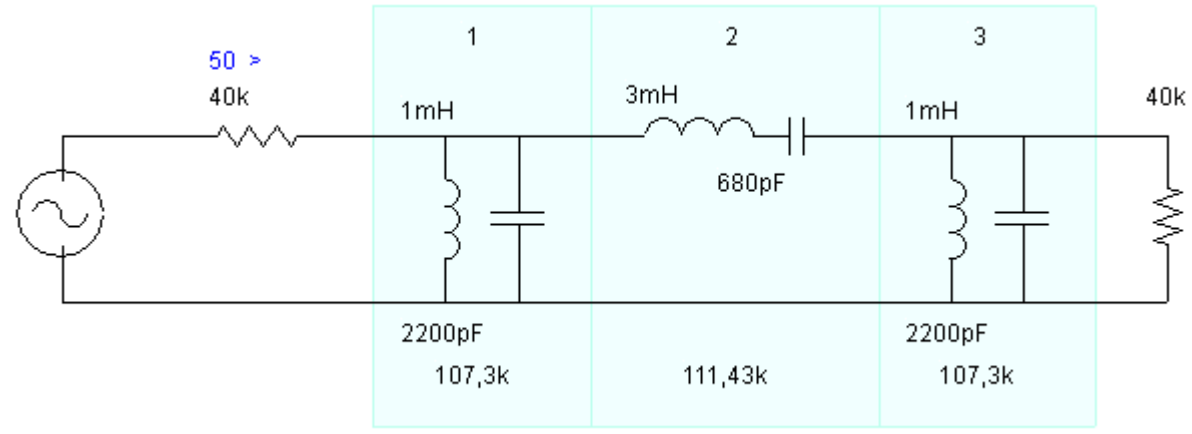


En ella se observa la primera versión del circuito a diseñar. El software calcula valores de componentes exactos. Como se puede ver en la imagen, los valores de inductancia propuestos idealmente por programa son muy elevados, 2 bobinas de 113 mHy y otro de 48 mHy.

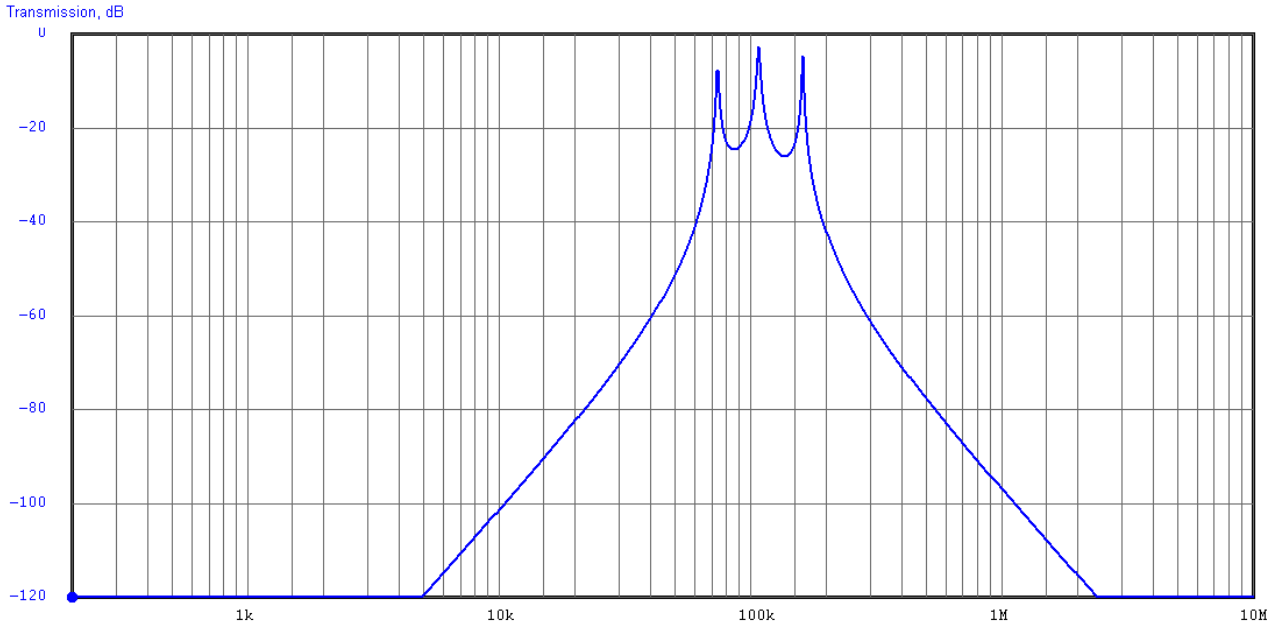
Si bien en filtro tiene la respuesta deseada, como se aprecia en la siguiente figura, dichos valores son no comerciales y el construirlos demandaría bastante tiempo y esfuerzo.



A los valores dados en primera instancia por el programa se los pueden modificar utilizando la solapa “Edit”.
Se modificaron tanto los valores de los inductores, como de los capacitores. Se adoptaron valores de 1mH y 3mH para las bobinas y de 2,2nF y 680pF para los capacitores. Esto se realizó teniendo en cuenta la resonancia de cada etapa del filtro, resaltadas para la edición, como se muestra en la figura



En la Figura se muestra el gráfico de la respuesta en frecuencia del filtro extraído de “Elise”. Se puede observar como cambio la respuesta al variar los valores de los elementos utilizados, ajustados a valores comerciales.



En la Figura se puede ver una captura de la pantalla de un osciloscopio, en donde se observan pulsos de 1ms de duración luego de pasar por el filtro pasa bandas.
Estos pulsos forman parte de un mensaje X-10, y dentro de ellos se encuentra la señal portadora de 120KHz.

Alimentación

La placa se alimenta con 8,5 volts, desde una fuente externa, de los cuales se obtienen los 3,3 volts para la alimentación del microcontrolador y los 5,6 volt para el integrado comparador LM311.

El microcontrolador de la familia PIC, 18F2550 puede ser alimentado en un rango desde 2.0 hasta 5.5 volt. Se eligió 3,3 para ser compatible con el GPIO de la raspberry pi.

Los 3,3 volt se obtienen con un circuito integrado LM 317.

El **LM317** es un regulador de tensión positivo con sólo 3 terminales y con un rango de tensiones de salida desde los 1.25 hasta 37 voltios. Las patillas son: Entrada (IN), Salida (OUT), Ajuste (ADJ). Para lograr esta variación de tensión sólo se necesita de 2 resistencias externas (una de ellas es una resistencia variable).

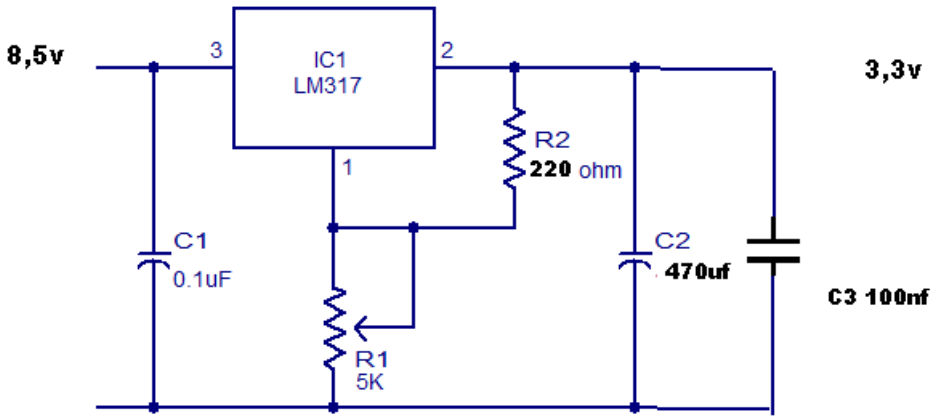
Entre sus principales características se encuentra la limitación de corriente y la protección térmica contra sobrecargas. La tensión entre la patilla ADJ y OUT es siempre de 1.25 voltios (tensión establecida internamente por el regulador) y en consecuencia la corriente que circula por la resistencia R2 es: IR2 = V / R2 = 1.25/R2. Esta misma corriente es la que circula por la resistencia R1. Entonces la tensión en R1: VR1 = IR2 x R1. Si se sustituye IR2 en la última fórmula se obtiene la siguiente ecuación: VR1 = 1.25 x R1 / R2.

Como la tensión de salida es:

•Vout = VR1 + VR2, entonces: Vout = 1.25 V. + (1.25 x R1/R2)V. Simplificando (factor común)
•Vout = 1.25 V (1 + R1/R2) V.

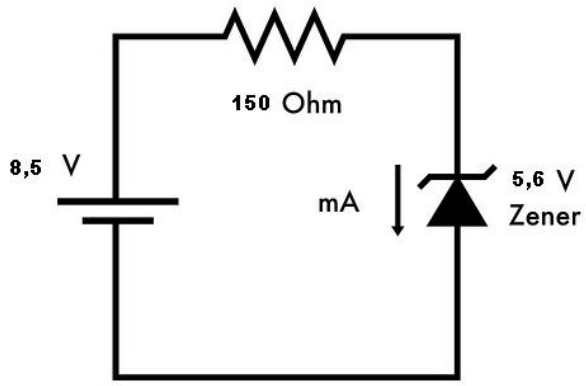
De esta última fórmula se ve claramente que si modifica R1 (resistencia variable), se modifica la tensión Vout. En la fórmula anterior se ha despreciado la corriente (IADJ) que circula entre la patilla de ajuste (ADJ) y la unión de R1 y R2.

Esta corriente se puede despreciar, tiene un valor máximo de 100 uA y permanece constante con la variación de la carga y/o de la tensión de entrada. Para mejorar la regulación la resistencia R2 se debe colocar lo más cercano posible al regulador, mientras que el terminal que se conecta a tierra de la resistencia R1 debe estar lo más cercano posible a la conexión de tierra de la carga



Vout = 1.25V (1+ (R2/R1)) + (I adj x R2)

Puesto que los 5,6 volts solo se utilizan para alimentar el integrado LM311, y para mayor simplicidad del PCB, en tamaño y ruteo de pistas, se diseñó el siguiente circuito con un diodo zener :



El diodo usado es de 1 watt.

Se calculó la resistencia limitadora de forma tal de mantener el valor de 5,6 volt

Como P = V*I = 1 watt =>I (máx) = P/V = 1/5,6 = 0,178 Amp

Para mantener el valor deseado la corriente que pase por el zener debe ser al menos el 10 ó 20 % del valor máximo de corriente.
En consecuencia R (lim) = (8,5-5,6)/0,018 =161 ohms

Se toma R (lim) = 150 ohms

4.4 Microcontrolador

Se usó un microcontrolador de la familia PIC del fabricante Microchip. El modelo 18F2550.
Es un micro que puede funcionar hasta 48 MHz, mediante un cristal externo de 20Mhz y un circuito interno con PLL. , 3 fuentes de interrupción externa, 4 módulos de timer, 2 ccp capture/compare/pwm), todo lo necesario para el proyecto
Una interrupción es usada por la señal de cruce por cero, y otra por la recepción de datos desde la Raspberry.
La frecuencia de operación es muy buena y muy superior a los tiempos del protocolo x10.

Características del micro

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 Endpoints (16 bidirectional)
- 1 Kbyte Dual Access RAM for USB
- On-Chip USB Transceiver with On-Chip Voltage Regulator
- Interface for Off-Chip USB Transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, Peripherals on
- Idle: CPU off, Peripherals on
- Sleep: CPU off, Peripherals off
- Idle mode Currents Down to 5.8 μA Typical
- Sleep mode Currents Down to 0.1 μA Typical
- Timer1 Oscillator: 1.1 μA Typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μA Typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

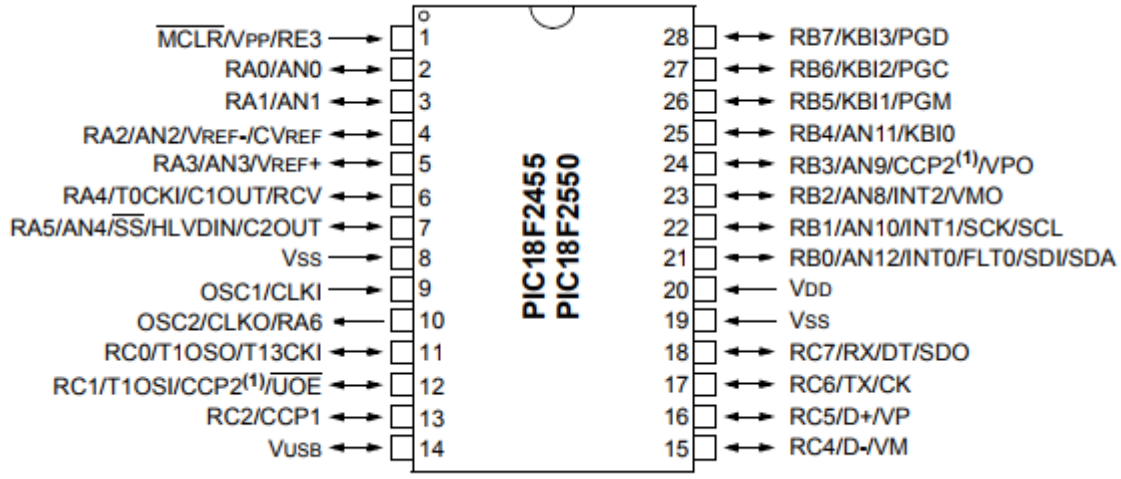
- Four Crystal modes, including High-Precision PLL for USB
- Two External Clock modes, Up to 48 MHz
- Internal Oscillator Block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary Oscillator using Timer1 @ 32 kHz
- Dual Oscillator Options allow Microcontroller and USB module to Run at Different Clock Speeds
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-Current Sink/Source: 25 mA/25 mA
- Three External Interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 5.2 ns (TCY/16)
  - Compare is 16-bit, max. resolution 83.3 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown and auto-restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module Supporting 3-Wire SPI (all 4 modes) and I^2C™ Master and Slave modes
- 10-Bit, Up to 13-Channel Analog-to-Digital Converter (A/D) module with Programmable Acquisition Time
- Dual Analog Comparators with Input Multiplexing

Special Microcontroller Features:

- C Compiler Optimized Architecture with Optional Extended Instruction Set
- 100,000 Erase/Write Cycle Enhanced Flash Program Memory Typical
- 1,000,000 Erase/Write Cycle Data EEPROM Memory Typical
- Flash/Data EEPROM Retention: > 40 Years
- Self-Programmable under Software Control
- Priority Levels for Interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) via Two Pins
- Optional Dedicated ICD/ICSP Port (44-pin, TQFP package only)
- Wide Operating Voltage Range (2.0V to 5.5V)



Se utilizó para programar el programa MPLAB, con el compilador CCS, que se instala como plugin.

MPLAB-IDE es una Plataforma de Desarrollo Integrada bajo Windows, con múltiples prestaciones, que permite escribir el programa para los PIC en lenguaje ensamblador (assembler) o en C (el compilador C se compra aparte), crear proyectos, ensamblar o compilar , simular el programa y finalmente programar el componente, si se cuenta con el programador adecuado.

MPLAB incorpora todas las utilidades necesarias para la realización de cualquier proyecto y, para los que no dispongan de un emulador, el programa permite editar el archivo fuente en lenguaje ensamblador de nuestro proyecto, además de ensamblarlo y simularlo en pantalla, pudiendo ejecutarlo posteriormente en modo paso a paso y ver como evolucionarían de forma real tanto sus registros internos, la memoria RAM y/o EEPROM de usuario como la memoria de programa, según se fueran ejecutando las instrucciones. Además el entorno que se utiliza es el mismo que si se estuviera utilizando un emulador.

Vamos a ver ahora algunas características que presenta el compilador PCW CCS y que hacen de él una buena opción para elegirlo como compilador de C para programar Microcontroladores PIC.

Algunas de esas características son:

- Al compilar genera un código máquina muy compacto y eficiente.
- Se integra perfectamente con MPLAB y otros simuladores/emuladores como PROTEUS para el proceso de depuración.
- Incluye una biblioteca muy completa de funciones pre compiladas para el acceso al hardware de los dispositivos (entrada/salida, temporizaciones, conversor A/D, transmisión RS-232, bus I2C…., etc.
- Incorpora drivers para dispositivos externos, tales como pantallas LCD, teclados numéricos, memorias EEPROM, conversores A/D, relojes en tiempo real, etc.(los drivers son pequeños programas que sirven de interfaz entre los dispositivos hardware y nuestro programa).

- Permite insertar partes de código directamente en Ensamblador, manteniendo otras partes del programa en C.

Características del lenguaje C para este compilador

El lenguaje C estándar es independiente de cualquier plataforma. Sin embargo, para la programación de microcontroladores es necesario disponer de determinados comandos que se refieran a partes específicas de su hardware, como el acceso a memoria, temporizadores, etc. Por este motivo, además de los comandos, funciones y datos del lenguaje ANSI C, el compilador PCW incluye bibliotecas que incorporan determinados comandos que no son estándar, sino específicos para la familia de microcontroladores PIC. Éstos son básicamente de dos tipos: directivas del preprocesador y funciones pre compiladas

Directivas del preprocesador.

El compilador PCW dispone de 7 tipos básicos de directivas:

- Directivas derivadas del estándar de C, que permiten, entre otras funciones, un control básico del código y del flujo en el proceso de compilación:

#DEFINE
#ELIF
#ELSE
#ENDIF
#ERROR
#IF
#IFDEF
#IFNDEF
#INCLUDE
#LIST
#NOLIST
#PRAGMA
#UNDEF

- Directivas asociadas a las bibliotecas pre compiladas, que proporcionan al compilador información relacionada con estas bibliotecas:

#USE DELAY
#USE FAST_IO
#USE FIXED_IO
#USE I2C
#USE RS232
#USE STANDARD_IO

- Directivas relacionadas con la especificación del dispositivo, por un lado, para definir los mapas de memoria y el juego de instrucciones, y por otro, incluir información necesaria para la programación del dispositivo en los ficheros de salida de la compilación:

#DEVICE
#ID
#FUSES
#TYPE

- Directivas de cualificación de funciones, para identificar características especiales de una función:

#INLINE
#INT_DEFAULT
#INT_GLOBAL
#INT_xxxxx
#SEPARATE

- Directivas de control del compilador, para definir opciones referidas a la compilación del código del programa:

#CASE
#OPT
#ORG
#PRIORITY

- Directivas de control de la memoria del microcontrolador, para gestionar y reservar el uso de determinadas zonas de memoria para variables:

#ASM
#BIT
#BYTE
#ENDASM
#LOCATE
#RESERVE
#ROM
#ZERO_RAM

- Identificadores predefinidos. Todas las directivas citadas hasta ahora, son comandos destinados a ser interpretados por el compilador, no por el microcontrolador. Dentro del término genérico de directiva se incluyen, además de estos comandos, unas variables que contienen información sobre el proceso de compilación. Estas variables son lo que se denominan identificadores predefinidos del compilador:

__DATE__
__DEVICE__
__PCB__
__PCH__
__PCM__

En un programa, las directivas se reconocen fácilmente porque comienzan por el símbolo #, mientras que los identificadores empiezan y acaban por doble subrayado (__).

Funciones pre compiladas.

Se puede facilitar considerablemente la tarea de programación si no es necesario construir por nosotros mismos aquellas funciones que son de utilización más frecuente, como leer la entrada de un teclado o imprimir un determinado mensaje en una pantalla LCD conectada como salida. Existen funciones en C incluidas en el compilador PCW para manejar los diferentes recursos del microcontrolador, desde el bus I2C hasta el conversor A/D.

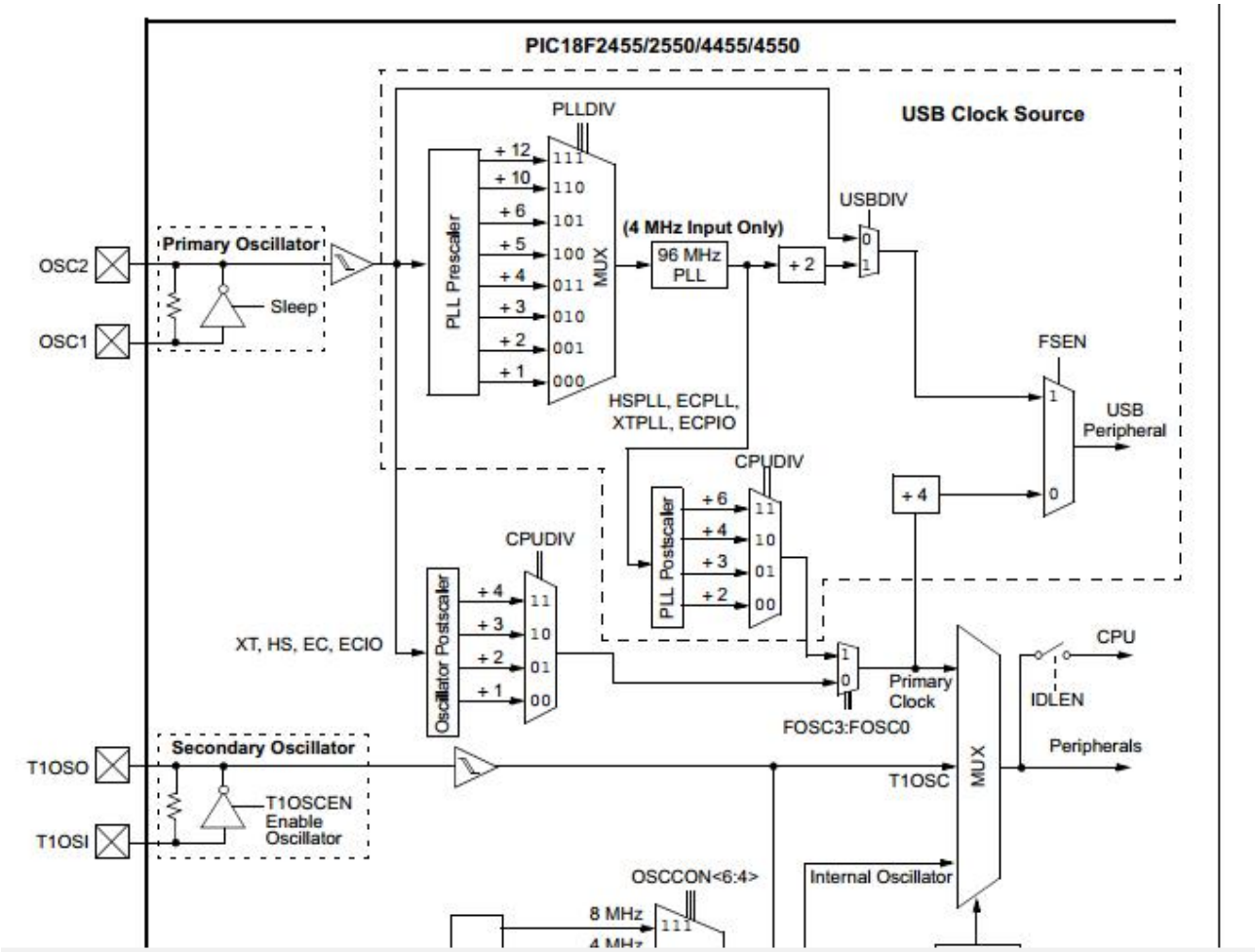
Configuración del clock

El chip microcontrolador 18F2550, cuenta con un sofisticado hardware que permite operar sus circuitos a través de un oscilador tanto interno como externo, a elección del usuario. Dicho hardware incluye también un circuito multiplicador de frecuencia PLL (Phase Locked Loop).
La velocidad de operación del microcontrolador 18F2550, se encuentra determinada no solamente por la frecuencia de su oscilador (interno ó externo), sino también por la configuración de su PLL, a través de la inicialización de sus registros, lo cual da al usuario una gran gama de opciones de operación.

A 48 Mhz, que es la velocidad máxima a la que puede operar el 18F2550, el período de ejecución de cada instrucción, es de 83.3 nanosegundos, considerando que cada instrucción se ejecuta en 4 ciclos de reloj. Esta fue la configuración elegida.

Se utilizó un cristal externo de 20 Mhz. Sin embargo, gracias a su circuito multiplicador de frecuencia PLL, (Phase Locked Loop), su velocidad efectiva de operación es de 48 Mhz..

. En la figura de abajo se muestra el diagrama de bloques del sistema de generación del reloj, tanto para el CPU como para los circuitos del puerto USB.



El 18F2550 recibe la señal de su cristal externo y usa un preescalador que divide su frecuencia (dependiendo de sus registros de configuración), entre alguno de los siguientes valores: 1, 2, 3, 4, 5, 6, 10, ó 12. La entrada al circuito PLL debe ser siempre de 4 Mhz. En el caso de funcionamiento a 20 Mhz, la división del preescalador es entre 5.

Estos 4 Mhz pasan a través del PLL y generan a su salida 96 Mhz, los cuales a su vez se dividen entre 2 para dar finalmente la señal de 48 Mhz, con la cual se obtiene un funcionamiento del puerto USB a la velocidad máxima de 12 Mbps. Esta velocidad permite la comunicación del puerto USB del microcontrolador 18F2550, con cualquier Host.

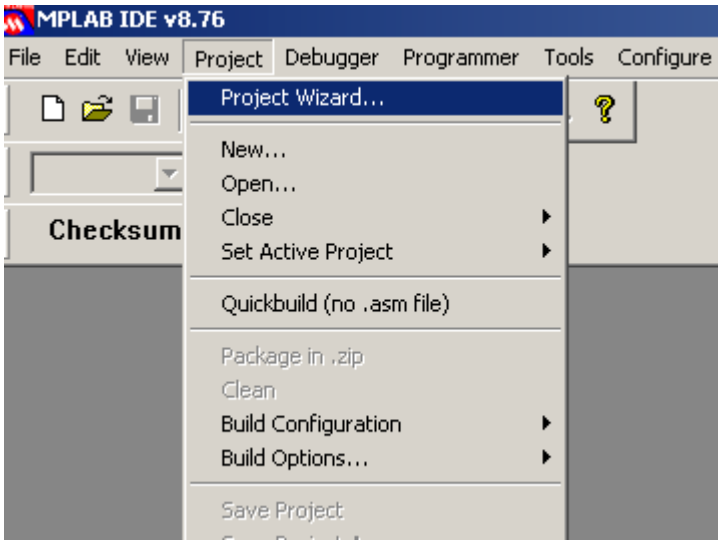
Para que esto sea así, se le indica al compilador en C, con las siguientes directivas:

```
#FUSES HSPLL              //High Speed Crystal/Resonator with PLL enabled
#FUSES PLL5               //Divide By 5(20MHz oscillator input)
#FUSES CPUDIV1            //System Clock by 1
#use delay(clock=48000000)
```

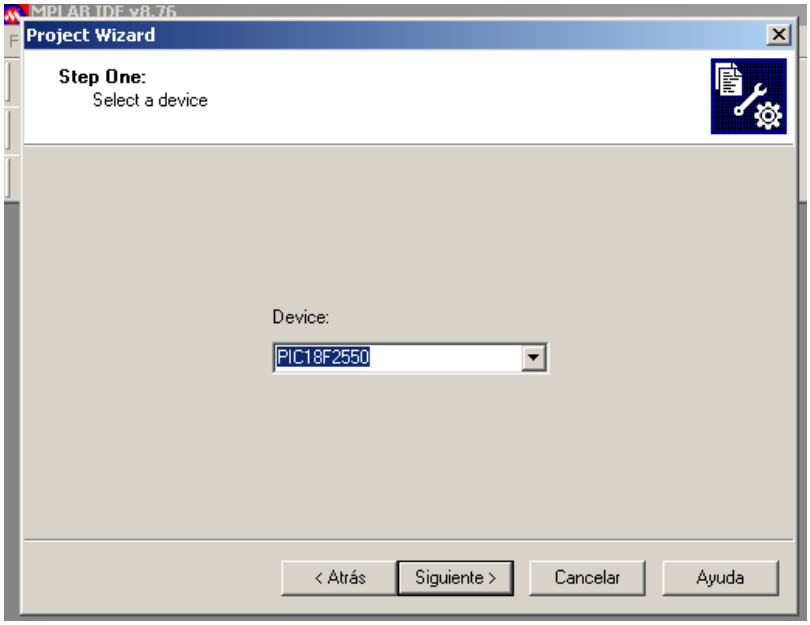
Creación del proyecto en MPLAB

A continuación se ilustra el procedimiento para crear un proyecto desde cero en el MPLAB, eligiendo el compilador CSS.

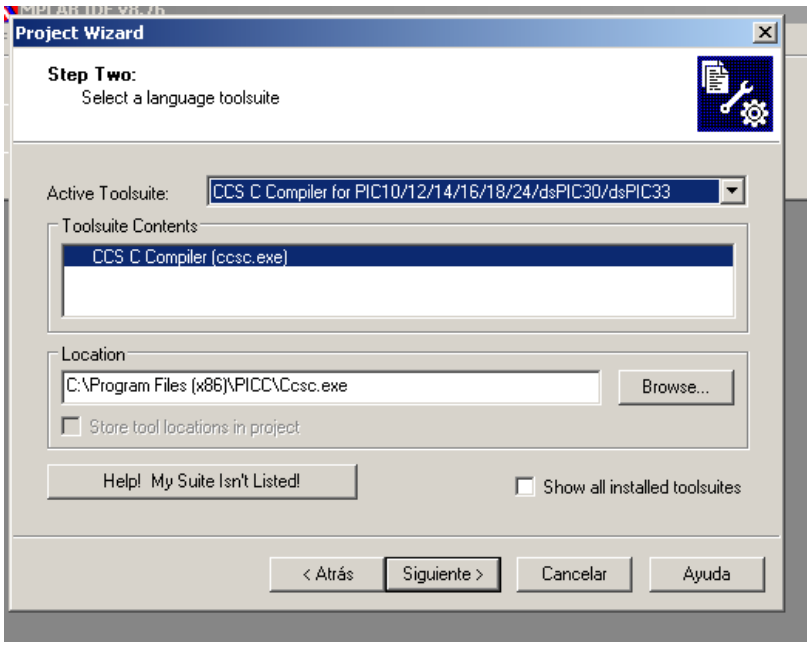
1er paso se selecciona ‘project wizard’ como se ve en la figura de abajo:



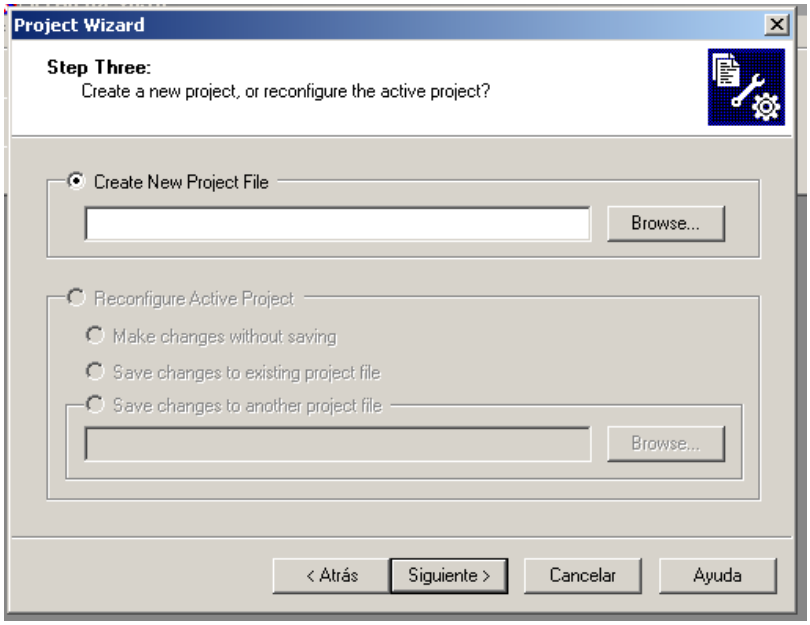
2 do paso se selecciona el modelo de microcontrolador a utilizar:



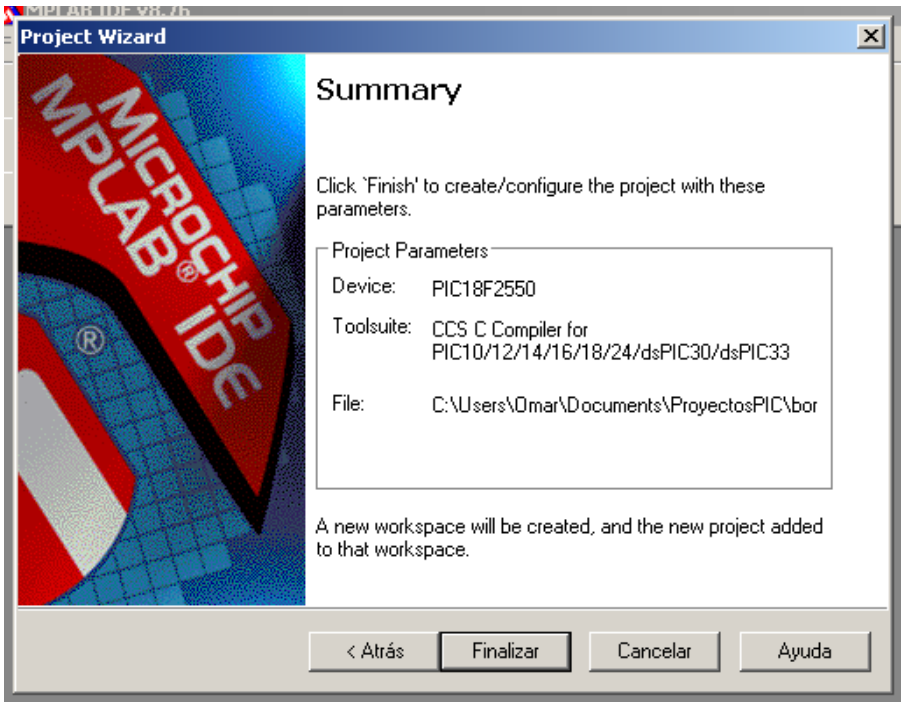
3er paso: se escoge el compilador:



4to paso: se crea o busca una carpeta para contener los archivos del proyecto:



Y así finalmente se termina de crear el proyecto nuevo.



Antes de compilar, el proyecto constaba solo de dos archivos, el de que lleva las directivas para configurar los fusibles del micro según el comportamiento deseado, y el main.c que contiene el código en C, con la programación en sí.

Configuración de fusibles, en fuses,h :

```
#include <18F2550.h>
#device adc=8

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES HSPLL                    //High Speed Crystal/Resonator with PLL enabled
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOBROWNOUT               //No brownout reset
#FUSES BORV20                   //Brownout reset at 2.0V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES NODEBUG                  //No Debug mode for ICD
#FUSES NOLVP                    //No low voltage prgming, B3(PIC16) or B5(PIC18) used for I/O
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES IESO                     //Internal External Switch Over mode enabled
#FUSES FCMEN                    //Fail-safe clock monitor enabled
#FUSES PBADEN                   //PORTB pins are configured as analog input channels on RESET
#FUSES NOWRTC                   //configuration not registers write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads
#FUSES NOCPB                    //No Boot Block code protection
```

```
#FUSES MCLR                 //Master Clear pin enabled
#FUSES LPT1OSC              //Timer1 configured for low-power operation
#FUSES NOXINST              //Extended set extension and Indexed Addressing mode disabled (Legacy
mode)
#FUSES PLL5                 //Divide By 5(20MHz oscillator input)
#FUSES CPUDIV1              //System Clock by 1
#FUSES USBDIV               //USB clock source comes from PLL divide by 2
#FUSES VREGEN               //USB voltage regulator enabled

#use delay(clock=48000000)
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)

#PRIORITY int_EXT ,int_EXT2
```

Código de main.c

El programa principal funciona por interrupciones y el empleo de variables que definen mediante el uso de 'IF' que parte del código se ejecuta.

Luego de incluir librerías, se definen variables y funciones a ser llamadas desde las interrupciones. Luego, a continuación está el código de las interrupciones; son tres:

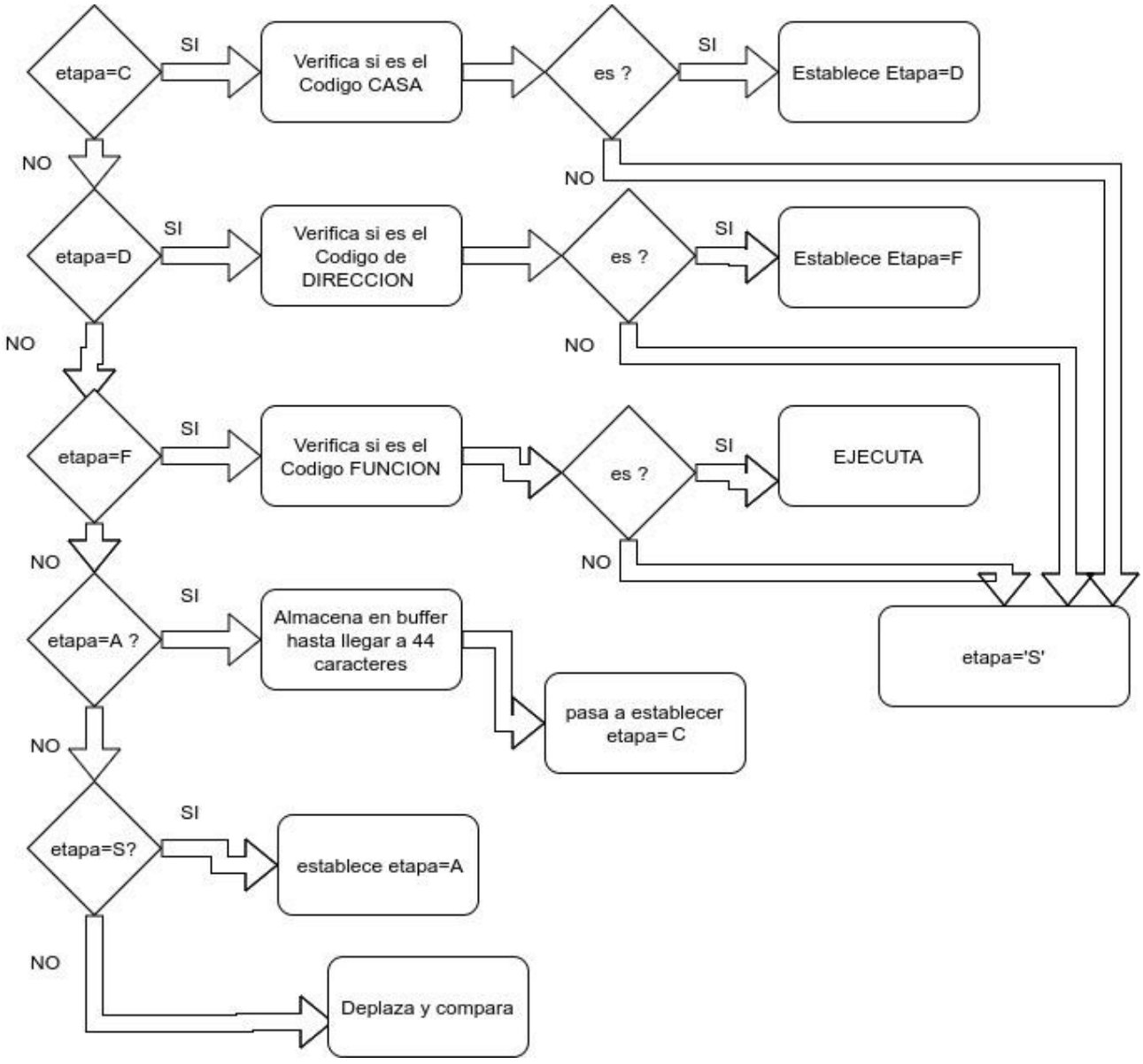
* EXT para la detección de cruce por cero, en el pin 21.
* EXT2 para la detección de datos enviadas desde la Raspberry Pi, en el pin 23.
* TIMER0 para el re envío de la trama si no llega el ACK.

En la interrupción por cruce por cero, hay una variable llamada 'modo', que divide el código en dos partes, una a ser ejecutada en caso de estar en modo ='T' (Transmisión), y el otro en en caso de estar esperando un código de start (modo ='R', Recepción).

Dentro del modo Recepción hay diferentes etapas, que se separan por una variable del mismo nombre.

Con cada interrupción de cruce por cero, si el modo está en 'R' entra al siguiente flujo:
Primeramente compara la etapa, que por default está en S (Start), etapa a la que se vuelve recurrente mente. Ahí se espera por el reconocimiento del código de start (1110).
Si se lo reconoce, se pasa a la siguiente etapa, que es la de almacenamiento; se agregan al buffer de recepción uno a uno, incrementando el índice, hasta que este llegue a 44.
Luego se pasa a la etapa='C' (código Casa), donde se verifica que el código asignado previamente al dispositivo, sea igual a la parte de lo almacenado en buffer que corresponde al código casa.
Si hay coincidencia se pasa a la siguiente etapa, asignando a la variable etapa el valor D; en caso contrario se le asigna el valor S, para volver a esperar el start.
En la siguiente etapa, se comprueba el código de dispositivo, para pasar en caso afirmativo a la etapa de función (etapa='F') o volver a esperar el start.

Similarmente funciona la última etapa, salvo que se compara el código con el de varias funciones almacenadas, al haber una coincidencia se ejecuta la parte del código adecuada.
Luego se pone en modo T, y envía el ACK, hacia la placa que oficia de enlace con el servidor.




```
#include "C:\Users\Omar\Documents\ProyectosPIC\temporizador\fuses.h"
  #include <stdio.h>
  #include <stddef.h>
  #include <stdlib.h>
  #include <stdlibm.h>
  #include <string.h>


int8 ack2[4];
int8 aux=0;
int8 * buf[5];
```

```
int8 buffer[14];  // buffer de rc desde la R-pi , por spi
int8 buffer2[44];  // buffer de rc desde el dispositivo
int8 * cmd[11];
int8 codigo[22]; //buffer de tx .(5+4)x2+4
int8 codigo2[22]; //buffer de tx .(5+4)x2+4
int8 contador = 0 ;
long cont=0;
int8 data;
char * etapa = 'S';
int funcion =0;
int8 indice = 0;
char  * modo = 'R'; // variable para distinguir entre modos de rc y tx
int8 retx_nro=0;
int8 rc ;//,p=4 ,s=0;
int8 send = 0;
int1 val = 0 ;
int1 valor = 0 ;
int8 vez = 0;
int16 x=0;


void ack_rpi()   //tx a 100us , o sea 10khz
{
        int8 j=0 ;
        while(j<4 )
  {   j++;
    output_bit( PIN_B4, 1); // pin 25
    delay_ms(1);
    output_bit( PIN_B4, 0); // pin 25
    delay_ms(1);
  }
}
void ack_rpi_2()   //tx a 100us , o sea 10khz
{
        int8 j=0 ;
        while(j<4 )
  {
   if(ack2[j]==1)
                {
                uno_rpi();
        }
        if (ack2[j]==0)
                {
        cero_rpi();
                }
      j++;

        }

}
```

La función arma_codigo() transforma la información recibida desde la Raspberry , para acomodarla a la forma requerida por el protocolo x10.
La información original estaba compuesta por código casa + código de dirección + código de función .Esto se dividió en dos tramas x10: codigo y codigo2. Cado uno almacenado en un vector.
El primer código se conforma de:

Código de start + cod. Casa + cod. Direccionamiento +código de start + cod. Casa + cod. Direccionamiento.

Mientras que el segundo, llamado codigo2 se compone de:

Código de start + cod. Casa + cod. Función +código de start + cod. Casa + cod. Función.

Donde cada cero recibido de la Raspberry , ahora son dos caracteres ; 0 y 1 , respectivamente.
A la vez que los unos se convirtieron en 1 y 0.

```
void arma_codigo()
{
int8 i;
int8 p=4;
codigo[0]=1;
codigo[1]=1;
codigo[2]=1;
codigo[3]=0;
for (i=0;i<4;i++)  //los 1eros 4 caracteres son el código de casa ,común a ambos pares de bloque.
    {
  if (buffer[i]==1)
     {
     codigo[p]=1;
           codigo2[p]=1;
     p++;
     codigo[p]=0;
           codigo2[p]=0;
     p++;
     }
  if (buffer[i]==0)
     {
     codigo[p]=0;
           codigo2[p]=0;
     p++;
     codigo[p]=1;
           codigo2[p]=1;
     p++;
     }
   }
```

```
for (i=4;i<9;i++)  //los siguientes 5 caracteres son el número de modulo.
    {
   if (buffer[i]==1)
      {
      codigo[p]=1;
      p++;
      codigo[p]=0;
      p++;
      }
   if (buffer[i]==0)
      {
      codigo[p]=0;
      p++;
      codigo[p]=1;
      p++;
      }
    }
p=12;
codigo2[0]=1;
codigo2[1]=1;
codigo2[2]=1;
codigo2[3]=0;
for (i=9;i<14;i++)  //los últimos 5 caracteres recibidos de la r-pi son el código de comando.
    {
   if (buffer[i]==1)
      {
      codigo2[p]=1;
      p++;
      codigo2[p]=0;
      p++;
      }
   if (buffer[i]==0)
      {
      codigo2[p]=0;
      p++;
      codigo2[p]=1;
      p++;
      }
    }
}// ***************************************************


int1 chequear()
{   int8 j ;
    val = 1 ;
   for (j=0;j<21;j++)
   {
      if(buffer2[j]!= buffer2[j+22])
         { val=0;
                        }
```

```
          }
  return(val);
}

int1 comparar(int8 pi,int8 pf)
{   int8 i ,valor=1;
  int8 k=0;
  for (i=pi;i<pf;i++)
  {
    if(buffer2[i]!= cmd[k])
       { valor=0;
       }
         k++;
}

 delay_us(2);
//   }
  return(valor);
}

void modula(data)
{
if (data == 1)
          {
          output_bit( PIN_A4,1);
          delay_ms(1);
          output_bit( PIN_A4,0);
          }
if (data == 0)
          {
  //salida en cero.
          }
}

#int_EXT2
void  EXT2_isr(void) //pin 23
{
```

Interrupción que indica detección de un mensaje desde la Raspberry Pi. Cuando se detecta un flanco descendente en el pin 23, se lee el dato presente en la entrada B3, el cual se almacena en el buffer, previamente definido, y se incrementa el índice.
Cuando se guardaron en buffer 14 caracteres, se resetea el índice, se llama la función que arma el código según la forma que corresponde al de una trama x10, y se establece el modo transmisión.

```
    output_bit( PIN_A1, 1); //enciendo un led ,pin 3
          buffer[indice] = input_state(pin_B3); //pin 24
    indice ++ ;
          if (indice  == 14 )
      {
                output_bit( PIN_A1, 0);
```

```
                indice = 0;
                arma_codigo();
                modo='T';
                send = 0;
                contador=0;
                retx_nro=0;
                enable_interrupts(INT_TIMER0);
        //      enable_interrupts(INT_EXT2);
      }
}
#int_EXT
void  EXT_isr(void)  //pin 21
{
if (modo == 'R')     // se utiliza la Recepción  para confirmar el ACK.
{
 output_bit( PIN_A3,1);
  while(x<400)
  {
        if( input(PIN_A5) )
                aux ++;
  x++;
  }
  if (aux > 30)
    rc=0b1;
  else
    rc=0b0;
        x = 0 ;
        aux = 0 ;

  if( etapa == 'C') // se guarda en buffer lo que se recibe y se lo compara para ver si el mensaje fue dirigido a este dispositivo.
  {
                cmd[0] = 1;
                cmd[1] = 0;
                cmd[2] = 1;
                cmd[3] = 0;
                cmd[4] = 1;
                cmd[5] = 0;
                cmd[6] = 1;
                cmd[7] = 0;

        if(comparar(4,11)) // si el código de Casa es correcto, paso a la siguiente etapa: ver el código de dispositivo
        {   if(!(funcion)){
                                        etapa = 'D';}
                                else{
                                        etapa = 'F';}
        }
        else{
```

```
                                etapa = 'S'; //sino se reconoce el cod ,se vuelve a esperar el cod de
start.
                                indice=0;
                        }
    }
  if((etapa == 'D')&&(!(funcion))) // se guarda en buffer lo que se recibe y se lo compara para ver si
el mensaje fue dirigido a este dispositivo.
   {
                cmd[0] = 1;
                cmd[1] = 0;
                cmd[2] = 1;
                cmd[3] = 0;
                cmd[4] = 1;
                cmd[5] = 0;
                cmd[6] = 1;
                cmd[7] = 0;
                cmd[8] = 1;
                cmd[9] = 0;
        if(comparar(12,21)) // si el código es el de este dispositivo, paso a la siguiente etapa: ver el
código de función
        {
                                funcion = 1;
                                etapa = 'S';
                                indice=0;
        }
        else{
                                etapa = 'S'; //sino se reconoce el código como de este dispositivo se
vuelve a esperar el cod de start.
                                indice=0;
                        }
    }
if(( etapa=='F')&&(funcion)) // acá se pueden agregar todas las funciones necesarias según el
dispositivo.
   {            // 10011
        cmd[0] = 1;
                cmd[1] = 0;
                cmd[2] = 0;
                cmd[3] = 1;
                cmd[4] = 0;
                cmd[5] = 1;
                cmd[6] = 1;
                cmd[7] = 0;
                cmd[8] = 1;
                cmd[9] = 0;
     if(comparar(12,21)) // función 1 , prender
        {
                        ack_rpi() ;
                        retx_nro=9; //esto es asi para desactivar el envío NACK en el còd. de int. del
timer0.
                        output_bit( PIN_A2,1);
```

```
                delay_ms(400);
                        output_bit( PIN_A2,0);
        }
            etapa = 'S';
                 funcion=0;
                indice=0;
   }

if (etapa == 'A') // almacena en buffer
        {
                buffer2[indice]=rc;
                indice++;
                if (indice > 43) // de 0 a 43 son 44
                        {
                        if(chequear())
                                {
                                etapa='C';
                                 }
                        else {etapa = 'S';
                                                indice=0;
                                }
                        }

        }

if( etapa == 'S') // Se espera el código de start .Cuando se lo recibe , se pasa a guardar en buffer los caracteres a medida que entran.
   {
            buffer2[0]=buffer2[1];
            buffer2[1]=buffer2[2];
            buffer2[2]=buffer2[3];
            buffer2[3]=rc;

        cmd[0] = 1;
        cmd[1] = 1;
        cmd[2] = 1;
        cmd[3] = 0;

     if(comparar(0,4)) //b3b2b1b0
        {
                        delay_ms(1);
                         etapa = 'A';
                        indice = 4 ;
                }
   }
} // fin modo Rx
```

En el modo de transmisión (modo=’T’), se usa una variable llamada send, para contar la cantidad de veces que se envió el código.

En un principio send tiene valor cero. Por lo que con cada cruce por cero entra al primer if, enviando un carácter por cruce. Cuando se llega a 22 caracteres enviados, se incrementa el contador send y se resetea el índice para volver a enviar el primer código.
Luego de enviarlo por segunda vez, se pasa a send =2, donde se cuentan 6 cruces por cero antes de incrementar la variable send y pasar al tercer if, cuya condición es send>2.
Similarmente al primer código, se envía dos veces el segundo.
Tras terminar de enviar el segundo código, entra al ante ultimo if, donde se establece el modo de recepción, a la espera del ACK, y se da inicio al temporizador.
La llegada del ACK, anula la cuenta del temporizador. Caso contrario, se dispara la interrupción del temporizador y se vuelve a enviar el mensaje.

```
if(modo=='T')  // con cada cruce por cero se lee un carácter del mensaje a enviar y se lo modula
{
output_bit( PIN_A3,0);
if(send<2){ // para send = 0 y 1 envía el cod 1
                    data=codigo[indice];
                    modula(data);
                    indice++;
                    if(indice > 21) { send++ ;
                                                        indice = 0;
                                                    }
}
if(send==2){  // para send = 2 espera 6 cruces por cero
                    contador++; //acá solo se incrementa esta var
}
if(send>2){  // para send = 3 y 4  envía el cod2
                    data=codigo2[indice];
                    modula(data);
                    indice++;
                    if(indice > 21) { send++ ;
                                                        indice = 0;
                                                    }
                    }
if(send==5){  //luego de enviar el código 2 por segunda vez (send =4) entra acá.
                    //output_bit( PIN_C4,1);
                    set_timer0(1);
                    modo='R';
                    send=0;
                    contador=0;
                    indice=0;
}
if(contador==6){send++;

                                    contador=0;//si no estaba esto incrementaba
                                            } //al principio está en cero, por lo que no incrementa send
}

}
```

```
#int_TIMER0
void  TIMER0_isr(void)
{
                if(retx_nro<=3){   //si ya se envió 4 veces, no se re envía , manda un ack2
                                indice=0;
                                modo='T';
                                retx_nro++;
                }
                else if(retx_nro!=9){    //se setea en 9 solo si se recibió el ack
                                        //enviar_ack2(); // este es el envío del NO ack
                                        modo="R";
                        disable_interrupts(INT_TIMER0);
                                        ack_rpi_2();
                                        retx_nro=0;
                                        }
}


void main()
{

  setup_adc_ports(NO_ANALOGS|VSS_VDD);
  setup_adc(ADC_OFF);
  setup_spi(SPI_SS_DISABLED);
  setup_wdt(WDT_OFF);
  //setup_timer_0(RTCC_INTERNAL);
  setup_timer_0(RTCC_INTERNAL|RTCC_DIV_256);
  setup_timer_1(T1_DISABLED);
  setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);
  enable_interrupts(INT_EXT);
  enable_interrupts(INT_EXT2);
  ext_int_edge( 2, H_TO_L);
  ext_int_edge( H_TO_L);

  //enable_interrupts(INT_TIMER0);
  enable_interrupts(GLOBAL);

output_bit( PIN_A2,0);
        output_bit( PIN_A4,0);
output_bit( PIN_C4,1);
cmd[0]=1;
cmd[1]=1;
cmd[2]=1;
cmd[3]=1;

ack2[0]=1;
```

```
ack2[1]=1;
ack2[2]=1;
ack2[3]=1;
  // TODO: USER CODE!!
while (1){

}
}
```

5-Conclusiones y posibles mejoras

El sistema funciona como se esperaba, y en las pruebas realizadas, nunca hubo necesidad de retransmitir el mensaje. El ruido se puede limpiar, ajustando el nivel de referencia con potenciómetro del circuito de recepción; quedando la sensibilidad del comparador LM311 como límite funcional.
El filtro de recepción no tiene una respuesta tan plana en la banda de paso como sería lo ideal, pero no es un factor crítico. Se lo podría mejorar cambiando los valores de los inductores y capacitores. Simulando por computadora, se comprobó que un inductor de 27mHy en lugar de los 3 de 1mhy empleados, daría una respuesta prácticamente ideal en la banda de paso.

Como posibles mejoras:

*Uso de un reloj de tiempo real, para evitar ante un apagón de luz, la des configuración de la fecha y hora de la Raspberry Pi, con el consiguiente desfasaje de lo agendado como tareas, y la realidad

*Simplificación del diseño del hardware. Se puede usar el PWM del microcontrolador en lugar del NE555, y utilizar una versión más básica para el amplificador de salida, con un solo Darlington.

*Se pueden agregar llaves multi punto para establecer el código casa y el código de dispositivo desde el hardware, en lugar de tenerlos preestablecidos desde la programación.

*Incorporación de una fuente de la alimentación integrada al circuito. Posiblemente una sin transformador, pero con algunas medidas de seguridad; un fusible y un varistor.

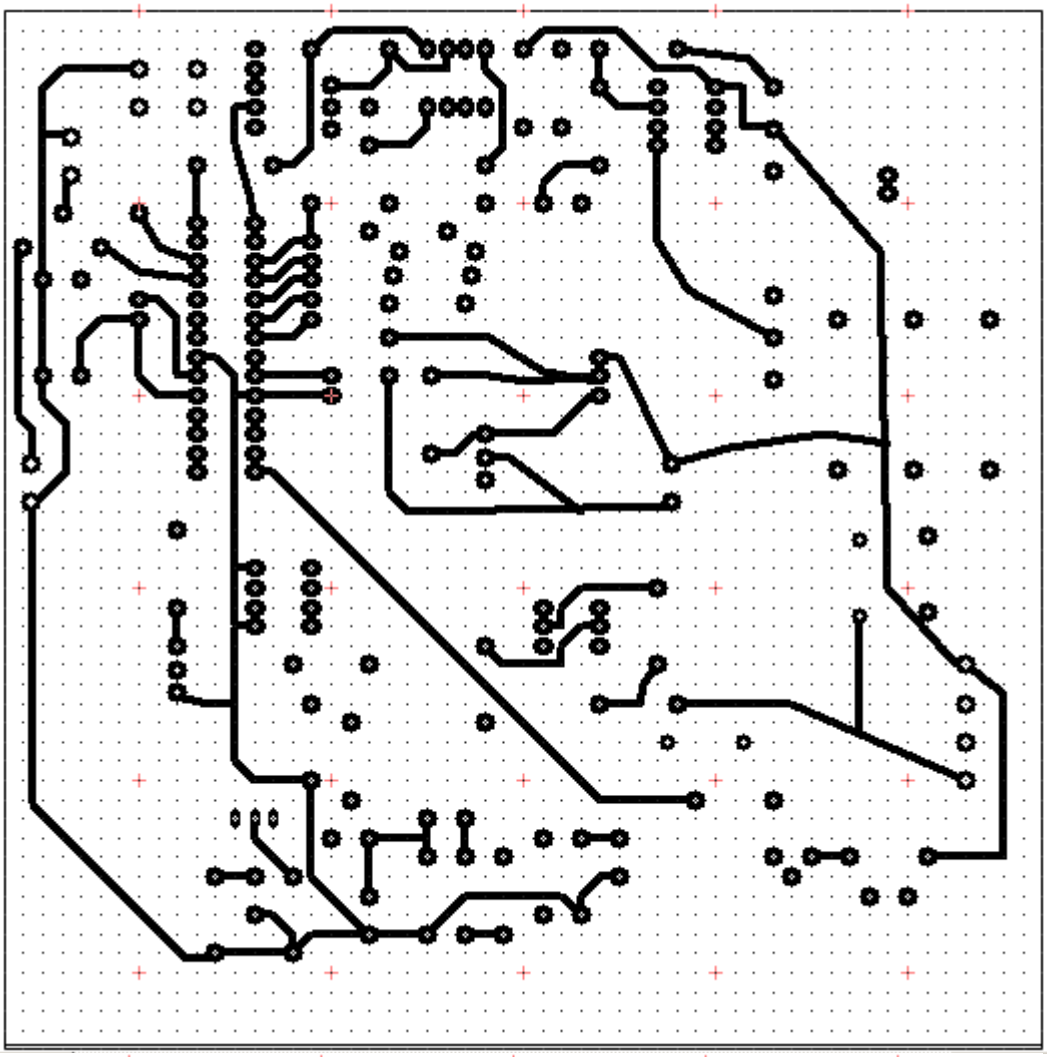
*Mejorar el aislamiento entre los bobinados primarios y secundarios del transformador mediante una película de poliéster.

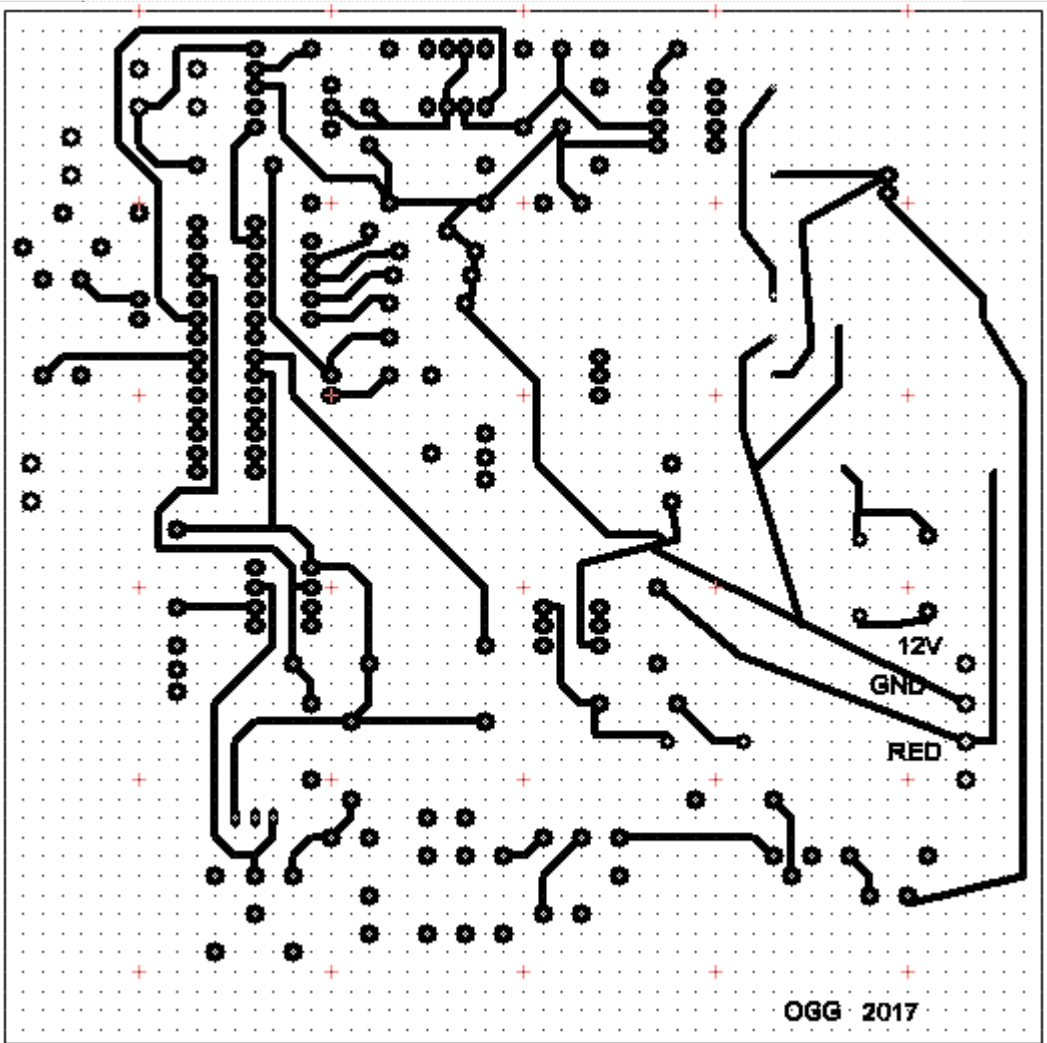
*El transformador esta sobredimensionado , se podría calcular el limite de saturación y hacer uno mas adecuado.

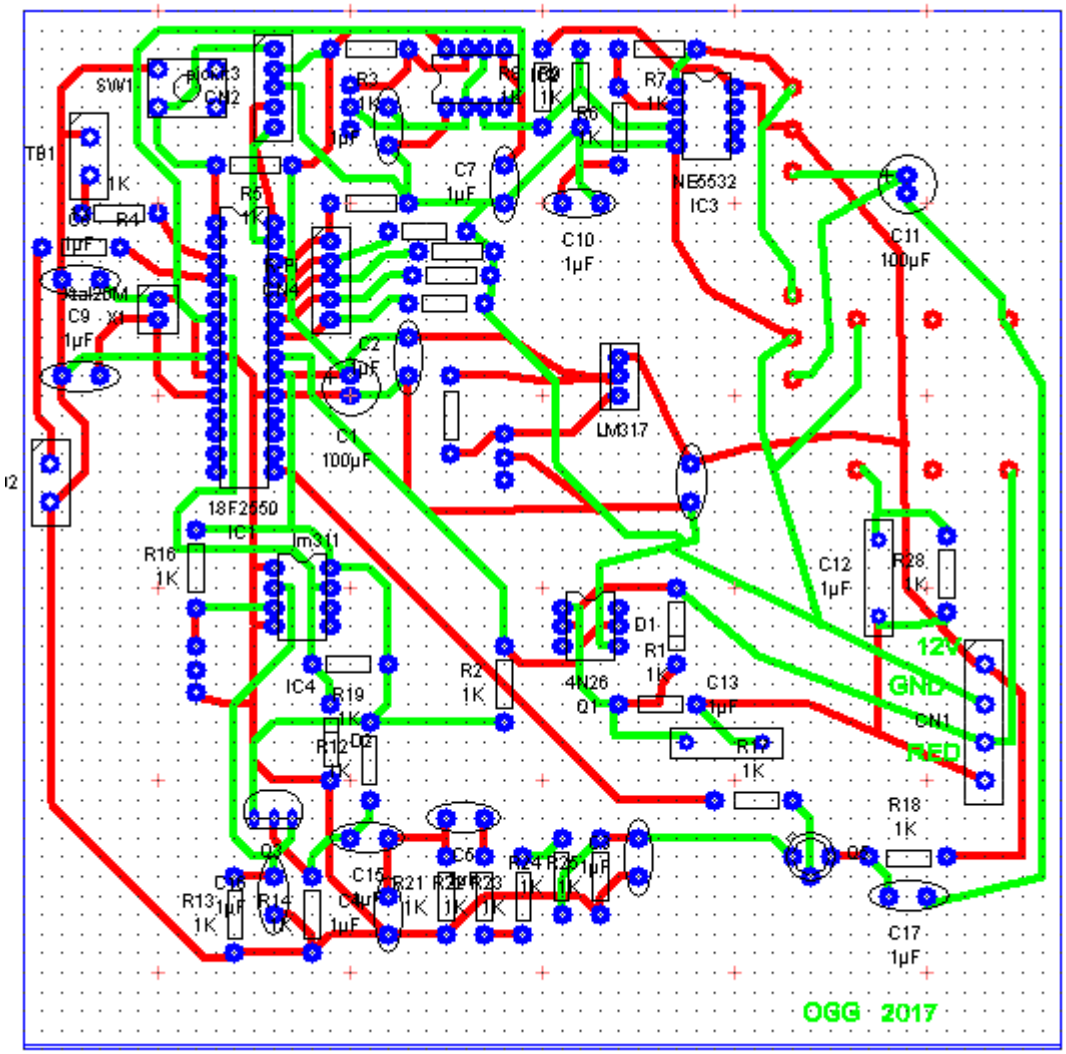
* Teniendo en cuentas las mejoras anteriores, también se podría con el mismo hardware, cambiar la velocidad de envío . En vez de mandar un bit por ciclo de red , enviar varios por cada cruce por cero. Agregar además , por ejemplo , un código de Hamming para salvar errores , en vez de aplicar redundancia de envío .

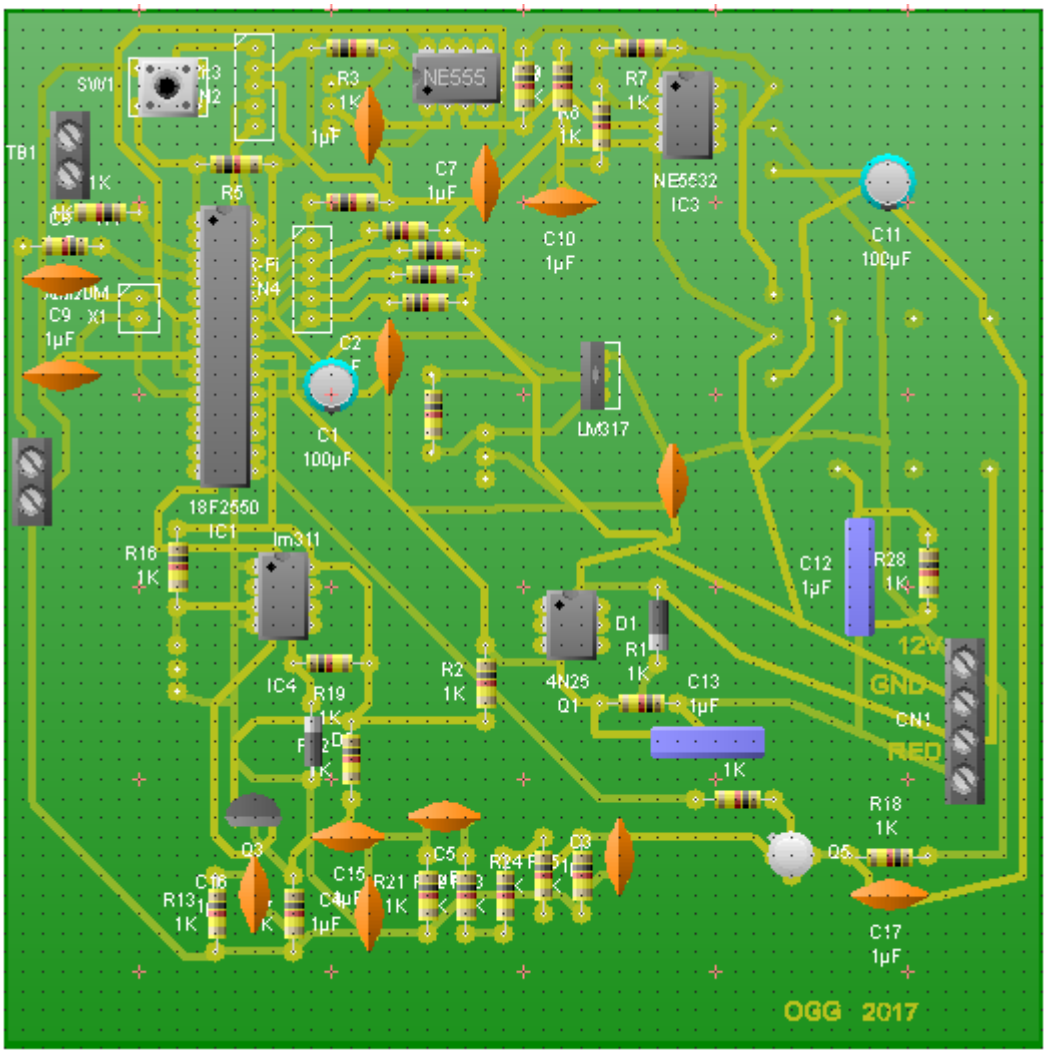
6-Anexo

Las placas se diseñaron con el programa PCBWizard. No todos los dispositivos utilizados en el diseño estaban presentes en las librerías del software y, aunque se tenía la opción de crear sus propios componentes y agregarlos, se optó por dejarlos fuera del diseño y luego dibujarlos manualmente con fibra indeleble. Se realizó un ruteo en doble faz.




12V
GND
RED
OGG 2017


SW1
TB1
1K
R4
1µF
Xtal20M
C9
1µF
X1
R3
1K
C7
1µF
R5
1K
C10
1µF
R7
1K
NE5532
IC3
C11
100µF
C2
1µF
C1
100µF
LM317
18F2550
IC1
lm311
R16
1K
IC4
R19
1K
R2
1K
4N26
Q1
D1
R1
1K
C13
1µF
C12
1µF
R28
1K
12V
GND
CN1
FIED
R18
1K
C17
1µF
R12
1K
Q2
C15
1µF
C6
1µF
R13
1K
R14
1K
R21
1K
R24
1K
R25
1K
OGG 2017



Bibliografía

Tesis del Ing. Martín Pérez, *"Control a través de Internet de una red domótica X-10 mediante microcontroladores"*, Mar del plata , noviembre 2013

Datasheet PIC 18F2550, de Microchip.

Datasheet LM311

Datasheet NE555

Datasheet NE3255

Datasheet CNY71

http://www.aquihayapuntes.com/compilador-pcw-ccs.html