UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

CONTROL Y MONITORIZACIÓN DE SISTEMAS DE DOMÓTICA MEDIANTE UN INTERFAZ WEB

Titulación: Ingeniero Técnico de Telecomunicación, especialidad en Telemática
Autor: Amit L. Babani Babani
Tutores: Norberto Ramos Calero
Fecha: Febrero 2014

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

Control y monitorización de sistemas de domótica mediante un interfaz web

HOJA DE FIRMAS

Alumno/a

Fdo.: Amit L. Babani Babani

Tutor/a

Fdo.: Norberto Ramos Calero
Fecha: Febrero 2014

UNIVERSIDAD DE LAS PALMAS
DE GRAN CANARIA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

CONTROL Y MONITORIZACIÓN DE SISTEMAS DE DOMÓTICA MEDIANTE UN INTERFAZ WEB

HOJA DE EVALUACIÓN

Calificación: _____

Presidente

Fdo.:

Vocal

Fdo.:

Secretario/a

Fdo.: Nombre del secretario

Fecha: Febrero 2014

Índice

Índice

1. Introducción General 11
2. Memoria 17
2.1 Introducción al Proyecto 17
2.2 Diseño de circuitos 19
2.2.1 Placa Principal 22
2.2.2 Placa Mando 27
2.3 Lista de Componentes 32
2.4 Comunicaciones entre interfaces 38
2.5 Software de la Placa Principal 42
2.5.1 Resumen 44
2.5.2 Descripción de Autómatas 47
2.5.3 Diagrama de Estados 50
2.6 Software de la Placa Mando 55
2.6.1 Resumen 57
2.6.2 Descripción de autómatas 59
2.6.3 Diagrama de bloques 62
2.7 Software de la Placa Web 67
2.7.1 Resumen 69
2.7.2 Descripción de funciones 73
2.9 Recursos 76
2.9.1 MPLab 8 77
2.9.2 Lenguaje Ensamblador 78
2.9.3 PICKit 2 79
2.9.4 Proteus Isis 81
2.9.5 HTML5 82
2.9.6 DIA Diagram 84
2.9.7 Trimble SketchUP 86
2.9.8 Adobe Photoshop 87
2.9.9 Protocolo IEEE 802.3 89
2.9.10 YAT (Yet Another Terminal) 90

2.10 Conclusión 92
3. Presupuesto 95
3.1 Pliego de condiciones 97
3.2 Presupuesto - Montaje placa principal 99
3.3 Presupuesto - Montaje placa mando (1 unidad) 100
3.4 Presupuesto -Desarrollo software y diseño de circuitos 101
3.5 Presupuesto – Sistema total (1 Placa Principal + 4 Mandos) 102
4. Anexos 103
4.1 Reglamento de baja tensión ITC-BT-51 105
4.2 Código Fuente Placa Principal 110
4.3 Código Fuente Placa Mando 138
4.4 Código Fuente Página Web 157
5.0 Bibliografía 165

1. Introducción General

1. Introducción General

En la actualidad, son mayores los casos en que nos encontramos con un gran avance tecnológico en herramientas que usamos de forma cotidiana en nuestras vidas, con mayores prestaciones y facilidades; en consecuencia, no sólo nos ha beneficiado incrementando su número de funciones, sino también nos ha cambiado en el modo en que interactuamos con ellas. Este cambio se puede apreciar especialmente en el modo que han evolucionado nuestros teléfonos móviles, desde la aparición de los smartphones, coches con la introducción de los navegadores de viaje GPS y los televisores inteligentes smartTVs. Este desarrollo tiene lugar mayormente gracias a la incorporación de una conexión internet a estos dispositivos, abriendo así un abanico con un sinfín de funcionalidades en nuestros dispositivos, ya que les brinda la posibilidad de interactuar e intercambiar datos con otros sistemas y dispositivos conectados a la misma red. No era de esperar pues que este "salto tecnológico" llegase a nuestros hogares con la introducción de la domótica.

Hoy en día podemos entender como domótica al conjunto de las tecnologías aplicadas destinadas al control y la automatización inteligente en un espacio específico, que abre la posibilidad de gestionar la eficiencia del uso de la energía, seguridad, confort, y la posibilidad al usuario de poder interactuar con el sistema en su conjunto. Un sistema domótico es capaz de recoger información proveniente de dispositivos, sensores u entradas, procesarlas y efectuar tareas determinadas. Dicho sistema, tiene la capacidad de comunicarse con otras redes e Internet.

Las prestaciones que puede ofrecer un sistema domótico pueden variar de un lugar a otro, dependiendo de para qué tipo de espacio fue diseñada su implantación. Por ejemplo, si nos planteamos instalar un sistema domótico en un espacio industrial, lo más conveniente sería destinar sus funcionalidades a la gestión y la optimización de la eficiencia del uso energético de sus máquinas, en un sistema de seguridad capaz de interactuar con las cámaras de vigilancia y sistemas de alarmas.

En cambio, si nos planteamos diseñar un sistema domótico destinado al uso doméstico como se plantea en la Ilustración 1 - Dispositivos que pueden interactuar con un sistema domótico en una casa, lo más probable sería adaptar sus funcionalidades al confort de sus habitantes, dando la posibilidad de gestionar la temperatura, activar o desactivar electrodomésticos cuando sean o no sean requeridos de forma autónoma, control de iluminación total de la casa desde un único punto, video porteros, telefonía, etc.

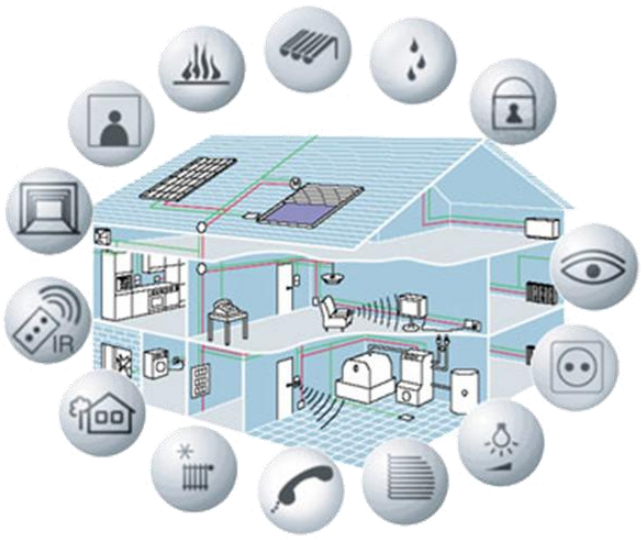


Ilustración 1 - Dispositivos que pueden interactuar con un sistema domótico en una casa

Otro ejemplo más práctico y común de un sistema domótico es el aplicado en el sector agrícola, cada vez más frecuente a día de hoy. Se trata de un sistema inteligente de regadío, que permite la actuación de los aspersores sobre un cultivo todos los días a una determinada hora; no obstante, si los sensores han captado lluvia ese mismo día, los aspersores no se activarán, lo que implica un ahorro del agua para el agricultor y, a su vez, contribuye a la mejora del medio ambiente, incrementando el bienestar social en general.

En cuanto a la regulación, la instalación interior eléctrica y la red de control del sistema domótico, estos se rigen por el **Reglamento Electrotécnico para Baja Tensión (REBT)**[1]. En particular, la red de control del sistema domótico está regulada por la instrucción **ITC-BT-51 Instalaciones de sistemas de automatización, gestión técnica de la energía y seguridad para viviendas y edificios.**

En conclusión, podemos entender la domótica como la respuesta a los requerimientos que plantean los cambios sociales y las nuevas tendencias en nuestra forma de vida, facilitando el diseño de hogares y espacios industriales más humanos, personales, funcionales y flexibles.

En este proyecto, el sistema domótico titulado ***"Control y monitorización de sistemas de domótica mediante un interfaz web"*** se propone como ejemplo práctico el control y gestión de un sistema de iluminación compuesto de 4 luces dentro de un espacio. Además de la luminaria, el espacio en donde el sistema domótico esté instalado, contendrá 4 sensores de presencia y otros 4 mandos inteligentes. Cada luz instalada en el espacio estará asociada a un sensor de presencia y a un mando inteligente. Cada uno de estos tendrá la funcionalidad de poder apagar, encender así como dejar en modo automático la luz que tenga asignada. Si una luz está funcionando en modo automático, solo se encenderá cuando el sensor de presencia detecte una persona u objeto, y se apagará en cuanto abandone la zona que el sensor de presencia esté muestreando.

Los modos en los que trabaje cada sistema de iluminación puede ser monitorizada y gestionada también a través de una página web como se puede observar en la Ilustración 2; dicha página web

[1] Consultar anexo 4.1 para más información

también da la posibilidad al usuario de controlar todas las luces del espacio; estos cambios realizados serán notificados a los mandos inteligentes que a su vez transmitirán estos a través de un juego de 3 led que llevan incorporados. El led rojo indicará que su luz asociada está apagada, el led verde que está encendido y el led amarillo indicarán que la luz se mantendrá encendida de forma automática mientras el sensor de presencia detecte un objeto en su área.

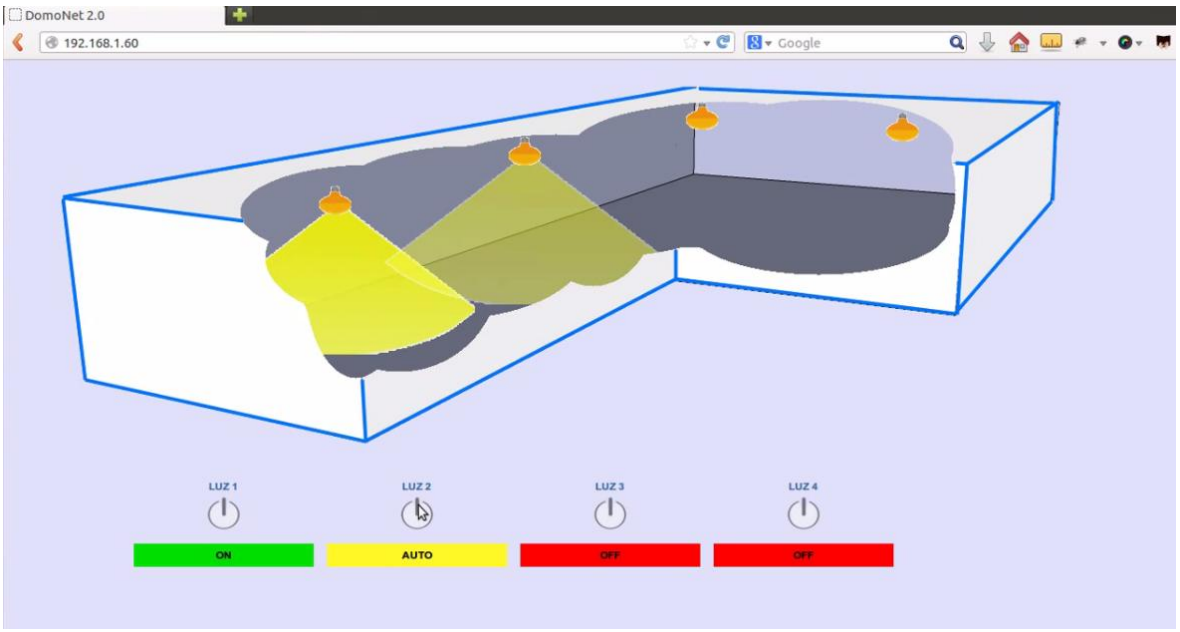


Ilustración 2 - Página web encargada de la gestión monitorización del sistema domótico

El sistema domótico propuesto tiene como objetivo plantear un sistema genérico, destinado para que pueda ser aplicado a cualquier tipo de espacio, tanto industrial, agrario o doméstico y ser un sistema flexible y de fácil adaptación para interactuar con otros interfaces y dispositivos con un mínimo esfuerzo. Para ello cabe destacar las siguientes características del proyecto:

- **Código abierto**: Permite el acceso al código de la aplicación, dando la posibilidad al usuario final de modificar y actualizar el software en caso necesario.

- **Construcción económica**: Para su construcción se emplean componentes habituales y de fácil obtención, haciendo posible que no solo grandes industrias puedan hacer uso del sistema, sino también aficionados.

- **Posibilidad de interactuar desde cualquier dispositivo con conexión a internet**: El panel de gestión y monitorización es accesible desde una página web bajo los estándares HTML5, esto hace posible su acceso desde cualquier dispositivo móvil u ordenador con una conexión internet y un navegador web.

- **Comunicación entre interfaces empleando un bus RS485**: Al emplear este tipo de comunicaciones, se añade flexibilidad absoluta en la incorporación de nuevos dispositivos al sistema.

El sistema completo domótico está compuesto por un total de 6 dispositivos que se comunican entre sí a través de un mismo canal de comunicaciones.

- **1 Dispositivo de procesado principal:** Todas las comunicaciones entre los dispositivos conectados al canal pasarán y serán gestionadas a través de esta placa. La placa principal será la que estará conectada directamente a las luces del sistema y a los sensores de presencia.

- **1 Dispositivo web:** Está placa será la destinada a habilitar una página web para la gestión y monitorización del sistema completo. Esta a su vez estará conectada al bus de datos para que se pueda comunicar con la placa principal y a un punto de acceso con internet usando un cable Ethernet para que el usuario pueda interactuar con él a través de un navegador web[2].

- **4 Dispositivos de mandos inteligentes:** Estos dispondrán de un pulsador, y estarán conectados al canal de datos para poder comunicarse con la placa principal.

En la construcción del prototipo para la demostración del funcionamiento del sistema domótico se diseñó una placa independiente por cada dispositivo empleado, salvo en el caso del dispositivo web y dispositivo principal, que fueron integradas en una única placa, aunque no es imprescindible seguir este diseño para la construcción de futuros prototipos.

Como base tecnológica para la construcción del sistema domótico empleamos microprocesadores PIC16F690[3] como núcleo de procesado en los dispositivos de mandos inteligentes y dispositivo principal. Para el dispositivo web, se empleó un Lynx EzWeb[4] como núcleo de procesado y servidor de la página web.

En los últimos años se ha podido observar en el mercado un aumento cada vez mayor de productos y servicios destinados a la domótica, dando cada vez una mayor oferta y consolidación. Hoy en día, la domótica aporta soluciones dirigidas a todo tipo de espacios, tanto a los destinados a la vivienda, como al ámbito laboral y la agricultura. Además, se ofrece más funcionalidad a un menor coste, más variedad de producto, y, gracias a la evolución tecnológica, son más fáciles de utilizar e instalar. Esta tendencia ha hecho que cada vez sea más habitual encontrarnos en la red la venta de dispositivos similares al propuesto en este proyecto utilizando tecnologías más avanzadas y costosas como puede ser My N3RD[5], una aplicación móvil que consigue apagar y

[2] Navegadores web compatibles y testados: Mozilla Firefox, Google Chrome, Safari o Internet Explorer.
[3] Microprocesador de 8 bits fabricados por Microchip.
[4] Microprocesador fabricado por Lynx, con un microservidor web integrado.
[5] http://www.myn3rd.com/

encender dispositivos que funcionen en un rango de entre 5 y 24 Voltios de potencia conectados al dispositivo wifi provistos de ellos, o bien Homeseer[6], un sistema de domótico más complejo, que permite interactuar con una mayor gama de interfaces fabricadas por la misma empresa destinadas al control de agua, iluminación, apertura de puertas, etc.. Por lo que es evidente resaltar, que el sector de la domótica tendrá una mayor demanda en los próximos años en la medida en la que nos vayamos habituando más a su uso.

La documentación del proyecto: **"Control y monitorización de sistemas de domótica mediante un interfaz web"** contendrá la explicación con mayor detalle, separado en diferentes apartados los siguientes aspectos; esquemáticos y diseños de los circuitos electrónicos, componentes utilizados para el ensamblaje de las placas prototipo y una documentación detallada sobre la construcción de las tramas de datos empleados para la comunicación entre los distintos interfaces.

Más adelante, una vez explicada la circuitería se dedicarán varios apartados acompañados de diagramas de flujo y esquemas, a mostrar las diferentes funcionalidades que poseen las aplicaciones con las que fueron programados los controladores que gestionan el sistema en conjunto. En total son 3 aplicaciones distintas las que gestionan el sistema completo; el microcontrolador principal, encargado de gestionar las comunicaciones; la aplicación del servidor web, encargada de servir la página web y por último la aplicación que contendrá el microcontrolador de la placa de mando, encargada de enviar las órdenes al sistema.

Terminamos la documentación con un pliego de condiciones, detallando los requerimientos para el correcto funcionamiento del sistema, los recursos tecnológicos empleados, tanto aplicaciones como documentación externa utilizada, y finalmente cerramos con nuestra conclusión personal y el presupuesto separado en diferentes apartados detallando el coste en mano de obra y material para la construcción del presente proyecto.

[6] http://www.homeseer.com/

2. Memoria

2. Memoria

2.1 Introducción al Proyecto

Con este proyecto, se pretende facilitar el control y la gestión a los usuarios en realizar acciones comunes en un espacio laboral, industrial y/o doméstico, tal como el control de la luz de una casa u oficina. Además este proyecto pretende que el sistema se pueda extrapolar a otros usos que pueden ser de gran utilidad para otras industrias donde las necesidades a cubrir sean de una mayor complejidad.

El método a seguir para cumplimentar dichos objetivos, será mediante la utilización de un sistema domótico, el cual, permitirá el control de un sistema de iluminación en un espacio determinado, permitiendo así al usuario, su control según sus necesidades. Este sistema también cuenta con la incorporación de sensores de presencia que encenderán y apagaran las luces en el caso de detectar una persona o un objeto dentro de su área de muestro.

El sistema domótico también ofrece la posibilidad de monitorizar y gestionar la iluminación de todos los habitáculos del espacio (habitaciones, cocina, pasillos, cuarto de almacenaje…) donde ha sido instalado de forma individual, con la posibilidad de encender, apagar las luces, o bien activar/desactivar los sensores de presencia para que las luces se enciendan únicamente cuando alguien esté dentro de la habitación. Todo ello es posible gracias a la implementación de un sistema de monitorización y control mediante una página web compatible programado bajo los estándares HTML5, lo que permitirá su total acceso desde cualquier dispositivo informático provisto de un navegador web y una conexión a internet.

La principal ventaja a destacar del actual sistema domótico propuesto radica en la facilidad del usuario para acceder y monitorizar tareas comunes a gestionar en un espacio de forma remota, desde cualquier lugar y en cualquier momento. Uno de los objetivos de este proyecto, una vez haya alcanzado una fase de desarrollo en el que pueda comercializar y ser empleado al mercado global, es dar la posibilidad de instalar dicho sistemas a otros sectores claves dentro de la industria como a la doméstica.

2.2 Diseño de circuitos

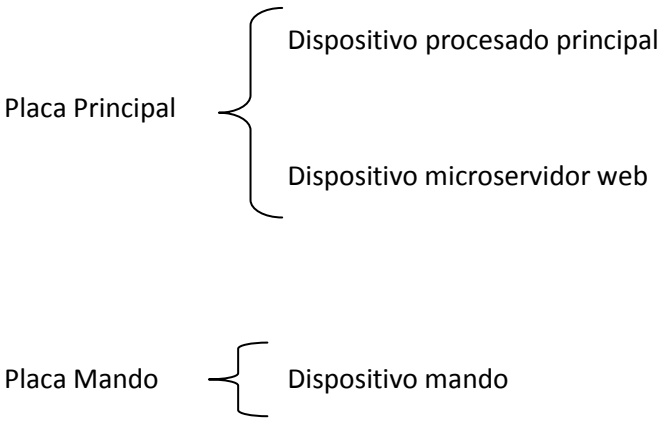
2.2 Diseño de circuitos

El sistema completo domótico propuesto consta del diseño y del montaje de varias placas, todas ellas siguen un patrón de trabajo, de diseño y programación similar entre ellas. Cabe destacar que todas ellas:

- Trabajan con tensiones TTL (5 Voltios)
- Tienen un regulador de tensión incorporado para la protección de pequeños sobre voltajes (hasta 25 Voltios)
- Utilizan una controladora MAX487 para gestionar la comunicación con el canal de datos
- Todas las placas están conectadas al mismo canal de datos
- Disponen de una microcontroladora con un software específicamente diseñado para realizar su función

Para la demostración del sistema domótico se han diseño dos tipos de placas prototipos. La primera, la placa principal, en donde han integrado los dispositivos del microservidor web, que da servicio a la página web dando las funcionalidades de gestión y monitorización remota, y el dispositivo de procesado principal, encargada de gestionar todas las comunicaciones entre las distintas placas, estados de las luces como muestrear los sensores de presencia en caso necesario.

La segunda placa, denominada placa de mando, tiene como objetivo el recoger las instrucciones enviadas por los usuarios dentro del espacio de forma manual, ya que estos dispositivos disponen un pulsador para tal efecto. En futuros diseños, la placa mando podrá contener un teclado numérico completo para poder enviar instrucciones más complejas al sistema domótico. El sistema puede contener más de una placa del tipo mando, siempre y cuando el número de dispositivos totales conectados al bus de datos no supere los 128, esta es una limitación de hardware que viene dada por la controladora empleada. En esta demostración se han simulado hasta un total de 4 placas de mando y se han construido dos 2 prototipos mando. Para que las placas mando puedan comunicarse entre sí, será necesario que todas ellas estén conectadas al mismo bus común de datos en donde está conectada la placa principal como se muestra en la Ilustración 3 .



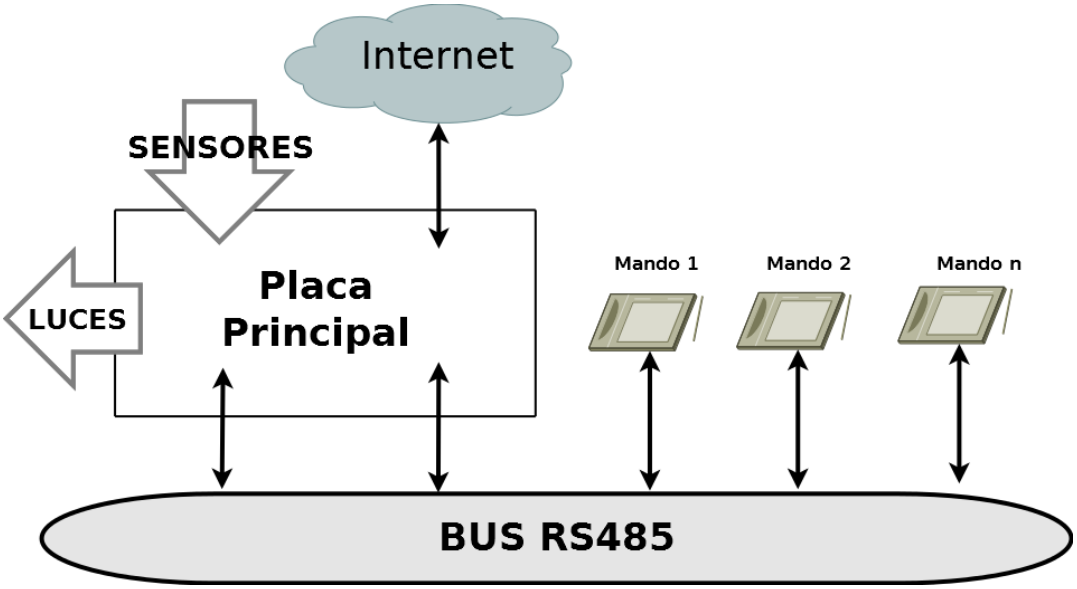


Ilustración 3 - Conexionado general

Todas las placas y/o dispositivos funcionan independientemente, el único dispositivo indispensable para que el sistema domótico pueda funcionar correctamente, sería de un dispositivo principal. El resto de los dispositivos que componen el sistema añaden funcionalidades extras al sistema no indispensables, de este modo se consigue un sistema domótico escalable en donde tiene cabida poder diseñar nuevos dispositivos adicionales en el futuro y poder agregarlo al bus de datos, previamente añadiendo su respectiva instancia dentro del dispositivo principal para que pueda interactuar con él.

Todos los componentes empleados para la construcción de todos los elementos han sido soldados sobre placas PCB estándares empleando capsulado DIL, ya que de esta forma facilitó su construcción, modificación y toma de medidas durante todo el proceso de construcción.

La placa principal desempeña dos funciones, la primera es el interactuar con los sensores de presencia así y el sistema de luces y el segundo es el gestionar la página web en donde se podrá monitorizar el estado de la iluminación en tiempo real.

El sistema puede tener varias placas de mando siempre y cuando los dispositivos totales conectados al sistema no superen los 128 dispositivos. La función de las placas mando es dar la posibilidad de configurar manualmente el estado de la luz a la que está asociada, ya que cada mando está asociado a una única luz. Los estados posibles que pueden estar cada luz son: *encendido, apagado y automático*. Las placas mando están provistas de un pulsador, cada vez que son actuados comunicarán su cambio de estado a la placa principal.

2.2.1 Placa Principal

2.2.1 Placa Principal

La placa principal se considera la unidad central, y está compuesta por dos dispositivos; el dispositivo microservidor web, encargada de albergar y ofrecer la página web que permite el control del sistema domótico completo, y el dispositivo de procesado principal, encargada de sincronizar todos los datos enviados a través del hilo de comunicaciones y el servidor web. Estos dos dispositivos se comunican entre sí a través del bus común de datos al que están conectados todos los dispositivos, inclusive las placas mando. En la placa principal se integran ambas unidades en una misma placa, pero no necesariamente necesitan que ambas estén en la misma placa para las posibles futuras modificaciones futuras, pudiendo utilizar dos placas de forma independiente, uno para cada uno si en la implementación real lo requiriese por cuestiones de conexionado.

En la siguiente ilustración se muestra el circuito diseñado para la construcción de la placa principal que tiene como función gestionar los sensores de presencia (utilizando unos pulsadores en su lugar), el sistema de iluminación gestionados a través de un juego de relés y el microservidor web.

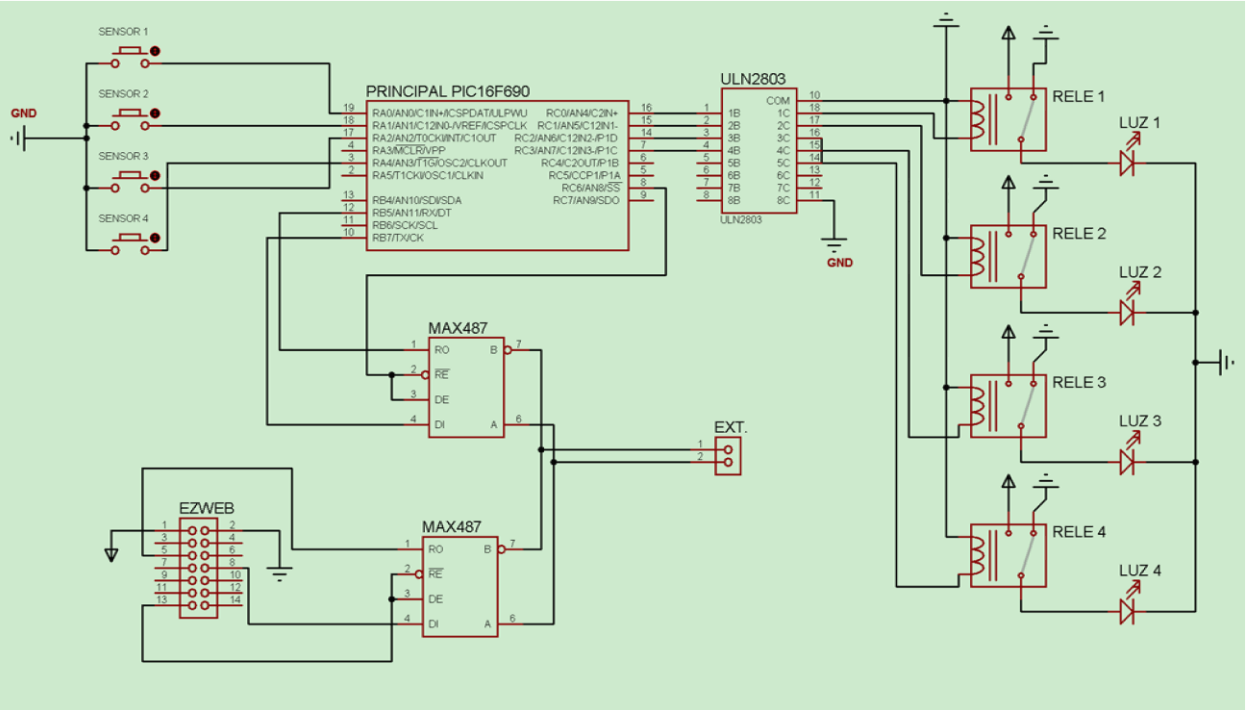


Ilustración 4 – Esquemático placa principal

Fijándonos en la Ilustración 4 de la placa principal, podemos observar que se emplea un microcontrolador PIC16F690 con la aplicación programada como unidad de procesado principal, esté estará conectado al juego de sensores de presencia y al sistema de relés para gestionar el sistema de iluminación, dos controlador MAX487 para gestionar las comunicaciones con el canal de datos común RS485[7], uno de ellos se utilizará para el microservidor web basado en el

[7] Consultar: 2.4 Comunicación entre interfaces para más información

microcontrolador EzWeb y el segundo para la unidad principal de procesado, la toma toma externa se usará como conector hacía el bus de datos donde se conectaran el resto de las placas mando.

Para el prototipo se han empleando unos pulsadores para simular los sensores de presencia que están conectados al microcontrolador PIC16F690 del dispositivo principal de procesador a través del puerto A configurado como entrada, estos sensores/pulsadores no requieren de una resistencia, ya que están configurados como pull-ups por software, la detección de la activación de los pull-ups se hace por el software integrado en el microcontrolador PIC16F690.

El relé conectado al microcontrolador se utiliza para simular un sistema de iluminación completo, estos están conectados al puerto C configurado como salida del microcontrolador. Para la construcción del prototipo con fines demostrativos se ha empleado un juego de diodos led en lugar de un sistema de iluminación completo para simular su funcionamiento.

La transmisión de datos entre el microservidor Ezweb y el microcontrolador se hace a a mediante el puerto de comunicaciones EUSART que viene integrado en los propios microcontroladores de fábrica. En este caso también se utiliza una controladora MAX487 para interconectar dichos puertos, y a su vez conectar el bus de datos creado entre ambas controladoras a la toma externa que también la comunicación a las demás placas mando del sistema. En un diseño previo se barajeo la posibilidad de implementar un segundo puerto serie por software en el microcontrolador para comunicarse con el servidor web Ezweb para ahorrar la implementación de una segunda controladora MAX487 utilizada por el microservidor Ezweb, pero finalmente se descartó dicha posibilidad, ya que de esta forma ofrece una mayor flexibilidad al sistema, dando la opción de separar el servidor web de la placa principal en futuros diseños en caso necesario. La controladora MAX487 es una controladora half-duplex, por lo que se establece en modo recepción por defecto, en el momento en el que el dispositivo de procesador principal o el microservidor web requiera transmitir un dato establecerá su respectiva controladora MAX487 en modo envío durante la transmisión de la trama. Para ello el microservidor Ezweb empleará el pin 13, configurado como salida digital, y el microcontrolador el pin 6 del puerto C configurado como salida para alternar entre envío y recepción sus respectivas controladoras MAX487.

La placa principal requerirá de una tensión continua de 5V TTL[8] para que todos los componentes funcionen correctamente. La entrada de corriente a la placa estará regulada por un circuito de regulador de tensión detallado en la Ilustración 5.

[8] *Transistor-transistor logic*, es decir, lógica transistor a transistor

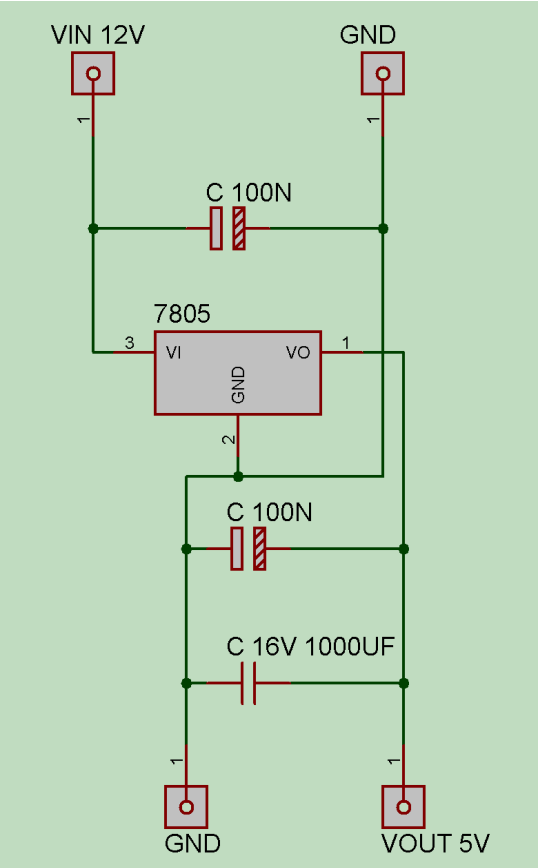


Ilustración 5 – Esquemático de regulador de tensión

En la Ilustración 5 se aprecia el conexionado y los componentes empleados para el regulador de tensión. Este regulador de tensión estará montando en la entrada corriente de ambos tipos de placa. Gracias a este regulador, podemos asegurarnos de que la placa siempre recibirá una tensión de 5 Voltios, y evitamos el sobre voltaje del mismo para evitar daños no deseados.

Como resultado final siguiendo diseño del circuito para la placa principal descrito en el presente apartado, se ha ensamblado el siguiente prototipo, este difiere del diseño inicial en el número de led y sensores que tiene, ya que solo se implementaron 2 led y 2 sensores en lugar de 4 como se puede observar en el diseño del circuito principal, ya que para efectos demostrativos no variará su resultado:

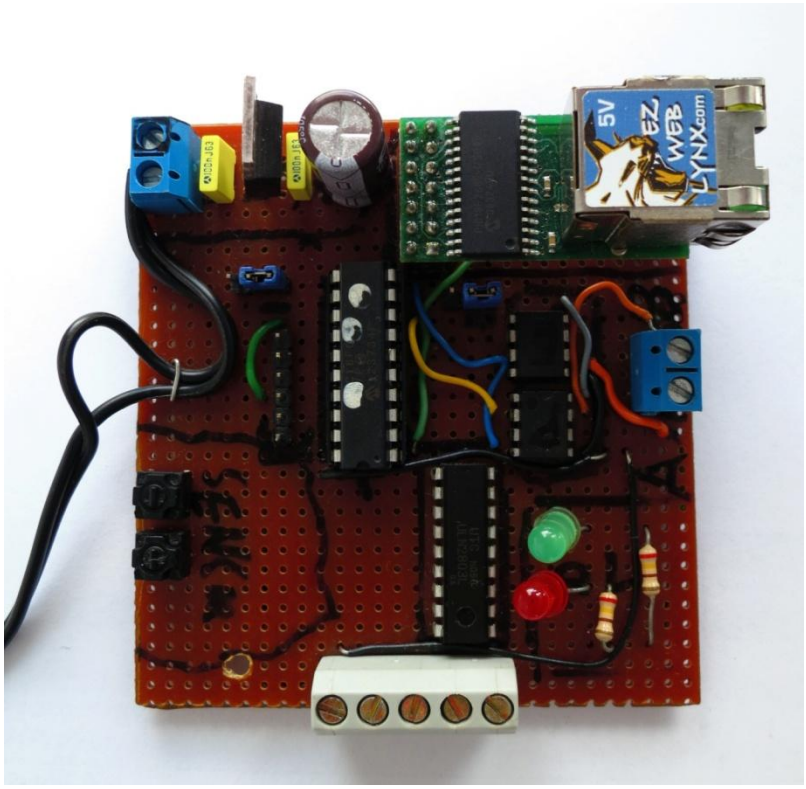


Ilustración 6 - Fotografía del prototipo de la placa principal

2.2.2 Placa Mando

2.2.2 Placa Mando

Las placas mando son las encargadas de recoger las ordenes enviadas por los usuarios dentro del espacio domótico. Cada una de estas placas estará asociada a una luz y a un sensor de presencia, está vinculación se hace a nivel software en la placa principal. La placa mando tiene como función el enviar y recibir tramas de datos con las órdenes pertinentes a la placa principal informando del modo de trabajo de la luz asignada a la misma como los posibles cambios a efectuar. Por ejemplo, si la luz 1 está encendida (operando en modo ON), este modo de trabajo estará señalado a través del juego de led del mando 1 encendiendo el led verde. Si un usuario que se encuentre dentro del espacio domótico pulsa sobre el pulsador del mando 1, la luz 1 se apagará (modo OFF), este cambio será indicado encendiendo el led rojo del mando 1, y a la vez la placa principal será notificado de dicho cambio para notificar al resto de los dispositivos en el sistema, como al servidor web para que sea mostrada en la página web.

De forma análoga, si otro usuario realiza un cambio en la luz 1 en la página web, el servidor web notificará a la placa principal para que este posteriormente notifique al mando 1 encienda su respectivo led para indicar a los usuarios de la sala en el modo en el que está trabajando.

Los modos posibles en que puede operar cada luz son: encendido (ON), apagado (OFF) y automático (AUTO) los cuales se alternan sucesivamente cada vez que se acciona el pulsador provisto en la placa mando o en la página web en el siguiente el orden descrito:

Modo ON →Modo OFF → Modo AUTO → Modo ON → Modo Off…

Cuando una luz está operando en modo automático, solo se encenderá mientras su sensor de presencia asociado detecte una persona dentro de su área de muestro.

La siguiente Ilustración 7 se muestra el esquemático de la placa mando, en ella se puede apreciar los elementos comunes en todas las placas, la controladora de comunicaciones RS485 (MAX487), el microcontrolador (PIC16F690), la toma externa y el regulador de tensión (Ilustración 5). Además de estos elementos comunes, la placa mando también viene equipada con un juego de 3 led; rojo, verde y amarillo, para indicar su estado y un pulsador. El led rojo indicará que está apagado, el verde encendido y el amarillo operando en modo automático.

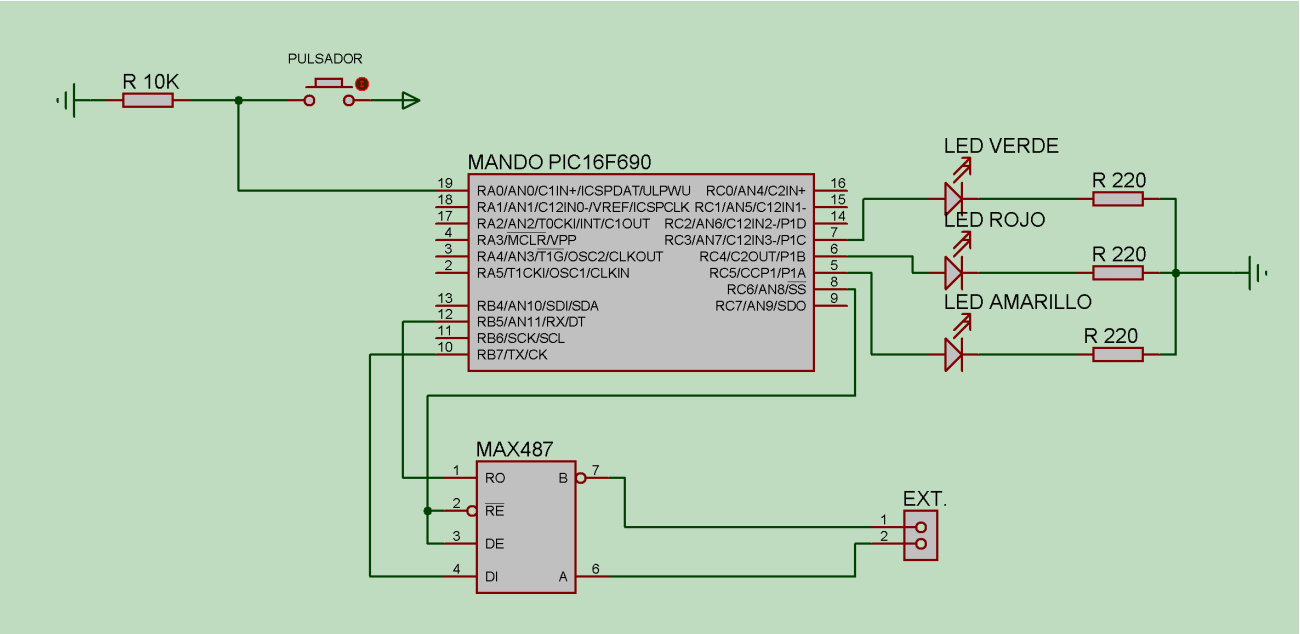


Ilustración 7 - Esquemático placa mando

El pulsador empleado, requiere de una resistencia, y esta, está conectada al puerto A configurado como entrada del microcontrolador. Hemos dejado el resto de los pines del puerto A libres para una posible actualización en el futuro, donde en lugar de trabajar con un solo pulsador, se emplee un teclado completo.

Los diodos led, cuya única función es la de indicar el modo en el que está operando el sistema de luz asignado, el diodo color verde indicará que está en modo encendido, el rojo en apagado y el amarillo en automático. Estos están conectados al puerto C configurado como salida del microcontrolador. Para la correcta iluminación de los diodos led se ha empleado una resistencia.

La controladora MAX487 está conectada a la toma externa y a los pines 5 y 7 del microcontrolador configurado para operar como puerto serie. La controladora MAX487 estará en modo recepción excepto cuando tenga que enviar un dato, en ese caso el pin 6 del puerto C configurado como salida lo establecerá en modo transmisión durante el proceso de envío de la trama poniendo el pin a nivel alto.

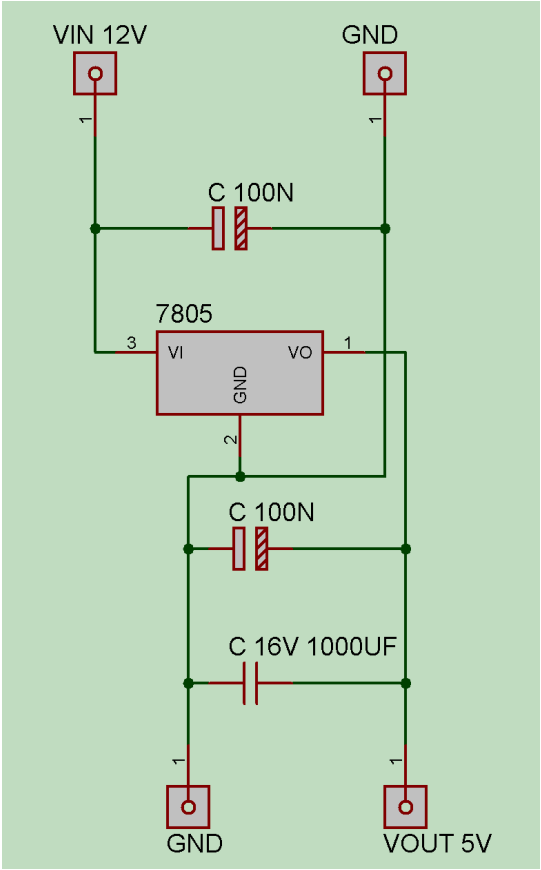


Ilustración 8 – Esquemático de regulador de tensión

En la Ilustración 8 se aprecia el conexionado y los componentes empleados para el regulador de tensión. Este regulador de tensión es idéntico al usado en la placa principal descrito en el apartado anterior, ya que la placa mando al igual que el resto de las placas requieren de una tensión TTL de 5 Voltios para su correcto funcionamiento.

A partir del circuito diseñado se construyó el siguiente prototipo sobre una placa de prototipos, en un diseño final el tamaño de dicha placa podría verse reducido empleando componentes con encapsulación SMD[9]:

[9] Componentes pequeños y livianos empleados en montajes superficiales que disminuyen considerablemente las dimensiones de construcciones eléctricas

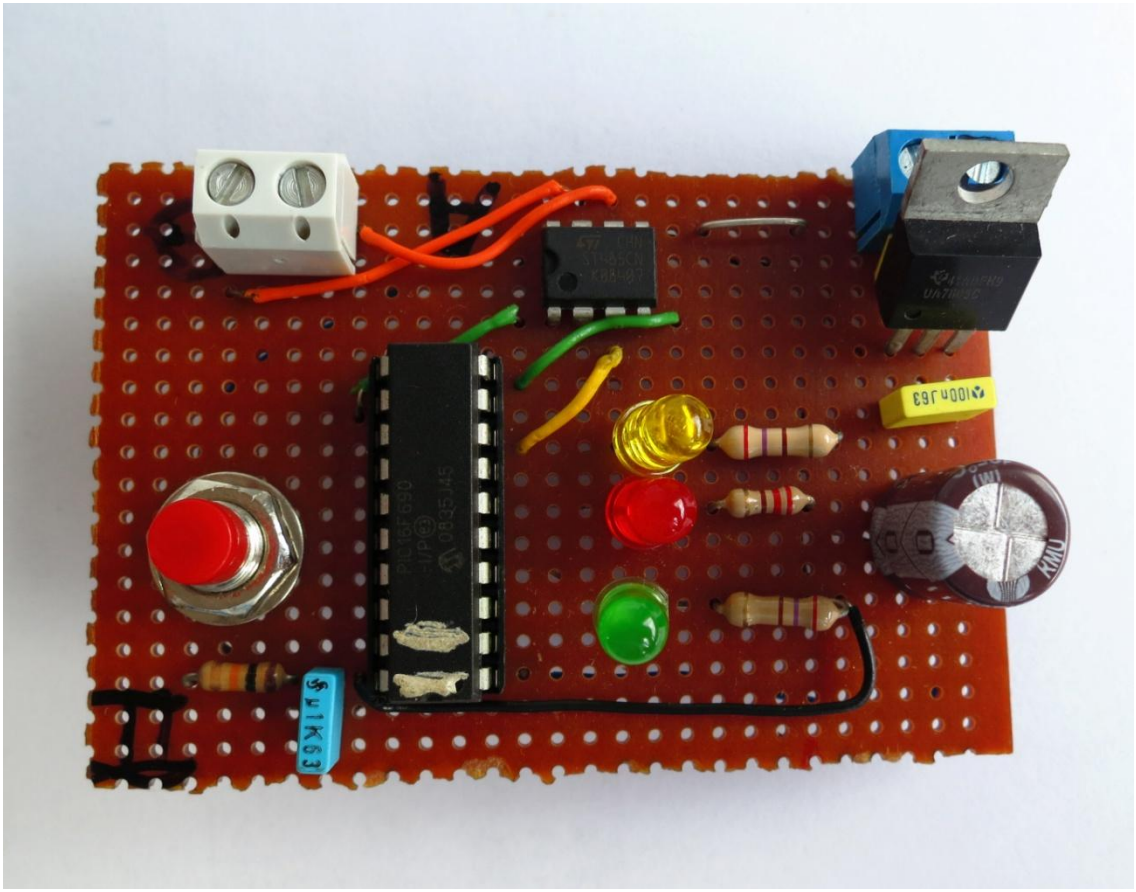


Ilustración 9 - Fotografía del prototipo de la placa mando

2.3 Lista de

Componentes

2.3 Lista de Componentes

Se emplearon los siguientes componentes para la construcción de los prototipos:

Placa Principal

- 1 x Ezweb Lynx 5V
- 1 x PIC16F690
- 2 x Max487
- 4 x Luces (Se emplearon Diodos Led para el prototipo)
- 4 x Resistencias de 240 Ohmios (para el prototipo)
- 1 x Resistencia de 4700 Ohmios (4k7), para el bus
- 1 x Controlador de relés ULN2803
- 4 x Sensores (Se emplearon pulsadores para el prototipo)
- 1 x Toma externa
- 2 x Jumperes
- 1 x Conector de 6 pines

Placa Mando

- 1 x PIC16F690
- 1 x Max487
- 3 x Diodos Led
- 3 x Resistencias 240 Ohmios
- 1 x Resistencia 10000 Ohmios (10K)
- 1 x Condensador 100nF
- 1 x Pulsador
- 1 x Toma externa

Estabilizador de corriente (por cada placa)

- 1 x Fuente de alimentación 9 voltios
- 1 x Condensador de 1000uF
- 2 x Condensadores 100nF
- 1 x Regulador 7805
- 1 x Toma externa

Microcontrolador PIC16F90

El microcontrolador PIC16F690 desarrollado por Microchip, en encapsulado DIP fue el empleado para los interfaces. Dicho microcontrolador contiene elementos necesarios para conformar una "unidad central de procesamiento" UCP (o CPU en inglés).

Un microcontrolador es un circuito integrado que incorpora las tres unidades funcionales de una computadora: CPU, memoria y unidades de E/S, es decir, se trata de una mini computadora integrada en un circuito.

Las principales características que posee son:

- Mapa de Memoria: Con un programa contador de 13-bit capaz de direccionar un programa de 8K x 14 espacios de memoria. Solo los primeros 4K x 14 (0000h-0FFFh) son físicamente ejecutados
- Puertos: Tiene 3 puertos configurables como entrada/salida
- PORTA: Puerto bidireccional de 6-bits de ancho
- PORTB: Puerto bidireccional de 4-bits de ancho
- PORTC: Puerto bidireccional de 8-bits de ancho
- Organización de Memoria: Los datos en la memoria son divididos en 4 bancos que contienen los Registros de Propósito General (GPR) y los Registros de Funciones Especiales (SFR). Los Registros de Funciones Especiales están localizados en los primeros 32 lugares de cada banco. Los Registros de Propósito General, implementado como RAM estática se localizan en los últimos 96 lugares de cada banco.
- Stack Pointer: Con un Nivel 8 x 13-bit de ancho de pila de hardware. Los espacios en la pila no son parte de los programas o espacios de datos y el puntero de pila no es legible o re-escribible. El PC es apilado en la pila cuando la instrucción CALL es ejecutada o una interrupción a causa de una rama. La pila es retirada por la ejecución de los comandos RETURN, RETLW o un RETFILE.
- Convertidor A/D: Convertidor integrado de datos analógicos y digitales
- EEPROM y Flash: La Memoria EEPROM contiene datos de lectura y escritura La memoria de programa Flash es la sección donde se almacenará nuestro programa, y es de solo lectura durante su ejecución. Estas Memorias no son directamente asignadas en el archivo de registro de espacio. En lugar de ello, son indirectamente tratadas a través de los Registros de Función Especial (FRS).

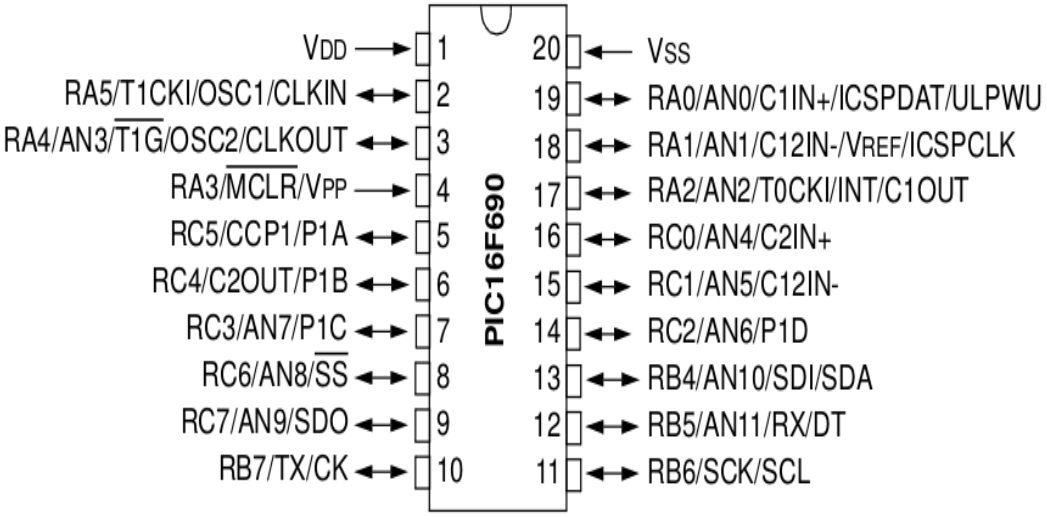


Ilustración 10 - Pinout PIC16F690

MAX487

La función de los integrados MAX487 es la creación de un bus de comunicación entre los distintos interfaces, empleando una comunicación RS-232. Todos los microcontroladores utilizados en el diseño vienen provistos de un puerto USART, lo que hace su implementación más sencilla como escalable hasta 128 dispositivos. El integrado MAX487 destaca por ser un transreceptor de bajo consumo, y permite transmisiones de hasta 250kbps sin errores con un consumo que varía entre 120µA y 500µA, aunque esta puede descender a 0.1µA cuando no está en uso, y requiere una entrada constante de 5V (TTL). Se empleo un encapsulado DIP para el montaje del prototipo.

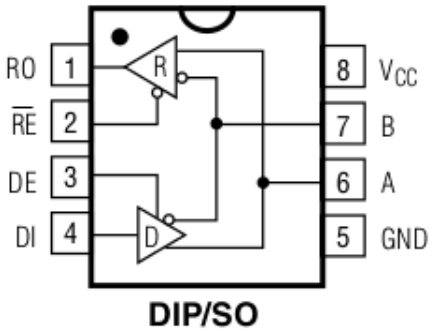


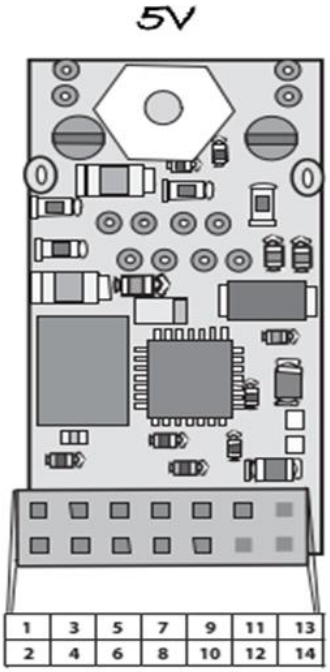
Ilustración 11 - Pinout MAX487

Ezweb Lynx 5V

El Ezweb Lynx es un integrado que permite añadir una comunicación con la red de internet y un servidor de páginas web integrado a un solo circuito. Las aplicaciones web que se pueden programar dentro del servidor web se pueden desarrollar en HTML5.

El Ezweb destaca por:

- Comunicación por comandos AT, AT sobre UDP y HTTP CGI, que se pueden integrar dentro del código HTML5
- 11 puertos de entrada/salida de los cuales:
  - 5 puertos pueden trabajar de forma analógica
  - 2 puertos como sensores de temperatura I2C Dallas en Celsius o Fahrenheit
- Bloqueo del periférico por contraseña
- 256 registros volátiles y no volátiles de 8 bits
- Comunicación serial
- Envío de emails a través de servidor SMTP
- Implementación para pantallas LCD
- ID único de periférico, para su identificación por si se trabaja con varios a la vez
- Control total de configuración IP



T4.2a **5V EZ Web Lynx**

Pin Number	*Description*
1	Analog Input 1/Digital I/O
2	Analog Input 2/Digital I/O
3	Analog Input 3/Digital I/O
4	Analog Input 4/Digital I/O
5	Analog Input 5/Digital I/O
6	Pulse Counter (not available on all hardware revisions)
7	PWM (not available on all hardware revisions)
8	TX - Serial Data Transmit (from EZ Web Lynx to PC or other serial device)/Digital I/O
9	RCV - Serial Data Receive (from PC to EZ Web Lynx)/ Digital I/O
10	Dallas DS1621 or DS1631 I²C Temp Sensor SCL (clock) line/Digital I/O
11	Dallas DS1621 or DS1631 I²C Temp Sensor SDA (data) line/Digital I/O
12	RST (Active-low Reset)
13	Vdd (5V)
14	GND

Ilustración 12 - Pinout Ezweb

ULN2803

El ULN2803 es un array de 8 transistores Darlington en un encapsulado de 18 pines. Es ideal para hacer de puente entre sistemas lógicos digitales (TTL y CMOS) a circuitos de mayor voltaje necesarios para alimentar lámparas, relés, impresoras, etc.

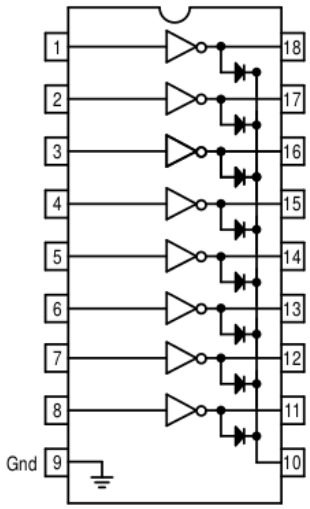


Ilustración 13 - Pinout ULN2803

Regulador 7805

78xx es la denominación de una familia de reguladores de tensión positiva, de tres terminales: voltaje de entrada, voltaje de salida y masa. El regulador 7805 es el más común utilizado en diseños empotrados que trabajan con tensiones TTL (5 voltios). El 7805 es un regulador lineal hecho por varios fabricantes como Fairchild, o ST Microelectronics. Incluyen protección por sobrecarga térmica, y contra cortocircuitos. Requiere una tensión de entrada de 7V- 25V para su correcto funcionamiento. Para corrientes de salida hasta de 1A existen dos tipos de encapsulados: TO-220 (vertical) y D-PAK (horizontal). El encapsulado es el TO-220.

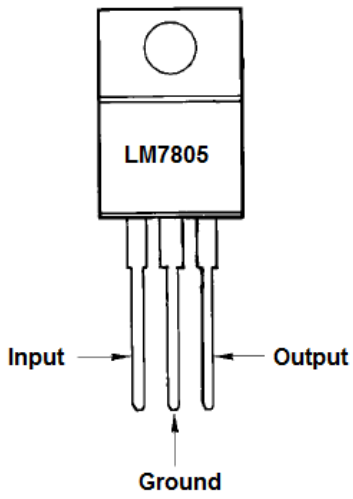


Ilustración 14 - Pinout 7805

2.4 Comunicaciones entre interfaces

2.4 Comunicaciones entre interfaces

Para la comunicación entre distintas placas que componen el sistema se ha empleado un bus que funciona bajo el protocolo RS485. RS485 destaca por ser un sistema de comunicaciones multipunto, las transmisiones se realizan sobre dos hilos half-duplex. Al ser su trasmisión en diferencial son idóneas para el sistema propuesto por la protección frente a ruidos que posee y la posibilidad de comunicación entre varios interfaces que vengan provistos de un puerto serie integrado sin tener que modificar el software integrado en el controlador para ello.

La estructura de las tramas empleadas en el sistema son similares a las empleadas en las comunicaciones de red con protocolo IEEE 802.3. Los mandos, o los sistemas genéricos están conectados a la placa principal siguiendo una topología tipo bus indicado en la Ilustración 15:

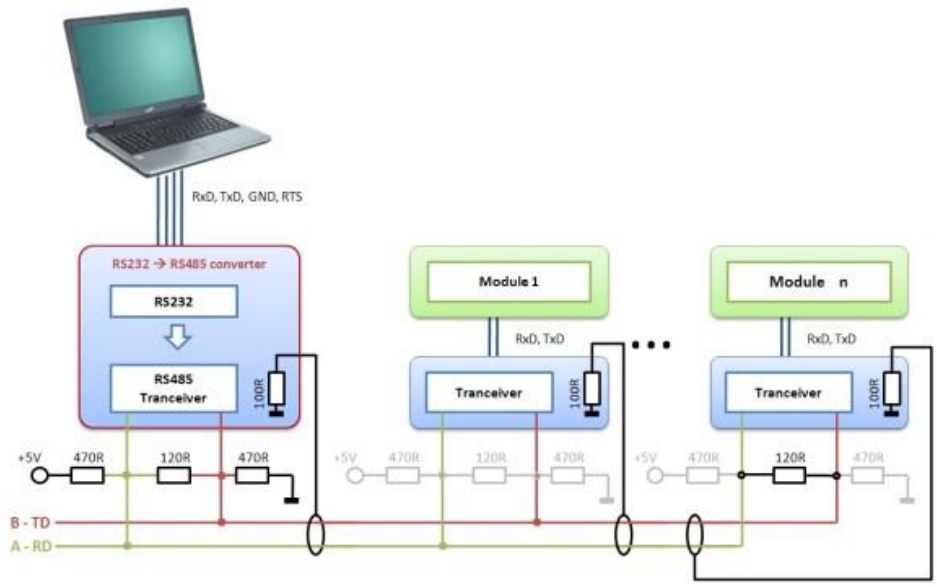


Ilustración 15 - Topología bus

Todas las placas envían datos de forma simultánea cuando lo requieran los comandos al bus, y la placa principal será la encargada de sincronizar dichos datos, procesarlos, y responder a las solicitudes realizadas en el sistema. Varios ejemplos de cómo las placas se comunicarían entre sí a través del hilo de comunicaciones común sería el siguiente:

Ejemplo de secuencia de tramas

La secuencia a seguir en las comunicaciones de tramas entre las distintas placas podría ser la siguiente:

Cuando la placa mando necesita notificar un cambio de estado:

- Mando envía trama → Es recibida y tratada por la Placa principal.
- Placa principal envía trama de verificación → Recibida por servidor web y la placa mando
- La placa mando procesa el dato y actualiza su juego de led, el servidor web también procesa el mismo dato y actualiza su página

Cuando el servidor web necesita notificar un cambio de estado:

- Servidor web envía trama → Es recibida y tratada por el controlador principal.
- Placa principal envía trama de verificación → Recibida por servidor web y la placa mando
- La placa mando procesa el dato y actualiza su juego de led, el servidor web también procesa el mismo dato y actualiza su página

La placa principal también es la encargada de actualizar el estado de las luces a través de los relés y tomar muestras del estado de los sensores (en el caso de que haya alguna luz operando en modo automático).

Formato de trama

El formato de los datos enviados por las distintas placas y del servidor web es semejante a las tramas empleadas en las comunicaciones de redes. Estas tramas siempre contendrán los siguientes campos:

Preámbulo	Destino	Origen	Long	Datos	CRC	FIN

Con ello se intenta estandarizar todas las comunicaciones tanto de las placas que el sistema contiene actualmente, como las posibles futuras placas que se podrán ir integrando al sistema en posteriores actualizaciones.

A continuación explicamos con mayor detalle que datos contendrá cada campo:

Preámbulo: Para detectar el inicio de trama, se emplea el byte 0xAB, semejante al inicio de trama para en las comunicaciones con protocolo IEEE 802.3, siendo en binario 10101011.

Destino: Este campo contendrá un byte en donde se especifica a quien está destinado la trama de datos, cada dispositivo en el sistema tiene un identificador único, detallado a continuación:

- 0x01 : Mando 1
- 0x02: Mando 2
- 0x03: Mando 3

- 0x04: Mando 4
- 0x10: Placa Principal
- 0x20: Servidor Web

Origen: Utilizando los mismos identificadores que en el campo destino, en este campo se define que dispositivo emitió la trama de datos.

Long: Este campo, definido por un byte, define la longitud del siguiente campo de datos. En el caso que la trama haya sido emitida por una placa mando, por el servidor web o bien desde la placa principal con destino a una placa mando, la longitud del campo de datos será de 7 (detallado a continuación), en cambio si la placa principal emite una trama al servidor web, este campo tendrá el valor de 11, ya que el campo de datos contendrá 11 bytes.

Datos: La longitud del campo datos, especificada anteriormente, viene definida por el campo LONG, en este sistema estamos empleando dos tipos de datos, datos usados para la comunicación entre:

Placa Principal <--> **Placa Mando**
Servidor Web → Placa Principal

El formato de trama utilizado entre la placa y las placas de mandos y del servidor web hacía la placa principal estará compuesto por 7 bytes y contendrá los siguientes valores:

LaEbMc (7 bytes)

- a = Identificador del juego de iluminación (para futuras implementaciones)
- b = Estado de la iluminación (para futura implementaciones)
- c = Modo de operación (0x00-Auto, 0x01-On, 0x02-Off)

El modo de operación indica el modo en el que está actuando la luz que tiene asignado dicho mando, si la luz esta en modo:

AUTO: Permanecerá encendida durante X segundos cada vez que se detecte movimiento a través de los sensores conectados al placa principal.

ON: La luz asignada se mantendrá encendida.

OFF: La luz asignada estará apagada

Por ejemplo, si una placa mando quisiera notificar que está trabajando en modo OFF, o bien la placa principal quisiera establecer una placa mando en modo OFF enviaría el siguiente dato:

L0E0M2

Placa Principal → Servidor Web

El siguiente formato de datos es el utilizado entre la placa principal y el servidor web. Este campo de datos estará compuesto por 15 bytes.

Inicio trama datos	**1º secuencia AT**	**Nº de Luz**	**2º parte de secuencia AT**	**Estado**	**Fin trama datos**
<CR><CL>	AT*NVREG	1→4	=	0 →2	<CR><CL>

El estado viene significando lo siguiente:

- 0: Modo Auto
- 1: Modo On
- 2: Modo Off

Por ejemplo, si fuera necesario notificar al Ezweb que la luz 2 está en modo auto, el campo datos tendría la siguiente información:

AT*NVREG2=0

Debido a que la placa web no está siempre a la espera de recibir un dato para tratarlo, se ha recurrido a la opción de usar uno de sus comandos AT internos para almacenar el estado de cada luz en una dirección de memoria distinta, así cada vez que la placa web se cargue tome los valores del estado de cada luz de sus respectivos bancos de memoria.

Es muy importante que los comandos enviados al microservidor Ezweb estén delimitados con los caracteres <CR><CL> [10]para que los pueda procesar, el carácter <CR> (equivalente equivale a un salto de carro y <CL> a un salto de línea.

CRC: Código de verificación, para asegurar que el sistema ha recibido la trama con los datos correctos. El cálculo del CRC se realizará en la placa destino, y posteriormente lo comprobará con el recibido en la trama, si el valor de CRC coincide, la trama es correcta, si no, será desechada, ya que será un indicio de que ha habido un error en la comunicación.

El cálculo del CRC se calcula sumando el valor de todos los bytes de la trama del campo origen hasta el último byte del campo datos con complemento a 1.

FIN: El campo de fin de trama está definido por dos bytes (0x05 0x05), esto es necesario para los interfaces que requieran un campo de fin de trama para sus sistemas de buffer como en el caso del servidor web (Ezweb)

[10] Los equivalentes en hexadecimal a dichos caracteres son los siguientes: <CR>=0x0D <CL>=0x0A

2.5 Software de la Placa Principal

2.5.1 Resumen

2.5.1 Resumen

La placa principal es el dispositivo de procesado central del sistema domótico. La función de dicha placa es hacer de enlace entre las demás placas del sistema para gestionar las luces y sensores. Se puede entender que la placa principal es quien ejecuta las órdenes.

El modelo empleado en el proyecto permite más de una placa principal para facilitar la implementación donde el área a cubrir sea muy extenso con una pequeña previa modificación en el software, ya que cada placa mando necesitará estar asociada solo a una placa principal.

En la placa principal se ha integrado el microcontrolador PIC16F690 junto al servidor web (Ezweb), conectado a través del canal de datos (bus RS485) con el objetivo de simplificar el prototipo, pero no necesariamente necesitan estar en la misma placa, ya que ambos trabajan de forma independiente. En este apartado se ha hecho un mayor hincapié en el funcionamiento de la placa principal y a sus conexiones. En el apartado dedicado al servidor web Ezweb se profundizará más en dicho microcontrolador.

El desarrollo del software se ha escrito en lenguaje ensamblador, ya que de esta forma se consigue una mayor optimización y control del microcontrolador, a contrapartida de tener que escribir más líneas de código para poder implementar las funciones requeridas.

Las funciones (o autómatas) que componen la aplicación completa son llamadas ininterrumpidamente una tras otra en un bucle infinito, cada autómata está separada en diferentes estados, y cada vez que un autómata es invocado sólo se ejecuta un estado únicamente para luego volver al bucle principal. El inicio de la ejecución de ciertos autómatas/funciones puede ser gestionado a través de un conjunto de indicadores globales, haciendo la tarea de un semáforo. Siguiendo esta metodología en la programación conseguimos un sistema no bloqueante y damos la apariencia de trabajar con un sistema multitarea, ya que podrán ejecutarse simultáneamente varios autómatas a la vez de forma independiente, aparte de obtener una mayor grado de optimización en el procesado, ya que en las tareas como en el envío o en la recepción de datos donde se requiere un tiempo de espera entre byte y byte podrán ser aprovechadas por otros autómatas para ejecutar otras tareas. Después de varias pruebas se ha demostrado que el microcontrolador trabajando con su reloj interno configurado a 8Mhz es más que suficiente para realizar el procesado requerido sin interrupciones ni tiempos de esperas perceptibles.

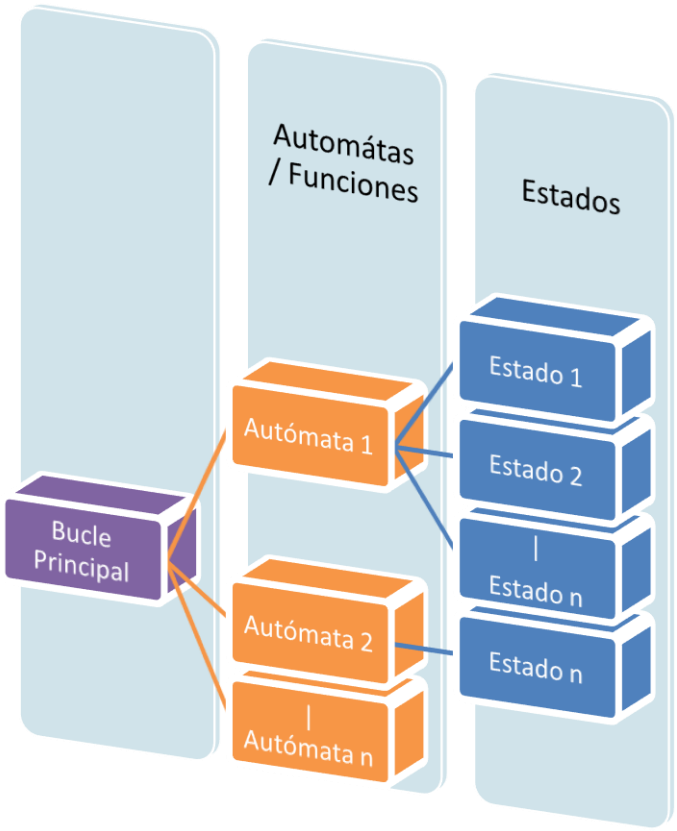


Ilustración 16 - Diagrama relacional entre autómatas y estados

Como sistema de trabajo y una mejor ordenación y estructuración del software, se diseñaron con anterioridad un diagrama de bloques con sus posibles estados para cada función necesaria de la aplicación completa antes de escribir el código final. Al diseñar el diagrama de bloques por cada función (o autómata) necesario se puede observar desde una perspectiva más global la estructura que sigue el software en su conjunto, facilitando así las tareas en el análisis y desarrollo.

2.5.2 Descripción de Autómatas

2.5.2 Descripción de Autómatas

La placa principal cuenta con una serie de funciones para el desarrollo de sus objetivos, estas funciones se ejecutan en un bucle infinito, y su ejecución está gestionada por indicadores (flags), para optimizar los ciclos del reloj del microcontrolador. Cada función, igualmente está separada en diferentes estados, ejecutándose un estado por cada llamada a la función, de esta forma se consigue un sistema de multitarea al igual que en la placa mando. Los indicadores empleados en las funciones de la placa principal son activados en los siguientes casos:

Descripción de indicadores:

- **Dataready:** Activado cuando el buffer de entrada tenga almacenada una trama válida.
- **Eval_rx:** Activado cuando el buffer de entrada tenga una trama aún no evaluada.
- **Eval_mando:** Activado cuando el buffer de entrada tenga una trama almacenada del tipo mando.
- **Serial:** Activado cuando el buffer de salida tenga una trama lista para ser enviada.

La aplicación de la placa principal contendrá las siguientes funciones, algunas ellas solo se ejecutaran si algunos flags cumplen los requerimientos que necesitan las dichas funciones para que puedan desarrollar su tarea.

Read_serial(): Encargada de leer en el bus datos, si detecta un byte de inicio de trama este empezará a almacenar los datos en un buffer. En el caso que la trama de datos sea válida, se activará el flag de **dataready** y **eval_rx**. Para que esta función se ejecute, se requerirá que el flag de **dataready** y **serial** esté desactivado.

Evaluar_rx(): Esta función será la encargada de evaluar el tipo de datos recibidos, en este proyecto solo se trabajará con datos del tipo mando, ya que el Ezweb también enviará los datos en el mismo formato que los mandos. Esta función solo se ejecutará si el flag **eval_rx** está activado. En el caso que la función identifique la trama como proveniente de un mando desactivará el flag **eval_rx** y activará el flag **eval_mando,** en caso contrario desactivará ambos flags.

Evaluar_Mando(): Función encargada de tratar los datos recibidos y actualizar la variable "modo_sync" y construir los buffers de salida, uno destinado para informar a los mandos y otro segundo para informar al microservidor web Ezweb. Una vez que haya terminado de construir los dos buffers, activará el flag **serial** y desactivará el flag **eval_mando** para indicar que hay un buffer listo para ser enviado. Para que esta función se ejecute será necesario que el flag **eval_mando** este activado, y el flag **serial** desactivado, para asegurarse que no hay ningún buffer esperando a ser enviado.

Enviar_tx(): Si el flag de **serial** está activado, enviará a través del bus la trama de datos que tenga almacenada en el buffer de salida, una vez que haya enviado dicha trama desactivará el flag de **serial**, para que otra función pueda reescribir de nuevo en el buffer de salida.

Sensor(): Función encargada de monitorizar los sensores de presencia del juego de luces que estén funcionando en modo auto, como el de encender y el de apagar las luces en el caso que dicho sensor detecte un objeto o persona dentro de área de muestro.

Clear_error(): Siempre en funcionamiento, función encargada de detectar errores en la eusart y corregirlo.

En la siguiente ilustración se puede observar de una forma más global los cambios sufridos en los indicadores a la medida que las funciones están siendo ejecutadas, este será muy parecido al empleado en la aplicación empleada en la del mando.

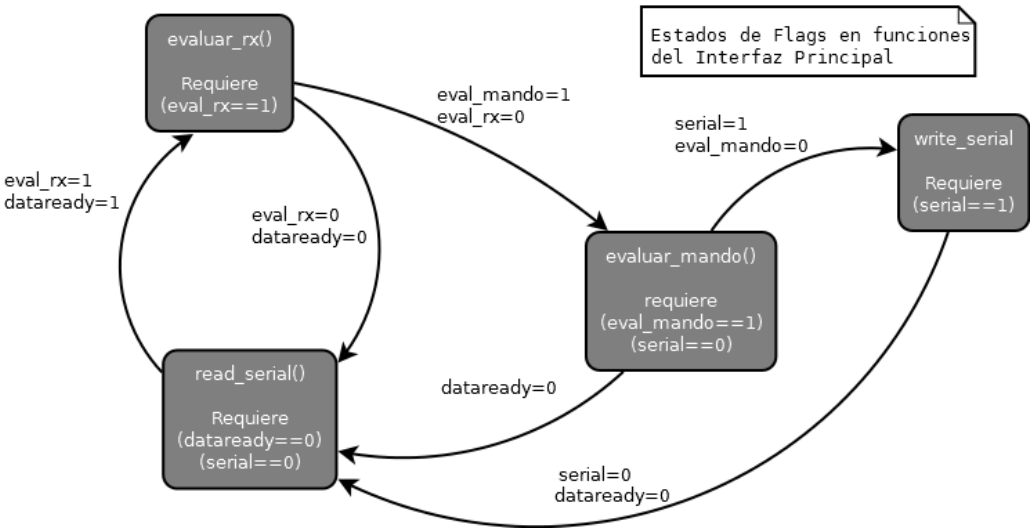


Ilustración 17 - Diagrama de cambios de flags en placa principal

2.5.3 Diagrama de Estados

2.5.3 Diagrama de Estados

Clear_error

Función encargada para detectar errores de sincronización de ventana del tipo "overrun". Los errores overrun tienen lugar cuando se produce un sobre desbordamiento de datos en el buffer de entrada.

Esta función siempre está en funcionamiento, sin importar el estado de los flags de estos. Si la función detecta un error de ventana o en el desbordamiento, se encargará de reiniciar la EUSART y vaciar el buffer de entrada para corregir el problema y para que la aplicación siga funcionando con normalidad.

DESCRIPCIÓN DE LOS ESTADOS

- **Estado Inicial:** Si detecta que el bit OERR en el registro RCSTA está a nivel alto, esta invocará la función RESET_OERR. Si OERR está a nivel bajo la función, terminará y volverá al bucle principal.
- **RESET_OERR:** Esta función se encargará de reiniciar la EUSART, y si detecta un dato en el buffer de entrada llamará a la función EMPTY_BUFFER
- **EMPTY_BUFFER:** Está función estará encargada de vaciar el buffer de entrada, no saldrá de su bucle hasta que el buffer de entrada esté totalmente vacío.

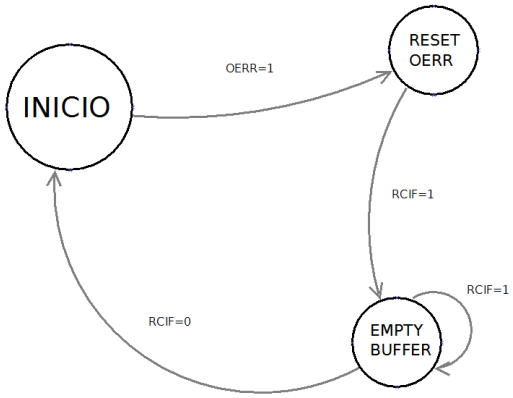


Ilustración 18 -Diagrama de bloques Clear_error

Evaluar_rx

Esta función es la encargada de evaluar el buffer de entrada en el caso que el flag **dataready** este activado. Este, evaluará su contenido e identificará el emisor de dicha trama (dirección 0x40) y, si la trama proviene de un placa que se conoce, se activará el flag indicado para que sea posteriormente evaluada por otra función. En este caso práctico solo reconocerá un tipo de trama, por lo que si recibimos una trama valida se activará el flag **eval_mando** para que sea posteriormente evaluada por la función evaluar_mando().

Read_serial

Para que esta función se ejecutara necesitaría que el flag de **dataready** y del de **serial** esté desactivado. Es la función encargada de leer en el bus datos si recibe alguna trama válida a la placa en cuestión, dicha trama será desechada si no cumple los siguientes requisitos:

- No comienza con el byte de inicio de trama (0xAB)
- El campo destino de la trama no coincide con la dirección *host* de la placa
- La trama no tiene un tamaño de datos válido
- El valor del CRC (suma de verificación), no coincide con el calculado en la recepción

En el caso de ser válida se marcará como activo el flag de **dataready** y **eval_rx**.

DESCRIPCIÓN DE LOS ESTADOS

- **E0:** Estado de detectar el inicio de trama, no irá al siguiente estado hasta que le llegue el byte 0xAB.
- **E1:** Estado de detectar el si la trama está destinada a la placa receptora. La dirección del receptor se define en los parámetros iníciales bajo la variable HOST. Si no está destinada a esta placa se vuelve al estado inicial E0, en caso contrario se iría al siguiente. En este estado se inicia el cálculo del CRC (suma de verificación).
- **E2:** Estado encargado de detectar la dirección del emisor de la trama. La placa sólo permitirá la recepción de tramas de la dirección definida bajo la variable CONTROL definida en los parámetros generales. En caso de estar permitida se salta al siguiente estado y se actualiza el CRC, en caso contrario se vuelve al estado inicial E0.
- **E3:** Estado encargado de recibir la longitud del campo datos. El sistema solo permite campos de datos de longitud 7. Si es diferente a este, se volverá al estado E0, en caso contrario se salta al siguiente estado y se actualiza el valor del CRC.
- **E4:** Estado encargado de almacenar el los 7 bytes del campo datos en el buffer de entrada. Una vez estén los 7 bytes, se saltará al siguiente estado y se actualizará el valor del CRC.
- **E5:** Estado de comprobación del CRC, se comparará el CRC recibido con el calculado localmente, si coinciden se marcará el flag de **dataready** a 1, de lo contrario, se dejará en 0. Se vuelve al estado inicial E0.

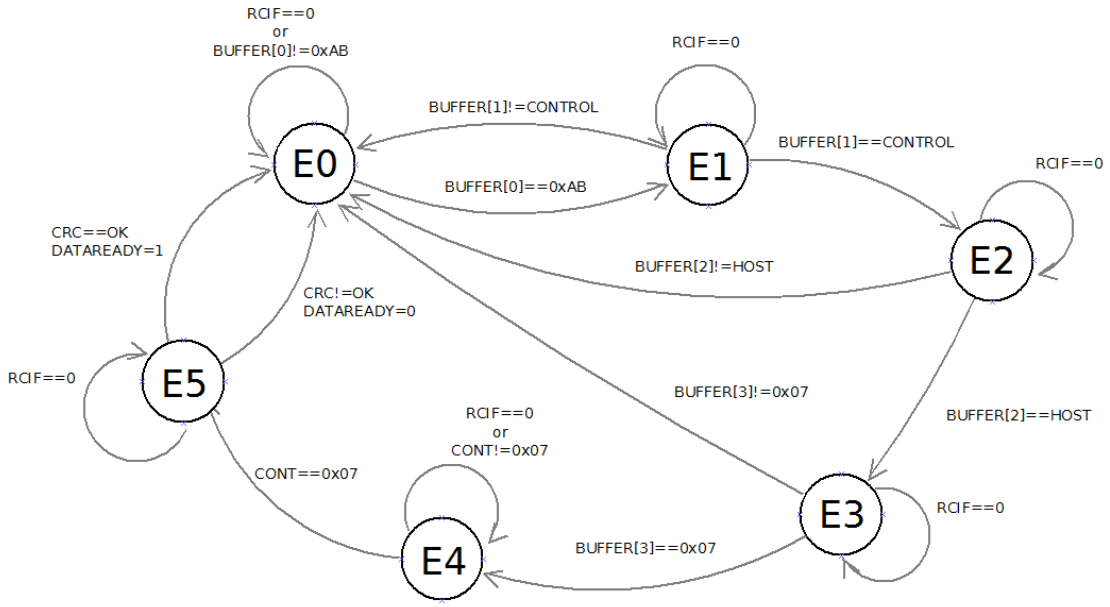


Ilustración 19 - Diagrama de bloques Read_serial

Enviar_tx

Si el flag de **serial** está activado, enviará a través del bus, el buffer de salida. Este buffer de salida debería de haber estado preparado por la función que activo el flag **serial**. El buffer tendrá el siguiente formato:

1º Parte del buffer destinado a los mandos

INICIO (1 byte)	DESTINO (1 byte)	ORIGEN (1 byte)	TIPO (1 byte)	DATOS (7 byte)	CRC (1 byte)

2º Parte del buffer destinado al Ezweb

"AT*NVREG" (8 bytes)	**Nº LUZ (1 byte)**	**"=" (1 byte)**	**MODO (1 byte)**

DESCRIPCIÓN DE LOS ESTADOS

- **E0:** Estado encargado de enviar la primera parte del buffer, una vez que haya terminado saltará al siguiente estado. Pondrá el estado del MAX485 en modo envío, el MAX485 no volverá al estado de recepción hasta que se termine de enviar la trama completamente. Antes de enviar cada byte se comprueba si el buffer de envío está libre, si no lo está volverá a la función principal. Cada vez que un byte haya sido enviado, se actualizará el valor del CRC. La primera parte del buffer estará compuesto de 11 bytes.
- **E1:** Se enviará el último byte de la trama que corresponderá al CRC en cuanto esté libre el buffer.

- **E2:** Añade un pequeño retardo necesario para que el microservidor Ezweb le dé tiempo en desechar la trama anterior y pueda procesar esta siguiente trama que se va a enviar.
- **E3:** Se enviará la segunda parte del buffer destinado al Ezweb compuesto de 11 bytes.
- **E4**: Se revisará si el buffer esta libre y se iniciará un contador.
- **E5:** Cada vez que se invoque este estado el contador se reducirá en 1. Una vez haya llegado a cero, se volverá a poner el MAX485 en modo recepción, y se reinicia todos los valores para preparar la función para el próximo envío.

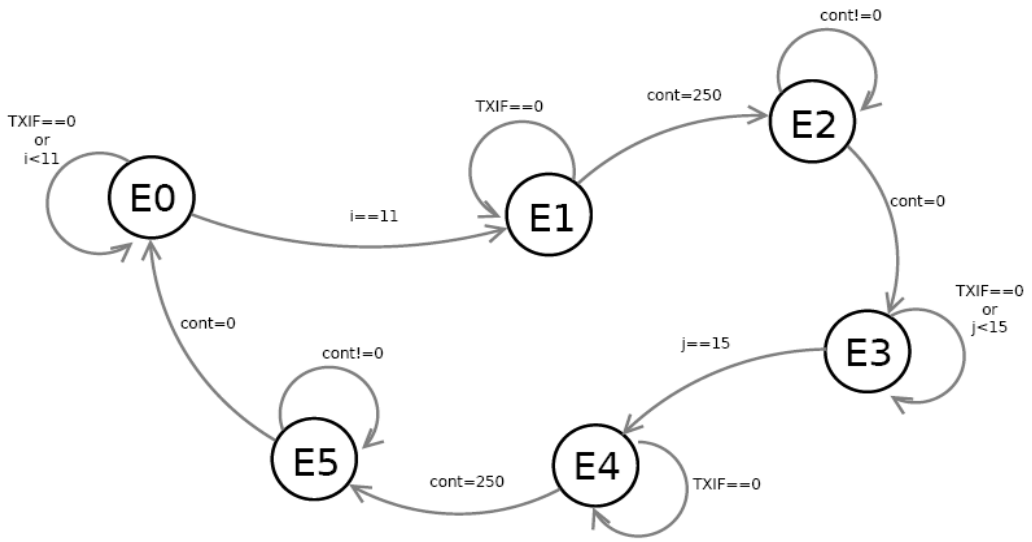


Ilustración 20 - Diagrama de bloques enviar_tx

Sensor

Función encargada de monitorear los sensores del juego de luces que se estén en modo auto. Cada vez que se detecte movimiento, se encenderá la luz y reiniciará el contador asignado a dicha luz. Cuando el contador llegue a 0, la luz asignada a dicho contador se apagará. En el caso de que la luz esté en modo on u off, la luz se quedará fija apagada o encendida.

Esta función contendrá un estado por cada sensor a monitorear, permitiendo así cierta escalabilidad a la hora de querer añadir sensores adicionales.

Evaluar_mando

Está función será la encargada de construir dos buffers, el primer buffer estará destinado para los mandos y el segundo para el microservidor Ezweb. Esto lo hará acorde a los datos recibidos por el puerto serial. Una vez que haya construido dicho buffer actualizará la variable interna MODO_SYNC, desactivará los **flags** de **dataready** y **eval_mando.**

- **E0:** Construcción del buffer de salida, se salta al estado E1

- **E1:** Identifica a que mando pertenece la última trama recibida y saltará al estado en consecuencia a ello.
    - E2 → Mando 1
    - E3 → Mando 2
    - E4 → Mando 3
    - E5 → Mando 4

- **E2 → E5:** Actualizará la variable MODO_SYNC y los últimos bytes de buffer de salida. Se vuelve al estado E0.

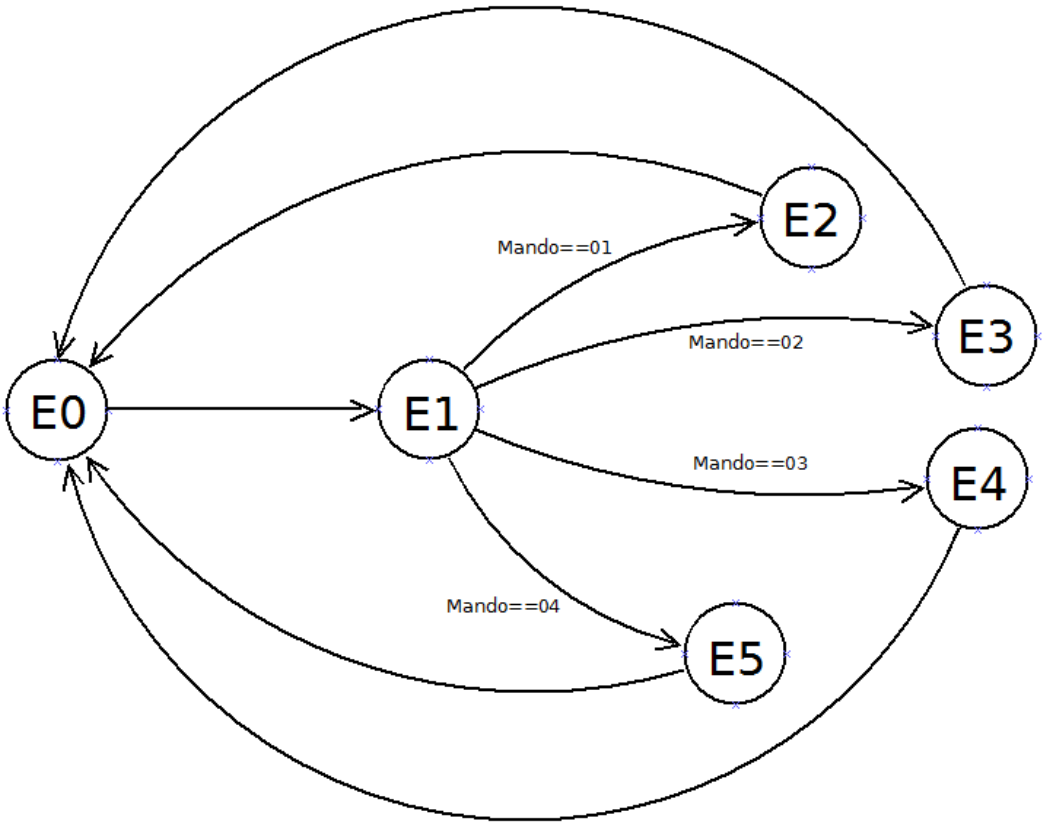


Ilustración 21 - Diagrama de bloques evaluar_mando

2.6 Software de la Placa Mando

2.6.1 Resumen

2.6.1 Resumen

La placa de mando tiene el objetivo de funcionar como un dispositivo inteligente e independiente en el sistema domótico. Esta placa de mando estará instalada en el espacio domótico a la accesible para los usuarios de dicho espacio, con los que podrá interactuará a través del botón que lleva instalado. En este prototipo solo se está empleando un botón haciendo la función de selección de modo de trabajo. En futuros diseños se podría implementar un teclado completo para así también aumentar su número de funciones.

La placa mando viene provista de un botón. Cada vez que el botón es pulsado por un usuario en la sala, la placa mando envía una trama con la instrucción requerida a la placa principal para que sea evaluada. La placa mando tiene la posibilidad de enviar comandos a través del canal de comunicaciones, recibirlos y desechar las tramas que no son destinadas a ella. En la simulación por software se trabaja con 4 placas mandos simultáneamente, aunque este número puede aumentar, siempre y cuando el número de placas totales conectadas al canal de comunicaciones no superé los 128, esto es una limitación que viene dada por utilizar la controladora MAX487.

La placa mando tiene el propósito de gestionar el modo en el que está trabajando el sistema de luz asignado. Este dispone de un botón, que una vez accionado este cambiará su estado, alternando entre los modos: auto, encendido y apagado. Cada vez que se acciona el botón, este enviará a través del bus una trama con dicho cambio, y también se indicará mediante el juego de led que la placa mando tiene integrado. Los mandos también cambiarían su estado en el juego de luces si recibe una trama que le indique que lo haga.

El desarrollo del software de la placa mando ha sido desarrollado en ensamblador, teniendo en cuenta los mismos criterios en su estructuración que los empleados en el desarrollo del software de la placa principal.

2.6.2 Descripción de autómatas

2.6.2 Descripción de autómatas

La placa mando cuenta con una serie de funciones para desarrollar su función, estas funciones se llamaran en un bucle infinito, y su ejecución será gestionado por indicadores (flags), para optimizar los ciclos del reloj del microcontrolador. Cada función, igualmente está separada en diferentes estados, ejecutándose un estado por cada llamada a la función, de esta forma conseguimos un sistema de multitarea al igual que en la placa principal. La ejecución de las tareas desarrolladas por las funciones vendrán delimitas por los siguientes indicadores.

Descripción de indicadores:

- **Dataready:** Si está activo indica que se ha recibido un dato que no se ha procesado.
- **Eval_rx:** Si está activo, indica que se tiene que identificar el tipo de dato recibido.
- **Receive:** Si está activo, indica que el dato recibido es del tipo mando, y necesita ser procesado como tal.
- **Click:** Se activará si se ha pulsado un botón.
- **Serial:** Indica que hay un buffer listo para ser enviado

Las funciones integradas en la placa mando son los siguientes:

Read_serial(): Está se encargará de leer en el bus datos si recibe alguna trama válida, si recibe una marcará el flag de **dataready** y **eval_rx**. Para que esta función se ejecutara necesitará que el flag de **dataready** y del de **serial** esté desactivado.

Evaluar_rx(): Esta función será la encargada de evaluar el tipo de datos recibidos. En este prototipo solo vamos a trabajar con datos del tipo mando, en el caso que esté reciba una trama de esto tipo desactivará el flag **eval_rx** y activará el flag **receive**.

Mando(): Está función se encargará de tratar los datos recibidos y actualizar la variable “modo” y el buffer de salida y de actualizar el sistema de led. Una vez que haya terminado desactivará los **flags** de **dataready, receive,** y según sus necesidades**,** activará el flag de **serial** en el caso el botón haya sido pulsado.

Enviar_tx(): Si el flag de **serial** está activado, enviará a través del bus, el buffer de salida.

Leer_teclado(): Función encargada de monitorear el botón, requiere que el flag de serial esté desactivado, si detecta una pulsación activará el flag de **clic.**

Clear_error(): Siempre en funcionamiento, función encargada de detectar errores en la eusart y corregirlo.

A continuación se detallará un diagrama de cómo son gestionadas dichas funciones con los indicadores, este es prácticamente igual al empleado en la placa principal.

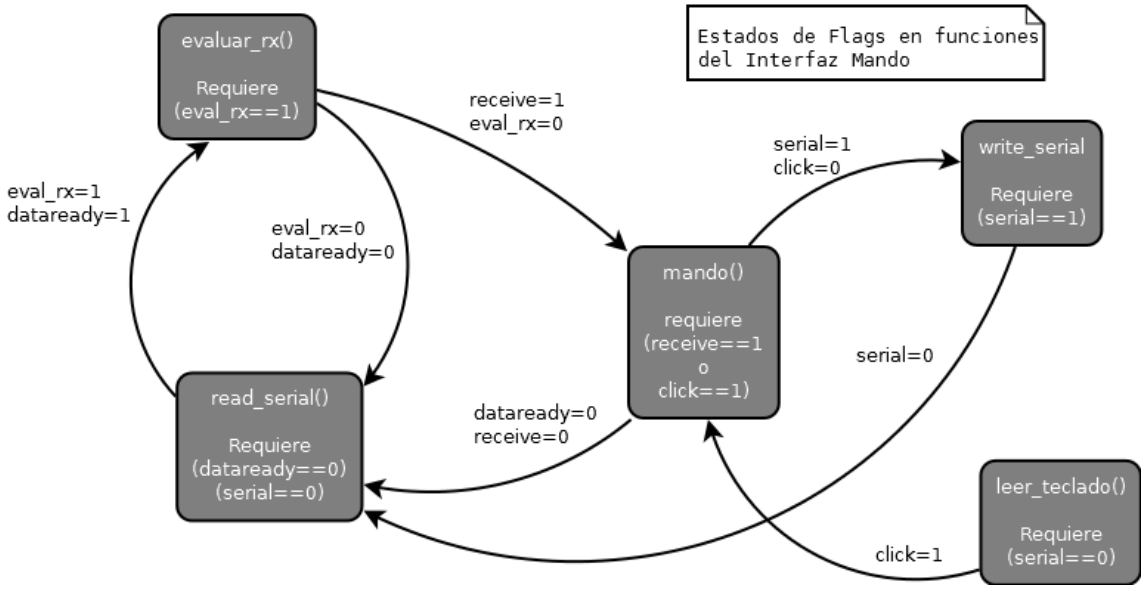


Ilustración 22 - Diagrama de cambios de flags en placa mando

2.6.3 Diagrama de bloques

2.6.3 Diagrama de bloques

Clear_error

Función encargada de detectar errores de sincronización de ventana del tipo "overrun". Los errores overrun tienen lugar cuando se produce un sobre desbordamiento de datos en el buffer de entrada.

Esta función siempre está en funcionamiento, sin importar el estado de los flags de esto. Si la función detecta un error de ventana o de desbordamiento, se encargará de reiniciar la EUSART y vaciar el buffer de entrada para corregir el problema y para que la aplicación siga funcionando con normalidad.

Esta función funciona idénticamente a la función clear_error empleada en el software de la placa principal.

DESCRIPCIÓN DE LOS ESTADOS

- **Estado Inicial:** Si detecta que el bit OERR en el registro RCSTA está a nivel alto, esta invocará la función RESET_OERR, si OERR está a nivel bajo la función terminará y volverá al bucle principal.
- **RESET_OERR:** Esta función se encargará de reiniciar la EUSART, y si detecta un dato en el buffer de entrada llamará a la función EMPTY_BUFFER
- **EMPTY_BUFFER:** Está función estará encargada de vaciar el buffer de entrada, no saldrá de su bucle hasta que el buffer de entrada esté totalmente vacío.

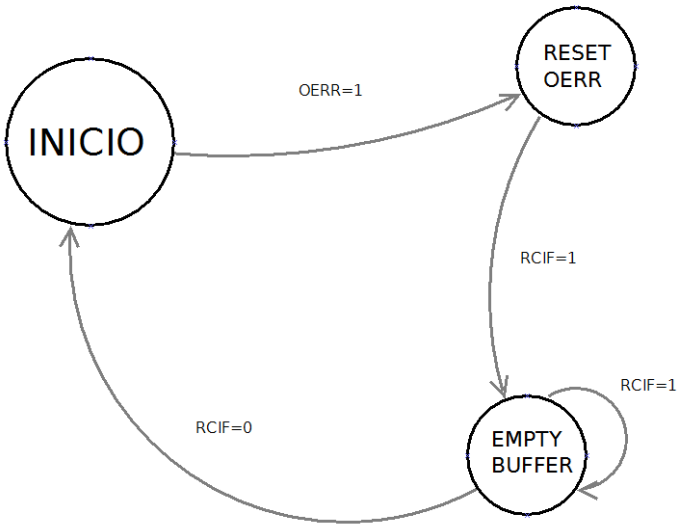


Ilustración 23 - Diagrama de bloques Clear_error

Evaluar_rx

Esta función leerá del buffer de entrada un byte en el caso que el flag **dataready** este activo, y según su valor actualizará la variable *modo* a los posibles estados *de auto, on* y *off*. Los valores posibles en el buffer de entrada serían los siguientes:

- 0x00 → Auto
- 0x01 → On
- 0x02 → Off

Si el byte es válido activará el flag **receive** para que el juego de led sea actualizado y el byte de **dataready** para marcar como que el dato ha sido evaluado, en caso contrario solo desactivará el byte de **dataready** para marcar que el buffer de entrada ya puede ser sobrescrito.

Read_serial

Para que esta función se ejecute, necesitaría que el flag de **dataready** y del **serial** esté desactivado. Función encargada de leer en el bus datos si recibe alguna trama válida a la placa en cuestión, dicha trama será desechada si no cumple los siguientes requisitos:

- No comienza con el byte de inicio de trama (0xAB)
- El campo destino de la trama no coincide con la dirección *host* de la placa
- La trama no tiene un tamaño de datos válido
- El valor del CRC (suma de verificación), no coincide con el calculado en la recepción

En el caso de ser válida se marcará el flag de **dataready** y **eval_rx**.

DESCRIPCIÓN DE LOS ESTADOS

- **E0:** Estado de detectar el inicio de trama, no irá al siguiente estado hasta que le llegue el byte 0xAB.
- **E1:** Estado de detectar el si la trama está destinada a la placa receptor. La dirección del receptor se define en los parámetros iníciales bajo la variable HOST. Si no está destinada a esta placa se vuelve al estado inicial E0, en caso contrario se iría al siguiente. En este estado se inicia el cálculo del CRC (suma de verificación).
- **E2:** Estado encargado de detectar la dirección del emisor de la trama. La placa sólo permitirá la recepción de tramas de la dirección definida bajo la variable CONTROL definida en los parámetros generales. En caso de estar permitida se salta al siguiente estado y se actualiza el CRC, en caso contrario se vuelve al estado inicial E0.
- **E3:** Estado encargado de recibir la longitud del campo datos. El sistema solo permite campos de datos de longitud 7. Si es diferente a este, se volverá al estado E0, en caso contrario se salta al siguiente estado y se actualiza el valor del CRC.
- **E4:** Estado encargado de almacenar el los 7 bytes del campo datos en el buffer de entrada. Una vez estén los 7 bytes, se saltará al siguiente estado y se actualizará el valor del CRC.

- **E5:** Estado de comprobación del CRC. Se compara el CRC recibido con el calculado localmente, si coinciden se marcará el flag de **dataready** a 1, de lo contrario, se dejará en 0. Se vuelve al estado inicial E0.

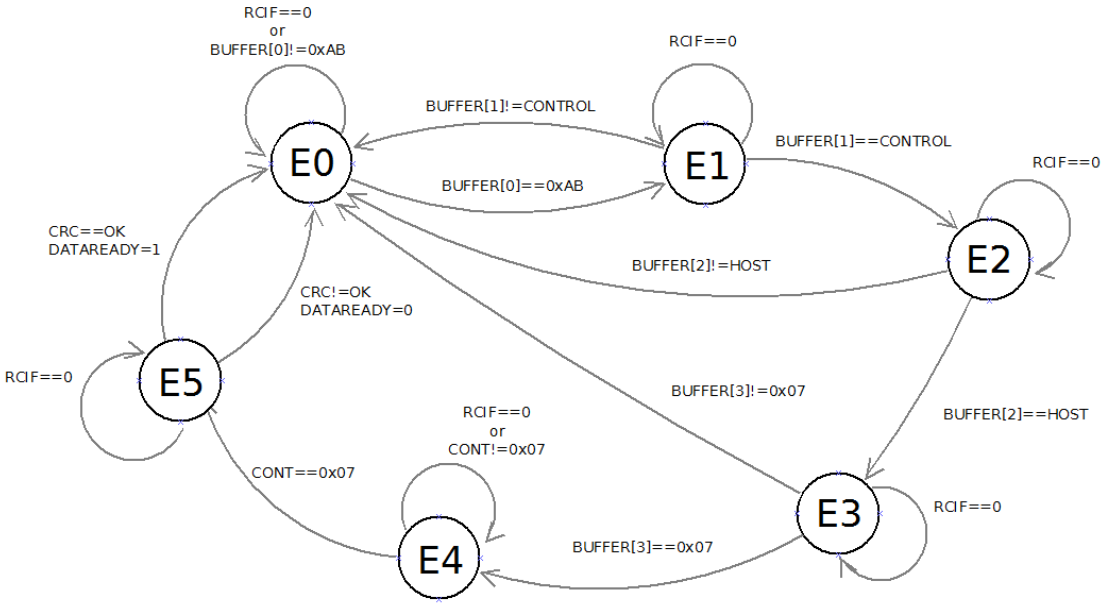


Ilustración 24 - Diagrama de bloques Read_serial

Enviar_tx

Si el flag de **serial** está activado, enviará a través del bus, el buffer de salida. Este buffer de salida debería de haber estado preparado por la función que activo el flag **serial**. El buffer tendrá el siguiente formato:

INICIO (1 byte)	DESTINO (1 byte)	ORIGEN (1 byte)	TIPO (1 byte)	DATOS (7 byte)	CRC (1 byte)

DESCRIPCIÓN DE LOS ESTADOS

- **E0:** Estado encargado de enviar la primera parte del buffer, una vez que haya terminado saltará al siguiente estado. Pondrá el estado del MAX485 en modo envío, el MAX485 no volverá al estado de recepción hasta que se termine de enviar la trama completamente. Antes de enviar cada byte comprobará si el buffer de envío está libre, si no lo está volverá a la función principal. Cada vez que un byte haya sido enviado, se actualizará el valor del CRC. La primera parte del buffer estará compuesto de 10 bytes.
- **E1:** Se enviará 1 byte al comprobar que el buffer está libre, que corresponderá al modo que está trabajando al juego de luces. Posteriormente actualizará el valor del CRC.
- **E2:** Se enviará el último byte de la trama que corresponderá al CRC en cuanto esté libre el buffer.

- **E3**: Se revisará si el buffer esta libre y se iniciará un contador.
- **E4:** Cada vez que se invoque este estado el contador se reducirá en 1. Una vez haya llegado a cero, se volverá a poner el MAX485 en modo recepción, y se reinicia todos los valores para preparar la función para el próximo envío.

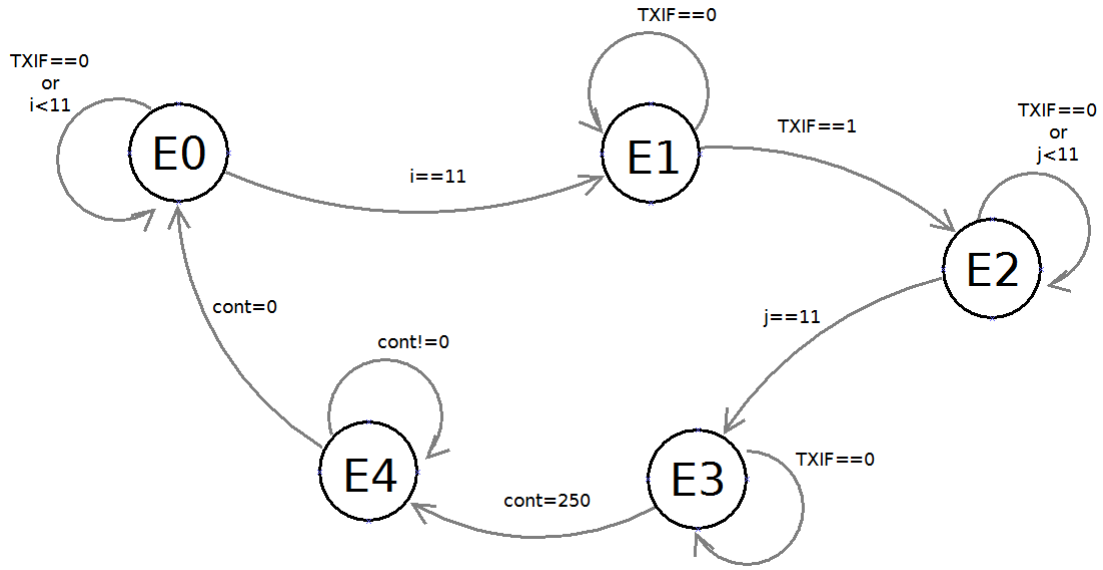


Ilustración 25 - Diagrama de bloques Enviar_tx

Leer_teclado

Función encargada de monitorear el botón, requiere que el flag de serial esté desactivado, de esta forma evitamos algunos errores en la trasmisión. Si detecta una pulsación, en el momento que se deje de pulsar el botón la función se encargará de activar el flag de **clic.** Esta función se podría actualizar en el futuro y ampliar el número de estados para detectar varios botones de un teclado para realizar otras diversas tareas.

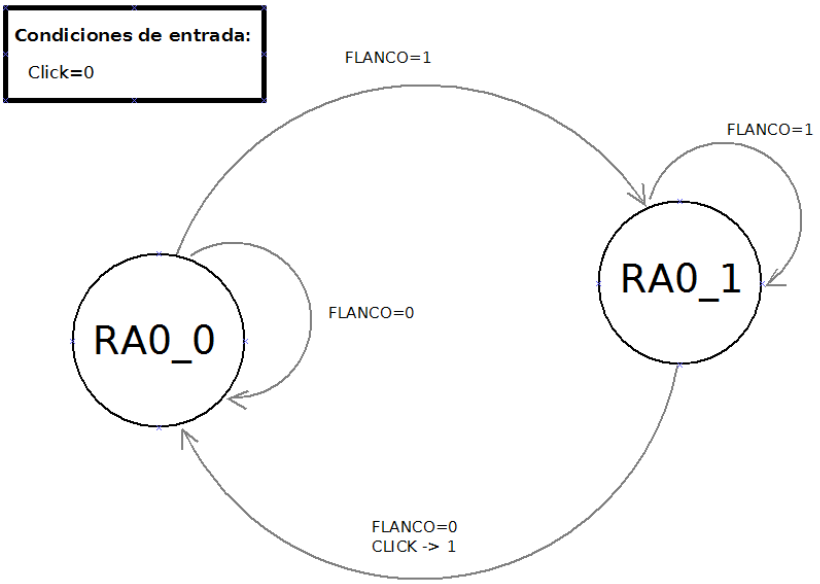


Ilustración 26 - Diagrama de bloques de Leer_teclado

Mando

Función que se encargará de tratar los datos recibidos (sea por mando o por bus de datos) y de actualizar la variable “modo”, el buffer de salida y actualizar el sistema de led. Una vez que haya terminado desactivará los flags de **dataready, receive** y **clic** y activará el flag de **serial** en el caso el botón haya sido pulsado para que notifique a los demás interfaces de dicho cambio.

2.7 Software de la Placa Web

2.7.1 Resumen

2.7.1 Resumen

La aplicación que tiene integrada la placa web tiene la función de facilitar la gestión y la monitorización del conjunto del sistema domótico remotamente gracias a una página web accesible para cualquier dispositivo informático (tablet, móvil u ordenador) provistas de una conexión internet. La página web permitirá al usuario revisar en qué modo está operando cada luz (ON, OFF y AUTO) como alternar cada uno de estos modos de forma independiente. Para ello, como se puede apreciar en la Ilustración 27, la página web ofrecida por el servidor web muestra un esquemático del espacio domótico a gestionar donde se puede visualizar en qué lugar están dispuestas cada luz dentro del espacio. Como un juego de 4 botones, uno por cada luz, en la parte inferior con lo que se puede gestionar cada luz independientemente. Al pulsar sobre cualquiera de los botones, esté cambiará el modo de su luz asignada, alternando entre On, Off y Automática. Estas notificaciones serán enviadas a la placa principal para que este se lo comunique al resto de los dispositivos.

El microservidor web Lynx Ezweb está conectado al bus rs485 para comunicarse con la placa principal, y a su vez también a un enrutador de red a través del conector Ethernet que incluye el propio Ezweb para interactuar con internet. Se podría acceder a dicho servidor web a través de la dirección ip local 192.168.1.60 y puerto 80 usando el protocolo http, para redes externas se debería de establecer un dominio o una ip pública fija. La página web se podrá cargar con cualquier navegador con soporte HTML5.

A la hora de desarrollar la aplicación web, por los requerimientos del propio microservidor web Lynx Ezweb, los nombres de los archivos que componen el proyecto no pueden exceder de 9 letras para su nombre, y 3 para su extensión, tampoco se permite crear carpetas, y para que los archivos se reconozcan como páginas web deberán de tener obligatoriamente la extensión HTM.

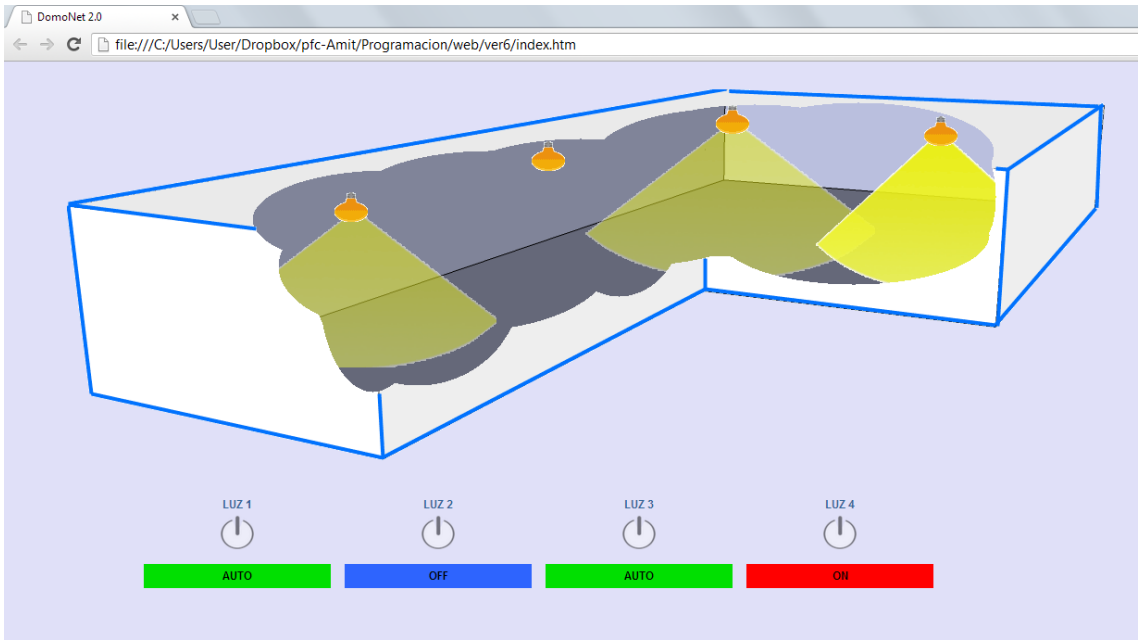


Ilustración 27 - Página web hospedada en Ezweb

Cada una de las luces mostradas en la página web está asociada a un botón por software, cada vez que se hace clic sobre uno de estos botones invocará uno de los eventos que alternará el modo de trabajo de su luz asignada entre ON, OFF y AUTO en dicho orden, al igual que cuando se pulsa al botón físico de una placa mando. Cada vez que hay un cambio en el modo de trabajo de una luz, este será comunicado a la placa principal a través del puerto serie, una vez que la placa principal haya aceptado dicho cambio este escribirá dentro del microservidor web en uno de sus registros de memoria no volátiles el estado de todas las luces, por lo que cuando se vuelva a recargar la página se verá el estado actualizado de todas las luces. Se ha optado en escribir el estado de las luces del sistema en la memoria no volátil del microservidor Ezweb respecto a la volátil para evitar perder datos en caso de cortes de electricidad inesperados.

En el caso que uno de los mandos inteligentes solicite un cambio de modo de trabajo en una de las luces, este comunicará, como se explico previamente, a la placa principal para que este notificar el cambio al servidor web escribiendo en su memoria no volátil. Para que este cambio se pueda apreciar en la página web el usuario deberá de recargar la página web de nuevo. Se descartó la posibilidad de programar un segundo script secundario para que se ejecutase en segundo plano en la aplicación para que recargase la página web cada X segundos de forma automática por que dicha función provocaba a veces bloqueos en el microservidor web por su limitación de procesado.

Para optimizar y acelerar la carga de la página web las imágenes empleadas han pasado por un procesado de indexado de color previo para reducir así su tamaño y poder hacer las transferencias de estas en un menor tiempo, ya que el Ezweb, por su limitación de hardware no dispone de una capacidad de procesado muy elevada, a veces provocando estados de inactividad durante varios segundos cuando necesita ejecutar varias tareas simultáneamente.

El conjunto de la aplicación web integrada dentro del microservidor Lynx Ezweb está compuesto por los siguientes archivos, en el caso que se necesite de acceder a cualquiera de los archivos independientemente dentro del servidor usando un navegador se podrá acceder siguiendo la siguiente estructura:

http://dirección_servidorweb/nombre_del_archivo

Por ejemplo, si se quisiera reiniciar el servidor introduciríamos lo siguiente en la barra de direcciones en nuestro navegador:

http://192.168.1.60/reset.htm

Descripción de archivos

- **index.htm:** Archivo principal cargado inicialmente de forma predeterminada al acceder al servidor web, esta hace llamada a las funciones requeridas y carga las imágenes.
- **error404.htm:** Página que devuelve si se accede a una dirección no encontrada en el servidor web.
- **config.htm:** Archivo de configuración, cada vez que se acceda a esta web configurará el Ezweb con los parámetros aquí indicados. Será necesario modificar este archivo, y

posteriormente ejecutado si se requeriré cambiar alguna configuración del servidor el futuro.

- **reset.htm:** Reinicia por software el Ezweb, será necesario si hacemos algún cambio de configuración, o si el dispositivo sufre de un bloqueo.
- **style.css:** Hoja de estilo utilizada en Index.htm donde se detalla los colores y estilos de tablas.
- **scripts.js:** Funciones que llama Index.htm encargada en la construcción de tramas de datos, lectura de estado de la memoria no volátil, y actualización del diagrama del espacio domótico.

Las instrucciones internas destinadas al Ezweb no se pueden integrar en instrucciones de Javascript para que se ejecute en los casos que fueran necesarios, ya que el Ezweb separa el contenido HTML y las instrucciones internas y las ejecuta individualmente. Por lo que se crearon otros archivos con extensión htm con las diferentes instrucciones necesarias para poder llamarlas a modo de pop ups a través de comandos Javascript a la medida que vayan haciendo falta.

Los siguientes archivos contienen pequeñas instrucciones internas del Ezweb:

- **bulb1.htm:** Envía 1º parte al bus, con destino la luz 1
- **bulb2.htm:** Envía 1º parte al bus, con destino la luz 2
- **bulb3.htm:** Envía 1º parte al bus, con destino la luz 3
- **bulb4.htm:** Envía 1º parte al bus, con destino la luz 4
- **auto.htm:** Envía 2º parte al bus, configurando el modo de la luz en AUTO
- **on.htm:** Envía 2º parte al bus, configurando el modo de la luz en ON
- **off.htm:** Envía 2º parte al bus, configurando el modo de la luz en OFF
- **pin1_0.htm:** Pone el pin1 del Ezweb a nivel bajo. Para poner la controladora de comunicaciones serie MAX487 conectada el servidor web en modo recepción.
- **pin1_1.htm:** Pone el pin1 del Ezweb a nivel alto, poniendo la controladora MAX487 en modo envío.

Las tramas enviadas a la placa principal, serán idénticas a las enviadas por las placas de mando, por lo que el servidor necesitará enviar tramas para gestionar 4 luces distintas alternando en sus 3 modos posibles. Para ello se ha separado cada trama a enviar en 2 pedazos, la primera donde albergará el número de la luz a gestionar, y el segundo, su modo de trabajo a cambiar. Estos pedazos de tramas tienen que estar predefinidas en distintos archivos con extensión htm para que sean llamadas según el caso necesario. Por ejemplo, en el caso que se necesite encender la luz 3, la aplicación web llamará a los siguientes archivos en el orden descrito:

1. **pin1_1.htm:** Activando modo envío
2. **bulb3.htm:** Envío de la 1º parte de la trama, para gestionar la luz nº3
3. **on.htm:** Envío de la 2º parte de la trama para encender la luz.
4. **pin1_0.htm:** Activando modo recepción

2.7.2 Descripción de funciones

2.7.2 Descripción de funciones

La página web, que a su vez funciona como panel de control y monitorización del sistema domótico requerirá de ciertas funciones para ir actualizando el diagrama del espacio web mostrando el estado de la iluminación como el envío de datos al canal de comunicaciones cada vez que un pulsador de esta pulsado, para ello dentro de la librería del archivo **scripts.js** se podrá encontrar las funciones requeridas para el correcto funcionamiento del sistema. Las funciones fueron escritas en Javascript que se ejecutan por parte del cliente, liberando parte del procesado total de la página web al servidor.

Algunas de las funciones dentro de la librería necesitaran la recepción de dos variables:

Estado: modo al que se desea cambiar

Posibles valores= [*"ON", "OFF", "AUTO"*]

Bombilla: luz destinada dicho cambio

Posibles valores= [*"bulb1", "bulb2", "bulb3", "bulb4"*]

Lista de funciones dentro de la librería **script.js**:

- **Pulsar(estado, bombilla):** Esta función recibirá en *estado* el identificador del controlador que la invoca, y en *bombilla* la luz a la que está cambiando. Cada vez que se pulsa un botón asignada a una bombilla específica, esteré alternará su modo entre los 3 posibles siguiendo la siguiente secuencia:

  Apagar → Encender → Automático → Apagar → Encender → ….

- **Apagar(estado, bombilla):** Subfunción usada dentro de Pulsar(), encargada de apagar la bombilla especificada y actualizar su indicador asociado. Para ello invocará a los siguientes archivos htm en modo como ventanas emergentes en donde se albergan instrucciones específicas del Ezweb para interactuar con el puerto serie.

  pin1_1.html, bulb[n][11].htm, off.htm, pin1_0.htm

- **Encender(estado, bombilla):** Subfunción usada dentro de Pulsar(), encargada de encender la bombilla especificada y actualizar su indicador asociado. Para ello invocará a los siguientes archivos htm al igual que la función anterior para interactuar con el puerto serie:

  pin1_1.html, bulb[n].htm, on.htm, pin1_0.htm

[11] Vendrá definida por el valor de ***bombilla***

- **Automático(estado, bombilla):** Subfunción usada dentro de Pulsar(), encargada de poner en modo automático la bombilla especificada y actualizar su indicador asociado. Para ello invocará a los siguientes archivos htm para interactuar con el puerto serie:

  pin1_1.html, bulb[n].htm, auto.htm, pin1_0.htm

- **Init():** Función ejecutada al cargar la página index.htm. Su función es leer en los registros internos no volátiles del microservidor Ezweb el estado actual de los modos que las luces del espacio domótico están operando. Este registro será actualizado por la placa principal.

- **Leer_rs232():** Esta función tiene el objetivo de leer por el puerto serie que datos se ha recibido y mostrarlos en pantalla. Esta función se utilizó en la fase de pruebas, pero para no tiene ninguna utilidad el prototipo actual. Se ha mantenido dicha función por si es requerida en futuras posibles actualizaciones.

2.9 Recursos

2.9 Recursos

A continuación se hace una breve descripción de las herramientas y tecnologías que han contribuido al desarrollo y estudio en el diseño y fabricación del sistema domótico.

Cabe destacar, que se ha hecho un gran esfuerzo en emplear tecnologías comunes y familiares para la mayoría de los desarrolladores, como puede ser el lenguaje ensamblador para el desarrollo del software en los microcontroladores, HTML5 para servir la página web, o el protocolo IEEE 802.3, formato en el que se baso para el diseño de la construcción de las tramas de datos empleadas en las comunicaciones.

En el caso de las aplicaciones utilizadas se ha intentado utilizar herramientas con licencia de código abierto o libre, pudiendo así obtener dichas aplicaciones a través de internet a coste 0, como *"Dia Diagram"* con licencia de código abierto para la elaboración de los diagramas o Sketchup, aplicación con licencia libre empleada para el diseño de las vistas del espacio domótico empleado en la web.

Como hardware para la programación de los microcontroladores empleamos PicKit 2, un interfaz con licencia de código abierto ya que se reconoce como el más completo en cuestión de calidad precio del mercado, pero no necesariamente se está ligado utilizar dicho hardware.

En dos casos se tuvo que emplear software comercial, ya que no se encontró un equivalente con licencia código abierto/libre que reuniera la mismas funcionalidades y calidad que estas, como fue el Photoshop, empleado para los últimos retoques y creación de imágenes adiciones para la página web, y Proteus Isis, la aplicación empleada para el desarrollo y simulación de los circuitos eléctricos.

2.9.1 MPLab 8

MPLAB es el editor IDE (con placa gráfica) empleado para el desarrollo y la compilación del código tanto para los microcontroladores PIC16F690, como para la unidad principal y los sistemas de mando. La aplicación viene desarrollada por la empresa Microchip, la misma que manufactura los microcontroladores y viene solo para versión Windows. Es gratuito y se puede descargar desde su web oficial:

http://www.microchip.com

La aplicación permite la programación en ensamblador y en lenguaje C de forma nativa, aunque se puede utilizar otros lenguajes empleando otros terceros programas. El compilador crea un archivo con extensión .HEX y otro .COF. El archivo con la extensión .HEX es el empleado para la programación del controlador a través del programador PICkit 2 y su respectiva aplicación. El archivo con extensión .COF se emplea para las simulaciones efectuadas con la aplicación PROTEUS, ya que facilita su seguimiento y corrección de fallos de una forma más sencilla.

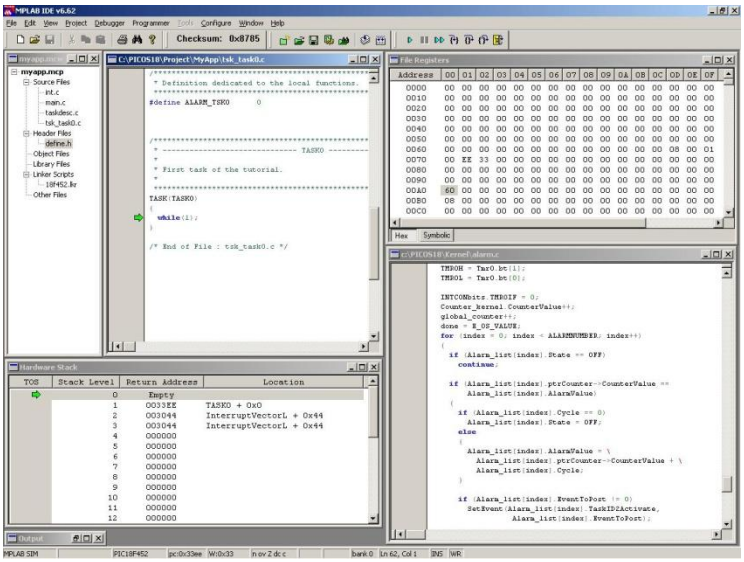


Ilustración 28 - Aplicación MPLab

2.9.2 Lenguaje Ensamblador

Es el lenguaje empleado para los programas contenidos en los microcontroladores PIC16F690. El lenguaje ensamblador es un lenguaje de bajo nivel, donde su inicio comenzó sobre el año 1950. Los programas realizados en ensamblador están muy ligados al procesador para el que se está programando, lo que dificulta, por un lado, que un programa realizado para un procesador determinado sea migrado a otro; pero por otro lado, permite una mayor control por lo tanto la posibilidad de optimizar el programa para el procesador con el que estamos trabajando. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. La sintaxis seguida en una instrucción sería el siguiente:

[etiqueta:] *nemónico* **[operando] [,operando] [;comentario]**

Ejemplo:

```
Inicio_tx          ;inicio de trasmisión
movlw 0x30         ;cargamos el valor 30 al acumulador
movwf PORTC        ;cargamos el dato del acumulador al registro PORTC
end                ;fin
```

En este pequeño código hemos cargado el dato 30 (hexadecimal) al acumulador y posteriormente lo hemos cargado de nuevo al registro PORTC, en nuestro caso el registro PORTC lleva el estado de los pines del microcontrolador RC0 al RC7.

2.9.3 PICKit 2

El PICKit 2 se introdujo en el año 2005, y es la placa empleada para programar los archivos con extensión. HEX creados con la aplicación MPLAB en los microcontroladores de 8 y 16 bits fabricados por Microchip. El PICkit está compuesto por un programador y un *debugger* y utiliza un PIC18F2550 internamente y una placa USB para la conexión a nuestro PC. La placa PICKIT 2 es un sistema totalmente libre. Desde su hardware, firmware y software y recibe varias contribuciones para mejorar su funcionalidad de una amplia comunidad de desarrolladores, una de las más destacadas fue realizada por Au Group Electronics donde mejoró el tamaño máximo de memoria permitido a 4 megas en la función "programer-to.go ", lo que permite que una imagen de memoria de un microcontrolador se descargue directamente al placa PICKit para que posteriormente pueda ser reprogramado en otro microcontrolador PIC sin emplear un ordenador.

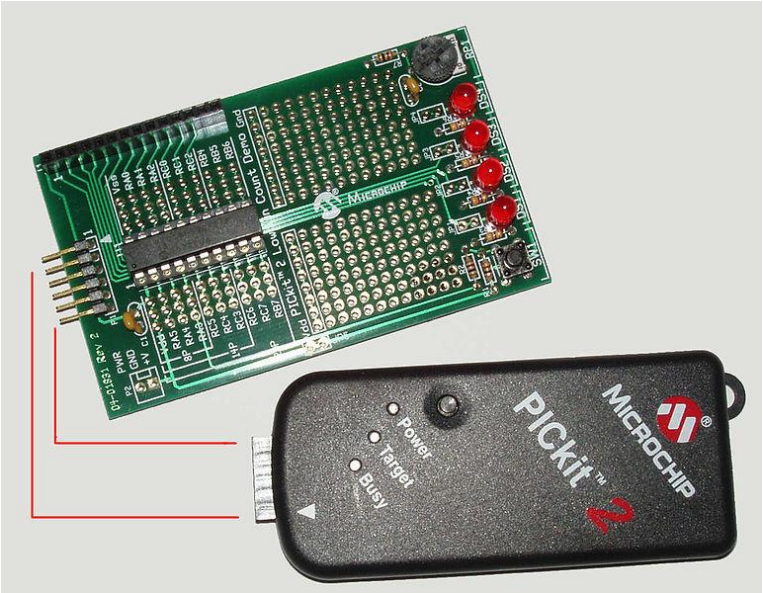


Ilustración 29 - Hardware de Pickit2

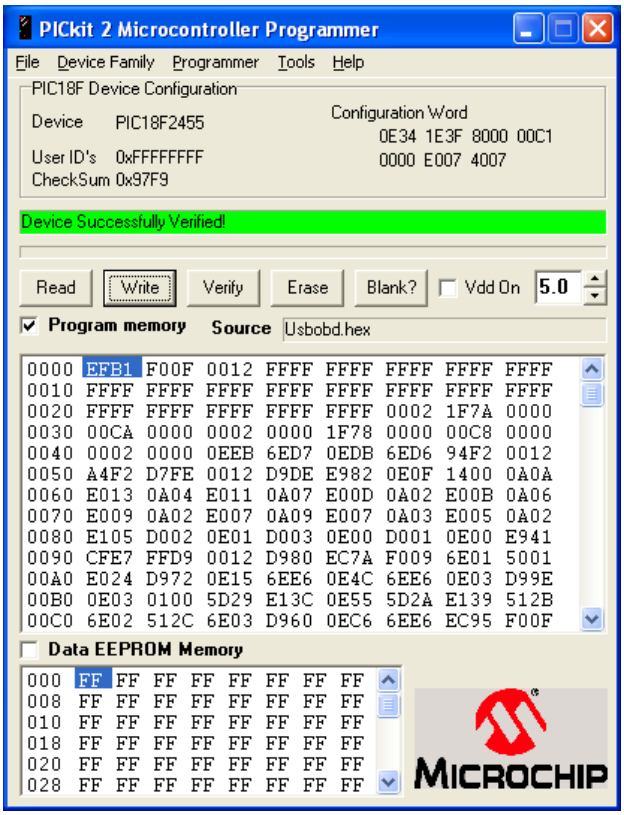


Ilustración 30 - Software Pickit2

2.9.4 Proteus Isis

Proteus es un paquete de software desarrollado por LabCenter Electronics orientado al diseño y simulación electrónica. Sus dos programas principales son Ares e Isis, Ares está orientado al diseño de PCB e ISIS a la simulación. En este proyecto se trabajo con la aplicación ISIS para hacer las simulaciones previas antes de la construcción de los prototipos.

El Programa ISIS, o Intelligent Schematic Input System (Sistema de Enrutado de Esquemas Inteligente) brinda la posibilidad de diseñar el plano eléctrico del circuito que se desea realizar empleando distintos componentes y microcontroladores, dando incluso la posibilidad de interactuar con un PC a través de su puerto serie y tomar medidas a través de un osciloscopio virtual. Los diseños realizados en Isis pueden ser simulados en tiempo real.

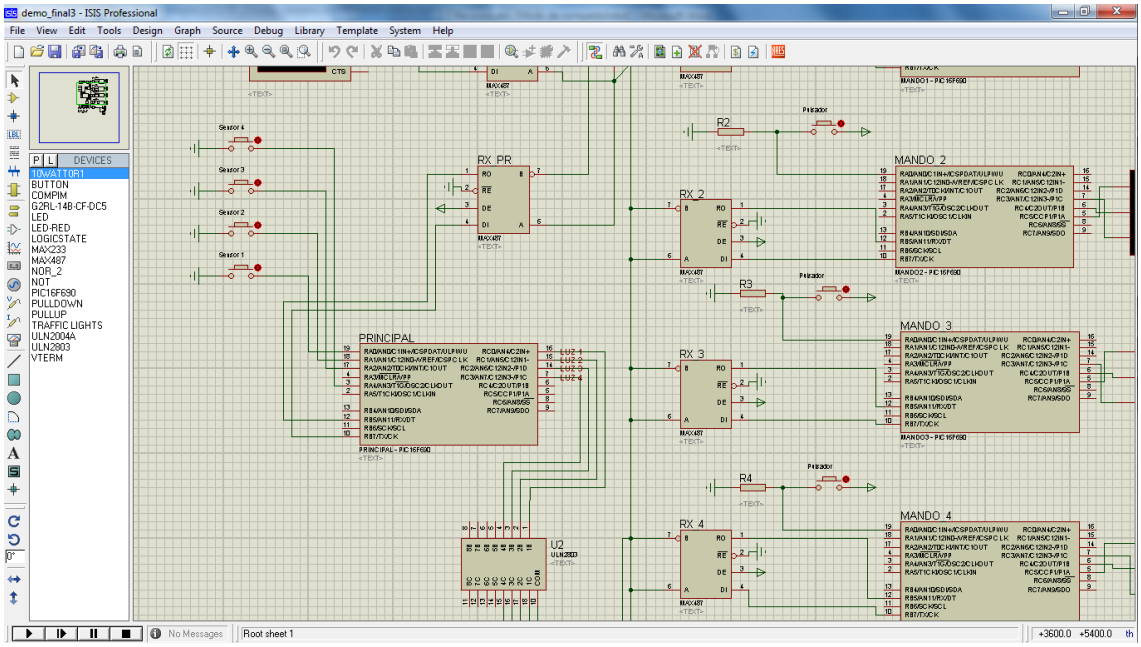


Ilustración 31 - Simulador Isis Proteus

2.9.5 HTML5

Las aplicaciones webs programadas dentro de los microservidores Lynx Ezweb están programadas bajo los estándares de HTML5.

HTML5 (HyperText Markup Language, versión 5), en conjunto con CSS3, define los nuevos estándares de desarrollo web para distintos sistemas informáticos que puedan ejecutar un navegador web. HTML5 es la quinta revisión importante del lenguaje básico de la World Wide Web, HTML. En él se especifica dos variantes de sintaxis para HTML: un «clásico» HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida en XML (XHTML) (application/xhtml+xml).

El HTML solamente formatea y muestra contenidos, y no tiene la capacidad de interactuar con variables, bases de datos, etc. Aún los sitios web más complejos, que utilizan grandes sistema de bases de datos y realizan complejas tareas, envían al navegador (Mozilla Firefox, Chrome, etc.) puro código HTML.

Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.

Todavía se encuentra en modo experimental, lo cual indica la misma W3C; aunque ya es usado por múltiples desarrolladores web por sus avances, mejoras y ventajas.

Al no ser reconocido en viejas versiones de navegadores por sus nuevas etiquetas, se le recomienda al usuario común actualizar a la versión más nueva, para poder disfrutar de todo el potencial que provee HTML5.

El desarrollo de este lenguaje de marcado es regulado por el Consorcio W3C.

Principales ventajas de HTML

- Simplificación: El nuevo código ofrece nuevas formas, más sencillas, de especificar algunos parámetros y piezas de código.
- Contenido multimedia: Reproducción de audio y video sin necesidad de plug-ins.
- Animaciones: Posibilidad de mostrar contenidos de manera similar a Adobe Flash, pero prescindiendo de este componente. HTML5 tendrá soporte nativo para una tecnología similar a Flash.
- Almacenamiento de datos del lado del cliente: Una diferencia fundamental entre las aplicaciones web era la necesidad, de éstas últimas, de procesar la información y consultas en bases de datos siempre en un servidor, haciendo que las aplicaciones sean más lentas y siempre requeridas de una conexión a Internet constante. HTML5 permitirá almacenar y procesar información en el cliente, convirtiendo a una aplicación web a una aplicación mucho más parecida a una de escritorio.
- Efectos y nueva versión de hojas de estilo CSS: La nueva versión de HTML acompañará a una nueva versión de las hojas de estilo CSS, el CSS3. Permite nuevas posibilidades de formato, por ejemplo la implementación de transparencias, sombras, bordes redondeados, etc.

- Geo-locación: Los sitios web podrán saber la ubicación del ISP (proveedor de servicios de internet) que lo visita, por lo que se podrá estimar la situación del usuario.
- Tipografías no estándar: Hasta ahora, quizás la mayor limitación que enfrentábamos los diseñadores era la imposibilidad de utilizar tipografías no-estándar en nuestros sitios web. Prácticamente todos estaban limitados a aquellas que fueron impuestas por los navegadores principales, por ejemplo: Times New Roman, Verdana, etc. La implementación de sistemas como Google Fonts hoy permite emplear muchas otras.

Ejemplo de HTML5:

```
<!DOCTYPE HTML>
<html>
  <head>
    <title>Título ejemplo</title>
  </head>
  <body>
        ¡Hola proyecto de Domótica!
  </body>
</html>
```

Ejemplo de código en HTML5 con instrucciones en javascript:

```
<!DOCTYPE HTML>
<html>
 <head>
  <title>Calculo de números primos</title>
 </head>
 <body>
  <p>El número primo más alto calculado por ahora es: <output id="result"></output></p>
  <script>
   var numero = new Worker('funciones_primos.js');
   numero.onmessage = function (event) {
   document.getElementById('result').textContent = event.data;
   };
  </script>
 </body>
</html>
```

// funciones_primos.js (fichero con la tarea del nuevo hilo de ejecución infinito)

```
var n = 1;
search: while (true) {
  n += 1;
  for (var i = 2; i <= Math.sqrt(n); i += 1)
    if (n % i == 0)
      continue search;
  // Primo encontrado!
  postMessage(n);
}
```

2.9.6 DIA Diagram

DIA está diseñado como un sustituto de la aplicación comercial Visio de Microsoft y fue lanzada bajo licencia de código abierto (GPL). Se puede utilizar para dibujar diferentes tipos de diagramas. Actualmente se incluyen diagramas entidad-relación, diagramas UML, diagramas de flujo, diagramas de redes, diagramas de circuitos eléctricos, etc. Nuevas formas pueden ser fácilmente agregadas, dibujándolas con un subconjunto de SVG e incluyéndolas en un archivo XML, el formato para leer y almacenar gráficos es XML (comprimido con gzip, para ahorrar espacio). Puede producir salida en los formatos EPS, SVG y PNG.

También conviene recordar que DIA, gracias al paquete dia2code, puede generar el esqueleto del código a escribir, si utilizáramos con tal fin un UML.

DIA es una aplicación de creación de diagramas bajo licencia de código abierto (GPL), la aplicación está basada en otra aplicación comercial conocida como “Visio” desarrollada por Microsoft. Las principales funciones que posee DIA es el dibujar diagramas orientadas al software, como diagramas UML, flujo, red, etc... DIA ya dispone de una librería de objetos, pero es posible ampliar dicha librería con solo crear un archivo XML con sus parámetros y la imagen en SVG, también permite la opción de programar módulos adicionales programados en Python.

DIA permite exportar también los diagramas creados en varios formatos para hacer impresiones multi-página. Los formatos soportados para su exportación son los siguientes:

- EPS (Encapsulated PostScript)
- SVG (Scalable Vector Graphics)
- DXF (Autocad's Drawing Interchange format)
- CGM (Computer Graphics Metafile defined by ISO standards)
- WMF (Windows Meta File)
- PNG (Portable Network Graphics)
- JPEG (Joint Photographic Experts Group)
- VDX (Microsoft's XML for Visio Drawing)

Esta aplicación fue usada en el proyecto para la elaboración de diagramas de flujos del código utilizado en los distintos microcontroladores.

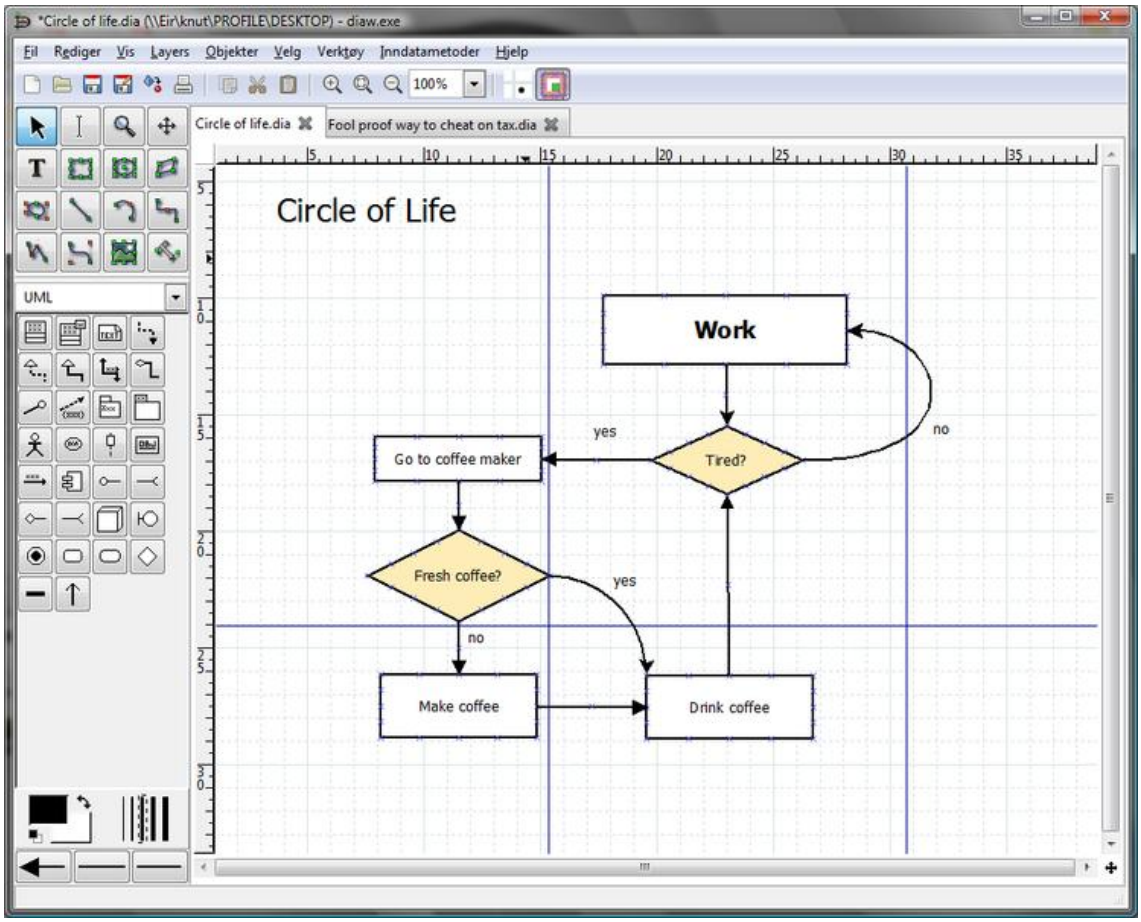


Ilustración 32 - Aplicación Dia

2.9.7 Trimble SketchUP

Aplicación gratuita de diseño gráfico y modelado en tres dimensiones (3D) basado en caras, para entornos de arquitectura, GIS, videojuegos, ingeniería civil, diseño industrial, o películas. La aplicación esta desarrollado por Last Software, empresa adquirida por Google en 2006 y finalmente vendida a Trimble en 2012.

SketchUp destaca por usarse de una manera intuitiva como simple. El programa incluye en sus recursos un tutorial en vídeo para ir aprendiendo paso a paso cómo se puede ir diseñando y modelando un entorno, llegando a modelar edificios, coches, personas y cualquier objeto en 3D. La aplicación también proporciona acceso a un repositorio de objetos y texturas para descargar de forma gratuita.

Los modelos creados en SketchUP se pueden exportar en los formatos .dae y .kmz (Google Earth), o bien tomar pantallazos en formato .png o jpg. La versión no gratuita conocida como Trimble SketchUP Pro permite la exportación en otros formatos adicionales.

SketchUP también permite la opción de desarrollar plugins en lenguaje Ruby para ampliar sus funcionalidades, se pueden encontrar varios de estos plugins de forma gratuita y de pago en la red.

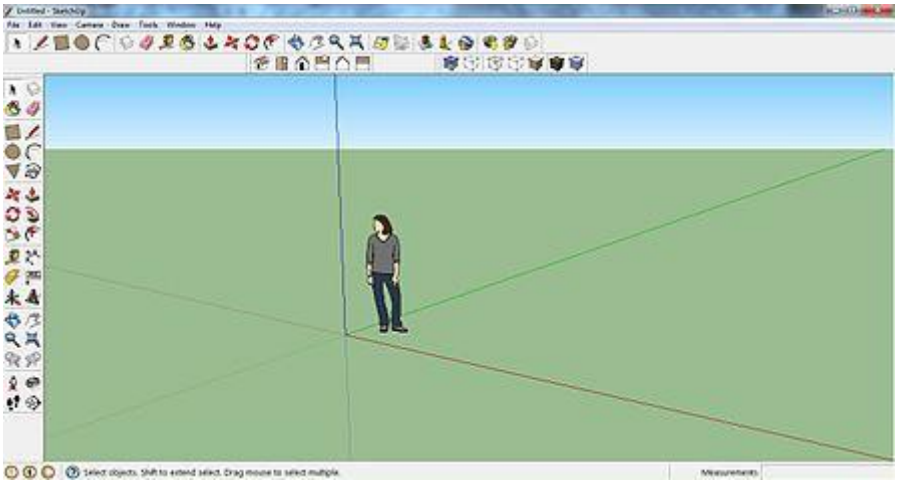


Ilustración 33 - Aplicación SketchUp

2.9.8 Adobe Photoshop

Aplicación empleada para el posprocesamiento de imágenes creadas con SketchUP e iconos varios usados en la aplicación web integrada en Ezweb. Adobe Photoshop es una aplicación de diseño en forma de taller de pintura y fotografía que trabaja sobre un "lienzo" y que está destinado a la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits en 2 dimensiones. Se reconoce como la aplicación más potente en retoque y modificación de fotografías del mundo.

Actualmente forma parte de la familia Adobe Creative Suite y es desarrollado y comercializado por Adobe Systems Incorporated inicialmente para computadores Apple pero posteriormente también para plataformas PC con sistema operativo Windows.

En las versiones iníciales de Adobe Photoshop se trabajaba en un imagen (bitmap) formado por una sola capa, donde se podían aplicar toda una serie de efectos, textos y tratamientos. A medida que la aplicación fue evolucionando el software ha ido añadiendo diversas mejoras fundamentales, la incorporación de un espacio de trabajo multicapa, inclusión de elementos vectoriales, gestión avanzada de color (ICM / ICC), tratamiento extensivo de tipografías, control y retoque de color, efectos creativos, posibilidad de incorporar plugins de terceros, exportación para sitios web entre otros.

Photoshop permite exportar las imágenes en los siguientes formatos:

- **PSD, PDD**: formato estándar de Photoshop con soporte de capas.
- **PostScript:** no es exactamente un formato, sino un lenguaje de descripción de páginas. Se suele encontrar documentos en PostScript. Utiliza primitivas de dibujo para poder editarlo.
- **EPS:** es una versión de PostScript, se utiliza para situar imágenes en un documento. Es compatible con programas vectoriales y de autoedición.
- **DCS:** fue creado por Quark (empresa de software para autoedición) y permite almacenar tipografía, tramas, etc. Se utiliza para filmación en autoedición.
- Prev. EPS TIFF: permite visualizar archivos EPS que no se abren en Photoshop, por ejemplo los de QuarkXPress.
- **BMP:** formato estándar de Windows.
- **GIF:** muy utilizado para las web. Permite almacenar un canal alfa para dotarlo de transparencia, y salvarlo como entrelazado para que al cargarlo en la web lo haga en varios pasos. Admite hasta 256 colores.
- **JPEG:** también muy utilizado en la WWW, factor de compresión muy alto y buena calidad de imagen.
- **TIFF:** una solución creada para pasar de PC a MAC y viceversa.
- **PICT:** desde plataformas MAC se exporta a programas de autoedición como QuarkXPress.
- **PNG:** la misma utilización que los GIF, pero con mayor calidad. Soporta transparencia y colores a 24 bits. Solo las versiones recientes de navegadores pueden soportarlos.
- **PDF:** formato original de Acrobat. Permite almacenar imágenes vectoriales y mapa de bits.
- **ICO:** es utilizado para representar archivos, carpetas, programas, unidades de almacenamiento, etc.

- **IFF:** se utiliza para intercambio de datos con Amiga.
- **PCX:** formato solo para PC. Permite colores a 1, 4, 8 y 24 bits.
- **RAW:** formato estándar para cualquier plataforma o programa gráfico.
- **TGA:** compatible con equipos con tarjeta gráfica de Truevision.
- **Scitex CT:** formato utilizado para documentos de calidad profesional.
- **Filmstrip:** se utiliza para hacer animaciones. También se puede importar o exportar a Premiere.
- **FlashPix:** formato originario de Kodak para abrir de forma rápida imágenes de calidad superior.
- **JPEG2000:** al igual que el JPEG, es nuevo formato de compresión que permite aumentar la calidad de la imagen.

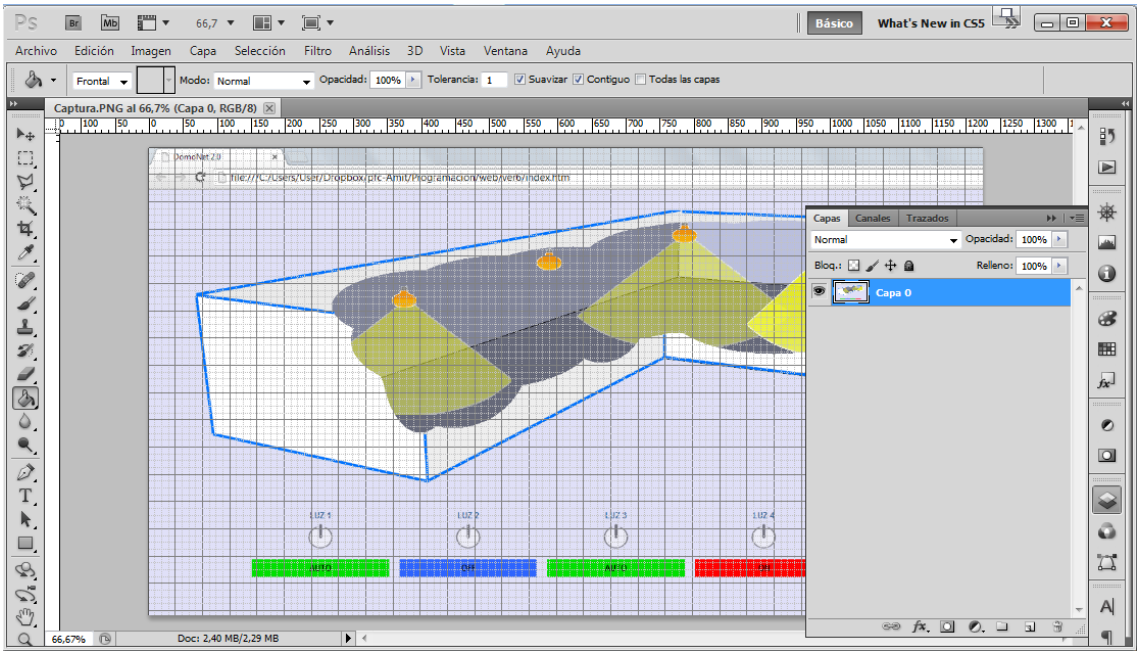


Ilustración 34 - Aplicación Adobe Photoshop

2.9.9 Protocolo IEEE 802.3

Para la comunicación entre los interfaces se empleo tramas con un formato similar al protocolo IEEE 802.3 Ethernet, para una posible integración a redes más extensas en el futuro. El protocolo IEEE 802.3 fue la primera propuesta para estandarizar Ethernet. Aunque hay ciertas variaciones en el campo de la cabecera que fue definido de forma diferente, posteriormente ha habido ampliaciones sucesivas al estándar que adaptarse a las mejoras de velocidad como Fast Ethernet, Gigabit Ethernet y 10 Gigabits Ethernet, redes virtuales, hubs, conmutadores, fibra óptica, etc.. Los estándares de este grupo no reflejan necesariamente lo que se usa en la práctica, aunque a diferencia de otros grupos este suele estar cerca de la realidad.

Formato de trama

7 bytes	1 byte	6 bytes	6 bytes	2 bytes	46 – 1500 bytes	4 bytes
Preámbulo	SDF	Destino	Origen	Tipo	Datos	CRC

- **Preámbulo:** Bits de sincronización "10101010" (x7).
- **SDF:** Delimitador de comienzo de trama "10101011".
- **Longitud:** Longitud del campo datos
- **Datos:** Datos de la trama
- **CRC:** Secuencia de chequeo de trama a través de un polinomio generador de orden 33:
- x^32+x^26+x^23+x^22+x^16+x^12+x^11+x^10+x^8+x^7+x^5+x^4+x^2+x+1

El formato de las tramas empleadas en la comunicación del sistema domótico ha sido diseñada teniendo en cuenta el formato de trama IEEE 802.3 Ethernet, utilizando los mismos campos salvo por una ligera modificación en los tamaños de bytes del preámbulo+SDF y el campo tipo, reduciendo cada uno de estos a 1 byte, más que suficientes para el sistema.

2.9.10 YAT (Yet Another Terminal)

Aplicación de software libre empleado para simulación de comunicaciones serial entre PC e interfaces antes de la implementación al bus RS-485. La conexión entre los periféricos se hizo con un adaptador USB-SERIAL FTDI. La aplicación destaca por permitir comunicaciones de una forma dinámica con tramas ASCII y HEXDECIMAL. Las principales características por la que destaca la aplicación son:

- Terminal se serie
- Envío de comandos por línea de texto, varias líneas de texto o archivos
- Posibilidad de predefinir ilimitadas tramas/comandos en memoria
- Intuitivo sistema de monitorización de datos enviados/recibidos
- Área de trabajo que permite múltiples sesiones de forma intuitiva

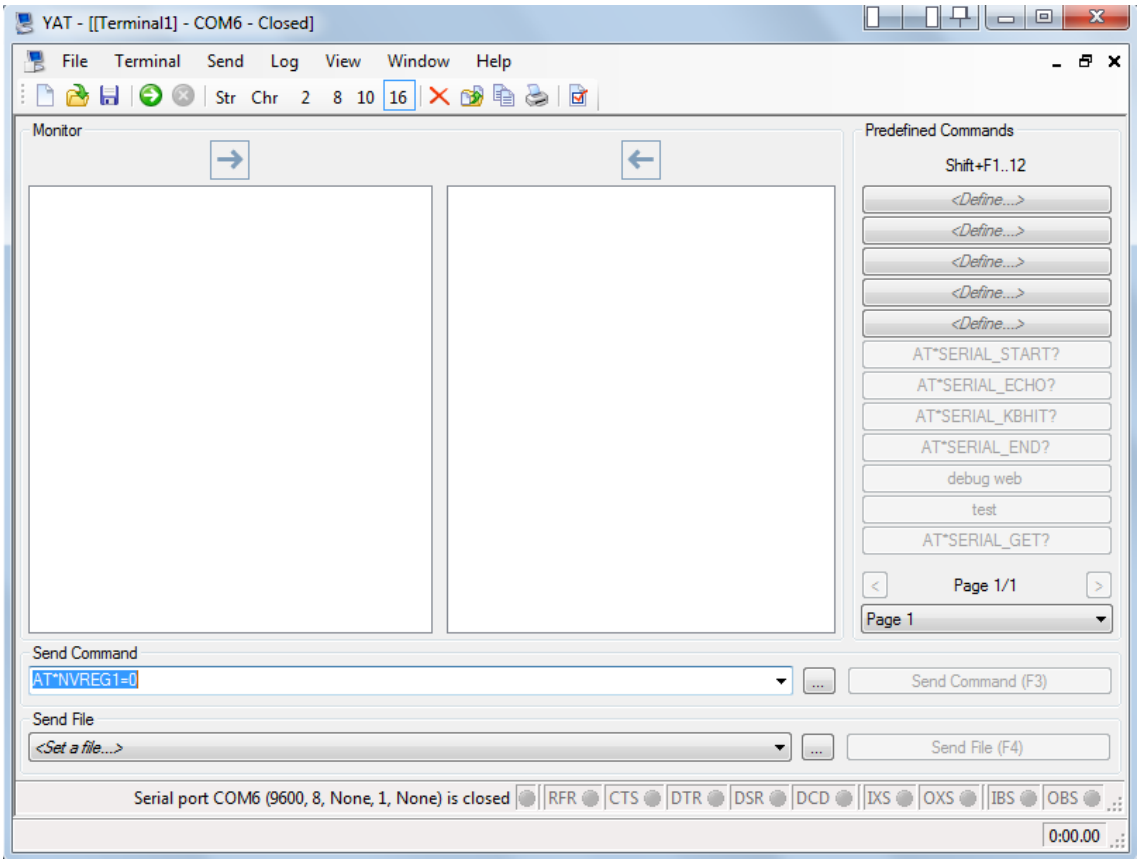


Ilustración 35 - Aplicación YAT

2.10 Conclusión

2.10 Conclusión

Concluimos la documentación de nuestro proyecto con nuestra valoración final respecto a las tecnologías empleadas para el desarrollo del proyecto.

Teniendo en cuenta, que desde el inicio del estudio para el desarrollo del sistema domótico los objetivos de estos fueron conseguir un sistema:

- Basado en componentes económicos de fácil adquisición
- Escalable y fácilmente modificable
- Compatible con todos los sistemas operativos y dispositivos móviles

Podemos concluir que estos fueron cumplidos. En primer lugar, la elección de trabajar con microcontroladores de la familia PIC[12] como unidad de procesado principal en la mayoría de los dispositivos, un micro-servidor web[13] y otros componentes básicos electrónicos de fácil adquisición en cualquier tienda electrónica hacen que la construcción de un sistema domótico como el planteado en este proyecto sea accesible para cualquier usuario o empresa.

También cabe destacar que se obtuvo un sistema escalable y fácilmente modificable por varias razones, en primer lugar, al utilizar un bus de datos basado en comunicaciones RS485 se abre la posibilidad de añadir dispositivos adiciones al canal de datos, siempre y cuando el número de dispositivos totales no supere los 128, con solo conectarlo al hilo de datos. El protocolo propuesto para el envío y recepción de datos no requiere de complejos sistemas, ya que el diseño para la trama de datos está basado en el formato empleado para las comunicaciones Ethernet, además de trabajar a 9600 baudios. Para el desarrollo de cada software integrado en cada uno de los dispositivos, se diseño previamente un diagrama de bloques, consiguiendo así un sistema de fácil modificación, ya que por la metodología de multitarea y modular empleada en su programación hace muy sencillo el añadir tareas adicionales al sistema, o bien modificar alguna tarea ya integrada.

Por último, se consiguió que el sistema domótico sea compatible para poder interactuar con cualquier otro dispositivo informático con una conexión internet, independientemente de su sistema operativo gracias a que el diseño y la programación del panel de control se desarrollo bajo los estándares de HTML5, por lo que da la posibilidad a cualquier dispositivo provisto de un navegador web y una conexión internet a interactuar con él.

Por lo tanto concluimos que las tecnologías y metodologías empleadas en este proyecto titulado "Control y monitorización de sistemas de domótica mediante un interfaz web" establecen un punto de partida en cuestión a la metodología seguida para el inicio de un desarrollo con el fin de crear un producto para ofrecer un sistema domótico completo capaz de satisfacer las necesidades más actuales del ámbito industrial, agrario y doméstico. Aún así se recomienda algunas actualizaciones si se plantea en ofrecer el producto final a un mercado más global como por ejemplo:

[12] Se empleo específicamente el PIC16F690 de 8 bits
[13] Lynx EzWeb, asequible desde la web https://www.ezweblynx.com/

Sustituir el micro-servidor Ezweb por uno de mayor potencia y prestaciones, como por ejemplo uno basado en las placas Ethernut, un interfaz completo con licencia libre perteneciente a la empresa Egnite GmbH, disponible en varias versiones, desde el más básico compuesto de un microcontrolador ATmega128, hasta otros más complejos con un procesador ARM9 de Atmel, una controladora de tarjetas SD y con el sistema operativo Nut/OS[14] integrado. Estas placas Ethernut pueden trabajar como servidores de web completos, o bien ofrecer una API para que otros servidores puedan realizar interacciones de forma asíncrona a dichos dispositivos y aumentar así el número máximo de usuarios que puedan interactuar con dicho interfaz web de forma simultánea.

Para el diseño en la disposición de la página web, sería idóneo hacerlo empleando el framework Bootstrap[15], liberado por Twitter basado en HTML5. Con esto no solo podemos conseguir unas vistas más profesionales por cuestión de diseño, sino también se puede obtener vistas con mayor grado de adaptación a los distintos tamaños de pantallas de los dispositivos que se pueden encontrar en el mercado, ya que dispone librerías basadas en jQuery[16] para facilitar dichas tareas.

[14] Versión minimalista del sistema operativo GNU Linux para procesadores ATMEL
[15] http://getbootstrap.com
[16] Liberia en Javascript que permite simplificar las tareas básicas para interactuar con el contenido escrito en HTML

3. Presupuesto

3. Presupuesto

A continuación mostramos el presupuesto para la producción de un sistema completo compuesto de una placa principal y 4 placas de mando. En el cálculo se incluyen tanto como el montaje, el precio de componentes y el desarrollo del software. Para una mejor visualización de los costes, se han elaborado tres presupuestos previos de donde se calcula cada fase del proyecto, y finalmente un presupuesto total en donde se suma los presupuestos previamente calculados. En resumen, las fases para el cálculo de dicho presupuesto son las siguientes:

- Fabricación de una placa principal
- Fabricación de una placa de mando
- Desarrollo de Software

Y finalmente, el presupuesto total, que sería igual al coste de la fabricación de una placa principal, el coste de la fabricación de cuatro placas mando (ya que en este proyecto estamos trabajando con cuatro placas) y el desarrollo del software. En el caso que necesitásemos desarrollar un sistema con un número mayor que cuatro placas de mando, con solo ajustar la cantidad de unidades en el presupuesto final, podríamos tener su cálculo también.

3.1 Pliego de condiciones

3.1 Pliego de condiciones

Como todo sistema domótico inteligente, estará ligado a que el espacio a ocupar cumpla con ciertos requisitos para que el sistema pueda operar correctamente y sin interrupciones. Después de realizar un estudio, y teniendo en cuenta las limitaciones técnicas indicadas por los distintos fabricantes de los componentes empleados para la construcción de los diferentes dispositivos que componen el sistema, para que el presente sistema domótico propuesto funcione correctamente será necesario cumplir las siguientes condiciones:

- El sistema domótico está diseñado para ser instalado en interiores, donde las condiciones climatológicas no sean muy adversas. Todos los componentes utilizados para la construcción de las placas están diseñados para que puedan operar en un rango de temperatura entre 0ºC y 70ºC. No se recomienda instalar el sistema domótico en exteriores para evitar posibles deterioros, ya que las placas que componen el sistema no has sido impermeabilizadas para resistir lluvias o índices de alta humedad.

- El número de dispositivos y electrodomésticos conectados al sistema no podrán superar de 128, esta es una limitación de hardware que viene dada por los controladores de comunicaciones rs485 MAX487 empleadas para la comunicación entre las distintas placas.

- Todos los dispositivos en el sistema, deberán de estar conectados al bus de datos común en *"daisy chain"*.[17]

- El tamaño de las tramas enviadas al sistema, por los distintos dispositivos y/o electrodomésticos no podrán superar los 255 caracteres, esto es una limitación que viene dada por el tamaño de los bufferes de entrada reservada definidas por las aplicaciones instaladas en las placas, las tramas que superen dicha longitud serán ignoradas y no provocaran un provocaran errores de desbordamiento.

- Se podrá acceder a la página web albergada en el microservidor web del sistema a través de una red local a través de la IP 192.168.1.60 y el puerto 80, dicha IP es modificable, pero no su puerto. En el caso de que sea necesario acceder a dicho dispositivo desde una red externa será necesario configurar el enrutador al que esté conectado con un nombre de dominio o una ip estática, y abrir un puerto para recibir las peticiones http y redireccionarlo a la ip fija del dispositivo y al puerto 80.

- Para poder interactuar con el interfaz web del sistema será necesario el emplear un navegador con soporte HTML5, los navegadores testados que han funcionado sin problemas han sido: Mozilla Firefox ver. 27, Google Chrome ver. 33 y Internet Explorer 11, aunque es probable que funcione con el resto de los navegadores.

[17] esquema de cableado usado en ingeniería eléctrica y electrónica.

- La distancia máxima entre los dispositivos conectados en el mismo canal de datos no podrá superar los 1200 metros, esta es una limitación que viene dada por el protocolo RS485.

Si se cumplen los requisitos previamente descritos, el sistema domótico debería de funcionar correctamente.

3.2 Presupuesto - Montaje placa principal

Para el cálculo de la producción de una placa principal, hemos tenido en cuenta la suma de todos los componentes necesarios y la mano de obra necesaria para su ensamblado. Cada sistema de domótica, necesitaría como mínimo una placa principal para su funcionamiento. En el modelo propuesto solo se necesitará uno.

Cantidad	Producto/Servicio	Unidad €	Coste €
1	Ezweb Lynx 5V	40,50	40,50
2	Micro Controlador PIC16F690 E-/P	1,56	3,12
2	MAX487 Interfaz rs485	1,74	3,48
4	Diodo Led Rojo	0,20	0,80
4	Resistencia 240 Ohmios	0,15	0,60
1	Resistencia 4K7 Ohmios	0,20	0,20
2	Condensador 100nF	0,21	0,42
1	Condensador 1000uF	0,74	0,74
1	Relé ULN2803	1,76	1,76
4	Sensor Infrarrojo DC 5V	4,60	18,40
2	Toma Externa con 2 conectores	0,22	0,44
2	Jumper	0,12	0,24
1	Tira conectores PIN 18cm	0,38	0,38
1	Regulador 7805 CT	0,32	0,32
1	Fuente Alimentación 9 Voltios	8,40	8,40
1	Placa PCB - 105mm x 148mm	2,80	2,80
5	Hora Montaje	36,00	180,00
		Subtotal	262,60
		IGIC (7 %)	18,38
		Importe Total	280,98

3.3 Presupuesto - Montaje placa mando (1 unidad)

Al igual que la placa principal, para el cálculo del montaje de una placa de mando hemos tenido en cuenta el coste de sus componentes, gastos de envío y la mano de obra para su montaje. Las placas de mando no son indispensables para que el sistema de domótica funcione, ya que su función es el enviar y recibir instrucciones a la placa principal y estas funcionan independientemente, puesto que la placa principal no requiere de las placas de mando para que pueda seguir gestionando el sistema de iluminación. El número de placas de mando conectados al sistema es variable, en el sistema propuesto se emplean cuatro placas de mando.

Cantidad	Producto/Servicio	Unidad €	Coste €
1	Micro Controlador PIC16F690 E-/P	1,56	1,56
2	MAX487 Interfaz rs485	1,74	3,48
3	Diodo Led (Varios colores)	0,20	0,60
3	Resistencia 240 Ohmios	0,15	0,45
1	Resistencia 10K Ohmios	0,15	0,15
3	Condensador 100nF	0,21	0,63
1	Condensador 1000uF	0,74	0,74
2	Toma Externa con 2 conectores	0,22	0,44
1	Pulsador 1 Amp.	0,44	0,44
1	Regulador 7805 CT	0,32	0,32
1	Fuente Alimentación 9 Voltios	8,40	8,40
1	Placa PCB - 105mm x 148mm	2,80	2,80
3	Hora Montaje	36,00	108,00
		Subtotal	128,01
		IGIC (7 %)	8,96
		Importe Total	136,97

3.4 Presupuesto -Desarrollo software y diseño de circuitos

Para el desarrollo del software necesario para el sistema de domótica son necesarios tres programas, uno para cada microcontrolador del sistema. La placa principal estará programado con la aplicación que gestiona la comunicación entre los distintos dispositivos, sensores y sistema de iluminación, es por ello que es el más costoso. La aplicación que contienen las placas de mando se limitan únicamente a enviar y recibir instrucciones simples a la placa principal, y por último, la aplicación que contiene el Ezweb, se ocupa en servir una página web con instrucciones similares a las de una placa de mando.

En el coste de trabajo por hora se ha tenido en cuenta la amortización de las herramientas y aplicaciones de pago necesarias para el desarrollo de las tareas.

Cantidad	Producto/Servicio	Unidad €	Coste €
31	Desarrollo software para placa principal y diseño de circuito electrónico (horas)	40,00	1240,00
5	Diseño circuito electrónico de placa principal (horas)	40,00	200,00
21	Desarrollo software para placa mando (horas)	40,00	840,00
3	Diseño circuitos electrónicos de placa mando (horas)	40,00	120,00
8	Desarrollo software de página web para Ezweb (horas)	40,00	320,00
		Subtotal	2720,00
		IGIC (7 %)	190,40
		Importe Total	2910,40

3.5 Presupuesto – Sistema total (1 Placa Principal + 4 Mandos)

Para el cálculo del presupuesto del sistema total propuesto, se suman el total de los tres presupuestos anteriormente calculados. El presupuesto calculado para la fabricación de una placa de mando se multiplica por cuatro, ya que se necesitan cuatro unidades para esta propuesta. Pero para otros casos el número de placas de mandos puede variar, por lo que para calcular su presupuesto total solo se necesitaría modificar la cantidad de mandos a utilizar.

Cantidad	Producto/Servicio	Unidad €	Coste €
1	Montaje placa principal	262,60	262,60
4	Montaje placa mando	128,01	512,04
1	Desarrollo software y diseño de circuitos	2720,00	2720,00
		Subtotal	3494,64
		IGIC (7 %)	244,62
		Importe Total	3739,26

El presente presupuesto asciende a un total de:

Tres mil setecientos treinta y nueve euros con 26 céntimos (3739,26€)

Firma: Amit L. Babani

Fecha: 10 Febrero 2014

4. Anexos

4.1 Reglamento de baja tensión ITC-BT-51

4.1 Reglamento de baja tensión ITC-BT-51

INSTALACIONES DE SISTEMAS DE AUTOMATIZACIÓN, GESTIÓN TÉCNICA DE LA ENERGÍA Y SEGURIDAD PARA VIVIENDAS Y EDIFICIOS

1. OBJETO Y CAMPO DE APLICACIÓN

Esta Instrucción establece los requisitos específicos de la instalación de los sistemas de automatización, gestión técnica de la energía y seguridad para viviendas y edificios, también conocidos como sistemas domóticos.

El campo de aplicación comprende las instalaciones de aquellos sistemas que realizan una función de automatización para diversos fines, como gestión de la energía, control y accionamiento de receptores de forma centralizada o remota, sistemas de emergencia y seguridad en edificios, entre otros, con excepción de aquellos sistemas independientes e instalados como tales, que puedan ser considerados en su conjunto como aparatos, por ejemplo, los sistemas automáticos de elevación de puertas, persianas, toldos, cierres comerciales, sistemas de regulación de climatización, redes privadas independientes para transmisión de datos exclusivamente y otros aparatos, que tienen requisitos específicos recogidos en las Directivas europeas aplicables conforme a lo establecido en el artículo 6 del Reglamento Electrotécnico para Baja Tensión.

Quedan excluidas también las instalaciones de redes comunes de telecomunicaciones en el interior de los edificios y la instalación de equipos y sistemas de telecomunicaciones a los que se refiere el Reglamento de Infraestructura Común de Telecomunicaciones (I.C.T.), aprobado por el R.D. 279/1999.

Igualmente están excluidos los sistemas de seguridad reglamentados por el Ministerio del Interior y Sistemas de Protección contra Incendios, reglamentados por el Ministerio de Fomento (NBE-CPI) y el Ministerio de Industria y Energía (RIPCI).

No obstante, a las instalaciones excluidas anteriormente, cuando formen parte de un sistema más complejo de automatización, gestión de la energía o seguridad de viviendas o edificios, se les aplicarán los requisitos de la presente Instrucción además los requisitos específicos reglamentarios correspondientes.

2. TERMINOLOGÍA

Sistemas de Automatización, Gestión de la Energía y Seguridad para Viviendas y Edificios:

Son aquellos sistemas centralizados o descentralizados, capaces de recoger información proveniente de unas entradas (sensores o mandos), procesarla y emitir órdenes a unos actuadores o salidas, con el objeto de conseguir confort, gestión de la energía o la protección de personas animales y bienes.

Estos sistemas pueden tener la posibilidad de accesos a redes exteriores de comunicación, información o servicios, como por ejemplo, red telefónica conmutada, servicios INTERNET, etc.

Nodo:
Cada una de las unidades del sistema capaces de recibir y procesar información comunicando, cuando proceda con otras unidades o nodos, dentro del mismo sistema.

Actuador:
Es el dispositivo encargado de realizar el control de algún elemento del Sistema, como por ejemplo, electroválvulas (suministro de agua, gas, etc.), motores (persianas, puertas, etc.), sirenas de alarma, reguladores de luz, etc.

Dispositivo de entrada:
Sensor, mando a distancia, teclado u otro dispositivo que envía información al nodo.

Los elementos definidos anteriormente pueden ser independientes o estar combinados en una o varias unidades distribuidas.

Sistemas centralizados:
Sistema en el cual todos los componentes se unen a un nodo central que dispone de funciones de control y mando.

Sistema descentralizado:
Sistema en que todos sus componentes comparten la misma línea de comunicación, disponiendo cada uno de ellos de funciones de control y mando.

3. TIPOS DE SISTEMAS

Los sistemas de Automatización, Gestión de la energía y Seguridad considerados en la presente instrucción, se clasifican en los siguientes grupos:

Sistemas que usan en todo o en parte señales que se acoplan y transmiten por la instalación eléctrica de Baja Tensión, tales como sistemas de corrientes portadoras.

Sistemas que usan en todo o en parte señales transmitidas por cables específicos para dicha función, tales como cables de pares trenzados, paralelo, coaxial, fibra óptica.

Sistemas que usan señales radiadas, tales como ondas de infrarrojo, radiofrecuencia, ultrasonidos, o sistemas que se conectan a la red de telecomunicaciones.

Un sistema domótico puede combinar varios de los sistemas anteriores, debiendo cumplir los requisitos aplicables en cada parte del sistema. La topología de la instalación puede ser de distintos tipos, tales como, anillo, árbol, bus o lineal, estrella o combinaciones de éstas.

4. REQUISITOS GENERALES DE LA INSTALACIÓN

Todos los nodos, actuadores y dispositivos de entrada deben cumplir, una vez instalados, los requisitos de Seguridad y Compatibilidad Electromagnética que le sean de aplicación, conforme a lo establecido en la legislación nacional que desarrolla la Directiva de Baja Tensión (73/23/CEE) y la Directiva de Compatibilidad Electromagnética (89/336/CEE). En el caso de que estén incorporados en otros aparatos se atendrán, en lo que sea aplicable, a lo requisitos establecidos para el producto o productos en los que vayan a ser integrados.

Todos los nodos, actuadores y dispositivos de entrada que se instalen en el sistema, deberán incorporar instrucciones o referencias a las condiciones de instalación y uso que deban cumplirse para garantizar la seguridad y compatibilidad electromagnética de la instalación, como por ejemplo, tipos de cable a utilizar, aislamiento mínimo, apantallamientos, filtros y otras informaciones relevantes para realizar la instalación. En el caso de que no se requieran condiciones especiales de instalación, esta circunstancia deberá indicarse expresamente en las instrucciones.

Dichas instrucciones se incorporarán en el proyecto o memoria técnica de diseño, según lo establecido en la ITC-BT-04.

Toda instalación nueva, modificada o ampliada de un sistema de automatización, gestión de la energía y seguridad deberá realizarse conforme a lo establecido en la presente Instrucción y lo especificado en las instrucciones del fabricante, anteriormente citadas.

En lo relativo a la Compatibilidad Electromagnética, las emisiones voluntarias de señal, conducidas o radiadas, producidas por las instalaciones domóticas para su funcionamiento, serán conformes a las normas armonizadas aplicables y, en ausencia de tales normas, las señales voluntarias emitidas en ningún caso superarán los niveles de inmunidad establecidos en las normas aplicables a los aparatos que se prevea puedan ser instalados en el entorno del sistema, según el ambiente electromagnético previsto.

Cuando el sistema domótico esté alimentado por muy baja tensión o la interconexión entre nodos y dispositivos de entrada este realizada en muy baja tensión, las instalaciones e interconexiones entre dichos elementos seguirán lo indicado en la ITC-BT-36.

Para el resto de los casos, se seguirán los requisitos de instalación aplicables a las tensiones ordinarias.

5. CONDICIONES PARTICULARES DE INSTALACIÓN

Además de las condiciones generales establecidas en el apartado anterior, se establecen los siguientes requisitos particulares.

5.1. Requisitos para sistemas que usan señales que se acoplan y transmiten por la instalación eléctrica de baja tensión

Los nodos que inyectan en la instalación de baja tensión señales de 3 kHz hasta 148,5 kHz cumplirán lo establecido en la norma UNE-EN 50.065 -1 en lo relativo a compatibilidad

electromagnética. Para el resto de frecuencias se aplicará la norma armonizada en vigor y en su defecto se aplicará lo establecido en el apartado 4.

5.2. Requisitos para sistemas que usan señales transmitidas por cables específicos para dicha función

Sin perjuicio de los requisitos que los fabricantes de nodos, actuadores o dispositivos de entrada establezcan para la instalación, cuando el circuito que transmite la señal transcurra por la misma canalización que otro de baja tensión, el nivel de aislamiento de los cables del circuito de señal será equivalente a la de los cables del circuito de baja tensión adyacente, bien en un único o en varios aislamientos.

Los cables coaxiales y los pares trenzados usados en la instalación deberán cumplir con las normas de la serie EN 61.196 y CEI 60.189 -2.

5.3. Requisitos para sistemas que usan señales radiadas

Adicionalmente, los emisores de los sistemas que usan señales de radiofrecuencia o señales de telecomunicación, deberán cumplir la legislación nacional vigente del "Cuadro Nacional de Atribución de Frecuencias de Ordenación de las Telecomunicaciones".

4.2 Código Fuente Placa

Principal

4.2 Código Fuente Placa Principal

```
;*****************************************************************
;                                                                *
;    Archivo:      principal18.asm                               *
;    Fecha:        febrero-2014                                  *
;    Version:            1.8                                     *
;                                                                *
;    Autor:        Amit Babani                                   *
;                                                                *
;*****************************************************************
;                                                                *
;    Archivos necesarios: P16F690.INC                            *
;                                                                *
;*****************************************************************
;                                                                *
;    Descripción: Aplicación encargada de sincronizar todas las co-  *
;    municaciones entre los distintos dispositivos, en este caso en  *
;    gestionará la comunicación entre el EZWEB los distintos mandos,  *
;    y los sensores como los temporizadores necesarios.          *
;                                                                *
;    Actualizaciones:                                            *
;      1.8 Corrección en trama enviada a EzWeb                   *
;      1.7 El max487 se pone en modo envío en la función enviar tx,  *
;        y en modo recepción en la función clear error           *
;      1.6 Cambio de la trama enviada al EZWEB                   *
;      1.5 Implementado control de flujo para los MAX485         *
;      1.4 Cambiando evaluar_mando, para que solo envíe una trama    *
;             como verificación.                                 *
;      1.3 Cambiado los sensores a PULL UP,                      *
;      1.2 Eliminada la función de evaluar trama Ezweb.          *
;       Ahora cuando el el pic principal recibe un cambio de estado  *
;       del mando, responde como cambio recibido                 *
;      1.1 Cambiado sensor 4 a RA4                               *
;      1.0 Implementación detección de errores con la funcion    *
;             clear error()                                      *
;      0.9 Corrección de funciones de envío y recepción          *
;      0.8 Comentarios y limpieza de código                      *
;      0.7 Implementación de Sensor()                            *
;      0.6 Implementación de la variable MODO SYNC y la evaluar mando *
;      0.5 Corrección de mapeado de mem. y actualización evaluar web  *
;      0.4 Implementación de enviar tx                           *
;      0.3 Implementación de evaluar_web                         *
;      0.2 Implementación de evaluar_rx()                        *
;      0.1 Implementación de la función Receive()                *
;                                                                *
;*****************************************************************
;                                                                *
;                                                                *
;                          ---------------                       *
;                    VCC ---|Vdd         Vss|--- GND             *
;                           |                |                  *
;                           |RA5         RA0|--- SENSOR 1        *
;                SENSOR 4 ---|RA4         RA1|--- SENSOR 2        *
;                           |RA3         RA2|--- SENSOR 3        *
;                           |                |                  *
;                           |RC5         RC0|--- LUZ 1           *
;                        ---|RC4         RC1|--- LUZ 2           *
;                   LUZ4 ---|RC3         RC2|--- LUZ 3           *
;          CONTROL MAX485 ---|RC6            |                   *
;          ERROR INDICATOR ---|RC7         RB4|---               *
;                           |            RB5|--- RX SERIAL       *
;                TX SERIAL ---|RB7         RB6|---               *
;                           -----------------                    *
;                                                                *
;                                                                *
; ---------------------------------------------------------------

    #include <p16f690.inc>

; ---------------------------------------------------------------------------
; Bits de configuración: adapte los parámetros a su necesidad
  __config  _FCMEN_OFF& _IESO_OFF& _MCLRE_OFF& _WDT_OFF& _INTOSCIO
```

```
 ;
 ;  _FCMEN_OFF           ; -- fail safe clock monitor enable off
 ;   IESO OFF            ; -- int/ext switch over enable off
 ;   BOR ON              ; default, brown out reset on
 ;  _CPD_OFF             ; default, data eeprom protection off
 ;  _CP_OFF              ; default, program code protection off
 ;   MCLR OFF            ; -- use MCLR pin as digital input
 ;   PWRTE OFF           ; default, power up timer off
 ;   WDT OFF             ; -- watch dog timer off
 ;  _INTOSCIO            ; -- internal osc, RA6 and RA7 I/O
 ;


;definición de constantes
; --------------CONSTANTES--------------------------------------------------------
HOST    equ 0x10       ;Dirección MAC de la máquina

AUTO           equ 0x00       ;Valor de la trama que indica modo AUTO
ON                    equ 0x01       ;Valor de la trama que indica modo ON
OFF                   equ 0x02       ;Valor de la trama que indica modo OFF
BUFFER_RX      equ 0x07       ;Tamaño de buffer de entrada

TIEMPO         equ 0xFE       ;tiempo que estará activo una luz N al activaste el sensor N
(segundos)

; --------------VARIABLES---------------------------------------------------------

; Indicador de uso general
               UDATA
sPORTC                 RES 1 ; sombra del puerto C
INDI                   RES 1 ; indicadores de carácter general
EVAL                   RES 1 ; donde se almacena el byte a ser evaluado
ORIGEN                 RES 1 ; variable donde se almacena el valor del origen de la trama de
datos
MODO SYNC              RES 1 ; variable donde sincronizará los distintos estados de los
mandos

; Variables usadas para gestionar estados

;cblock 0x55
EL2                    RES 1          ; indica el estado en el que se encuentra la función
read serial()
EL3            RES 1          ; indica el estado en el que se encuentra la función
enviar_tx()
EL4            RES 1          ; indica el estado en el que se encuentra la función sensor()
EL5            RES 1          ; indica el estado en el que se encuentra la función
evaluar_mando()

;BUFFER_RX     RES 1                  ; variable usado como buffer de entrada en la
comunicación serial
CRC_RX         RES 1                  ; para el cálculo del crc de la trama recibida
LONG_RX        RES 1                  ; longitud del campo dato de la trama recibida
CONT_RX        RES 1                  ; contador de la posición del campo dato

CRC_TX         RES 1             ; para el cálculo del crc de la trama recibida
;LONG_TX       RES 1             ; longitud del campo dato de la trama recibida
;BUFFER_TX     RES 1             ; variable usado como buffer de salida en la comunicación
serial
CONT_TX        RES 1          ; contador de la posición del campo dato

CONTADOR_RETRASO       RES 1 ; Variable para gestionar el temporizador de Luz cuando está en
AUTO, usado en sensor()
PDel0                          RES 1   ; variables usadas para los contadores de los
temporizadores
PDel1                          RES 1
;PDel2                         RES 1
;endc


; Variables globales
INT_VAR UDATA_SHR
w_saved      RES 1 ; variable utilizada para guardar contexto
status_saved RES 1 ; variable utilizada para guardar contexto
pclath_saved RES 1 ; variable utilizada para guardar contexto
```

```
; reiniciar vector
STARTUP CODE 0x000
    nop                   ; requerido para el depurado ICD2
    movlw   high start    ; cargar el byte superior de la etiqueta "start"
    movwf   PCLATH        ; inicializar PCLATH
    goto    start         ; ir al inicio del código principal

; vector de interrupciones
INT VECTOR CODE 0x004
d    goto    interrupt      ; ir al inicio del código de interrupción

; código reubicable
PROG CODE
interrupt
    movwf   w saved       ; guardar contexto
    swapf   STATUS,w
    movwf   status_saved
    movf    PCLATH,w      ; solamente se requiere si se utiliza más de la primera página
    movwf   pclath_saved
    clrf    PCLATH
    ; << agregue el código de interrupción >>
    movf    pclath_saved,w ; restaurar contexto
    movwf   PCLATH
    swapf   status_saved,w
    movwf   STATUS
    swapf   w saved,f
    swapf   w_saved,w
    retfie

;---------------------INICIO----------------------------------------
;-------------------------------------------------------------------

;comienzo de la aplicación
start

       ;Configuración de los puertos de salida y entrada

       bsf    STATUS,RP0     ;seleccionamos el banco 3
       bcf    STATUS,RP1

       ;OPTION REG[7]=0
       bcf OPTION_REG,7      ;activamos pull-up

       bcf    STATUS,RP0
       bcf    STATUS,RP1     ;seleccionamos el banco 0

       ;configuramos los puertos
       clrf   PORTA          ;inicializamos puerto A
       clrf   PORTB          ;inicializamos puerto B
       clrf   PORTC          ;inicializamos puerto C
    clrf    sPORTC      ;borramos la sombra del puerto C


       bsf    STATUS,RP1     ;seleccionamos el banco 2

       clrf   ANSEL          ;modo digital
       clrf   ANSELH         ;modo digital

       ;sensores
       bsf    STATUS,RP0     ;seleccionamos el banco 1
       bcf    STATUS,RP1

       movlw  b'00010111'    ;configuramos RA0 --> RA3 como entrada para los sensores
              ;               |||||
              ;               |||||
              ;               |||||---- RA0 = Input
              ;               ||||----- RA1 = Input
              ;               |||------ RA2 = Input
              ;               ||------- RA3 = Sin uso
              ;               |-------- RA4 = Input

       movwf  TRISA

       ;los configuramos como pulls ups
       movwf  WPUA

       movlw  b'00000000'    ;configuramos RC0 --> RC3 como salida para las luces
```

```
        ;          |  ||||
        ;          |  ||||
        ;          |  ||||---- RC0 = Output
        ;          |  |||----- RC1 = Output
        ;          |  ||------ RC2 = Output
        ;          |  |------- RC3 = Output
        ;          |
        ;          |---------- RC6 = Control Multimaster

        movwf   TRISC


;configuración del puerto serie

        bsf     TRISB,7
        bcf     TRISB,6
    movlw   b'01100000'     ; Fosc= 4Mhz
    movwf   OSCCON

Stable
          btfss   OSCCON,HTS      ; osc estable?
             goto    Stable          ; repetir hasta estabilizar

        movlw   .25       ; [(4Mhz/9615)/64]-1
        movwf   SPBRG      ; 9615 baudios con Fosc=4MHz, 8bits
        movlw   .00    ;
    movwf   SPBRGH


  movlw   b'00100100'     ; '0-------' CSRC, n/a (async)
                          ; '-0------' TX9 off, 8 bits
                          ; '--1-----' TXEN, tx enabled
                          ; '---0----' SYNC, async mode
                          ; '----0---' SENDB, send brk
                          ; '-----1--' BRGH, high speed
                          ; '------00' TRMT, TX9D
  movwf   TXSTA           ;
  bcf     STATUS,RP0      ; bank 0
  movlw   b'10010000'     ; '1-------' SPEN, port enabled
                          ; '-0------' RX9 off, 8 bits
                          ; '--0-----' SREN, n/a (async)
                          ; '---1----' CREN, rx enabled
                          ; '----0---' ADDEN off
                          ; '-----000' FERR, OERR, RX9D
  movwf   RCSTA           ;
  movf    RCREG,W         ; flush Rx Buffer
  movf    RCREG,W         ;


;inicialización de indicadores
        movlw  b'00000000'
;                |  |||||
;                |  ||||-- DataReady usado en: read serial(), eval_web() y eval_mando() ->
Indica que hay un dato (buffer rx) listo para ser evaluado
;                 |  |||--- eval_rx: usado en enviar_ezweb() y evaluar_rx(), indica que se
ha recibido una trama de un mando
;                 |  ||---- Sin uso
;                 |  |----- eval_mando: usado en evaluar_rx() y mando() -> Indica que el
dato recibido está evaluado
;                       |     ------ serial:
;                ---------- sensor usado en leer_teclado() -> Para detectar el flanco de
subida del pulsador
        movwf   INDI

;inicialización de variables
        movlw   .00
        movwf   MODO_SYNC      ; iniciamos los MODOS de las luces a AUTO

        movlw   .00
        movwf   EL5                   ; indicador de estado para la función evaluar_mando

        movlw   .00
        movwf   EL4                   ; indicador de estado para la función sensor

        movlw   .00
        movwf   EL3                   ; indicador de estado para la función enviar_tx
```

```
        movlw   .00
        movwf   EL2                     ; indicador de estado para la función read_serial

        movlw   .00
        movwf   CONT_TX         ; inicializamos el contador para el buffer de salida

        movlw   .00
        movwf   CONT_RX         ; inicializamos el CRC del buffer de salida

        movlw   .00
        movwf   CRC_RX          ; inicializamos el CRC del buffer de salida

        movlw   .00
        movwf   CRC_TX          ; inicializamos el CRC del buffer de salida


;---------------------BUCLE PRINCIPAL--------------------------------
;--------------------------------------------------------------------

repetir

        ;repeat

        ;read serial()
        call read_serial        ;función encargada de leer la trama recibida por el puerto
serial

        ;read_sensor()
        call sensor                     ;función encargada de gestionar luces y sensores

        ;evaluar rx()
        call evaluar_rx                 ;identifica de donde viene la trama

        ;enviar mando()
        call evaluar_mando      ;La trama recibida es del mando

        ;enviar_tx()
        call enviar_tx          ;Enviar el buffer de salida a través del serial

        ;clear error()
        call clear_error                ;Función encargada de detectar errores y corregirlos
en su caso

        goto repetir            ;fin repeat


;---------------------------------CLEAR ERROR------------------------------
;--------------------------------------------------------------------------
;                                                                          *
;       Comprobamos si los bits OERR y FERR ha detectado un error,         *
;       si lo han detectado reinciamos la USART.                           *
;                                                                          *
;       OERR= Overrun error                                                *
;       FERR= Frame error                                                  *
;                                                                          *
;--------------------------------------------------------------------------

clear_error

        bcf     STATUS,RP0      ;seleccionamos banco 0
        bcf     STATUS,RP1


        ;if (OERR==1)
        btfsc   RCSTA,OERR      ;si se detecta error de desbordamiento
        ;reset_usart()
        goto reset_oerr                 ;ejecutamos función de corrección

        ;if (FERR==1)
        btfsc   RCSTA,FERR      ;si detectamos error de ventana
        ;reset_ferr()
        goto reset_ferr                 ;ejecutamos función de corrección

        ;return()
        return
```

```
reset_oerr                          ;función de corrección en desbordamiento
       bcf          RCSTA,CREN      ;desactivamos eusart
       nop
       nop
       bsf          RCSTA,CREN      ;activamos eusart
       nop
       nop

       ;if (RCIF==1)                ;vaciando el buffer
       btfsc  PIR1,RCIF             ;hay algún dato?
       ;empty buffer()
       goto empty_buffer
       ;return()
       return               ;no se ha recibido ningún dato, volvemos a la función
principal

empty buffer
       movf   RCREG,W
       nop
       nop
       ;if (RCIF==1)
       btfsc  PIR1,RCIF     ;repetimos el vaciado hasta que no quede ningún dato
       ;empty buffer()
       goto empty buffer
       ;return()
       return

reset_ferr                                  ;función encargada de corregir errores de
ventana
       ;bcf    TXSTA,TXEN
       ;bsf    TXSTA,TXEN
       ;return()
       return


;---------------------------------SENSOR-------------------------------------
;----------------------------------------------------------------------------
;
;      Cada mando esta asociado a un sensor y una luz y un contador
;      Los sensores solo funcionarán si el respectivo mando está en modo Auto
;      El EzWeb tiene control absoluto para controlar los Modos de todas las
;      luces.
;
;      RA0, RA1, RA2 y RA4 entrada sensores
;   RC0 --> RC3 sálida a reles luces
;
;      Mando1 --> Luz1 (RC0) --> Sensor1 (RA0) --> Contador (0x60)
;      Mando2 --> Luz1 (RC1) --> Sensor1 (RA1) --> Contador (0x61)
;      Mando3 --> Luz1 (RC2) --> Sensor1 (RA2) --> Contador (0x62)
;      Mando4 --> Luz1 (RC3) --> Sensor1 (RA3) --> Contador (0x63)
;
;----------------------------------------------------------------------------


sensor

       ;inicialización del buffer_tx
       bcf    STATUS,RP0
       bcf    STATUS,RP1     ;seleccionamos el banco 0

       movlw  HIGH switch_sensor
       movwf  PCLATH

switch sensor
       ;switch(EL4)
       movf EL4, W    ; cargo W con el estado (0 --> 3)
       addwf PCL, F   ; modifico el contador del programa en función del estado coincidente

       goto seg4 caso0              ;Comprobamos el sensor 1 y actualizamos la luz 1
       goto seg4 caso1              ;Comprobamos el sensor 2 y actualizamos la luz 2
       goto seg4_caso2              ;Comprobamos el sensor 3 y actualizamos la luz 3
       goto seg4_caso3              ;Comprobamos el sensor 4 y actualizamos la luz 4


seg4_caso0
```

```
        movlw   .01                         ;Cambiamos al siguiente estado
        movwf   EL4

        ;comprobamos si la luz 1 está en MODO ON
        movlw b'00000011'
        andwf MODO_SYNC,W            ;filtramos los bits que vamos a evaluar y guardamos en
W
        ;if (MODO SYNC=XXXXXX01)
        sublw   0x01                 ;si la luz 1 está en modo ON
        btfss   STATUS,Z
        goto seg4_caso0_1
        ;LUZ1=1
        bsf     sPORTC,0             ;encendemos la luz
    movf sPORTC, W
    movwf PORTC
        ;return()
        return          ;volvemos para comprobar el modo de la siguiente luz

        ;else
seg4 caso0 1
        ;comprobamos si la luz 1 está en MODO OFF
        movlw b'00000011'
        andwf MODO_SYNC,W            ;filtramos los bits que vamos a evaluar y guardamos en
W
        ;if (MODO SYNC=XXXXXX10)
        sublw   0x02                 ;si la luz 1 está en modo OFF
        btfss   STATUS,Z
        goto    seg4_caso0_2
        ;LUZ1=0
        bcf             sPORTC,0             ;apagamos la luz 1
        movf    sPORTC, W
        movwf   PORTC
        ;return()
        return          ;volvemos para comprobar el modo de la siguiente luz

        ;else

seg4_caso0_2
        ;esta en MODO AUTO, comprobamos los sensores y temporizadores

        ;if (sensor1=1)
                btfsc   PORTA,0                     ;si no se ha activado el sensor
                ;cont seg4 caso0()
                goto cont_seg4_caso0         ;disminuimos el temporizador
        ;else
                ;BUFFER[0x50]=TIEMPO
                movlw   0x60                        ;si se ha activado el sensor
                movwf   FSR                                 ;reiniciamos su temporizador
individual con TIEMPO
                movlw   TIEMPO
                movwf   INDF
                ;LUZ1=1
                bsf             sPORTC,0                                ;encendemos la luz 1
                movf    sPORTC, W
                movwf   PORTC
                ;return()
                return

cont seg4 caso0                                 ; disminución del contador
                ;if (LUZ=0)
                btfss   sPORTC,0                            ;comprobamos si la luz está
apagada
                ;return()
                return          ;si la luz está apagada, por lo que el temporizador está a 0
no hay nada que descontar

                ;else                                                   ;si la luz está encendida
                ;CONTADOR RETRASO=BUFFER[0x50]
                movlw   0x60            ;dirección donde debería de estar la temporizador
individual
                movwf   FSR
                movf    INDF,W          ;cargamos el contador individual
                movwf   CONTADOR_RETRASO

                ;retardo()
                call retardo   ;la luz está encendida, disminuimos el temporizador individual
```

```
                ;BUFFER[0x50]=CONTADOR_RETRASO
                movlw   0x60            ;volvemos a almacenar el contador individual en su
respectiva
                movwf   FSR
                movf    CONTADOR_RETRASO,W    ;dirección de memoria
                movwf   INDF

                ;if (CONTADOR RETRASO=0)
                movf    CONTADOR RETRASO,W            ;si el contador llego a 0
                sublw   0x00
                btfsc   STATUS,Z
                ;LUZ1=0
                bcf             sPORTC,0                                        ;apagamos su
respectiva luz
                movf    sPORTC, W
                movwf   PORTC

                ;return()
                return


seg4_caso1
        movlw   .02                                             ;Cambiamos al siguiente estado
        movwf   EL4

        ;comprobamos si la luz 1 está en MODO ON
        movlw b'00001100'
        andwf MODO_SYNC,W             ;filtramos los bits que vamos a evaluar y guardamos en
W
        ;if (MODO SYNC=XXXX01XX)
        sublw   0x04                  ;si la luz 2 está en modo ON
        btfss   STATUS,Z
        goto seg4_caso1_1
        ;LUZ2=1
        bsf             sPORTC,1              ;encendemos la luz
        movf    sPORTC, W
        movwf   PORTC
        ;return()
        return          ;volvemos para comprobar el modo de la siguiente luz

        ;else
seg4 caso1 1
        ;comprobamos si la luz 1 está en MODO OFF
        movlw b'00001100'
        andwf MODO_SYNC,W             ;filtramos los bits que vamos a evaluar y guardamos en
W
        ;if (MODO SYNC=XXXX10XX)
        sublw   0x08                  ;si la luz 2 está en modo OFF
        btfss   STATUS,Z
        goto seg4_caso1_2
        ;LUZ2=0
        bcf     sPORTC,1              ;encendemos la luz
    movf sPORTC, W
    movwf PORTC
        ;return()
        return          ;volvemos para comprobar el modo de la siguiente luz

        ;else


seg4_caso1_2
        ;esta en MODO AUTO, comprobamos los sensores y temporizadores

        ;if (sensor2=0)
                btfsc   PORTA,1               ;disminuimos el temporizador
                ;cont_seg4_caso0()
                goto cont_seg4_caso1
        ;else
                ;BUFFER[0x51]=TIEMPO
                movlw   0x61                            ;si se ha activado el sensor
                movwf   FSR                                     ;reiniciamos su temporizador
individual con TIEMPO
                movlw   TIEMPO
                movwf   INDF
                ;LUZ2=1
                bsf             sPORTC,1                                ;encendemos la luz 1
                movf    sPORTC, W
```

```
                movwf   PORTC
                ;return()
                return                                  ;comprobamos la siguiente luz

cont_seg4_caso1                                                 ; disminución del contador
                ;if (LUZ=0)
                btfss   sPORTC,1                                        ;comprobamos si la luz
está apagada
                ;return()
                return          ;si la luz está apagada, por lo que el temporizador está a 0
no ha nada que descontar

                ;else                           ;si la luz está encendida
                ;CONTADOR RETRASO=BUFFER[0x51]
                movlw   0x61            ;dirección donde debería de estar la temporizador
indidual
                movwf   FSR
                movf    INDF,W          ;cargamos el contador individual
                movwf   CONTADOR_RETRASO

                ;retardo()
                call retardo   ;la luz está encendida, disminuimos el temporizador individual

                ;BUFFER[0x51]=CONTADOR RETRASO
                movlw   0x61            ;volvemos a almacenar el contador individual en su
respectiva
                movwf   FSR
                movf    CONTADOR_RETRASO,W              ;dirección de memoria
                movwf   INDF

                ;if (CONTADOR RETRASO=0)
                movf    CONTADOR RETRASO,W              ;si el contador llego a 0
                sublw   0x00
                btfsc   STATUS,Z
                ;LUZ2=0
                bcf             sPORTC,1                                ;apagamos su respectiva
luz
                movf    sPORTC, W
                movwf   PORTC

                ;return()
                return                                  ;comprobamos la siguiente luz

seg4_caso2
        movlw   .03
        movwf   EL4

        ;comprobamos si la luz 1 está en MODO ON
        movlw b'00110000'
        andwf MODO_SYNC,W               ;filtramos los bits que vamos a evaluar y guardamos en
W
        ;if (MODO_SYNC=XX01XXXX)
        sublw   .16
        btfss   STATUS,Z
        goto seg4_caso2_1
        ;LUZ3=1
        bsf             sPORTC,2                ;Si lo está, encendemos la luz
        movf    sPORTC, W
        movwf   PORTC
        ;return()
        return          ;volvemos para comprobar el modo de la siguiente luz

        ;else
seg4 caso2 1
        ;comprobamos si la luz 1 está en MODO OFF
        movlw b'00110000'
        andwf MODO_SYNC,W               ;filtramos los bits que vamos a evaluar y guardamos en
W
        ;if (MODO SYNC=XX10XXXX)
        sublw   .32
        btfss   STATUS,Z
        goto seg4_caso2_2
        ;LUZ3=0
        bcf             sPORTC,2                ;Si lo está, apagamos la luz
        movf    sPORTC,W
        movwf   PORTC
        ;return()
```

```
        return          ;volvemos para comprobar el modo de la siguiente luz

        ;else
seg4 caso2 2
        ;esta en MODO AUTO, comprobamos los sensores y temporizadores

        ;if (sensor3=0)
                btfsc   PORTA,2
                ;cont seg4 caso2()
                goto cont_seg4_caso2
        ;else
                ;BUFFER[0x52]=TIEMPO
                movlw   0x62
                movwf   FSR
                movlw   TIEMPO
                movwf   INDF
                ;LUZ2=1
                bsf             sPORTC,2
                movf    sPORTC,W
                movwf   PORTC
                ;return()
                return

cont seg4 caso2
                ;if (LUZ=0)
                btfss   sPORTC,2
                ;return()
                return          ;la luz está apagada, por lo que el temporizador está a 0

                ;else
                ;CONTADOR RETRASO=BUFFER[0x52]
                movlw   0x62            ;dirección donde debería de estar la dirección de
origen
                movwf   FSR
                movf    INDF,W          ;cargamos el dato del buffer
                movwf   CONTADOR_RETRASO

                ;retardo()
                call retardo    ;la luz está encendida, disminuimos el contador retardo

                ;BUFFER[0x52]=CONTADOR_RETRASO
                movlw   0x62
                movwf   FSR
                movf    CONTADOR_RETRASO,W
                movwf   INDF

                ;if (CONTADOR RETRASO=0)
                movf    CONTADOR RETRASO,W
                sublw   0x00
                btfsc   STATUS,Z
                ;LUZ3=0
                bcf             sPORTC,2
                movf    sPORTC,W
                movwf   PORTC

                ;return()
                return

seg4 caso3
        movlw   .00
        movwf   EL4

        ;comprobamos si la luz 1 está en MODO ON
        movlw b'11000000'
        andwf MODO_SYNC,W               ;filtramos los bits que vamos a evaluar y guardamos en
W
        ;if (MODO_SYNC=01XXXXXX)
        sublw   .64
        btfss   STATUS,Z
        goto seg4_caso3_1
        ;LUZ4=1
        bsf             sPORTC,3        ;Si lo está, encendemos la luz
        movf    sPORTC,W
        movwf   PORTC
        ;return()
        return          ;volvemos para comprobar el resto de las luces
```

```
        ;else
seg4_caso3_1
        ;comprobamos si la luz 1 está en MODO OFF
        movlw b'11000000'
        andwf MODO_SYNC,W            ;filtramos los bits que vamos a evaluar y guardamos en
W
        ;if (MODO SYNC=XX10XXXX)
        sublw   .128
        btfss   STATUS,Z
        goto seg4_caso3_2
        ;LUZ4=0
        bcf             sPORTC,3             ;Si lo está, apagamos la luz
        movf    sPORTC,W
        movwf   PORTC
        ;return()
        return          ;volvemos para comprobar el resto de las luces

        ;else
seg4 caso3 2
        ;esta en MODO AUTO, comprobamos los sensores y temporizadores
        ;if (sensor4=0)
                btfsc   PORTA,4  ;--- ANTES 3
                ;cont_seg4_caso3()
                goto cont_seg4_caso3
        ;else
                ;BUFFER[0x53]=TIEMPO
                movlw   0x63
                movwf   FSR
                movlw   TIEMPO
                movwf   INDF
                ;LUZ3=1
                bsf             sPORTC,3
                movf    sPORTC,W
                movwf   PORTC
                ;return()
                return

cont_seg4_caso3
                ;if (LUZ=0)
                btfss   sPORTC,3
                ;return()
                return          ;la luz está apagada, por lo que el temporizador está a 0

                ;else
                ;CONTADOR_RETRASO=BUFFER[0x53]
                movlw   0x63            ;dirección donde debería de estar la dirección de
origen
                movwf   FSR
                movf    INDF,W          ;cargamos el dato del buffer
                movwf   CONTADOR_RETRASO

                ;retardo()
                call retardo   ;la luz está encendida, disminuimos el contador retardo

                ;BUFFER[0x53]=CONTADOR_RETRASO
                movlw   0x63
                movwf   FSR
                movf    CONTADOR_RETRASO,W
                movwf   INDF

                ;if (CONTADOR_RETRASO=0)
                movf    CONTADOR RETRASO,W
                sublw   0x00
                btfsc   STATUS,Z
                ;LUZ4=0
                bcf             sPORTC,3
                movf    sPORTC,W
                movwf   PORTC

                ;return()
                return

;--------------------FUNCIONES USADAS DENTRO DE SENSOR()------------------
retardo                                                 ;descuenta el temporizador individual y
produce un pequeño retardo
        nop
        ;CONTADOR_RETRASO--
```

```
       decf CONTADOR_RETRASO,F             ;Contados individual por LUZ

              movlw     .20
        movwf     PDel0
PLoop0  movlw     .72         ;Repetición el bucle principal de retardo
        movwf     PDel1
PLoop1                                              ; Repetición del bucle secundario de
retardo
        decfsz    PDel1,1
        goto      PLoop1
        decfsz    PDel0,1
        goto      PLoop0
       nop                                                ;Al terminar los dos bucles sale
    return



;----------------------------------ENVIAR TX-------------------------------------
;--------------------------------------------------------------------------------
;*****************************************************************************
; Formato de trama                                                           *
;   El formato a usar es idéntico a la trama IEEE 802.3 con la               *
;   omisión del campo 'preámbulo'                                            *
;                                                                            *
;            |---------------------------------------------|                 *
;            | SOF | Destino | Origen | Tipo | Datos  | FCS |                 *
;            |---------------------------------------------|                 *
; N° Bytes      1       1         1       1      7       4                   *
;                                                                            *
;                                                                            *
; Formato del campo de Datos para Mando                                      *
;                                                                            *
;        L00E0M0  (7 bytes)                                                  *
;         || | |                                                            *
;         || | |                                                            *
;         || | -- Modo de operación (0x00-Auto,0x01-On,0x02-Off)            *
;         || |                                                              *
;         || --- Estado de la iluminación (0-Off, 1-On)[NO DISPONIBLE]*
;         ||                                                                 *
;         ---- Identificador del juego de iluminación (00-99)[NO DISP]*
;                                                                          *
;                                                                          *
;                                                                          *
; Formato del campo de Datos para EzWeb                                      *
;                                                                            *
;        CCCCCCM  (7 bytes)                                                  *
;        |    ||                                                            *
;        |    ||                                                            *
;        |    |-- Modo de operación (0x00-Auto,0x01-On,0x02-Off)            *
;        |    |                                                             *
;        -------- Estado de los controles [NO DISPONIBLE]                   *
;                                                                          *
;                                                                          *
;                                                                          *
;  Primera parte del buffer a enviar                                       *
;  Para que se detectada por los mandos                                    *
;
;      |0x30| 0x31  |0x32|0x33| 0x34 --> 0x3A  | VARI | --> Dirección
;   |-------------------------------------------|
;   |0xAB|CONTROL|HOST|0X07|X X X X X MODO|CRC_TX|
;   |-------------------------------------------|
;     |                                 |
;           Buffer almacenado en memoria
;
;
;      Segunda parte del buffer a enviar
;      Para el Ezweb, el valor de los modos y luces varía
;      ya que él los interfaces web pueden tener problemas a la hora
;      de trabajar con valores menores de 0x30
;      _____________________________________________
;      |0x50|0x51|0x52|0x53|0x54|0x55|0x56|0x57|0x58|0x59|0x5A| --> Dir.
;   |-----------------------------------------------------|
;   |0x41|0x54|0x2A|0x4E|0x56|0x52|0x45|0x47|LUZ |0x3D|MODO|      --> Hex
;   | A    T    *    N    V    R    E    G    XX   =    XX |      --> Ascii
;   |-----------------------------------------------------|
```

```
;     |_____________________________________|
;           Buffer almacenado en memoria
;
;       Luz 1 = 31h          Auto = 30h
;       Luz 2 = 32h          On = 31h
;       Luz 3 = 33h          Off = 32h
;       Luz 4 = 34h
;
;
;-------------------------------------------------------------------------


enviar tx
       ;if (serial==0)
       btfss   INDI,4
       ;return()
       return                ;No hay nada que enviar a través del puerto serial

       ; ponemos el MAX485 en modo de envío
       ;PORTC[6]=1
       bsf           sPORTC,6
       movf    sPORTC,W
       movwf   PORTC

       ;else                 ;tenemos una trama en el buffer para ser enviada
       movlw   HIGH switch_enviar_tx
       movwf   PCLATH

       switch_enviar_tx
       ;switch(EL3)
       movf EL3, W    ; cargo W con el estado (0 -->   )
       addwf PCL, F   ; modifico el contador del programa en función del estado coincidente

       goto seg3_caso0              ;Comprobamos que el TXREG está libre para enviar datos
y enviamos datos del 1° Buffer
       goto seg3 caso1              ;Comprobamos que el TXREG está libre para enviar datos
y enviamos CRC, reiniciamos contadores
       goto seg3_caso2              ;Pequeño retardo
       goto seg3_caso3              ;Enviamos 2° Buffer para el EZWEB
       goto seg3_caso4              ;Iniciamos contadores de espera
       goto seg3_caso5              ;Volvemos al Max485 en recepción, y inicialización de
flags

;-------------------------------------------------------------------------------------------

seg3_caso0
              bcf     STATUS,RP0
              bcf     STATUS,RP1     ;seleccionamos el banco 0



              ;if (TXIF==0)
              btfss   PIR1,TXIF      ;comprobar si TXREG está vacio
              ;return()
              return         ;no está el buffer listo, esperamos

              ;else

              ;enviamos dato
              ;N=CONTADOR_TX+0x30
              movlw   0x30           ;apuntamos a la 1° dirección del buffer
              addwf   CONT_TX,W      ;sumamos al puntador el valor del contador
              movwf   FSR

              ;TXREG=BUFFER[N]
              movf    INDF,W
              movwf   TXREG

              ;CRC TX=CRC TX+BUFFER[N]
              movf    INDF,W
              addwf   CRC_TX,1       ;actualizamos el valor de CRC
              ;CONT_TX
              incf    CONT_TX,1              ;incrementamos el contador

              movf    CONT TX,W
              sublw   .11            ;el buffer tiene una longitud de 11 bytes
              btfsc   STATUS,Z
```

```
                ;if(CONT_TX=11)
                goto    fin_tx          ;hemos terminado de enviar el buffer
                ;if(CONT TX!=11)
                goto    fin_no_tx       ;no hemos terminado de enviar el buffer

fin_tx
                ;EL3=1
                movlw .01
                movwf EL3
                ;return()
                return

fin no tx
                ;return()
                return

;-------------------------------------------------------------

seg3 caso1

                bcf     STATUS,RP0
                bcf     STATUS,RP1      ;seleccionamos el banco 0


                ;if (TXIF==0)
                btfss   PIR1,TXIF       ;comprobar si TXREG está vacio
                ;return()
                return          ;no está el buffer listo, esperamos

                ;else

                ;TXREG=CRC TX
                movf    CRC_TX,W        ;envio del CRC
                movwf   TXREG

                ;CRC TX=250
                movlw   .250
                movwf   CRC_TX          ; inicializamos el CRC y usamos dicha variable como
temporizados de delay
                ;CONT_TX=0
                movlw   0x00
                movwf   CONT_TX         ;inicializamos el contador

                ;EL3=2
                movlw .2                        ;vamos al siguiente estado
                movwf EL3
                ;return()
                return

;-------------------------------------------------------------------
seg3_caso2
; este estado introduce un retardo
        nop
        ;DELAY--
        decfsz  CRC_TX,1
        ;if (delay!=0) return
        return

        movlw .03
        movwf EL3

;------------------------------
seg3 caso3
                bcf     STATUS,RP0
                bcf     STATUS,RP1      ;seleccionamos el banco 0

                ;if (TXIF==0)
                btfss   PIR1,TXIF       ;comprobar si TXREG está vacio
                ;return()
                return          ;no está el buffer listo, esperamos

                ;else

                ;enviamos dato
                ;N=CONTADOR TX+0x30
                movlw   0x4E            ;apuntamos a la 1° dirección del buffer
                addwf   CONT_TX,W       ;sumamos al puntador el valor del contador
```

```
                movwf   FSR

                ;TXREG=BUFFER[N]
                movf    INDF,W
                movwf   TXREG

                ;CONT TX
                incf    CONT_TX,1               ;incrementamos el contador

                movf    CONT_TX,W
                sublw   .15             ;el buffer tiene una longitud de 15 bytes
                btfsc   STATUS,Z
                ;if(CONT TX=15)
                goto    fin tx2         ;hemos terminado de enviar el buffer
                ;if(CONT TX!=15)
                goto    fin_no_tx2      ;no hemos terminado de enviar el buffer

fin tx2
                ;EL3=1
                movlw .04
                movwf EL3
                ;CONT_TX=0
                movlw   0x00
                movwf   CONT_TX         ;inicializamos el contador

                ;return()
                return

fin_no_tx2
                ;return()
                return

;-----------------------------------------------------------------------
seg3_caso4

        ;if (TXIF==0)
        btfss   PIR1,TXIF       ;comprobar si TXREG está vacio
        ;return()
        return                  ;no se ha recibido ningún dato, volvemos a la función
principal

        movlw .05
        movwf EL3

        movlw   .250            ;iniciamos contador de delay
        movwf   CRC_TX          ;empleamos la variable CRC_TX ya que no la estamos usando
ahora

        ;return()
        return

;-------------------------------------------------
seg3_caso5

        nop
        ;DELAY--
        decfsz  CRC_TX,1
        ;if (delay!=0) return
        return

        movlw .00
        movwf EL3

        ;serial=0
        bcf     INDI,4                  ;cambiamos indicador

        movlw   .00
        movwf   CRC_TX          ; inicializamos el CRC del buffer de salida


        ;ponemos el MAX485 en modo recepción
        ;PORTC[6]=0
        bcf             sPORTC,6
        movf    sPORTC,W
        movwf   PORTC

        return
```

```
;---------------------------------EVALUAR RX-----------------------------------
;------------------------------------------------------------------------------
;**********************************************************************
; Analizamos el campo de ORIGEN
;
;
;       ORIGEN
;        0x01  Mando 1        ----|
;        0x02  Mando 2               |---- Marcamos el flag:
;        0x03  Mando 3               |                     eval_mando
;        0x04  Mando 4----|
;
;
;**********************************************************************

evaluar rx
               ;if (EVAL RX==0)
               btfss   INDI,1
               ;return()
               return                  ;no tenemos un dato listo para ser evaluado

               ;else
               bcf     STATUS,RP0      ;seleccionamos el banco 0
               bcf     STATUS,RP1

               movlw   0x40            ;dirección donde debería de estar la dirección de
origen
               movwf   FSR
               movf    INDF,W          ;cargamos el dato del buffer
               movwf   ORIGEN

               ;if (ORIGEN==0x01)
               movf    ORIGEN,W        ;Comprobamos si en el mando 1
               sublw   0x01
               btfsc   STATUS,Z
               ;trama_mando()          ;indicamos que hemos recibido un dato de un mando
               goto trama_mando

               ;if (ORIGEN==0x02)
               movf    ORIGEN,W        ;Comprobamos si es el mando 2
               sublw   0x02
               btfsc   STATUS,Z
               ;trama_mando()          ;indicamos que hemos recibido un dato de un mando
               goto trama_mando

               ;if (ORIGEN==0x03)
               movf    ORIGEN,W        ;Comprobamos si es el mando 3
               sublw   0x03
               btfsc   STATUS,Z
               ;trama_mando()          ;indicamos que hemos recibido un dato de un mando
               goto trama_mando

               ;if (ORIGEN==0x04)
               movf    ORIGEN,W        ;Comprobamos si es el mando 4
               sublw   0x04
               btfsc   STATUS,Z
               ;trama mando()          ;indicamos que hemos recibido un dato de un mando
               goto trama_mando

               ;eval rx=0;
               bcf     INDI,1          ;indicamos que hemos evaluado el dato, y no es una
trama válida
               ;Dataready=0
               bcf     INDI,0
               ;return()
               return

trama mando
               ;eval mando=1
               bsf     INDI,3
               ;eval_rx=0;
               bcf     INDI,1          ;indicamos que hemos evaluado el dato, y no es una
trama válida
               ;return
               return
```

```
;---------------------READ SERIAL-------------------------------------
;---------------------------------------------------------------------

;*********************************************************************
; Formato de trama                                                   *
;   El formato a usar es idéntico a la trama IEEE 802.3 con la       *
;   omisión del campo 'preámbulo'                                    *
;                                                                    *
;            |-----------------------------------------------|       *
;            | SOF | Destino | Origen | Tipo | Datos  | FCS |       *
;            |-----------------------------------------------|       *
; N° Bytes      1       1         1       1     Var.     4          *
;                                 |              |                   *
;                                 |            ----                  *
;                               - -            |                    *
;                              |               |                    *
; Se almacenará los campos: ORIGEN (0x40) y DATOS (0x41 en adelante) *
;                                                                    *
;                                                                    *
;---------------------------------------------------------------------

CODE 0x200

read serial    ;selector de estado

       ;if (dataready==1)
       btfsc   INDI,0         ;el último dato recibido aún no ha sido evaluado, así evitamos
machacar el buffer
       ;return()
       return

       ;if (serial==1)
       btfsc   INDI,4
       ;return()
       return                 ; Aún estamos enviando


       movlw   HIGH switch_read_serial
       movwf   PCLATH

switch read serial
       ;switch(estado)
       movf EL2, W    ; cargo W con el estado (0 --> 11)
       addwf PCL, F   ; modifico el contador del programa en función del estado coincidente

       goto seg2 caso0
       goto seg2_caso1
       goto seg2_caso2
       goto seg2_caso3
       goto seg2_caso4
       goto seg2_caso5

;---------------------ESTADOS READ_SERIAL-----------------------------------
seg2_caso0     ;estado encargado de comprobar si se ha recibido un byte

               ;if (RCIF==0)
               btfss   PIR1,RCIF      ;comprobar si se ha recibido un dato
               ;return()
               return                 ;no se ha recibido ningún dato, volvemos a la función
principal

               ;if (RCIF==1)
               ;Recepción del preámbulo
               ;if(RCREG=="AB")
               movf    RCREG,W        ;Comprobamos el byte recibido con el preambulo (AB =
10101011)
               sublw   0xAB
               btfsc   STATUS,Z
               goto AB_OK
               ;else
               goto AB_no_OK

AB OK          ;EL2=1
               movlw   .01            ; Si es el preámbulo vamos al caso 1
               movwf   EL2
```

```
                ;return()
                return

AB no OK
                ;return()
                return

;-------------------------------------------------------------------------------

seg2_caso1      ;estado encargado de comprobar si se ha recibido un byte

                ;if (RCIF==0)
                btfss   PIR1,RCIF       ;comprobar si se ha recibido un dato
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (RCIF==1)
                ;Recepción de campo de DESTINO
                ;if (RCREG==HOST)
                movf    RCREG,W         ;Si la trama no está destinada a este periférico,
volvemos al estado inicial caso 0
                sublw   HOST            ;El valor de HOST está definido en los valores
iniciales
                btfsc   STATUS,Z
                goto    HOST_OK

                ;else
                goto HOST_no_OK

HOST OK         ;CRC RX=RCREG
                movlw   HOST            ;*** Es host literal??
                movwf   CRC_RX          ; Inicializamos el valor de CRC con el valor de HOST
                ;EL2=4
                movlw   .02             ;carga 0x02 en EL2, y saltamos al caso 2
                movwf   EL2
                ;return()
                return

HOST_no_OK
                movlw   .00             ;carga 0x00 en EL2, y volvemos al caso 0
                movwf   EL2
                ;return()
                return

;-------------------------------------------------------------------------------

seg2_caso2      ;estado encargado de comprobar si se ha recibido un byte

                ;if (RCIF==0)
                btfss   PIR1,RCIF       ;comprobar si se ha recibido un dato
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (RCIF==1)
                ;Recepción del campo ORIGEN

                bcf     STATUS,RP0      ;seleccionamos el banco 0
                bcf     STATUS,RP1

                movlw   0x40            ;dirección de inicio para el buffer de entrada
                movwf   FSR

                ;BUFFER RX=RCREG
                movf    RCREG,W
                movwf   INDF            ;grabamos el byte recibido del campo dato en el buffer
                ;CRC RX=CRC RX+RCREG
                movf    INDF,W
                addwf   CRC_RX,1
                ;EL2=3
                movlw   .03             ;carga 0x03 en EL2, y saltamos al caso 3
                movwf   EL2
                ;return()
                return

;-------------------------------------------------------------------------------
```

```
seg2_caso3     ;estado encargado de comprobar si se ha recibido un byte

               ;if (RCIF==0)
               btfss   PIR1,RCIF      ;comprobar si se ha recibido un dato
               ;return()
               return                 ;no se ha recibido ningún dato, volvemos a la función
principal

               ;if (RCIF==1)

       ;Recepción del campo TIPO, definirá el tamaño del campo DATO

               ;LONG RX=RCREG
               movf    RCREG,W
               movwf   LONG_RX



               ;if (RCREG==LONG RX)
               ;movf   RCREG,W        ;Si la trama no está destinada a este periférico,
volvemos al estado inicial caso 0
               ;movwf  LONG_RX
               sublw   BUFFER_RX             ;El valor de BUFFER_RX está definido en los
valores iniciales
               btfsc   STATUS,Z
               goto    LONG_RX_OK

               ;else
               goto LONG_RX_no_OK

LONG RX no OK
               movlw   .00            ;carga 0x00 en EL2, y volvemos al caso 0
               movwf   EL2
               ;return()
               return

LONG_RX_OK
               ;CRC_RX=CRC_RX+RCREG
               movlw   BUFFER_RX
               movwf   LONG RX
               addwf   CRC_RX,1

               ;i=0;
               movlw   0x00           ;inicializamos el contador para el campo datos
               movwf   CONT_RX

               ;EL2=4
               movlw   .04            ;carga 0x04 en EL2, y saltamos al caso 4
               movwf   EL2

               ;return()
               return

;-------------------------------------------------------------------------------

seg2_caso4     ;estado encargado de comprobar si se ha recibido un byte

               ;if (RCIF==0)
               btfss   PIR1,RCIF      ;comprobar si se ha recibido un dato
               ;return()
               return                 ;no se ha recibido ningún dato, volvemos a la función
principal

               ;if (RCIF==1)

               ;Recepción del campo DATO
               ;inicializamos el inicio del buffer para almacenar el campo de datos

               bcf     STATUS,RP0     ;seleccionamos el banco 0
               bcf     STATUS,RP1

               movlw   0x41           ;dirección de inicio para el buffer de entrada
               addwf   CONT_RX,W      ;sumamos a la posición el número del contador
               movwf   FSR

               ;BUFFER_RX[i]=RCREG
```

```
                movf    RCREG,W
                movwf   INDF            ;grabamos el byte recibido del campo dato en el buffer
                ;CRC RX=CRC RX+RCREG
                movf    INDF,W
                addwf   CRC_RX,1
                ;LONG--
                decf    LONG_RX,1
                ;i++
                incf    CONT_RX,1

                ;if (LONG_RX==0)
                movf    LONG_RX,W
                sublw   .0
                btfsc   STATUS,Z
                goto    FIN_DATO        ;fin de la lectura del campo DATO, saltamos al
siguiente caso
                ;else
                goto    FIN_no_DATO     ;aún que por recibir bytes de la trama datos, volvemos
al estado anterior

FIN_DATO
                ;EL2=5
                movlw   .05
                movwf   EL2             ;carga 0x05 en EL2, y saltamos al caso 5
                ;return()
                return

FIN_no_DATO
                ;return()
                return

;-------------------------------------------------------------------------------

seg2_caso5      ;estado encargado de comprobar si se ha recibido un byte

                ;if (RCIF==0)
                btfss   PIR1,RCIF       ;comprobar si se ha recibido un dato
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (RCIF==1)

        ;Recepción del CRC y **verificación

                ;EL2=0
                movlw   .0              ;carga 0x00 en EL2, y volvemos al caso inicial
                movwf   EL2
                ;if(CRC_RX==RCREG)
                movf    RCREG,W         ;comprobamos el CRC recibido con el calculado

CRC_OK  ;DataReady=1
                bsf     INDI,0          ;marcamos el flag de nuestro indicador como que hay un
dato
                ;eval_rx=1
                bsf INDI,1
                ;return()
                return



;---------------------------------EVALUAR MANDO-----------------------------------
;---------------------------------------------------------------------------------
;

;             |-------------------------------------------------|
;  INFO       | 0xAB | 0x20 | 0x10 | 0x07 |     L00E0M "X"   | FCS |
;             |-------------------------------------------------|
;
;             |-------------------------------------------------|
;             | SOF | Destino | Origen | Long |    Datos     | FCS |
;             |-------------------------------------------------|
; N° Bytes      1       1         1       1       7           1
;                                 |               |
;                                 |               |
;                               - -             ---
;                             |               |
```

```
; Se almacenará los campos: ORIGEN (0x40) y MODO (0x47)
;
;       Dirección Host
;       Pic Principal: 0x10
;       EzWeb:            0x20
;      Mando:0x01 --> 0x04
;
;Se actualizará la variable MODO SYNC que a su vez se almacenará en la
;dirección 0x38 del buffer de salida
;
;       MODO SYNC:     M4 M3 M2 M1    Cada par de bytes detalla el modo que está
;                                     cada mando.
;                      00 - Auto
;                      01 - On
;                      10 - Off
;
;-------------------------------------------------------------------------
        org 0x300
evaluar_mando

        ;if (eval_mando==0)
        btfss   INDI,3
        ;return()
        return                  ;no se ha recibido ninguna trama de algún mando, por lo que no
hay que actualizar los mandos

        ;if (serial==1)
        btfsc   INDI,4
        ;return()
        return                  ;ya se está enviando una trama, así evitamos machacar el
buffer

        ;else
        movlw   HIGH switch_evaluar_mando
        movwf   PCLATH

switch evaluar mando
        ;switch(EL5)
        movf EL5, W     ; cargo W con el estado (0 --> 5)
        addwf PCL, F    ; modifico el contador del programa en función del estado coincidente

        goto seg5 caso0             ;Construcción del Buffer de salida
        goto seg5 caso1             ;Identificamos de que mando es la trama de entrada
        goto seg5_caso2             ;La trama recibida es del mando 1
        goto seg5_caso3             ;La trama recibida es del mando 2
        goto seg5_caso4             ;La trama recibida es del mando 3
        goto seg5_caso5             ;La trama recibida es del mando 4



seg5_caso0

        ;inicialización del buffer_tx
        bcf     STATUS,RP0
        bcf     STATUS,RP1      ;seleccionamos el banco 0

        movlw   0x30            ;apuntamos a la 1° posición del buffer
        movwf   FSR
        movlw   0xAB            ;Cargamos el PREAMBULO
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;Cargamos el dato " ", esta es  (ADDR:0x31)
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   HOST            ;Cargamos la dirección HOST
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x07            ;Cargamos el valor TIPO con la longitud del campo datos (7)
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;1° byte del campo datos " "
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;2° byte del campo datos " "
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;3° byte del campo datos " "
```

```
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;4° byte del campo datos " "
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;5° byte del campo datos " ", aquí debería de almacenar modo
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;6° byte del campo datos " "
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;7° byte del campo datos " "
        movwf   INDF

; Segunda parte del buffer para el EzWEB, creamos el buffer con el dato
; AT*NVREG_=_
; Al finalizar de enviar la trama tenemos que terminar con un salto de línea <CR>=0x0D
<CL>=0x0A para que el Ezweb lo acepte

        movlw   0x4E            ;apuntamos a la 1° posición del buffer
        movwf   FSR


        movlw   0x0D            ;Cargamos <CR> que equivale a un salto de línea
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x0A            ;Cargamos <CL> que equivale a un salto de línea
        movwf   INDF

        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x41            ;Cargamos "A"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x54            ;Cargamos "T"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x2A            ;Cargamos "*"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x4E            ;Cargamos "N"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x56            ;Cargamos "V"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x52            ;Cargamos "R"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x45            ;Cargamos "E"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x47            ;Cargamos "G"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer (Dir: 0x58)
        movlw   0xFF            ;Dejamos la posición en blanco para el número de la MANDO
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x3D            ;Cargamos "="
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer (Dir: 0x5A)
        movlw   0xFF            ;Dejamos la posición en blanco para el MODO
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x0D            ;Cargamos <CR> que equivale a un salto de línea
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x0A            ;Cargamos <CL> que equivale a un salto de línea
        movwf   INDF

        ;Modificamos el 1° BUFFER_TX para los MANDOS
        ;Actualizando MANDO
        ;EVAL=BUFFER_RX[MANDO(40h)]
        movlw   0x40
        movwf   FSR
        movf    INDF,W
        movwf   EVAL
        ;BUFFER_TX[31h]=EVAL
```

```
        movlw   0x31
        movwf   FSR
        movf    EVAL,W
        movwf   INDF

        ;Actualizando MODO
        ;EVAL=BUFFER_RX[MODO(47h)]
        movlw   0x47
        movwf   FSR
        movf    INDF,W
        movwf   EVAL
        ;BUFFER TX[3Ah]=EVAL
        movlw   0x3A
        movwf   FSR
        movf    EVAL,W
        movwf   INDF

        ;Modificamos el 2° BUFFER_TX para el EZWEB
        ;Actualizando MANDO
        ;EVAL=BUFFER_RX[MANDO(40h)]+30h
        movlw   0x40
        movwf   FSR
        movf    INDF,W
        addlw   0x30            ;Añadimos 30h para que el EzWeb la entiend
        movwf   EVAL
        ;BUFFER TX[58h]=EVAL
        movlw   0x58
        movwf   FSR
        movf    EVAL,W
        movwf   INDF

        ;Actualizando MODO
        ;EVAL=BUFFER_RX[MODO(47h)]+30h
        movlw   0x47
        movwf   FSR
        movf    INDF,W
        addlw   0x30            ;Añadimos 30h para que el Ezweb la entienda
        movwf   EVAL
        ;BUFFER_TX[5Ah]=EVAL
        movlw   0x5A
        movwf   FSR
        movf    EVAL,W
        movwf   INDF


        ;EL5=1
        movlw   .01                     ;pasamos al siguiente estado
        movwf   EL5
        ;return()
        return                          ;volvemos

seg5_caso1

        ;EVAL=BUFFER RX[0x40]
        movlw   0x40            ;dirección donde debería de estar el id del Mando
        movwf   FSR
        movf    INDF,W
        movwf   ORIGEN

                ;if (ORIGEN==0x01)
                movf    ORIGEN,W        ;Comprobamos si es el mando 1
                sublw   0x01
                btfss   STATUS,Z
                goto $+4
                ;EL5=2
                movlw   .02                     ;pasamos al siguiente estado porque la trama es
del mando 1
                movwf   EL5
                ;return()
                return                          ;volvemos
                ;else           seguimos buscando

                ;if (ORIGEN==0x02)
                movf    ORIGEN,W        ;Comprobamos si es el mando 2
                sublw   0x02
                btfss   STATUS,Z
                goto $+4
```

```
                ;EL5=3
                movlw   .03                     ;pasamos al siguiente estado porque la trama es
del mando 2
                movwf   EL5
                ;return()
                return                          ;volvemos
                ;else           seguimos buscando

                ;if (ORIGEN==0x03)
                movf    ORIGEN,W        ;Comprobamos si es el mando 3
                sublw   0x03
                btfss   STATUS,Z
                goto $+4
                ;EL5=4
                movlw   .04                     ;pasamos al siguiente estado porque la trama es
del mando 3
                movwf   EL5
                ;return()
                return                          ;volvemos
                ;else           seguimos buscando

                ;if (ORIGEN==0x04)
                movf    ORIGEN,W        ;Comprobamos si es el mando 4
                sublw   0x04
                btfss   STATUS,Z
                goto $+4
                ;EL5=2
                movlw   .05                     ;pasamos al siguiente estado porque la trama es
del mando 4
                movwf   EL5
                ;return()
                return                          ;volvemos
                goto fin_eval_mando_error                   ;hemos recibido un dato no
válido, no enviamos nada


seg5_caso2              ;detectado Mando 1

        ;grabamos en el buffer de salida en el campo destino la dirección del mando

        bcf     STATUS,RP0
        bcf     STATUS,RP1      ;seleccionamos el banco 0

        ;EVAL=BUFFER_RX[0x47]
        movlw   0x47            ;dirección donde debería de estar el MODO en el que está el
mando
        movwf   FSR
        movf    INDF,W
        movwf   EVAL
        ;if (eval==000000 00)
                movf    EVAL,W
                sublw   .00
                btfss   STATUS,Z
                goto    $+4
                ;MODO_SYNC= XXXXXX 00
                bcf             MODO_SYNC,0                 ;actualizamos la variable
                bcf             MODO_SYNC,1
                ;fin eval mando
                goto fin_eval_mando

        ;if (eval==000000 01)
                movf    EVAL,W
                sublw   .01
                btfss   STATUS,Z
                goto    $+4
                ;MODO_SYNC= XXXXXX 01
                bsf             MODO_SYNC,0                 ;actualizamos la variable
                bcf             MODO_SYNC,1
                ;fin eval mando
                goto fin_eval_mando

        ;if (eval==000000 10)
                movf    EVAL,W
                sublw   .02
                btfss   STATUS,Z
                goto    $+4
                ;MODO_SYNC= XXXXXX 10
```

```
                bcf             MODO_SYNC,0                     ;actualizamos la variable
                bsf             MODO_SYNC,1
                ;fin eval mando
                goto fin_eval_mando

                goto fin_eval_mando_error                       ;hemos recibido un dato no
válido, no enviamos nada

seg5_caso3              ;detectado Mando 2

;grabamos en el buffer de salida en el campo destino la dirección del mando

        bcf     STATUS,RP0
        bcf     STATUS,RP1      ;seleccionamos el banco 0

        ;EVAL=BUFFER_RX[0x47]
        movlw   0x47            ;dirección donde debería de estar el MODO en el que está el
mando
        movwf   FSR
        movf    INDF,W
        movwf   EVAL
        ;***if (eval==000000 00)
                movf    EVAL,W
                sublw   .00
                btfss   STATUS,Z
                goto    $+4
                ;MODO_SYNC= XXXX 00 XX
                bcf             MODO_SYNC,2                             ;actualizamos la variable
a enviar al EZWEB
                bcf             MODO_SYNC,3
                ;fin eval mando
                goto fin_eval_mando

        ;if (eval==000000 01)
                movf    EVAL,W
                sublw   .01
                btfss   STATUS,Z
                goto    $+4
                ;MODO_SYNC= XXXXXX 01
                bsf             MODO_SYNC,2                     ;actualizamos la variable a
enviar al EZWEB
                bcf             MODO_SYNC,3
                ;fin eval mando
                goto fin_eval_mando

        ;if (eval==000000 10)
                movf    EVAL,W
                sublw   .02
                btfss   STATUS,Z
                goto    $+4
                ;MODO_SYNC= XXXXXX 10
                bcf             MODO_SYNC,2                     ;actualizamos la variable a
enviar al EZWEB
                bsf             MODO_SYNC,3
                ;fin_eval_mando
                goto fin_eval_mando

                goto fin eval_mando_error                       ;hemos recibido un dato no
válido, no enviamos nada

seg5_caso4              ;detectado Mando 3

;grabamos en el buffer de salida en el campo destino la dirección del mando

        bcf     STATUS,RP0
        bcf     STATUS,RP1      ;seleccionamos el banco 0

        ;EVAL=BUFFER RX[0x47]
        movlw   0x47            ;dirección donde debería de estar el MODO en el que está el
mando
        movwf   FSR
        movf    INDF,W
        movwf   EVAL
        ;if (eval==000000 00)
                movf    EVAL,W
                sublw   .00
                btfss   STATUS,Z
```

```
                goto    $+4
                ;MODO_SYNC= XX 00 XXXX
                bcf             MODO_SYNC,4             ;actualizamos la variable a enviar al
EZWEB
                bcf             MODO_SYNC,5
                ;fin_eval_mando
                goto fin_eval_mando

        ;if (eval==000000 01)
                movf    EVAL,W
                sublw   .01
                btfss   STATUS,Z
                goto    $+4
                ;MODO SYNC= XX 01 XXXX
                bsf             MODO_SYNC,4             ;actualizamos la variable a enviar al
EZWEB
                bcf             MODO_SYNC,5
                ;fin eval mando
                goto fin_eval_mando

        ;if (eval==000000 10)
                movf    EVAL,W
                sublw   .02
                btfss   STATUS,Z
                goto    $+4
                ;MODO SYNC= XX 10 XXXX
                bcf             MODO_SYNC,4             ;actualizamos la variable a enviar al
EZWEB
                bsf             MODO_SYNC,5
                ;fin eval mando
                goto fin_eval_mando

                goto fin_eval_mando_error                       ;hemos recibido un dato no
válido, no enviamos nada

seg5 caso5              ;detectado Mando 4

        ;grabamos en el buffer de salida en el campo destino la dirección del mando

        bcf     STATUS,RP0
        bcf     STATUS,RP1      ;seleccionamos el banco 0

        ;EVAL=BUFFER RX[0x47]
        movlw   0x47            ;dirección donde debería de estar el MODO en el que está el
mando
        movwf   FSR
        movf    INDF,W
        movwf   EVAL
        ;if (eval==000000 00)
                movf    EVAL,W
                sublw   .00
                btfss   STATUS,Z
                goto    $+4
                ;MODO SYNC=  00 XXXXXX
                bcf             MODO_SYNC,6             ;actualizamos la variable a enviar al
EZWEB
                bcf             MODO_SYNC,7
                ;fin eval mando
                goto fin_eval_mando

        ;if (eval==000000 01)
                movf    EVAL,W
                sublw   .01
                btfss   STATUS,Z
                goto    $+4
                ;MODO_SYNC= 01 XXXXXX
                bsf             MODO_SYNC,6                     ;actualizamos la variable a
enviar al EZWEB
                bcf             MODO_SYNC,7
                ;fin eval mando
                goto fin_eval_mando

        ;if (eval==000000 10)
                movf    EVAL,W
                sublw   .02
                btfss   STATUS,Z
                goto    $+4
```

```
                ;MODO_SYNC= 10 XXXXXX
                bcf             MODO_SYNC,6             ;actualizamos la variable a enviar al
EZWEB
                bsf             MODO_SYNC,7
                ;fin_eval_mando
                goto fin_eval_mando

                goto fin eval_mando_error                       ;hemos recibido un dato no
válido, no enviamos nada


;------------------FUNCIONES USADAS DENTRO DE EVALUAR_MANDO--------------------------------
---------

fin_eval_mando

        ;serial=1
        bsf     INDI,4          ;la trama ya está lista para ser enviada
        ;eval mando=0
        bcf     INDI,3
        ;DataReady=0
        bcf INDI,0              ;ya hemos terminado de trabajar con el buffer en entrada, por
lo que se puede machacar

        movlw   .00             ;inicializamos la variable EL5
        movwf   EL5

        ;return()
        return

fin eval mando error
        ;eval mando=0
        bcf     INDI,3
        ;DataReady=0
        bcf INDI,0              ;ya hemos terminado de trabajar con el buffer en entrada, por
lo que se puede machacar

        movlw   .00             ;inicializamos la variable EL5
        movwf   EL5

        ;return
        return

;------------------------------------------------------------------------------------------
-------

        END     ; Aquí no tiene que llegar nunca
```

4.3 Código Fuente Placa

Mando

4.3 Código Fuente Placa Mando

```
;*****************************************************************
;                                                                *
;    Archivo:      mando63.asm                                   *
;    Fecha:          mayo-2013                                   *
;    Version:            0.63                                    *
;                                                                *
;    Autor:        Amit Babani                                   *
;                                                                *
;                                                                *
;                                                                *
;*****************************************************************
;                                                                *
;    Archivos necesarios: P16F690.INC                            *
;                                                                *
;                                                                *
;*****************************************************************
;                                                                *
;    Descripción: Aplicación encargada de la gestión de los mandos    *
;    táctiles. Estos notificarán al pic principal si se ha activado   *
;    un pulsador para que este haga un cambio de MODO solicitar un.   *
;    cambio de modo.                                             *
;    El PIC principal notificará a los mandos de los cambios de MODO  *
;    mediante la línea serial. Los mandos indicarán mediante LEDs     *
;    el MODO en el que se encuentran. Cada mando estará asignado      *
;    a un conjunto de iluminación.                               *
;                                                                *
;                                                                *
;    Actualizaciones:                                            *
;       0.63 El max487 se pone en modo envío en la función enviar tx,  *
;         y en modo recepción en la función clear_error             *
;       0.62 Max487 Master control                                *
;       0.61 Corrección en bug de recepción del puerto serie      *
;       0.6 Implementación detección de errores con la función    *
;              clear error()                                      *
;       0.51 Corrección de funciones de envío y recepción y cambio *
;               en leds de salida                                 *
;       0.5  No se responde con trama de confirmación si el       *
;               cambio de modo se solicitó a través de puerto serie    *
;       0.48 Implementación de tramas en recepción                *
;       0.4  Implementación de read serial y evaluar rx           *
;    0.33 Implementación de enviar_tx                             *
;       0.32 Reordenamiento de variables empleadas                *
;    0.3  Se ha cambiado los puertos del led                      *
;    0.23 Todas las funciones implementadas.                      *
;    0.22 Cambiado el n° de estados en la función de lectura por el   *
;         puerto serie.                                           *
;                                                                *
;    0.21 Implementación de la función de lectura del puerto serie.   *
;                                                                *
;    0.2  Se ha reducido el número de estados de 5 a 3           *
;         Se ha aÃ±adido un LED rojo que se iluminará cuando se opere  *
;         en modo AUTO y otro verde que se iluminará al operar en modo *
;         OFF.                                                   *
;                                                                *
;                                                                *
;                                                                *
;*****************************************************************
;                                                                *
;                                                                *
;                            ----------------                    *
;                     VCC ---|Vdd          Vss|--- GND           *
;                            |                 |                 *
;                            |RA5          RA0|--- PULSADOR      *
;                            |RA4          RA1|---              *
;                            |RA3          RA2|---              *
;                            |                 |                 *
;                LED ON   ---|RC5          RC0|---              *
;                LED OFF  ---|RC4          RC1|---              *
;                LED AUTO ---|RC3          RC2|---              *
;             MASTER CONTROL ---|RC6            |                *
;          ERROR INDICATOR ---|RC7          RB4|---              *
```

```
;                           |           RB5|--- RX SERIAL           *
;              TX SERIAL ---|RB7         RB6|---                    *
;                           -----------------                      *
;                                                                  *
;                                                                  *
;                                                                  *
;                                                                  *
; -----------------------------------------------------------------

    #include <p16f690.inc>
       errorlevel      -302, -207

; --------------------------------------------------------------------------
; Bits de configuración: adapte los parámetros a su necesidad
  __config  _FCMEN_OFF& _IESO_OFF& _MCLRE_OFF& _WDT_OFF& _INTOSCIO
 ;
 ;  _FCMEN_OFF           ; -- fail safe clock monitor enable off
 ;   IESO OFF            ; -- int/ext switch over enable off
 ;   BOR ON              ; default, brown out reset on
 ;   CPD OFF             ; default, data eeprom protection off
 ;  _CP_OFF              ; default, program code protection off
 ;  _MCLR_OFF            ; -- use MCLR pin as digital input
 ;  _PWRTE_OFF           ; default, power up timer off
 ;   WDT OFF             ; -- watch dog timer off
 ;  _INTOSCIO            ; -- internal osc, RA6 and RA7 I/O
 ;



;definición de constantes
; ---------------CONSTANTES-----------------------------------------------------
radix   dec
HOST    equ 0x01       ;Dirección MAC de la máquina
CONTROL equ 0x10       ;Dirección MAC del PIC conectado al EZWEB
ID              equ 0x01       ;Id del conjunto de iluminación

AUTO    equ 0x00       ;Valor de la trama que indica modo AUTO
ON              equ 0x01       ;Valor de la trama que indica modo ON
OFF             equ 0x02       ;Valor de la trama que indica modo OFF

; ---------------VARIABLES------------------------------------------------------

; Indicador de uso general
                UDATA
INDI        RES 1 ; indicadores de carácter general

;cblock 0xA0
BUFFER_RX       RES 1                 ; variable usado como buffer de entrada en la
comunicación serial
CRC_RX          RES 1                 ; para el cálculo del crc de la trama recibida
LONG_RX         RES 1                 ; longitud del campo dato de la trama recibida
CONT_RX         RES 1                 ; contador de la posición del campo dato
;endc

; Variables usadas en enviar_tx()

EL3             RES 1          ; indica el estado en el que se encuentra la función
read_serial()
BUFFER_TX       RES 1          ; variable usado como buffer de entrada en la comunicación
serial
CONT_TX         RES 1          ; contador de la posición del campo dato


;cblock 0xB0
CRC_TX          RES 1             ; para el cálculo del crc de la trama recibida
LONG_TX         RES 1             ; longitud del campo dato de la trama recibida
;endc

; Variables usadas en read_serial()

EL2                RES 1 ; indica el estado en el que se encuentra la función read_serial()

; Variables usadas en mando()

EL1                    RES 1 ; indica el estado en el que se encuentra la función mando()
MODO            RES 1 ; determina el modo de la iluminación (0x00=OFF, 0x01=ON, 0x02=AUTO)
```

```
; Variables usadas en evaluar_rx()

MODO_RX    RES 1 ; variable donde se almacena el modo recibido a través del puerto serial


; Variables globales
INT VAR UDATA SHR
w saved      RES 1 ; variable utilizada para guardar contexto
status_saved RES 1 ; variable utilizada para guardar contexto
pclath_saved RES 1 ; variable utilizada para guardar contexto


; reiniciar vector
STARTUP CODE 0x000
    nop                     ; requerido para el depurado ICD2
    movlw   high start      ; cargar el byte superior de la etiqueta "start"
    movwf   PCLATH          ; inicializar PCLATH
    goto    start           ; ir al inicio del código principal

; vector de interrupciones
INT_VECTOR CODE 0x004
    goto    interrupt       ; ir al inicio del código de interrupción

; código reubicable
PROG CODE
interrupt
    movwf   w_saved         ; guardar contexto
    swapf   STATUS,w
    movwf   status saved
    movf    PCLATH,w        ; solamente se requiere si se utiliza más de la primera página
    movwf   pclath_saved
    clrf    PCLATH
    ; << agregue el código de interrupción >>
    movf    pclath_saved,w ; restaurar contexto
    movwf   PCLATH
    swapf   status_saved,w
    movwf   STATUS
    swapf   w_saved,f
    swapf   w_saved,w
    retfie


;-------------------------------------------------------------------
;comienzo de la aplicación
start

;---------------------INICIO----------------------------------------
;-------------------------------------------------------------------


;inicialización
init

       ;inicialización del buffer_tx
       bcf     STATUS,RP0
       bcf     STATUS,RP1     ;seleccionamos el banco 0

       movlw   0x30           ;apuntamos a la 1° posición del buffer
       movwf   FSR
       movlw   0xAB           ;Cargamos el PREAMBULO
       movwf   INDF
       incf    FSR,1          ;vamos a la siguiente posición del buffer
       movlw   CONTROL        ;Cargamos la dirección DESTINO
       movwf   INDF
       incf    FSR,1          ;vamos a la siguiente posición del buffer
       movlw   HOST           ;Cargamos la dirección HOST
       movwf   INDF
       incf    FSR,1          ;vamos a la siguiente posición del buffer
       movlw   0x07           ;Cargamos el valor TIPO con la longitud del campo datos (7)
       movwf   INDF
       incf    FSR,1          ;vamos a la siguiente posición del buffer
       movlw   0x4C           ;1° byte del campo datos "L"
       movwf   INDF
       incf    FSR,1          ;vamos a la siguiente posición del buffer
       movlw   0xFF           ;2° byte del campo datos " "
       movwf   INDF
```

```
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   ID                      ;3° byte del campo datos ID
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x45            ;4° byte del campo datos "E"
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0xFF            ;5° byte del campo datos " "
        movwf   INDF
        incf    FSR,1           ;vamos a la siguiente posición del buffer
        movlw   0x4D            ;6° byte del campo datos "M"
        movwf   INDF


        ;inicializamos los indicadores
        movlw   b'00000000'
;               |   ||||
;               |   |||-- DataReady usado en: read_serial() y evaluar_rx() -> Indica que hay
un dato (buffer rx) listo para ser evaluado
;                   |   ||--- Click usado en: leer_teclado() y mando() -> Indica que se ha
pulsado un botón
;                   |   |---- Send usado en: mando() y enviar_tx() -> Buffer_tx listo para
ser enviado
;                   |   ----- Receive usado en evaluar_rx() y mando() -> Indica que el dato
recibido está evaluado
;                   --------- Flanco usado en leer_teclado() -> Para detectar el flanco de
subida del pulsador
        movwf   INDI

        ;configuramos los puertos
        bcf     STATUS,RP0      ;seleccionamos el banco 0
        bcf     STATUS,RP1

        clrf    PORTA           ;inicializamos puerto A
        clrf    PORTB           ;inicializamos puerto B
        clrf    PORTC           ;inicializamos puerto C

        bsf     STATUS,RP1      ;seleccionamos el banco 2

        clrf    ANSEL           ;modo digital
        clrf    ANSELH          ;modo digital

        ;pulsador
        bsf     STATUS,RP0      ;seleccionamos el banco 1
        bcf     STATUS,RP1

        movlw   b'00000001'     ;configuramos RA0 como entrada para el pulsador
                ;                       |
                ;                       |
                ;                       |---- RA0 = Input
        movwf   TRISA

        ;leds
        movlw   b'00000000'     ;configuramos RC3 - RC5 como salida
        ;           ||||
        ;           |||--RC3 = Led Auto
        ;           |||- RC4 = Led Off
        ;                 ||-- RC5 = Led On
        ;                 |--- RC6 = Max487 Control

        movwf   TRISC


        ;configuración del puerto serie
bcf     TRISB,7
bcf     TRISB,6


    movlw   b'01100000'     ; Fosc= 4Mhz
    movwf   OSCCON

Stable
          btfss   OSCCON,HTS      ; osc stable? yes, skip, else
             goto    Stable          ; test again

        movlw   .25       ; [(4Mhz/9615)/64]-1
        movwf   SPBRG      ; 9615 baudios con Fosc=4MHz, 8bits
```

```
       movlw   .00    ;
    movwf   SPBRGH


  movlw   b'00100100'     ; '0-------' CSRC, n/a (async)
                          ; '-0------' TX9 off, 8 bits
                          ; '--1-----' TXEN, tx enabled
                          ; '---0----' SYNC, async mode
                          ; '----0---' SENDB, send brk
                          ; '-----1--' BRGH, high speed
                          ; '------00' TRMT, TX9D
  movwf   TXSTA           ;
  bcf     STATUS,RP0      ; banco 0
  movlw   b'10010000'     ; '1-------' SPEN, port enabled
                          ; '-0------' RX9 off, 8 bits
                          ; '--0-----' SREN, n/a (async)
                          ; '---1----' CREN, rx enabled
                          ; '----0---' ADDEN off
                          ; '-----000' FERR, OERR, RX9D
  movwf   RCSTA           ;
  movf    RCREG,W         ; flush Rx Buffer
  movf    RCREG,W         ;



       ;**MODO=AUTO, más adelante MODO=read_serial() no saldrá del bucle hasta que reciba
un dato
       ;case_auto()
       movlw   AUTO
       movwf   MODO
       call    case_auto             ;modifica el estado de los leds, también activará el
flag de envio_rx

       movlw   .00
       movwf   EL2            ; inicializamos en el estado 0, la función de lectura de
puerto serie

       movlw   .00
       movwf   EL3            ; inicializamos en el estado 0, la función de escritura de
puerto serie

       movlw   .00
       movwf   CONT_TX        ; inicializamos el contador para el buffer de salida

       movlw   .00
       movwf   CRC_TX         ; inicializamos el CRC del buffer de salida

;---------------------BUCLE PRINCIPAL--------------------------------
;--------------------------------------------------------------------

repetir

       ;repeat

       ;read_serial()
       call read_serial      ;función encargada de leer la trama recibida por el puerto
serial

       ;evaluar rx()
       call evaluar_rx              ;evalúa el dato que se recibe por el puerto serial y
almacena el MODO recibido en MODO_RX

       ;leer teclado()
       call leer teclado     ;función encargada en detectar si se ha pulsado una tecla, si
lo hace pone el flag click a 1

       ;mando()
       call mando            ;función encargada de gestionar el MODO y los LEDS asignados
al mandos

       ;enviar tx()
       call enviar_tx ;encargada de enviar una trama al PIC principal la solicitud del
cambio de MODO realizado por el teclado

       ;clear error()
       call clear error             ;Función encargada de detectar errores y corregirlos
en su caso, ponde el MAX487 en recepción
```

```
        goto repetir          ;fin repeat


        ;return()
        return


;--------------------------------CLEAR ERROR-------------------------------
;--------------------------------------------------------------------------
;
;       Comprobamos si los bits OERR y FERR ha detectado un error,                  *
;       si lo han detectado reinciamos la USART y vaciamos buffer.                  *
;
;       OERR= Overrun error
;       FERR= Frame error
;
;--------------------------------------------------------------------------


clear_error

        bcf     STATUS,RP0      ;seleccionamos banco 0
        bcf     STATUS,RP1


        ;if (OERR==1)
        btfsc   RCSTA,OERR      ;si se detecta error de desbordamiento
        ;reset_usart()
        goto reset_oerr               ;ejecutamos función de corrección

        ;if (FERR==1)
        btfsc   RCSTA,FERR      ;si detectamos error de ventana
        ;reset_ferr()
        goto reset_ferr               ;ejecutamos función de corrección

        ;return()
        return


reset oerr                            ;función de corrección en desbordamiento
        bcf             RCSTA,CREN      ;desactivamos eusart
        nop
        nop
        bsf             RCSTA,CREN      ;activamos eusart
        nop
        nop

        ;if (RCIF==1)                 ;vaciando el buffer
        btfsc   PIR1,RCIF             ;hay algún dato?
        ;empty_buffer()
        goto empty_buffer
        ;return()
        return                  ;no se ha recibido ningún dato, volvemos a la función
principal

empty_buffer
        movf    RCREG,W
        nop
        nop
        ;if (RCIF==1)
        btfsc   PIR1,RCIF       ;repetimos el vaciado hasta que no quede ningún dato
        ;empty buffer()
        goto empty buffer
        ;return()
        return

reset ferr                                      ;función encargada de corregir errores de
ventana
        ;bcf    TXSTA,TXEN
        ;bsf    TXSTA,TXEN
        ;return()
        return


;--------------------------------LEER_TECLADO-----------------------------------------------
------------------
```

```
;-------------------------------------------------------------------------------------------
------------------
;
;Se intenta detectar un cambio de flanco, para evitar detectar una larga pulsación como
varias pulsaciones
;                      ____________________________________________________
;RA0                 |
                    |
;                    |
                              |___________
;     Flanco=1        Click=1 -> Flanco=0   Click=0                Flanco=1
;
;-------------------------------------------------------------------------------------------

leer_teclado

        ;if (SEND==1)
        btfsc   INDI,2
        ;return()
        return                  ;se está enviando un dato, volvemos

        ;switch (RA0)
        bcf     STATUS,RP0      ;seleccionamos el banco 0
        bcf     STATUS,RP1

        btfss   PORTA,0         ;leemos el puerto de entrada

        ;case 0
        goto    RA0_0           ;si leemos un 0
        ;case 1
        goto    RA0_1           ;si leemos un 1

RA0_0
        ;Flanco=1
        bsf     INDI,7
        ;return()
        return

RA0_1
        ;if (Flanco==1)
        btfss   INDI,7
        return
        ;Click=1
        bsf     INDI,1          ;activamos el flag como que se ha pulsado el botón porque
hemos detectado un cambio de transición de 0->1
        ;Flanco=0
        bcf     INDI,7          ;ponemos el detector de cambio de estado a 0
        ;return()
        return

;---------------------------------MANDO-------------------------------------------------
;-------------------------------------------------------------------------------------------

mando                                                   ;selector de modo
        movlw   HIGH switch_mando
        movwf   PCLATH

switch mando                    ;switch(MODO)
        movf EL1, W             ; cargo W con el estado (0x00=Auto 0x01=On 0x02=Off)
        addwf PCL, F            ; modifico el contador del programa en función del estado
coincidente

        ;case(auto)
        goto seg1_caso0                 ; modo AUTO
        ;case(on)
        goto seg1_caso1                 ; modo ON
        ;case(off)
        goto seg1 caso2                 ; modo OFF
        ;end switch


;-------------------------------------------ESTADOS MANDO----------------------------------


seg1 caso0                      ;modo AUTO
```

```
        ;if(receive==1)
        btfsc   INDI,3          ;comprueba si el flag de dato recibido por serial está activo
        ;cambiar rx()
        call    cambiar rx      ;si lo está, esta función cambiará el modo al modo recibido a
través del puerto serial

        ;if(click==1)
        btfss   INDI,1          ;si se ha pulsado un botón del teclado
        goto    switch1_0

        ;MODO=ON
        movlw   OFF             ;actualizamos el modo
        movwf   MODO
        ;send=1
        bsf     INDI,2          ;activamos el flag para que se notifique a través del puerto
serie el cambio de modo
        ;click=0
        bcf     INDI,1          ;ponemos el flag de click a 0

switch1 0
        movlw   HIGH switch1_0a         ;movemos a W la parte alta donde está mapeado el
switch
        movwf   PCLATH                  ;modificamos PCLATH (parte alta de PCL)

switch1 0a                              ;switch(modo)  --> función encargada de modificar EL1,
en función al modo en el que estamos trabajando
        movf MODO, W            ; cargo W con el estado (0x00=Auto 0x01=On 0x02=Off)
        addwf PCL, 1            ; modifico el contador del programa en función del caso
coincidente

        return                          ;case(auto), no tenemos que modificar nada, volvemos a
main()
        goto case_on            ;case(on)
        goto case_off           ;case(off)

;--------------------------------------------------------------------------------

seg1_caso1                      ;modo ON

        ;if(receive==1)
        btfsc   INDI,3          ;comprueba si el flag de recibido dato por serial está activo
        ;cambiar rx()
        call    cambiar rx      ;si lo está, esta función cambiará el modo al modo recibido a
través del puerto serial


        ;if(click==1)
        btfss   INDI,1          ;si se ha pulsado un botón del teclado
        goto    switch1_1

        ;MODO=OFF
        movlw   AUTO    ;actualizamos el modo
        movwf   MODO
        ;send=1
        bsf     INDI,2          ;activamos el flag para que se notifique a través del puerto
serie el cambio de modo
        ;click=0
        bcf     INDI,1          ;ponemos el flag de click a 0

switch1_1
        movlw   HIGH switch1_1a         ;movemos a W la parte alta donde está mapeado el
switch
        movwf   PCLATH                  ;modificamos PCLATH (parte alta de PCL)

switch1 1a              ;switch(modo)  --> función encargada de modificar EL1, en función al
modo en el que estamos trabajando
        movf MODO, W    ; cargo W con el estado (0x00=Auto 0x01=On 0x02=Off)
        addwf PCL, F    ; modifico el contador del programa en función del caso coincidente

        goto case auto          ;case(auto)
        return                          ;case(on), no tenemos que modificar nada, volvemos a
main()
        goto case_off           ;case(off)

;--------------------------------------------------------------------------

seg1_caso2                      ;modo OFF
```

```
        ;if(receive==1)
        btfsc   INDI,3          ;comprueba si el flag de recibido dato por serial está activo
        ;cambiar rx()
        call    cambiar_rx      ;si lo está, esta función cambiará el modo al modo recibido a
través del puerto serial


        ;if(click==1)
        btfss   INDI,1          ;si se ha pulsado un botón del teclado
        goto    switch1_2

        ;MODO=AUTO
        movlw   ON              ;actualizamos el modo
        movwf   MODO
        ;send=1
        bsf     INDI,2          ;activamos el flag para que se notifique a través del puerto
serie el cambio de modo
        ;click=0
        bcf     INDI,1          ;ponemos el flag de click a 0

switch1_2
        movlw   HIGH switch1_2a         ;movemos a W la parte alta donde está mapeado el
switch
        movwf   PCLATH                  ;modificamos PCLATH (parte alta de PCL)


switch1_2a              ;switch(modo)  --> función encargada de modificar EL1, en función al
modo en el que estamos trabajando
        movf MODO, W    ; cargo W con el estado (0x00=Auto 0x01=On 0x02=Off)
        addwf PCL, F    ; modifico el contador del programa en función del caso coincidente

        goto case_auto          ;case(auto)
        goto case_on            ;case(on),
        return                          ;case(off), no tenemos que modificar nada, volvemos a
main()



;------------funciones usadas dentro de MANDO---------------------------------

cambiar rx
        ;MODO=MODO RX
        movf    MODO_RX, W
        movwf   MODO

        ;receive=0
        bcf     INDI,3          ;ponemos el flag de receive a 0

        ;return()
        return


;------------casos posibles, usados dentro de MANDO-----------------------------


;casos encargados de modificar la variable EL1 y los LEDS según el modo en el que estamos
operando.
;la variable EL1 es usada en la función mando para realizar el salto al estado correcto

case_auto
        ;EL1=00
        ;goto seg1 caso0
        movlw   AUTO            ;carga 0x00 en EL1
        movwf   EL1
        ;led_red=On
        ;led_green=Off
        movlw   b'00001000'     ;ponemos a nivel alto RC6
        movwf   PORTC

        ;return
        return


case on
        ;EL1=01
        ;goto seg1_caso1
```

```
        movlw   ON              ;carga 0x01 en EL1
        movwf   EL1
        ;led red=Off
        ;led green=Off
        movlw   b'00100000'     ;ponemos a nivel alto RC5
        movwf   PORTC

        ;return
        return

case_off
        ;EL1=02
        ;goto seg1 caso2
        movlw   OFF             ;carga 0x02 en EL1
        movwf   EL1
        ;led_red=Off
        ;led_green=On
        movlw   b'00010000'     ;ponemos a nivel alto RC4
        movwf   PORTC

        ;return
        return

;fin de los casos posibles

;---------------------------------Enviar tx---------------------------------
;*****************************************************************************
; Formato de trama                                                           *
;   El formato a usar es idéntico a la trama IEEE 802.3 con la               *
;   omisión del campo 'preámbulo'                                            *
;                                                                            *
;            |-----------------------------------------------|               *
;            | SOF | Destino | Origen | Tipo | Datos  | FCS |               *
;            |-----------------------------------------------|               *
; N° Bytes      1       1         1       1     Var.     4                   *
;                                                                            *
;                                                                            *
; Formato del campo de Datos                                                 *
;                                                                            *
;        L00E0M0  (7 bytes)                                                  *
;         || | |                                                             *
;         || | |                                                             *
;         || | -- Modo de operación (0x00-Auto,0x01-On,0x02-Off)             *
;         || |                                                               *
;         || --- Estado de la iluminación (0-Off, 1-On)                      *
;         ||                                                                 *
;         ---- Identificador del juego de iluminación (00-99)               *
;
;
;  Buffer a enviar
;
;   |-------------------------------------------------------------|
;   |0xAB|CONTROL|HOST|0X07|"L" 0xFF ID "E" 0xFF "M" MODO|CRC TX|     *
;   |-------------------------------------------------------------|     *
;     |_____________________________________________|
;          Buffer almacenado en memoria
;
;-----------------------------------------------------------------------------
org 0x100
enviar_tx
        ;if (SEND==0)
        btfss   INDI,2
        ;return()
        return                  ;no hay ningún dato a enviar, volvemos

        movlw   HIGH enviar_tx
        movwf   PCLATH

        ;ponemos el MAX485 en modo de envío
        ;PORTC[6]=1
        bsf             PORTC,6
        nop

        ;switch(estado)
        movf EL3, W     ; cargo W con el estado
        addwf PCL, F    ; modifico el contador del programa en función del estado coincidente
```

```
        goto seg3_caso0
        goto seg3 caso1
        goto seg3 caso2
        goto seg3_caso3
        goto seg3_caso4


;--------------------------------inicio de estados--------------------------

seg3_caso0     ;estado encargado de comprobar si TXREG está vacio para poder enviar un byte


               ;if (TXIF==0)
               btfss   PIR1,TXIF      ;comprobar si TXREG está vacio
               ;return()
               return                 ;no se ha recibido ningún dato, volvemos a la función
principal

               ;if (TXIF==1)

               ;Envio del buffer_tx

               bcf     STATUS,RP0
               bcf     STATUS,RP1     ;seleccionamos el banco 0

               ;N=CONTADOR_TX+0x30
               movlw   0x30           ;apuntamos a la 1° dirección del buffer
               addwf   CONT_TX,W      ;sumamos al puntador el valor del contador
               movwf   FSR

               ;TXREG=BUFFER[N]
               movf    INDF,W
               movwf   TXREG



               ;CRC_TX=CRC_TX+BUFFER[N]
               movf    INDF,W
               addwf   CRC_TX,1       ;actualizamos el valor de CRC
               ;CONT TX
               incf    CONT_TX,1              ;incrementamos el contador

               movf    CONT_TX,W
               xorlw   .10            ;el buffer tiene una longitud de 10 bytes
               btfsc   STATUS,Z
               ;if(CONT TX=10)
               goto    fin tx         ;hemos terminado de enviar el buffer
               ;if(CONT_TX!=10)
               goto    fin_no_tx      ;no hemos terminado de enviar el buffer

fin_tx
               ;CONT TX=0
               movlw   0x00
               movwf   CONT_TX        ;inicializamos el contador
               ;EL3=2
               movlw   .01
               movwf   EL3            ;vamos al siguiente estado
               ;return()
               return

fin_no_tx

               ;return()      ;seguimos en el mismo estado
               return

;-----------------------------------------------------------------------

seg3 caso1             ;estado encargado de comprobar si TXREG está vacio para poder enviar
un byte



               ;if (TXIF==0)
               btfss   PIR1,TXIF      ;comprobar si TXREG está vacio
               ;return()
```

```
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (TXIF==1)

                ;TXREG=MODO
                movf    MODO,W          ;envío del modo en el que el mando está trabajando
                movwf   TXREG


                ;CRC_TX=CRC_TX+MODO
                addwf   CRC_TX,1        ;actualizamos el valor de CRC

                ;EL3=2;
                movlw   0x02            ;saltamos al siguiente estado
                movwf   EL3

                ;return()
                return

;-------------------------------------------------------------------------

seg3_caso2              ;estado encargado de comprobar si TXREG está vacio para poder enviar
un byte



                ;if (TXIF==0)
                btfss   PIR1,TXIF       ;comprobar si TXREG está vacio
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (TXIF==1)

                ;TXREG=CRC TX
                movf    CRC TX,W        ;envio del CRC
                movwf   TXREG

                ;EL3=0;
                movlw   .03             ;volvemos al estado principal a la espera de enviar
otra trama
                movwf   EL3

                ;return()
                return

;-----------------------------------------------------------------------------

seg3_caso3
                ;if (TXIF==0)
                btfss   PIR1,TXIF       ;comprobar si TXREG está vacio
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal


                ;EL3=0;
                movlw   .04             ;volvemos al estado principal a la espera de enviar
otra trama
                movwf   EL3

                movlw   .250            ;iniciamos contador de delay
                movwf   CRC_TX          ;empleamos la variable CRC_TX ya que no la estamos
usando ahora
                return

seg3 caso4

                nop
                ;DELAY--
                decfsz  CRC_TX,1
                ;if (delay!=0) return
                return

                ;SEND=0
```

```
                bcf     INDI,2          ;cambiamos el flag para indicar que la trama ya está
enviada

                ;EL3=0;
                movlw   .00             ;volvemos al estado principal a la espera de enviar
otra trama
                movwf   EL3

                movlw   .00
                movwf   CRC_TX          ; inicializamos el CRC del buffer de salida

                bcf     PORTC,6
                return

;---------------------READ SERIAL-----------------------------------------
;-------------------------------------------------------------------------

;*************************************************************************
; Formato de trama                                                       *
;   El formato a usar es idéntico a la trama IEEE 802.3 con la           *
;   omisión del campo 'preámbulo'                                        *
;                                                                        *
;            |-----------------------------------------------|           *
;            | SOF | Destino | Origen | Tipo | Datos  | FCS |           *
;            |-----------------------------------------------|           *
; N° Bytes      1       1         1       1     Var.     4               *
;                                                                        *
;                                                                        *
; Formato del campo de Datos                                             *
;                                                                        *
;        L00E0M0  (7 bytes)                                              *
;         || | |                                                         *
;         || | |                                                         *
;         || | -- Modo de operación (0x00-Auto,0x01-On,0x02-Off)        *
;         || |                                                           *
;         || --- Estado de la iluminación (0-Off, 1-On)                 *
;         ||                                                            *
;         ---- Identificador del juego de iluminación (00-99)           *
;                                                                        *
; En esta función solo usaremos el dato del modo de operación           *
;                                                                        *
;-------------------------------------------------------------------------


       org 0x200
read_serial    ;selector de estado


       ;if (dataready==1)
       btfsc   INDI,0          ;el último dato recibido aún no ha sido evaluado, así evitamos
machacar el buffer
       ;return()
       return


       ;if (SEND==1)
       btfsc   INDI,2
       ;return()
       return                  ;se está enviando un dato, volvemos


    bcf     STATUS,RP1      ; bank 0
    bcf     STATUS,RP0

       movlw   HIGH read_serial
       movwf   PCLATH

       ;switch(estado)
       movf EL2, W    ; cargo W con el estado (0 --> 5)
       addwf PCL, F   ; modifico el contador del programa en función del estado coincidente

       goto seg2_caso0
       goto seg2_caso1
       goto seg2 caso2
       goto seg2 caso3
       goto seg2 caso4
       goto seg2_caso5
```

```
;---------------------ESTADOS READ_SERIAL-----------------------------------


seg2_caso0     ;estado encargado de comprobar si se ha recibido un byte

               ;if (RCIF==0)
               btfss   PIR1,RCIF      ;comprobar si se ha recibido un dato
               ;return()
               return                 ;no se ha recibido ningún dato, volvemos a la función
principal


               ;if (RCIF==1)

               ;Recepción del preámbulo

               ;if(RCREG=="AB")
               movf    RCREG,W        ;Comprobamos el byte recibido con el preambulo (AB =
10101011)
               xorlw   0xAB
               btfsc   STATUS,Z
               goto AB_OK
               ;else
               goto AB_no_OK

AB_OK          ;EL2=1

               movlw   .01            ; Si es el preambulo vamos al caso 1
               movwf   EL2

               ;return()
               return


AB no OK       ;EL2=0
               ; Si no es el preambulo volvemos al caso 0


               ;return()
               return


;-------------------------------------------------------------------------------


seg2_caso1     ;estado encargado de comprobar si se ha recibido un byte

               ;if (RCIF==0)
               btfss   PIR1,RCIF      ;comprobar si se ha recibido un dato
               ;return()
               return                 ;no se ha recibido ningún dato, volvemos a la función
principal

               ;if (RCIF==1)

       ;Recepción de campo de DESTINO


               ;if (RCREG==HOST)
               movf    RCREG,W        ;Si la trama no está destinada a este periférico,
volvemos al estado inicial caso 0
               xorlw   HOST           ;El valor de HOST está definido en los valores
iniciales
               btfsc   STATUS,Z
               goto    HOST_OK

               ;else
               goto HOST_no_OK

HOST_OK        ;CRC_RX=RCREG


               movlw   HOST
               movwf   CRC_RX         ; Inicializamos el valor de CRC con el valor de HOST
               ;EL2=2
```

```
                movlw   .02             ;carga 0x02 en EL2, y saltamos al caso 2
                movwf   EL2
                ;return()
                return


HOST_no_OK      movlw   .00             ;carga 0x00 en EL2, y volvemos al caso 0

                movwf   EL2
                ;return()
                return


;--------------------------------------------------------------------------------

seg2_caso2      ;estado encargado de comprobar si se ha recibido un byte

                ;if (RCIF==0)
                btfss   PIR1,RCIF       ;comprobar si se ha recibido un dato
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (RCIF==1)

        ;Recepción del campo ORIGEN

                ;if (RCREG==CONTROL)
                movf    RCREG,W         ;Si la trama no es enviada por el PIC Principal
volvemos al estado inicial 0
                xorlw   CONTROL         ;El valor de CONTROL está definido en los valores
iniciales
                btfsc   STATUS,Z
                goto    CONTROL_OK

                ;else
                goto CONTROL_no_OK

CONTROL_OK              ;CRC_RX=CRC_RX+RCREG
                movlw   CONTROL_OK
                addwf   CRC_RX,1                ; Actualizamos el valor del CRC
                ;EL2=3
                movlw   .03             ;carga 0x03 en EL2, y saltamos al caso 3
                movwf   EL2
                ;return()
                return


CONTROL_no_OK                   ;carga 0x00 en EL2, y volvemos al caso 0
                movlw   .00
                movwf   EL2
                ;return()
                return

;--------------------------------------------------------------------------------

seg2_caso3      ;estado encargado de comprobar si se ha recibido un byte

                ;if (RCIF==0)
                btfss   PIR1,RCIF       ;comprobar si se ha recibido un dato
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (RCIF==1)

                ;Recepción del campo TIPO, definirá el tamaño del campo DATO

                ;LONG RX=RCREG
                movf    RCREG,W
                movwf   LONG_RX

                ;CRC_RX=CRC_RX+RCREG
                movf    LONG RX,W
                addwf   CRC_RX,1

                ;i=0;
```

```
                movlw   0x00            ;inicializamos el contador para el campo datos
                movwf   CONT_RX

                ;EL2=4
                movlw   .04             ;carga 0x08 en EL2, y saltamos al caso 4
                movwf   EL2

                ;return()
                return


;--------------------------------------------------------------------------------

seg2_caso4      ;estado encargado de comprobar si se ha recibido un byte

                ;if (RCIF==0)
                btfss   PIR1,RCIF       ;comprobar si se ha recibido un dato
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (RCIF==1)


                ;Recepción del campo DATO

                ;inicializamos el inicio del buffer para almacenar el campo de datos

                bcf     STATUS,RP0      ;seleccionamos el banco 0
                bcf     STATUS,RP1

                movlw   0x40            ;dirección de inicio para el buffer de entrada
                addwf   CONT_RX,W       ;sumamos a la posición el número del contador
                movwf   FSR


                ;BUFFER RX[i]=RCREG
                movf    RCREG,W
                movwf   INDF            ;grabamos el byte recibido del campo dato en el buffer
                ;CRC_RX=CRC_RX+RCREG
                movf    INDF,W
                addwf   CRC_RX,1
                ;LONG--
                decf    LONG_RX,1
                ;i++
                incf    CONT_RX,1


                ;if (LONG_RX==0)
                movf    LONG_RX,W
                xorlw   .0
                btfsc   STATUS,Z
                goto    FIN_DATO        ;fin de la lectura del campo DATO, saltamos al
siguiente caso
                ;else
                goto    FIN_no_DATO     ;aún que por recibir bytes de la trama datos, volvemos
al estado anterior

FIN_DATO
                ;EL2=5
                movlw   .05
                movwf   EL2             ;carga 0x05 en EL2, y saltamos al caso 5
                ;return()
                return

FIN_no_DATO
                ;volvemos a este caso hasta que se reciba la trama completa
                ;return()
                return



;--------------------------------------------------------------------------------

seg2_caso5      ;estado encargado de comprobar si se ha recibido un byte

                ;if (RCIF==0)
```

```
                btfss   PIR1,RCIF       ;comprobar si se ha recibido un dato
                ;return()
                return                  ;no se ha recibido ningún dato, volvemos a la función
principal

                ;if (RCIF==1)

                ;Recepción del CRC y **verificación

                ;EL2=0
                movlw   .0              ;carga 0x00 en EL2, y volvemos al caso inicial
                movwf   EL2
                ;if(CRC RX==RCREG)
                movf    RCREG,W         ;comprobamos el CRC recibido con el calculado


CRC_OK          ;DataReady=1
                bsf     INDI,0          ;marcamos el flag de nuestro indicador como que hay un
dato
                ;return()
                return




;---------------------------------EVALUAR_RX-----------------------------------
;------------------------------------------------------------------------------
;**********************************************************************
; Formato del campo de Datos                                          *
;                                                                     *
;        L00E0M0  (7 bytes)                                           *
;         || | |                                                     *
;         || | |                                                     *
;         || | -- Modo de operación (0x00-Auto,0x01-On,0x02-Off)     *
;         || |                                                       *
;         || --- Estado de la iluminación (0-Off, 1-On)             *
;         ||                                                        *
;         ---- Identificador del juego de iluminación (00-99)       *
;**********************************************************************

evaluar rx
                ;if (DATAREADY==0)
                btfss   INDI,0
                ;return()
                return                  ;no tenemos un dato listo para ser evaluado

                ;else
                bcf     STATUS,RP0      ;seleccionamos el banco 0
                bcf     STATUS,RP1

                movlw   0x46            ;dirección donde debería de estar el MODO de operación
recibido  (40+6=46)
                movwf   FSR
                movf    INDF,W          ;cargamos el dato del buffer
                movwf   MODO_RX         ;almacenamos el valor en modo_rx

                ;if (MODO RX==0x00)
                movf    MODO_RX,W       ;Comprobamos el byte 0x00
                xorlw   0x00
                btfsc   STATUS,Z
                ;RECEIVE=1              ;indicamos que tenemos un dato válido
                bsf     INDI,3

                ;if (MODO_RX==0x01)
                movf    MODO_RX,W       ;Comprobamos el byte 0x01
                xorlw   0x01
                btfsc   STATUS,Z
                ;RECEIVE=1              ;indicamos que tenemos un dato válido
                bsf     INDI,3

                ;if (MODO_RX==0x02)
                movf    MODO RX,W       ;Comprobamos el byte 0x02)
                xorlw   0x02
                btfsc   STATUS,Z
                ;RECEIVE=1              ;indicamos que tenemos un dato válido
```

```
                bsf     INDI,3

                ;DATAREADY=0
                bcf     INDI,0          ;indicamos que estamos listos para recibir otro dato

                ;return()
                return


;*******************************************************************************


        END     ; Aquí no tiene que llegar nunca
```

4.4 Código Fuente

Página Web

4.4 Código Fuente Página Web

```
<------ archivo: index.html ------->

<html>

<!--CABECERA Y CARGA DE ARCHIVOS-->
      <head>
      <title>DomoNet 2.0</title>
      <script type="text/javascript" src="scripts.js"></script>
      <LINK REL=StyleSheet HREF="style.css" TYPE="text/css">
      </head>

<body onLoad="init();">

<!--ESQUEMA DE HABITACIÓN-->

      <div class="domo" style="z-index: 5;">
      <img src="domoout.png" border="0"></div>

      <div class="domo" style="z-index: 1;">
      <img src="domoin.png" ></div>

      <div class="domo"  style="z-index: 3;">
      <img src="luz1.png" id="bulb1" class="bulb" ></div>

      <div class="domo" style="z-index: 3;">
      <img src="luz2.png" id="bulb2" class="bulb" ></div>

      <div class="domo" style="z-index: 3;">
      <img src="luz3.png" id="bulb3" class="bulb"  ></div>

      <div class="domo"  style="z-index: 3;">
      <img src="luz4.png" id="bulb4" class="bulb" ></div>


<!--CONTROLES-->

      <div class="mando"  style="z-index: 3;">
      <table>
      <tR>
      <tH>LUZ 1<br><img class="pulsar"
onClick="pulsar('estado1','bulb1')" src=boton.png ></tH>
      <tH>LUZ 2<br><img class="pulsar"
onClick="pulsar('estado2','bulb2')" src=boton.png></tH>
      <tH>LUZ 3<br><img class="pulsar"
onClick="pulsar('estado3','bulb3')" src=boton.png></tH>
      <tH>LUZ 4<br><img class="pulsar"
onClick="pulsar('estado4','bulb4')" src=boton.png>

      </tH></tR>
      <tR>
      <th class="estado" id="estado1">AUTO</th>
      <tH class="estado" id="estado2">AUTO</tH>
      <tH class="estado" id="estado3">AUTO</tH>
      <tH class="estado" id="estado4">AUTO</tH>
      </tR>
      </table>
      </div>
```

```
</body></html>


<------ archivo: script.js ------->

function encender(bombilla, estado){

      document.getElementById(bombilla).style.MozOpacity=0.9; //Mozilla
      document.getElementById(bombilla).style.opacity=0.9;
      document.getElementById(estado).style.backgroundColor = "#01DF01";
//verde
      document.getElementById(estado).innerHTML  = "ON";

}

function apagar(bombilla, estado){
      document.getElementById(bombilla).style.MozOpacity=0; //Mozilla
      document.getElementById(bombilla).style.opacity=0;
      document.getElementById(estado).style.backgroundColor = "#FF0000";
//rojo
      document.getElementById(estado).innerHTML  = "OFF";
}

function automatico(bombilla, estado){
      document.getElementById(bombilla).style.MozOpacity=0.5; //Mozilla
      document.getElementById(bombilla).style.opacity=0.5;
      document.getElementById(estado).style.backgroundColor = "#fff727";
//amarillo
      document.getElementById(estado).innerHTML  = "AUTO";
}



// función invocada al pulsar un boton
function pulsar(estado, bombilla){              //recibe el estado actual
de la LUZ y el ID de la luz

      //switch que identifica que bombilla tiene que cambiar, y envía la
primera trama según el caso

      switch(bombilla) {
            case "bulb1":
                  window.open('bulb1.htm','pop','width=400,height=200');
            break;

            case "bulb2":
                  window.open('bulb2.htm','pop','width=400,height=200');
            break;

            case "bulb3":
                  window.open('bulb3.htm','pop','width=400,height=200');
            break;

            case "bulb4":
                  window.open('bulb4.htm','pop','width=400,height=200');
            break;
            }

      switch(document.getElementById(estado).innerHTML) {
```

```
            case "OFF":
                  //enviamos segunda parte de la trama
                  window.open('auto.htm','pop2','width=400,height=200');
                  automatico(bombilla, estado);
            break;

            case "ON":
                  //enviamos segunda parte de la trama
                  window.open('off.htm','pop2','width=400,height=200');
                  apagar(bombilla, estado);
            break;

            case "AUTO":
                  //enviamos segunda parte de la trama
                  window.open('on.htm','pop2','width=400,height=200');
                  encender(bombilla, estado);
            break;
            }
}



function leer_RS232(){

      // lectura RS232 = simA

      //Dato recibido ej. AB 10 01 07 00 00 00 00 00 00 02 CC EE EE

      //RS232=asc2hex(document.simul.simA.value);
      //RS232=document.simul.simA.value;
      mando=document.simul.simA.value.charCodeAt(2).toString(16);
      estado=document.simul.simA.value.charCodeAt(10).toString(16);

      alert(estado);

      switch(mando) {
            case "1":
                  mando="bulb1";
            break;

            case "2":
                  mando="bulb2";
            break;

            case "3":
                  mando="bulb3";
            break;

            case "4":
                  mando="bulb4";
            break;
                  } //fin switch mando

      switch(estado) {
            case "ffd":
                  alert("auto");
                  automatico(mando);

            break;
```

```
            case "1":
                  alert("on");
                  encender(mando);
            break;

            case "3":
                  alert("off");
                  apagar(mando);

            break;

            } //fin switch estado

} // Fin leer_RS232


function inicio(){

      setInterval("leer_RS232()",10000);

}


function init(){

//Mando 1

      <!--(IF NVREG1=0)-->
            automatico("bulb1", "estado1");
      <!--(ENDIF)-->

      <!--(IF NVREG1=1)-->
            encender("bulb1", "estado1");
      <!--(ENDIF)-->

      <!--(IF NVREG1=2)-->
            apagar("bulb1", "estado1");
      <!--(ENDIF)-->


//Mando 2

      <!--(IF NVREG2=0)-->
            automatico("bulb2", "estado2");
      <!--(ENDIF)-->

      <!--(IF NVREG2=1)-->
            encender("bulb2", "estado2");
      <!--(ENDIF)-->

      <!--(IF NVREG2=2)-->
            apagar("bulb2", "estado2");
      <!--(ENDIF)-->

//Mando 3

      <!--(IF NVREG3=0)-->
            automatico("bulb3", "estado3");
      <!--(ENDIF)-->
```

```
	<!--(IF NVREG3=1)-->
		encender("bulb3", "estado3");
	<!--(ENDIF)-->

	<!--(IF NVREG3=2)-->
		apagar("bulb3", "estado3");
	<!--(ENDIF)-->


//Mando 4

	<!--(IF NVREG4=0)-->
		automatico("bulb4", "estado4");
	<!--(ENDIF)-->

	<!--(IF NVREG4=1)-->
		encender("bulb4", "estado4");
	<!--(ENDIF)-->

	<!--(IF NVREG4=2)-->
		apagar("bulb4", "estado4");
	<!--(ENDIF)-->

}


<------ archivo: style.css ------->


.pulsar{
	cursor: hand;
	}

.estado{
	color: black;
	background: #01DF01 ;
	}

.bulb {
	opacity:0.5;filter:alpha(opacity=50);
	}

.mando {
	position: absolute; top: 450px; left: 10%; visibility: visible;
	}

.domo { position: absolute; top: 30px; left: 5%; visibility: visible;
	}


img { border:0px;

    }

body {
	background-color: #E0E0F8;
	font-family: Arial, Helvetica, sans-serif;
	font-size: 12px;
	line-height: 24px;
	color: #336699;
}
```

```
td, th {
      font-family: Arial, Helvetica, sans-serif;
      font-size: 12px;
      line-height: 24px;
      color: "#333333";
      width: 200px;

}

table {
      border: 3px;
      border-color: black;
      border-spacing: 15px;
}

.image{

      border-width:0px;
      }



<------ archivo: error404.html ------->

<html>
<head>
<title>DOMONET ERROR:<!--(READ ID)--></title>
</head>
<body>
      <div>
            <h1>404 Error</h1>
            <h2>P&aacute;gina no encontrada</h2>

      </div>
      <div>
            <p>Esta URL no existe</P>
<a href=index.htm>Volver a la p&aacute;gina de inicio de DOMONET</a>

      </div>
</body></html>



<------ archivo: auto.html ------->

<body onLoad=window.close();>

<!--(EXEC SERIAL_SEND=0x00,0x00,0x00,0x00,0x00,0x00)-->
<!--(EXEC PIN_OUTPUT1=0)-->
Enviando ...

</body>

<------ archivo: on.html ------->

<body onLoad=window.close();>

<!--(EXEC SERIAL_SEND=0x00,0x00,0x00,0x00,0x01,0x00)-->
<!--(EXEC PIN_OUTPUT1=0)-->
Enviando ...
```

```
</body>

<------ archivo: off.html ------->

<body onLoad=window.close();>

<!--(EXEC SERIAL SEND=0x00,0x00,0x00,0x00,0x02,0x00)-->
<!--(EXEC PIN_OUTPUT1=0)-->
Enviando ...

</body>

<------ archivo: pin1_0.html ------->

<!--EXEC KEY=PIN_OUTPUT1=0-->

<------ archivo: pin1_1.html ------->

<!--EXEC KEY=PIN_OUTPUT1=1-->

<------ archivo: bulb1.html ------->

<body onLoad=window.close();>

<!--(EXEC PIN_OUTPUT1=1)-->
<!--(EXEC SERIAL_SEND=0xAB,0x10,0x01,0x07,0x00,0x00)-->
Enviando ...

</body>
<------ archivo: bulb2.html ------->

<body onLoad=window.close();>

<!--(EXEC PIN_OUTPUT1=1)-->
<!--(EXEC SERIAL_SEND=0xAB,0x10,0x02,0x07,0x00,0x00)-->
Enviando ...

</body>

<------ archivo: bulb3.html ------->

<body onLoad=window.close();>

<!--(EXEC PIN_OUTPUT1=1)-->
<!--(EXEC SERIAL_SEND=0xAB,0x10,0x03,0x07,0x00,0x00)-->
Enviando ...

</body>

<------ archivo: bulb4.html ------->

<body onLoad=window.close();>

<!--(EXEC PIN_OUTPUT1=1)-->
<!--(EXEC SERIAL_SEND=0xAB,0x10,0x04,0x07,0x00,0x00)-->
Enviando ...

</body>
```

5.0 Bibliografía

5.0 Bibliografía

Para el desarrollo del presente proyecto ha sido necesaria la documentación de distintas tecnologías de la programación y de las distintas características y capacidades de los componentes electrónicos empleados para su construcción, para su estudio fue necesaria de la siguiente documentación.

Título	Editorial	Aplicación
HTML: The Definitive Guide, 3rd Edition	O'Reilly Media	Desarrollo de la aplicación Web
Javascript: La guía definitiva	Anaya Multimedia	Desarrollo de la aplicación Web
Manual y videotutoriales oficiales de Google sketchup	Google Inc.	Planos de la página EZWEB
Photoshop CS3 superfacil	Editorial Ra-Ma	Tratado de imágenes empleados en el EZWEB
Manual oficial de uso 'EZWEB LYNX'	Lynx	Código en EZWEB
DsPIC – Diseño práctico de aplicaciones	McGranhill	Elaboración de los programas en los microcontroladores PIC
PIC16F690 Datasheet	Microchip	Elaboración de los programas en los microcontroladores PIC
ULN2803 DataSheet	Microchip	Configuración de relé para la gestión de sistema de iluminación
MAX485 DataSheet	Microchip	Configuración del bus RS485
Microcontroladores PIC Diseño práctico de aplicaciones 2ª parte	McGraw Hill	Elaboración de los programas en los microcontroladores PIC

La documentación en donde se detallan las características de los componentes electrónicos necesarios también puede ser obtenida de forma gratuita desde la propia página web oficial del fabricante del componente en cuestión.

Aparte de la documentación citada anteriormente se han consultado las siguientes páginas webs para hacer un estudio de las alternativas en el mercado respecto al producto y tecnologías empleadas:

Web	Fecha de acceso	Aplicación
Ethernut Project http://www.ethernut.de	10 diciembre 2013	Búsqueda de alternativas para el microservidor EzWeb
Twitter Bootstrap http:www.getbootstrap.com	8 octubre 2013	Estudio para alternativas para desarrollo de vistas en HTML5
My N3rd http://www.myn3rd.com/	5 Enero	Estudio de alternativas en el mercado al sistema propuesto
Home Automation Systems - HomeSeer http://www.homeseer.com/	5 Enero 2014	Estudio de alternativas en el mercado al sistema propuesto