SISTEMA INALÁMBRICO DE CONTROL DE ILUMINACIÓN

Memòria del Treball Final de Carrera
d'Enginyeria Tècnica de Telecomunicació,
especialitat Sistemes Electrònics

realitzat per:

Jaime Vázquez Brazo.

i dirigit per

Juan Carlos Moure López.

Bellaterra, 14 de Juny de 2007

ÍNDICE

1. INTRODUCCIÓN

Hoy en día la tecnología con la que diariamente convivimos hace que la vida de las personas sea cada vez más cómoda, pero el avance tecnológico no siempre está enfocado al bien común, sino a la comodidad y el beneficio a corto plazo. Esto es en muchos casos un error, ya que tenemos los recursos necesarios para poder contribuir al bienestar de lo que nos rodea.

Es importante controlar el consumo diario de muchos de los componentes habituales en nuestras vidas, ya que cada persona puede consumir mucha más energía eléctrica de la que necesita. Este gasto adicional repercute tanto en la economía de cada persona como en algo más importante aún, en el sobrecalentamiento global del planeta. Esto provoca una producción de energía superior a la que realmente es necesaria.

Una de los avances en el campo de la habitabilidad domestica está representada por la "Domótica". Ésta nos introduce en el uso de la tecnología dentro de nuestras viviendas, intentando conseguir la casa del futuro, un hogar donde todo esté automatizado. El concepto de domótica consiste en el conjunto de sistemas que permiten automatizar una vivienda, aportando un control energético, un sistema de seguridad y una gestión de comunicaciones. Estos sistemas están compuestos normalmente por redes de comunicaciones cableadas, aunque la tendencia es a diseñar sistemas inalámbricos, siendo ésta una alternativa sencilla e interesante para la integración de estos sistemas en un hogar.

Una de las aplicaciones de la domótica es el control de iluminación, permitiendo disminuir el consumo diario de manera considerable. Estos pequeños cambios que nos aporta la domótica son muy importantes, ya que pueden ser enfocados para un bien común. Si lo miramos desde el punto de vista del consumo excesivo que ocasionan los electrodomésticos en general, también es muy interesante la necesidad de adaptar estos recursos al ahorro y adaptación al medio ambiente, ya que es uno de los grandes problemas que sufrimos hoy en día.

A pesar de las ventajas que nos aporta la domótica, normalmente tiene dificultades prácticas, ya sea por el cableado de una instalación o el coste que conlleva. Mucha gente lo ve como algo lejano y futurista, ya que el alto coste y la falta de integración en futuros hogares en construcción no hacen pensar que sea posible en un presente inmediato. No obstante, hay que tener en cuenta que la rápida integración de este tipo de tecnología depende de la adaptación a las necesidades básicas de cada persona y no de una incursión global. Es por esto que el avance en la automatización de hogares será paulatino, pero llegará a ser una realidad.

OBJETIVOS

El trabajo que se presenta en este proyecto tiene el objetivo de estudiar el control de iluminación automático e inalámbrico de cualquier habitación, dentro de una vivienda. Estudiaremos las posibilidades que tiene un modelo de placa creada por Motorola, en la que nos centraremos para conseguir un diseño completo, tanto en comunicación como gestión de un sistema de control de iluminación.

A continuación se desglosarán los objetivos más detallados haciendo referencia a los apartados de la memoria que correspondan.

El consumo de todos los componentes eléctricos de un hogar es un problema a tratar, como la utilización ajustada de algo tan necesario e imprescindible como es la iluminación de una habitación. Aún siendo nuestro mayor objetivo la reducción del consumo de iluminación, se tratan ideas como la inactividad de los dispositivos del sistema de control cuando éstos no deben funcionar, la transmisión de datos sólo cuando sea necesario y el ajuste de potencias tanto de actuación como de transmisión de datos. Estos puntos serán tratados en el apartado "***Diseño de los componentes inalámbricos***", con el fin de obtener un sistema capaz de conseguir una reducción de consumo, tanto eléctrico como del propio dispositivo encargado del control.

La detección de presencia es necesaria para la realización de un sistema con estas características. Mediante un sistema de detección de personas, podemos controlar la automatización de iluminación cuando sea necesario. Para poder tener un sistema con características de detección se hará un estudio de los dispositivos que nos proporcionan esta posibilidad, tratando varios de estos tipos de elementos de detección y haciendo una propuesta para la realización de un prototipo en el apartado de "***Detección de presencia***".

El control de una línea eléctrica convencional mediante un dispositivo de bajo consumo conlleva la instalación de componentes de aislamiento eléctrico. Para este fin se utilizará un componente capaz de actuar sobre la línea eléctrica mediante un controlador de bajo consumo. Uno de los apartados de la memoria estará centrado en este componente, haciendo un estudio de las características y posibilidades de integración en el sistema en el apartado "***Elemento actuador sobre el terminal de luz***".

Los datos proporcionados por el detector deben ser interpretados para poder actuar sobre una línea eléctrica. Esta información puede ser procesada para poder actuar sobre la luz mediante los dos dispositivos de los que se compone el sistema. Uno de estos dispositivos es el encargado de la detección, mientras que el otro debe actuar sobre la luz en función de la detección. El funcionamiento básico del sistema es el de encender la luz si se detecta presencia de una persona y de apagarla si deja de detectarla. Hay que destacar que la decisión de mantener encendida o apagada la luz recae totalmente sobre el sistema, siendo mucho más práctico y despreocupado para cualquier usuario que entre dentro del radio de control del sistema. Uno de los objetivos es estudiar en el apartado "***Diseño de los componentes inalámbricos***" las dos placas de Motorola que se encargaran tanto de la adquisición de datos como del transporte de la información con la que se tomarán las decisiones, por lo que cada una de las placas tendrá un diseño de software específico y definido para su función.

La utilización de un sistema inalámbrico nos facilita la elección de ubicación de dispositivos, pero conlleva mayor dificultad en la gestión de datos entre ambos. El sistema se compone de dos dispositivos en los que el intercambio de información

entre ellos se produce mediante una comunicación inalámbrica, aumentando la versatilidad y la fácil adaptación a cualquier ubicación. Estos dos dispositivos son en realidad dos controladores, los cuales tienen una función y ubicación muy diferentes, de aquí la importancia de un sistema de comunicación inalámbrica, el cual nos proporciona una libertad de movimiento que no es posible con otros equipos. El sistema de comunicación inalámbrica que se utilizará se tratará haciendo un estudio del diseño realizado en función de las limitaciones que conlleva el utilizar este tipo de intercambio de datos, tratándolo en el apartado "***Diseño de la comunicación entre dispositivos***".

La versatilidad y el consumo son primordiales en el dispositivo de detección, ya que la ubicación de éste no está definida, depende del habitáculo donde queramos instalar el sistema. Esta versatilidad no debe estar ensombrecida por la necesidad de conexión a una toma de corriente, ya que esto reduciría su mayor característica, la movilidad. Para evitar este problema, el dispositivo debe estar equipado con un sistema de alimentación que proporcione una autonomía de larga duración, obteniendo así la libertad necesaria para un sistema con los objetivos mencionados. Este problema será tratado en el apartado "***Diseño de funcionamiento***" y "***Ayuda para un bajo consumo***", en el que se hará un diseño y estudio para el ahorro de consumo en el controlador del dispositivo detector.

El dispositivo de actuación carece de tal autonomía, ya que su diseño y ubicación natural proporciona la corriente que necesita. El actuador debe estar instalado en sitio donde pueda controlar la iluminación, teniendo acceso a corriente constante, por lo que la carencia de un sistema de alimentación autónomo es irrelevante. Podemos decir que otro de los objetivos que se pretende conseguir es el de dotar de una gran autonomía a uno de los dispositivos, sin descuidar el bajo consumo del otro.

Cada uno de los dispositivos está compuesto de varios componentes. El dispositivo detector, está compuesto por una placa y un sensor de infrarrojos, el cual es capaz de obtener información al detectar cambios de temperatura debido a un cuerpo que desprende calor, pudiendo saber si hay presencia. El dispositivo actuador está compuesto de un controlador igual al del dispositivo detector, pero éste está

acompañado de un actuador, el cual nos proporciona la posibilidad de actuar sobre la corriente eléctrica a 220v mediante información digital que nos da el controlador, que sólo es capaz de trabajar entre 0 y 5v. Ambos dispositivos serán estudiados en conjunto, proponiendo un prototipo final, el cual tendrá una serie de características de utilización que serán explicadas en el apartado "***Manual del prototipo***", para poder tener una idea más clara del producto final que se pretende diseñar.

Los sistemas de control automático son caros, en muchos casos inaccesibles para la gente. El apartado "***Costes de fabricación***" tratará los costes que tiene la realización de un prototipo, para poder realizar un estudio de la relación que hay con el ahorro eléctrico que nos proporciona el sistema. El conjunto de todos los componentes se pretende que sea simple pero eficaz, siendo éste un objetivo primordial para que el sistema sea accesible a cualquier persona, tanto técnica como económicamente.

En conclusión, el último objetivo que se pretende alcanzar con este Proyecto es la propia formación del autor. El desarrollo del software ha tenido una gran dificultad, pero quedan compensados con los conocimientos prácticos adquiridos durante todo este tiempo.

2. DISPOSITIVO DE CONTROL Y COMUNICACIÓN

2.1. DEFINICIÓN.

Para empezar a hablar sobre los dispositivos que se han utilizado en este proyecto, tenemos que separar los distintos componentes que la forman, para esto, empezaremos explicando los "*Dispositivos de control y comunicación*", que son en realidad, las dos placas de **Motorola** que disponemos para realizar el proyecto.

Es importante decir que, hay muchos tipos de controladores en el mercado, unos más potentes que otro, unos tienen características técnicas superiores a otros, pero las placas utilizadas en este proyecto se ajustan perfectamente a las necesidades requeridas. Hay que tener en cuenta que la función que tendrán que realizar no será excesivamente exigente para los microcontroladores de las placas, ya que lo que se busca es un funcionamiento estable y eficaz, pero sí que se intentará sacar el máximo provecho a las cualidades de ahorro de consumo de las que disponen, ya que uno de los objetivos que se pretende es hacer que estos componentes tengan una gran autonomía.

Como ya hemos comentado, disponemos de dos placas (controladores), ambas son las encargadas de gestionar el sistema. El modelo de controlador en concreto es el "***MC09S8GT60***", este controlador tiene la posibilidad de comunicarse con otra placa gemela a través de un sistema de comunicaciones basado en "***Zigbee***". Mediante los dos controladores de los que disponemos, podemos controlar dos puntos distantes de forma inalámbrica, intercambiando información de uno a otro mientras que cada controlador por si sólo toma decisiones respecto a la función para la cual ha sido encomendado.

Los objetivos de cada placa son muy distintos entre cada una de ellas, ya que una tiene el objetivo de controlar la presencia y la otra la de actuar sobre la luz, ambas ayudándose de componentes externos. Los componentes externos serán tratados en el capítulo ***Detección de presencia*** y en el de ***Elemento actuador sobre terminal de luz***.

A continuación se mostrarán las características de la placa y partes que la componen, viendo así la estructuración de las placas antes de empezar a programar las diferentes funciones que definirán al controlador hacia una forma de actuar u otra.

2.2. ESPECIFICACIONES.

En este apartado se procederá a dar una explicación más técnica y concreta de las partes y funcionamiento de las placas "***MC9S08GT60***" (Figura 1). Como ya he comentado antes, las placas que se utilizarán son exactamente iguales, por lo que las características técnicas que se expondrán a continuación son exactamente iguales para las dos placas, dejando a parte el funcionamiento específico que se le quiera dar a cada una.

Para empezar, describiremos cada una de las partes de las que se compone el controlador, haciendo un repaso de los puntos generales que se puedan destacar.

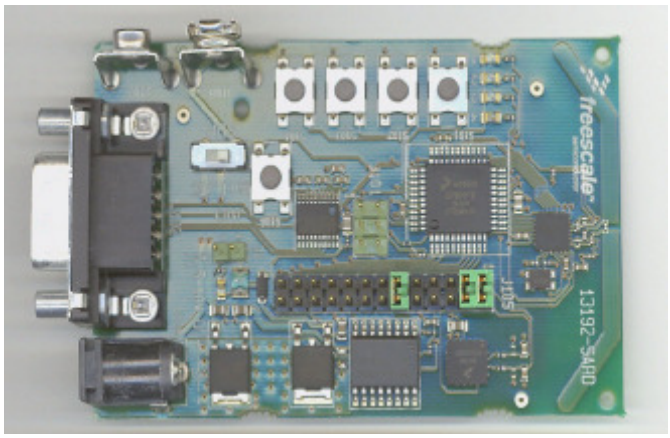


Figura 1. Placa *MC9S08GT60 SARD.*

El *sistema de alimentación* que dispone puede ser de dos clases. Podemos alimentar la placa mediante una entrada de 9v, gracias a una ***Entrada de alimentación*** mediante un transformador, pudiendo tener una alimentación continua pero limitando la movilidad de la placa. Otra manera de la que podemos disponer para alimentar la placa es mediante una ***pila*** *o* ***Batería,*** la cual nos

proporcionará la alimentación necesaria durante un tiempo limitado, pero de esta manera se puede conseguir una movilidad absoluta con un funcionamiento completo, ya que por usar este modo de alimentación no perdemos ningún tipo de funcionalidad.

Este modelo dispone también de un ***Puerto serie RS-232***, para poder conectar diferentes periféricos, aunque en este proyecto no es necesario, por lo que el espacio que ocupa nos lo podemos evitar en un posible prototipo definitivo.

El modelo ***"MC9S08GT60"*** incorpora una serie de acelerómetros, compuestos por dos microcontroladores que controlan la aceleración del movimiento efectuado por la placa. Para este tipo de cálculo utiliza dos modelos de controladores, el "***MMA6261QR***" y el "***MMA1260D***", los cuales controlan los ejes de coordenadas *XY* y *Z* respectivamente. La utilidad que se le pueda dar a estos controladores depende mucho de su funcionamiento, ya que estos no controlan la orientación de la placa, sino la aceleración de su movimiento hacia cualquier lado sin dar importancia a la posición inicial de la placa antes de moverse.

Como la gran mayoría de las placas diseñadas por ***Motorola/Freescale***, estas placas disponen de una serie de pulsadores, uno de los cuales es el de reset, pero a parte de éste, la placa dispone de cuatro pulsadores más, los cuales comparten el Puerto D con cuatro *Leds*, haciendo un total de ocho componentes dependientes de un puerto, con lo que tenemos una serie de 8 bits destinados al control de cada uno de los componentes. Los pulsadores son controlados por *PTD2, PTD3, PTD4 y PTD5*, mientras que los *Leds* son controlados por el *PTD0, PTD1, PTD6* y *PTD7* todos dispuestos en el registro *PTDD* [1].

El resto de puertos de los que se compone la placa (Tabla 1) están dispuestos para su utilización, tanto para E/S como para algún otro tipo de finalidad como, entradas de reloj externas o entradas para hacer una conversión A/D. Estos puertos van desde el PORTA hasta el PORTG.

Port Pins	Alternate Function	Reference[1]
PTA7–PTA0	KBI1P7–KBI1P0	Keyboard Interrupt (KBI) Module
PTB7–PTB0	AD1P7–AD1P0	Analog-to-Digital Converter (ATD) Module
PTC7–PTC4	—	Parallel Input/Output
PTC3–PTC2	SCL1–SDA1	Inter-Integrated Circuit (IIC) Module
PTC1–PTC0	RxD2–TxD2	Serial Communications Interface (SCI) Module
PTD7–PTD3	TPM2CH4–TPM2CH0	Timer/PWM (TPM) Module
PTD2–PTD0	TPM1CH2–TPM1CH0	Timer/PWM (TPM) Module
PTE7–PTE6	—	Parallel Input/Output
PTE5 PTE4 PTE3 PTE2	SPSCK1 MISO1 MOSI1 $\overline{SS1}$	Serial Peripheral Interface (SPI) Module
PTE1–PTE0	RxD1–TxD1	Serial Communications Interface (SCI) Module
PTF7–PTF0	—	Parallel Input/Output
PTG7–PTG3	—	Parallel Input/Output
PTG2–PTG1	EXTAL–XTAL	Internal Clock Generator (ICG) Module
PTG0	BKGD/MS	Development Support

Tabla 1. Puertos HCS08.

Para realizar un estudio más definido sobre la placa debemos separar el funcionamiento interno en dos partes muy distintas, una es sobre el microcontrolador "***HCS8***" y la otra sobre el "***transceptor integrado MC13192***".
El Microcontrolador "*HCS08*" es el encargado de controlar toda la gestión de datos y toma de decisiones de los que se compone el programa, a más de dirigir el resto de componentes incluidos en la placa para asegurar un funcionamiento sincronizado en conjunto. La otra parte que hay que tener en cuenta es la del *transceptor integrado "MC13192"*, el cual nos proporcionará el control de la comunicación inalámbrica. Éste está vinculado al "*HCS08*" pero tiene un funcionamiento autónomo, el cual sólo será quebrado por las peticiones del núcleo "*S08".*

El diagrama de bloques de ambas placas "***MC09S8GT60 SARD***" (Figura 2) nos proporciona la idea general del funcionamiento modular que se propone para estos modelos fabricados por *Freescale*, viendo también la estructuración de dependencias que tienen algunos componentes mencionados anteriormente. Esto se puede definir básicamente como la representación de cada una de las funcionalidades que tienen las placas "*MC09S8GT60 SARD*" y la dependencia de cada una de ellas, pudiendo ver como componentes más destacables el "*HCS08*"

como núcleo del bloque, los acelerómetros y el microcontrolador *"MC13192"* encargado de comunicación [2].

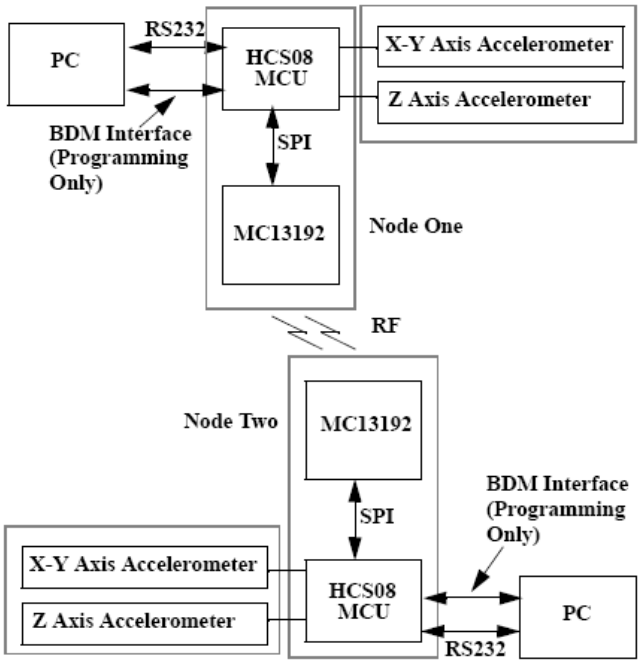


Figura 2. Estructura del MC09S8GT60 SARD.

Respecto a la comunicación entre componentes, debemos ver que hay diferentes modos de actuación sobre el microcontrolador *"HCS08"*, una de ellas es la conexión con un *PC* compatible con ***"Code Warrior"*** para la programación del sistema mediante la conexión estándar **RS-232** o **USB**, mientras que la dependencia que hay del *"MC13192"* respecto al *"HCS08"* se produce mediante una conexión serie *SPI*. Los Acelerómetros tienen una conexión directa con el **HCS08** [3].

2.2.1. CARACTERÍSTICAS HCS08.

El microcontrolador basado en el *"HCS08"* es uno de las que más rápido está creciendo dentro de la familia de los *"HC08"* creados por **Motorola/Freescale**. Estos contienen una tecnología avanzada en el campo de bajo consumo, basándose en microcontroladores de 8 bits, siendo estos en algunos casos tan eficaces como los *"HCS12"*, los cuales trabajaban a 16 bits. Esta característica ha sido muy

importante para poder realizar y llegar a uno de los objetivos del proyecto, ya que una de las cualidades que se desean es la de un bajo consumo con buen rendimiento.

La posibilidad de programar la frecuencia a la que trabajan estos microcontroladores nos proporciona la ventaja de programar el *µC S08* para que trabaje ajustándose tanto a los requisitos de rendimiento como a los de consumo. El microcontrolador HCS08 puede trabajar desde los 8MHz hasta los 40MHz, gracias a la función **FLL** ***(****Frequency Locked Loop****)*** la cual es capaz de coger tanto tiempos de reloj internos como externos, pudiendo multiplicarlos como máximo hasta "x4". El resonador o cristal del que disponemos para generar la señal de reloj, sólo puede generarnos una frecuencia de 32kHz-100kHz o de 2MHz-10MHz, pero siempre contando que la máxima velocidad del Bus podrá ser de 20MHz [4]. Gracias a esta polivalencia, podemos seleccionar la velocidad de procesado con la que queremos que trabaje nuestro sistema y así obtener diferente rendimiento según nuestras necesidades.

- Un ciclo de 50ns (20MHz) nos supone un consumo a 2.1v.
- Un ciclo de 125ns (8MHz) nos supone un consumo a 1.8v.

Podemos ver que la reducción de la velocidad de procesado nos proporciona un ahorro del consumo muy notable, mientras que el aumento de ciclos por segundo nos dará un gran rendimiento, pero a costa del mayor consumo.

La empresa *Freescale* ha querido poner mayor hincapié en la reducción de consumo con estos modelos de µC S08, creando un sistema interno de puesta en bajo consumo que puede ser configurado por el programador según convenga. Este tipo de características ya las tenían otros productos similares, pero en esta generación se han integrado mayores cualidades relacionadas con este tema. Las posibilidades del *"HCS08"* de pasar a un modo de bajo consumo son cuatro, que se pueden clasificar en dos modos, el *STOP* y el *WAIT*. El modo *STOP* se compone de 3 posibles opciones, es una forma de pasar a un estado de bajo consumo o *Standby* con el que conseguimos una reducción de consumo considerable, pero que tiene una serie de requisitos que hay que tener en cuenta, ya que inhabilitamos algunas

funcionalidades del procesador y reducimos las formas de poder despertarlo. Por otro lado tenemos el *WAIT*, que es el modo que pueden utilizar otros modelos de microcontroladores, tanto de la familia *"HC08"* como *"HC12"*. Este modo de *WAIT* tiene la misma finalidad que la de *STOP*, la de reducir procesos para un bajo consumo y paso a modo espera, pero no deshabilitamos ninguna de las funciones importantes, con lo que el consumo siempre será mayor que cualquiera de los modos *STOP*. Estos modos se darán con mayor profundidad en los apartados el ***Diseño de funcionamiento***.

Las características principales en los microcontroladores de la familia *S08* son variables según el modelo, tanto en *memoria flash*, como memoria *RAM*, aunque la mayor diferencia la podemos ver en los tipos de temporizadores que se pueden utilizar simultáneamente o los tipos de comunicaciones con periféricos ext/int de los que dispone. Todas las posibilidades del microcontrolador que se utilizará no son del todo importantes por lo que a continuación nombraré las diferentes características del modelo incorporado en la placa *"MC09S8GT60 SARD"*:

- Memoria Flash de 60KBytes.
- Memoria RAM de 4KBytes.
- 32 pins de entrada/salida.
- 8 pins de los 32 para poder conectar un teclado controlado por KBI.
- 2 Timers de 16 bits.
- Rango de trabajo en las entradas de 1.8v a 3.6v.
- 2 módulos SCI (Serial Comunications Interface).
- 1 módulo SPI (Serial Peripherial Interface).
- 4 modos de bajo consumo.
- Auto wake up timer.

2.2.2. CARACTERÍSTICAS MC13192.

Freescale ofrece una variedad enorme de componentes que trabajan conjuntamente con microcontroladores de la familia "*HC08"* y "*HC12"*. En este caso nos centramos en un componente que tiene la finalidad de conseguir comunicarse mediante radio frecuencia con otros componentes de igual categoría, el microcontrolador "*MC13192"*. Este componente normalmente llamado transceptor, nos ofrece la posibilidad de trabajar cumpliendo con la norma IEEE 802.15.4, y funcionando en la gama de frecuencias de 2.4GHz.
A este apartado se le dará mayor importancia que al apartado del μC HC08, ya que este microcontrolador es básico para conseguir una movilidad inalámbrica de los dispositivos y tiene características propias en frente de las de microcontroladores como los de la familia "*HC08"*, que son más conocidas.

Las características principales del ***transceptor MC13192 RF*** son:

- 16 canales en banda de 2,4 GHz.
- Amplias funciones de transmisión para datos en paquetes.
- Diseño económico CMOS.
- Pocos componentes externos para reducir costes y complicaciones.
- Sensibilidad Rx de -92 dBm a 1% PER.
- Salida de corriente programable de 0 dB (típica) en un margen de 20 dB.
- Modos de bajo consumo para prolongar la vida de las baterías.
- Tensión de entrada de 2,0 V a 3,4 V con regulador en el chip.
- 4 comparadores internos de reloj para consumir menos recursos de la MCU.
- Siete E/S de uso general (GPIO).
- Interfaz de datos SPI con la MCU.
- Salida de reloj programable para la MCU.
- Amplio margen de temperaturas: de -40ºC a +85ºC.
- Encapsulado QFN-32 de 5 mm x 5 mm sin plomo.

A continuación se nombrarán y se explicarán varias de las características más importantes que tiene el Microcontrolador "*MC13192"*, para poder tener una idea del funcionamiento en el que se basa.

El *MC13192 RF* se conecta al *MCU HCS08* mediante una interfaz en serie *SPI* y constituye una solución económica y de bajo consumo para una gran variedad de aplicaciones de transmisión de datos a baja velocidad cumpliendo con la normativa IEEE 802.15.4 [5]. La finalidad del protocolo en el que se basa el *transceptor MC13192 RF* es la de supervisar a distancia los datos a baja velocidad, los cuales por su característico rendimiento requieren un bajo consumo. Esto constituye el entorno en el que nos basaremos para programar el sistema proyectado.

Las distintas interrupciones que puede generar el *transceptor MC13192 RF* pueden ser debidas a múltiples actividades del propio microcontrolador, como de sucesos ocurridos por la comunicación de datos. Estas interrupciones son dirigidas hacia el μC HCS08 mediante la *interfaz SPI* pudiendo avisar al sistema de cualquier acontecimiento ocurrido referentes a las actividades externas a la placa. Las interrupciones que van sucediendo entre los dos microcontroladores pueden ser en los dos sentidos, ya que tanto uno como el otro deben intercambiar los datos mediante la interfaz SPI. Podemos ver la estructuración interna de ambos controladores (Figura 3), donde podemos ver que el único camino de la comunicación entre ambos dispositivos es la interfaz *SPI* [6].

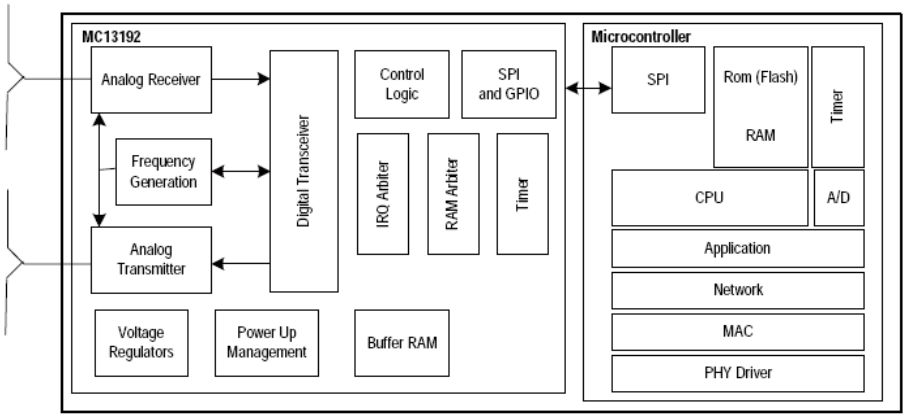


Figura 3. Estructura interna de MC13192 y HCS08.

La frecuencia a la que trabaja el *"MC13192"* es sobre los 2.4GHz, más concretamente entre los 2.405GHz y los 2.48GHz, se puede configurar su rango de trabajo con 16 canales al rededor de esta frecuencia, siendo capaz de distinguir los

diferentes canales con una diferencia de 5MHz. Esta capacidad es muy útil para poder trabajar con varios sistemas que utilicen el estándar 802.15.4 sin que se molesten unos a otros, sólo cambiando el canal en el que van a trabajar. La elección del canal es muy importante, ya que una mala elección coincidiendo con el mismo canal de otro sistema cercano puede hacer que el *"MC13192"* esté detectando transmisión de información ajena al interés de nuestro sistema. Este tipo de interferencias, si le podemos llamar así, no nos deben afectar al funcionamiento global de las funciones para las que se ha diseñado el sistema, ya que en la emisión y recepción de paquetes se debe tener en cuenta mucha más cosas que no solo el canal que se está utilizando, como la verificación de datos e identificación de cada placa, pero estas interferencias si que nos afectarán en el trabajo que realice el *"MC13192 RF"*, ya que éste no discrimina la información por el contenido, sino por la frecuencia a la que le llega. Podemos añadir que toda la información que reciba el transceptor afectará al μC HCS08, ya que sin ningún tipo de verificación previa por parte del *"MC13192"*, almacenará los paquetes y generará la debida interrupción despertando o cambiando el estado del "*HCS08"*.

Este microcontrolador soporta *"Payloads"* de hasta 125Bytes, siendo un sistema capaz de intercambiar mucha información en un mismo paquete de envío. La estructuración de estos paquetes se divide en cuatro secciones (Figura 4), en primer lugar nos encontramos con 4 bits llamados preámbulos, a continuación hay una ristra de 8 bits llamado **SFD** ***(****Start frame delimiter****)*** el cual nos delimita la información necesaria para el receptor, la longitud total la trama que se envia tiene un tamaño de 1byte y está situada entre el **SFD** y los datos de información útil que puede ser de las dimensiones máximas antes mencionadas de 125Bytes, una vez tenemos los datos, para finalizar el paquete se calcula un **FCS** ***(****Frame check sequence****)*** el cual nos verificará, en la recepción, los datos que los valores de SFD y del preámbulo han fijado [7].

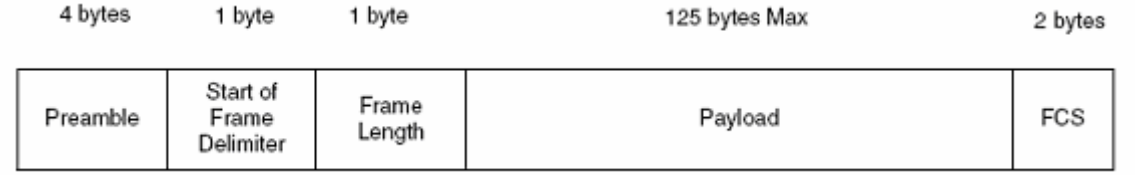


Figura 4. Estructura paquetes.

El protocolo de comunicación que utiliza el *"MC13192"* se centra en las capas inferiores de red, en concreto la capa Física y la capa MAC. La señal que genera este transceptor es del tipo **O-QPSK**, con lo que conseguimos un mínimo desplazamiento de fase [8]. La velocidad de transmisión puede llegar a ser de unos 250Kbps, esta capacidad es suficiente para el intercambio de información de gestión, pero es muy difícil hacer mediante este sistema de comunicación, una utilización semejante a otros sistemas inalámbricos, como por ejemplo los basados en el sistema ***Bluetooth***.

La potencia de salida que puede proporcionar el *"transceptor MC13192 RF"* es de entre -16.6 dBm y 3.6 dBm. Estos niveles de transmisión son configurables por software, pudiendo configurar desde su mínima potencia hasta la máxima en una escala del 0 al 11. Respecto a niveles en recepción que es capaz de detectar, el mínimo nivel de señal que es capaz de captar es de -92 dBm con sensibilidad en la recepción de paquetes de 1% ***(PER)***, muy por encima de las especificadas en el estándar. Con estos niveles en los que podemos trabajar, el consumo puede ser ajustado a las necesidades, teniendo en cuenta principalmente las distancias, ya que según especificaciones, la distancia que podemos albergar es de 10 a 20 metros, dependiendo de los diferentes obstáculos con los que puede encontrarse la señal.

Todo dispositivo que deba generar o recibir una señal RF debe incorporar una antena. En el caso del *"MC13192"* incorpora dos antenas compuesta por una pista integrada en la placa (Figura 5). Cada una de las antenas tiene su función, una para la transmisión y otra para la recepción, ya que el microcontrolador debe controlar por separado cada puerto y cada estado.

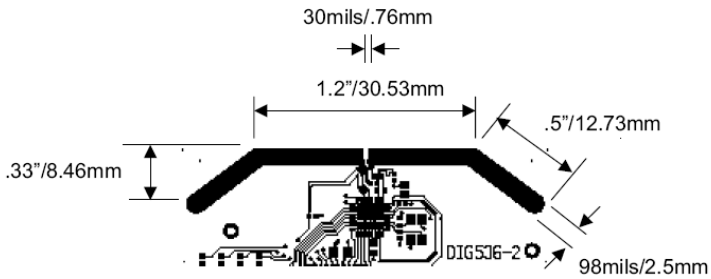


Figura 5. Integración de antenas.

El *transceptor MC13192*, al igual que el *µC HCS08*, tiene propiedades de bajo consumo. Esta reducción no es referente a la potencia de transmisión elegida, es mediante la reducción de trabajo del microcontrolador en sí, ya que tenemos varias opciones de poder ponerlo en estados de hibernación o espera. Los diferentes estados en los que se puede configurar son:

- **Estado Idle:** Este es el estado de funcionamiento normal, teniendo total funcionalidad del Microcontrolador.
- **Estado Off:** Todas las funciones del "*MC13192"* quedarán desconectadas.
- **Estado Hivernate:** Este es el estado en el que se consume menos recursos, detrás del estado Off. Sólo podemos salir mediante una actuación mediante una señal externa digital sobre la entrada *ATTN* del *µC MC13192*. El interfaz SPI dirección "*HCS08"* queda deshabilitado además de los sistemas de reloj del Microcontrolador.
- **Estado Doze:** Este estado está un nivel por debajo del *Hivernate*, en cuanto a consumo. En este contamos con los sistemas de reloj para poder generar un *TIMEOUT*, el cual nos vuelva al estado *Idle*. Otra manera de despertar el "*MC13192"* es la de la entrada *ATTN*, pero seguiremos teniendo la interfaz *SPI* deshabilitada.
- **Estado Receiver:** Tendremos total funcionalidad, pero sólo podremos estar en modo de recepción.
- **Estado Transmit:** Tendremos total funcionalidad, pero sólo podremos estar en modo de transmisión.

Cada uno de ellos conlleva un nivel de consumo y una serie de características a tener en cuenta para poder cambiar el estado del Microcontrolador. Para la utilización de estos estados, además de que nos deje pasar al estado de funcionamiento normal, hay que tener en cuenta el tiempo (Tabla 2) que necesitan para pasar de un estado de espera al de funcionamiento completo.

Mode	Transition Time To or From Idle
Off	23.332 ms to Idle
Hibernate	18.332 ms to Idle
Doze	332 µs to Idle
Idle	
Receive	144 µs from Idle
Transmit	144 µs from Idle

Tabla 2. Tiempos de cambio.

El *"transceptor MC13192 RF"* es un componente imprescindible para conseguir una comunicación inalámbrica, dándole mucha importancia sobre todo, a su función sobre las placas *"MC9S08GT60 SARD"* y a sus características de bajo consumo, las cuales son las que aportan un mayor apoyo a uno de los objetivos que nos hemos propuesto en el proyecto.

2.3. METODO DE IMPLEMENTACIÓN DEL SOFTWARE.

Hace unos años, el sistema más utilizado para poder programar una serie de controladores de manera más o menos eficiente, era necesaria una programación a muy bajo nivel, en Ensamblador, ya que era la única manera de que los programas hicieran exactamente lo que deseaba el programador. Toda la serie de compiladores para programar de manera más sencilla y rápida no eran efectivos, ya que estos no conseguían traducir de un código a otro de una manera eficaz, haciendo perder mucho tiempo en intentar solucionar lo que debería haber hecho el compilador.

Hoy en día, la programación de estos microcontroladores es mucho más sencilla, ya que los compiladores han evolucionado enormemente, y son capaces de traducir de un código a otro con una enorme efectividad, incluso haciendo un código mejor del que un programador realizaría a bajo nivel para el mismo programa.

El lenguaje de programación con el que se puede generar el código del proyecto que se ha realizado podía haber sido "*ensamblador*", pero como he dicho antes, es un código en el que hay que tener muchísimas cosas en cuenta y realmente no es un código fácil de estructurar y depurar, por lo que se ha realizado con el código en "*Lenguaje C*". Este código es mucho más intuitivo, ya que trabajamos con un nivel más alto de programación, con lo que conseguimos despreocuparnos un poco de los detalles de cada instrucción y lo que conllevan, centrándonos en la estructuración general y condiciones de actuación del código.

La elección de la utilización del "*Lenguaje C*" ha sido por la gran cantidad de programas realizados para placas similares, estos están programados en este código y puede ser muy útil para poder utilizar conceptos descritos en los programas y así poder usarlos o mejorarlos en el nuestro. Es lógico que la programación del código generado para el proyecto no se haya realizado desde cero, ya que lo programado para estas placas mediante código existente ha sido muy útil y necesario para poder conseguir en un tiempo relativamente rápido un funcionamiento correcto del sistema. Este concepto de reutilización es necesario, ya que la evolución está en la innovación y para ello es necesario no perder tiempo en reconstruir código que ya existe.

2.3.1. INTEGRACIÓN DEL SOFTWARE.

Como antes se ha comentado, la programación que se ha utilizado es la del "*Lenguaje C*", por su fácil modo de empleo y código intuitivo, pero para poder realizar esta programación es necesario también un programa que pueda entender el código generado, siendo capaz, tanto de poder encontrar fallos en nuestro código como de compilarlo a un nivel inferior, el cual pueda ser entendido directamente por el Microcontrolador que se a utilizar.

Los programas que son capaces de realizar estos trabajos son múltiples, pero la casa *Freescale* de *Motorola* nos aconseja la utilización del programa "***CodeWarrior™ Development Studio***" como programa base para toda la programación de sus microcontroladores. Realmente es uno de los programas más

potentes en este campo, por lo que al hacernos con las placas de cualquier modelo de *Freescale* nos vendrá una versión de este programa.

La utilización del "***Code Warrior"*** es importante, no por sus cualidades como programa intuitivo y completo, sino también debido a que al venir conjuntamente con las placas que vamos a utilizar, incorpora una serie de *Drivers* muy importantes para que el propio programa se adapte a las peculiaridades de compilación del microcontrolador que se desea programar, dejando de lado la necesidad de que el usuario o programador tenga que tener en cuenta esas peculiaridades, evitando así una perdida de tiempo en encontrar un programa que compile los códigos para nuestra placa eficientemente.

El ***"CodeWarrior™ Development Studio"*** puede ser configurado para compilar y cargar el código deseado en el microcontrolador mediante diferentes tipos de conexionado entre el *PC* y la *Placa*. El método utilizado en el proyecto es el del "***Multilink USB***" (Figura 6), el cual nos permite tener acceso al **BDM *(Background Debug Mode)*** del microcontrolador. Conectado al PC, puede procesar la señal que mandamos y generar una serie de impulsos hacia la placa mediante un bus de 6 cables. Podemos decir que "***CodeWarrior™ Development Studio***" nos proporciona un control del "*Multilink USB*" para poder sincronizar el envió de datos por el puerto USB hacia la placa. Con este sistema conseguimos una comunicación del bus que lo une con el Microcontrolador de unos 35MHz, dependiendo del microcontrolador con el que nos queremos comunicar. Este sistema compuesto por el "*Code Warrior*" y el "*Multilink USB*" nos proporciona además un control de los pasos del programa, pudiendo tener un control directo del proceso de ejecución del Microcontrolador y sus pines, *leer/escribir* registros y valores de memoria, además de detectar los errores del código en el procesador.

Figura 6. Multilink USB.

El otro sistema de programación del microcontrolador es mediante la interfaz de comunicación serie *"RS-232"*, pero este sistema está un tanto obsoleto, ya que actualmente la mayoría de ordenadores y más aun los portátiles, no poseen este tipo de conexión, por lo que es un método que persiste por la gran utilización que ha tenido durante años, pero que con la aparición del sistema *USB* a ido perdiendo adeptos y se ha quedado atrás. Como se ha comentado antes este conector ocupa mucho espacio en la placa, por lo que personalmente diseñaría una placa sin este conector, con el fin de reducir las dimensiones de la placa, siempre con el punto de mira en el prototipo que se pretende hacer.

2.3.2. SISTEMA DE COMUNICACIÓN.

Las placas *"MC9S08GT60 SARD"* tienen muchas posibilidades de comunicación inalámbrica, pudiendo utilizar una comunicación ***Zigbee*** con muchas posibilidades y un sistema mucho más simple, pero basada en la misma norma IEEE 802.15.4, el *Simple Media Access Control* ***(SMAC)***, basado en *Zigbee* [9].

El *"Zigbee"* pretende proveer dispositivos que permitan construir redes móviles flexibles y proveer también una plataforma estándar para monitoreo y aplicaciones de control. Por otro lado tenemos el *Simple Media Access Control* (SMAC), una subestructura dentro de *Zigbee*, la cual está representada por un software proporcionado por *Freescale* que nos puede servir de punto de partida para empezar una comunicación muy sencilla con otros dispositivos del mismo genero. El *SMAC* es un código basado en *ANSI C*, este código lo proporciona *Freescale* para poder usarlo en desarrollo de aplicaciones en cualquiera de los *"Transceptores MC1319X"*. El software del que depende, efectúa las funciones básicas de transmisión y recepción como rutinas redimibles y puede ser el fundamento para aplicaciones simples que efectúan ambas funciones, la de aplicación y capa de red. Además, el *SMAC* fue construido para trabajar con cualquier *MCU HCS08* con *SPI*, pero puede adaptarse fácilmente a cualquier otro Microcontrolador.

En nuestro caso utilizaremos el sistema *SMAC*, ya que la utilización que le daremos en el proyecto será más que suficiente para hacer una comunicación con los dispositivos y conseguir el correcto funcionamiento del sistema.

2.3.2.1. CARACTERISTICAS DE LA COMUNICACIÓN SMAC.

A continuación se darán las características principales de las que se compone la comunicación SMAC.

Especificaciones:

- Fácil de usar.
- Sólo 16 primitivas.
- Código menor a 2.5Kbytes.
- Código ANSI C disponible.
- Flexibilidad.
- SPI con cualquier MCU.
- Bi-direccional.
- Camino para migrar a ZigBee.

Destinado:

- Sistemas punto a punto y estrella.
- Bajo consumo.
- Poca memoria.

Como se comenta en estas características, contamos con una serie de primitivas, las cuales pueden ser utilizadas si tenemos los *Drivers* del modelo de microcontrolador que vamos a utilizar, en nuestro caso el *MC13192*. Estas primitivas son increíblemente útiles para la realización de todos los pasos que debe ir haciendo el *MC13192* en nuestro código. Son una serie de funciones predefinidas que nos proporcionan la configuración por software del transceptor, con ellas podemos conseguir estados de transmisión, recepción, hibernación, etcétera, consiguiendo una configuración de las capas de red. Si nos fijamos en la Estructura de Configuración (Figura 7) podemos ver que las capas *SMAC* y Física son inicializadas o configuradas, con lo que la complicación que supone el realizar esa

estructuración nos la evitamos. Las 16 primitivas que nos proporciona este método son la clave para poder conseguir un sistema de comunicación, ya sea punto a punto o en estrella. Estas funciones proporcionan al programador una libertad para programar fácilmente el resto de código que complementará a la parte de comunicación, haciéndolo más accesible y sencillo para aquellos programadores no experimentados.

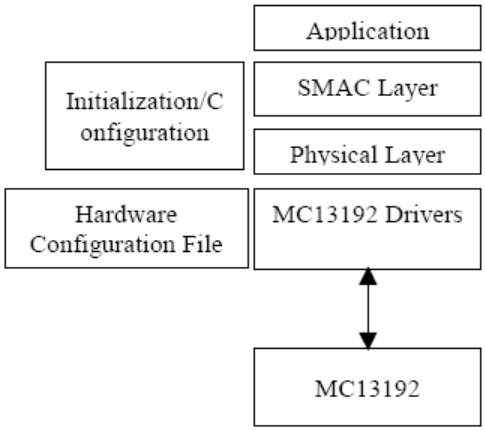


Figura 7. Configuración SMAC.

3. DISEÑO DE LOS COMPONENTES INALÁMBRICOS.

En este capitulo procederemos a la explicación del diseño y programación que se ha realizado con las placas del ***Kit de comunicación MC09S8GT60 SARD*** que nos ofrece ***Freescale*** de ***Motorola***.

Una de las cualidades más importantes de estas placas, en comparación con otras placas similares en cuanto a Hardware, es la posibilidad de comunicación inalámbrica entre ellas. Esto nos proporciona la ventaja de movilidad de los dispositivos, pero conlleva una programación mucho más específica y con mayores dificultades que las cableadas. Para esto cada una de las placas incorpora unos microcontroladores con funciones muy definidas, aunque por otra parte nos ofrece la posibilidad de controlarlo desde el microcontrolador central de la placa, que en este caso es el *"HCS08"*.

Cada uno de los dispositivos debe encargarse de una función específica, por lo que la programación de las placas será significativamente diferente entre ellas. Esto que podría ser un problema se traduce en una simplicidad de la programación de gestión de datos a costa del aumento de dificultad en la de comunicación.

La programación del microcontrolador *"HCS08"* es simple, ya que el lenguaje utilizado es "C". Este tipo de programación acompañado de un compilador que traduce correctamente lo que queremos representar, nos proporciona una programación sencilla en cuanto a gestión. Respecto a la programación de la comunicación entre placas, la utilización de funciones prediseñadas para la comunicación de cada placa *"MC09S08GT60 SARD"* nos facilita el trabajo de adaptación al complicado sistema de comunicación que pueden tener las placas.

El prototipo realizado se centra en el funcionamiento del sistema de gestión de datos y comunicación entre las dos placas, por lo que cada una de ellas utilizará algunos componentes de la placa para simular el funcionamiento global. Una de las placas estará programada para el control del sensor, mientras que la otra será programada para una gestión de la activación o desactivación de un puerto, simulando una actuación sobre el componente que encienda o apague la luz. Ambas

placas han sido programadas para que se autogestionen entre ellas, cerrando un vínculo muy necesario para un trabajo eficiente entre ambas placas.

3.1. DISEÑO DE FUNCIONAMIENTO.

Para el diseño de funcionamiento de cada placa debemos tener muy claro cual va a ser la finalidad de cada placa, a que condiciones se tiene que adaptar y que estados debemos programar para estar disponible en cualquier momento para un intercambio de comunicación con la placa gemela.

Primero debemos concretar que funcionalidades tendrá el sistema, de esta manera será mucho más fácil entender cual es la finalidad para la que trabaja cada placa. Como se ha comentado otras veces, tenemos un dispositivo encargado de la detección y por otro la de un dispositivo para la actuación, el pequeño hilo que une ambos dispositivos es la comunicación inalámbrica por lo que debemos estar siempre pendientes de una posible recepción de datos por parte de una de las placas. Esto conlleva una programación de todo el sistema en función de los tiempos necesarios para una comunicación fiable.

El esquema de funcionamiento general (Figura 8) de ambas placas se puede definir en grandes bloques, con lo que podremos ver una sencilla representación de las partes que componen el conjunto de placas del prototipo.

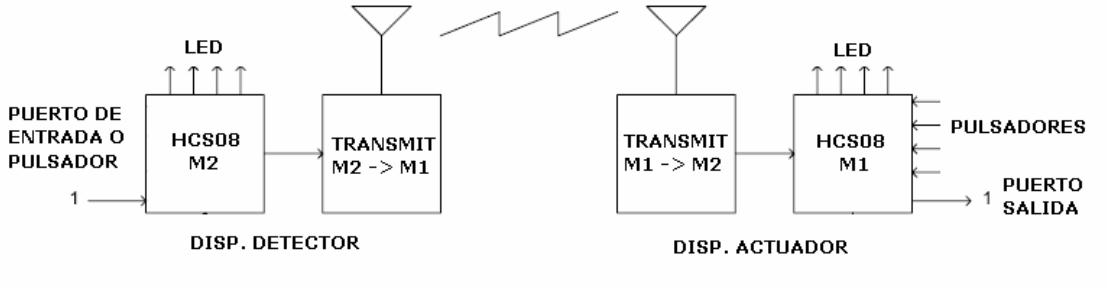


Figura 8. Esquema de funcionamiento general.

Como hemos dicho antes, la programación de ambos microcontroladores será por separado, aunque parte del código de programación sea similar, sobre todo en la parte de comunicación.

El dispositivo detector dispone de un pulsador que simula la detección, implicando detección al mantenerlo pulsado, mientras que si es soltado implica no detección. Una vez que el dispositivo se ha dado cuenta de que hay detección gestiona la serie de estados en los que se encontraba y emite al dispositivo de actuación un cambio en su estado. En el momento en el que el dispositivo actuador recibe todos los paquetes se realiza la verificación de los datos y devuelve información concreta, para verificar con el dispositivo de detección que la transmisión se ha realizado correctamente. Una vez tenemos los cambios sufridos por el detector podemos actuar en consecuencia sobre un puerto, activándolo o desactivándolo, en el prototipo será simulado por el encendido de un *Led*.

Básicamente las funcionalidades de cada placa son programadas para ir alternando estados, como el de recepción, transmisión y el de gestión de datos. Como se ha explicado antes, estos datos son información común entre ambas placas, ya que la base para una buena comunicación es el entendimiento entre ambas y un conocimiento del estado en el que se encuentra una placa por parte de la otra, pudiéndolo denominar como un funcionamiento con conocimiento de causa, ya que ambas saben de antemano que información deben recibir. La comunicación eficiente la conseguimos con una serie de "palabras" (Tabla 3) prefijadas que tendrán el mismo significado para una placa como para la otra, consiguiendo con el envío de un solo dato múltiples significados.

ID	ESTADOS
00	***La luz está apagada.***
FF	***La luz está encendida.***
AA	***Modo automático.***
EE	***Modo manual.***
D1	***El sensor está detectando.***
D0	***El sensor no está detectando.***
88	***Búsqueda de estado para MC1.***
89	***Búsqueda de estado para MC2.***
A0	***Estado automático y luces apagadas.***
AF	***Estado automático y luces encendidas.***
E0	***Estado manual y luces apagadas.***
EF	***Estado manual y luces encendidas.***
ED	***El modo de detección está activado.***
DD	***El modo de detección está desactivado.***

Tabla 3

El diagrama de funcionamiento (Figura 9) por el que se rige el sistema tiene una dinámica propensa al estado de recepción, ya que nuestra finalidad es que ambos dispositivos sean capaces de detectar información del otro en cualquier momento, pero también debemos pensar en el resto de estados que debemos utilizar frecuentemente, por lo que se ha programado ambos dispositivos de manera que cumplan unos tiempos acordes a las necesidades del sistema. Cada uno de estos estados se explicará en los apartados del diseño de cada dispositivo, pero debemos tenerlo en cuenta para el entendimiento del diagrama de funcionamiento.

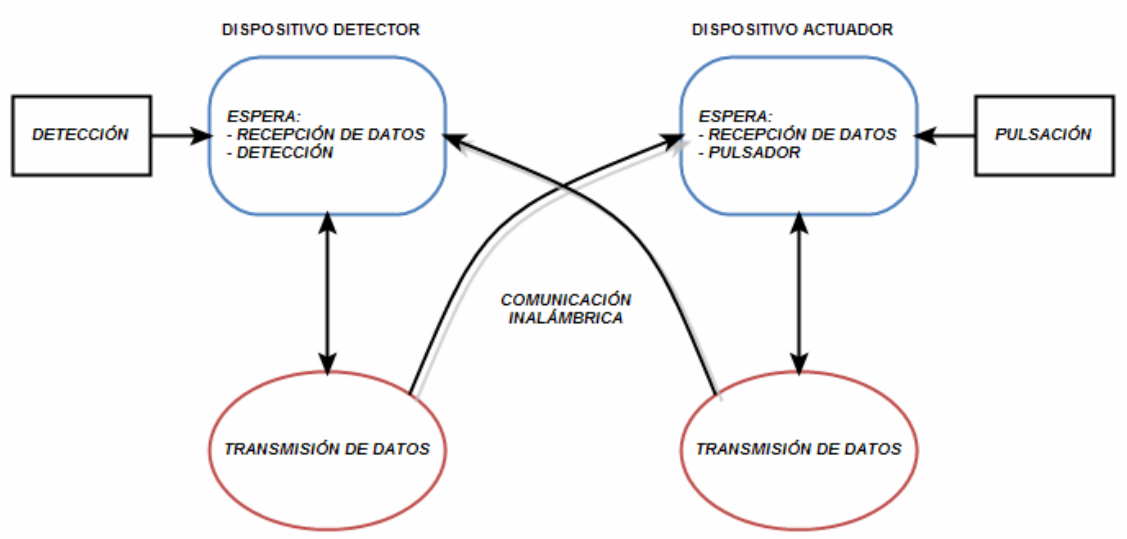


Figura 9. Diagrama de funcionamiento general.

Podemos ver (Figura 9), ambos dispositivos descansan en el estado de recepción, esto nos ofrece la posibilidad de adoptar un estado de reposo, por lo que el consumo puede ser reducido en una gran parte del tiempo. Hay que destacar que en uno de los dispositivos del prototipo no será posible la conexión de alimentación, siendo esto una solución para un ahorro de energía que pueda hacer aumentar el tiempo de autonomía del dispositivo. La reducción de consumo se dará en el apartado de ***Ayuda para bajo consumo*** del "*MC09S8GT60*".

3.1.1. DISEÑO DEL DISPOSITIVO ENCARGADO DE LA DETECCIÓN.

Vamos a tratar de manera más profunda el funcionamiento de la placa encargada del control de detección. Ésta será la encargada de decidir si hay que encender una luz o hay que apagarla, mediante la información que reciba de uno de sus puertos, donde puede estar conectado el sensor o en el caso del prototipo, un pulsador.

La inicialización en la placa de detección consta de una serie de librerías proporcionadas por el kit de desarrollo "***Beekit***" en las que nos basaremos para la programación de múltiples funcionalidades que tienen los microcontroladores que tiene la placa, el "*HCS08*" y "*MC13192*". Estas librerías han sido una base importante para conseguir el resultado final, ya que se ha utilizado código existente para realizar las funciones que utiliza el programa del prototipo.
Posteriormente a las librerías se han declarado las variables necesarias para la gestión de datos, tanto para la comunicación como para el control interno. Tras declarar las variables se realizan las primeras llamadas a funciones (Código 1) predefinidas en las librerías, para inicializar el sistema de transmisión del "*MC13192*", su frecuencia, la interfaz *SPI*, la configuración del *Clock externo*. En la (Código 1) se muestran todas las funciones llamadas en la inicialización, ya que además de las nombradas antes hay otras relacionas con el nivel de potencia al que transmitir mediante el transceptor "*MC13192*" o desactivación de algunos funcionalidades predefinidas en las librerías, las cuales serán nombradas los apartados de ***Limitaciones inalámbricas***.

```
MCUInit();                                  /*Ini MC13192, sistema de transmisión */
RadioInit();                                /*Ini SPI*/
(void)MLMESetMC13192ClockRate(1);           /*Ini frecuencia MC13192 */
UseExternalClock();                         /*Ini tiempo de reloj HCS08 */
(void)MLMEMC13192PAOutputAdjust(OUTPUT_POWER); /*Nivel de potencia TX */
SRTISC=SRTISC&~0x07;                        /* Desactivar wake up timer. */
SPMSC2=SPMSC2&~0x03;                        /* Activa estado bajo consumo modo stop3 */
MC13192_IRQ_IE_BIT = 1;                     /* Configuración Interrupción MC13192 */
EnableInterrupts;                           /* Configuración Interrupciones activas */
```

Código 1. Funciones predefinidas.

Al encontramos dentro del bucle infinito del que está formado el programa principal, podemos ver que la estructuración (Figura 10) que tiene este dispositivo es la de esperar en reposo una recepción de información por parte del *MC1*

(Actuador) durante un cierto tiempo. Una vez se sobrepasa el *TIMEOUT* programado deja el estado de reposo para consultar el estado del sensor, seguidamente toma una serie de decisiones según el dato obtenido, "1" en detección y "0" cuando no detecta. Si el valor es un "0" el dispositivo vuelve al estado de reposo y recepción, mientras que si el resultado es de "1" pasaremos al estado de transmisión e intercambio de datos entre ambas placas. Este último estado actualiza el modo de trabajo, la configuración y el propio estado que tienen ambos dispositivos, teniendo siempre información común que aumentará la estabilidad funcional del sistema. Una vez se han gestionado el intercambio de información pasa otra vez al estado inicial de espera, reposo y recepción.

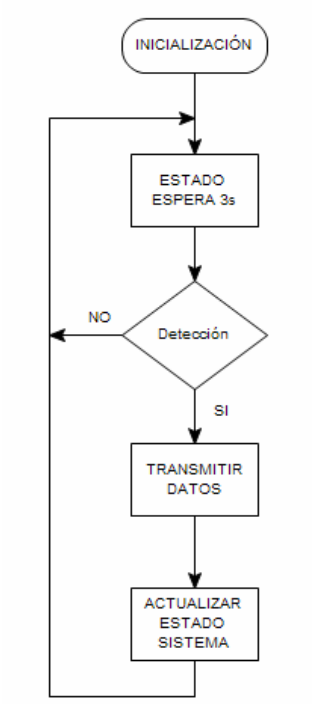


Figura 10. Diagrama del funcionamiento.

La finalidad del sistema global es el de hacer un control automático de la iluminación, por lo que el estado natural de ambos dispositivos es el de un funcionamiento autónomo, sin control humano, pero también se dispone de la posibilidad de un control personal desactivando parcialmente el sistema. Esta decisión la debe tomar el usuario sobre el dispositivo actuador, pero el de detección también juega su papel en este cambio de estado. Para esto hay que remarcar que

el dispositivo detector estará entre dos estados muy generales, estado automático y manual (Figura 11). Este estado vendrá dado, como se ha comentado, por la placa de actuación *MC1* mediante el envío de datos de información, siendo el dato representado por el valor "***AA***" para la elección de modo automático, mientras que el modo manual será dado por el valor "***EE***".

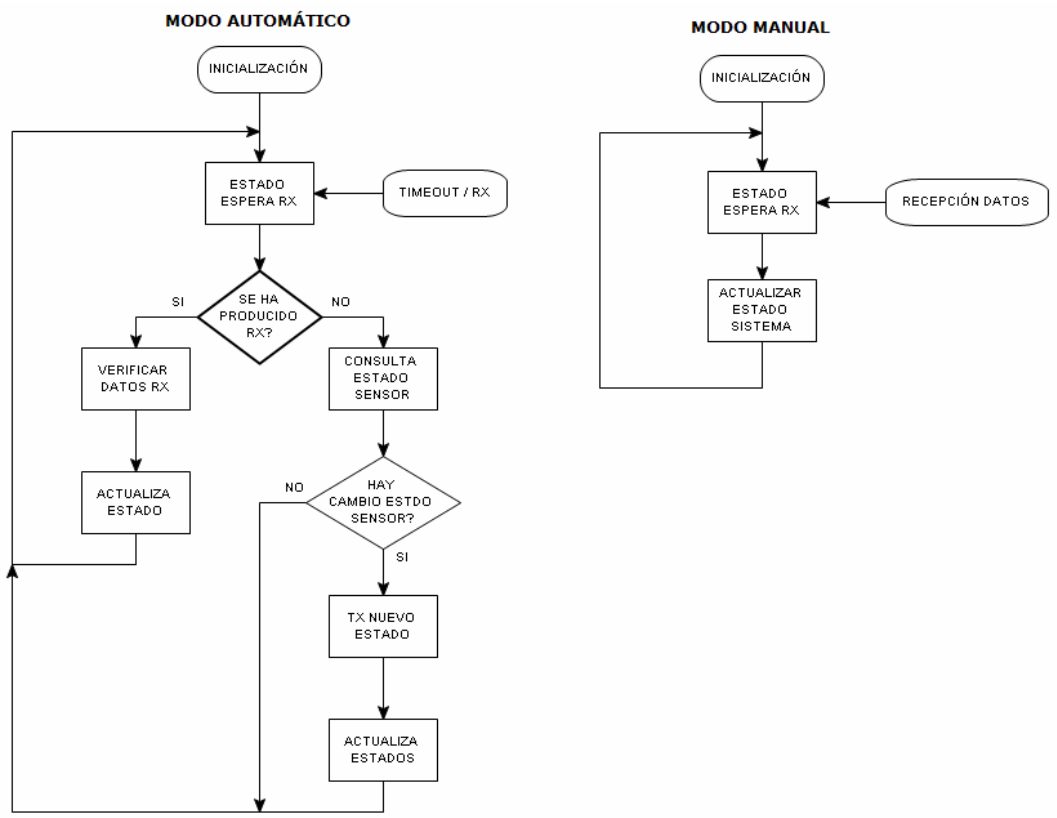


Figura 11. Diagramas de cada modo.

MODO AUTOMÁTICO:

El estado automático (Figura 11) verifica cada cierto tiempo la información que nos da el puerto al que está conectado el sensor, cuando no estamos verificando esto, estaremos a la espera de una recepción de información del dispositivo actuador mediante un bucle, el cual espera un cierto tiempo definido por una variable *TIMEOUT*, que tiene una duración de unos 3 segundos. En el caso de que se reciba información mientras estamos en el bucle, se deja de cumplir una de las condiciones de éste, pasando al siguiente estado, que es el de verificación de la información obtenida.

El sistema de alternar estados se ha utilizado por el hecho de que nos importa por igual la recepción de información y la verificación del estado del sensor **(***puerto***)**. La duración de cada estado es más favorable para la recepción de datos por el simple hecho de que la comunicación se produce en un instante de tiempo muy corto y hay que estar muy pendiente de cuando se produce, mientras que el sensor debe darnos su información durante un periodo de tiempo mayor y su verificación requiere de muy poco tiempo, dándonos mucha más libertad para verificar su estado en cualquier momento.

Aunque la duración del estado de espera para recibir datos es elevada, hay que tener en cuenta que no se ha programado un tiempo aleatorio o en función de los tiempos de comunicación, sino que se ha tenido más en cuenta la frecuencia con la que queremos saber si hay detección, dejando un margen de actuación de entre 0s y 3s después de detectar para una actuación menos brusca, consiguiendo también un tiempo de margen para posibles errores de información producidos por el sensor. Esto nos aporta un tiempo en el que este dispositivo no hace absolutamente nada, y esto es aprovechado para introducir al microcontrolador "*HCS08"* en un estado de bajo consumo.

Para obtener bajo consumo hay que inducir al microcontrolador a un estado en el que se pierden ciertas funcionalidades que ya se han explicado en el apartado correspondiente a las ***Características del HCS08***, por lo que debemos tener muy en cuenta cuales serán nuestras posibilidades de despertar al microcontrolador y con qué método se hará. En nuestro caso, las posibilidades son múltiples, pero sólo se han elegido dos, la interrupción provocada por el "*MC13192"* al recibir información y la del *TIMEOUT* que nos provocará otra interrupción, despertando al microcontrolador para la verificación del estado del sensor.

Este estado de bajo consumo ha sido programado con el modo *STOP3*, el cual sólo actúa sobre el microcontrolador "*HCS08"*. Este modo es el único que nos proporciona la posibilidad de utilizar la interfaz de comunicación serie entre el "*MC13192"* y el "*HCS08*" para crear una interrupción *RTI*. La programación de este modo es muy sencilla, ya que sólo hay que modificar una serie de bits ubicados en el registro *SPMSC2*, con lo que podremos jugar para estar en el modo *STOP1*,

STOP2 o *STOP3* y la propia configuración de cada modo. Para ejecutar el estado sólo es necesaria la ejecución de la instrucción en código ensamblador ***asm {stop}***.

La utilización de este método, para el ahorro de corriente, es sumamente importante para aumentar la autonomía del dispositivo, pero no para su funcionamiento, por lo que desde el punto de vista de programación no tiene mucho interés. Por otro lado no hay que subestimar la necesidad de su utilización, ya que la ventaja obtenida por las posibilidades inalámbricas, en cuanto a movilidad, debe ser conservada con un control mucho más eficiente del consumo generado por los microcontroladores, en este caso del *HCS08*.

MODO MANUAL:

El modo manual es mucho más simple en cuanto a trabajo para el microcontrolador, esto es debido a que en un estado así la necesidad de verificar el estado del sensor no existe, por lo que nuestra única función será la de esperar una recepción de datos por parte del dispositivo actuador, dándonos la orden de volver al modo automático o recibiendo nueva información del estado en el que deben entrar ambas placas.

La necesidad de recibir información de estados de la placa actuadora es muy importante para la estabilidad del sistema, ya la decisión de una de las placas debe ser escuchada por la otra para mantener siempre un sincronismo que asegure la compenetración entre ambas, esto quiere decir que las dos placas deben trabajar al mismo son y no ser terminales "tontos".

Debido a la inactividad que puede llegar a sufrir la placa de detección en el modo manual, el tiempo que se utiliza para el *TIMEOUT*, mencionado en el modo automático, es infinito, lo que quiere decir que el microcontrolador HCS08 permanecerá en estado de reposo de manera constante hasta que el dispositivo reciba información mediante una comunicación con el dispositivo actuador.

En relación al diseño realizado de la comunicación entre placas, será estudiado en el apartado de **Diseño de la comunicación entre dispositivos**, donde podremos

ver como se ha planteado el entorno de verificación y envío de datos dentro del programa principal en cada una de las placas.

3.1.2. DISEÑO DEL DISPOSITIVO ENCARGADO DE LA ACTUACIÓN.

En este apartado trataremos la programación y diseño que se a planteado en la placa encargada de actuar sobre un terminal de luz, en el caso del prototipo se utiliza la salida de un *LED* para simular una señal de salida digital hacia un terminal capaz de interpretar esa señal y cerrar o abrir un circuito de alto amperaje.

El código de inicialización que se propone para el funcionamiento de este dispositivo es similar al del dispositivo detector, con pocos cambios respecto a las variables declaradas, siendo idénticas las variables con finalidad de comunicación entre placas y variables diferentes para la gestión interna del dispositivo. Respecto a las librerías utilizadas, son exactamente iguales a las del dispositivo de detección, ya que utilizaremos prácticamente las mismas funciones que él.

La estructura (Figura 12) que tiene el dispositivo es el de estar en estado automático, intercalando tiempos de espera de recepción de información del dispositivo detector y de la verificación del estado de los pulsadores, prácticamente igual que el dispositivo detector, pero verificando el estado de los pulsadores en vez del detector. Esto nos lleva a realizar una programación de los tiempos de espera más eficiente posible para sincronizar ambas funciones y que no sufran retardos o errores. Para la función de espera debemos utilizar una variable TIMEOUT, como en la programada en el de detección, pero con un tiempo inferior, ya que debemos proporcionar una actuación rápida en caso de utilizar un pulsador. En el estado de espera, si recibimos datos que llegan del dispositivo detector pasaremos a un nuevo estado para su verificación, donde seguidamente se generará una señal a la salida en un puerto, actualizando el estado general del dispositivo y devolviendo la información de los cambios al dispositivo detector.

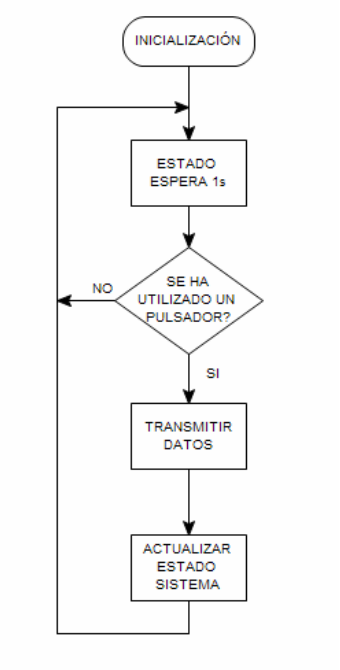


Figura 12. Diagrama general.

Como se ha comentado en el apartado relacionado con el Diseño del dispositivo detector, la placa de actuación trabaja entre dos modos expuestos a la elección del usuario, son el modo automático y el modo manual. Estos modos nos definen las propiedades que adquiere la placa en los dos estados generales en los que se puede encontrar.

MODO AUTOMÁTICO:

En el modo automático el dispositivo detector tiene la finalidad de esperar una recepción de datos por parte del detector, a expensas de una posible llamada por parte de un pulsador (Figura 13). El único pulsador que puede ser utilizado en este dispositivo en modo automático es el de elección de modo (Pulsador1: Auto/Manual), el resto están inutilizados para no afectar el buen funcionamiento del sistema. El tiempo que estamos en el estado de espera inducimos al *"HCS08"* a un modo reposo mediante la función *STOP*, ya nombrada en otros apartados, pasando de este modo a un estado de bajo consumo. Con este estado podremos reducir las necesidades de alimentación, aunque no con el fin de ahorrar batería, ya que este dispositivo, por su ubicación, se le proporciona alimentación constante.

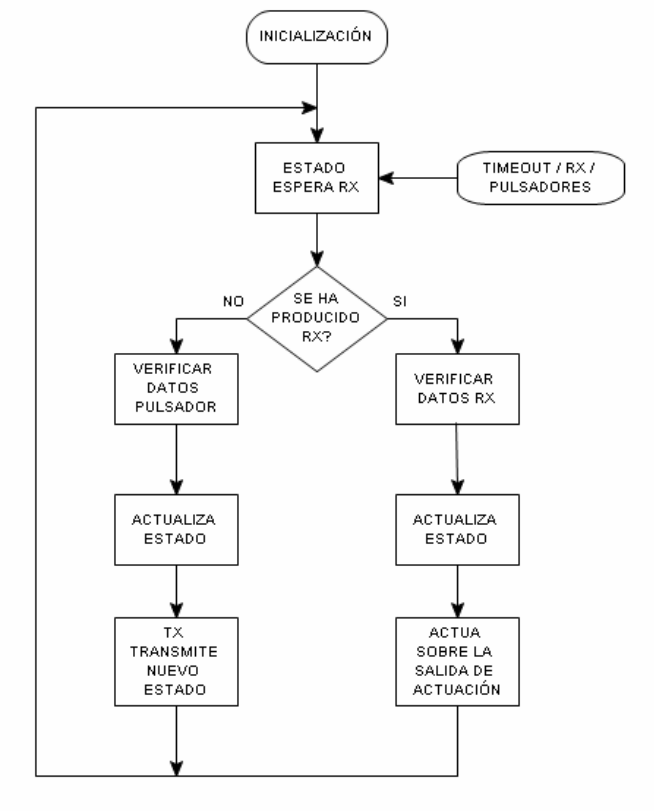


Figura 13. Diagrama de funcionamiento.

MODO MANUAL:

En el modo manual pasaremos al estado en que todos los pulsadores estarán activos dejando de lado una posible recepción. Esto no implica que no entremos en un estado de espera, aunque con la misma frecuencia de verificación del estado de los pulsadores.

- ***Pulsador1:*** Tiene la función de pasar de automático a manual.
- ***Pulsador2:*** Su función es similar a un interruptor, enciende o apaga la luz.
- ***Pulsador3/4:*** Regulador de nivel (no def.).

El *Pulsador1* es el encargado de controlar el modo de funcionamiento, este siempre está activo, mientras que el *Pulsador2* sólo queda activo en modo manual, teniendo la función de poder actuar sobre el puerto de salida de manera manual *(por el usuario)*, ordenando de esta manera el encendido o apagado de las luces.

Los *Pulsadores 3 y 4* no están definidos en el prototipo, pero están preprogramados para una utilización como regulador de nivel, dando la opción de personalizar la potencia de emisión u otro tipo de opciones regulables. Estos están programados para poder pulsar las veces que se desee antes de que el sistema reconozca que se ha llegado a una elección, mediante un tiempo de espera de 3 segundos después de cada pulsación.

El diagrama de funcionamiento (Figura 14) muestra que una vez el sistema ha reconocido la actuación sobre uno de los pulsadores, pasamos a actualizar toda la información que implica el cambio efectuado por el pulsador y envía la información en dirección al dispositivo detector. Se establece esta comunicación debido a que el dispositivo detector debe saber siempre en que estado estamos, aunque sea el encendido o apagado de una luz en modo manual. De esta manera podemos establecer un vínculo vital entre las dos placas, reduciendo un posible estado desincronizado entre ambas.

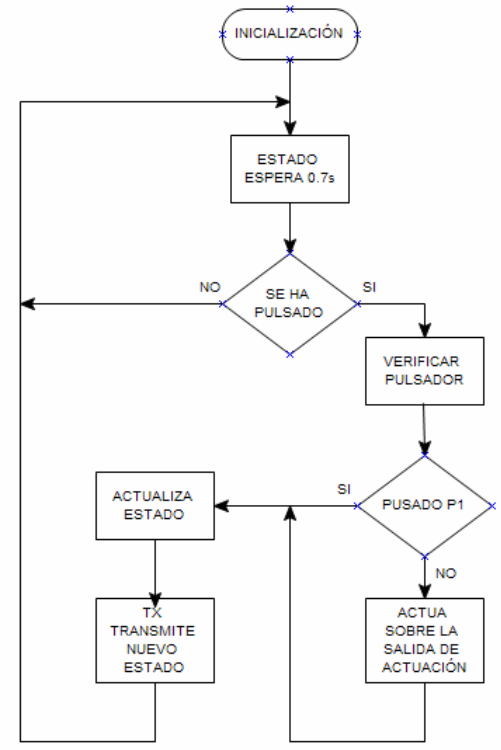


Figura 14. Diagrama modo manual.

Una vez vistos el diseño efectuado en ambos dispositivos, debemos destacar que la programación que se ha llevado a cabo nos muestra la posibilidad de envío y recepción de datos en prácticamente cualquier momento, con lo que la comunicación entre ambas placas es posible en un tanto por ciento muy elevado sin tener ningún tipo de error. El siguiente apartado nos mostrará como se ha diseñado el código destinado a la comunicación mediante el sistema *SMAC*.

3.2. DISEÑO DE LA COMUNICACIÓN ENTRE DISPOSITIVOS.

Muchos sistemas de control utilizan una comunicación mediante cable, teniendo un sistema de comunicación entre dos dispositivos relativamente sencillo de programar, pero la utilización de un sistema de intercambio de datos de manera inalámbrica supone un reto un tanto mayor. El beneficio de utilizar un sistema inalámbrico es el de disponer de una libertad de movimiento de ambos dispositivos que no podemos conseguir con un sistema mediante cable.

El sistema que utilizamos para la comunicación es el **SMAC** ***(****Simple Media Access Control****)***, el cual es un modo de utilización de ***Zigbee*** muy sencillo y que nos evita la programación para un sistema de comunicación complejo. Para esto utilizamos las librerías que nos proporciona el ***Beekit***, el cual nos da la posibilidad de utilizar funciones predefinidas para controlar todo lo relacionado con el intercambio de datos de ambas placas.

El *SMAC* se basa en la utilización del microcontrolador *"MC13192"* para establecer una comunicación con otro µC de la familia ***MC13x***, aunque su utilización será gestionada por el *µC HCS08*.

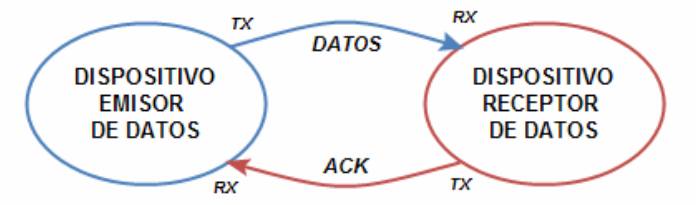


Figura 15. Diagrama funcionamiento.

El funcionamiento (Figura 15) que tendrán los dos dispositivos en cuanto a comunicación se basará en envío de información desde un dispositivo origen hacia un dispositivo destino, una vez en el destino se verificará el resultado obtenido, para cerciorarse de que el origen es conocido, una vez se a verificado el destino devuelve una respuesta **ACK (***Acknowledge***)** hacia el origen para certificar la recepción de los datos. Estos son los tres pasos básicos que se han programado para cualquiera de las comunicaciones que entablen las placas. Esta comunicación está respaldada por un sistema de reenvío de datos, en el caso de que no haya respuesta *ACK* por parte del dispositivo destino.

3.2.1. INPLEMENTACIÓN DEL SOFTWARE DE COMUNICACIÓN.

Dentro del código programado hay dos partes bien diferenciadas para la comunicación, una para la recepción y otra para la transmisión. La de recepción se da justo después del bucle de espera de recepción de datos, explicado parcialmente en los apartados anteriores, pero hay que tener en cuenta que este bucle depende de un estado del *"MC13192"*, en nuestro caso utilizamos el identificador de estado *IDLE*, que significa vacío, el cual cambia de estado una vez se ha rellenado la bandeja de recepción, con lo que nos aseguramos de que salimos del bucle una vez tenemos la información recibida disponible. Si hay recepción directamente se pasa a su verificación, mientras que la transmisión está ubicada después de la verificación de puertos, ya que la transmisión se dará después de un cambio de un pulsador o una señal de entrada a algún puerto. Ambas partes del código se componen de una doble sección destinada tanto a la transmisión como a la recepción, ya que al ***recibir*** verificamos datos y ***transmitimos*** *ACK*, y al realizar una ***trasmisión*** esperamos una ***recepción*** de *ACK*. Esto es así debido a que ambas placas deben tener la posibilidad de entablar una comunicación en cualquier dirección. Lo podemos ver mejor en una representación de los pasos a seguir en ambos estados (Figura 16).

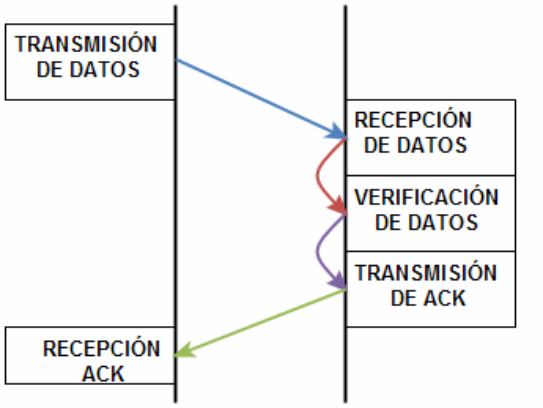


Figura 16. Pasos en la comunicación.

Para programar el dispositivo para una comunicación hay que tener bien claro que sólo podemos estar en una de los dos estados, recibimos o transmitimos, esto hay que tenerlo muy en cuenta porque no tenemos la posibilidad de hacer ambas cosas a la vez. Para poder configurar el estado en el que nos vamos a encontrar, hay que utilizar una serie de funciones (Código 2) que nos proporciona el *Beekit*, con los que podremos cambiar a estado de recepción o transmisión en cualquier parte del programa.

```
(void)MLMERXEnableRequest(&gsRxPacket,0x);              // Rx
(void)MLMERXEnableRequest(&gsRxPacket,TIMEOUTRXMCA);// Rx + interrupción
(void)MCPSDataRequest(&gsTxPacket);                     // Tx
```

Código 2. Funciones de Rx y Tx.

Mediante las funciones ***MLMERXEnableRequest*** (Código 2) podemos configurar el sistema como recepción, mandando una posición de memoria ***&gsRxPacket*** para guardar los paquetes recibidos y otro valor que tiene la función de temporizar una interrupción, en el caso de que no la queramos podemos mandar un ***0x***, con lo que no recibiremos ninguna interrupción. Este ***TIMEOUT*** es el encargado de controlar la frecuencia con la que estaremos en modo recepción, ya que después de llamar a esta función se utiliza el bucle de reposo y espera de una recepción, explicado en los apartados anteriores.

Por si misma la función ***MLMERXEnableRequest*** no interrumpe el programa ni directamente espera una recepción, realmente configura todo el sistema para poder

recibir y guardar los datos obtenidos de manera ordenada en las posiciones del buffer que se han inicializado para este fin, para que nosotros podamos gestionar esos datos de manera sencilla.

En la recepción de datos hay una variable muy importante a tener en cuenta, la ***gsRxPacket.u8Status***, ésta es la encargada de verificar si la secuencia de paquetes recibidos es lógica y tiene sentido, dando un valor de ***SUCCESS*** en el caso de haber recibido correctamente la información. Esta variable no nos marca si la información es para nosotros o la identificación del origen es la que deseamos, sólo nos verifica la lógica de los paquetes recibidos. Podemos decir que la comprobación del estado de esta variable es el primer paso que debemos hacer para pasar al estado de verificación de datos, ya que de lo contrario la verificación daría un resultado incorrecto.

Después de saber que los paquetes han llegado correctamente, pasamos a la verificación de los datos de cada paquete, comprobando la identidad del origen, siendo el significado del mensaje para saber si es una recepción de información o una respuesta de conformidad *ACK*. Una vez hecha esta verificación pasamos a una actualización de estados y actuación en función de la información obtenida. El único paso que queda es la transmisión hacia la otra placa, mandando la información de que todo el proceso ha sido correcto, añadiendo información de estados para la placa que inició la comunicación, consiguiendo lo que ya se ha dicho en otros apartados, una afiliación entre ambas placas que da estabilidad y sincronismo al trabajo que realizan.

Para que el dispositivo origen reciba el *ACK*, debemos cambiar del estado de recepción al de transmisión. Esto lo conseguimos mediante la función **MCPSDataRequest** (Código 2) que configura el dispositivo y realiza la transmisión, pero antes debemos hacer el traspaso de información del las posiciones del buffer de recepción al buffer de transmisión, cambiando lo datos referentes a la actualización de estados y sobre todo actualizar el valor de la posición destinada a identificador de mensaje, cambiando un *TOGGLECMD* por un *ACK*.

Una vez tenemos todas las posiciones del buffer preparadas para la emisión, debemos llamar a la función ***MCPSDataRequest*** la cual coge las posiciones del buffer guardadas en ***&gsTxPacket*** y directamente envía el paquete de datos, que será recibido por el otro dispositivo, es decir, el que fue el origen de la comunicación. De esta manera se acaba la comunicación entre dispositivos.

Por parte del dispositivo origen, después de la emisión de paquetes indiscriminadamente, se prepara para la recepción de una respuesta por parte del dispositivo destino justamente después de haber mandado los paquetes, para esto nos introducimos en una estado de espera idéntico al de espera del diagrama de funcionamiento general, induciendo al *"HCS08"* a un estado de reposo también. El tiempo que estamos en este estado es de unos 0.7s y ha sido ajustado lo suficiente para que de tiempo al dispositivo destino a verificar los datos, reorganice el paquete a enviar y lo envíe. En realidad se ha utilizado un tiempo tan largo para asegurar la recepción independientemente de cuanto tarde el dispositivo destino en preparar la confirmación, ya que un tiempo tan grande es un seguro de que no cortaremos la comunicación antes de tiempo. En el caso de que el dispositivo origen no reciba una respuesta de confirmación procederemos al reenvío de esta información, este procedimiento lo trataremos en el siguiente apartado.

3.2.2. SEGURIDAD EN LA IDENTIFICACIÓN Y RECEPCIÓN DE DATOS.

Para realizar un intercambio de datos de forma segura hay muchos factores a tener en cuenta, ya que debemos tener en cuenta la procedencia de esa información y si lo que se transmite ha llegado a su destino.

IDENTIDAD:

Las posibilidades de identificación de cada dispositivo son amplias y se basan en una parte de la información que emitamos. En nuestro caso hemos utilizado dos posiciones del buffer destinados al envió de información para la identificación de cada placa, una de estas posiciones es ***gauTxDataBuffer[1] = u8Device;,*** donde el *u8Device* coge un valor de 0x01, pudiendo ser cualquiera desde 0x00 a 0xFF***.*** Se ha tomado la decisión de configurar tanto una placa como la otra con la misma

identificación, como si fuera una identificación gemela. Este sistema no tiene porque ser así, ya que la identificación de cada placa puede ser independiente, pero con la condición de que la otra sepa cual es. En el prototipo esta identificación nos proporciona una seguridad ante posibles datos recibidos por otras placas que utilicen el mismo sistema de comunicación, evitando crear un error en el funcionamiento del sistema.

FRECUENCIA:

Otra de las decisiones que se deben tomar es el canal de comunicaciones que vamos a tomar. Como en una comunicación entre radio enlaces de telefonía, las emisiones tienen que ser controladas seleccionando unas frecuencias que no interfieran con sistemas próximos. Para esta elección tenemos 16 posibilidades de elección de canal, desde 2.405GHz hasta 2.480GHz separando cada canal 5MHz. La elección en el prototipo no ha sido una prioridad y se ha optado por la elección del canal preestablecido de 2.405GHz. La elección de un canal adquiere mayor importancia si utilizamos varios de los prototipos que se han diseñado para controlar varias habitaciones de una misma vivienda, ya que no queremos que se comuniquen los dispositivos de unas habitaciones con las de otra, recordemos que estos dispositivos son parejas independientes, trabajando como un sistema de comunicación punto a punto y no como un sistema en estrella, por lo que la elección de una frecuencia para cada pareja de dispositivos evitaría posibles comunicaciones cruzadas y la activación de un dispositivo que a priori debería estar en estado de reposo. Podemos decir que el primer método de seguridad para una comunicación aislada del resto es la elección del canal a utilizar y después la identificación. Con estas dos configuraciones podemos asegurar que sólo habrá comunicación coherente entre los dos dispositivos de nuestro sistema.

Hasta ahora se había hablado de la seguridad desde el punto de vista de establecer una comunicación sin interferencias con otros sistemas, pero ahora pasaremos a la explicación de los métodos de seguridad que se han tomado para asegurar el entendimiento y recepción de datos entre ambas placas una vez se ha establecido una comunicación.

REENVIO:

Uno de los problemas que se nos puede presentar es el de enviar una información desde el origen y no ser recibida por el dispositivo destino. Para identificar este suceso nos basamos en la decisión que debe tomar el dispositivo que ha iniciado el intento de comunicación, mediante un tiempo de espera posterior al envido de datos. Este tiempo de espera es de unos 0.7s aproximadamente, como ya he comentado anteriormente, esto nos asegura que si pasa este tiempo y no hemos recibido respuesta de confirmación por parte del destinatario la comunicación no se ha realizado con éxito y con toda seguridad el destino no ha gestionado ningún tipo de recepción. Una de las posibles causa de este hecho es que la emisión de información a coincidido exactamente en el momento en que el dispositivo destino estaba en un estado de verificación de puertos y no en espera de recepción, algo que hay que asumir. Como tenemos en cuanta que esto puede pasar, se ha diseñado un sistema de reenvió de los datos si no se recibe un *ACK*, volviendo al estado de espera de confirmación, así hasta 4 veces. Se ha comprobado que la comunicación se realiza al primer envío en un 75%, mientras que el reenvío de datos se produce como máximo a 2 veces, esto nos hace pensar que 4 veces es una cantidad más que suficiente para asegurar la comunicación entre las placas.

INICIALIZACIÓN SINCRONIZADA:

Una de las cosas que se le ha querido dar mayor importancia al sistema es el de que ambas placas sepan exactamente que es lo que hace o en que estado está su compañera. El sistema está configurado para que en un inicio las dos placas establezcan un estado común, pero ¿que es lo que puede pasar si pasado un tiempo una de las dos placas se queda sin alimentación o sencillamente se resetea?. Esto lo hemos tenido en cuanta y se ha diseñado un sistema de seguridad que nos garantiza que la última placa en inicializarse se adaptará al estado de la otra.

Este sistemas de seguridad que se ha impuesto en el prototipo es el de inicialización sincronizada entre dispositivos, esto significa que la parada o reset de uno de los dispositivos no implica un fallo en el sistema o desincronización de los dispositivos.

Se ha utilizado el mismo sistema de intercambio de comunicación que se usa en el funcionamiento general del prototipo, pero ejecutándolo en la parte del código del

programa destinado a la inicialización. La diferencia radica en que los datos enviados por la placa inicializada hacen referencia a una *petición* de los estados en que se encuentra la otra placa y no a un envío de información útil para el destinatario. Una vez el destinatario identifica esta petición, envía la información en la que se encuentra, que será muy probablemente el estado en que se encontraba la solicitante antes de su reseteo. En el caso de que haya cambiado el estado de una de las placas mientras la otra estaba sin funcionar, el sistema de sincronización de estados funcionará igualmente, ya que su función no es la de recuperar el estado perdido por el reseteo, sino la adaptación al estado en que se encuentra la placa que había continuado encendida.

3.2.3. LIMITACIONES INALÁMBRICAS.

El sistema de control inalámbrico tiene cualidades de movilidad muy buenas, en comparación a sistemas cableados, pero también tiene una serie de limitaciones vinculadas a la movilidad. Las dos limitaciones más grandes a tener en cuenta son la distancia y la potencia de emisión máxima que puede producir el trasceptor MC13192. Éstas dos limitaciones están íntimamente relacionadas, ya que cuanta mayor potencia de emisión, mayor longitud podremos alcanzar.
Para la configuración de la potencia de salida se utiliza la función ***(void)MLMEMC13192PAOutputAdjust(OUTPUT_POWER);*** donde el valor de la variable Output_Power puede ser variada entre un valor decimal 0 y 11, con los que nos moveremos por unas potencias de salida de -16.6 dBm a 3.6 dBm. Hay que contar con el hecho de que la potencia mínima que puede detectar el transceptor MC13192 es de -92 dBm, por lo que la distancia con las que podremos jugar será de unos ±3 m, ya que la distancia mínima que puede llegar a alcanzar depende de la Rx mínima de -16.6 dBm hasta perder potencia por debajo de -92 dBm.

Uno de los factores a tener en cuenta es los posibles obstáculos con los que se puede encontrar la señal emitida antes de llegar a su destino, siendo las paredes el mayor obstáculo que nos podemos encontrar. El estudio que se ha hecho sobre la distancia media que podemos conseguir en un escenario normal, se hace con una

salida de máxima potencia, ya que el mínimo que podamos configurar estará dentro de las medidas obtenidas por el punto límite. Los valores de distancia media que se han obtenido han sido de 7m sin sufrir ningún problema de comunicación o reenvío, mientras que el punto límite para nuestro prototipo se encontraba a unos 10m. Los resultados que se han presentado han sido obtenidos en una vivienda, en la que nos encontramos en una situación similar a la que se debería encontrar el sistema.

3.3. PROBLEMAS Y SOLUCIONES EN EL DISEÑO.

En este apartado se exponen los problemas más importantes que se han tenido a la hora de programar los dispositivos, en realidad estos problemas afectan a los dos dispositivos, ya que la mayor complicación que se ha tenido es en la comunicación.

PROBLEMA 1:

Uno de los grandes problemas que se ha tenido en la programación es el de controlar el reenvío de información por parte de cualquiera de las placas. El reenvío de información se puede separar en dos posibles ocasiones. La primera es cuando mandamos información al dispositivo destino y éste no lo recibe, pero como el dispositivo origen no sabe si lo ha recibido, cuando pasa un cierto tiempo volvemos a enviarlo; este sería el caso más sencillo y con menor problema. La segunda causa por la que tenemos que hacer un reenvío es la de enviar información al dispositivo destino, éste la verifica y envía el *ACK*, pero éste no llega al dispositivo origen, es decir, el que inició la comunicación, por lo que tampoco sabe si se ha recibido correctamente y procede al reenvío. En este caso, el reenvío de información por segunda vez puede ocasionar algún tipo de error en el destinatario, ya que si se este está programado para cambiar de estado una variable por el simple hecho de recibir información, estaremos haciendo dos cambios cuando en realidad sólo queríamos hacer uno.

SOLUCIÓN 1:

Una solución que se podía plantear era el control por parte de las dos placas de cual era el número de veces que se había enviado una información, pero esto conlleva el control estricto de conteo de envíos por parte de ambas placas y podía resultar un

peligro si se pierden las cuentas, haciendo un corte permanente en la comunicación, por lo que fue descartado por la complicación.

La solución que se ha tomado finalmente, es la creación de las "*palabras*" que contienen la información, como se ha explicado en el punto **Diseño de funcionamiento**, donde estas "*palabras*" contienen la información de lo que ha pasado en el dispositivo que las ha enviado. Este sistema consigue que en vez de llegar una información de lo que debemos hacer, nos llegan noticias de lo que ha pasado en la otra placa, con lo que el receptor de la información sabe como debe modificar sus estados para adaptarse a la nueva situación. Finalmente podemos decir que además de tener un sistema comunicativo más eficiente entre las placas, descartamos un posible problema, pudiendo hacer reenvíos de información sin sufrir alteraciones de funcionamiento.

PROBLEMA 2:

La asignación de tiempos de cada estado en la comunicación ha sido un problema, ya que una vez que se entabla una comunicación entre dos placas hay varios pasos en la comunicación, tanto en una dirección como en otra, por lo que hay que saber que tiempos se deben ajustar para que no se desincronice la comunicación y podamos perder datos. Este suceso normalmente cuando la comunicación está acabando, ya que una vez que nuestro programa se entera de que hay una comunicación, la información ya ha llegado a su destino, pero los pasos que hay que hacer posteriormente dependen de una buena sincronización entre dispositivos.

SOLUCIÓN 2:

El alargar al máximo la espera de una recepción *ACK*, conforme se han verificado los datos de identificación, etcétera, nos proporciona la seguridad de que si no recibimos en un intervalo de 0 a 0.7s, más o menos, es que el destinatario de la información no ha recibido nada, por lo que no es necesaria mayor espera. Este tiempo parece muy elevado para la velocidad con la que podríamos actuar, pero nuestro objetivo no necesita de una actuación excesivamente corta, por lo que no se ajusta al máximo la recepción. Un ajuste excesivo del tiempo de espera de confirmación, provocaría que el dispositivo que debe recibir el *ACK* en algún caso no lo recibiera y por lo tanto procediera a un reenvío de información, anticipándose al dispositivo emisor del *ACK*, que probablemente aun no habría llegado al estado

de espera, con lo que tampoco recibiría el reenvío. Esto que acabo de explicar podríamos decir que es un error acumulativo que podría afectar mucho a la comunicación y que generaría muchos fallos, por lo que es mejor prevenir esta situación alargando el tiempo de espera de confirmación. Por otro lado hay que decir que en el caso de recibir la confirmación, automáticamente se sale de la espera, siendo ese su propio ajuste de tiempo, es decir, si todo funciona bien los tiempos serán lo más ajustados posible por el propio sistema de comunicación planteado.

PROBLEMA 3:

El mayor de los problemas que se han tenido es el de planteamiento de funcionamiento, ya que ambas placas debían tener tres funciones básicas, transmisión, recepción y gestión de datos. La sincronización de las tres funciones, contando que deben ser sincronizadas en ambos dispositivos, plantea el problema de que la elección de cuando y como pasar a un estado de recepción o transmisión recae sobre el programador, que es el que debe elegir la disponibilidad de estados para un funcionamiento fluido.

SOLUCIÓN 3:

En este caso la transmisión de un dispositivo suponía un problema para el dispositivo receptor, en ambos sentidos de la comunicación. Las recepciones de datos en un momento desconocido suponía el problema de planteamiento de cuando debo cambiar de estado. Por todo esto se tomó la decisión de estar la mayor parte del tiempo en espera de una recepción, ya que la transmisión es decisión del propio dispositivo y sabe cuando quiere hacerla, mientras que la recepción podía ser en cualquier momento. Finalmente los dos dispositivos tienen un estado prefijado de recepción, con pequeñas verificaciones de datos que se dan con una frecuencia determinada pero de duración muy corta, con lo que podemos decir que el estado natural de los dos dispositivos es de esperar una recepción de datos de manera prácticamente constante.

PROBLEMAS GENERALES:

El resto de problemas que se han planteado han sido los ya conocidos por la mayoría de programadores, la elección de condiciones que cumplan exactamente las normas de funcionamiento que se pretenden aplicar a una función.

También podemos decir que uno de los problemas ha sido la programación de los pulsadores para evitar los posibles rebotes al soltar o pulsar un botón, creando una serie de temporizadores que tienen la función de esperar un pequeñísimo tiempo sin hacer nada, para asegurar que el tiempo de estabilización del pulsador se haya completado.

Estos pequeños y habituales problemas han sido solucionados sin ningún tipo de problemas, aunque se ha dedicado mucho tiempo en depurar el código de cada condición y temporizador para ir cerrando posibles fallos.

3.4. TESTADO DEL PROTOTIPO.

El testado de funcionamiento que se ha realizado ha sido muy paulatino, ya que se ha ido diseñando, comprobando y ajustando el código a medida que se iba realizando. Para verificar que el funcionamiento del prototipo final es el que esperamos, se han realizado una serie de comprobaciones a tiempo real que consisten en la programación de *"chivatos"* que nos indican que es lo que está haciendo cada placa en cada cambio o suceso. Para la visualización de estos "*chivatos"* se han utilizado los escasos *Leds* de los que disponen las placas, cuatro por placa, por lo que podemos visualizar los pasos, tiempos y estados en que se encuentra el dispositivo, a demás de proporcionar un fácil entendimiento del funcionamiento en la presentación del prototipo.

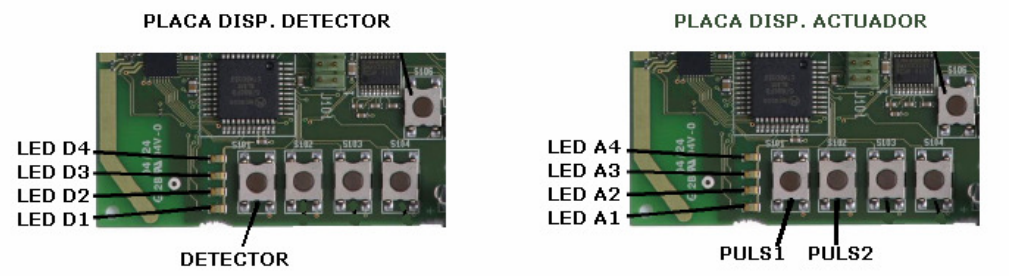


Figura 17. Elementos de testado.

Procederemos a la explicación de cada uno de los *"chivatos"* que se han utilizado para el seguimiento del funcionamiento del sistema.

Empezando por la placa destinada al ***dispositivo detector*** (Figura 17), está compuesto de un sólo pulsador activo que nos simula la detección y de cuatro *Leds*, de los cuales tres están destinados a visualizar los estados en los que se puede encontrar el dispositivo.

- **LED D4:** La finalidad que tiene es la de saber que estamos en modo automático. Este Led realiza un parpadeo cada 3s, mientras está apagado está en estado de espera y reposo, mientras que cuando está encendido, una fracción de segundo, indica que está verificando el estado del sensor.
- **LED D3:** Nos indica que el dispositivo ha pasado a modo automático, quedándose encendido el Led de modo permanente, este estado es el de recepción y espera constantes.
- **LED D1:** Indicador del estado en el que se encuentra la luz en ese momento. Si el dispositivo actuador está dando paso a la corriente para que la luz esté encendida, este *Led* (Disp. Detector) estará encendido, mientras que si la luz está apagada, este *Led* también.

La placa destinada al ***dispositivo actuador*** (Figura 17), está compuesto de cuatro pulsadores activos y cuatro Leds indicadores de estados. Dos de los cuatro *Leds* están destinados a visualizar estados de la luz, mientras que los otros dos entran en el conjunto de *Leds* que visualizan la funcionalidad de regulación de nivel, en el que entrarán en juego los cuatro *Leds*.

- **LED A4:** Utilizado como simulación del puerto de salida para activar o desactivar el componente que controla el encendido o apagado de la luz. Tiene el mismo estado que debería tener la luz, encendido o apagado.
- **LED A1:** Indicador del estado en el que se encuentra la luz en ese momento. Este es un indicador idéntico al *LED D1* que nos encontramos en el dispositivo detector.

Recordemos que el *Pulsador 1* pasa a modo automático o manual, el *Pulsador 2* actúa como interruptor en modo manual y que los *Pulsadores 3 y 4* se pueden utilizar como reguladores de nivel. Los cuatro *Leds* son utilizados para visualizar el estado de nivel, donde el modo de visualización de este nivel en los *Leds* será distinto si utilizamos el *Pulsador 3* o el *4*.

Otro "*chivato"* es el que utilizamos para comprobar si ha habido algún reenvío, en el que se vuelven a utilizar todos los *Leds*. Cada reenvío enciende un *Led* en dirección *LED4* a *LED1* manteniendo el anterior encendido, siendo el máximo de reenvíos 4, perfecto para el número de *Leds* que disponemos. Una vez se ha efectuado correctamente el envío de datos la visualización de los reenvíos se apaga y vuelven a encenderse los *Leds* que había anteriormente activados. En el caso de que no se llegue a entablar una comunicación después de los cuatro preenvíos, los cuatro *Leds* se encenderán y apagaran repetidas veces para señalizar que no se ha podido realizar una comunicación. Hay que destacar que este modo de visualización de reenvíos se utiliza en ambas placas y que es difícil encontrarnos con su aparición por la gran efectividad que aporta el sistema.

3.5. AYUDA PARA UN BAJO CONSUMO.

Las placas proporcionadas por *Freescale* tienen propiedades de bajo consumo, aunque este tipo de cualidades hay que utilizarlas expresamente si queremos reducir el consumo. Adentrándonos un poco más en los componentes de una de las placas, podemos diferenciar dos componentes con los que podremos reducir el consumo, que son los microcontroladores "*MC13192"* y el "HCS08", como ya se ha comentado en anteriores apartados. Cada uno de estos controladores tiene unas características propias y una forma de inducirlos a un estado de reposo.

HCS08:

Este microcontrolador es el núcleo de la placa y tiene una serie de comandos muy sencillos que podemos utilizar para llevarlo a un estado de bajo consumo o reposo. Las instrucciones que se pueden utilizar son cuatro y la forma de llamarlos es mediante un comando en código *ensamblador*, teniendo cada uno de ellos unas

características muy concretas respecto al funcionamiento y tipos de configuración (Tabla 4).

Mode	CPU, Digital Peripherals, FLASH	RAM	Clock Module	ATD	KBI	Regulator	I/O Pins	RTI
Stop1	Off	Off	Off	Disabled	Off	Off	Reset	Off
Stop2	Off	Standby	Off	Disabled	Off	Standby	States held	Optionally on
Stop3	Standby	Standby	Standby (1)	Disabled	Optionally on	Standby	States held	Optionally on

Tabla 4. Modos de bajo consumo.

STOP1:

- Bajo consumo de corriente, 20nA a 2v.
- Podemos salir del estado mediante IRQ o botón de Reset.

Ventajas:

- Consumo de muy bajo potencia.
- Posibilidad de despertarlo mediante un evento externo.

Limitaciones:

- El contenido de la memoria RAM se pierde.
- Todos los contenidos de registros son reseteados al iniciar el proceso de Power-on-reset.

STOP2:

- Bajo consumo parcial de unos 400nA de corriente a 2v.
- Se puede salir del estado mediante IRQ, reset, o Timer interno (RTI).

Ventajas:

- Mantenimiento constante de muy bajo consumo.
- El contenido de RAM se mantiene.

Limitaciones:

- El valor de los registros se resetea, por lo que requiere restauración de algunos periféricos.

STOP3:

- Equivalente al Stop de otros HCS08 con 500nA IDD a 2v.
- Podemos salir con cualquiera de las interrupciones, IRQ, KBI, LVD, RTI o Reset.
- La referencia de reloj externo se puede usar para RTI.

Ventajas:

- Tiene un bajo consumo constante.
- La memoria RAM y los registros retienen sus valores.
- No requiere restaurar los periféricos.
- Puede ser usado el MC13192 Clock Outpu para una RTI como auto wake up.
- Cualquier interrupción puede ser usada.

Limitaciones:

- No conseguimos la misma reducción de consumo como el STOP1 y 2.

WAIT:

- El reloj de bus se mantiene activo. Corriente media de 60 μA a 2v.

Ventajas:

- Consumo reducido en comparación con Run Mode.
- Las interrupciones son ejecutadas inmediatamente.

Limitaciones:

- El regulador de voltaje se mantiene activo, consumiendo más corriente que en los modos STOP.

Los modos que podemos utilizar son variados y se pueden ajustar a cada sistema de diversas maneras, pero lo que está muy claro es que hay que utilizar uno de los comandos *STOP* para conseguir una verdadera reducción de consumo. En el caso de nuestro prototipo se ha utilizado el *STOP3*, ya que una de nuestras condiciones es la de despertar el microcontrolador *HCS08* mediante una interrupción provocada por el *"MC13192"* al recibir un paquete de datos, cosa que con los otros modos no podemos.

MC13192:

Este microcontrolador es el encargado de la gestión de información que se envía o recibe mediante comunicación inalámbrica, por lo que su función en nuestro prototipo será muy importante. Este controlador puede ser inducido a tiempos de reposo mediante una serie de funciones que nos proporciona el kit de desarrollo *Beekit* de *Freescale*, ya comentado en otros apartados, para utilizarlas mediante el *SMAC*. Estas instrucciones son muy sencillas de utilizar, ya que están predefinidas y sólo hay que llamar algunas funciones para poder inducir al reposo a este controlador.

Como he comentado estas funciones nos las proporciona el *Beekit* para desarrollo en SMAC, el cual está creado para una fácil adaptación por parte del programador. Uno de los manuales de SMAC [10] nos muestra cada una de estas funciones, entre las que nos podemos encontrar las dedicadas al ahorro de consumo.

HIBERNATE MODE:

Este modo de reposo es utilizado para conseguir el más bajo nivel de consumo al que se puede inducir al "*MC13192"*.
Es un modo en el que deshabilitamos los bloques de hardware del que se compone el "*MC13192"* y los tiempos de reloj "*Clock Output"* que utiliza.

- *Tiempo de retardo HIVERNATE to IDLE: 18.332 ms.*
- *Instrucción para llamar a la función Hivernate:* **MLMEHibernateRequest();**

DOZE MODE:

Este modo de reposo tiene un nivel medio de bajo consumo. La gran diferencia que hay entre este modo y el de hibernación es que con éste tenemos un nivel de consumo mayor, pero podemos utilizar un *Timeout* para salir automáticamente del estado de reposo.

- *Tiempo de retardo DOZE to IDLE: 332 us.*
- *Instrucción para llamar a la función Doze:* **MLMEDozeRequest();**

WAKE UP MODE:

Esta función nos servirá para salir de cualquiera de los dos modos de reposo al que se puede inducir el *"MC13192"*. Nos permite volver a un estado *IDLE* (*modo normal*) y volver a tener todas las funcionalidades del controlador *"MC13192"*.

- *Instrucción para llamar a la función Wake Up:* **MLMEWakeRequest();**

Estas instrucciones son muy útiles para el ahorro de energía, pero no se puede hacer ningún tipo de cálculo para saber cual es el beneficio que obtenemos, ya que no se ha podido hacer un estudio del consumo de este microcontrolador.

En el caso del prototipo, hemos introducido la hibernación del *"MC13192"*, usándolo en los momentos del programa que sabemos perfectamente que no vamos a utilizar el trasceptor. Como hemos comentado en los apartados de diseño, el sistema está programado para que vaya alternando un estado de recepción con el de verificación de puertos, por lo que la decisión que se ha tomado es la de establecer un reposo del *"MC13192"* durante la verificación de puertos que se hace paulatinamente, ya que de utilizarlo en el estado de recepción perderíamos todas las facultades del *"MC13192"* y no podríamos llevar a cabo ninguna comunicación.

Como hemos visto en el apartado de hibernación, tenemos un tiempo de respuesta del *"MC13192"* bastante elevado después de despertarlo, por lo que en nuestro caso nos hemos asegurado poniendo un tiempo de espera de 30ms para asegurarnos la plena funcionalidad del transceptor una vez lo despertamos.

Con la variedad de posibilidades que nos proporciona Freescale para un reducido consumo de sus placas, podemos ajustar el consumo de ambos controladores para poder conseguir una mayor autonomía, ya que uno de los objetivos de estas placas inalámbricas es el de no tener ninguna restricción en cuanto a movilidad.

3.5.1. SOFTWARE CALCULADOR DE BATERÍA.

Uno de los productos que nos proporciona Freescale para poder realizar un estudio de la reducción de consumo es el Calculador de Batería. Este producto es un software que podemos descargar desde la página oficial de Freescale *(http://www.freescale.com/),* éste nos proporciona un cálculo aproximado de la duración que puede tener una batería al alimentar constantemente una de las placas.

Concretando un poco más, el programa es un software muy sencillo con el que podemos modificar variables concretas (Figura 18) como, tantos por ciento del tiempo en el que estamos en *Stop1 Stop2, Stop3, Wait o Run*, intervalo de tiempo que hay entre *Run* y Reposo, utilización de *RTI*, Voltaje al que se alimenta la *CPU*.

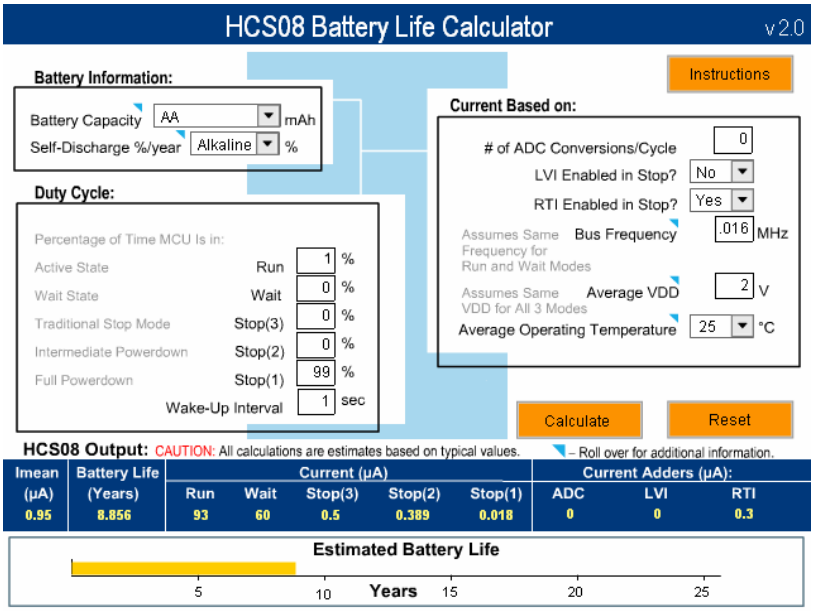


Figura 18. Software calculador de batería.

En el caso de nuestro prototipo debemos tener en cuenta tres variables, tiempo que está funcionando, tiempo que esta en reposo *Stop3* y el intervalo de tiempos del *Wake Up*. La elección de la batería es variable, por lo que el cálculo será aproximado, pero nos podrá dar una idea de las posibilidades que nos aporta la reducción de consumo.

Primero vamos a presentar un cálculo del consumo que genera el sistema sin introducir ningún tipo de modo reposo y utilizando una batería ***Alcalina*** con capacidad ***AA***, con el que podremos ver en la barra inferior de la imagen el tiempo en años que puede durar la batería.

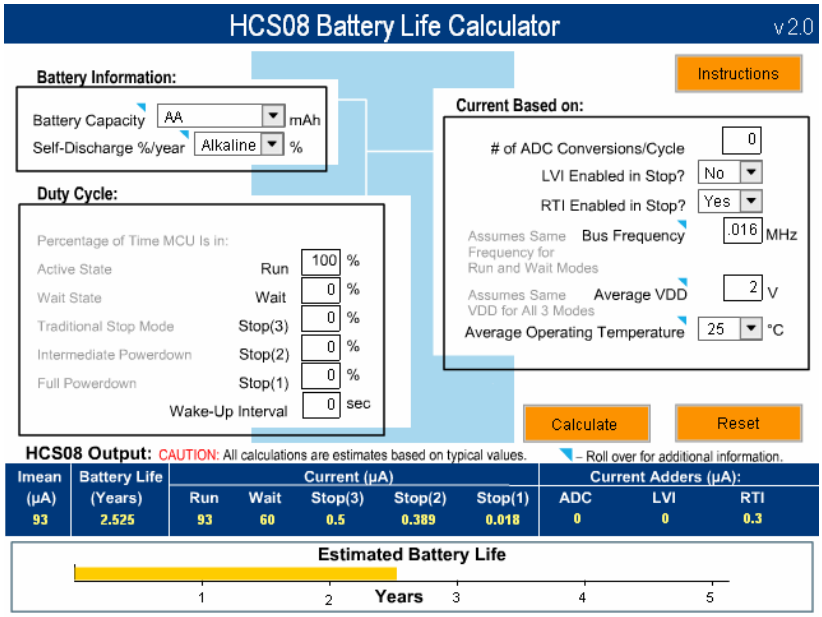


Figura 19. Cálculo sin reposo.

Podemos ver (Figura 19) que el tiempo de duración de la batería es bastante aceptable, unos 2.5 años, pero si nos fijamos, el nivel de consumo es de 93uA, un nivel de consumo muy elevado en comparación al del Stop3, que son 0,5uA. Sabiendo que nuestro programa está basado en tiempos de espera para recibir información de otra placa, llegamos a la conclusión que sería muy recomendable la utilización de estos modos de bajo consumo.

En la (Figura 20) podemos ver la duración que obtenemos al utilizar el *Stop3* en una de las placas, concretamente la del sensor, ya que es la que con seguridad necesitará una alimentación por batería.

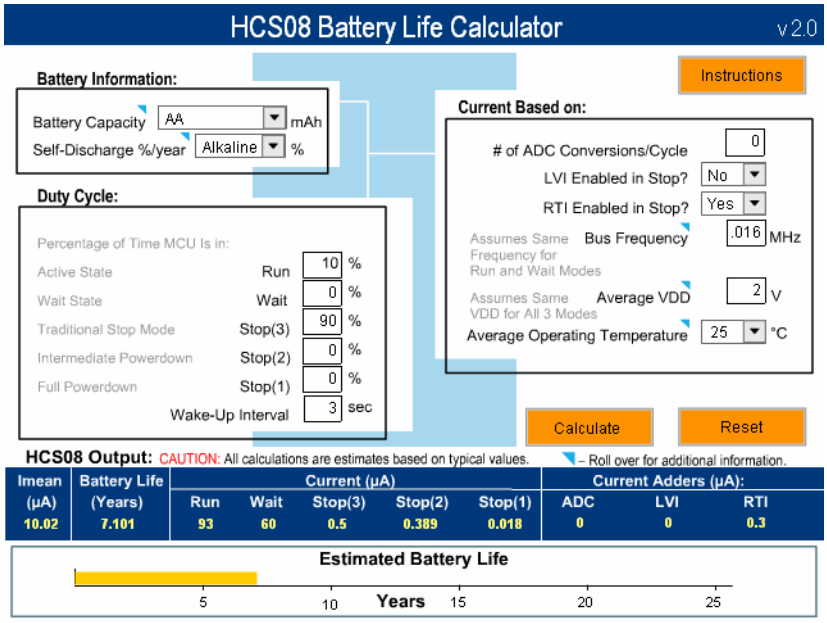


Figura 20. Cálculo con reposo.

En el estudio de consumo que hacemos del dispositivo detector, podemos ver que el nivel de duración de la batería al usar modos de reposo es casi el triple que al no utilizarlos, llegando a una duración de 7.1 años, un valor nada despreciable. Para este estudio hemos tenido en cuenta que el dispositivo actuador estará en modo de espera durante 3s, por lo que también estará en modo reposo, y la duración de la verificación de puertos es una décima del tiempo de espera, por lo que por promedio estaremos un 90% del tiempo en reposo, un 10% del tiempo funcionando y un intervalo de *Wake up* de 3s.

La diferencia que obtenemos entre la utilización de modos de espera y la no utilización de ellos es significativa, por lo que si contamos con el sistema de reposo del *"MC13192"*, del cual no tenemos un calculador, podemos decir que el tiempo de duración del sistema será muy elevado por parte de las placas *"MC09S8GT60 SARD"*.

3.6. ADAPTACIÓN A OTROS SISTEMAS.

El enfoque que se ha dado al sistema es el de controlar automáticamente la iluminación de una habitación, pero no es el único fin para el que pueden estar destinado el funcionamiento que se le ha dado a las placas.

Si miramos el funcionamiento del sistema y la programación que se ha realizado, podemos ver que tiene una funcionalidad muy abierta a múltiples aplicaciones, ya que básicamente controlamos la actuación de un dispositivo mediante la decisión de otro dispositivo a distancia. Las múltiples funciones para las que puede ser adaptado el prototipo que se ha diseñado son amplias, pudiendo ser usadas en:

- Sistemas de detección de temperaturas para activación de terminales de calefacción o refrigeración.
- Control de flujo de aire para cerrar ventanas o bajar persianas.
- Sistemas de encendido de electrodomésticos a distancia.
- Control de intrusión y seguridad.
- Sistemas de control general en ámbito industrial.

Estas sólo son unas cuantas posibilidades que puede tener el prototipo diseñado, teniendo en cuenta que con pequeñas modificaciones del programa se podría ampliar la funcionalidad del sistema de manera sencilla. Este punto es muy importante, ya que el estudio realizado de las dos placas de Freescale puede tener mucho más beneficio del que se puede obtener en una programación cerrada o demasiado enfocada a un sistema en particular.

4. DETECCIÓN DE PRESENCIA.

En este capitulo se explicará el problema referente a la detección de presencia, en el que daremos a conocer las diferentes técnicas que se utilizan para detectar. Se hará un estudio de una serie de detectores explicando sus características principales, ventajas e inconvenientes que tienen cada uno respecto a nuestras necesidades. Una vez terminado el estudio se explicará la elección del sensor que más se ajusta a nuestras necesidades y cual sería su diseño para poder integrarlo en nuestro sistema.

4.1. DEFINICIÓN.

Para poder realizar una detección de cualquier tipo, es necesaria la utilización de un sensor. Un sensor es un dispositivo para detectar y señalar una condición de cambio. Estas condiciones de cambio pueden ser la presencia o ausencia de un objeto o materia [11].

La detección de un cuerpo puede ser muy sencilla y más cuando hay una gran cantidad de sensores en el mercado que nos proporcionan esa posibilidad, pero no todos los sensores son aplicables a un sistema en concreto, cada uno tiene una manera de trabajar y características propias que lo hacen idóneo para un cierto trabajo.

Hay sensores de presencia que se basan en el movimiento, como los basados en microondas, con un gran abanico de sensores en los que escoger, el problema es que la presencia no es un acto que implique movimiento. El paso de una persona por el radio de actuación de un sensor de movimiento hace que éste sea detectado mientras siga en movimiento, pero en cuanto deja de moverse o sus movimientos no son lo suficientemente bruscos, la detección termina, siendo un gran problema cuando lo que queremos es saber si la persona está ahí, no si se está moviendo, por eso es difícil encontrar un sensor que por si solo te pueda proporcionar ese dato tan importante, la presencia. La utilización de múltiples sensores es más habitual, ya que de esta manera se cubren muchos más campos en la detección.

Otros sensores de presencia son los basados en la detección de calor por infrarrojos o sistemas *PIR*, donde también nos encontramos muchos modelos diferentes y con características muy dispares. Hay que tener en cuenta que de sensores por infrarrojos hay varios modelos, los más conocidos son los activos, siendo los detectores de paso, pero nosotros nos centraremos en los modelos de sensor de infrarrojos pasivo (*PIR)*, los cuales tienen un sistema de funcionamiento más complejo.

La radiación infrarroja es un tipo de radiación electromagnética con una longitud de onda mayor a la de la luz visible [12]. Esta longitud de onda está entre 700 nanómetros y un milímetro, siendo la posterior en longitud al la del color rojo, la cual es la longitud más larga de la luz visible.

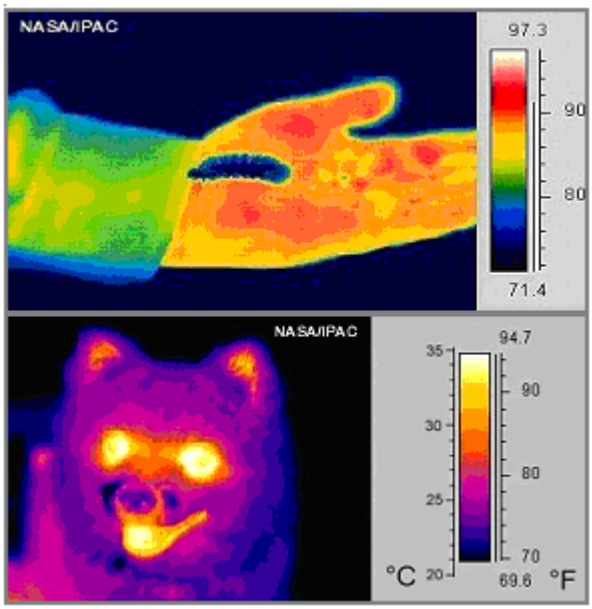


Figura 21. Radiación infrarroja media coloreada.

Cualquier tipo de materia, por sus características energéticas emite radiación (Figura 21). Los seres vivos son un ejemplo, especialmente los mamíferos, ya que emiten una gran radiación en la parte del espectro infrarrojo, debido a su emisión de calor corporal. Esta emisión de calor es debida a que los cuerpos absorben radiación de su entorno, lo que quiere decir, que un cuerpo que está más caliente que su entorno tiende a enfriarse, ya que la emisión de energía excede la rapidez

con la que absorbe. Una vez se han equilibrado, la emisión y la absorción se igualan, aunque este último caso no llega a darse la mayor parte de las veces.

La base de funcionamiento de los infrarrojos es la principal característica que debe tener nuestro componente detector de presencia, ya que el sensor infrarrojo no detecta un movimiento de objetos no calidos, sino un movimiento en la emisión de radiación infrarroja. Estos sensores están compuestos de una lente que concentra los rayos infrarrojos en el centro focal, donde está instalado el sensor propiamente dicho. Esta lente no enfoca todos los rayos que inciden en el sensor, éste presenta zonas de sombra que se intercalan con zonas de detección, de forma que cuando un cuerpo caliente se mueve, se produce un cambio en la distribución de zonas de sombra y detección de radiación, lo que produce una ligera modificación en la temperatura de elemento cerámico del sensor, que es interpretada como detección de una persona [13]. Como se ha comentado, especialmente los mamíferos son los que desprenden una gran cantidad de esta energía, por lo que la detección de una persona se puede realizar comparando el nivel de luz infrarroja que hay en una habitación cuando no hay nadie y cuando hay una persona dentro del radio de actuación.

Un sensor infrarrojos pasivo dentro del sistema puede ser muy efectivo, pero hoy en día se utilizan combinaciones de varios tipos de sensor para dar una mayor fiabilidad al sistema. Uno de los métodos que se puede utilizar es el de incorporar un sensor de ultrasonidos para trabajar conjuntamente con el sensor de infrarrojos, haciendo que el sistema dependa de dos factores que se tienen que dar a la vez, el de detectar la energía desprendida por el calor y la de distancia de objetos. Con esto conseguimos que no nos despierte el sistema si sólo hay movimiento de objetos o por otro lado si sólo es detectado por infrarrojos.

Hay que tener en cuenta que el sensor de ultrasonidos puede tener una sensibilidad muy grande, ya que la detección se basa en el calculo de la distancia mediante la emisión de ondas de sonido a una frecuencia de 40KHz [14]. El funcionamiento básico de los sensores de ultrasonidos como medidores de distancia se muestra de una manera muy clara en (Figura 22), donde se tiene un receptor que emite un

pulso de ultrasonido que rebota sobre un indeterminado objeto y la reflexión de ese pulso es detectada por un receptor de ultrasonidos.

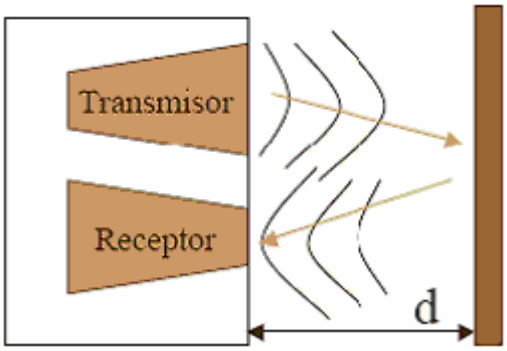


Figura 22. Funcionamiento Sensor Ultrasonidos.

Si medimos el tiempo que transcurre entre la emisión del sonido y la percepción del eco se puede establecer la distancia a la que se encuentra el obstáculo que ha producido la reflexión de la onda, para este calculo hay una representación matemática (1) [15] que nos puede relacionar este efecto.

$$d = \frac{1}{2} ct \tag{1}$$

$$c = c_0 + 0.6\,T\ m/s \qquad c_0: 331 \text{m/s} \tag{2}$$

Tenemos tres variables que nos determinan la velocidad *(V)* del sonidos, el cual es un dato que podemos saber, el tiempo *(T)* que ha transcurrido entre la emisión de la onda y la recepción del rebote, y por último el resultado de la distancia *(d)*.
Hay que tener en cuenta también que la velocidad de propagación de la onda en el aire será diferente en función de la temperatura, para tener en cuenta este factor habrá que sustituir en (2) la representación de la velocidad en función de la temperatura *(T).*

La elección del tipo de sensor principal es claramente uno o más del tipo Infrarrojo pasivo, con la posibilidad de trabajar conjuntamente con un sensor de ultrasonidos, esto nos proporcionará la detección continua de una persona, se mueva o no. Sus

características de funcionamiento hacen de estos sensores las piezas básicas de cualquier sistema de detección de presencia para el control de iluminación.

4.2. ELECCIÓN DEL SENSOR.

En el mercado existen muchos tipos de sensores, y dentro de cada tipo hay una gran cantidad de modelos con diferentes características. Hay sensores de movimiento, de temperatura, de presión y un largo etcétera, pero nosotros nos vamos a centrar en el sensor de Infrarrojos y de ultrasonidos como posibles aplicaciones al sistema proyectado.

Dentro de los sensores de Infrarrojos hay una serie de modelos que hay que diferenciar y como ya se ha dicho antes, el sensor de infrarrojos que nosotros necesitamos es el de Infrarrojos pasivo, ya que el fin para el que está hecho es el de detectar los rayos de energía que desprende cualquier cosa mediante su temperatura. La enorme variedad de sensores nos obligan a tomar decisiones sobre los diferentes modelos con los que podremos disponer para hacer la parte de detección por infrarrojos. A continuación se expondrán alguno de los modelos que se pueden encontrar en el mercado, diferenciados en grandes rasgos, ya que existe la posibilidad de utilizar sensores de infrarrojos que estén prediseñados para esta función o la creación de uno basándonos en la lente del sensor y toda la serie de componentes que caracterizan al sensor en conjunto. Cada uno de ellos será estudiado para valorar las posibilidades que tienen de aplicarse a nuestro sistema.

SENSORES PREDISEÑADOS:

MERCURY EL-500:

Este detector de infrarrojos pasivo es uno de los modelos prediseñados para la detección de presencia (Figura 23). Tiene una serie de características propias que debemos ver si son apropiadas para nuestro sistema. Este modelo en concreto, de la marca ***Mercury***, nos aporta unas cualidades que están bien, pero que parecen centrarse en la detección momentánea de presencia, enfocando más a un trabajo

de seguridad que al de control. Vamos a ver su funcionamiento y partes de las que se compone.

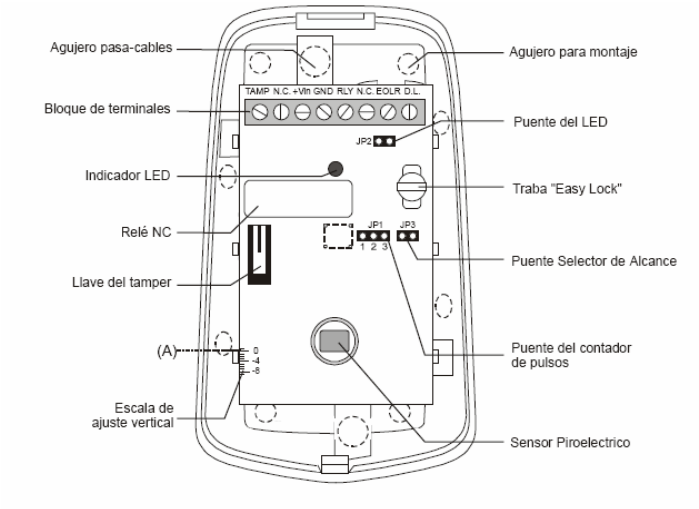


Figura 23. Partes del *Mercury EL-500*.

Como cualquier dispositivo prediseñado se compone de partes destinada al anclaje en una ubicación, pero eso no es lo realmente interesante de este dispositivo. El funcionamiento básico se compone de una detección y posteriormente una emisión de pulsos mientras se detecta por medio de un bloque de terminales. Este bloque de terminales está compuesto de 8 puntos en los que nos podemos encontrar la alimentación, la salida de alarma, la entrada de corriente para la conmutación deseada en la alarma, etcétera. Este sensor es configurable respecto a la cantidad de pulsos de detección que debe esperar para emitir una alarma por el terminal de salida, esto se puede conseguir jugando con las combinaciones de un jumper de tres puntas, siendo la sensibilidad mayor con un aviso cada pulso *(detección)* y siendo más eficiente con un aviso cada tres pulsos *(detecciones)*. El sensor tiene la característica de que utiliza la combinación de dos sensores de infrarrojos para una mayor fiabilidad y el alcance que puede llegar a tener es de 15m dependiendo de la lente que se le instale al dispositivo (FIGURA 24).

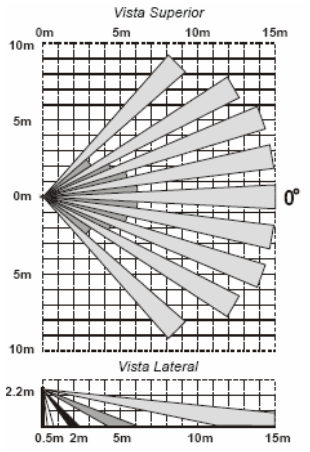


Figura 24. Radio de detección.

Características generales:

- Tensión de entrada: 9 – 16Vcc.
- Consumo: En reserva a 12V – 9mA. Max a 16v – 25mA.
- Duración de la alarma: mínimo 1 segundo.
- Cómputo de impulsos: 1,2 o 3 seleccionables por puente.
- Sensor infrarrojo pasivo doble.
- Salida de alarma 10W máximo.

Ventajas:

- Este sensor puede ser viable, ya que la tensión de funcionamiento que requiere podemos proporcionarla como mínimo a 9Vcc, siendo la máxima que podemos proporcionar conjuntamente con las placas SARD.
- También es adaptable por la posibilidad de personalizar la salida que deseamos para la alarma, mediante la entrada del Tamper, pudiendo ajustar estas corrientes a las necesarias para la placa SARD.
- Posibilidad de cambio de lentes para poder ajustar las distancias detectables.

Inconvenientes:

- El ángulo de apertura, aunque suficiente es un tanto justo, rondando los 90º.
- El encapsulado sólo nos permitiría la utilización de la detección por infrarrojos.

- No tendríamos muchas posibilidades de diseñar un dispositivo único para albergar sensor y controlador en un mismo encapsulado.
- Coste elevado, ya que es un sensor prediseñado.

PIR ISC-PPR1-W16 serie profesional:

El detector ***ISC-PPR1*** (Figura 25) fabricado por ***Bosch*** es uno de los sensores más fiables que se ha podido encontrar. Está compuesto por cuatro sensores, dos piroeléctricos *(infrarrojos)*, un sensor de temperatura y un sensor de nivel de luz. Este sistema se denomina tecnología de fusión de datos, ya que todos están expuestos para poder detectar con un alto grado de eficiencia la detección de una persona [16].

Su funcionamiento se basa en la tecnología óptica trifocal, ésta utiliza tres lentes enfocada a diferentes distancias, con esta combinación podemos controlar hasta 86 zonas de detección, mientras que cada sensor procesa múltiples señales para proporcionar un rendimiento preciso y prácticamente libre de falsas alarmas.

Figura 25. Sensor ISC-PPR1.

Este detector está integrado por un controlador que gestiona completamente el funcionamiento de los componentes y toma las medidas oportunas según la situación que hay en la habitación antes de emitir la señal de alarma. Estas propiedades nos pueden ser muy útiles a la hora de despreocuparnos de la gestión de los sensores por parte de la placa *MC09S8GT60 SARD*. La cobertura que podemos alcanzar es similar a la del *Mercury EL-500* (Figura 26).

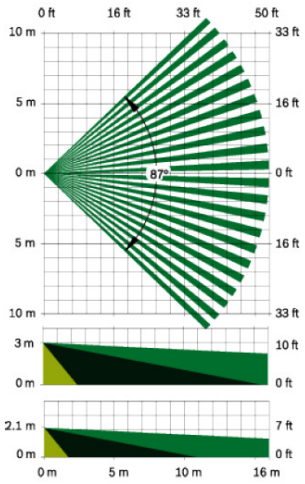


Figura 26. Radio de detección.

Características generales:

- Tensión de entrada: de 9 a 15Vcc.
- Consumo: 15mA a 15Vcc. 10mA a 12Vcc.
- Sensor infrarrojo pasivo tripe.
- Salida de corriente: 125mA máximo.
- Salida de tensión: 25Vcc.
- Salida de alarma 3W máximo.
- Zonas de detección 89.

Ventajas:

- Supresión activa de luz blanca, con lo que conseguimos que la luz directa de una bombilla no nos afecte en la detección de presencia.
- Cobertura del campo (16 m x 21 m o 7,5 m x 10 m)
- Inmunidad contra corrientes de aire, insectos y animales pequeños.
- Compensación de temperatura, siendo igual de efectivo en situaciones con temperaturas que otros detectores darían falsas alarmas.

Inconvenientes:

- Coste, ya que al ser uno de los detectores de última generación tiene un coste de alrededor de 200€.

- Salida alarma de 25Vcc, 10 Ohms. Habría que ajustar la tensión de salida mediante un regulador de tensión a 5v para poder utilizarla en la placa SARD.
- Detector enfocado para trabajar en seguridad perimetral.

SENSORES BASICOS

PIR PARALLAX:

Este sensor (Figura 27) no tiene nada que ver con los detectores antes mencionados, ya que no es un dispositivo prediseñado con una finalidad concreta. La utilización que se le pueda dar a este detector de infrarrojos depende directamente del usuario y de la finalidad para la que lo quiere utilizar. Este modelo de sensor puede ser utilizado tanto para sistemas de control de iluminación como para proyectos de robótica, por lo que es bastante polivalente.

Figura 27. Vistas del sensor PIR de Parallax.

La gran cualidad por la que se puede tener un especial interés en este detector es que tiene unas dimensiones muy reducidas y cumple con casi todas las especificaciones que se pueden pedir para un sistema de detección de presencia, también contamos con una alimentación y una salida de aviso que se adaptan perfectamente a cualquier microcontrolador, siendo un componente diseñado para el acoplamiento a cualquier sistema sin problemas, gracias a que tiene un sistema de conexión al controlador muy sencillo (Figura 28).

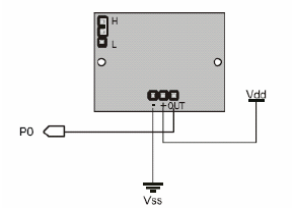


Figura 28. Conexionado.

El nivel de señal que nos proporciona es de 5VDC constante durante unos segundos después de detectar un cambio en el parámetro de la habitación, con lo que nos dará tiempo a que la placa *MC09S8GT60 SARD* pase alguno de sus ciclos y verifique el estado del sensor.

Este dispositivo tiene un tiempo de calibración que puede rondar unos 60 segundos, siendo necesaria la ausencia de personas en el radio de actuación mientras se hace el proceso de "aprendizaje" del estado de la habitación.

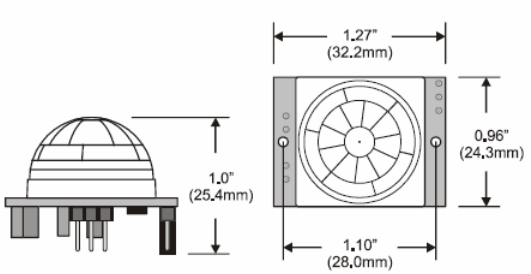


Figura 29. Dimensiones.

Ventajas:

- Entrada de alimentación de 3.3V a 5VDC.
- Salida de aviso de detección ajustable mediante jumper de 3 puntas, para proporcionar señales digitales directas para cualquier microcontrolador.
- Dimensiones muy reducidas (Figura 29).
- Bajo consumo de 100 µA.
- Distancia para habitación estándar, con cobertura de hasta 6m.
- Detección de cambios importantes, por lo que no detecta cambios en sistemas de calefacción.

Inconvenientes:

- Detección de cambios bruscos del patrón Infrarrojo, por lo que no se sabe exactamente si se comporta por igual en cambios al aumentar la señal infrarroja y al disminuirla.
- Necesario otro sensor de respaldo para detección de presencia y no movimientos bruscos.
- No sabemos exactamente el ángulo de apertura que tienen las lentes.

MP MOTION Sensor "Napion":

Los sensores ***MP Motion*** (Figura 30) de la marca "*Panasonic*" son sensores básicos de infrarrojos los cuales no tienen ningún tipo de configuración prediseñada para una utilización concreta al detectar señales infrarrojas. Este sensor es tan básico que la única finalidad que tiene es la de aportar un nivel de señal digital al detectar un nivel de energía infrarrojo. Dentro de la gama de ***MP Motion*** sensor nos centraremos en el sensor capaz de detectar a 10m y el estándar, que puede detectar hasta 5m (Figura 30), ya que el resto son sensores de menor alcance y destinados a otro tipo de aplicaciones (Data Sheet en Anexos).

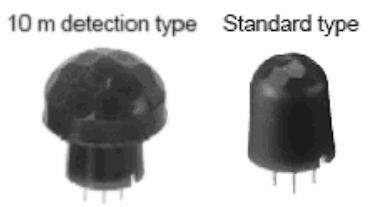


Figura 30. Detectores MP Motion.

Estos sensores tienen un funcionamiento muy simple, incorporando un circuito amplificador y comparador, preparado para conectar directamente a un microcontrolador. Se compone de tres conectores, una de ellas es la alimentación con corriente continua, otra es la salida para avisar de la detección y otra por supuesto es masa (Figura 31).

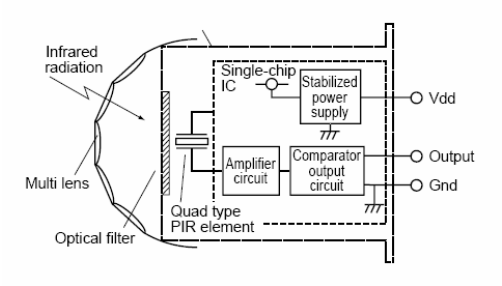


Figura 31. Partes del MP Motion.

Las señales que nos aporta este sensor se pueden representar mediante un diagrama de tiempos (Figura 32), en la que se representa la señal de entrada, la señal de detección interna del sensor y la salida del sensor.

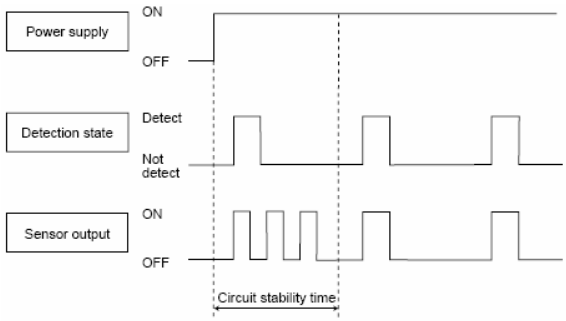


Figura 32. Diagrama de tiempos**.**

El rango de trabajo del sensor de 10 metros (Figura 30) es un tanto superior que el del estándar (Figura 30), aunque las únicas diferencias entre el ***MP Motion de 10m*** (Figura 33) y el ***MP Motion Standar*** (Figura 34) son las de ángulos de apertura y el número de zonas de detección (Tabla 5), ya que técnicamente tienen las mismas características.

Items		Standard type	10m detection type
Rated detection distance		5m (Max.)	10m (Max.)
Detection range	Horizontal	100°	110°
	Vertical	82°	93°
	Detection zone	64 zones	80 zones

Tabla 5. Zonas de detección.

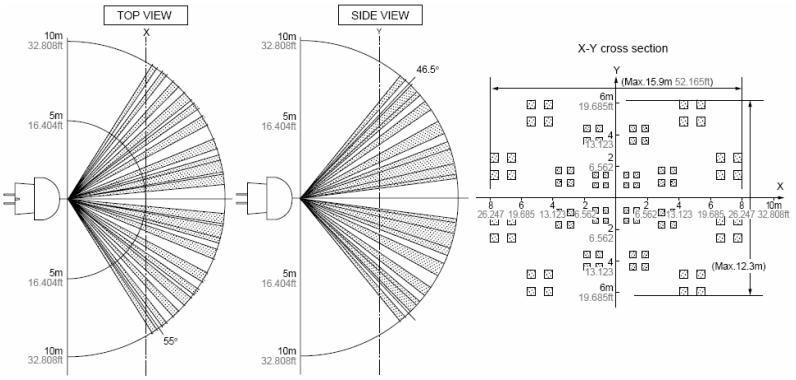


Figura 33. *MP Motion* 10m**.**

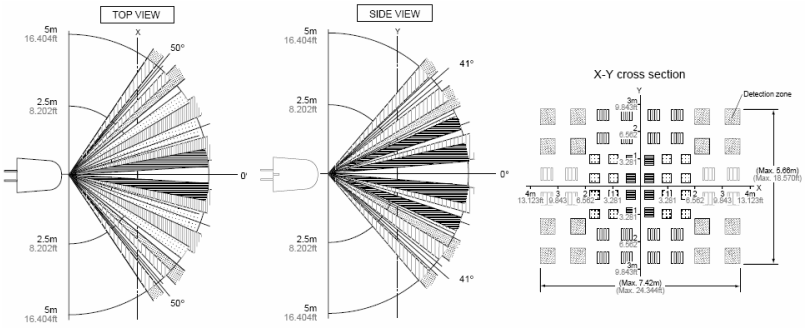


Figura 34. *MP Motion* standar**.**

Las especificaciones respecto a temas de alimentación (Tabla 6) que tienen los dos sensores es igual para uno que para el otro, ya que como se ha dicho antes, la diferencia entre ambos radica en el número de zonas y el radio de actuación.

Items		Symbol	Specified value	Measured conditions
Reted operating voltage	Minimum Typical Maximum	Vdd	3.0V DC — 6.0V DC	
Reted consumption current (Standby)	Typical Maximum	Iw	170μA 300μA	Iout = 0
Output — Current	Maximum	Iout	100μA	Vout ≧ Vdd–0.5
Output — Voltage	Maximum	Vout	Vdd (Same as operating voltage)	
Circuit stability time	Typical Maximum	Twu	7s 30s	

Tabla 6. Especificaciones técnicas**.**

Las grandes cualidades que tiene este sensor lo hacen una posible propuesta para formar parte del sistema proyectado, pero por las características técnicas que hemos podido ver, es un sensor que no podría trabajar solo, ya que la necesidad de

detectación de presencia la cumple al un nivel muy ajustado al que se desea, ya que se basa en la detección de movimiento mediante la detección de infrarrojos en diferentes zonas de la lente, por lo que su utilización debería ser compartida con otro componente detector o un conjunto de estos sensores.

Ventajas:

- Hay un modelo de bajo consumo de corriente en cada sensor.
- Fácil instalación y adaptación a cualquier microcontrolador.
- No necesita una fuente de alimentación externa a las placas SARD.
- Tamaño reducido adaptable a cualquier encapsulado.
- Distancias de 5m a 10m.
- Alto número de zonas de detección en el sensor de 10m de alcance.
- Un gran ángulo de apertura tanto vertical como horizontal.
- Existen modelos con salida de aviso Analógica.
- Reducido coste.

Inconvenientes:

- Nivel de detección de movimiento muy ajustado.
- No detecta niveles de energía por infrarrojos, sino el paso por zonas sensibles.

Acabamos de ver una serie de detectores que nos proporcionan un control de presencia dentro de un perímetro mediante la técnica de detección por infrarrojos, pero otra posibilidad es la de incorporar un sensor de ultrasonidos, ya que hay sistemas que lo utilizan y pueden tener una utilidad dentro del nuestro, siempre que cumpla los requisitos de bajo consumo y eficiencia que deseamos.

La mayor parte de los sensores de infrarrojos están destinados a proyectos de robótica, ya que es un sistema de rastreo de obstáculos muy eficiente en este campo. Vamos a ver a continuación si estos componentes pueden ser efectivos en nuestro sistema.

SENSOR ULTRASONIDOS

MaxSonar EZ1

Éste es unos de los sensores de ultrasonidos (Figura 35) más pequeños y con mejores características en cuanto a consumo que se ha encontrado. Teniendo en cuenta que los sensores por ultrasonidos se utilizan para detectar distancias en robótica, hay que decir que la finalidad es la de detectar objetos estático, pero la función que nosotros deseamos es que el sensor sea el elemento estático, por lo que se debe tener en cuenta para la interpretación de los datos que nos dará.

Figura 35. Detector Maxsonar.

Este sensor funciona de una manera diferente a la mayoría, ya que al sólo tener un único elemento detector, este componente debe tener la función de emisión y recepción, en comparación con la mayoría de los sensores de ultrasonidos, los cuales tienen dos. Al sólo tener un componente emisor y receptor, este sensor tiene una función automática de calculo de distancia, por lo que el cálculo de distancia por parte del microcontrolador que lo controle no es muy complicado. La utilización de un solo componente, en vez de dos, nos aporta un menor consumo.

Las características más importantes de este sensor es que tiene tres salidas diferentes con las que aportarnos la información de la distancia (pulgadas) a la que se encuentran los obstáculos. Los diferentes conectores de entrada salida del dispositivo tienen las siguientes funciones:

- **TX:** Nos proporciona la información de la distancia mediante una serie de datos enviados por medio de una conexión serie **RS-232**.
- **RX:** En este terminal podemos actuar para que el sensor haga medidas, si le aplicamos un nivel lógico alto medirá distancias, mientras que al aplicarle un

nivel lógico bajo dejará de medir y pasará a estado de reposo, en caso de no conectar este terminal a ningún controlador, el sistema funcionará midiendo automáticamente las distancia.

- **AN:** Nos proporciona una señal analógica de un nivel entre 0v y 2.55v el cual representa el valor de la medida (10mV/Pulgada).
- **PW:** Este terminal nos proporciona un pulso con una duración, la cual determina la distancia (147us/Pulgada).
- **GND y +5V:** Conectores para alimentación del sensor de ultrasonidos.

Ventajas:

- Ocupa la mitad de espacio que otros sensores de la misma categoría.
- Consumo muy reducido, en comparación a otros modelos similares.
- Cada ciclo de medida se puede ejecutar automática o manualmente.
- Salidas de lectura directa.
- Las 3 salidas se pueden utilizar simultáneamente.
- Distancia máxima de 6 metros.
- Diseño para trabajar en interiores.

Inconvenientes:

- Un alto coste por unidad ($24).
- Un ángulo de apertura muy por debajo de lo deseado, alrededor de 10º-15º.

Respecto a los sensores de ultrasonidos no es necesario el nombramiento de más modelos, ya que el resto de sensores que nos podemos encontrar con esta tecnología son para detecciones de corto alcance y los que tienen un alcance mayor también tienen un coste, consumo y dimensiones mayores al que hemos presentado.

Para la utilización de éste u otro sensor de ultrasonidos, sería necesaria la instalación de varios de estos sensores en un mismo dispositivo para poder albergar el radio mínimo deseado de 90º, con lo que el precio y el consumo podrían subir excesivamente, incumpliendo con los objetivos marcados.

Como se ha comentado al principio del capítulo, la utilización de este tipo de sensores se utiliza en algunos sistemas parecidos al que se presenta en este proyecto, pero poco a poco se van dejando de utilizar, debido al alto coste, consumo y bajo perímetro de control. En vez de la utilización de detectores de ultrasonidos, los sistemas de hoy en día están haciendo servir cada vez más los detectores por microondas, que tienen un fin muy parecido al de ultrasonidos. La integración de un sistema de detección por microondas en el sistema proyectado es difícil, ya que no es fácil encontrar un detector de microonda barato y sencillo, por no comentar el alto consumo que requieren. Por todo esto se ha descartado la utilización de este tipo de detección.

4.3. PROPUESTA.

Una vez vistos algunos posibles detectores que se podrían utilizar en el sistema proyectado, debemos hacer una elección del componente que más se ajusta a las necesidades del sistema, pero realmente es un tanto difícil la elección, ya que ningún detector por si solo reúne todas las características necesarias para un correcto y eficiente funcionamiento.

Los sensores prediseñados son descartados, ya que estos dispositivos incorporan un microcontrolador que los gestiona. Al tener el controlador *MC09S8GT60 SARD*, la idea sería la de que el propio procesador de la placa gestionara las decisiones, ya que de otra manera quedaría desaprovechado, aumentando el coste del sistema en general sin necesidad.

Todos los sistemas avanzados de detección se basan en la utilización simultánea de diferentes componentes para lograr un objetivo común. En nuestro caso se podrían utilizar diferentes métodos, pero para ello se requiere de un estudio avanzado de cada uno de los componentes para ver cual es el funcionamiento real y que nos puede aportar conectado a nuestro controlador.

Por mi parte la elección sería basar la detección de presencia únicamente con la tecnología de sensores de infrarrojos, ya que los de ultrasonidos incrementan

considerablemente el coste y el consumo. Uno de los modelos que más me ha convencido de los sensores de infrarrojos son los ***MP Motion***, ya que tienen un bajo consumo y su utilización múltiple nos podría dar un rendimiento mayor al de otros componentes, sin afectarnos excesivamente en el consumo.

El ***MP Motion*** utiliza la energía infrarroja para detectar cambios entre zonas, aunque tenemos un número muy elevado de zonas de detección en estos sensores, hay que tener en cuanta que su finalidad es la detección del movimiento de esta energía al paso por las zonas sensibles, por lo que los cambios producidos por una persona en reposo son omitidos. Si el número de zonas fuera mucho mayor, esta sensibilidad aumentaría, pero no se ha encontrado un sensor con mayor número. La manera de solucionar este problema es la instalación de múltiples sensores con igual características, pero con desviaciones en su trayectoria de detección, con lo que podemos conseguir un aumento de la eficiencia a la hora de detectar. La superposición de las zonas en sólo dos sensores (Figura 36), con una pequeña desviación nos duplica el número de zonas, por lo que si aumentamos el número de detectores podemos cubrir virtualmente todo el radio de control con mayores posibilidades de detección.

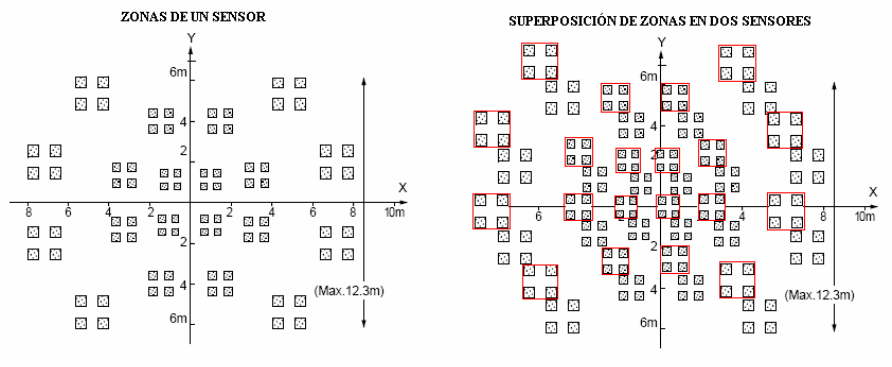


Figura 36. Superposición de zonas.

Está claro que este funcionamiento no es igual de sensible que un único sensor con el doble de zonas de detección, pero podemos decir que si un sensor no detecta presencia, hay posibilidades de que otro si lo haga, siendo la mejor opción dentro de las posibilidades que nos aporta este modelo.

Respecto al consumo que nos supone, es relativamente poco, ya que si se utilizaran por ejemplo cuatro sensores ***MP Motion***, el consumo global en detección sería de un máximo de 1.6mA (400uAx4) y en estado de reposo de un máximo de 1.2mA (300uAx4), esto supone un consumo muy bajo en comparación con cualquiera de los sensores antes mencionados.

El único problema que conlleva la utilización múltiple de detectores es que nuestro controlador *MC09S8GT60 SARD* debería ser programado para poder controlar, gestionar y tomar decisiones de lo que pasa en cada uno de los detectores conectados, pero como he comentado antes, cualquier sistema de hoy en día trabaja con múltiples sensores para que la fiabilidad sea mayor.

4.4. ADQUISICIÓN DE DATOS.

Para el control de un dispositivo externo al microcontrolador utilizado necesitamos un sistema de adquisición de datos, con el que poder tratar la información que nos proporcionará el sensor. En el caso descrito anteriormente en el que se comenta la utilización de cuatro sensores ***MP Motion***, debemos tener en cuenta que cada uno de ellos debe mandar una información propia de la detección, esto implica que cada sensor irá conectado a la placa *MC09S8GT60 SARD* para poder trasmitir directamente al microcontrolador las decisiones que ha tomado. El microcontrolador *HCS08* dispone de varios puertos de entrada/salida que pueden ser utilizados para este fin.

En este caso en concreto, los componentes detectores están adaptados para proporcionar una señal totalmente aceptable por la gran mayoría de los microcontroladores de manera directa, ya que su salida nos proporciona un nivel de tensión y corriente que puede ser tratado por el microcontrolador sin reguladores de tensión o corriente. En definitiva, la adquisición de datos que se propone es de forma directa (Figura 37), sin ningún tipo de elemento intermedio que adapte la señal.

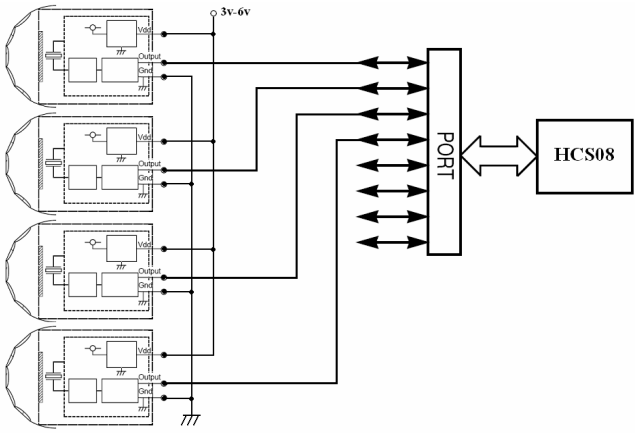


Figura 37. Propuesta de adquisición de datos.

La finalidad del microcontrolador debe ser la de controlar los estados de cada uno de los detectores y tomar decisiones en consecuencia. La señal recibida por el *HCS08* será binaria, ya que la tensión que aplica el detector está dentro del rango de decisiones binarias del Microcontrolador. Haciendo un seguimiento de los puertos en los que están conectados los detectores podemos tratar sus estados como simples "0" o "1", dependiendo si están en contacto abierto *(no detecta)* o en contacto cerrado *(detecta).*

Una de las posibilidades a utilizar en la adquisición de datos, respecto a las entradas de la placa *MC09S8GT60 SARD*, es que se pueden configurar interrupciones por entradas al puerto de un nivel lógico "1", consiguiendo que el sistema sepa en ese mismo instante que se ha producido una detección en alguno de los detectores, evitando así la perdida de información por saltos de ciclo o por la no sincronización entre el *HCS08* y los sensores. El sistema de gestión de decisiones es más una cuestión de programación adaptada al sistema que un problema técnico de adquisición de datos.

5. ELEMENTO ACTUADOR SOBRE EL TERMINAL DE LUZ.

En el siguiente capitulo veremos como trabaja el dispositivo encargado de actuar sobre la iluminación. Este dispositivo no es ninguna de las placas *MC09S8GT60 SARD* con las que hemos realizado el centro de control del sistema, sino que son una serie de componentes adicionales para poder trabajar con otros niveles de potencia requeridos para actuar sobre la iluminación. Para dar una idea mucho más concreta empezaremos estudiando que es un actuador y en que nos puede servir dentro de nuestro sistema. Finalizaremos desarrollando un actuador con la mínima cantidad de componentes, hecho a medida para nuestro sistema.

5.1. DEFINICIÓN.

Un actuador es básicamente un mecanismo por el cual una acción sobre él puede influir en su entorno. Esta acción puede venir de elemento artificial como de un elemento autónomo, en nuestro caso la placa *MC09S8GT60 SARD*.
El actuador es un mecanismo que pone otro elemento en acción de manera automática. También se puede definir como un elemento dentro de un sistema de control que convierte una señal en una acción física [17]. En nuestro caso la finalidad que tendrá el actuador es la de abrir o cerrar un circuito de alto amperaje con un sistema de control digital.

Como todos sabemos, la iluminación de cualquier casa está alimentada a una corriente muy elevada, ya que tanto la iluminación alógena como la iluminación por medio de una simple bombilla requieren de una potencia elevada. También hay otros métodos muy económicos, como la iluminación con fluorescentes, pero también este sistema necesita de una entrada de corriente elevada. La actuación sobre la iluminación siempre está enfocada al campo de la alta tensión, alta en comparación a circuitos electrónicos, por lo que una manipulación de estas corrientes normalmente es realizada por componentes con gran resistencia, como los reguladores de iluminación, o como en la gran mayoría de los casos, con componentes mecánicos con los que accionamos las luces, los interruptores de cualquier casa son un ejemplo.

En nuestro caso, la posibilidad de actuar directamente sobre la luz con los componentes que controlan el conjunto de placas *MC09S8GT60 SARD* es prácticamente imposible, ya que las corrientes que manejamos con estas placas son de muy bajo amperaje. Es normal que un circuito electrónico de estas características no pueda alimentar a un componente que requiera un elevado nivel de potencia, como por ejemplo una bombilla.

Respecto al control sobre la línea de corriente habitual en una instalación de casa, también es muy difícil que un sistema de bajo consumo como los de las placas *MC09S8GT60 SARD* puedan actuar directamente, ya que el rango de trabajo en el que se mueve es de entre 0 y 9v y corrientes de pocos mA, por lo que no hay pista o componente en estas placas capaz de soportar un paso de corriente tan grande como el que se necesita para encender una bombilla.

Para la actuación sobre una línea de alta corriente con un sistema o placa que trabaja con muy poco amperaje hay que utilizar una serie de componentes externos a la placa utilizada, para poder actuar sobre la luz. Estos componentes deben ser capaces de adaptar las necesidades y señales que utiliza un sistema *(Placa de bajo consumo)* con las actuaciones que deben realizarse sobre el otro punto *(la línea eléctrica)*. Estos componentes pueden estar conjuntamente integrados con la placa de bajo consumo, aunque con un formato parcialmente independiente, ya que la función de estos componentes es la de aislar ambas zonas de trabajo, sin repercutir en el rendimiento o en el peor de los casos, un mayor consumo.

A continuación se planteará el diseño y desarrollo que se propone para poder realizar un actuador, el cual nos proporcione la misma función que un interruptor, cambiando sus estados mediante una decisión de la placa *MC09S8GT60 SARD* encargada de decidir el encendido o apagado de la iluminación.

5.2. DESARROLLO.

Para el desarrollo de un actuador en las condiciones que nosotros queremos trabajar, debemos tener en cuenta dos cosas, las limitaciones por parte de la placa y las necesidades que requiere el componente actuador.

La alimentación que nos puede proporcionar la placa *MC09S8GT60 SARD* está limitada, de 0v a 9v proporcionados por la entrada de alimentación y de 0v a 3.6v por parte de los puertos de salida de los que dispone. Esta parte es importante para poder elegir el componente más adecuado a nuestras posibilidades, ya que cada componente tiene unas características que lo definen para un cierto trabajo y nosotros debemos encontrar el componente que se ajuste con mayor rigor, para de esta manera poder tener el mínimo consumo posible tanto en la placa *MC09S8GT60 SARD* como en la parte diseñada para actuar.

Para poder actuar como interruptor en una línea de corriente similar a la de cualquier vivienda debemos encontrar un componente capaz de simular el cierre y apertura de un interruptor mecánico. Este componente debe soportar una tensión de 220v cuando mantiene la línea abierta y una corriente elevada cuando el circuito está cerrado, de alrededor de los 0.3A para una bombilla de 60W. Otra de las características que debemos tener en cuenta es que, el componente debe tener la posibilidad de ejecución sobre el cambio de abierto a cerrado mediante una entrada de corriente lo más baja posible, ya que esta corriente debe ser suministrada por el mismo dispositivo de alimentación que el de la placa *MC09S8GT60 SARD*. Al compartir alimentación ambos placas, debemos tener en cuenta uno de los objetivos principales del proyecto, el de consumir la mínima corriente posible en todo el sistema.

Para el desarrollo del dispositivo actuador se había pensado en la utilización de un *Tiristor* para poder actuar con la corriente que nos proporcionan los puertos de la placa *SARD* y que a su vez poder alimentar la entrada de un *Relé* mecánico en el lado de carga del *Tiristor*, el cual nos proporcionaría la posibilidad de actuar sobre una línea de alta corriente. Por otro lado, esta posibilidad se desestimó, debido a que hay componentes que pueden hacer la función de los dos y utilizando un

mínimo espacio. Esto quiere decir que la idea de crear una placa actuadora desaparece, ya que la posibilidad de contar con un único componente que haga las dos funciones nos lleva a ver posible la integración de éste en la misma placa SARD, consiguiendo una reducción de las dimensiones de manera considerable.

A continuación veremos el componente que se propone instalar para dar una nueva cualidad a la placa *MC09S8GT60 SARD*, la de proporcionar una conexión de control para cargas de gran potencia.

AQ-B Relay PANASONIC:

Estos componentes de la familia **AQ-B** y marca "***Panasonic***" son *relés* de estado sólidos, capaz de controlar unos niveles de tensión y de corrientes perfecto para lo que deseamos realizar. Las dimensiones del *relé* (Figura 37) son muy reducidas respecto a otro tipo de *relés*, dándonos la posibilidad de integración a casi cualquier tipo de placa sin aumentar excesivamente las dimensiones, mientras que la posibilidad de tener modelos en formato horizontal y vertical también nos da mayor libertad para adaptarlo a los requerimientos de espacio que tengamos.

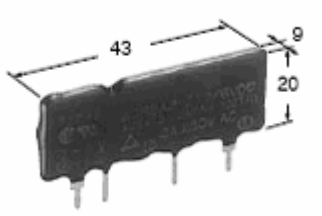


Figura 37. Relé de estado sólido AQ-B.

Una de las cosas que se debían tener en cuenta eran las limitaciones de alimentación que nos puede proporcionar la placa y las necesidades capacitivas que debe tener el *relé* respecto a la carga que conectemos, siendo en este caso todas cumplidas, ya que esta gama de *relés* de estado sólido incorpora una entrada sensible a tensiones que van de 0v a 18v con modelos en los que su rango de actuación en la entrada es de 0v a 3v. La adaptación a la entrada cumple completamente con las necesidades del sistema. Por otro lado, mirando la parte de carga aplicable a esta *relé* podemos ver (Tabla 7) que la capacidad que tiene es de

75v a 250v AC con una corriente máxima de 1A a 2A según el modelo, lo que nos lleva a entender que podemos conectar una carga con potencia máxima de 500W, sólo en el modelo con mayor capacidad.

Item		Part No. AQB2A1-ZT 3/6VDC	Remarks
Input side	Input voltage	3 to 6 V DC	
	Drop-out voltage, min.	1 V	
Load side	Max. load current	2 A	See "DATA 1"
	Load voltage	75 to 125 V AC	
	Frequency	45 to 65 Hz	
	Repetitive peak voltage, max.	400 V	
	Non-repetitive surge current	20A	In one cycle at 60 Hz
	"OFF-state" leakage current	0.6 mA/100 V applied	at 60 Hz
	Max. "ON-state" voltage drop	1.6 V	at max. carrying current
	Min. load current	10 mA	
	OFF state dV/dt	100 V/μs	

Tabla 7. Especificaciones AQ-B.

El *relé* que acabamos de ver cumple con todas las especificaciones, pero el aspecto de sus dimensiones es un tanto ajustado. Para solucionar este aspecto podemos adentrarnos un poco más en los modelos más modernos. La marca ***Panasonic*** nos proporciona un componente con las mismas cualidades que el anterior. Este componentes es un *relé* de la gama **AQ-G**, estos tienen una característica mejores que el de la gama **AQ-B** respecto a las dimensiones (Figura 38). Estas medidas nos pueden ayudar a contribuir en el ahorro de espacio, haciendo de esta manera una placa un poco más flexible en ese aspecto (Data Sheet en Anexos).

Figura 38. Relé de estado sólido AQ-G.

Respecto a las características de funcionamiento no dista mucho del Modelo *AQ-B*, viendo (Tabla 8) pequeños cambios en positivo hacia nuestras necesidades, ya que soporta una tensión mayor a la entrada y una capacidad para soportar una tensión de hasta 264V. Estas mejoras técnicas no son nuestro objetivo, ya que nuestra intención es la de reducir dimensiones, pero siempre es un dato a tener en cuenta.

Item	Type	AQG12105	Remarks
Input side	Input voltage	4 to 6 V DC	
	Drop-out voltage, min.	1 V	
	Reverse voltage	3 V	
Load side	Max. load current	1 A AC	
	Load voltage	75 to 264 V AC	
	Frequency	45 to 65 Hz	
	Non-repetitive surge current	8 A	In one cycle at 60 Hz
	Max. "ON-state" voltage drop	1.6 V	at Max. carrying current
	Min. load current	20 mA	

Tabla 8. Especificaciones AQ-G.

5.3. ACONDICIONAMIENTO DEL COMPONENTE ACTUADOR.

Este modelo de la familia AQ-G hace la función de varios componentes (Figura 39), en este caso en concreto hace la función de un ***Triac*** activado por ***Fotoacoplador***, un ***Triac standar*** y un ***circuito RC*** que sirve como amortiguador para los cambios de estado del *Triac*. Estos componentes pueden ser utilizados por separado, pero la posibilidad de tenerlos en un componente sumamente pequeño nos proporciona una ventaja muy grande, tanto en espacio como en consumo.

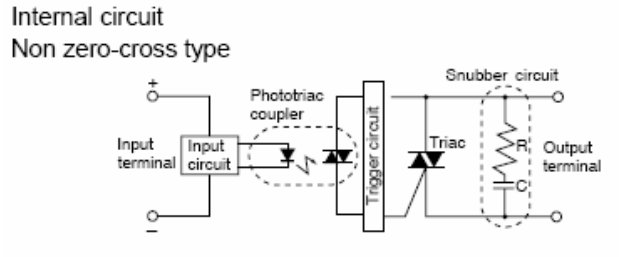


Figura 39. Composición del AQ-G.

El acoplamiento con la placa *MC09S8GT60 SARD* es muy sencillo, ya que como he dicho antes, este componente actuador está capacitado para ser activado con unos niveles de tensión muy bajos, con lo que la conexión entre ambos dispositivos es directa.

Para poder actuar sólo es necesario un solo puerto de la placa *MC09S8GT60 SARD*, configurando el puerto que utilizaremos con dirección de salida. Esta conexión (Figura 40) en un modelo final deberá ser mediante pistas en una réplica de la placa *MC09S8GT60 SARD* adaptada a nuestras necesidades.

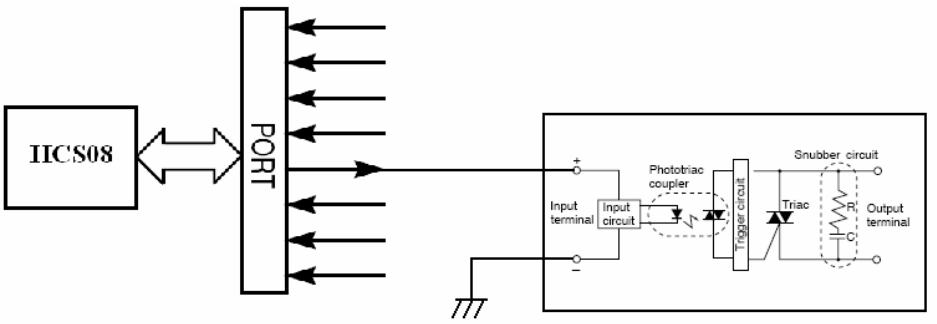


Figura 40. Conexión MC09S8GT60 SARD y AQ-G.

6. MANUAL DEL PROTOTIPO.

En el siguiente capitulo se expone el punto de vista del producto una vez se ha realizado toda la serie de prototipos y testado de funcionamiento global. Una visión de lo que sería el producto final una vez se presenta al mercado, con el visionado de los dispositivos encapsulados, el modo de uso y una serie de referencias necesarias para poder realizar la instalación de los dispositivos.

El objetivo que tiene este proyecto respecto a la visión modular de los dispositivos es la de obtener un producto sencillo, fácil de instalar y muy accesible para posibles usuarios. La dirección que se desea tomar respecto a la accesibilidad, además de intentar conseguir un producto asequible para mucha gente, también es la de que la gente que quiera un sistema automático de iluminación pueda ver en este producto un sistema sencillo de instalar, sin necesidades de preinstalaciones que requieren más que dinero una difícil y problemática instalación en la mayoría de los casos. La posibilidad de una instalación sencilla y sin costes adicionales para el usuario es posible gracias a la movilidad que tienen sus componentes, ya que al utilizar una tecnología inalámbrica y de bajo consumo podemos hacer un sistema con gran versatilidad y fácil instalación.

A continuación se explicarán cada una de las características que tienen los dos dispositivos que completan el sistema, dando detalles de la visión del producto de cara al mercado, utilización de los dispositivos, ubicaciones posibles según dispositivo y montaje de cada uno, finalizando con una explicación de a que tipo de usuario y necesidades se adaptan mejor el sistema.

6.1. VISTA PREVIA DEL DISPOSITIVO DETECTOR.

Este dispositivo, aunque su nomenclatura sea como detector, está compuesto por diferentes componentes, no sólo del elemento que detecta *(sensor MP Motion)*, sino también de una de las placas *MC09S8GT60 SARD* que nos controlará la detección por parte del sensor, decidiendo y actuando en consecuencia, realizando una comunicación inalámbrica con la otra placa en los casos en que sea necesario como ya se ha dicho en otros apartados. Además de estos dos componentes, hay que

añadir que este dispositivo completo debe ser autónomo en alimentación, por lo que también tendremos que contar un elemento de alimentación o batería, los cuales nos proporcionen la corriente necesaria para trabajar durante un largo periodo de tiempo.

El dispositivo completo debe tener un aspecto muy parecido al de la mayoría de detectores del mercado, por lo que la adaptación del sensor *MP Motion*, el *MC09S8GT60 SARD* y la batería deben tener una forma homogénea, de manera en que todo esté en un mismo encapsulado. El tipo de aspecto que tendrá el dispositivo depende del tipo de sensor que vayamos a utilizar, teniendo diferentes posibilidades según la cobertura que deseemos tener con el sensor o la posición en la que se prefiera poner, ya que si es un dispositivo ubicado en el techo con posición horizontal no tendrá las mismas características técnicas en cobertura y forma que otro que requiera posición vertical.

PARTES DEL DISPOSITIVO Y ENCAPSULADO:

A continuación se presentará cada parte del dispositivo y la integración dentro de un mismo encapsulado (Figura 41). En este caso analizaremos el dispositivo detector con formato para techo, teniendo exactamente los mismos componentes que en el caso de un dispositivo de pared, pero con diferente distribución.

El encapsulado del dispositivo tendrá una forma circular, ya que esa propiedad geométrica nos proporciona la posibilidad de obtener una cobertura de 360º horizontales y un ángulo de casi 180º de apertura vertical. La placa irá en el centro del encapsulado, conectada mediante los puertos de salida a los múltiples sensores ubicados estratégicamente para obtener una desviación de orientación adecuada a la interposición de zonas de detección, como se ha explicado en el apartado **Propuesta** en el capítulo de **Detección de presencia**. La posición de la batería debe ser en la parte posterior del encapsulado, haciendo su reemplazo o carga mucho más sencilla.

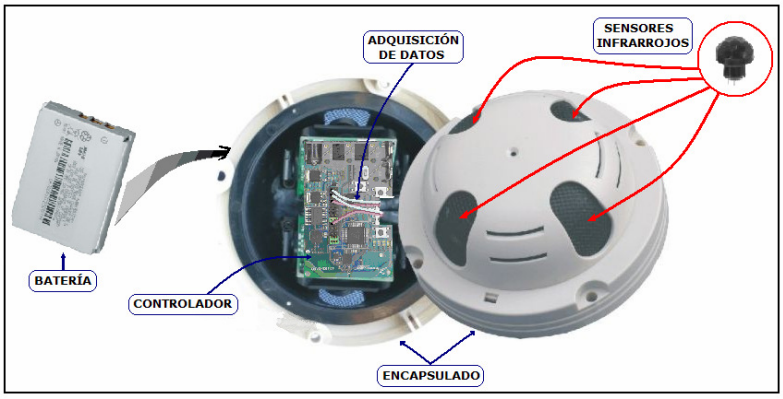


Figura 41. Parte del dispositivo detector.

6.2. VISTA PREVIA DEL DISPOSITIVO ACTUADOR.

La existencia de un dispositivo actuador totalmente separado del dispositivo detector, en comparación a un sistema similar, es debido a que se desea la posibilidad de que el usuario forme parte del sistema, actuando sobre él y personalizando sus funciones de manera sencilla y accesible.

El dispositivo actuador está compuesto, igual que el de detección, de una de las placas *MC09S8GT60 SARD*, conjuntamente con otro componente, el *Relé AQ-G,* encargado de actuar sobre la línea eléctrica en función de las órdenes del controlador. Este conjunto está en realidad diseñado en una misma placa para reducir dimensiones y de esa manera conseguir un dispositivo más manejable y adaptable a las diferentes ubicaciones posibles.

El dispositivo presenta una serie de cualidades en las que el propio usuario puede tomar parte, ya que el dispositivo está compuesto por una serie de pulsadores que tienen diferentes funciones dentro del sistema diseñado. Dada esta especificación debemos tenerlas en cuenta a la hora de decidir su ubicación, por lo que está claro que deberá ser un sitio donde el propio usuario tenga acceso, además de ser un sitio donde se pueda actuar sobre la luz. Para poder entender un poco mejor la necesidad de acceder a este dispositivo por parte del usuario, a continuación se explicará la función de cada pulsador y el fin que tienen dentro del sistema.

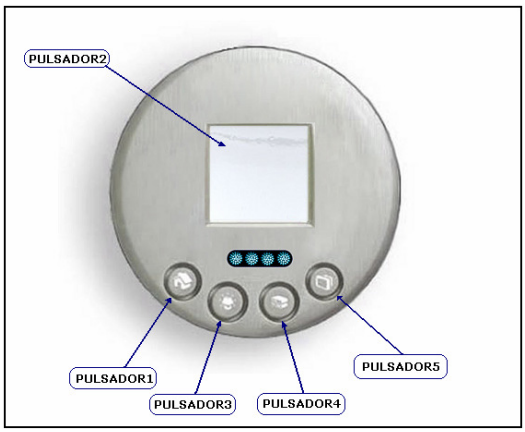


Figura 42. Encapsulado dispositivo Actuador.

- ***Pulsador1:*** Tiene la función de pasar de automático a manual (Figura 42).
- ***Pulsador2:*** Interruptor por contacto para el apagado o encendido de la luz en modo manual (Figura 42).
- ***Pulsador3:*** Regulador de nivel de potencia (dBm) de emisión en ambos dispositivos (Figura 42).
- ***Pulsador4/5:*** Regulador no predefinido (Figura 42).

PARTES DEL DISPOSITIVO Y ENCAPSULADO:

El dispositivo encargado de actuar sobre la luz está compuesto de tres elementos (Figura 43), donde el más importante es el controlador con relé integrado, ya que es el componente que recibirá y controlará la información de encender la luz. Otro componente importante es el que nos proporciona la alimentación para la placa y los terminales para actuar sobre la línea eléctrica, incorporando un transformador para pasar la corriente alterna que nos proporcione la línea eléctrica a una corriente continua con la que poder alimentar la placa a 9v. El último elemento del dispositivo actuador es el encapsulado, que como hemos hablado antes incorporaría una serie de pulsadores para actuar en el sistema de diferentes maneras, incluyendo también un indicador de nivel para poder visualizar diferentes tipos de información del estado del sistema.

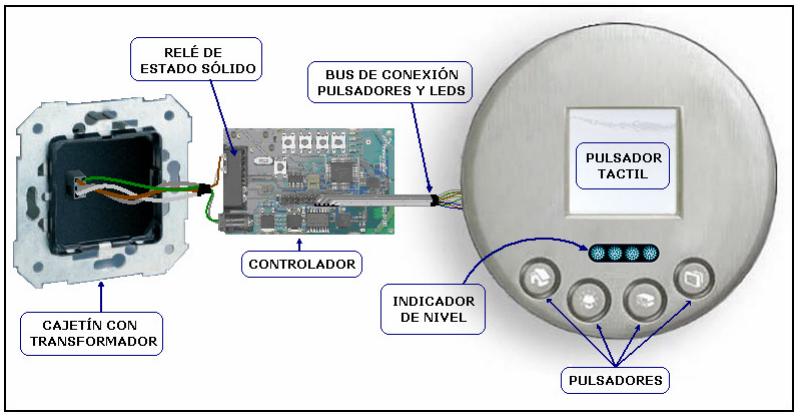


Figura 43. Partes del dispositivo actuador.

6.3. UBICACIÓN DE DISPOSITIVOS.

Cada sistema tiene una serie de indicaciones de utilización y ubicación, en este caso, la gran característica es la de movilidad absoluta, gracias a la comunicación inalámbrica entre los dispositivos. Esta movilidad nos presenta un pequeño problema en la ubicación natural donde colocar los dispositivos. Las posibilidades son muy elevadas, pero en este apartado se intentará explicar la ubicación de los dispositivos en las posiciones más idóneas posibles.

6.3.1. UBICACIÓN DEL DISPOSITIVO DE DETECCIÓN.

La ubicación del dispositivo encargado de la detección no está predefinida, ya que la ubicación depende de si el terminal es en formato de techo o de pared. Dentro de cada modelo se deben tener en cuenta varios aspectos, ya que tanto las características del propio dispositivo, como el ángulo de apertura, como elementos externos al sistema, pueden hacer que el dispositivo emita una señal no deseada o su funcionamiento no sea totalmente eficiente. A continuación se darán a conocer las instrucciones más habituales para la colocación de un dispositivo detector.

- La ubicación del detector debe ser lo más alto posible, dependiendo del radio que queremos detectar. Altura entre 2m y 3m.
- El dispositivo debe tener una dirección de manera que el ángulo de apertura abarque toda la habitación en el caso de un dispositivo de pared, en el caso del dispositivo de techo colocar en una ubicación central de la habitación.
- El dispositivo de pared debe tener una inclinación propicia para poder detectar en todo el perímetro de la habitación. En el caso de un detector de techo debe estar posicionado lo más en el centro posible de la habitación.
- Evitar ubicar el detector en contacto directo con radiadores, conductos de calefacción, refrigeración o acondicionadores de aire.
- No ubicar el detector frente a ventanas o expuesto a luz solar directa.
- No colocar en posiciones muy cercanas a elementos de iluminación o indicando directamente a ellos.
- No se debe ubicar el detector frente a objetos voluminosos o que puedan dificultar la detección de toda la habitación.
- Si la automatización es sobre una luz ubicada en la pared, es recomendable colocar el dispositivo cerca del terminal de luz, pero sin entrar dentro del radio de detección.
- Restricción de lejanía sobre el dispositivo actuador de entre 7m y 10m (Recomendado).

6.3.2. UBICACIÓN DEL DISPOSITIVO ACTUADOR.

La ubicación del dispositivo de actuación está mucho más definida que el del detector, ya que la necesidad de alimentación y de accesibilidad lo limita mucho más. Su ubicación natural sería la misma en la que se encuentran los interruptores mecánicos de cualquier habitación, ya que en este tipo de terminales podemos encontrar tanto alimentación eléctrica como el contacto de actuación sobre la luz deseada. Para la instalación del dispositivo se debe tener en cuenta el elemento adaptador (Figura 44) que se ha explicado en **Partes del dispositivo y encapsulado** de este capítulo, con el que podremos empotrar el dispositivo, ocupando un espacio similar al de un interruptor.

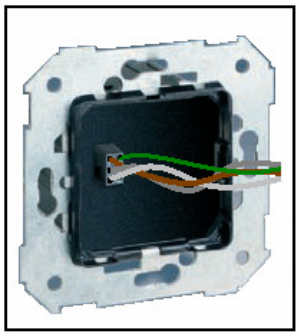


Figura 44. Adaptador a infraestructura.

Otra posibilidad de ubicación depende del tipo de alumbrado que se desea encender, ya que por ejemplo una lámpara de mesa no tiene el mismo sistema de alimentación que la lámpara principal de una habitación. La adaptación del dispositivo actuador no sería difícil, ya que más que cambios técnicos, serían cambios en el diseño del dispositivo.

Para la ubicación de este tipo de dispositivo hay que tener en cuenta varias cosas, dependiendo de la luz en la que se desea tener control.

- El dispositivo actuador debe colocarse en un terminal de alimentación, donde podamos encontrar tanto alimentación para el dispositivo como la alimentación del terminal de luz en el que se actuará.
- El dispositivo debe estar adaptado a los estándares de construcción, referente a las ubicaciones de interruptores.
- Este dispositivo no tiene restricciones de distancia, ya que el objetivo de su ubicación es el de accesibilidad y control de la línea eléctrica. Las restricciones las debe tener el dispositivo detector.
- Ajustar al mínimo el nivel de potencia de emisión necesario para reducir al máximo el nivel de consumo.

6.4. PUESTA EN FUNCIONAMIENTO.

Uno de los momentos más críticos de un sistema automático es el de encendido, ya que hay que seguir una serie de indicaciones de comienzo para que el sistema no se inicie con un funcionamiento incorrecto. Para evitar este problema, los dos dispositivos tienen un sistema de comunicación en el que ambos se ponen de acuerdo para funcionar de forma sincronizada y de manera inmediata. Este sistema proporciona una simplicidad añadida a la puesta en marcha de los dispositivos, ya que el orden de encendido no está predefinido. Este sistema también nos ayuda en el caso de que un dispositivo se reinicie o se apague por falta de batería, ya que cualquiera de los dos dispositivos al iniciarse efectúa una comunicación con el otro dispositivo para configurarse en el mismo estado *(p.e manual o automático)*, el cual debe ser el mismo estado que tenía antes de reiniciarse.

Una vez el sistema está funcionando, tenemos la posibilidad de manipular cualquiera de los pulsadores que incorpora el dispositivo actuador. El primer paso que hay que dar es el de ajustar el nivel de potencia de emisión para conseguir el mínimo consumo. El sistema está configurado inicialmente con el nivel mínimo de potencia de emisión, por lo que para el ajuste de potencia hay que ir aumentando desde el nivel más bajo hasta conseguir que el sistema detecte la presencia del usuario.

Una vez realizadas estas pautas, el sistema estaría completamente en funcionamiento, a partir de ese momento, el usuario podría ajustar el resto de modos y niveles o simplemente dejar el control de la luz al sistema.

6.5. USUARIOS.

El tipo de usuario para el que se enfoca el sistema es muy amplio, esto es debido a que los objetivos que se han marcado para la realización del proyecto no reducen las posibilidades de uso a un cierto tipo de personas. El precio del producto es asequible para un gran abanico de usuarios, como veremos en el capítulo de **Costes**, siendo un punto a favor del producto, si además se le proporciona un

sistema de montaje relativamente sencillo nos encontramos con un sistema con grandes expectativas en cuanto a mercado.

El dispositivo en el que puede interactuar el usuario es sencillo, aunque es más complicado que un simple interruptor, el dispositivo de control puede ser utilizado fácilmente con un aprendizaje rápido para cualquier persona.

Las posibilidades que nos da la comunicación inalámbrica nos proporciona una ventaja a la hora del montaje o desmontaje del mismo, ya que no es necesaria la realización de ningún tipo de obra adicional, algo que sin duda juega a favor del producto, haciendo su instalación mucho más factible en cualquier habitación.

En conclusión, el sistema proyectado está enfocado a un tipo de usuario que no quiera un sistema de control global en su hogar, pero si con el que conseguir un control eficiente de la iluminación y en el que la reducción considerable del precio y sencilla instalación sea un factor prioritario.

7. COSTES DE FABRICACIÓN.

El gran problema que podemos tener en cuanto a la accesibilidad económica que tenga el sistema depende de la reducción de costes en la fabricación. Realizar un prototipo lo más barato posible incrementa las posibilidades de conseguir un producto económico y accesible para posibles usuarios. Para esto es necesario conseguir que con el mínimo de componentes posibles tengamos un funcionamiento correcto pero también con el mínimo gasto posible.

En este caso el prototipo utilizado está compuesto de componentes prediseñados, en referencia a las placas utilizadas para la gestión y comunicación del sistema. Estas placas forman parte de un kit de comunicación que provee Motorola, con lo que el precio de cada placa es superior al de una réplica de la placa diseñada por nosotros. Es normal que un producto acabado sea mucho más caro, por lo que la forma de saber cuanto será el coste real de la placa es la de montarla de nuevo nosotros mismos, además de de intentar rediseñar la placa utilizando sólo las funcionalidades de las placas *MC09S8GT60 SARD* que hemos utilizado para verificar la viabilidad del sistema.

7.1. REDUCCIÓN DE COSTES DE LA PLACA CONTROLADORA.

Para rediseñar una placa con múltiples funcionalidades, primero debemos saber que partes o funcionalidades de la placa original nos son útiles. Si conseguimos reducir a lo indispensable la cantidad de componentes de la placa, conseguiremos un prototipo con menor coste, tanto por el precio de los componentes por separado como por la eliminación de componentes del la placa *MC09S8GT60 SARD* no necesarios.

Alguno de los componentes que no necesitamos de las placas SARD ya se ha mencionado en el capítulo de **Dispositivos de control y comunicación**, como el conector *RS-232* para comunicación serie, pero dentro de la placa *MC09S8GT60 SARD* nos encontramos con más elementos que no serán necesarios en un prototipo definitivo.

Dos de estos componentes son los microcontroladores *MMA6261QR* y *MMA1260D*, que son los controladores de los acelerómetros. Estos microcontroladores no son necesarios en el sistema, ya que están diseñados para controlar aceleraciones de la propia placa y en nuestro caso las placas tendrán una ubicación fija. Estos componente tiene un precio alrededor de 6.5€ para el *MMA6261* y de 5.90€ para el *MMA1260D*, estos reducen el coste de fabricación notablemente. La eliminación de estos microcontroladores nos proporciona espacio para la reducción de la placa o en el caso del dispositivo actuador, el espacio para la colocación del Relé.

También podemos evitar la instalación de otro microcontrolador, el *MAX3318EEUP*, ya que su función es la del control y gestión de la comunicación serie que nos proporciona el conector *RS-232*. El precio de este microcontrolador es de alrededor de 3.5€, un valor elevado en comparación con la mayoría de los componentes que componen la placa original.

Otra serie de componentes no necesarios serían los conectores especiales para la alimentación y algunas resistencias y condensadores. En el caso de la alimentación no es necesario un conector por el simple hecho de que no serán dispositivos con necesidades de conexión o desconexión frecuente. En el caso de algunas resistencias y condensadores no son necesarios ya que estas están vinculadas al acelerómetro o a otro microcontrolador no necesario.

El presupuesto (Tabla 9) realizado para el controlador, indica la cantidad de componentes, marcas y referencias que debemos utilizar, siendo estas las propuestas para el montaje del prototipo. En referencia al precio, éste es orientativo, ya que la búsqueda de los componentes exactos y la gran variedad de proveedores que nos los pueden proporcionar hace la exactitud del coste muy difícil.

Resistencias					
Valor	**Descripción**	**Fabricante**	**Unit**	**Precio u.**	**Sumatorio**
220R	Resistencia Integrable	YAGEO	5	0,03 €	**0,15 €**
470KR	Resistencia Integrable	ROHM	1	0,03 €	**0,03 €**
10R	Resistencia Integrable	YAGEO	1	0,03 €	**0,03 €**
Condensadores					
Valor	**Descripción**	**Fabricante**	**Unit**	**Precio u.**	**Sumatorio**
100nF	Condensador Integrable	MURATA	4	0,03 €	**0,12 €**
1uF	Condensador Integrable	MURATA	1	0,03 €	**0,03 €**
220pF	Condensador Integrable	ROHM	3	0,03 €	**0,09 €**
10pF	Condensador Integrable	MURATA	2	0,03 €	**0,06 €**
18pF	Condensador Integrable	MURATA	2	0,03 €	**0,06 €**
1,5pF	Condensador Integrable	MURATA	1	0,03 €	**0,03 €**
4,7uF	Condensador Integrable	MURATA	3	0,03 €	**0,09 €**
Pulsadores					
Tipo	**Descripción**	**Fabricante**	**Unit**	**Precio u.**	**Sumatorio**
SPST SMD	Pulsador	ALPS	5	1,07 €	**5,35 €**
Contact on/off	Contacto de encendido	ALPS	1	0,50 €	**0,50 €**
Reguladores de tensión					
Tipo	**Descripción**	**Fabricante**	**Unit**	**Precio u.**	**Sumatorio**
LP2950CDT-5	Regulador de tensión	National	1	1,70 €	**1,70 €**
LP2950CDT-3	Regulador de tensión	National	1	1,70 €	**1,70 €**
Componentes Varios					
Tipo	**Descripción**	**Fabricante**	**Unit**	**Precio u.**	**Sumatorio**
Jumper 3p	Conector cable	Keystone	3	0,26 €	**0,78 €**
Quarz 16MHz	Cristal de quarzo	KDS	1	1,50 €	**1,50 €**
entrada DC	Conector	Digi-key	1	1 €	**1,00 €**
entrada Bat	Conector	Digi-key	1	1 €	**1,00 €**
Green Led	Led	Citizen	4	0,23 €	**0,92 €**
Placa foto	Placa virgen 2 caras	Vero Tec.	1	7 €	**7,00 €**
Microcontroladores					
Tipo	**Descripción**	**Fabricante**	**Unit**	**Precio u.**	**Sumatorio**
MC9S08GT60	Controlador HCS08 8bits	Motorola	1	8,50 €	**8,50 €**
MC13192	Transceptor RF	Motorola	1	5,50 €	**5,50 €**
MC9S08GT60	*Controlador HCS08 8bits*	*Motorola*	*1000*	*5,20 €*	***5.200,00 €***
MC13192	*Transceptor RF*	*Motorola*	*1000*	*2,55 €*	***2.550,00 €***
				TOTAL:	**36,14 €**

Tabla 9. Precios de componente.

El coste de fabricación de la placa es reducido, en comparación al precio de venta que pone *Motorola*. Hay que tener en cuenta que el coste que se presenta (Tabla 9)

es en relación a una sola placa, por lo que el valor para el sistema sería el doble, ya que el prototipo final debe estar compuesto de dos de estas placas.
La reducción del coste (Tabla 10) por los componentes no necesarios de la placa utilizada para el estudio es elevado, sobre todo por la eliminación de los acelerómetros.

Componentes no necesarios de la placa SARD					
Tipo	**Descripción**	**Fabricante**	**Unit**	**Precio u.**	**Sumatorio**
1,5KR	Resistencia Integrable	YAGEO	1	0,03 €	0,03 €
1KR	Resistencia Integrable	YAGEO	5	0,03 €	0,15 €
100nF	Condensador Integrable	MURATA	7	0,03 €	0,21 €
1uF	Condensador Integrable	MURATA	1	0,03 €	0,03 €
826632-3	Conector Serie	AMP	1	2 €	2,00 €
MAX3318E	Controlador RS232	Maxim	1	3,80 €	3,80 €
MMA6261QR	Acelerómetro planos XY	Motorola	1	6,50 €	6,50 €
MMA1260D	Acelerómetro plano Z	Motorola	1	5,69 €	5,69 €
				TOTAL:	**18,41 €**

Tabla 10. Componentes no necesarios.

El estudio que hemos realizado sobre los costes para la fabricación de un prototipo nos indica que podemos ahorrarnos casi un 30% del coste real de cada placa, siendo un valor muy significativo a la hora de la realización del presupuesto.

Como hemos comentado antes, necesitamos dos placas para la realización de un prototipo del sistema, por lo que el valor de las dos placas, el impuesto por la compra de los componentes y una posible tasa de envió de materiales nos da un coste total para la realización de las placas de unos 100€ (Tabla 11), esto supone menos de la mitad del valor pagado al proveedor ***Motorola*** por la adquisición del kit de desarrollo con dos de las placas *MC09S8GT60 SARD*. En conclusión, podemos estar satisfechos con el coste que supone la realización de una replica adaptada de las placas *MC09S8GT60 SARD* para el prototipo.

Precio Placa:		**36,14 €**
Unidades:	**2**	**72,28 €**
IVA:	**16%**	**83,84 €**
Tasas envio:	**8,50 €**	**92,34 €**
TOTAL:		**92,34 €**

Tabla 11. Coste total de las placas.

7.2. COSTES DE LOS COMPONENTES SENSOR Y ACTUADOR.

El estudio de costes que se ha realizado sobre los componentes adicionales que completan el sistema se basa en la elección realizada en los capítulos **Detección de presencia** y **Elemento actuador sobre terminal de luz**, que hacen referencia a los sensores y al actuador.

Los costes calculados por la adquisición de los sensores son altos, ya que es un componente con características muy buenas. Por otro lado también se debe matizar que el coste es mayor por la cantidad de sensores utilizados, ya que la propuesta sobre el montaje del sensor propone la integración de múltiples sensores para una mayor fiabilidad. Esto supone un elevado coste para el prototipo, pero esto está a expensas de un estudio mejor sobre que resultados se consiguen en la relación entre la eficiencia que se consigue con la instalación de más sensores y el coste que supone comprarlos (Tabla 12).

Respecto al actuador, es mucho más sencilla la elección de tal componente, por lo que cabe la posibilidad de ajustarnos más sobre el precio (Tabla 12). Hay muchos componentes parecidos y en el estudio realizado se expone un modelo que reúne todas las características necesarias, el precio es elevado, pero la fiabilidad del producto nos podría recompensar.

Componentes adicionales a las placas SARD					
Tipo	**Descripción**	**Fabricante**	**Unit**	**Precio u.**	**Sumatorio**
AQG22112	Relé de estado sólido	Panasonic	1	6,00 €	**6,00 €**
AMN14111	Sensor de movimiento infrarrojo	Panasonic	3	14,00 €	**42,00 €**
				TOTAL:	**48,00 €**

Tabla 12. Costes sensor y actuador.

7.3. COSTE TOTAL DEL PROTOTIPO.

Los costes para la realización de un solo prototipo no sobrepasa los 150€ (Tabla 13), por lo que la realización de múltiples de estos dispositivos podría reducir hasta un 15% los costes, ya que una demanda elevada de cada componente reduce su coste.

Precio Placa:		**36,14 €**
Unidades:	**2**	**72,28 €**
Componentes:	**48 €**	**120,28 €**
IVA:	**16%**	**139,52 €**
Tasas envio:	**8,50 €**	**148,02 €**
TOTAL:		**148,02 €**

Tabla 13. Coste total del prototipo.

Hay que tener en cuenta que el coste del prototipo no incluye un encapsulado acorde a lo expuesto en el capítulo **Manual de prototipo**, por lo que habría que contar con un incremento del coste para la realización de un modelo de carcasa diseñado exclusivamente para el sistema. Este incremento deja un poco en el aire el coste real del producto definitivo, pero viendo el coste total (Tabla 13) del prototipo sin encapsulado podemos decir que el 15% de la reducción de costes por la producción de un número elevado de estos dispositivos compensaría el gasto del diseño y fabricación del encapsulado, pudiendo notar un sensible aumento de coste, pero asumible.

8. CONCLUSIONES

En base al estudio realizado y a raíz de los resultados obtenidos una vez acabado el proyecto, puedo concluir que el sistema diseñado es viable y que los objetivos planteados han sido cumplidos.

El diseño del software de las placas, ha sido la parte del trabajo que me ha supuesto mayores dificultades, ya que dicha parte ha sido la más práctica que se ha realizado y en la que mayor tiempo he dedicado. Este tiempo me ha servido para adquirir unos conocimientos más avanzados en el campo de la programación y gestión de microcontroladores, algo que sin duda era importante para mí cuando tomé la decisión de realizar este proyecto.

El análisis realizado sobre la detección de presencia ha sido muy gratificante, aunque este análisis no ha quedado representado en el prototipo final. En cambio, se ha optado por un estudio de las múltiples posibilidades que tenemos en el mercado. Sin embargo, esta desventaja material se ha traducido en una gran ventaja intelectual, ya que esta labor añadida ha supuesto un mayor interés en la elección de un componente adecuado a las necesidades del sistema. Otro aspecto de la detección que me ha sido de gran utilidad es la idea de que el movimiento implica presencia, pero la presencia no implica movimiento, siendo éste el gran problema en la detección de personas. Todo esto ha hecho que me planteara el diseño desde otro punto de vista, haciendo una propuesta acorde al objetivo deseado en función de las posibilidades del elemento detector.

Los temas abordados tanto de forma teórica como empírica, han sido todos ellos de gran interés para mí. Especialmente, la elección del enfoque que se le ha dado al sistema ha sido una decisión propia, ya que me he basado en los dispositivos que se me han proporcionado. No obstante, el funcionamiento específico que se le ha dado a dicho proyecto ha sido en su mayoría teórico.
La parte dedicada al diseño de los dispositivos se ha enfocado a un control de iluminación, aunque el trabajo realizado con éstos, nos brinda la posibilidad de enfocar su función a una gran variedad de utilidades, ya que puede realizar un control de otro tipo de componentes.

En conclusión, el último objetivo que se pretendía alcanzar con este Proyecto es la propia formación del autor. El desarrollo del software ha supuesto un gran esfuerzo, pero quedan compensados con los conocimientos teóricos y prácticos adquiridos durante todo este tiempo. Creo que es un estudio interesante sobre comunicación y consumo de componentes, que considero que es importante en este campo. Si contamos con el hecho de que mi intención es la de seguir trabajando sobre este sistema para uso propio, mi valoración final del Proyecto es muy positiva.

9. REFERENCIAS

[1] HCS08 Microcontroller Technical Data, ***Motorola.***
MC09S8GB/GT Data Sheet V2.1 (Pg 141).

[2] Sensor Application Reference Design (SARD), ***Motorola.***
User's Guide (Pg 1-3).

[3] MC13192 Packet información, ***Motorola.***
MC13192/D Data Sheet Rev 2.4 06/2004 (Pg 1).

[4] HCS08 Microcontroller Technical Data, ***Motorola.***
MC09S8GB/GT Data Sheet V2.1 (Pg 84).

[5] MC1319x Marketing Overview, ***Motorola.***
RF/IF Product. Freescale Semiconductor, Inc (Pg 6).

[6] MC13192 Packet información, ***Motorola.***
MC13192/D Data Sheet Rev 2.4 06/2004 (Pg 4).

[7] MC13192 Reference Manual, ***Motorola.***
2.4 GHz Low Power Transceiver for 802.15.4 (Pg 16).

[8] MC13192 Reference Manual, ***Motorola.***
2.4 GHz Low Power Transceiver for 802.15.4 (Pg 13-15).

[9] Simple Media Access Controller (SMAC). ***Motorola.***
User's Guide (Pg 12).

[10] Simple Media Access Controller (SMAC). ***Motorola.***
User's Guide (Pg 17-35).

[11] "Tecnologías de telecomunicaciones", A.A. V.V.
Capítulo: Domótica (Pg 338).

[12] "Circuitos de optoelectrónica", R.M. Marston.
Capítulo: Sistema PIR detector de movimiento (Pg 139).

[13] "Circuitos de optoelectrónica", R.M. Marston.
Capítulo: Sistema PIR detector de movimiento (Pg 141).

[14] Curso Teoría de Sensores - ***Universidad pontificia Comillas de Madrid.***
http://www.iit.upcomillas.es/~alvaro/teaching/Clases/Robots/teoria (Pg 13).

[15] Curso Teoría de Sensores - ***Universidad pontificia Comillas de Madrid.***
http://www.iit.upcomillas.es/~alvaro/teaching/Clases/Robots/teoria (Pg 8).

[16] Detector PIR ISC-PPR1-W16 serie Professional. ***Bosch.***
http://www.boschsecurity.us/pdf/ES/ISC-PPR1-W16_Datasheet_es-ES_F2366673803.pdf

[17] "Circuitos integrados para tiristores y triacs", Marc Couëdic.
Capítulo: Circuitos de potencia y su control (Pg 21).

10. BIBLIOGRAFÍA

- "***SISTEMAS ELECTRÓNICOS DIGITALES***", Enrique Mandado. 1998. Edición MARCOMBO – ISBN: 84-267-1169-3

- "***INSTRUMENTACIÓN ELECTRÓNICA***", A.A. V.V. 1995. Edición MARCOMBO – ISBN: 84-267-1011-5

- "***CIRCUITOS DE OPTOELECTRÓNICA***", R.M Marston. 2000. Edición CEAC – ISBN: 84-329-8064-1

- "***TECNOLOGÍAS DE TELECOMUNICACIONES***", A.A. V.V. 2005. Edición CREACIONES COPYRIGHT – ISBN: 84-96300-08-0

- "***COMPONENTES ELECTRÓNICOS***", Francisco Ruiz Vassallo. 2004. Edición CEAC – ISBN: 84-329-8027-7.

- "***CIRCUITOS Y DISPOSITIVOS ELECTRÓNICOS",*** A.A. V.V. 1999. Edición EDICIONES UPC – ISBN: 84-8301-291-X

- "***CIRCUITOS INTEGRADOS PARA TIRISTORES Y TRIACS***", Marc Couëdic. 1999. Edición MARCOMBO – ISBN: 84-267-1197-9

- "***MC09S8GB/GT DATA SHEET***", HCS08 Microcontrollers Technical Data. Rev2.1 4/2004. 8/16 bits Products división - *MOTOROLA*, Inc.

- "***HCS08 Family Reference Manual***", HCS08 Technical Data. Rev1 6/2003. Volumen I – *MOTOROLA*, Inc.

- "***9S08GT60/GT32/GT16 FACT SHEET***", HCS08 Microcontrollers. Rev1 2004. Freescale Semiconductor, Inc – *MOTOROLA*.

- "***MC1319X TECHNICAL PRESENTATION***", PDF Document. Freescale Semiconductor, Inc – *MOTOROLA*.

- "***MC13192 REFERENCE MANUAL***", 2.4GHz Low Power transceiver for 802.15.4. Rev1 5/2004. Freescale Semiconductor, Inc – *MOTOROLA*.

- "***MC13192 PACKAGE INFORMATION***", Product Preview. Rev2.4 6/2004. Freescale Semiconductor, Inc – *MOTOROLA*.

- "***SENSOR APPLICATIONS REFERENCE DESIGN***", (SARD) User's Guide. Rev1.6 10/2006. Doc nº MC13192SARDUG. Freescale Semiconductor, Inc.

- "***802.15.4 PHY (****Physical Layer****)***", Setup and Porting Circuit Board (PCB). Rev1.1 2/2006. Doc nº AN2769. Freescale Semiconductor, Inc.

- "***SIMPLE MEDIA ACCESS CONTROL (****SMAC****)***", User's Guide. Rev1.4 10/2006. Doc nº SMACRM. Freescale Semiconductor, Inc.

- Página oficial de freescale – ***Motorola***. http://www.freescale.com/

- Teoría de Sensores de Ultrasonidos – *Universidad de Alcalá*. http://www.depeca.uah.es/docencia/ING-TELECO/sec/senso3.pdf

- Teoría de Sensores - *Universidad Pontificia Colmillas*. http://www.iit.upcomillas.es/pfc/

- The Engineering Search Engine – *Globalspec* http://search.globalspec.com

- Componentes – ***Panasonic.*** http://www.panasonic-electric-works.es/pewes/es/html/415.php

- Data Sheet Sensor ultrasonidos Maxsonar EZ1.
  http://www.pololu.com/products/misc/0726/

- Data Sheet Sensor PIR Parallax.
  http://www.parallax.com/dl/docs/prod/audiovis/PIRSensor-V1.1.pdf

- Data Sheet Sensor MP Motion – *Panasonic.*
  http://pewa.panasonic.com/pcsd/product/sens/select_motion.html

- Precios de componentes – *RS Online.*
  http://www.amidata.es/

- Precio de componentes – *Micropik.*
  http://www.micropik.com/

11. ANEXOS

11.1. CÓDIGO DISEÑADO PARA EL DISPOSITIVO DETECTOR.

```
/*****************************************************************
CÓDIGO DESTINADO AL CONTROL DE DETECCIÓN
*****************************************************************/

/*****************************************************************
*    Includes
*****************************************************************/
#include <hidef.h> /* for EnableInterrupts macro */
#include "pub_def.h"
#include "APP_SMAC_API.h"
#include "freescale_radio_hardware.h"
#include "remote_controller.h"
#include "bootloader user api.h"
#include "SCI.h"
#include "LCD.h"
#include "ascii_utilities.h"

/*****************************************************************
*    Data definitions
*****************************************************************/
  UINT8 gu8RTxMode; /* Current mode of the MC13192 XCVR */
  tTxPacket gsTxPacket;
  tRxPacket gsRxPacket;

  UINT8 gauTxDataBuffer[26];
  UINT8 gauRxDataBuffer[26];
  UINT8 gau8String[20];

/*****************************************************************
*    Main Program
*****************************************************************/
void main(void)

{

  /* Initialize variables. */
  UINT8 u8testini = 0;     // Testeo por si se ha colgado MCsensor.

  UINT8 u8Channel = 1;      // Define el valor del Canal.
  UINT8 u8Device = 0x01; // Identificador de la comunicación predefinida.

  UINT8 u8estadoluz = 0;    //Apagado=0; Encendido=1;
  UINT8 u8automanual = 0;   //Control del estado automatico-manual

  UINT8 u8detection = 0xDD; //Variable que marca la detección por PB0.
                      //Value==ED->Enable Detection
                      //Value==DD->Disable detection
  UINT8 u8contaTx = 0;
  UINT8 u8contafixTx = 0;
  UINT8 u8contareenvio = 0; // Contador de veces he reenviado un paquete
  UINT8 u8saltaled = 1;     // Apunta al led que se va encendiendo al reenvio

  UINT16 u16OldTime = 0;    // Variable para tiempos.
  UINT16 u16RTXOldTime = 0; // Variable para tiempos.
  UINT16 u16NewTime = 0;    // Variable para tiempos.
```

```
  UINT8 u8Countpack = 0;       // Contador para pasar packet RX a TX.
  UINT16 u16Count = 0;      // Contador estado espera si no es mi paquete.
  UINT8 u8ToCount = 0;      // Contador de 5 segundos al pulsar PB1.
  UINT8 u8AppStatus = 0;    // Apuntador de estado.
  UINT8 u8AppStatusRX = 0;  // Apuntador de estado Rx.
  UINT8 u8LightLed = 0x05;  // Variable para controlar encendido de leds.
  UINT8 u8GetLed = 0;

  UINT16 u16LagTime2 = 0;
  UINT16 u16LagTime1 = 0;

  /* Definición de los paquetes Rx y Tx*/
  gsTxPacket.u8DataLength = 0;
  gsTxPacket.pu8Data = &gauTxDataBuffer[0];
  gsRxPacket.u8DataLength = 0;
  gsRxPacket.pu8Data = &gauRxDataBuffer[0];
  gsRxPacket.u8MaxDataLength = 30;
  gsRxPacket.u8Status = 0;

  /* Inicialización Micro MC13192 y su sistema de transmisión */
  MCUInit();
  RadioInit();

  /* Direcciones de los Leds PTDDD salida */
  LED1DIR = 1;
  LED2DIR = 1;
  LED3DIR = 1;
  LED4DIR = 1;

  /* 8MHz CLKo, 4MHz bus clock */
 (void)MLMESetMC13192ClockRate(1);
  UseExternalClock();

/*************************************************************************
  To adjust output power call the MLME_MC13192_PA_output_adjust() with:
  MAX_POWER     (+3 to +5dBm)
  NOMINAL_POWER (0 dBm)
  MIN_POWER     ~(-16dBm)
  or somewhere custom ? (0-15, 11 (NOMINAL_POWER) being Default power)
*************************************************************************/
  (void)MLMEMC13192PAOutputAdjust(OUTPUT_POWER);    /*Set MAX power setting
*************************************************************************/

  /* Application init code. */
  SRTISC=SRTISC&~0x07; /* Disable wake up timer. */
  SPMSC2=SPMSC2&~0x03; /* Enable deep sleep mode stop3. */

  TPM1SC = 0x0E; /* Timer divide by 64. (16uS timebase for 4MHz bus clock). */

/* Pushbutton directions and pull-ups */
  PB0PU = 1;
  PB0DIR = 0;
  PB1PU = 1;
  PB1DIR = 0;
  PB2PU = 1;
  PB2DIR = 0;
  PB3PU = 1;
  PB3DIR = 0;

  MC13192_IRQ_IE_BIT = 1;   // Activar IRQ pin.
  EnableInterrupts;         // Macro para activar interrupciones.
```

```
/*****************************************************************/
/*   Indicación de leds para indicar que es el controlador     */
/*****************************************************************/
LED1 = 0;
LED2 = 0;
LED3 = 0;
LED4 = 0;
MCUDelay(LONGFLASHON);
LED1 = 1;
LED2 = 1;
LED3 = 1;
LED4 = 1;


/*****************************************************************/
/*   PETICIÓN INICIAL DE ESTADO DELS ACTUADOR                    */
/*****************************************************************/

while (gsRxPacket.u8Status != SUCCESS)
{
    gauTxDataBuffer[0] = 0xE1;      // Code bytes non-ZigBee
    gauTxDataBuffer[1] = u8Device;  // Identidad de la placa
    gauTxDataBuffer[2] = 0x88;      // Guarda led que se debe encender
    gauTxDataBuffer[4] = u8contaTx; // Variable abierta a nueva utilización.
    gauTxDataBuffer[3] = TOGGLECMD; // Guarda led que se debe encender
    gsTxPacket.u8DataLength = 5;    // Cantidad de paquetes

    /* TRANSMITIR PAQUETES*/
    (void)MCPSDataRequest(&gsTxPacket);

    /**************************************
    DAMOS TIEMPO PARA LA TRANSMISIÓN
    **************************************/

    u16NewTime = MCUReadTmr1 ();
    u16RTXOldTime = u16NewTime;
    while ((u16NewTime-u16RTXOldTime) < DWELLTIME)
    {
       u16NewTime = MCUReadTmr1 ();
    }

    /**************************************/
    /* PASAMOS AL MODO RX PARA RECIBIR RESPUESTA */
    (void)MLMERXEnableRequest(&gsRxPacket, RXTIMEOUT2);

    /* Esperamos a la recepción de datos */
    while (gu8RTxMode != IDLE_MODE)
    {
       LOW_POWER_WHILE(); // Conserve MCU power
    }

    /* Verificación de Datos */
    if ((gsRxPacket.u8Status == SUCCESS) && (gauRxDataBuffer[0] == 0xE1) &&
    (gauRxDataBuffer[1] == u8Device) && (gauRxDataBuffer[3] == ACKBACK))
    {
       if (gauRxDataBuffer[2] == 0xA0)
       { u8automanual = 0; u8estadoluz = 0; LED4=1;}
       if (gauRxDataBuffer[2] == 0xAF)
       { u8automanual = 0; u8estadoluz = 1; LED4=0;}
       if (gauRxDataBuffer[2] == 0xE0)
       { u8automanual = 1; u8estadoluz = 0; LED4=1;}
       if (gauRxDataBuffer[2] == 0xEF)
       { u8automanual = 1; u8estadoluz = 1; LED4=0; u8detection = 0xED;}
    }
```

```
 }

 /*************************************************************
 *    Main Loop, repeated now forever
 *************************************************************/
 for(;;)
 {
  /*Preselecciono el estado 1 para pasar del identificador*/
  u8AppStatus = 1;

  /* MC13192 Reset mode and MCU STOP3. */
  /* This is the stand-by low power mode. */
  /* MCU internal clock selected since we'll lose our external clock */
  UseMcuClock();
  MC13192ContReset(); /* Place the MC13192 into Reset mode */

  /* Restart to external clock appx. 10.8mS */
  MC13192Restart(); /* Bring the MC13192 into the desired Idle condition. */
  RadioInit();

  (void)MLMESetMC13192ClockRate(1); /* Back to 8MHz CLKo */
  UseExternalClock(); /* Use external accurate clock */

/************************ ESTADO DE RECEPCIÓN ************************
SI DESDE ACTUADOR HEMOS PASADO A MANUAL, ESTE MCSENSOR SE QUEDARÁ EN
STANDBY (WHILE) HASTA RECIBIR UNA SEÑAL*/
/*********************************************************************/

  if (u8automanual == 1)  //MANUAL
  {
   LED2 = 0;
   LED1 = 1;
   (void)MLMERXEnableRequest(&gsRxPacket,0x);

   while (gu8RTxMode != IDLE_MODE)
   {
    LOW_POWER_STOP(); // Conserve MCU power
   }
  }

  if (u8automanual == 0)  //AUTOMATICO
  {
   LED2 = 1;
   LED1 = 1;
   (void)MLMERXEnableRequest(&gsRxPacket,TIMEOUTRXMCA);

   while ((gu8RTxMode != IDLE_MODE))
   {
    LOW_POWER_STOP(); // Conserve MCU power
   }

   LED1 = 0;        //MARCADOR DE TIEMPO EN RX.
  }

/*********************************************************************/

  if ((gsRxPacket.u8Status == SUCCESS) && (gauRxDataBuffer[0] == 0xE1) &&
  (u8AppStatusRX == 0))
  {
     if (gauRxDataBuffer[1] == u8Device)
     {
        /* Enciende o apaga los Led segun informacion recibida */
        if ((gauRxDataBuffer[3] == TOGGLECMD) &&
        (gauRxDataBuffer[4] == u8contafixTx))
```

```
      {
         if (gauRxDataBuffer[2] == 0x00)
         {
            u8estadoluz = 0;     //Estado luz apagado
            u8detection = 0xDD;
            LED4 = 1;
         }
         if (gauRxDataBuffer[2] == 0xFF)
         {
            u8estadoluz = 1;     //Estado luz encendido
            u8detection = 0xED;
            LED4 = 0;
         }
         if (gauRxDataBuffer[2] == 0xAA)
         {
            u8automanual = 0;     //Estado automatico
         }
         if (gauRxDataBuffer[2] == 0xEE)
         {
            u8automanual = 1;     //Estado manual
         }

         /* Enciende el led con número enviado */
         if (gauRxDataBuffer[2] == 1)
         {
            LED1 ^=1;
         }
         if (gauRxDataBuffer[2] == 2)
         {
            LED2 ^=1;
         }
         if (gauRxDataBuffer[2] == 3)
         {
            LED3 ^=1;
         }
         if (gauRxDataBuffer[2] == 4)
         {
            LED4 ^=1;
         }
         u8AppStatusRX = 1;

      }
   }
}

/* PASAMOS AL TRASPASO DE INFORMACIÓN PARA MANDAR ACK */
if ((u8AppStatusRX == 1) || (gauRxDataBuffer[4] != u8contafixTx))
{
   u8AppStatusRX = 0;
   for (u8Countpack=0; u8Countpack<gsRxPacket.u8DataLength+1;u8Countpack++)
   {
      gauTxDataBuffer[u8Countpack] = gauRxDataBuffer[u8Countpack];
   }
   if (gauRxDataBuffer[4] != 0)
   {
      gauTxDataBuffer[4] = 0;
   }


   //PREGUNTA DE TEST 89. DONDE PREGUNTAMOS EN QUE ESTADO ESTÁ EL OTRO
   if (gauRxDataBuffer[2] == 0x89)
   {
      if ((u8automanual == 0) && (u8estadoluz == 0))
      {
```

```
            gauTxDataBuffer[2] = 0xA0;
          }
          if ((u8automanual == 0) && (u8estadoluz == 1))
          {
            gauTxDataBuffer[2] = 0xAF;
          }
          if ((u8automanual == 1) && (u8estadoluz == 0))
          {
            gauTxDataBuffer[2] = 0xE0;
          }
          if ((u8automanual == 1) && (u8estadoluz == 1))
          {
            gauTxDataBuffer[2] = 0xEF;
          }
        }

        /* PASAMOS A GUARDAR EN BUFFER EL ACK UNA VEZ VERIFICADO */
        if (gauRxDataBuffer[3] == TOGGLECMD)
        {
          gauTxDataBuffer[3] = ACKBACK;
        }
        gsTxPacket.u8DataLength = gsRxPacket.u8DataLength;
        u16NewTime = MCUReadTmr1 ();
        u16RTXOldTime = u16NewTime;

        /* Dwell for possible repeats. */
        while ((u16NewTime-u16RTXOldTime) < DWELLTIME) /* 46mS */
        {
          u16NewTime = MCUReadTmr1 ();
        }

        /* ENVIA EL PAQUETE */
        (void)MCPSDataRequest(&gsTxPacket);
        u16NewTime = MCUReadTmr1 ();
        u16RTXOldTime = u16NewTime;
        while ((u16NewTime-u16RTXOldTime) < DWELLTIME2) /* 44mS */
        {
          u16NewTime = MCUReadTmr1 ();
        }
        u16RTXOldTime = u16NewTime;
      }

/*****************************************************************************
AQUÍ SE ACABA LA PARTE DE RECEPCIÓN Y ENVIO DE REPORTE ACK.
*****************************************************************************/

      /*************************************************************
      *    Main While Loop - activated by pushbutton
      *************************************************************/

/*****************************************************************************/
      /* u8AppStatus=9 ESTADO PARA ENTRAR EN ESTADO ESPERA SOLTAR PULS */
      /* u8AppStatus=1 PULSAMOS PB0 PULSADOR1 PARA DEFINIR DETECCIÓN */
      /* u8AppStatus=2 PULSAMOS PB2 PARA ENVIAR Y PASAR A MODO RX. */
      /* u8AppStatus=4 SI LA RECEPCIÓN ES OK, do ACK action. */
/*****************************************************************************/

/*****************************************************************************
EN ESTA PARTE SE VERIFICA EL PULSADOR 1 (SIMULADOR DE DETECTOR) ESTÁ PULSADO
(DETECTA) O NO ESTÁ PULSADO (NO DETECTA)
*****************************************************************************/

       /* Pasamos a modo de espera Hibernate */
       UseMcuClock();
```

```
    MLMEHibernateRequest();
    /************************************/

      if (u8AppStatus == 1)
      {
         if ((PB0 == 0) && (u8detection != 0xED) &&
         (u8estadoluz == 0) && (u8automanual == 0))
         {
            u8detection = 0xED;
            u8LightLed = 0xD1;
            u8AppStatus = 2;      //pasamos directamente a enviar.
         }

         if ((PB0 == 1) && (u8detection != 0xDD) && (u8estadoluz == 1) &&
         (u8automanual == 0))
         {
            u8detection = 0xDD;
            u8LightLed = 0xD0;
            u8AppStatus = 2;      //pasamos directamente a enviar.
         }

         if (u8AppStatus == 9)
         {

            while ((u16NewTime-u16OldTime) < DEBOUNCE)
            {
               u16NewTime = MCUReadTmr1 ();
            }

            /* Seguridad al soltar los pulsadores */
            while ((PB0 == 0) || (PB1 == 0) || (PB2 == 0) || (PB3 == 0));
              u16OldTime = MCUReadTmr1 ();
              u16NewTime = u16OldTime;
            while ((u16NewTime-u16OldTime) < DEBOUNCE)
            {
               u16NewTime = MCUReadTmr1 ();
            }
              u16OldTime = MCUReadTmr1 ();
              u16NewTime = u16OldTime;
              u8AppStatus = 2;
              //break;
         }

      }
      u16NewTime = MCUReadTmr1 ();

    /* Volvemos a activar el MC13192 */
    MLMEWakeRequest();
    UseExternalClock();
    MCUDelay(DEBOUNCE);
    /*********************************/

/*****************************TRANSMISIÓN*********************************/
/*************************************************************************
AQUÍ EMPIEZA LA PARTE EN LA QUE PASAMOS A ESTADO2 Y EMPIEZA A PREPARAR
LOS DATOS A TRANSMITIR.
*************************************************************************/
/*************************************************************************/

      u8contareenvio = 0;
      u8saltaled = 7;

      /* Transaction selected. Transceive in process */
      while (u8AppStatus == 2)
```

```
    {

      /*************************************************************
      PREPARAMOS LOS DATOS PARA LA TRANSMISIÓN.
      *************************************************************/
      gauTxDataBuffer[0] = 0xE1; // Code bytes non-ZigBee
      gauTxDataBuffer[1] = u8Device; // Identidad de la placa
      gauTxDataBuffer[2] = u8LightLed; // Guarda led que se debe encender
      gauTxDataBuffer[4] = u8contaTx; // Guarda led que se debe encender
      gsTxPacket.u8DataLength = 5; // Cantidad de paquetes

      /* Guardamos un TOGGLECMD (0x11) */
      if (u8AppStatus == 2)
      {
         gauTxDataBuffer[3] = TOGGLECMD;
         u8AppStatus = 4; // Pasamos a estado espera ACK.
      }

      /* TRANSMITE EL PAQUETE 44ms*/
      (void)MCPSDataRequest(&gsTxPacket);

      /**************************************
      DAMOS TIEMPO PARA LA TRANSMISIÓN
      **************************************/
      u16NewTime = MCUReadTmr1 ();
      u16RTXOldTime = u16NewTime;
      while ((u16NewTime-u16RTXOldTime) < DWELLTIME)
      {
         u16NewTime = MCUReadTmr1 ();
      }
      /**************************************/

      /* PASAMOS AL MODO RX PARA RECIBIR RESPUESTA DEL DEVICE */
      (void)MLMERXEnableRequest(&gsRxPacket, RXTIMEOUT2);


/*****************************************************************************
AQUÍ EMPIEZA A VERIFICAR SI RECIBIMOS RESPUESTA DEL RECEPTOR CORRECTA
*****************************************************************************/

      if (u8AppStatus == 4)
      {
         /* ESPERA A QUE SE HAGA LA RECEPCIÓ */
         while (gu8RTxMode != IDLE_MODE)
         {
            LOW_POWER_WHILE(); /* Conserve MCU power */
         }

         /* Check for timeout. */
         if ((gsRxPacket.u8Status == SUCCESS)) /* Good packet received */
         {
            /* Check to see if the response is intended for this unit */
            if ((gauRxDataBuffer[0] == 0xE1) &&
            (gauRxDataBuffer[1] == u8Device) &&
            (gauRxDataBuffer[3] == ACKBACK))
            {
               MCUDelay(ESPERAREENVIO);
               LedDrive(0x05);

               /* ACTUALIZA EL ESTADO DE LAS LUCES, PARA MANTENER
               UN ESTADO DE TRABAJO CONJUNTO CON EL OTRO MC */
               if (gauRxDataBuffer[2] == 0xFF)
               {
                  LED4 = 0;
```

```
            u8estadoluz = 1;
          }
          if (gauRxDataBuffer[2] == 0x00)
          {
            LED4 = 1;
            u8estadoluz = 0;
          }
          u8AppStatus = 1;
        }

      }
      /*******************************************************
      SI NO SE HA RECIBIDO BIÉN O HAY UN TIMEOUT LO INDICA
      ENCENDIENDO UN LED CADA VEZ QUE LO INTENTA HASTA 4 VECES.
      SI NO LO CONSIGUE, PASA AL ESTA DE ESPERA.
      *******************************************************/
      else
      {
       LedDrive(u8saltaled);

       /* Informa de que a intentado reenviar 4 veces y cesa */
       if (u8contareenvio == 4)
       {
       LedDrive(0x05);
       MCUDelay(SHORTFLASHON);
       LedDrive(0x06);
       MCUDelay(SHORTFLASHON);
       LedDrive(0x05);
       MCUDelay(SHORTFLASHON);
       LedDrive(0x06);
       MCUDelay(SHORTFLASHON);
       LedDrive(0x05);
       MCUDelay(SHORTFLASHON);
       LedDrive(0x06);
       MCUDelay(REENVIOFALLIDO);
       LedDrive(0x05);
       u8saltaled = 1;
       u8AppStatus = 1;
       }

       /* Contador de veces reenvio */
       if (u8contareenvio != 4)
       {
       u8contareenvio = u8contareenvio+1;
       u8saltaled = u8saltaled+1;
       u8AppStatus = 2;
       }

      }


    }

/*****************************************************************************
AQUÍ SE ACABA LA PARTE DONDE ENVIA LOS DATOS Y VERIFICA SI HAN LLEGADO.
*****************************************************************************/

    } //TERMINA EL WHILE DE APPSTATUS=2

    /*****************************/
    /*CONTROL SEGURIDAD ESTADO LUZ*/
    /*****************************/
    if (u8estadoluz == 0)
    {
```

```
      LED4=1;
    }
    if (u8estadoluz == 1)
    {
      LED4=0;
    }
    /*******************************/
  }
}


/*****************************************************************
*    Interrupt:     MC13192 initiated interrupt handler
*    Parameters: none
*    Return:        none
*    Actions:    Disables the pushbutton interrupt
*****************************************************************/
interrupt void KBD_ISR()
{
 /* Disable the PB0 IRQ. Not needed until a new STOP. */
 KBI1SC_KBIE = 0;
 PB0IE =0;
 KBI1SC_KBACK = 1;
}

/*****************************************************************
*    Function:     LED driver
*    Parameters: state. 1-4 is light only that LED.
*   5 is turn all off. 6 is all on.
*    Return:        none
*****************************************************************/
void LedDrive (UINT8 state)
{
 switch (state)
 {
   case 0x01:
   LED1 = 0;
   LED2 = 1;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x02:
   LED1 = 1;
   LED2 = 0;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x03:
   LED1 = 1;
   LED2 = 1;
   LED3 = 0;
   LED4 = 1;
   break;
   case 0x04:
   LED1 = 1;
   LED2 = 1;
   LED3 = 1;
   LED4 = 0;
   break;
   case 0x05:
   LED1 = 1;
   LED2 = 1;
   LED3 = 1;
   LED4 = 1;
```

```
   break;
   case 0x06:
   LED1 = 0;
   LED2 = 0;
   LED3 = 0;
   LED4 = 0;
   break;

   case 0x07:
   LED1 = 0;
   LED2 = 1;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x08:
   LED1 = 0;
   LED2 = 0;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x09:
   LED1 = 0;
   LED2 = 0;
   LED3 = 0;
   LED4 = 1;
   break;
   case 0x0A:
   LED1 = 0;
   LED2 = 0;
   LED3 = 0;
   LED4 = 0;
   break;
  }
  return;
}

/*****************************************************************
*    Function:     Received data handler
*    Parameters: rx_packet_t
*****************************************************************/
void MCPSDataIndication(tRxPacket *gsRxPacket)
/* Just a direct return. Main loop will handle it. */
{
  (void)*gsRxPacket;
}

/*****************************************************************
*    Function:     MC13192 reset handler
*    Parameters: none
*****************************************************************/
void MLMEMC13192ResetIndication (void)
/* Not implemented. */
{
}

/*****************************************************************
*    Function:     Read MCU timer.
*    Parameters: none
*    Return:       16-bit timer value.
*****************************************************************/
UINT16 MCUReadTmr1(void)
{
  UINT16  w; /* w[0] is MSB, w[1] is LSB */
  ((UINT8*)&w)[0] = TPM1CNTH; /* MSB */
```

```
  ((UINT8*)&w)[1] = TPM1CNTL; /* LSB */
  return w;
}

/*************************************************************
*    Function:     Delay.
*    Parameters: Delay u16Count
*    Return:       none.
*************************************************************/
void MCUDelay (UINT16 delay_t)
{
  UINT16 u16MCUOldTime;
  UINT16 u16MCUNewTime;
  u16MCUOldTime = MCUReadTmr1();
  u16MCUNewTime = u16MCUOldTime;
  while ((u16MCUNewTime-u16MCUOldTime) < delay_t)
  {
    u16MCUNewTime = MCUReadTmr1();
  }
}
```

11.2. CÓDIGO DISEÑADO PARA EL DISPOSITIVO ACTUADOR.

```
/*****************************************************************
CÓDIGO DESTINADO AL CONTROL DE ACTUACIÓN SOBRE LA ILUMINACIÓN
*****************************************************************/

/*****************************************************************
*    Includes
*****************************************************************/
#include <hidef.h> /* for EnableInterrupts macro */
#include "pub_def.h"
#include "APP_SMAC_API.h"
#include "freescale_radio_hardware.h"
#include "remote_controller.h"
#include "bootloader user api.h"
#include "SCI.h"
#include "LCD.h"
#include "ascii_utilities.h"

/*****************************************************************
*    Data definitions
*****************************************************************/
  UINT8 gu8RTxMode; /* Current mode of the MC13192 XCVR */
  tTxPacket gsTxPacket;
  tRxPacket gsRxPacket;

  UINT8 gauTxDataBuffer[26];
  UINT8 gauRxDataBuffer[26];
  UINT8 gau8String[20];

/*****************************************************************
*    Main Program
*****************************************************************/
void main(void)

{

  /* Initialize variables. */
  UINT8 u8Channel = 1;            // Define el valor del Canal.
  UINT8 u8Device = 0x01;          // Identificador de la comunicación predefinida.

  UINT8 MODO = 0;                 //MODO=1 -> TX; MODO=2 -> TX & Cambia estado luz.
                                  //MODO=0 -> DISABLE
  UINT8 u8automanual = 0;         //Variable para controlar encendido manual/auto.
  UINT8 u8estadoluz = 0;          //Apagado=0; Encendido=1;

  UINT8 u8contaTx = 0;
  UINT8 u8contafixTx = 0;
  UINT8 u8contareenvio = 0;       // Contador de veces he reenviado un paquete
  UINT8 u8saltaled = 1;           // Apunta al led que se va encendiendo al reenvio

  UINT16 u16OldTime = 0;          // Variable para tiempos.
  UINT16 u16RTXOldTime = 0;       // Variable para tiempos.
  UINT16 u16NewTime = 0;          // Variable para tiempos.

  UINT16 u16OldTimepulsa = 0;     // Variable para tiempos.
  UINT16 u16NewTimepulsa = 0;     // Variable para tiempos.

  UINT8 u8Countpack = 0;          // Contador para pasar packet RX a TX.
  UINT16 u16Count = 0;            // Contador estado espera si no es mi paquete.
  UINT8 u8ToCount = 0;            // Contador de 5 segundos al pulsar PB1.
  UINT8 u8AppStatus = 0;          // Apuntador de estado.
  UINT8 u8AppStatuspulsa = 0;     //Controla el estado del pulsador pulsa-suelta.
  UINT8 u8AppStatuspulsa2 = 0;    //Variable para no mirar más pulsaciones.
```

```
 UINT8 u8AppStatusRX = 0;        // Apuntador de estado Rx.
 UINT8 u8LightLed = 0x05;        // Variable para controlar encendido de leds.
 UINT8 u8LightLedcon = 0;
 UINT8 u8GetLed = 0;

 UINT16 u16LagTime2 = 0;
 UINT16 u16LagTime1 = 0;

 /* Configuración de los paquetes para Tx y Rx */
 gsTxPacket.u8DataLength = 0;
 gsTxPacket.pu8Data = &gauTxDataBuffer[0];
 gsRxPacket.u8DataLength = 0;
 gsRxPacket.pu8Data = &gauRxDataBuffer[0];
 gsRxPacket.u8MaxDataLength = 30;
 gsRxPacket.u8Status = 0;

 /* Inicialización Micro MC13192 y su sistema de transmisión */
 MCUInit();
 RadioInit();

 /* Direcciones de los Leds PTDDD salida */
 LED1DIR = 1;
 LED2DIR = 1;
 LED3DIR = 1;
 LED4DIR = 1;

 /* 8MHz CLKo, 4MHz bus clock */
(void)MLMESetMC13192ClockRate(1);
UseExternalClock();

/*************************************************************************
 To adjust output power call the MLME_MC13192_PA_output_adjust() with:
 MAX_POWER     (+3 to +5dBm)
 NOMINAL_POWER (0 dBm)
 MIN_POWER     ~(-16dBm)
 or somewhere custom ? (0-15, 11 (NOMINAL_POWER) being Default power)
*************************************************************************/
 (void)MLMEMC13192PAOutputAdjust(OUTPUT_POWER);    /*Set MAX power setting
*************************************************************************/

 /* Application init code. */
 SRTISC=SRTISC&~0x07; /* Disable wake up timer. */
 SPMSC2=SPMSC2&~0x03; /* Enable deep sleep mode stop3. */
 TPM1SC = 0x0E; /* Timer divide by 64. (16uS timebase for 4MHz bus clock). */

 /* Pushbutton directions and pull-ups */
 PB0PU = 1;
 PB0DIR = 0;
 PB1PU = 1;
 PB1DIR = 0;
 PB2PU = 1;
 PB2DIR = 0;
 PB3PU = 1;
 PB3DIR = 0;

 MC13192_IRQ_IE_BIT = 1;   // Activar IRQ pin.
 EnableInterrupts;         // Macro para activar interrupciones.


 /*************************************************************/
 /*   Indicación de leds para indicar que es el controlador     */
 /*************************************************************/
 LED1 = 0;
 LED2 = 0;
```

```
LED3 = 0;
LED4 = 0;
MCUDelay(LONGFLASHON);
LED1 = 1;
LED2 = 1;
LED3 = 1;
LED4 = 1;

/*****************************************************************/
/*   PETICIÓN INICIAL DE ESTADO DELS ACTUADOR                    */
/*****************************************************************/

while (gsRxPacket.u8Status != SUCCESS)
{
    gauTxDataBuffer[0] = 0xE1;      // Code bytes non-ZigBee
    gauTxDataBuffer[1] = u8Device;  // Identidad de la placa
    gauTxDataBuffer[2] = 0x89;      // Guarda led que se debe encender
    gauTxDataBuffer[4] = u8contaTx; // Variable abierta a nueva utilización.
    gauTxDataBuffer[3] = TOGGLECMD; // Guarda led que se debe encender
    gsTxPacket.u8DataLength = 5;    // Cantidad de paquetes

    /* TRANSMITIR PAQUETES*/
    (void)MCPSDataRequest(&gsTxPacket);

    /**************************************
    DAMOS TIEMPO PARA LA TRANSMISIÓN
    **************************************/

    u16NewTime = MCUReadTmr1 ();
    u16RTXOldTime = u16NewTime;
    while ((u16NewTime-u16RTXOldTime) < DWELLTIME)
    {
       u16NewTime = MCUReadTmr1 ();
    }

    /**************************************/
    /* PASAMOS AL MODO RX PARA RECIBIR RESPUESTA */
    (void)MLMERXEnableRequest(&gsRxPacket, RXTIMEOUT2);

    /* Esperamos a la recepción de datos */
    while (gu8RTxMode != IDLE_MODE)
    {
       LOW_POWER_WHILE(); // Conserve MCU power
    }

    /* Verificación de Datos */
    if ((gsRxPacket.u8Status == SUCCESS) && (gauRxDataBuffer[0] == 0xE1)
    && (gauRxDataBuffer[1] == u8Device) && (gauRxDataBuffer[3] == ACKBACK))
    {
       if (gauRxDataBuffer[2] == 0xA0)
       { u8automanual = 0; u8estadoluz = 0; LED4=1; LED1=1;}
       if (gauRxDataBuffer[2] == 0xAF)
       { u8automanual = 0; u8estadoluz = 1; LED4=0; LED1=0;}
       if (gauRxDataBuffer[2] == 0xE0)
       { u8automanual = 1; u8estadoluz = 0; LED4=1; LED1=1;}
       if (gauRxDataBuffer[2] == 0xEF)
       { u8automanual = 1; u8estadoluz = 1; LED4=0; LED1=0;}
    }
}

/*************************************************************
*   Main Loop, repeated now forever
*************************************************************/
for(;;)
```

```
 {
  u8AppStatus = 1;/*Preselecciono el estado 1 para pasar del identificador*/
  u8AppStatuspulsa = 1;
  u8AppStatuspulsa2 = 1;

  /* MC13192 Reset mode and MCU STOP3. */
  /* This is the stand-by low power mode. */
  UseMcuClock();
  /* MCU internal clock selected since we'll lose our external clock */
  MC13192ContReset(); /* Place the MC13192 into Reset mode */

  /* Restart to external clock appx. 10.8mS */
  MC13192Restart(); /* Bring the MC13192 into the desired Idle condition. */
  RadioInit();

  (void)MLMESetMC13192ClockRate(1); /* Back to 8MHz CLKo */
  UseExternalClock(); /* Use external accurate clock */


/************************ ESTADO DE RECEPCIÓN ******************************/

  (void)MLMERXEnableRequest(&gsRxPacket, RXTXTIMEOUTPRUEBA);

  while ((gu8RTxMode != IDLE_MODE))
  {
   LOW_POWER_STOP(); // Conservar MCU power
  }
/*****************************************************************************/

  /* Verificación de datos */
  if ((gsRxPacket.u8Status == SUCCESS) && (gauRxDataBuffer[0] == 0xE1)
  && (u8AppStatusRX == 0))
  {
     if (gauRxDataBuffer[1] == u8Device)
     {
        /* Enciende o apaga los Led segun informacion recibida */
        if ((gauRxDataBuffer[3] == TOGGLECMD) &&
        (gauRxDataBuffer[4] == u8contafixTx))
        {

           /* El sensor manda que está detectando */
           if (gauRxDataBuffer[2] == 0xD1)
           {
              if (LED1 == 1)
              {
                LED1 = 0;
                LED4 = 0;
                u8estadoluz = 1;
              }
           }
           /* El sensor manda que no está detectando */
           if (gauRxDataBuffer[2] == 0xD0)
           {
              if (LED1 == 0)
              {
                LED1 = 1;
                LED4 = 1;
                u8estadoluz = 0;
              }
           }

           /* Enciende el led con número enviado */
           if (gauRxDataBuffer[2] == 1)
           {
```

```
            LED1 ^=1;
         }
         if (gauRxDataBuffer[2] == 2)
         {
            LED2 ^=1;
         }
         if (gauRxDataBuffer[2] == 3)
         {
            LED3 ^=1;
         }
         if (gauRxDataBuffer[2] == 4)
         {
            LED4 ^=1;
         }
         u8AppStatusRX = 1;

      }
   }
}

/* PASAMOS AL TRASPASO DE INFORMACIÓN PARA MANDAR ACK */
if ((u8AppStatusRX == 1) || (gauRxDataBuffer[4] != u8contafixTx))
{
   u8AppStatusRX = 0;
   for (u8Countpack=0; u8Countpack<gsRxPacket.u8DataLength+1;u8Countpack++)
   {
      gauTxDataBuffer[u8Countpack] = gauRxDataBuffer[u8Countpack];
   }
   if (gauRxDataBuffer[4] != 0)
   {
      gauTxDataBuffer[4] = 0;
   }


   //ACTUALIZACIÓN EN DETECCIÓN.
   if (gauRxDataBuffer[2] == 0xD1)
   {
      gauTxDataBuffer[2] = 0xFF;
   }
   if (gauRxDataBuffer[2] == 0xD0)
   {
      gauTxDataBuffer[2] = 0x00;
   }

   //PETICIÓN DE TEST. SEGÚN INFORMACIÓN ACTUALIZA EL ESTADO.
   if (gauRxDataBuffer[2] == 0x88)
   {
      if ((u8automanual == 0) && (u8estadoluz == 0))
      {
        gauTxDataBuffer[2] = 0xA0;
      }
      if ((u8automanual == 0) && (u8estadoluz == 1))
      {
        gauTxDataBuffer[2] = 0xAF;
      }
      if ((u8automanual == 1) && (u8estadoluz == 0))
      {
        gauTxDataBuffer[2] = 0xE0;
      }
      if ((u8automanual == 1) && (u8estadoluz == 1))
      {
        gauTxDataBuffer[2] = 0xEF;
      }
   }
```

```
      //PRE. DE CONFIRMACIÓN.
      if (gauRxDataBuffer[3] == TOGGLECMD)
      {
         gauTxDataBuffer[3] = ACKBACK;
      }

      gsTxPacket.u8DataLength = gsRxPacket.u8DataLength;
      u16NewTime = MCUReadTmr1 ();
      u16RTXOldTime = u16NewTime;

      /* Dwell for possible repeats. */
      while ((u16NewTime-u16RTXOldTime) < DWELLTIME) /* 46mS */
      {
         u16NewTime = MCUReadTmr1 ();
      }

      /* ENVIA EL PAQUETE */
      (void)MCPSDataRequest(&gsTxPacket);
      u16NewTime = MCUReadTmr1 ();
      u16RTXOldTime = u16NewTime;
      while ((u16NewTime-u16RTXOldTime) < DWELLTIME2) /* 44mS */
      {
         u16NewTime = MCUReadTmr1 ();
      }
      u16RTXOldTime = u16NewTime;
   }

/*****************************************************************************
AQUÍ SE ACABA LA PARTE DE RECEPCIÓN Y ENVIO DE REPORTE ACK.
*****************************************************************************/

   /*************************************************************
   *    Main While Loop - activated by pushbutton
   *************************************************************/

/*****************************************************************************/
   /* u8AppStatus=9 ESTADO PARA ENTRAR EN ESTADO ESPERA SOLTAR PULS */
   /* u8AppStatus=1 PULSAMOS PB1 PULSADOR2 PARA MOVERNOS POR LOS LEDS */
   /* u8AppStatus=2 PULSAMOS PB2 PARA ENVIAR Y PASAR A MODO RX. */
   /* u8AppStatus=4 SI LA RECEPCIÓN ES OK, do ACK action. */
/*****************************************************************************/

/*****************************************************************************
EN ESTA PARTE SE VERIFICA SI SE HA ACCIONADO UN PULSADOR, SI SE PULSA ENTRAMOS
EN UN IF QUE MIRARÁ CUAL Y ACTUA EN CONSECUENCIA.
*****************************************************************************/

    /* Pasamos a modo de espera Hibernate */
    UseMcuClock();
    MLMEHibernateRequest();
    /*************************************/

    u8LightLed = 0;
    u8LightLedcon = 6;
    u16OldTimepulsa = MCUReadTmr1();
    u16NewTimepulsa = u16OldTimepulsa;
    u8ToCount = 0;

    if ((PB0 == 0) || (PB1 == 0) || (PB2 == 0) || (PB3 == 0))
    {
    MODO = 1;
    u8AppStatuspulsa2 = 0;
    }
```

```
/* CONTADOR PARA LOS PULSADORES 3 Y 4 */
while ((u8ToCount < TONUMP) && (u8AppStatuspulsa2 != 1))
{
  if ((u16NewTimepulsa-u16OldTimepulsa) > TO)
  {
     u8ToCount++;
     u16OldTimepulsa = MCUReadTmr1 ();
     u16NewTimepulsa = u16OldTimepulsa;
  }
  u16NewTimepulsa = MCUReadTmr1 ();

  if (u8AppStatuspulsa == 1)
  {
     /* PASAREMOS A MANUAL O AUTOMÁTICO SEGÚN ESTADOS */
     if (PB0 == 0)
     {
        u8automanual ^= 1;
        u8AppStatuspulsa = 2;
        u8AppStatuspulsa2 = 1;
        MODO = 2;
        if (u8automanual == 0)
        {
          u8LightLed = 0xAA;
        }
        if (u8automanual == 1)
        {
          u8LightLed = 0xEE;
        }
        u16OldTime = MCUReadTmr1 ();
        u16NewTime = u16OldTime;
     }

     /* ENCENDER LA LUZ O APAGARLA SEGÚN ESTADO (Sólo en Automático) */
     if ((PB1 == 0) && (u8automanual == 1))
     {
        u8estadoluz ^= 1;
        u8AppStatuspulsa = 2;
        u8AppStatuspulsa2 = 1;
        MODO = 2;
        if (u8estadoluz == 0)
        {
          LED1=1;
          LED4=1;
          u8LightLed = 0x00;
        }
        if (u8estadoluz == 1)
        {
          LED1=0;
          LED4=0;
          u8LightLed = 0xFF;
        }
        u16OldTime = MCUReadTmr1 ();
        u16NewTime = u16OldTime;
     }

     /* Regulador de nivel pulsador3 (Sólo en Automático) */
     if (PB2 == 0)
     {
          u8LightLedcon = u8LightLedcon+1;
          u8AppStatuspulsa = 2;
          LedDrive(u8LightLedcon);
          if (u8LightLedcon == 0x07) { u8LightLed = 0x01; MODO = 0; }
          if (u8LightLedcon == 0x08) { u8LightLed = 0x02; MODO = 0; }
```

```
            if (u8LightLedcon == 0x09) { u8LightLed = 0x03; MODO = 0; }
            if (u8LightLedcon == 0x0A) { u8LightLed = 0x04; MODO = 0; }
            if (u8LightLedcon > 0x0A) { u8LightLedcon = 0x05;
                                  LedDrive(u8LightLedcon);
                                  u8LightLed = 0x00;
                                  u8LightLedcon = 0x06;
                                  MODO = 3; }
            u16OldTimepulsa = MCUReadTmr1 ();
            u16NewTimepulsa = u16OldTimepulsa;
            u8ToCount = 0;
            u16OldTime = MCUReadTmr1 ();
            u16NewTime = u16OldTime;
        }

        /* Regulador de nivel pulsador4 (Sólo en Automático) */
        if (PB3 == 0)
        {
            u8LightLedcon = u8LightLedcon+1;
            u8AppStatuspulsa = 2;
            if (u8LightLedcon == 0x07) { u8LightLed = 0x01; MODO = 0; }
            if (u8LightLedcon == 0x08) { u8LightLed = 0x02; MODO = 0; }
            if (u8LightLedcon == 0x09) { u8LightLed = 0x03; MODO = 0; }
            if (u8LightLedcon == 0x0A) { u8LightLed = 0x04; MODO = 0; }
            if (u8LightLedcon > 0x0A) { u8LightLed = 0x05;
                                  u8LightLedcon = 0x6;
                                  MODO = 3; }
            LedDrive(u8LightLed);
            u16OldTimepulsa = MCUReadTmr1 ();
            u16NewTimepulsa = u16OldTimepulsa;
            u8ToCount = 0;
            u16OldTime = MCUReadTmr1 ();
            u16NewTime = u16OldTime;
        }

        if ((u8AppStatuspulsa == 2))
        {

           while ((u16NewTime-u16OldTime) < DEBOUNCE)
           {
              u16NewTime = MCUReadTmr1 ();
           }

           /* Seguridad al soltar los pulsadores */
           while ((PB0 == 0) || (PB1 == 0) || (PB2 == 0) || (PB3 == 0));
             u16OldTime = MCUReadTmr1 ();
             u16NewTime = u16OldTime;
           while ((u16NewTime-u16OldTime) < DEBOUNCE)
           {
              u16NewTime = MCUReadTmr1 ();
           }

             u8AppStatus = 2;
             u8AppStatuspulsa = 1;
             //break;
        }

    }
}

/* Volvemos a activar el MC13192 */
MLMEWakeRequest();
UseExternalClock();
MCUDelay(DEBOUNCE);
/*********************************/
```

```
    if (MODO == 1){ LedDrive(0x05); MODO = 0; }
    if (MODO == 2){ MODO = 0; }
    //MODO3 --> Sig. hemos pulsado PB2 o PB3 con leds apagados,
    //en este caso no quiero enviar nada.
    if (MODO == 3){ u8AppStatus = 1; }
    u16NewTime = MCUReadTmr1 ();

/*****************************TRANSMISIÓN*********************************/
/*************************************************************************
AQUÍ EMPIEZA LA PARTE EN LA QUE PASAMOS A ESTADO2 Y EMPIEZA A PREPARAR
LOS DATOS A TRANSMITIR.
*************************************************************************/
/*************************************************************************/

     u8contareenvio = 0;
     u8saltaled = 7;

     /* Transaction selected. Transceive in process */
     while (u8AppStatus == 2)
     {

      /*********************************************************
      PREPARAMOS LOS DATOS PARA LA TRANSMISIÓN, OPERATION 1,5ms
      *********************************************************/
      gauTxDataBuffer[0] = 0xE1;          // Code bytes non-ZigBee
      gauTxDataBuffer[1] = u8Device;      // Identidad de la placa
      gauTxDataBuffer[2] = u8LightLed;    // Guarda led que se debe encender
      gauTxDataBuffer[4] = u8contaTx;     // Guarda led que se debe encender
      gsTxPacket.u8DataLength = 5;        // Cantidad de paquetes

      /* Guardamos un TOGGLECMD (0x11) */
      if (u8AppStatus == 2)
      {
         gauTxDataBuffer[3] = TOGGLECMD;
         u8AppStatus = 4; // Pasamos a estado espera ACK.
      }

      /* TRANSMITE EL PAQUETE 44ms*/
      (void)MCPSDataRequest(&gsTxPacket);

      /***************************************
      DAMOS TIEMPO PARA LA TRANSMISIÓN
      ***************************************/
      u16NewTime = MCUReadTmr1 ();
      u16RTXOldTime = u16NewTime;
      while ((u16NewTime-u16RTXOldTime) < DWELLTIME)
      {
         u16NewTime = MCUReadTmr1 ();
      }
      /***************************************/

      /* Total max RX time is 135-44ms */
      /* PASAMOS AL MODO RX PARA RECIBIR RESPUESTA DEL DEVICE */
      (void)MLMERXEnableRequest(&gsRxPacket, RXTIMEOUT2);



/*************************************************************************
AQUÍ EMPIEZA A VERIFICAR SI RECIBIMOS RESPUESTA DEL RECEPTOR CORRECTA
*************************************************************************/

      if (u8AppStatus == 4)
```

```
{
   /* ESPERA A QUE SE HAGA LA RECEPCIÓ */
   while (gu8RTxMode != IDLE_MODE)
   {
      LOW_POWER_STOP(); /* Conserve MCU power */
   }
   /* Check for timeout. */
   if ((gsRxPacket.u8Status == SUCCESS)) /* Good packet received */
   {
      /* Check to see if the response is intended for this unit */
      if ((gauRxDataBuffer[0] == 0xE1) &&
      (gauRxDataBuffer[1] == u8Device) &&
      (gauRxDataBuffer[3] == ACKBACK))
      {
         MCUDelay(ESPERAREENVIO);
         LedDrive(0x05);
         u8AppStatus = 1;
      }
   }

   /****************SISTEMA DE REENVIO**********************
   SI NO SE HA RECIBIDO BIÉN O HAY UN TIMEOUT LO INDICA
   ENCENDIENDO UN LED CADA VEZ QUE LO INTENTA HASTA 4 VECES.
   SI NO LO CONSIGUE, PASA AL ESTA DE ESPERA.
   ******************************************************/
   else
   {
    LedDrive(u8saltaled);

    /* Informa de que a intentado reenviar 4 veces y cesa */
    if (u8contareenvio == 4)
    {
    LedDrive(0x05);
    MCUDelay(SHORTFLASHON);
    LedDrive(0x06);
    MCUDelay(SHORTFLASHON);
    LedDrive(0x05);
    MCUDelay(SHORTFLASHON);
    LedDrive(0x06);
    MCUDelay(SHORTFLASHON);
    LedDrive(0x05);
    MCUDelay(SHORTFLASHON);
    LedDrive(0x06);
    MCUDelay(REENVIOFALLIDO);
    LedDrive(0x05);
    u8saltaled = 1;
    u8AppStatus = 1;
    }

    /* Contador de veces reenvio */
    if (u8contareenvio != 4)
    {
    u8contareenvio = u8contareenvio+1;
    u8saltaled = u8saltaled+1;
    u8AppStatus = 2;
    }
   }
}

/*************************************************************************
AQUÍ SE ACABA LA PARTE DONDE ENVIA LOS DATOS Y VERIFICA SI HAN LLEGADO.
*************************************************************************/

  } //FIN DEL WILE APPSTATUS=2
```

```
    /*****************************/
    /*CONTROL SEGURIDAD ESTADO LUZ*/
    /*****************************/
    if (u8estadoluz == 0)
    {
      LED1=1;
      LED4=1;
    }
    if (u8estadoluz == 1)
    {
      LED1=0;
      LED4=0;
    }
    /*****************************/
  }
}

/*************************************************************
*    Interrupt:     MC13192 initiated interrupt handler
*    Parameters: none
*    Return:        none
*    Actions:    Disables the pushbutton interrupt
*************************************************************/
interrupt void KBD_ISR()
{
 /* Disable the PB0 IRQ. Not needed until a new STOP. */
 KBI1SC_KBIE = 0;
 PB0IE =0;
 KBI1SC_KBACK = 1;
}

/*************************************************************
*    Function:     LED driver
*    Parameters: state. 1-4 is light only that LED.
*   5 is turn all off. 6 is all on.
*    Return:        none
*************************************************************/
void LedDrive (UINT8 state)
{
 switch (state)
 {
   case 0x01:
   LED1 = 0;
   LED2 = 1;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x02:
   LED1 = 1;
   LED2 = 0;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x03:
   LED1 = 1;
   LED2 = 1;
   LED3 = 0;
   LED4 = 1;
   break;
   case 0x04:
   LED1 = 1;
   LED2 = 1;
   LED3 = 1;
```

```
   LED4 = 0;
   break;
   case 0x05:
   LED1 = 1;
   LED2 = 1;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x06:
   LED1 = 0;
   LED2 = 0;
   LED3 = 0;
   LED4 = 0;
   break;

   case 0x07:
   LED1 = 0;
   LED2 = 1;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x08:
   LED1 = 0;
   LED2 = 0;
   LED3 = 1;
   LED4 = 1;
   break;
   case 0x09:
   LED1 = 0;
   LED2 = 0;
   LED3 = 0;
   LED4 = 1;
   break;
   case 0x0A:
   LED1 = 0;
   LED2 = 0;
   LED3 = 0;
   LED4 = 0;
   break;
  }
  return;
}

/*************************************************************
*    Function:     Received data handler
*    Parameters: rx_packet_t
*************************************************************/
void MCPSDataIndication(tRxPacket *gsRxPacket)
/* Just a direct return. Main loop will handle it. */
{
  (void)*gsRxPacket;
}

/*************************************************************
*    Function:     MC13192 reset handler
*    Parameters: none
*************************************************************/
void MLMEMC13192ResetIndication (void)
/* Not implemented. */
{
}

/*************************************************************
*    Function:     Read MCU timer.
```

```
*    Parameters: none
*    Return:        16-bit timer value.
*************************************************************/
UINT16 MCUReadTmr1(void)
{
 UINT16  w; /* w[0] is MSB, w[1] is LSB */
 ((UINT8*)&w)[0] = TPM1CNTH; /* MSB */
 ((UINT8*)&w)[1] = TPM1CNTL; /* LSB */
 return w;
}

/*************************************************************
*    Function:     Delay.
*    Parameters: Delay u16Count
*    Return:        none.
*************************************************************/
void MCUDelay (UINT16 delay_t)
{
 UINT16 u16MCUOldTime;
 UINT16 u16MCUNewTime;
 u16MCUOldTime = MCUReadTmr1();
 u16MCUNewTime = u16MCUOldTime;
 while ((u16MCUNewTime-u16MCUOldTime) < delay_t)
 {
  u16MCUNewTime = MCUReadTmr1();
 }
}
```

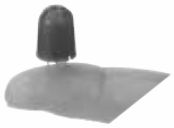
11.3. DATA SHEET DETECTOR.

MP Motion Sensor (AMN1)

Panasonic
ideas for life

MOTION SENSOR (PASSIVE INFRARED TYPE)

MP MOTION SENSOR 'NaPiOn'




Standard type


Slight motion detection type


Spot type

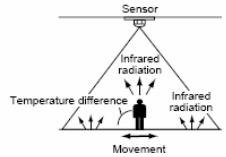

10 m detection type

What is passive infrared type?

This sensor detects changes in infrared radiation which occur when there is movement by a person (or object) which is different in temperature from the surroundings.

① As this sensor detects temperature differences, it is well suited to detecting the motion of people by their body temperature.

② Wide sensing area.



FEATURES

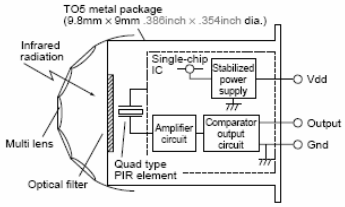
1. The world's smallest with a built-in amplifier
Extremely compact. Ideal for use in miniaturized devices.

2. Dual lens colors (white and black) are provided
With an ultrasmall design and dual lens colors (white and black), it is inconspicuous, allowing the user to select either white or black to match the equipment color. This provides greater flexibility in equipment design.

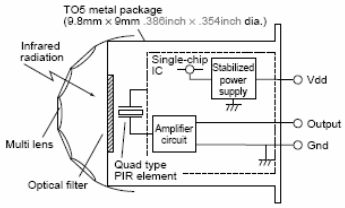
3. Both digital output and analog output (with adjustable sensitivity) are available.

4. Built-in amplifier for easy use
Has a built-in amplifier, and can be connected directly to a microcomputer.

• Block diagram of the digital output circuit



• Block diagram of the analog output circuit



NaPiOn web page URL:
http://www.napion.com/

5. Detects even slight motion of a person
With our sensor, even slight motions made by people will be detected easily.

• Fine motion detection capability within approximately 2 meters of sensor.
Standard type:
Detects movement of approximately 30cm 11.811inch.
Slight motion detection type:
Detects movement of approximately 20cm 7.874inch.

6. Noise withstanding capability
Circuitry is contained in a TO5 metal package, providing at least twice the noise withstanding capability as conventional type.

• Comparison example of noise withstanding capability

	Distance at which motion sensor is not affected by cellular phone noise
Conventional type	Min. 1 to 2m 3.281 to 6.562ft
MP Motion Sensor	Min. 1 to 2cm .394 to .787inch

APPLICATIONS

1. Home appliances
Useful for saving energy in air conditioner, television, personal computer, or ventilator and air purifier

2. Amusement machine market
Useful for saving energy and for automated guidance in theme parks and large video games

3. Equipment in service market
Useful for automated guidance, automated announcements and energy saving in vending machines, ATMs, etc.

4. Lighting market
Automated on/off controls, etc. for lamps, desk lamps, indoor lights, halls, stairway lights, etc.

ORDERING INFORMATION

AMN □ □ 1 1 □

Output
1: Digital output
2: Analog output

Detection performance
1: Standard detection type
2: Slight motion detection type
3: Spot detection type
4: 10m detection type

Feature
1: PC board mounting type

Operating voltage
1: 5V DC

Lens color
1: Black
2: White

PRODUCT TYPES

1. Digital output

Rated operating voltage	Detection performance	Ambient temperature	Lens color	Part No.	Packing quantity	
					Inner	Outer
3 to 6 V DC	Standard detection type	–20 to +60°C –4 to +140°F	Black	AMN11111	50 pcs.	1,000 pcs.
			White	AMN11112		
	Slight motion detection type		Black	AMN12111		
			White	AMN12112		
	Spot detection type		Black	AMN13111		
			White	AMN13112		
	10m detection type		Black	AMN14111		
			White	AMN14112		

2. Analog output

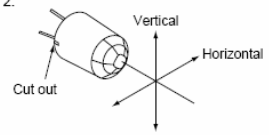
Rated operating voltage	Detection performance	Ambient temperature	Lens color	Part No.	Packing quantity	
					Inner	Outer
4.5 to 5.5 V DC	Standard detection type	–20 to +60°C –4 to +140°F	Black	AMN21111	50 pcs.	1,000 pcs.
			White	AMN21112		
	Slight motion detection type		Black	AMN22111		
			White	AMN22112		
	Spot detection type		Black	AMN23111		
			White	AMN23112		
	10m detection type		Black	AMN24111		
			White	AMN24112		

PERFORMANCE

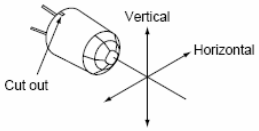
1. Detection performance

Items		Standard detection type	Slight motion detection type	Spot detection type	10m detection type	Conditions of objects to be detected
Rated detection distance*Remark 1		5m 16.404ft (Max.)	2m 6.562ft (Max.)	5m 16.404ft (Max.)	10m 32.808ft (Max.)	1. Detectable difference in temperature between the target and background for the spot type is more than 4°C 39.2°F. 2. Movement speed • Standard detection type/Spot detection type/ 10m detection type: 1.0 m/s • Slight motion detection type: 0.5 m/s 3. Detection object = human body (size is 700mm × 250mm 27.559inch × 9.843inch, but for the slight motion detection type the size is 200mm × 200mm 7.874inch × 7.874inch)
Detection range	Horizontal*Remark 2	100°	91°	38°	110°	
	Vertical*Remark 2	82°	91°	22°	93°	
	Detection zone*Remark 3	64 zones	104 zones	24 zones	80 zones	

2.



(Standard detection/ Slight motion detection type)



(Spot detection type/ 10 m detection type)

*Remarks 1. Depending on the difference in temperature between the background and detection target and the speed at which the target moves, these sensors may be capable of detection beyond the detection distances stated above. Nevertheless, they should be used within the prescribed detection distances. For further details, refer to the detection range diagram on page 7.

3. Regarding of detection zone, please refer to "DETECTION PERFORMANCE" on page 7.

2. Rating (Measuring condition: ambient temp. = 25°C 77°F) (Common to All types)

Items	Specified value	Remarks
Power supply voltage	–0.3 to 7 V DC	
Usable ambient temperature	–20 to 60°C –4 to +140°F	No freezing and condensing at low temperature.
Storage temperature	–20 to 70°C –4 to +158°F	

3. Electrical characteristics (Measuring condition: ambient temp. = 25°C 77°F; operating voltage = 5V) (Common to All types)

1) Digital output

Items			Symbol	Specified value	Measured conditions
Reted operating voltage		Minimum Typical Maximum	Vdd	3.0 V DC — 6.0 V DC	
Reted consumption current (Standby) *Remark		Typical Maximum	Iw	170 μA 300 μA	Iout = 0
Output (when detecting)	Current	Maximum	Iout	100 μA	Vout ≥ Vdd–0.5
	Voltage	Minimum Maximum	Vout	Vdd –5 Vdd (Same as operating voltage)	Open when not detecting
Circuit stability time		Typical Maximum	Twu	7 s 30 s	

Remark: The current which is consumed during detection consists of the standby consumed current plus the output current.

MP Motion Sensor (AMN1)

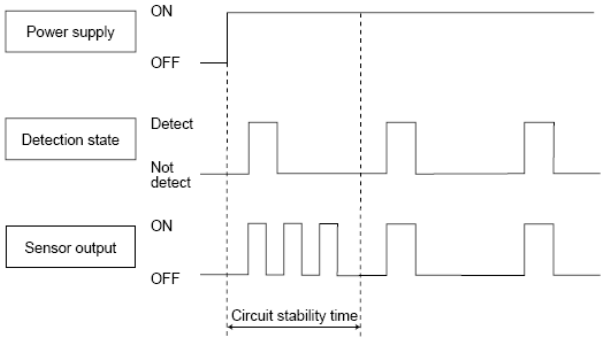
2) Analog output

Items		Symbol	Specified value	Measured conditions
Reted operating voltage	Minimum Maximum	Vdd	4.5 V DC 5.5 V DC	
Reted consumption current	Typical Maximum	Iw	0.17 mA 0.3 mA	Iout = 0
Output current	Maximum	Iout	50 µA	
Output voltage	Minimum Typical Maximum	Vout	0 V 2.5 V Vdd	
Output offset average voltage	Minimum Typical Maximum	Voff	2.3 V 2.5 V 2.7 V	Steady-state output voltage when not detecting
Steady-state noise	Typical Maximum	Vn	130 m Vp-p 300 m Vp-p	
Circuit stability time	Typical Maximum	Twu	7 s 45 s	

Note: To set to the same detection performance as the digital type, set the output voltage to the offset voltage (2.5V) ±0.45V (i.e. 2.95V or more and 2.05V or less).

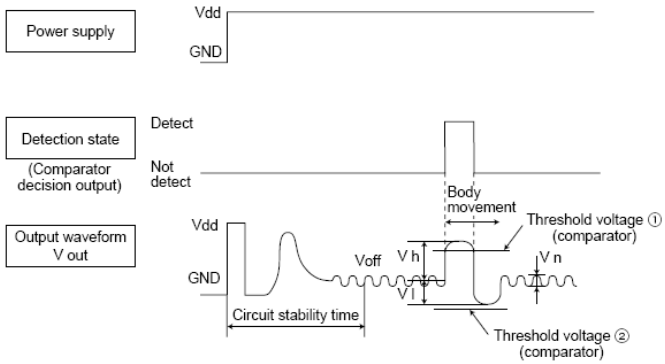
[Timing chart]

1) Digital output



Remark: Circuit stability time: 45s max. (45s max. for the 10m detection type)
While the circuitry is stabilizing after the power is turned on, the sensor output is not fixed in the "on" state or "off" state. This is true regardless of whether or not the sensor has detected anything.

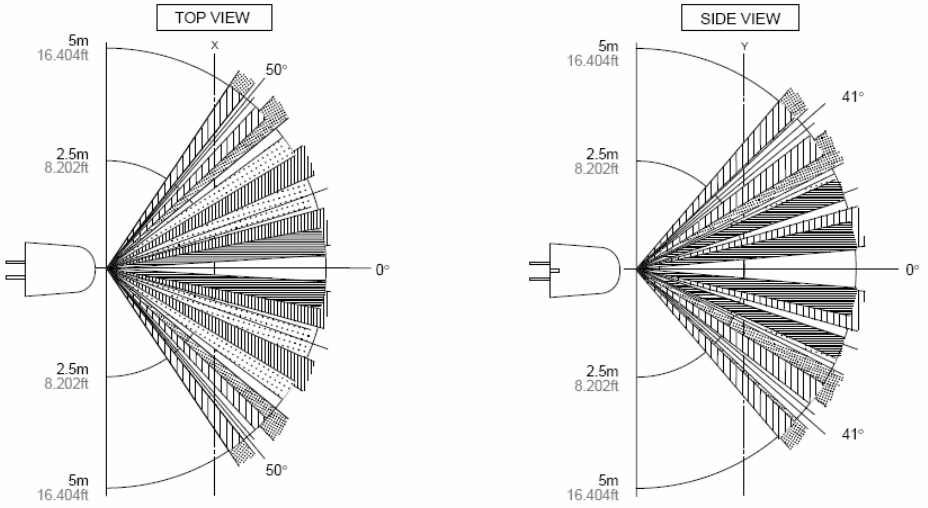
2) Analog output



Remark: Circuit stability time: 30s max.
While the circuitry is stabilizing after the power is turned on, the sensor output is not fixed in the "on" state or "off" state. This is true regardless of whether or not the sensor has detected anything.

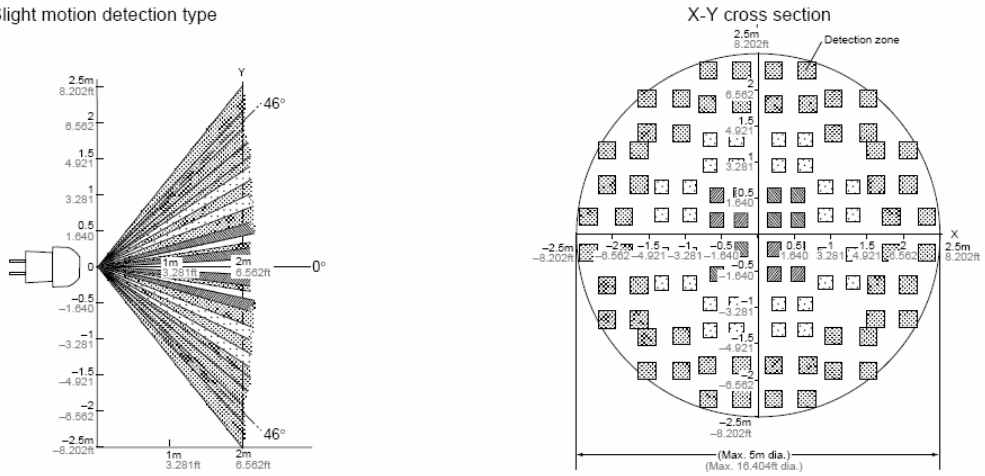
DETECTION PERFORMANCE

1. Standard detection type



Remarks: 1. The X-Y cross-sectional diagram shows the detection area.
2. The differences in the detection zone patterns are indicative of the projections of the 16 lenses with single focal point and with five optical axes. An object whose temperature differs from the background temperature and which crosses inside the detection zone will be detected.

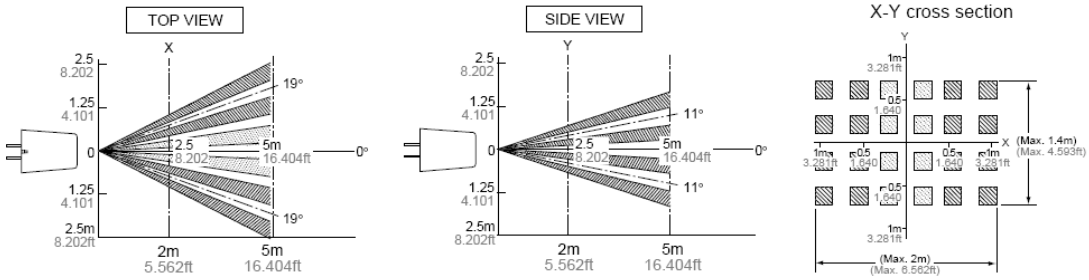
2. Slight motion detection type



Remarks: 1. The X-Y cross-sectional diagram shows the detection area.
2. The differences in the detection zone patterns are indicative of the projections of the 26 lenses with single focal point and with three optical axes. An object whose temperature differs from the background temperature and which crosses inside the detection zone will be detected.

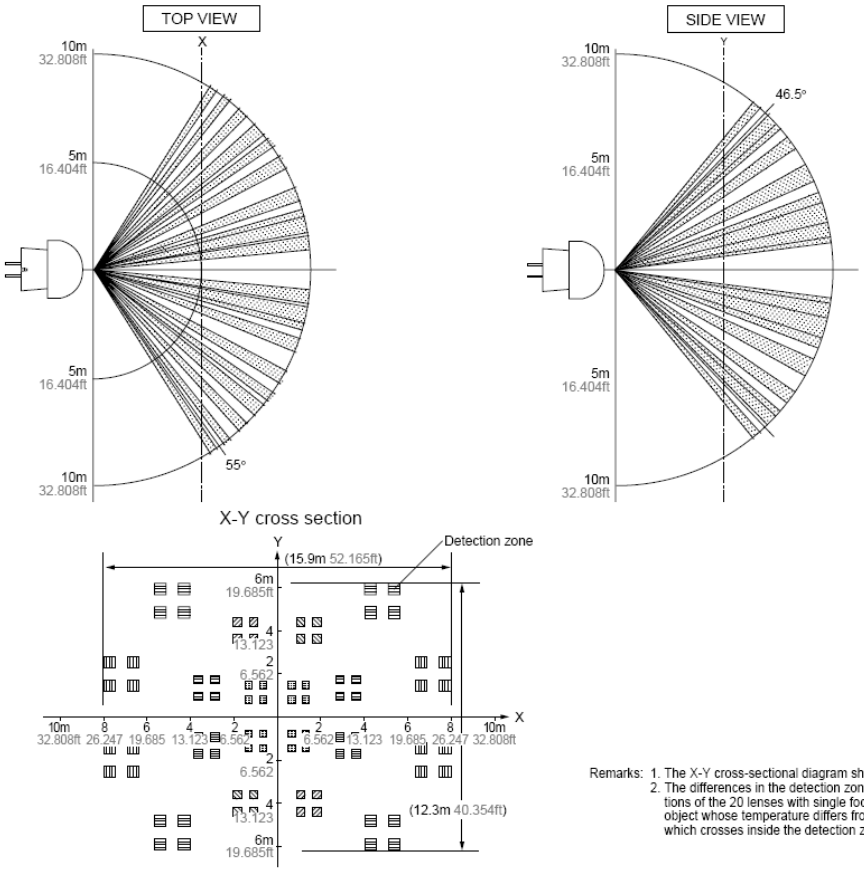
MP Motion Sensor (AMN1)

3. Spot detection type



Remarks: 1. The X-Y cross-sectional diagram shows the detection area.
2. The differences in the detection zone patterns are indicative of the projections of the 6 lenses with single focal point and with two optical axes. An object whose temperature differs from the background temperature and which crosses inside the detection zone will be detected.

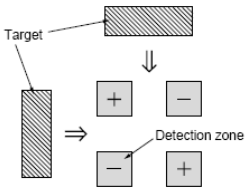
4. 10m detection type



Remarks: 1. The X-Y cross-sectional diagram shows the detection area.
2. The differences in the detection zone patterns are indicative of the projections of the 20 lenses with single focal point and with five optical axes. An object whose temperature differs from the background temperature and which crosses inside the detection zone will be detected.

5. Notes regarding the detection zone

The detection zone has the polarity shown in the diagram on the right. When targets enter both the + and – zones with the same timing, the signals are cancelled each other, thus in this case there is a possibility that the object cannot be detected at the maximum specified detection distance.

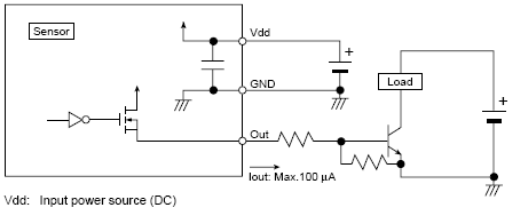


MP Motion Sensor (AMN1)

HOW TO USE

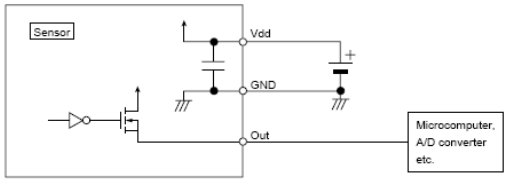
1. Wiring diagram

1) Digital output



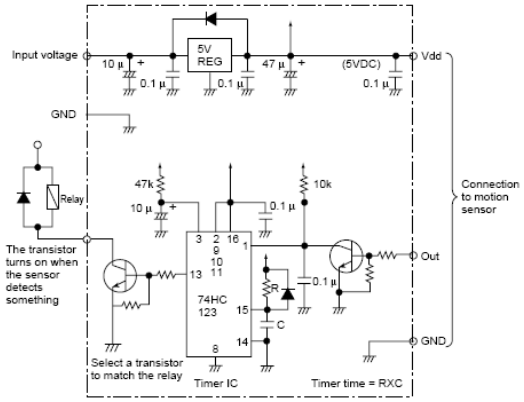
Vdd: Input power source (DC)
GND: GND
Out: Output (Comparator)

2) Analog output



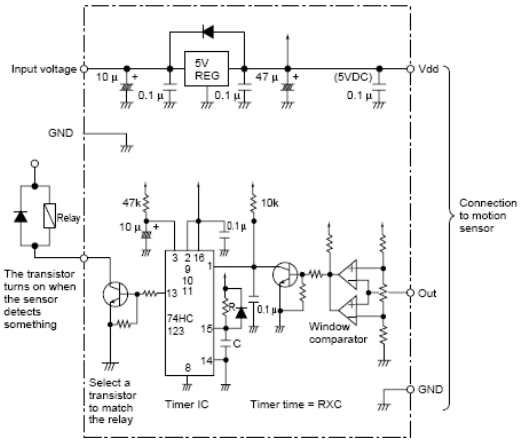
2. Timer circuit example

1) Digital output



Note: This is the reference circuit which drives the MP motion sensor. Install a noise filter for applications requiring enhanced detection reliability and noise withstanding capability.
Differences in the specifications of electronic components to which the units are connected sometimes affect their correct operation; please check the units' performance and reliability for each application.

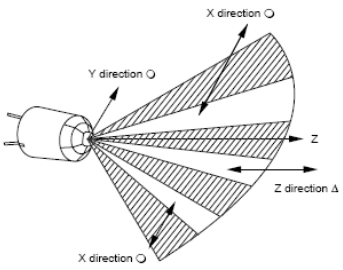
2) Analog output



Note: This circuit is a sample of a drive circuit for the MP Motion Sensor. Its noise resistance and long-term reliability are not considered or investigated.
To improve the detection reliability and noise resistance of the circuit, consider adding a noise filter.
Matsushita Electric Works, Ltd. accepts no responsibility for damages resulting from the use of this circuit.

3. Installation

Install the sensor so that people will be entering from the X or Y direction shown below. If persons approch the sensor from the Z direction, detection distance will be shortened.

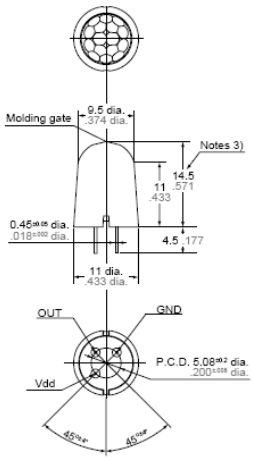


MP Motion Sensor (AMN1)

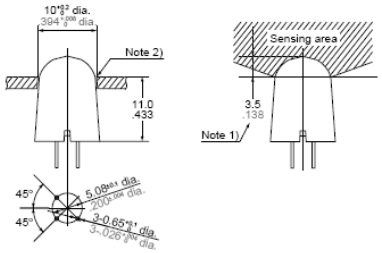
DIMENSIONS

mm inch General tolerance ±0.5 ±.020

1. Standard detection type

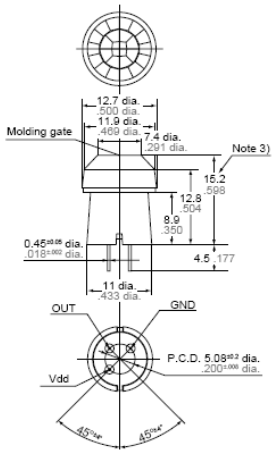


Recommended PC board pattern (BOTTOM VIEW)

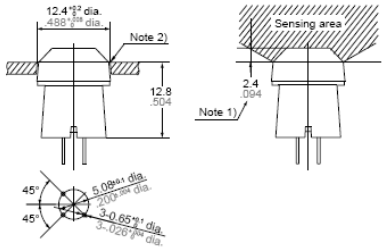


Notes: 1. In order to ensure proper detection, install it with the lens exposed at least 3.5mm .138inch.
2. As for panel mounting hole, tapering or making a large size hole should be done.
3. The height dimension does not include the remaining molding gate.

2. Slight motion detection type



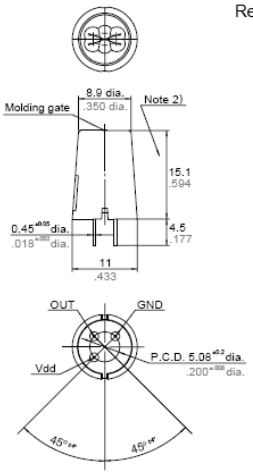
Recommended PC board pattern (BOTTOM VIEW)



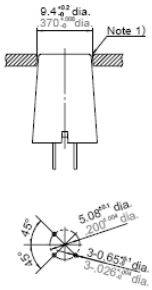
Notes: 1. In order to ensure proper detection, install it with the lens exposed at least 2.4mm .094inch.
2. As for panel mounting hole, tapering or making a large size hole should be done.
3. The height dimension does not include the remaining molding gate.

MP Motion Sensor (AMN)

3. Spot detection type

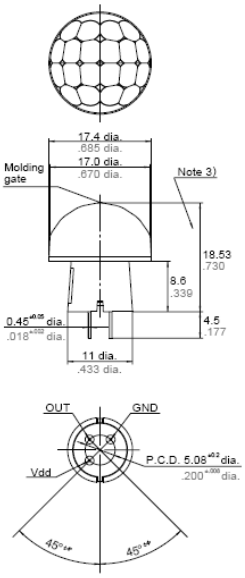


Recommended PC board pattern (BOTTOM VIEW)

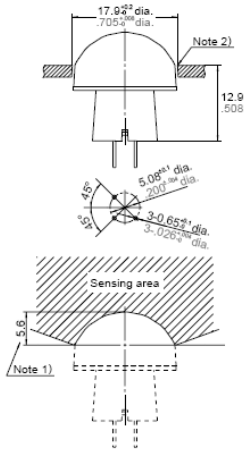


Notes: 1. As for panel mounting hole, tapering or making a large size hole should be done.
2. The height dimension does not include the remaining molding gate.

4. 10m detection type



Recommended PC board pattern (BOTTOM VIEW)



Notes:
1. In order to ensure proper detection, install it with the lens exposed at least 5.6mm .220inch.
2. As for panel mounting hole, tapering or making a large size hole should be done.
3. The height dimension does not include the remaining molding gate.

mm inch General tolerance ±0.5 ±.020

NOTES

1. Checkpoints relating to principle of operation

MP motion sensors are passive infrared sensors which detect changes in the infrared rays. They may fail to detect successfully if a heat source other than a human being is detected or if there are no temperature changes in or movement of a heat source. Care must generally be taken in the following cases. The performance and reliability of the sensors must be checked out under conditions of actual use.

<1> Cases where a heat source other than a human being is detected.

1) When a small animal enters the detection range.
2) When the sensor is directly exposed to sunlight, a vehicle's headlights, an incandescent light or some other source of far infrared rays.
3) When the temperature inside the detection range has changed suddenly due to the entry of cold or warm air from an air-conditioning or heating unit, water vapor from a humidifier, etc.

<2> Cases where it is difficult to detect the heat source

1) When an object made of glass, acrylic or other subject which far infrared rays have difficulty passing through is located between the sensor and what is to be detected.
2) When the heat source inside the detection range hardly moves or when it moves at high speed; for details on the movement speed, refer to the section on the performance ratings.

2. When the detection area becomes larger

When the difference between the ambient temperature and body temperature is large (more than 20°C 68°F), detection may occur in isolated areas outside the specified detection range.

3. Other handling cautions

1) Be careful not to allow dust or dirt to accumulate on the lens as this will adversely affect the detection sensitivity.
2) The lens is made of a soft material (polyethylene).
Avoid applying a load or impact since this will deform or scratch the lens, making proper operation impossible and causing a deterioration in its performance.
3) The sensor may be damaged if it is exposed to static with a voltage exceeding ±200V. Therefore, do not touch its terminals directly, and exercise adequate care in the handling of the sensor.
4) When the leads are to be soldered, solder them by hand for less than 3 seconds at a temperature of less than 350°C 662°F at the tip of the soldering iron.
Avoid using a solder bath since this will causing a deterioration in the sensor's performance.
5) Do not attempt to clean the sensor. Cleaning fluid may enter inside the lens area causing a deterioration in performance.
6) When using the sensors with cables, it is recommended that cables which are shielded and as short as possible be used in order to safeguard against the effects of noise.

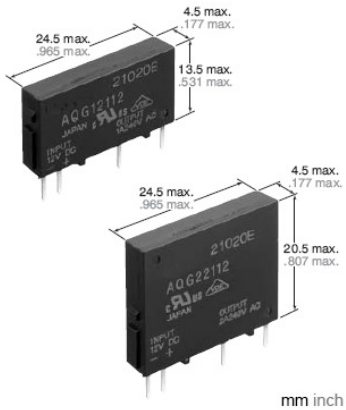
For the general precautions, refer to the Notes for Motion Sensors on page 24.

11.4. DATA SHEET ACTUADOR

Panasonic
ideas for life

1A and 2A space saving PC board terminal type

AQ-G RELAYS



mm inch

FEATURES

1. Space saving, Vertical size with a maximum thickness of 4.5 mm.
Mounting space has been reduced to 30% (compared to conventional SSR's) while meeting high density PC board mounting requirements.

2. 1A and 2A load types available

3. Zero-cross type and Non zero-cross type available

4. High dielectric strength of 3,000V AC
(between input and output)

5. Snubber circuit integrated
The snubber circuit is integrated to prevent malfunction caused by the rapid rise of voltage on the output side, such as inductive load and current.

TYPES

Type	Load current	Load voltage	Input voltage	Part No.
Zero-cross	1A	75 to 264 V AC	5 V DC	AQG12105
			12 V DC	AQG12112
			24 V DC	AQG12124
	2A	75 to 264 V AC	5 V DC	AQG22105
			12 V DC	AQG22112
			24 V DC	AQG22124
Non zero-cross	1A	75 to 264 V AC	5 V DC	AQG12205
			12 V DC	AQG12212
			24 V DC	AQG12224
	2A	75 to 264 V AC	5 V DC	AQG22205
			12 V DC	AQG22212
			24 V DC	AQG22224

TYPICAL APPLICATIONS

- **Manufacturing equipment**
  - NC machines
  - Injection molders
  - Robots
- **Air conditioners**
- **Computers**

ORDERING INFORMATION

Ex. AQG [1] [2] [1] [0] [5]

Load current	Load voltage	Type	Input voltage
1: 1 A 2: 2 A	2: 75 to 264 V AC	1: Zero-cross (3,000 V) 2: Non zero-cross (3,000 V)	05: 5 V DC 12: 12 V DC 24: 24 V DC

(Note) Standard packing: Carton 20 pcs., Case 500 pcs.

AQ-G

SPECIFICATIONS

1. Ratings (at 20°C 68°F, Input voltage ripple: 1% or less)
1) Zero-cross type

Item	Type	Part No.						Remarks
		AQG12105	AQG12112	AQG12124	AQG22105	AQG22112	AQG22124	
Input side	Input voltage	4 to 6 V DC	9.6 to 14.4 V DC	19.2 to 28.8 V DC	4 to 6 V DC	9.6 to 14.4 V DC	19.2 to 28.8 V DC	
	Input impedance	Approx. 0.3k Ω	Approx. 0.8k Ω	Approx. 1.6k Ω	Approx. 0.3k Ω	Approx. 0.8k Ω	Approx. 1.6k Ω	
	Drop-out voltage, min.	1 V						
	Reverse voltage	3 V						
Load side	Max. load current	1 A AC			2 A AC			
	Load voltage	75 to 264 V AC						
	Frequency	45 to 65 Hz						
	Non-repetitive surge current	8 A			30 A			In one cycle at 60 Hz
	Max. "OFF-state" leakage current	1.5 mA (applied 200 V)						
	Max. "ON-state" voltage drop	1.6 V						at Max. carrying current
	Min. load current	20 mA						

2) Non zero-cross type

Item	Type	Part No.						Remarks
		AQG12205	AQG12212	AQG12224	AQG22205	AQG22212	AQG22224	
Input side	Input voltage	4 to 6 V DC	9.6 to 14.4 V DC	19.2 to 28.8 V DC	4 to 6 V DC	9.6 to 14.4 V DC	19.2 to 28.8 V DC	
	Input impedance	Approx. 0.3k Ω	Approx. 0.8k Ω	Approx. 1.6k Ω	Approx. 0.3k Ω	Approx. 0.8k Ω	Approx. 1.6k Ω	
	Drop-out voltage, min.	1 V						
	Reverse voltage	3 V						
Load side	Max. load current	1 A AC			2 A AC			
	Load voltage	75 to 264 V AC						
	Frequency	45 to 65 Hz						
	Non-repetitive surge current	8 A			30 A			In one cycle at 60 Hz
	Max. "OFF-state" leakage current	1.5 mA (applied 200 V)						
	Max. "ON-state" voltage drop	1.6 V						at Max. carrying current
	Min. load current	20 mA						

2. Characteristics (at 20°C 68°F, Input voltage ripple: 1% or less)

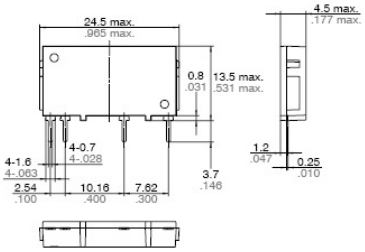
Item	Zero-cross type	Non zero-cross type	Remarks
Operate time max.	(1/2 cycle of voltage sine wave) + 1 ms	1 ms	
Release time, max.	(1/2 cycle of voltage sine wave) + 1 ms		
Insulation resistance, min.	10^9 Ω between input and output		Using 500 V DC megger
Breakdown voltage	3,000 Vrms between input and output		Initial for 1 min.
Vibration resistance	10 to 55 Hz double amplitude of 0.75 mm		X, Y, Z axes
Shock resistance	1,000 m/s²		X, Y, Z axes
Ambient temperature	–30°C to +80°C –22°F to +176°F		Non-condensing at low temperatures
Storage temperature	–30°C to +100°C –22°F to +212°F		
Operational method	Zero-cross (Turn-ON and Turn-OFF)	Non zero-cross turn ON, Zero-cross turn OFF	

AQ-G

DIMENSIONS

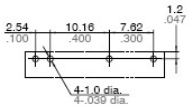
mm inch

1. 1A type



General tolerance: ±0.2 ±.008

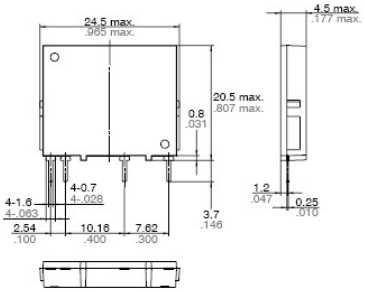
PC board pattern (Bottom view)



Tolerance: ±0.1 ±.004

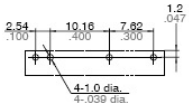
Schematic
AC type

2. 2A type



General tolerance: ±0.2 ±.008

PC board pattern (Bottom view)

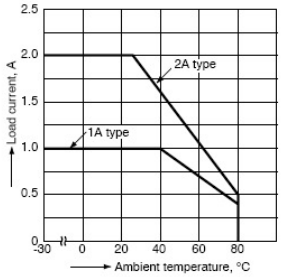


Tolerance: ±0.1 ±.004

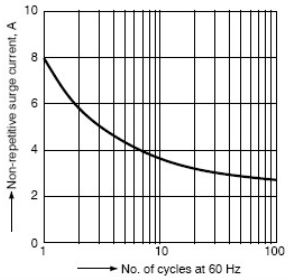
Schematic
AC type

REFERENCE DATA

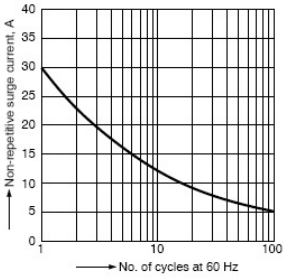
1. Load current vs. ambient temperature



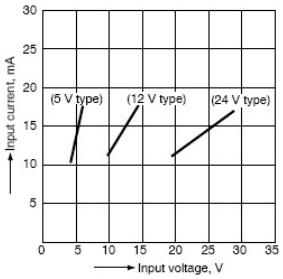
2.-(1) Non-repetitive surge current vs. carrying time (1A type)



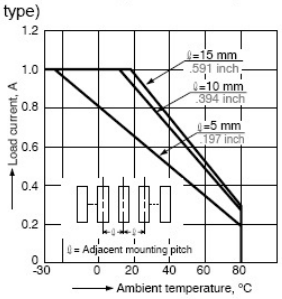
2.-(2) Non-repetitive surge current vs. carrying time (2A type)



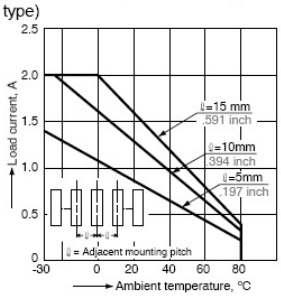
3. Input current vs. input voltage characteristics



4.-(1) Load current vs. ambient temperature characteristics for adjacent mounting (1A type)



4.-(2) Load current vs. ambient temperature characteristics for adjacent mounting (2A type)



Cautions for Use

12. RESUMEN

CASTELLANO:

El objetivo principal del proyecto ha sido diseñar un sistema de control automático de iluminación en un recinto cerrado, mediante la detección de presencia de una persona que entra dentro del radio de actuación del sistema. Utilizando dos placas con capacidad de comunicación inalámbrica se controlarán dos componentes: un sensor y un actuador. El sensor tiene la función de detectar la presencia de una persona mediante infrarrojos, mientras que el actuador nos proporciona la funcionalidad de un interruptor con el que poder actuar sobre la luz, encendiéndola o apagándola en función del estado de detección.

CATALÀ:

L'objectiu principal del projecte es dissenyar un sistema de control automàtic de il·luminació en un recinte tancat, mitjançant la detecció de presencia d'una persona que entra dins del radi d'actuació del sistema. Utilitzant dues plaques amb capacitat de comunicació inalàmbrica es controlaran dos components, un sensor i un actuador. El sensor té la funció de detectar la presencia d'una persona per mitjà d'infraroigs, mentre que l'actuador ens proporciona la funcionalitat d'un interruptor amb el que pogué actuar sobre la llum, encenent-la o apagant-la en funció de l'estat de detecció.

ENGLISH:

The main purpose of this project has been the design of an automatic light control system. The system controls lighting in an enclosed space through the detection of any person who enters the area where the system operates. With two wireless boards two components are controlled: a sensor and an actuator. The sensor's infrared rays detect the presence of a person, whereas the actuator acts as a regular switch. It switches lights on or off depending on the state of the sensor detection function.