Trabajo Fin de Grado

DISEÑO DE SISTEMA DOMÓTICO DE BAJO COSTE PARA LA GESTÓN ENERGÉTICA DE UNA VIVIENDA UNIFAMILIAR

Autor:

Julio Alejandro Marichal Lorenzo

Tutor:

Ricardo Mesa Cruz

Titulación:

Grado en Ingeniería Electrónica, Automática e Industrial

Escuela Superior de Ingeniería y Tecnología

Universidad de La Laguna

Julio 2017

Tenerife

Agradecimientos

Agradecer a mi tutor Ricardo Mesa por su dedicación y enseñanza en la elaboración de este proyecto, sin el cual no habría sido posible. Además, agradecer al profesor Alberto Hamilton por su ayuda en el campo de la programación

Igualmente, quiero agradecer a mi pareja, familia y amigos por su apoyo incondicional a lo largo del proyecto.

Resumen

El proyecto consiste en el desarrollo de un sistema domótico de bajo coste en una vivienda unifamiliar. El ámbito del proyecto se ha dirigido hacia el ahorro energético de la vivienda realizando determinadas acciones de control para este propósito.

En primer lugar, se han establecido las características de la vivienda que se pretenden automatizar siendo las luminarias, persianas, consumo eléctrico y climatización las magnitudes a controlar en este proyecto.

En segundo lugar, se ha realizado un proceso de elección de dispositivos electrónico para el desarrollo de los diferentes sistemas de control. Por una parte, se ha elegido la plataforma de código abierto Arduino para el procesamiento de la información que se obtenga de los sensores instalados en el interior de la vivienda. A su vez, esta elección se ha caracterizado por el bajo coste de los dispositivos cumpliendo uno de los principales objetivos del proyecto.

En tercer lugar, se ha optado por elegir un sistema inalámbrico para realizar la conexión entre los diferentes dispositivos eléctricos que se implemente en la vivienda, obteniendo una instalación más sencilla comparada con un sistema de comunicación cableado.

Por último, se ha realizado la programación necesaria para cada sistema de control en el entorno de desarrollo de Arduino escogiendo una programación orientada objetos en el lenguaje de programación C++ para su desarrollo.

Abstract

The project consists of the development of a low-cost home automation system in a single-family home. The project has been directed towards the energy saving of the home.

In the first place, the light, blinds, power consumption and climate will be the control system that we will install at home.

In the second place, we have made a process of choosing electronic devices to carry out control system. On the one hand, we have selected Arduino platform to process the information that we receive of sensors and actuators of the house.

Thirdly, we have chosen an inalambric system for connect differents electric devices instead of a wired communication system. This inalambric system will be the ZigBee protocol.

Finally, we have made the programming required for each control system in the Arduino choosing a Object Oriented Programming, in the programming language C++.

Memoria

Descriptiva

Índice:

Índice de figuras:

1. Alcance

El proyecto consistirá en el diseño de un sistema domótico controlando determinadas magnitudes de la vivienda, teniendo dos objetivos claros, la eficiencia energética y el confort del usuario en el hogar. Los parámetros que afectarán al ahorro energético principalmente serán la iluminación, el consumo eléctrico de los electrodomésticos y demás dispositivos eléctricos en las distintas zonas de la vivienda.

Por otra parte, el confort del usuario en el hogar se realizará primordialmente a partir de la climatización en los distintos sectores de la vivienda, obteniendo la temperatura óptima que desee el usuario en las diferentes zonas. Además, en el aspecto del confort del usuario, se realizará la automatización de las persianas y en conjunto con el apartado de ahorro energético, la automatización de la iluminación del hogar.

2. Objeto

El presente proyecto se basará en la aportación de una serie de dispositivos electrónicos al hogar para la recopilación de información en su interior. La gestión energética en la vivienda será el principal objetivo del proyecto, a partir de técnicas de control de determinados parámetros.

En la vivienda será necesario la instalación de sensores y actuadores para conseguir información de su interior. La implementación de estos dispositivos se realizará mediante una conexión inalámbrica, evitando cableado adicional de la instalación eléctrica y obteniendo una instalación más sencilla y confortable. Por lo tanto, será necesario incorporar a los sensores y actuadores un sistema de comunicación que permita el intercambio de información entre ellos.

Los dispositivos a instalar tendrán como referencia su bajo coste respecto a los recursos que aportan dichos dispositivos. De acuerdo con el apartado anterior, será necesarios dispositivos electrónicos que aporten información al hogar y un sistema de comunicación, optando por un protocolo de comunicación de bajo coste, como podría ser el protocolo de comunicación ZigBee.

3. Antecedentes

El diseño de este sistema domótico proviene de la necesidad de automatizar ciertas magnitudes del domicilio con el mínimo coste posible. La plataforma Arduino ha sido la principal motivación para la elaboración de este sistema debido a que se tratará de un sistema de código abierto. Esta característica aumenta los recursos de estos dispositivos y disminuye su precio en el mercado, incluyendo los sensores que son capaces de trabajar con este microcontrolador.

En conclusión, las propiedades de esta serie de instrumentos han habilitado la creación de este proyecto, teniendo una gran variedad de dispositivos compatibles con Arduino que aumenten la calidad del sistema que se pretende diseñar.

4. Referencias

4.2 Bibliografía

1. Protocolo ZigBee:
    - Domótica para ingenieros, JOSE Mª MAESTRE TORREBLANCA.
2. Módulos XBee:
    - https://mecatronicauaslp.wordpress.com/2013/07/04/xbee-parte-1-que-es-un-xbee-y-que-es-necesario/
    - http://xbee.cl/que-es-xbee/
    - http://fuenteabierta.teubi.co/2014/03/arduino-y-el-xbee-series-1-modo-api.html
    - https://aprendiendoarduino.wordpress.com/2016/11/16/zigbeexbee/
    - http://www.vintagecomputercables.com/datasheet/ds_xbeezbmodules.pdf

4.3 Programas de cálculo

4.3.1 LTSpice

El programa LTSpice consiste en un software de alto rendimiento desarrollado para la simulación de circuitos electrónicos y distribuido por la empresa "Linear Technology". En su interior se trabajará con esquemas eléctricos teniendo la posibilidad de simular circuitos electrónicos a partir de los componentes reales, obteniendo comportamientos aproximados a los circuitos diseñados con esta serie de elementos.

4.3.2 AutoCAD

AutoCAD es un software de diseño desarrollado para la creación de modelos en 2D y 3D, comercializado por la empresa Autodesk. Este software es reconocido a nivel mundial por su elevado nivel en el campo del dibujo digital, teniendo la posibilidad de diseñar planos de edificios o modelos 3D.

4.3.3 IDE Arduino

El IDE Arduino es una plataforma de código abierto desarrollada para la creación de programas en Arduino. Las instrucciones elaboradas para la programación del microcontrolador se compilarán en el interior de este entorno de programación, teniendo la posibilidad de introducir el código diseñado en la memoria interna del Arduino.

4.3.4 XCTU

XCTU consistirá en una aplicación libre proyectada por la empresa Digi Internacional. El objetivo de este programa será la programación de los módulos XBee, optando a modificar la configuración de esta serie de dispositivos.

4.3.5 KiCad

El programa KiCad es un entorno de desarrollo de circuitos eléctricos. La plataforma aporta un elevado número de componentes para la creación de esquemas eléctricos. Además, la plataforma incluye la posibilidad de diseñar placas de circuito impreso.

5. Definiciones y abreviaturas

Dirección PAN: dirección de área personal

Comando GT: domando guard time

ASCII: american standard for information interchange

SM: sleep mode
ST: time before sleeping

SP: cyclic sleep period

AES: Advance Encryption Standard

SKKE: Symmetric-Key Key Establishment

MIC: Message Integrity Code

CMM*: enhanced Counter with Cipher Block Chaining Message Authentication Code

LSB: Least Significant Byte

MSB: Most Significant Byte

6. Requisitos de diseño

En el presente apartado se presentará los requisitos de diseño establecidos por el cliente para la instalación del sistema domótico en la vivienda. En primer lugar, el principal objetivo ha sido la implementación de un sistema inalámbrico evitando conexiones mediante cables. El siguiente propósito será el control de las luminarias, persianas, temperaturas y consumo eléctrico.

A continuación, se especifica las magnitudes que se pretenden controlar en cada sector del hogar:

Entrada:

En el espacio de la entrada se pretenderá controlar las luminarias mediante un modo automático que trabaje en los momentos que se circula por esta área.

Cocina:

En la cocina se propone un sistema de luces automático entrando en funcionamiento cuando se accede al lugar. A su vez, también es necesario un sistema de persianas activándose automáticamente al comienzo y final del día, además de un dispositivo para modificar la posición de este elemento a lo largo de la jornada.

Sala:

En el siguiente sector se pretende instalar los tres tipos de control, luces, persianas y climatización. Por una parte, el control de luminarias y persianas requiere dos modos de funcionamiento, automático y manual. Además, la zona precisará de un sistema de climatización regulando la temperatura del área a través de un dispositivo manipulado por el usuario.

Después, se pretende conocer el consumo eléctrico de la televisión, desactivando este aparato cuando la zona no se encuentre habitada.

Balcón, pasillo y baños:

Las zonas comunes del hogar requerirán un control de luminarias automáticas, iluminando cada sector cuando son atravesadas por el usuario.

Habitaciones:

Por último, las habitaciones solicitarán los tres tipos de sistemas. Por una parte, los sistemas de iluminación y persianas implementarán funcionamientos automáticos y manuales, mientras que la regulación de temperatura requiere el control manual del usuario.

7. Análisis de soluciones

En el proyecto se tendrán en cuenta diferentes funciones que requiere el cliente, para el desarrollo del sistema domótico. En primer lugar, el usuario no pretende realizar una instalación cableada en él domicilio, solicitando una conexión inalámbrica entre los dispositivos. Por lo tanto, será necesario implementar un sistema de comunicación entre los sensores y actuadores que se pretenden implementar.

Por otra parte, el usuario ha requerido que se establezcan distintos tipos de control. Estos sistemas precisarán de diferentes elementos para su desarrollo. En cada uno se necesitarán una serie de sensores y actuadores, descritos en los apartados posteriores. Además, se precisa un instrumento capaz de leer los datos provenientes de los sensores para procesar esta información y realizar el control en los actuadores dependiendo de los valores obtenidos de los sensores.

En los apartados posteriores se describen las posibilidades para abordar las funciones requeridas por el usuario y a su vez la solución escogida para la elaboración del proyecto.

7.1 Sistemas de comunicación

El sistema de comunicación será el que realice la comunicación entre los distintos sensores y actuadores que se instalarán en el domicilio. La comunicación se realizará mediante una conexión inalámbrica, por lo tanto, se opta por soluciones que trabajen en el ámbito de la radiofrecuencia. A continuación, se presentan las soluciones propuestas.

En la elección de este sistema se han tenido en cuenta dos tipos de dispositivos. En primer lugar, un módulo de radiofrecuencia y, en segundo lugar, un dispositivo el cual tendrá instalado un protocolo de comunicación destacando las siguientes características:

- **Protocolo de comunicación:** el protocolo de comunicación consistirá en una serie de reglas o estándares establecidos para realizar una comunicación entre dos o más dispositivos.

- **Distancia de transmisión**: la distancia de transmisión será un apartado a destacar teniendo en cuenta que este sistema se instalará en un domicilio existiendo numerosos obstáculos, siendo casi nula la visión directa entre dispositivos.

- **Seguridad:** este aspecto deberá ser de gran importancia con la intención de tener un sistema seguro en el hogar y no ser vulnerable a posibles ataques externos.

- **Consumo eléctrico:** la prioridad del proyecto será instalar dispositivos de bajo coste, por lo tanto, se destacarán aquellos elementos que necesiten una baja potencia de transmisión disminuyendo su consumo.

- **Precio:** al diseñar un sistema de bajo coste la prioridad los dispositivos será el menor coste posible respecto a las características que aporte él instrumento.

7.1.1 Módulos de radiofrecuencia

Los módulos de radiofrecuencias estarán compuestos por dos dispositivos, transmisor y receptor, capaces de intercambiar información a distancia. Estos módulos trabajan en dos tipos de frecuencias las cuales serán de banda libre como son 433 MHz y 315 MHz, modulando el mensaje en ASK. En ambos dispositivos será necesario instalar una antena dependiendo del tipo se podrá conseguir diferentes distancias de transmisión. El rango de distancias de estos módulos será entre 20 y 200 metros dependiendo del medio donde se encuentren. Este aspecto se verá afectado directamente por la potencia de transmisión de ambos dispositivos, siendo este valor de 10 mW.

Hay dos posibilidades de adaptar esta pareja de dispositivos para realizar la transmisión de datos, una de ellas será realizar un circuito conectando el transmisor a un codificador y al receptor un decodificador. Otra posibilidad sería conectar a cada módulo un microcontrolador.

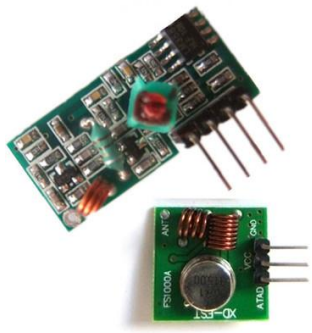


Figura 1.

La primera opción será crear un circuito con la pareja de módulos. Por una parte, el transmisor está conectado a un codificador el cual codifica una palabra digital de 12 bits, donde 8 bits será la dirección del codificador que se puede modificar en sus pines y los 4 bits restantes serán entradas digitales, por donde se enviará la información que se pretende transmitir. Por otra parte, el módulo receptor será el encargado de recibir el mensaje y transmitirlo al decodificador para obtener el mensaje original. La transmisión de datos requiere que ambos dispositivos posean la misma dirección.

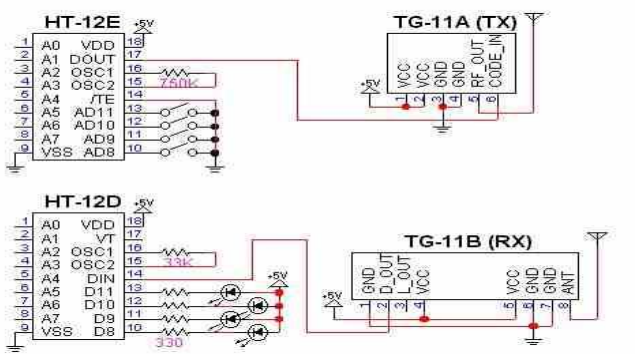


Figura 2.

La siguiente opción para el intercambio de información entre estos módulos será la conexión con un microcontrolador, el cual podrá ser Arduino. La transmisión de datos se realizará por medio de librerías diseñados para estos módulos.

En estos dispositivos será necesario el diseño de un protocolo de comunicación ya que no está implementado. A su vez, en el propio protocolo diseñado se tendría que crear un sistema de seguridad para la transmisión segura de los datos, empaquetando dicha información con un sistema de encriptación de datos.

Respecto al precio de ambos elementos, transmisor y receptor tendrán un precio extremadamente bajo, que estará en el rango de 2 a 5 euros.

7.1.2 Módulos XBee

Los módulos XBee serán dispositivos de radiofrecuencia diseñados para realizar una comunicación entre dos o más dispositivos siendo su rango de frecuencias la banda libre de 2,4 GHz. Esta serie de elementos tendrán implementados el protocolo de comunicación ZigBee. Este protocolo está diseñado para la comunicación entre dispositivos con una tasa de transmisión de datos baja. En su diseño, tendrá integrado el modelo de referencia OSI, el cual fue creado para la comunicación de datos haciendo que los datos necesiten atraviesen una serie de capas para que la comunicación se realice con éxito y de una forma segura. Además, el protocolo incluye un sistema de seguridad con una elevada fiabilidad al realizar métodos de encriptación y autentificación de datos.

La distancia de transmisión que indica el fabricante será entre 10 y 75 metros con obstáculos entre los dispositivos. A su vez, estos módulos tendrán sistema de bajo consumo como puede ser el "modo sleep" teniendo grandes resultados en cuanto ahorro de energía se refiere. El precio será bastante elevado respecto a los módulos de radiofrecuencia mencionados anteriormente, siendo de 20,63 euros.

7.1.3 Solución

En este caso, los módulos XBee serán los encargados de realizar las funciones de comunicación entre dispositivos. Esta decisión tendrá como prioridad la implementación del protocolo ZigBee en estos componentes teniendo un sistema bastante avanzado para la transmisión de datos. Además, garantiza la seguridad que tendrá integrada este protocolo al ser prácticamente invulnerable a ataques externos a la red.

También es destacado el bajo consumo de estos dispositivos al poseer sistemas de ahorro de energía a diferencia de los módulos de radiofrecuencia descritos siendo necesario crear técnicas de ahorro de energía en este caso. A su vez tendrá un inconveniente, que será el alto precio respecto a al sistema opuesto pero las elevadas

características que dispone este componente respecto al anterior se consideran prioritarias para su elección.

7.2 Microcontrolador

Los microcontroladores serán circuitos integrados con una capacidad de procesamiento de datos y ejecución de programas guardados en su memoria. Los principales módulos que hacen posible las funciones de los microcontroladores son la unidad central de procesamiento (CPU), memoria (RAM o ROM) y periféricos siendo las entradas y salidas del microcontrolador.

Un microcontrolador será un dispositivo programable teniendo la posibilidad de adaptarse a las necesidades del usuario debido a que ya tiene implementado el hardware necesario para realizar diferentes tareas. Por lo tanto, será necesario interactuar con esta serie de dispositivos mediante software, almacenando en su memoria programas diseñados por el programador con el propósito de que el microcontrolador realice una o varias tareas en un determinado tiempo. En el mercado existe una multitud de microcontroladores diferenciándose entre ellos la velocidad de procesamiento, la capacidad de la memoria o el número de periféricos que tienen implementados. Estas características se tendrán que elegir dependiendo del uso que se quiera dar a este circuito integrado.

En este proyecto se necesitará un dispositivo capaz de leer datos de una serie de sensores, procesar esa información y ejecutar una serie de acciones sobre los actuadores que se implementen. Las opciones que se han encontrado para realizar esta tarea ha sido el microcontrolador Arduino y la Raspberry Pi. En la elección se han precisado una serie de características mostradas a continuación.

- **La memoria Flash:** esta memoria será la encargada de almacenar los programas que se instalen en el microcontrolador teniendo una capacidad limitada. Por lo tanto, se necesitará una memoria acorde al tamaño del programa que se quiera instalar en este dispositivo. El programa será del orden inferior a los 32 KB.

- **Velocidad de procesamiento:** la velocidad de procesamiento se verá reflejada en el reloj principal que contenga cada uno de estos dispositivos y los núcleos que tengan implementados en su procesador.

- **Consumo:** el consumo eléctrico que se genera de estos circuitos integrados será relevante, debido a que unos de los objetivos del proyecto será el bajo coste de los dispositivos.

- **Facilidad de implementación:** en la elección se tendrá en cuenta la necesidad de instalar sistemas en su interior para utilizar esta serie de dispositivos para el funcionamiento que se le pretende dar.

- **Periféricos:** los periféricos serán las unidades externas que se le quieran añadir al microcontrolador. En este proyecto los periféricos a instalar serán los sensores y actuadores necesarios para la instalación.

7.2.1 Raspberry pi

La Raspberry no estará compuesta por un microcontrolador debido a que cuando se refiere a este componente se situarán todos los dispositivos necesarios integrados en un único chip preparado para realizar las funciones mencionadas en el apartado anterior. Mientras, la Raspberry Pi constará de un microprocesador tendiendo sus periféricos implementados en el exterior de este chip. Esta característica aporta una mayor potencia al dispositivo.

En el mercado existen diferentes tipos de Raspberry Pi en este caso se ha optado por elegir la Raspberry Pi 3 para realizar la comparación con el microcontrolador Arduino. Las Raspberry Pi especificada tendrá un microprocesador ARM - córtex A53 de cuatro núcleos y un reloj de 1,2 GHz obteniendo una velocidad de procesamiento bastante elevada debido a su reloj y al número de núcleos que tiene el microprocesador. La memoria estará compuesta por una tarjeta micro SD teniendo capacidades tan elevadas como podrían ser de 16 o de 32 GB. En cuanto al consumo, se encontrará este valor en el rango de 230 mW y 1,2 W cuando se encuentra en reposo alcanzando un consumo de 1,8 W cuando está a pleno rendimiento.

Posteriormente, se destaca la necesidad de instalar un sistema operativo para su correcto funcionamiento. Después, respecto a los periféricos que ofrece este dispositivo tendremos una cantidad bastante amplia de pines, 40 concretamente, incluyendo pines de alimentación teniendo la posibilidad de instalar una amplia cantidad de sensores y actuadores en esta serie de pines. Además, tendrá una entrada de video con la intención de conectar una cámara u otros periféricos de gran utilidad para proyectos específicos.

7.2.2 Arduino

El Arduino a diferencia de la Raspberry Pi si tiene instalado un microcontrolador de la marca Atmel, existiendo diferentes modelos en cada uno de los Arduino que se comercializan.

La velocidad de procesamiento respecto a la Raspberry será significativamente menor al tener un reloj de tan solo 16 MHz y un núcleo en los microprocesadores que llevan instalados internamente los microcontroladores. Además, similar con la velocidad de procesamiento la memoria será menor a la Raspberry Pi teniendo una memoria de 32 KB en su microcontrolador, considerando la posibilidad de aumentarla con tarjetas SD instalando una shield a la placa Arduino.

A diferencia de las características anteriores el consumo eléctrico de este circuito integrado será representativamente mejor debido a que reduce su consumo en menos de la mitad, obteniendo valores de 46 mA en el Arduino uno y un valor extremadamente bajo de 15 mA en el Arduino nano. En cuanto a los periféricos, dependiendo del Arduino se tendrán diferentes cantidades de pines. En el Arduino uno, por ejemplo, se tendrán 14 entradas y salidas digitales y 6 entradas analógicas. Además, en este microcontrolador no será necesaria la instalación de ningún sistema operativo para su funcionamiento instalando los programas en el interior de la memoria mediante el programa IDE Arduino.

7.2.3 Solución

La solución por la que se ha optado para la realización de este proyecto ha sido el microcontrolador Arduino. La característica principal por la que se ha elegido este dispositivo ha sido su bajo consumo, debido a que la intención del proyecto será instalarlo

alimentado con baterías necesitando el menor consumo posible. También la memoria que ofrece el Arduino será suficiente para el programa que se desea instalar en él, teniendo la posibilidad de ampliarla con una tarjeta SD. Respecto a la velocidad de procesamiento y el número de periféricos, la Raspberry será significativamente mejor pero estas características en Arduino serán suficientes para la tarea que se desea realizar.

En el presente proyecto se empleará un Arduino nano. Este modelo de Arduino se caracterizará por sus dimensiones (18x45 mm) y su bajo consumo eléctrico respecto a los demás modelos (15 mA). El objetivo del proyecto será el bajo consumo de sus dispositivos y a su vez la instalación de este circuito en un módulo remoto con las mínimas dimensiones posibles. Por lo tanto, el Arduino nano debido a las características ofrecidas será el modelo adecuado para este proyecto.

7.3 Sensores

Los sensores otorgan la posibilidad de transformar una magnitud física en una magnitud eléctrica habitualmente en voltaje o intensidad. A partir de diferentes sensores se abarcarán todas las características necesarias para realizar el control deseado en el hogar. Las magnitudes que se deberán medir en las diferentes zonas de la casa serán temperatura, movimiento, luz y consumo eléctrico. Por lo tanto, los sensores que se utilizarán en este proyecto dependiendo cada sistema de control serán:

- Sistema de luz:
    - Sensores de movimiento
    - Sensor de luz
- Sistema de temperatura:
    - Sensor de temperatura
- Sistema de consumo eléctrico:
    - Sensor de consumo eléctrico

- Dispositivos que interaccionan con los sistemas

    - Sensor capacitivo
    - Potenciómetro

7.3.1 Sensores de movimiento

El sensor de movimiento será capaz de detectar personas en su zona de trabajo. Existen diferentes tipos y distintas tecnologías para el desarrollo de estos sensores. En primer lugar, una de las tecnologías más desarrolladas es a partir de la radiación infrarroja. Todos los cuerpos emiten una cantidad de energía infrarroja determinada aumentando cuando su temperatura es mayor, teniendo como condición que dicho cuerpo debe de estar a una temperatura superior a los 273, 15 Kelvin (0 ºC). Esta radiación incide sobre un dispositivo piezoeléctrico capaz de detectar cambios de energía en la banda de infrarrojos indicando que existe un cuerpo en movimiento.

Por otra parte, se utilizan técnicas de ultrasonidos enviando ondas acústicas que rebotan en los cuerpos, a los cuales son enviadas determinando el tiempo que tarda la onda en volver. También existen sensores que unen ambas tecnologías transmitiendo ondas de ultrasonidos que inciden sobre los cuerpos que se encuentran en su zona de trabajo, produciendo que el sensor piezoeléctrico detecte cambios de energía infrarroja generada por las ondas de ultrasonido.

Este conjunto de sensores habitualmente tendrá instalados una lente Fresnel. La función de esta lente será aumentar su zona de trabajo debido a que sin ella sólo se detectarían las señales que fueran dirigidas directamente al dispositivo piezoeléctrico. Esta pieza reflejará todas las señales infrarrojas que van dirigidas al dispositivo a un punto concreto que será el elemento piezoeléctrico pudiendo alcanzar rangos de hasta 360º.

En la elección del sensor de movimiento se tendrán en cuenta las siguientes características:

- **Distancia de detección:** la distancia máxima que posee el sensor para detectar los cambios de temperatura de un cuerpo.

- **Alimentación:** la alimentación necesaria que tendrá que tener el sensor para su adecuado funcionamiento. En el proyecto se pretende instalar este sensor en un módulo inalámbrico, junto a un microcontrolador. Por lo tanto, es necesario que la alimentación del sensor sea de 5 voltios como la del microcontrolador que se utilizará en el proyecto.

- **Precio:** el precio será una característica importante a la hora de realizar la elección de este sensor debido a que la intención del proyecto es instalar dispositivos de bajo coste.

7.3.1.1 Sensor DT8012F4

El dispositivo DT8012F4 será un sensor de movimiento de doble tecnología uniendo las técnicas de infrarrojo y ultrasonidos, teniendo una gran fiabilidad para la detección de cuerpos. La alimentación del sensor estará en el rango de 9 a 15 voltios. Posteriormente, el sensor tendrá una serie de filtros que lo harán inmune a la luz blanca y además a la luz fluorescente. En cuanto a la distancia de detección se podrán detectar objetos que se encuentren a 12 metros en línea recta al sensor y hasta 8 metros en los extremos de su campo de visión. El precio de este dispositivo será aproximadamente de 18 euros.

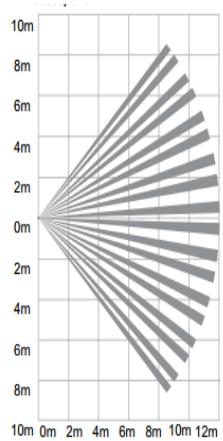


Figura 3.

7.3.1.2 Sensor PIR HC-SR501

El siguiente dispositivo será un sensor de movimiento a partir de la tecnología infrarroja nombrado habitualmente PIR. Este será un aparato digital ofreciendo una salida de nivel alto cuando el elemento piezoeléctrico detecta variaciones infrarrojas en su ambiente de trabajo. Además, será un dispositivo pasivo, es decir, no genera ningún tipo de energía para realizar la detección teniendo un consumo eléctrico bastante bajo.

Este elemento tendrá dos modos de funcionamiento modificables con un jumper, "retriggering" y "non-retriggering". El modo "retriggering" activará la salida a nivel alto mientras el sensor detecte movimiento y en el momento que no se detecte variación la salida se mantendrá a nivel alto durante un determinado tiempo y se desactiva transcurrido este tiempo. Después, el modo "non-retriggering" activará la salida durante un periodo de tiempo desactivando la salida cuando se cumpla este tiempo. Los periodos de tiempo que se ha mencionado en los dos modos se podrá configurar mediante un potenciómetro que tiene instalado este instrumento.

Posteriormente, la distancia de detección de este sensor podrá localizar cuerpos hasta 7 metros en línea recta. Respecto a la alimentación se podrá alimentar a 5 voltios cumpliendo la condición que se pretendía en la elección de este sensor. Además, teniendo como ventaja su bajo coste en comparación al sensor anterior siendo de aproximadamente 8 euros.

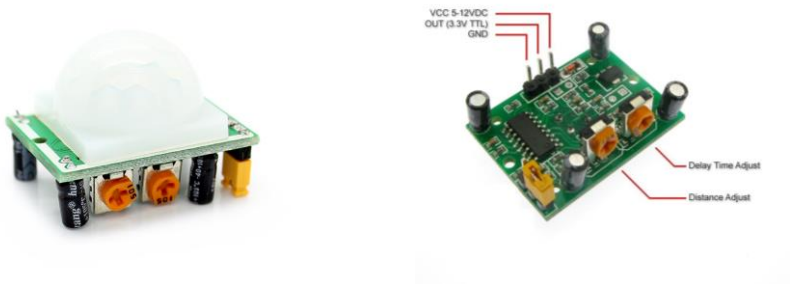


Figura 4.

7.3.1.3 Solución

La elección del sensor de movimiento ha sido el expuesto en último lugar, el sensor PIR HC-SR501. A pesar de que este sensor no aporta tanta fiabilidad como el anterior al utilizar solamente la tecnología de infrarrojo será suficiente para la función que se le quiere dar en este proyecto. La prioridad principal que se ha tenido en la elección de este sensor ha sido el precio reducido con respecto al otro sensor descrito. Además, las otras características serán significativamente peores pero suficientes para realizar la aplicación correspondiente.

7.3.2 Sensores de temperatura

Los sensores de temperatura nos permitirán medir la temperatura del ambiente en el que se encuentren. En el presente proyecto esta serie de sensores se instalarán en las diferentes áreas del domicilio para conocer la magnitud que aporta este instrumento.

La elección del sensor de temperatura se realizará teniendo en cuenta algunas características del mismo. Las características principales para la elección del sensor serán:

- **Rango de temperatura:** esta característica representa los valores de temperaturas que puede soportar el sensor sin que sea dañado. Los sensores se instalarán en zonas interiores donde habitualmente las temperaturas se encontrarán por encima de los cero grados por lo que no será indispensable que el sensor soporte temperaturas negativas.

- **Sensibilidad:** la sensibilidad del sensor será un apartado muy importante debido a que es la cantidad mínima de un cierto valor que se produce en la entrada del sistema para que afecte a su salida. Por lo tanto, si la sensibilidad es mayor se apreciarán mejor los cambios que se producen en la entrada respecto a la salida.

- **Rango de voltaje:** será la tensión mínima y máxima que puede soportar el sensor. En el proyecto existirán dos tensiones de referencia, 5 voltios, para alimentar el microcontrolador elegido y tres con 3 voltios para alimentar los módulos XBee.

Dependiendo de estos voltajes de referencia se buscará un sensor que pueda soportar dichas tensiones.

- **Electrónica adicional:** los sensores que ofrece el mercado puede que estén preparado para su funcionamiento directo con un microcontrolador o necesite electrónica adicional para su correcto funcionamiento. Por lo tanto, se tendrá en cuenta la electrónica necesaria para ello respecto a su complejidad y su coste adicional.

- **Error de temperatura:** el error que se producirá en la lectura del sensor respecto a la temperatura será un factor que se tendrá en cuenta en la elección.

- **Precio:** el precio será un apartado importante también al ser el proyecto de bajo coste, en el cual se buscará la mejor opción en precio respecto a su calidad.

7.3.2.1 LM35

El dispositivo LM35 es un sensor de temperatura analógico fabricado en un encapsulado TO-92. El rango de temperaturas en el que trabajará este sensor será desde -55 a 150 ºC teniendo la posibilidad de medir temperaturas negativas. Después, una característica importante será la sensibilidad la cual será de 10 mV / ºC, siendo una sensibilidad aceptable respecto a la aplicación que se quiere hacer en el proyecto. El rango de voltaje se establecerá en un rango de tensiones entre 4 y 30 voltios.

Posteriormente se destaca la no necesidad de introducir electrónica adicional al poder trabajar directamente con un microcontrolador y leer el voltaje de salida del sensor. También otros de los aspectos a señalar será el error que se produzca en la lectura del sensor siendo ±0,5 ºC a una temperatura aproximada a los 25 ºC, la cual será la temperatura habitual en la que trabaje en el presente proyecto. Por último, su precio será aproximadamente de 1,72 euros.

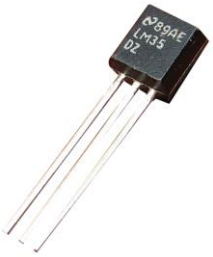


Figura 5.

7.3.2.2 TMP36

El sensor TMP36 es similar al sensor anterior variando algunas de sus características. Este será capaz de medir temperaturas por debajo de los ceros grados al ser su rango de temperaturas de -40 a 125 ºC. La sensibilidad será similar al del sensor descrito en el apartado anterior de 10 mV / ºC.

Después, el rango de voltajes estará dentro de las especificaciones requeridas entre 2,7 y 5,5 V. En este dispositivo, nuevamente no será necesario añadir componentes electrónicos para obtener un funcionamiento adecuado. Posteriormente, el error que producirá este elemento en su salida se encontrará en los rangos de ±1 ºC y teniendo la posibilidad de producir un error máximo de ±2 ºC. El precio en este caso disminuye respecto al sensor anterior siendo de 1,5 euros.

7.3.2.4 AD22103

El siguiente instrumento utilizará una salida analógica para representar los valores de temperaturas obtenidos. El dispositivo será capaz de medir temperaturas comprendidas entre 0 y 100 ºC cumpliendo la condición que se establecido en la elección de este elemento. Respecto a la sensibilidad es una característica destacable de este sensor al conseguir un valor de 28 mV / ºC.

Posteriormente, el dispositivo trabajará con una tensión de alimentación de 3,3 voltios. Además, será necesario añadir electrónica adicional para su correcto funcionamiento necesitando un filtro paso bajo en su salida. Luego, el error que se

produce en la salida se encontrará en los rangos aceptables para este tipo de sensores siendo de ±0,5 ºC considerando que alcanza valores máximos de ±2 ºC. El precio corresponderá con un coste de 4,13 euros siendo superior a los descritos previamente.

7.3.2.5 Solución

El sensor de temperatura que se escogerá entre las diferentes alternativas será el LM35. La elección de este componente se ha destacado por su carecía de electrónica adicional, el error producido en su salida, su sensibilidad y el precio considerado aceptables estas características para la aplicación en la que se pretende utilizar este sensor.

7.3.3 Sensor de luz

El sensor de luz será el que nos permita conocer la iluminación de la zona en la que se encuentre el dispositivo. Este instrumento ayudará a la hora de realizar el control de las luminarias del hogar conociendo el nivel de iluminación de la zona. La elección del sensor dependerá de dos características principalmente.

- **Precisión de iluminación:** la información de iluminación que puede aportar el sensor. En el presente proyecto el control de las luminarias será ON/OFF. Por lo tanto, no es necesario el conocimiento preciso de iluminación que se encuentre en la zona en la que esté instalado el sensor.

- **Electrónica adicional:** la electrónica necesaria para el correcto funcionamiento del sensor.

7.3.3.1 Resistencia LDR

La resistencia LDR es una fotoresistencia sensible a la luz que incide en ella, por lo tanto, su resistencia variará dependiendo de la luz de su ambiente. Por el motivo anterior, esta resistencia se utiliza como sensor de luz, pero se deberá adaptar para esta aplicación. Por lo tanto, este tipo de resistencia estará conectada a un circuito acondicionador aportando una salida en tensión proporcional a la luz que incide en este

componente. El motivo por el que la salida del acondicionador tenga que ser en tensión se debe a que dicha salida estará conectada a un microcontrolador y las entradas de este dispositivo estarán compuestas por un convertidor analógico - digital destacando que este componente electrónico trabaja en tensión a su entrada.

La información aportada por este sensor será un nivel de tensión dada por el convertidor analógico - digital. Por otra parte, al sensor se le debe aportar un circuito acondicionador para obtener una tensión proporcional al nivel de luz que incide sobre el sensor.

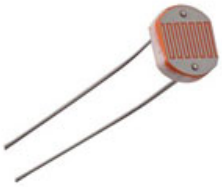


Figura 6.

7.3.3.2 Sensor BH1750

El sensor BH1750 será un sensor de luz. A diferencia con la resistencia LDR, este sensor nos aportará una señal digital. Este componente será capaz de dar mediciones en LUX siendo un sensor bastante preciso para conocer la iluminación de la zona en la que se instale. El BH1750 ya tendrá en su interior un circuito acondicionador sin necesidad de adaptar el sensor para obtener la magnitud deseada, aportando una tensión digital entre 0 y 65535.

Figura 7.

7.3.3.3 Solución

La opción escogida para el sensor de luz será la resistencia LDR. La elección se ha realizado principalmente por la falta de precisión que necesita el proyecto al tratarse de un sistema ON/OFF priorizando este apartado en vez de la electrónica adicional que necesitará el sensor para su correcto funcionamiento. Añadiendo, el elevado precio del sensor BH1750 en comparación con la resistencia LDR y su circuito acondicionador.

7.3.4 Sensor de consumo eléctrico

El sensor de consumo eléctrico, como su nombre indica se utilizará para la medición del consumo eléctrico. Esta serie de sensores se utilizará en los circuitos de fuerza del hogar para su medición de corriente y conocer con exactitud el consumo de ciertos dispositivos conectados a él.

En la elección del sensor de consumo eléctrico se tendrán en cuenta las siguientes características.

- **Invasivo o no invasivo:** en este apartado se considerará la necesidad de conectar el sensor en serie con el circuito de fuerza o por lo contrario, realiza la conexión mediante magnetismo sin necesidad de estar conectado directamente al circuito al que se quiere realizar la medición.

- **Electrónica adicional:** la electrónica necesaria para el correcto uso del sensor.

- **Precio:** la elección se realizará teniendo en cuenta el precio del dispositivo respecto a las prestaciones que ofrece el sensor.

7.3.4.1 Sensor ACS712

El dispositivo ACS712 será un sensor capaz de medir la corriente que circula por un circuito. Este dispositivo se conectará en serie con el circuito mediante un sensor de efecto Hall, a partir del campo magnético producido por la corriente aportará una salida en tensión proporcional a la corriente del circuito al que esté conectado. La relación entre

estas dos magnitudes será su sensibilidad la cual variará respecto al modelo del dispositivo. Existirán tres modelos diferenciando entre ellos la corriente que son capaces de soportar (5, 20 y 30 A), y su sensibilidad (185, 100, 66 mV/A). Por lo tanto, el dispositivo al ofrecer a su salida una tensión proporcional a la corriente no será necesario ningún componente electrónico adicional para su correcto funcionamiento con un microcontrolador.

7.3.4.2 Sensor SCT-013

El sensor SCT-013 es un dispositivo que contiene en su interior un transformador ofreciendo a su salida una tensión o corriente proporcional a la corriente de entrada al sensor, dependiendo del modelo del sensor se tendrá uno de los dos tipos de salidas. El funcionamiento del sensor consistirá de forma similar a una pinza amperimétrica introduciendo el conductor al que se pretende realizar la medición en el interior de un núcleo de hierro, situando alrededor de este componente una bobina con el objetivo de formar un transformador. El conductor que se introduce en el núcleo de hierro ejercerá de devanado primario en el transformador y la bobina enrollada en el núcleo de hierro como devanado secundario. El número de vueltas de la bobina se modificará dependiendo del modelo determinando la relación de transformación de cada uno.

La familia de sensores SCT-013 estará constituida por el SCT-013 100, SCT-013 030, SCT-013 020, donde el número acompañante al nombre del sensor serán los amperios que podrá medir como máximo. El modelo que puede soportar hasta 100 amperios aportará una salida en corriente proporcional a la corriente que circula por el conductor que se introduzca en el núcleo de hierro, aportando una relación de transformación de 100 A / 50 mA. Por lo contrario, los modelos restantes dispondrán de una resistencia a la salida del devanado secundario ofreciendo una salida en tensión proporcional a la corriente de entrada del sensor, presentando una relación de transformación de 30 A / 1 V y 20 A / 1 V, respectivamente.

La corriente que circula por el conductor al que se le quiere realizar la medición será una corriente alterna por lo que la salida del sensor será una corriente o tensión alterna la cual no se puede introducir en un convertidor analógico-digital. Este motivo es

debido a que el convertidor analógico-digital solo se puede introducirse tensión continua por las características de este componente electrónico.

En el proyecto, se conectará el sensor a un microcontrolador al cual en su entrada hay un convertidor analógico-digital. Por lo tanto, se deberá acondicionar la salida del sensor para conectarlo con el convertidor analógico-digital teniendo en cuenta la condición anterior y realizar una correcta lectura del sensor.

En conclusión, el sensor no necesitará conectarse en serie con el circuito de fuerza al que requiera la medición al tratarse de un funcionamiento electromagnético a diferencia del sensor anterior caracterizado por conexión en serie con el circuito. La salida del sensor deberá ser acondicionada para su correcto funcionamiento, por lo tanto, requerirá de electrónica adicional. Respecto al precio, el sensor aproximadamente tendrá un coste de 12 euros sin añadir los componentes adicionales del acondicionador de señal.

7.3.4.3 Solución

En la elección del sensor de consumo eléctrico se ha priorizado la característica de "invasivo o no invasivo" respecto a las demás características. Esta priorización se ha realizado por el hecho de que el sensor medirá la corriente que circula por la red eléctrica del hogar produciéndose habitualmente picos de corriente, dañando así los sensores que se encuentren conectados directamente a la red para realizar la medición. Por lo tanto, los sensores invasivos en este ámbito se dañarían con frecuencia teniendo un alto nivel de mantenimiento. En conclusión, se ha optado por elegir el sensor SCT-013 por su característica de no invasivo, teniendo como aspectos negativos su precio respecto del otro sensor a elegir y su electrónica adicional para su correcto funcionamiento, aun así, será la mejor opción al no sufrir daños por la red eléctrica.

Además, se destaca la elección del sensor SCT-013 30, siendo posible medir corriente hasta 30 amperios suficiente al ser instalado en los circuitos de fuerza del hogar que pueden llegar a soportar hasta 25 amperios como se pueden encontrar en los circuitos situados en la cocina.

7.3.5 Sensor capacitivo

El sensor capacitivo será utilizado para la interacción del usuario con diferentes sistemas. El funcionamiento de este elemento consistirá en la variación de capacitancia actuando como un condensador la placa del sensor y el cuerpo humano. La capacitancia de la zona de trabajo del sensor tendrá un cierto valor que aumentará cuando el usuario pulse o se acerque entre 1-5 milímetros, siendo detectada esta variación. Este sensor, es un dispositivo digital aportando una salida a nivel alto cuando se presione su zona de actuación.

Figura 8.

7.3.6 Potenciómetro

Un potenciómetro consistirá en la variación del valor de una resistencia. Este elemento, se utilizará para la regulación de temperatura en determinadas zonas del domicilio.

En este proyecto se optará por la elección de un potenciómetro rotatorio modificando la resistencia a medida que se gire un cursor que lleva instalado. Por lo tanto, se tendrá un valor proporcional a la tensión de alimentación ajustando los rangos de voltajes que aporta el sensor a los valores que se pretenden en el sistema de temperatura.

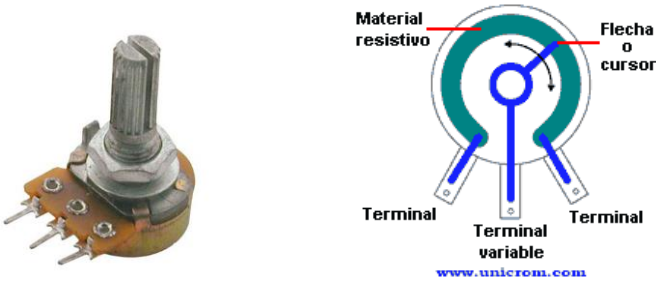


Figura 9.

7.4 Actuadores

Los actuadores son dispositivos encargados de accionar un proceso de control. Existen diferentes tipos de actuadores neumáticos, hidráulicos, eléctricos. La diferencia entre estos componentes será la magnitud física de activación para realizar un proceso concreto. En el proyecto actual se implementarán actuadores eléctricos, es decir, un dispositivo activado por una señal eléctrica.

En el proyecto se instalarán varios tipos de control en el hogar, control de luz, persianas y climatización. A continuación, se describen los actuadores elegidos, para cada uno de los procesos a controlar.

7.4.1 Luz

El control de las luminarias estará compuesto por un sistema de control ON/OFF. El actuador para accionar este sistema necesitará una serie de características. En primer lugar, será necesario un dispositivo que conceda la posibilidad de hacer circular o interrumpir la corriente dependiendo de una señal eléctrica. En segundo lugar, deberá ser capaz de soportar las características eléctricas de un circuito de luminarias de un hogar. Las características de un circuito de este tipo corresponderán a una tensión de 230 voltios y una corriente menor a 10 amperios. Por último, la señal de control hacia el actuador se realizará mediante un microcontrolador aportando este dispositivo una señal de 5 o 3.3 voltios, entonces, el actuador debe de ser capaz de accionar el proceso como mínimo a partir de una señal con los valores de tensión mencionados.

El dispositivo que cumple con las características descritas anteriormente será un elemento electromagnético denominado relé. El componente estará compuesto por dos circuitos independientes, un circuito de control y otro de potencia. El circuito de control estará formado por una bobina, un transistor y un diodo polarizado inversamente con la bobina. El transistor permite circular corriente por la bobina dependiendo de la tensión enviada a su base. Después, el diodo será necesario para proteger al transistor debido a los picos de corrientes que puede llegar a descargar la bobina. Por último, la bobina será un elemento indispensable para el funcionamiento del relé encargándose de conmutar el circuito de potencia, este efecto se realizará mediante magnetismo debido al campo magnético que genera este elemento cuando circula corriente por ella y produciendo el cambio de estado del conmutador por la atracción del campo magnético. Por otro lado, el circuito de potencia será el circuito a controlar accionado magnéticamente por la bobina. El circuito de potencia podrá soportar habitualmente más de 230 voltios y corrientes de hasta 10 amperios.

En conclusión, el relé será el dispositivo elegido para el control de las luminarias de la vivienda cumpliendo con las especificaciones necesarias para el control del circuito mencionado.

7.4.2 Persianas

El control de persianas requerirá dos tipos de actuadores. En primer lugar, un dispositivo que pueda realizar los movimientos habituales de este elemento siendo este un motor eléctrico. Dependiendo del tipo de persiana que se quiera automatizar deberá tener una serie de características. Posteriormente, será necesario un actuador que sea capaz de realizar un control sobre el motor debido a que un microcontrolador no es capaz de aportar al motor la suficiente corriente para su funcionamiento.

En este proyecto el cliente ha pedido dos tipos de automatización. Uno de ellos será la de persianas domésticas y el segundo la automatización de las cortinas que se encuentran en la sala. Por lo tanto, será necesario dos tipos de motores mostrados a continuación.

Por un lado, existirán motores diseñados para la automatización de persianas domésticas denominados motores tubulares. Los motores de este tipo se introducirán en el eje de la persiana. A su vez, la elección del motor dependerá de las características de este elemento a automatizar como puede ser su material, peso, anchura y altura. Las persianas del hogar tendrán las siguientes características.

- Diámetro del eje: 40 mm
- Material: aluminio
- Anchura: 130 cm
- Altura: 100 cm

Atendiendo a las especificaciones mencionadas el motor elegido será el ESC35/10, de la compañía "Motor & Blinds" cumpliendo con las condiciones de diseño estipuladas, anexo 5. El control del motor se realizará mediante cuatro conductores provenientes de este componente. Los dos primeros conductores corresponderán con la alimentación y tierra del actuador y los restantes a las señales de control para la orientación del giro del motor.

Por otra parte, la automatización de las cortinas se realizará con un motor para riel capaz de extender y retraer las cortinas a la posición que se desee. El motor a instalar será un motor específico para esta función que ofrece la compañía "Motor & Blinds".

Después, es necesario establecer un elemento entre el microcontrolador y el motor de las persianas por el motivo mencionado previamente para accionar el sistema de luminarias. Por lo tanto, se instalará un relé entre estos dos dispositivos conectando el circuito de control al microcontrolador y el motor al circuito de potencia del relé, siendo capaz de realizar un control de las persianas mediante el microcontrolador.

En conclusión, en el control de persianas será necesario la instalación de dos actuadores, un motor encargado del movimiento de la persiana y un relé para el control del motor mediante un microcontrolador.

7.4.3 Climatización

El control del sistema de climatización que se pretende instalar será un ON/OFF necesitando dos instrumentos que sean capaces de modificar la temperatura de una determinada zona para instalar el sistema que se pretende.

Por una parte, será necesario un aire acondicionado capaz de enfriar el área donde esté instalado. Después, el sistema de climatización se completará junto a un calefactor aumentando la temperatura ambiente.

Posteriormente, es necesario añadir un dispositivo entre los aparatos mencionados y el Arduino debido a que el microcontrolador no es capaz de activarlos con sus propias salidas. Por lo tanto, se instalará un relé teniendo la posibilidad de controlar el circuito de potencia al que estarán conectados los instrumentos que se utilizarán en este sistema.

En conclusión, el sistema de temperatura estará formado por un aire acondicionado y un calefactor realizando las acciones necesarias para realizar un control de la temperatura ambiente en determinaos sectores del domicilio.

7.5 Acondicionadores de señal

Los acondicionadores de señal son principalmente utilizados en los procesos de instrumentación. Los circuitos de este tipo se ocupan de la adaptación de señales eléctricas a unas características específicas dependiendo de la señal que se quiera obtener. Habitualmente, los circuitos acondicionadores son diseñados para lograr una mejor lectura de la señal que aportan los sensores. Normalmente, las señales aportadas por los sensores tienen rangos de medición pequeños u ofrecen no linealidad, por ello, los circuitos acondicionadores son capaces de amplificar, filtrar, linealizar o aislar señales para obtener una lectura adecuada del sensor.

En el presente proyecto, se utilizará un microcontrolador para realizar las medidas de los sensores. Los microcontroladores utilizan convertidores analógicos-digitales y digitales-analógicos para realizar las mediciones que se encuentren a sus entradas. Por lo tanto, la entrada tiene que ser una entrada continúa debido a que los márgenes de entrada de los conversores a utilizar serán de 0 a 5 voltios.

Los sensores que se han escogido para realizar el proyecto requieren a su salida un circuito acondicionador para que el microcontrolador realice una correcta lectura de estos dispositivos. Los sensores de temperatura y movimiento elegidos no requieren uso de este tipo de circuitos, al estar diseñados para trabajar con microcontroladores siendo instrumentos digitales. A diferencia de los sensores de luz y de consumo eléctrico que sí requieren un circuito acondicionador para que su salida pueda ser leída por el microcontrolador.

7.5.1 Sensor de consumo eléctrico

El sensor de consumo eléctrico elegido es el SCT-013 030 aportando una salida en tensión sinusoidal de ± 1 V. La lectura del sensor se obtendrá a partir de un microcontrolador, donde el margen de entrada de los convertidores analógicos-digitales encargados de leer la salida del sensor para que pueda ser procesada será de 0 a 1,1 V. Por lo tanto, la señal de entrada al convertidor analógico - digital tiene que ser continua.

Entre la salida del sensor y la entrada al microcontrolador será necesario un circuito acondicionador encargado de rectificar la señal transformándola en una señal continua. Existirán diferentes circuitos acondicionadores que pueden realizar esta función.

En primer lugar, los circuitos rectificadores formados por diodos serán bastante frecuentes en este ámbito debido al funcionamiento de estos dispositivos, siendo capaces de circular la corriente en un sentido y cortando el paso de corriente en sentido contrario. Los circuitos rectificadores compuestos por diodos se caracterizarán por la tensión umbral del diodo polarizando este componente en inversa hasta que la tensión en el diodo no supere aproximadamente 0.6 o 0.7 V, polarizando así el diodo directamente. Esto será un grave inconveniente debido a que la señal que aportará el sensor estará comprendida entre

± 1 V eliminando gran parte de la señal. Además, la señal se atenuará dependiendo de la tensión umbral del dispositivo y por la configuración del circuito (media onda u onda completa) disminuyendo así la señal dada por el sensor. En una configuración de media onda, figura 10, la señal es atenuada por la tensión umbral de un diodo. Mientras si el diseño se realiza para la rectificación de onda completa, figura 11 será necesario implementar 4 diodos polarizando directamente una pareja de diodos en un semiciclo de la señal encontrándose la otra pareja de diodos polarizados en inversa y viceversa para el semiciclo opuesto. En este caso, a diferencia del rectificador de media onda existirán dos diodos por cada semiciclo por lo tanto se atenuará el doble de la tensión umbral eliminando prácticamente la señal dada por el sensor.

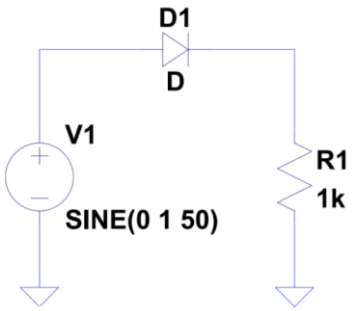


Figura 10.

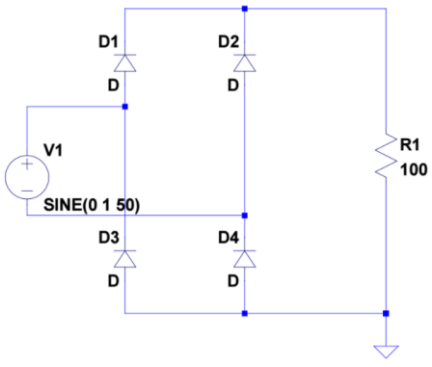


Figura 11.

Ante los inconvenientes que ofrecen los diodos es necesario rectificar la señal con otro dispositivo que no produzca estos efectos indeseados. Un circuito rectificador se puede diseñar también con amplificadores operacionales denominados habitualmente rectificadores de precisión. Los circuitos rectificadores con esta serie de componentes nos

permiten rectificar una señal alterna sin producir prácticamente atenuación. Por lo tanto, al tener que rectificar una señal con un nivel de tensión bastante bajo será necesario utilizar esta serie de circuitos para rectificar dicha señal.

En conclusión, se optará por diseñar un rectificador de precisión con el objetivo de transformar la señal alterna aportada por el sensor a una señal continúa leída por el conversor analógico-digital. La lectura obtenida por el microcontrolador será procesada y a partir de la relación de corriente - tensión aportada por las características del sensor (en este caso 30 A / 1 V), se obtendrá la corriente que se produce en el circuito de fuerza en cada instante. Por lo tanto, se calculará la potencia, aproximando la tensión de fase a 230 V obteniendo un error por realizar esta aproximación.

7.5.1.1 Rectificador de Precisión

Los rectificadores de precisión serán necesarios a la hora de rectificar señales con un nivel de tensión bajo rectificando la señal sin apenas atenuación. En el diseño del rectificador de precisión se ha optado por implementar un rectificador de onda completa, esto es debido a su gran rendimiento respecto al rectificador de media onda, es decir, su señal de salida estará compuesta por un nivel de continua superior y disminuyendo su componente de alterna ofreciendo una señal más estable a niveles de tensión continua se refiera.

En la implementación del circuito rectificador de onda completa se han tenido varias posibilidades.

7.5.1.1.1 Configuración: Un AO

La configuración del rectificador de precisión con un único amplificador operacional se mostrará en la figura 12. A continuación se mostrará el funcionamiento de este circuito.

En primer lugar, el diodo será el dispositivo que modifique el comportamiento del amplificador operacional. En el momento que el diodo este polarizado directamente el amplificador operacional estará realimentado y la tensión de salida corresponderá con la

configuración del amplificador operacional. En cambio, si el diodo se encuentra polarizado inversamente el amplificador operacional se comportará como un comparador, siendo su salida los niveles de alimentación alto o bajo dependiendo de la diferencia de tensión entre sus patillas de entrada. En este caso, la alimentación positiva será de 5 voltios y la tensión negativa conectado a tierra, 0 voltios.

A continuación, para conocer la polarización del diodo se realizará un circuito equivalente sin el diodo y se calculará la tensión que hay en ambos extremos del dispositivo dependiendo de la señal de entrada. Por una parte, el ánodo se encontrará conectado a la salida del amplificador operacional al no estar realimentado (circuito equivalente sin diodo) su comportamiento será el de un comparador, dependiendo de la diferencia de tensión que exista en sus entradas. Por otra parte, el cátodo estará conectado directamente a la señal de salida, por lo tanto, la relación de tensión entre la tensión de salida y la de entrada será un divisor de tensión.

Al tener en cuenta estas dos características dependiendo de la señal de entrada el amplificador operacional se comportará como un comparador en los semiciclos positivos de la señal de entrada y a su vez un inversor en los semiciclos negativos, rectificando así la onda que se encuentra en la entrada del circuito. La señal de salida rectificada será mostrada en el anexo 7.

También se destaca sólo la necesidad de alimentar el amplificador operacional con una sola fuente de tensión, alimentando de forma positiva a 5 voltios y conectando a tierra la alimentación negativa.

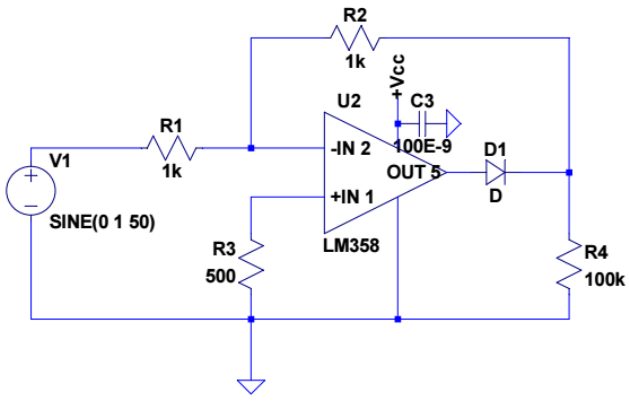


Figura 12.

7.5.1.1.2 Configuración: Dos AO

El rectificador de precisión con una configuración de dos operacionales estará representado en la figura 13. El funcionamiento de este rectificador será similar al descrito en el apartado anterior al funcionamiento de los diodos y a la polarización de estos dispositivos se refiera. A diferencia con el anterior circuito cada uno de los amplificadores operacionales representará en la salida un semiciclo de la señal de entrada.

En el instante que la señal de entrada se encuentre en el semiciclo positivo se polariza directamente el diodo del primer amplificador operacional, realimentado dicho amplificador operacional y formando un seguidor de tensión. Por lo contrario, el diodo del segundo operacional se encontrará polarizado inversamente, por lo tanto, el amplificador operacional se comportará como un comparador. Entonces, en el semiciclo positivo de la señal de entrada se refleja en la salida mediante un seguidor de tensión. Por otra parte, en el semiciclo negativo modificará el comportamiento de los amplificadores, invirtiendo el funcionamiento descrito para el semiciclo positivo. Entonces, la señal de entrada se mostrará en la salida debido al segundo amplificador operacional que actúa en una configuración inversora, rectificando así el semiciclo negativo.

En resumen, la señal de entrada será rectificada entregando a su salida una tensión continua. La condición de esta configuración será la alimentación de los operacionales que necesariamente se deben alimentar con una fuente dual.

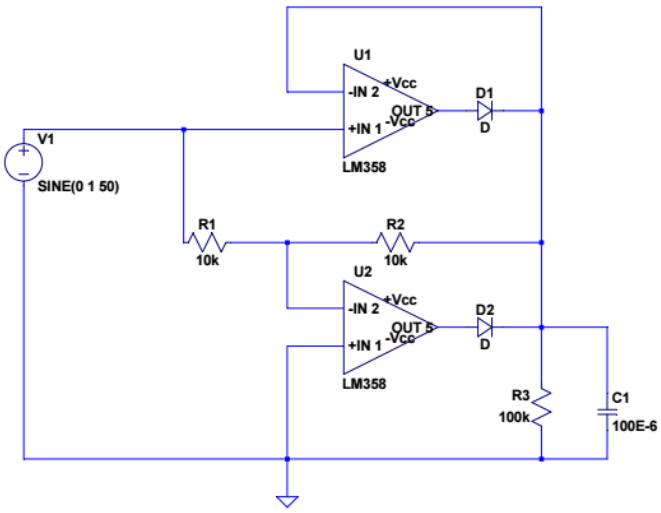


Figura 13.

7.5.1.1.3 Solución

En la elección del rectificador de precisión de onda completa será necesario elegir la configuración con un amplificador operacional, debido a su correcto funcionamiento con una única fuente de alimentación. La condición descrita es debido a la intención del proyecto de instalar este dispositivo en un módulo inalámbrico que requerirá una alimentación mediante baterías, eliminando así la necesidad de instalar una batería adicional para alimentar la fuente opuesta de los amplificadores operacionales.

7.5.1.1.4 Filtro

El objetivo de la medición es realizar el cálculo de la corriente eficaz de la señal. La corriente eficaz será la corriente continua que resulta al aplicar la misma potencia de la señal de pulsos que se obtiene del sensor a una resistencia con idéntico valor. En el cálculo de la corriente eficaz será necesario muestrear la señal para realizar un sumatorio del valor de la corriente en cada instante de tiempo de unas determinadas muestras para obtener dicho valor.

El sensor realizará la lectura de la corriente que circula por un conductor obteniendo la información del consumo eléctrico en el valor de pico de la señal. Entonces, al calcular el valor eficaz de la corriente se deberá realizar un muestre de la señal

consiguiendo una variedad de valores que no corresponderán con los deseados, aportando una lectura errónea.

La solución de este problema será mantener la señal de pico a lo largo del tiempo. Este efecto se podrá realizar con un condensador consiguiendo una tensión constante proporcional a la tensión de pico. Después, el condensador producirá un rizado de la señal que fluctúa entre la tensión de pico, ofreciendo una señal con unos valores muchos más cercanos al valor de pico que la señal de salida original. La gráfica que representa la señal de salida con él filtro se encontrará en el anexo 8.

En conclusión, al muestrear la señal filtrada para el cálculo de la corriente eficaz se obtendrá en cada instante valores cercanos al valor de pico de la señal. Por lo tanto, a partir de la señal filtrada se realizarán mediciones correctas con un cierto error que será la tensión de rizado de la señal producida por el condensador.

7.5.2 Sensor de luz

El sensor de luz se compone de una resistencia sensible a la luz, es decir, su resistencia variará dependiendo del nivel de luz que incide sobre ella. El dispositivo que se utilizará para realizar la lectura del sensor será un microcontrolador, donde la magnitud que es capaz de leer es voltaje. Entonces, será necesario realizar un circuito acondicionador que aporte una salida en tensión dependiente de la resistencia que se quiere utilizar como sensor de luz.

El circuito acondicionador escogido que cumpla con el objetivo descrito anteriormente será un divisor de tensión. Un divisor de tensión consta de dos o más resistencias, en serie, alimentadas entre una tensión determinada y tierra. La corriente que circula por la resistencia que lo componen, será la misma en todas ellas. La corriente variará, si una de las resistencias modifica su valor. Entonces, realizando una lectura en tensión, en un punto entre las resistencias que forman parte del divisor de tensión, se obtendrá medida dependiente del valor del conjunto de las resistencias. Cumpliendo con el objetivo necesario, para realizar la lectura con un microcontrolador.

En conclusión, el divisor de tensión formado por la resistencia LDR y una resistencia de valor fijo, formarán el sensor de luz del proyecto. En la figura 14, se muestra el diseño del circuito especificado en este apartado.

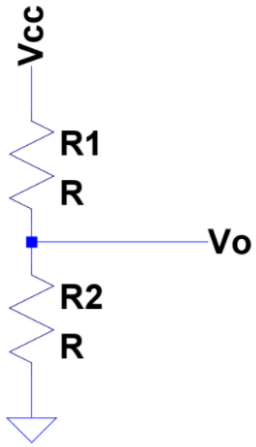


Figura 14.

7.6 Desarrollo de soluciones

En este capítulo se desarrollarán las soluciones que se han considerado oportunas analizando detalladamente la comunicación que se utilizará para las conexiones inalámbricas de los sensores y actuadores.

7.6.1 Protocolo ZigBee

El estándar ZigBee es creado en 2002 por ZigBee Alliance con la intención de avanzar en el ámbito de redes inalámbricas de sensores. ZigBee es un conjunto de protocolos de alto nivel con el propósito de diseñar redes inalámbricas por bajo coste, consumo y flexibilidad para poder adaptarse a diferentes medios.

La alianza ZigBee presenta dos tipos de especificaciones para abarcar diferentes mercados, se trata de Especificación ZigBee y Especificación ZigBee RF4CE. En primer lugar, la Especificación ZigBee ofrece la oportunidad de crear una red inalámbrica de sensores con 65.536 dispositivos en una misma red en la cual engloba los siguientes estándares:

- Cuidados para la salud (ZigBee Health Care)
- Domótica e inmótica (ZigBee Home Automation & ZigBee Building Automation)
- Consumo de energía inteligente (ZigBee smart Energy)
- Servicio de telecomunicación (ZigBee Telecom Services)
- Servicios de consumo de productos (ZigBee Retail Services)

Por otra parte, la Especificación ZigBee RF4CE la cual está pensada para productos de uso cotidiano con el objetivo de poder crear redes simples, robustas y con una comunicación de bajo coste. En este caso incluye los siguientes estándares:

- Control Remoto (ZigBee Remote Control)
- Dispositivos de entrada de ordenadores y dispositivos de consumo electrónico (ZigBee Input Device)
- Sistemas de movimiento en imágenes 3D (ZigBee 3D Sync)

7.6.1.1 Modelo de referencia OSI

En 1984 la organización Internacional para la estandarización creó el modelo de referencia OSI, para normalizar las comunicaciones. Este modelo describe siete capas por las cuales deberían pasar los datos cuando son transmitidos de un dispositivo a otro independientemente del medio en el que se realiza la comunicación. Las siete capas definidas serán las siguientes:

- **Capa física:** Define las características del medio físico tanto mecánicas como eléctricas. Por lo tanto, se establecen el tipo de conectores y la forma de la señal eléctrica. Además, las topologías que puede soportar (estrella, anillo, bus, árbol o malla), frecuencia de transmisión y el modo de emisión con el que se realizará la transmisión. Definiendo el hardware de la comunicación.

- **Capa de enlace:** En la presente capa se establece la longitud adecuada de los paquetes de datos y la detección de errores. Esta capa se puede dividir en dos protocolos de bajo nivel, MAC y LLC. El protocolo MAC (Medium Access Control) el cual forma el control de acceso al medio de las entidades de la comunicación. Por otra parte, la LLC (Logical Link Control) establece el control

de flujo y errores con la intención de presentar a la capa superior una interfaz más uniforme.

- **Capa de RED:** esta capa controla el direccionamiento y el enrutamiento entre diferentes redes.

- **Capa de transporte:** esta capa se encarga de organizar la información en paquetes para ser enviados a su destino, encargándose además del control de errores y el mantenimiento del flujo de la red.

- **Capa de sesión:** en este caso, la capa de sesión administra las comunicaciones entre equipos encargándose del sincronismo en el intercambio de datos.

- **Capa de presentación:** esta capa se utilizará como traductor entre redes, encargado de convertir los datos en un formato común para la comunicación entre diferentes redes.

- **Capa de aplicación:** establece los métodos que se pueden utilizar en las diferentes aplicaciones para la comunicación de datos. La presente capa será la de más alto nivel en referencia a las capas descritas.

7.6.1.2 Modelo de referencia OSI: ZigBee

La ZigBee Alliance aplicó el modelo de referencia OSI para definir su estándar optando por alguna de las diferentes capas que contiene este modelo. En primer lugar, define la capa física y la subcapa MAC establecidos en la norma IEEE 802.15.4. Por otra parte, en este protocolo se definen los estándares de red, seguridad y aplicación.

7.6.1.2.1 Norma IEEE 802.15.4

En la norma IEEE 802.15.4 se establecerán la capa física y la subcapa MAC las cuales estarán constituidas de una serie de protocolos de bajo nivel. Este estándar está diseñado para la creación de redes inalámbricas de área personal (WPAN), teniendo como características principales el bajo consumo y velocidad, dirigido hacia redes con una tasa de transmisión de datos baja. Añadiendo, que el estándar no contempla los niveles de

capas superiores a las mencionadas previamente, dando lugar a su creación como lo ha podido realizar ZigBee diseñando tanto la capa de red como la capa de aplicación.

7.6.1.2.1.1 Capa física

La capa física la cual también se puede nombrar PHY se encargará de definir las características tanto mecánicas como eléctricas. En primer lugar, se especificarán las frecuencias a las que se trabaja este estándar y además el alcance al que puede llegar la transmisión de datos. El rango de frecuencias que se utilizará variará según la geografía en donde se está trabajando y también cambiará la velocidad de transmisión dependiendo del rango de frecuencias:

- Banda de frecuencia de 868 MHz, el cual utilizará un único canal con una velocidad de datos de 20 Kbits/s, dicho rango de frecuencias sólo se utilizará en Europa.

- Banda de frecuencia de 915 MHz, en este caso se tendrán diez canales teniendo una gran diferencia con el rango de 868 MHz y una velocidad de transmisión de 40 Kbits/s destinado al norte de América y en el área del Pacífico.

- Banda de frecuencias 2,4 GHz, este rango de frecuencias es usada libremente en todo mundo teniendo grandes características respecto a los dos rangos de frecuencias mencionados anteriormente. Está constituido por 16 canales con una tasa de transmisión de datos de 250 Kbits/s.

La distancia de transmisión está caracterizada por las condiciones del medio y según en la banda de frecuencia en la que se trabaje. En un espacio libre sin obstáculos la comunicación puede llegar en los rangos de 868 y 915 MHz a 1 Km de distancia, mientras que en la banda de 2,4 GHz a penas se podrá llegar a 200 m. En un medio con obstáculos en el rango de frecuencias de 2,4 GHz, por ejemplo, el alcance se situará en el rango entre 10 y 75 metros de distancia, lo cual es la distancia idónea para su uso en el ámbito de la domótica.

La norma IEEE 802.15.4 establece dos tipos de dispositivos. En primer lugar, los de funcionalidad total (FFD: Full-Function Device), siendo el coordinador de la red y posteriormente, los dispositivos de funcionalidad reducida (RFD: Reduced-Function Device) los cuales tendrán menos recursos respecto a los de funcionalidad total. En el momento que se conectan dos dispositivos FFD y RFD se formará una red de área local (WPAN), teniendo la posibilidad de establecer dos tipos de topologías que serán en estrella y malla.

ZigBee en una topología malla, será capaz de crear diferentes rutas de transmisión de datos por distintos dispositivos, dicha función la realizará en capas superiores más concretamente en la capa de red.

7.6.1.2.1.2 Subcapa MAC

La norma IEEE 802.15.4, establece el nivel de enlace a partir de la subcapa MAC, definiendo su funcionalidad y los protocolos que la definen a dicha subcapa.

Las principales funciones que realizará serán:

- La unión y separación entre los dispositivos además del acceso al medio de dichos elementos.

- El control de seguridad, el cual esta función se llevará a cabo en esta subcapa y capas superiores.

- Métodos adicionales para la topología en estrella como pueden ser la generación de balizas.

Los protocolos que comprobarán el acceso al medio serán el CSMA-CA. Por una parte, el CSMA a partir de un método de probabilidad evitará las colisiones de transmisión entre distintos dispositivos. Este método consiste principalmente en esperar un tiempo aleatorio cuando un dispositivo quiere transmitir. Posteriormente, comprobará si el canal por donde se quiere transmitir está ocupado, si no lo está se realizará la transmisión y si está ocupado se volverá a realizar el mismo proceso. La función de comprobar la ocupación del canal

a transmitir se realizará a partir del protocolo CCA (Clear channel assessment), donde la capa física deberá comprobar la energía de la antena del dispositivo y determinar la fuerza de la señal recibida. Por lo tanto, a partir del protocolo CCA se conocerá si el medio está ocupado y permitirá a los dispositivos estar inactivos por un tiempo, transmitiendo en periodos de tiempo pequeños alargando así la vida de la batería que se utilice.

7.6.1.2.2 Capa de red

En esta capa de red (NWK) diseñada por ZigBee al margen de la norma IEEE 802.15.4 descrita en los apartados anteriores, proporciona dos tipos de funciones con el objetivo de ordenar los datos transmitidos. Los métodos serán por una parte el servicio de datos, el cual se ocupará de producir la unidad de datos y además define la topología a utilizar mediante el protocolo (PDU). Este protocolo facilita la comunicación entre los diferentes dispositivos utilizando las mismas unidades de datos, en definitiva, se podría decir que se trata de un traductor. Por otra parte, el segundo método que proporciona esta capa será el sistema de control, el cual lo proporciona la unidad de gestión de la capa de red (NLME: Network Layer Data Entity). Esta función tendrá como objetivo detectar los dispositivos conectados, el direccionamiento que tendrán los paquetes de datos y tanto la creación de la red como la conexión y desconexión de nodos.

7.6.1.2.2.1 Nodos y topologías

ZigBee especifica tres tipos de dispositivos:

- **Coordinador:** El coordinador es el dispositivo más completo de los tres dispositivos que se describirán. Este elemento será el encargado de crear la red en la que se conectarán los diferentes nodos, donde solo habrá un único coordinador en la red. Este dispositivo será el encargado de crear las rutas de transmisión entre los diferentes nodos.

- **Router:** Este dispositivo se encarga de conectar diferentes nodos que se encuentren fuera de rango de transmisión entre sí.

- **Dispositivo final**: El dispositivo final será el más sencillo de los tres debido que solo ejecutará las acciones que le soliciten los dispositivos superiores que serán el

coordinador y el router. Esta serie de dispositivos no se encargará de transmitir información a otro dispositivo teniendo la opción de "dormir", la cual se quedarán inactivos hasta que los demás dispositivos superiores lo soliciten aumentando la vida útil de las baterías que lo alimenten.

Respecto a las topologías que puede soportar este estándar serán las siguientes:

- **Estrella:** esta tipología se caracteriza por la conexión de los dispositivos finales y router conectados al coordinador únicamente. Por lo tanto, si dos nodos quieren comunicarse entre sí tendrá que ser mediante el coordinador de la red.

- **Malla:** esta topología consistirá en conectar todos los dispositivos entre ellos. En este tipo de topologías se pueden crear diferentes rutas de transmisión de datos, la cual será un sistema de seguridad al tener la posibilidad que un nodo de la red se quede inhabilitado. Esta topología es de las más utilizadas por la característica ya mencionada de redireccionamiento.

- **Árbol:** en este caso combinará los dos tipos de topologías anteriores diseñada para tener una gran cantidad de dispositivos, teniendo como característica una gran cobertura, pero teniendo como desventaja que puede alargarse el tiempo de transmisión de los datos por la cantidad de dispositivos que se pueden encontrar en la red.

7.6.1.2.3 Capa de aplicación

La capa de aplicación será la de más alto nivel entre las capas que forman el protocolo ZigBee. La capa en cuestión se dividirá en dos partes, la capa de soporte de aplicación (APS) y la capa de objetos de dispositivos ZigBee (ZDO: ZigBee Device Object). La capa de soporte tendrá la función de realizar la comunicación entre la capa de red y la capa de aplicación. A su vez la capa de soporte estará compuesta por la unidad de datos de aplicación (APSDE: Application Support Sub-Layer Data Entity) aportará la transmisión de datos a nivel de aplicación entre dispositivos de la misma red y detectando también la conexión de dispositivos. Por otra parte, para completar la capa de soporte se

tendrá la unidad gestora de soporte (APSME: APS Management Entity), elaborando el mantenimiento de una base de datos, nombrada APS Information base (AIB).

La estructura de aplicación ZigBee podrá estar compuesta por 240 objetos de aplicación, los cuales se enviarán y recibirán datos a través del APSDE-SAP y la unidad de datos de aplicación. En el APSDE-SAP para tener una mejor organización sean numerado los objetos, donde del 1 al 240 estarán los objetos del marco de aplicación, del 241 al 254 estarán destinados por los desarrolladores para cualquier uso y dos numeraciones específicas, el 0 para la interfaz de datos con la ZDO y el 255 para transmitir los datos a todos los objetos, denominado broadcast.

Respeto a la capa ZDO, la capa de objetos de dispositivos ZigBee, se encargará de inicializar la APS, la NWK y las especificaciones de servicio de seguridad (SSS). Otra de las funciones de esta capa será la de realizar la comunicación entre objetos de aplicación, tanto las direcciones de los elementos de la red como las conexiones de nuevos nodos.

7.6.1.3 ZigBee: Seguridad

Las características de la seguridad que utiliza ZigBee estarán repartidos a lo largo de las diferentes capas que se han descritos anteriormente. Estas especificaciones estarán situadas en la capa MAC, NWK y APS, donde la función de cada capa es la transmisión segura de los datos. ZigBee con la intención de mejorar la seguridad de la red utilizará métodos para la confidencialidad, encriptando los datos que se transmitirán en la red y la autenticación de los datos que el objetivo de este método es corroborar la validez y la no modificación de los datos.

7.6.1.3.1 Claves y Gestión de seguridad

Las redes ZigBee utilizará un procedimiento para mantener la seguridad en las conexiones entre los dispositivos mediante claves de seguridad. Existirán en una red tres tipos de claves, maestra, de red y de enlace. A su vez, mediante un protocolo de gestión de clave se generarán estas claves con unos procesos determinados descritos en los apartados siguientes.

7.6.1.3.1.1 Claves de seguridad

Como ya se mencionó en el apartado anterior habrá tres tipos de claves en una red ZigBee. Los diferentes tipos serán:

- **Clave maestra:** esta clave será la encargada de producir las claves de enlaces para las conexiones seguras entre dispositivos. La obtención de esta clave tendrá que ser a través de métodos fiables para mantener, un nivel alto de seguridad en las conexiones entre los dispositivos que componen la red.

- **Clave de red:** la siguiente clave estará establecida a nivel de red y cada dispositivo conocerá esta clave.

- **Clave de enlace:** en este caso se utilizará esta clave para las comunicaciones entre dos dispositivos. Por lo tanto, esta clave será única para cada pareja de dispositivos que deseen realizar una comunicación. El objetivo de esta clave es aumentar el nivel de seguridad en el transporte de la clave maestra.

7.6.1.3.1.2 Protocolo de gestión de seguridad

ZigBee utiliza un protocolo de gestión de seguridad para la obtención de las claves de seguridad mencionadas en el apartado anterior. Establecerá diferentes métodos para obtener las claves:

- **Preinstalación:** este método se realizará para la obtención de la clave maestras solamente, la cual el fabricante la introducirá en el propio dispositivo. En el caso de que el fabricante incluya varias claves para seleccionar el usuario podrá cambiar la clave maestra mediante jumpers que se encuentren en el dispositivo. Este método será uno de los más seguros al no ser necesario transmitir la clave maestra para su obtención.

- **Transporte de claves:** el proceso de transporte de claves consistirá en aportar a un nodo el rol de "centro de confianza". Los nodos que requieran alguna de los tres tipos de claves que se describieron en el apartado anterior, deberán

comunicarse con el nodo especificado para su obtención. El centro de confianza tendrá dos modos de funcionamiento posibles:

- **Modo comercial:** en este modo el centro de confianza será el encargado de almacenar una lista con los dispositivos, claves de red, de enlace y maestra que se encuentre en la red. El inconveniente de este módulo es la capacidad de almacenamiento de los dispositivos al almacenar cada vez más información a medida que la red aumente.

- **Modo residencial:** en este caso el centro de confianza sólo almacenará la clave de red. Por lo tanto, cada nodo almacenará el resto de la información necesaria. En este proceso no se tendrá en cuenta el control para verificar si algún intruso ha modificado el contador de tramas. En conclusión, este método será vulnerable hacia posibles ataques a la red.

- **Establecimiento de claves sin conexión:** este método consistirá en generar claves sin necesidad de transmitirlas para aumentar la seguridad de la comunicación entre dispositivos. ZigBee utilizará el protocolo SKKE el cual conseguirá establecer una clave aleatoria entre dos dispositivos sin necesidad que se transmita dicha clave. En este caso, se destaca la necesidad de conocer la clave maestra para realizar la comunicación entre los dispositivos, la cual se obtendrá a partir de alguno de los dos procedimientos descritos anteriormente.

7.6.1.3.2 Centro de confianza

El centro de confianza será el rol que se le aplique a un dispositivo concreto de la red, habitualmente al coordinador. Este dispositivo tendrá la misión del transporte de las claves para cada dispositivo de la red. El centro de confianza tendrá tres funciones fundamentales:

- **Gestor de validación:** se encargará de la validación para que un dispositivo obtenga este rol, de centro de confianza.

- **Gestor de red:** este proceso asumirá la distribución de la clave de red para almacenarla y para transmitirla a los dispositivos de la red.

- **Gestor de configuración:** El gestor de configuración se ocupará de la comunicación entre los dispositivos asegurando dicho enlace. Este proceso será mediante la distribución de la clave maestra y la clave de enlace.

7.6.1.3.3 Autenticación de dispositivos

En la seguridad de ZigBee se evaluará si los datos que se transmiten entre los dispositivos son auténticos, es decir, que sean los datos correctos y que no han sufrido modificaciones mediante el transporte de los mismos. Este procedimiento se realizará a partir del MIC (código de integridad de mensaje), este código lo genera el nodo transmisor y el nodo receptor deberá ser capaz de calcularlo. Los dispositivos no autorizados no podrán ser capaces de conocer este código. El nivel de seguridad se evaluará a partir del número de bits que contenga el MIC, los bits establecidos por orden de seguridad serán de 32, 64 y 128 bits.

7.6.1.3.3.1 Protocolo CMM y estándar AES*

ZigBee utilizará el protocolo CMM* para la generación del MIC en conjunto con el estándar AES encargado de encriptar los datos transmitidos. A partir de estos dos procedimientos se conseguirá un alto nivel de confidencialidad y autenticación de los mensajes transmitidos.

*7.6.1.3.3.2 Transmisión de datos: AES-CMM**

En la transmisión de datos se genera una trama de datos con el protocolo CMM* y el estándar AES. A partir, de los datos que se desean transmitir, una clave de seguridad y una cadena especial se realizará la trama de datos. La cadena especial se genera mediante el contador de tramas y no se producirán dos cadenas iguales debido a que el contador aumenta si se envían un mensaje con la misma clave de seguridad. Por lo tanto, el AES encriptará esta serie de datos y el CMM* genera el MIC.

En el caso de ataque externos, si se intenta retener el mensaje el receptor mediante el contador de tramas detectará que el mensaje se debería haber recibido anteriormente y el mensaje no se transmite con éxito. En el caso, que se intente modificar el contador de tramas el MIC calculado en el receptor no será similar con el que se encuentra en la trama de datos, por lo que el mensaje no será válido.

7.6.2 XBee

Los módulos XBee son dispositivos diseñados por la marca Digi International "Digi International, empresa líder en soluciones Connectware, facilita la conexión de dispositivos a la red (LAN) al fabricar productos y tecnologías rentables y sencillos".

Los módulos son utilizados para sistemas de comunicación entre diferentes dispositivos, conexiones punto a punto o redes mesh. El protocolo empleado en el enlace de los distintos módulos será el protocolo ZigBee, el cual será capaz de soportar redes mesh.

En el mercado existe una amplia variedad de XBee en los cuales todos los modelos tendrán el mismo número de pines, por lo tanto, se podrán intercambiar si fuera necesario en cualquier proyecto que requiera estos módulos. La diferencia que existe entre ellos serán el hardware, la potencia de transmisión y el tipo de antena.

7.6.2.1 Hardware XBee

La empresa Digi International ofrece dos tipos de XBee "Serie 1" y "Serie 2", a su vez dentro de la gama de "Serie 2" existen varios tipos descritos en los apartados siguientes. Uno de los aspectos importantes de estos dispositivos al trabajar en el ámbito de la radiofrecuencia es la banda de frecuencias en la que trabajan, siendo esta la de 2,4 GHz, excepto los módulos XBee XCS que trabajan en la banda de 900 MHz.

Los dispositivos de la "Serie 1" y de la "Serie 2" serán incompatibles, es decir, no se podrán realizar ninguna comunicación entre ellos, debido al hardware instalado en cada uno.

7.6.2.1.1 XBee serie 1

Los módulos XBee serie 1 serán los más sencillos de utilizar de todos los tipos de XBee presentes en este apartado. Esta serie soporta conexión punto a punto y punto a multipunto, pero no es capaz de implementar una red mesh. Por lo tanto, se podrán utilizar estos módulos cuando se requiera utilizar conexiones sencillas entre módulos. En cuanto a la potencia de transmisión será del orden de 1 mW.

Las especificaciones de este dispositivo se encontrarán en el anexo 9.

7.6.2.1.2 XBee serie 2

Los XBee de la "Serie 2" están representados por diferentes modelos descritos en los siguientes apartados. A diferencia con los XBee "Serie 1" los dispositivos de la "Serie 2" serán compatibles entre sí, por lo tanto, será posible crear conexiones entre estos tipos de XBee sin necesidad que sean del mismo modelo. La potencia de transmisión en esta serie será de 2 mW.

7.6.2.1.2.1 XBee Znet 2.5

Los módulos Znet 2.5 tendrán un hardware diferente a los módulos de la serie 1. Los cuales pueden trabajar en modo transparente o mediante comandos AT, dependiendo de la configuración que tenga el módulo XBee, las diferentes funcionalidades de los XBee se especificarán en el apartado de "modos de *funcionamiento". Se destaca que este tipo de modelo de XBee se ha retirado del mercado.*

7.6.2.1.2.2 XBee ZB

Los XBee ZB será similar a los módulos descritos en el apartado anterior, pero con diferente firmware. Dichos módulos podrán trabajar en modo transparente o en modo de comandos AT, con la posibilidad de crear redes mesh. La configuración de este tipo de XBee se deberá de establecer respecto a la utilidad de cada dispositivo de este tipo, coordinador, router o dispositivo final. Las configuraciones se describirán en el apartado de modos de funcionamiento. Las especificaciones de este módulo se establecen en el anexo 10.

7.6.2.1.2.4 XBee 2B

Los XBee 2B son módulos más actuales que los XBee ZB. La diferencia con estos módulos es la mejora del hardware respecto a estos, destacando la mejora de la eficiencia en la potencia transmisión.

7.6.2.1.2.4 XBee XSC

Los módulos XBee XSC serán los únicos que trabajan a una frecuencia diferente a los anteriores, 900 MHz. Al bajar la frecuencia de trabajo el alcance de transmisión será mayor respecto a los modelos anteriores que trabajan a una frecuencia de 2,4 GHz. A su vez para conseguir mayor distancia se reduce la velocidad de transmisión de estos módulos considerablemente siendo la velocidad de transmisión de 10 Kbps comparados con los 250 Kbps que puede alcanzar los tipos de XBee descritos en los apartados anteriores.

7.6.2.1.2.5 XBee PRO

Los módulos con la extensión "PRO" serán similares a los anteriores con la diferencia del alcance de transmisión de datos. Los XBee sin esta extensión serán capaces de alcanzar distancias de 100 metros en exteriores y unos 30 metros en interiores con obstáculos, aproximadamente, mientras que los módulos con dicha extensión tendrán la posibilidad de alcanzar distancias de 1,6 Km en exteriores y 90 metros en interiores con obstáculos. Este gran aumento de distancia de transmisión influye en la potencia de transmisión, siendo superior respecto a los dispositivos de los apartados anteriores, aumentando esta potencia en 50 mW aproximadamente.

7.6.2.2 Tipos de antenas

En los XBee vistos en el apartado anterior, se podrán colocar diferentes tipos de antenas, según las necesidades que se necesiten en el proyecto.

7.6.2.2.1 Antena de chip

Este tipo de antena estará introducida en el circuito impreso del dispositivo XBee. Las ventajas de este tipo de antena serán sus dimensiones, introduciéndola en espacios

reducidos. Por otro lado, las desventajas que tendrá será la atenuación de la señal transmitida en muchas direcciones.

7.6.2.2.2 Cable de antena

La antena de cable consistirá en un pequeño conductor colocado en el XBee. Este tipo de antenas conseguirá tener la misma señal de transmisión en todas las direcciones aproximadamente. Uno de los inconvenientes es su fragilidad y que se necesitará modificar el espacio en donde se instale el dispositivo, con el objetivo que la antena no se rompa.

7.6.2.2.3 Antena PCB

Este tipo consistirá en un conector en la placa del XBee, aportando la posibilidad de soldar la antena que se desee instalar. Según la aplicación que se necesite, se podrá colocar una amplia variedad de antenas.

7.6.2.2.4 Antena RPSMA

Los módulos XBee que implementen este tipo de antenas tendrán un conector para poder colocar una antena con un conector SMA, teniendo como ventaja instalar diferentes tipos de antenas y en diferentes direcciones.

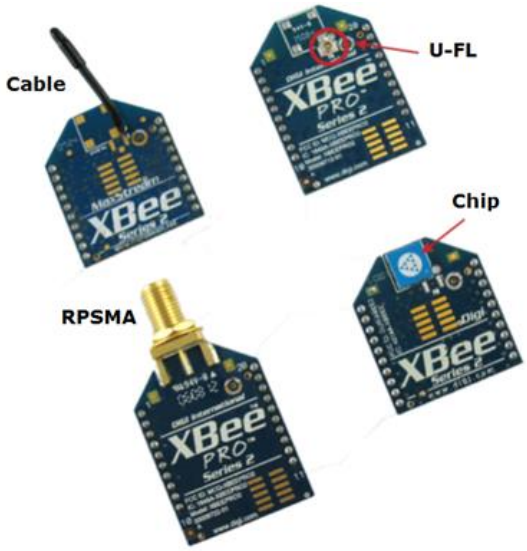


Figura 15.

7.6.2.3 Configuración

Los XBee tienen tres posibles configuraciones, coordinador, router y dispositivo final. Cada una de ellas serán necesarias para la creación de redes de dispositivos teniendo cada una su función.

7.6.2.3.1 Coordinador

El dispositivo XBee configurado como coordinador será el encargado de crear la red y el único con dicha configuración. Este dispositivo tendrá como misión establecer los parámetros del canal de transmisión y dirección PAN, ambos parámetros descritos en el apartado de 7.6.2.4. Posteriormente de realizar esta tarea, este dispositivo podrá enviar y recibir datos hacia otros nodos.

7.6.2.3.2 Router

Los dispositivos router se utilizarán principalmente para la transmisión de paquetes de datos entre nodos. Por lo tanto, será necesario tener esta serie de dispositivos en una red cuando dos nodos están muy alejados y no se puedan comunicar entre ellos. Los diferentes nodos de la red utilizarán esta serie de dispositivos para tener rutas secundarias para comunicar los nodos de la red, teniendo un alto nivel de seguridad si algún nodo queda inhabilitado.

7.6.2.3.3 Dispositivo final

Los dispositivos finales solo se podrán comunicar con los dispositivos padres como el coordinador o router. Esta serie de dispositivos no se encarga del enrutamiento de paquetes como lo pueden hacer el coordinador o el router, por lo tanto, no tendrán tanto consumo como lo pueden tener los otros dispositivos. Además, esta configuración tendrá una opción de poner el módulo XBee en modo sleep, el cual disminuye el consumo de este modo considerablemente al funcionar solamente en unos periodos de tiempo cortos, descrito en el apartado 7.6.2.5.

7.6.2.4 XBee: Direccionamiento

Los módulos XBee tendrán una serie de características para establecer la conexión entre diferentes dispositivos teniendo varios sistemas de direccionamiento para localizar los XBee que estén dentro de la misma red.

7.6.2.4.1 Dirección de 64 bits

La empresa que se encarga de la fabricación de estos dispositivos, Digi International, cada vez que se crea un XBee se le asigna una dirección única de 64 bits. Esta dirección se encontrará en la parte posterior del XBee en un formato de 16 dígitos en hexadecimal. La dirección de 64 bits no se podrá modificar debido a que estará instalada en el firmware del XBee y este firmware no se puede cambiar. Por lo tanto, no existirán dos módulos con la misma dirección de 64 bits teniendo la posibilidad teórica de crear redes de hasta 1.8E19 dispositivos.

Figura 16.

7.6.2.4.2 Dirección de 16 bits

En los XBee aparte de la dirección de 64 bits tienen establecida una dirección de 16 bits que a diferencia de la anterior se podrá modificar. Por lo tanto, en una misma red de dispositivos los módulos se podrán comunicar a partir de ambas direcciones de 64 o 16 bits. La dirección de 16 bits nos dará la oportunidad de crear una red de 65.536 dispositivos.

7.6.2.4.3 Dirección PAN y canal

En la creación de redes de dispositivos se especifican dos parámetros necesarios para la comunicación entre ellos, la dirección PAN y el canal en el cual se quiera transmitir los datos. Los dispositivos que quieran comunicarse necesitarán tener la misma dirección PAN y el mismo canal. Existirán 16 canales disponibles por la norma IEEE 802.15.4 cada uno de ellos estará distanciado por 5 MHz partiendo desde la frecuencia de 2,405 GHz hasta 2,480 GHZ. Por lo tanto, se podrá crear subredes con el mismo canal y diferentes direcciones PAN. Estos parámetros se podrán modificar descritos en el apartado 7.6.2.5.

7.6.2.4.4 Dirección broadcast

La dirección broadcast se utilizará para mandar el mismo paquete de información a todos los nodos con que se encuentren en el mismo canal y misma dirección PAN. La dirección de broadcast será en hexadecimal (0x0000 0000 0000 FFFF) la cual se introducirá en la dirección de destino del módulo XBee. También hay una posibilidad de hacer un broadcast para todas las redes PAN que se encuentran en el mismo canal de transmisión introduciendo una dirección PAN de 0xFFFF enviando un mensaje a todas las redes que encuentren en ese canal. A su vez también será posible realizar un doble broadcast, esto quiere decir, introducir la dirección de broadcast a la dirección PAN y la dirección de destino, con ellos se enviará el mismo paquete de datos a todos los nodos que se encuentren en el mismo canal.

7.6.2.5 Modos de funcionamiento

Los XBee tendrán diferentes modos de funcionamiento con los cuales se podrán realizar diferentes tareas según las necesidades. Los distintos modos se presentarán en los siguientes apartados.

7.6.2.5.1 Modo transparente

El modo transparente consistirá en enviar la información que se introduzca por el pin 3 del XBee (Data in) y se transmita por medio del pin 2 (Data out) hacia otro XBee configurado con el mismo modo de funcionamiento. Por lo tanto, la información que se

introduce por el pin 3 es guardada en el buffer de entrada y el XBee tendrá dos opciones para transmitir la información. La primera posibilidad será esperar a que se introduzca un carácter por el pin 3 o bien se espere un tiempo tras recibir la información. Otra de las alternativa para el envío de datos será cuando el buffer esté lleno esto sucede cuando el buffer contiene 100 bytes.

Este modo de funcionamiento se utilizará para lo que se denomina cable virtual conectando dos módulos XBee mediante una conexión serial sin necesidad de conductores eléctrico.

7.6.2.5.2 Modo de comandos

El modo de comando se utilizará para modificar o leer las propiedades en los módulos XBee. Los parámetros que utilizarán esta serie de dispositivos se denominan parámetros AT con los que una computadora o un microcontrolador se puede comunicar con el XBee a través de su puerto serial.

En el momento de comunicarnos con el XBee se ingresarán los caracteres "+++" esperando un tiempo GT para la respuesta del módulo, respondiendo con los caracteres "OK" indicando que se ha realizado con éxito la comunicación. Posteriormente a la respuesta del XBee se podrán modificar los parámetros del dispositivo. El formato para ingresar los comandos AT será introduciendo el prefijo AT seguido de los caracteres en formato ASCII del nombre del parámetro AT. Posteriormente se introducirá el valor del parámetro en hexadecimal.

Los módulos al introducir comandos AT podrán realizar dos opciones, modificar o devolver el valor del parámetro que tiene actualmente. La acción de lectura se realizará introduciendo los comandos AT con el formato descrito en el párrafo anterior, pero sin introducir el valor del parámetro.

La lista de comandos AT se encontrará en el anexo 11.

7.6.2.5.4 Modo Sleep

En el modo sleep se obtendrán grandes resultados de ahorro de energía en los dispositivos. Los XBee inicializará este modo con el comando AT SM dependiendo del valor de este parámetro se realizarán diferentes funciones.

- **Comando AT SM 1:**

Cuando se ingresa el valor 1 de este comando el dispositivo entrará en modo hibernación, es decir, el módulo no atenderá a las tareas de recepción, transmisión o asociación con otros dispositivos. Este modo estará activo mientras el pin SLEEP_RQ (pin 9) se encuentre en HIGH, por lo tanto, solo saldrá de este modo cuando el pin SLEEP_RQ se encuentre en LOW. En este modo se ahorrará una gran cantidad de energía siendo la función que menos consuma del modo sleep.

- **Comando AT SM 2:**

Esta función será similar cuando se introduce el valor 1, pero no tendrá tanto ahorro de energía como tenía la anterior función y la acción cuando se despierta el módulo será más rápida.

- **Comando AT SM 4:**

Cuando se le ingresa el comando AT SM con el valor 4, el módulo entrará en modo sleep dependiendo de una serie de parámetros. El primero será el comando ST el cual controlará el tiempo que transcurrirá para que el módulo entre en modo sleep siendo su rango de 0 a 0xFFFF (multiplicado por 1 ms). El otro parámetro que tendrá que analizar el dispositivo será el comando SP estableciendo el periodo de tiempo que dure el módulo en modo sleep teniendo un rango de tiempo de 0 a 0x68B0 (multiplicado por 10 ms). Por lo tanto, el módulo se despertará cada cierto periodo de tiempo analizando su entrada de datos y si fuera necesario ejecutará las tareas que tenga que realizar para posteriormente entrar nuevamente en modo sleep.

- **Comando AT SM 5**

Esta función será similar a la anterior dependiendo también del parámetro ST, pero la acción de despertar del modo sleep será diferente. En este modo utilizará el pin SLEEP_RQ para despertar del modo sleep cuando el estado del pin se encuentre en nivel bajo. En el instante que se produzca esta acción el módulo despertará y en el caso de detectar datos entraste realizará las tareas necesarias. Posteriormente colocará el timer del comando ST a 0 mientras este timer no termine y el dispositivo entre en modo comandos se ignorará cualquier actividad del pin SLEEP_RQ.

En la tabla 1 se mostrará los valores del modo sleep y el consumo que se tendrá en cada una de las funciones de este modo.

Modo Sleep	Consumo de los módulos XBee		
	2,8 - 3 V	**3,2 V**	**3,4 V**
SM 1	**< 3 uA**	**32 uA**	**255 uA**
SM 2	**< 35 uA**	**48 uA**	**170 uA**
SM 4	**< 34 uA**	**49 uA**	**240 uA**
SM 5	**< 34 uA**	**49 uA**	**240 uA**

Tabla 1.

7.6.2.5.4 Modo API

El modo API será el más robusto de todos los modos de funcionamiento descritos en los apartados anteriores por su variedad de funciones. En este modo los módulos esperarán recibir la información en un formato concreto donde la información se encontrará en paquetes de datos denominados tramas API. Similarmente, para el envío de información cada módulo empaqueta los datos en una trama API y lo envía al módulo de destino. Dependiendo del contenido de la trama se realizará una determinada función. Las funciones permitirán la modificación de comandos AT locas como remotos o enviar datos entres distintos dispositivos entre otras. Se destaca que cuando se envía una trama API entre XBee, el módulo receptor responderá con una trama API concreta dependiendo del tipo de función que reciba este módulo receptor.

7.6.2.5.4.1 Estructura

A continuación, se describirá la estructura de una trama API, figura 17.

- Byte 1: describe el inicio de la trama, el cual siempre empezará por el byte 0x7E. Por lo tanto, avisará al módulo que recibe la trama que cuando reciba este byte está comenzando la trama API.

- Bytes 2 y 3: estos bytes serán los encargados de informar el número de bytes que se encuentran posteriormente al byte 3 de la trama.

- Bytes 4 - n: entre esta serie de bytes se encontrará el mensaje que se quiera transmitir. La estructura de estos bytes dependerá del tipo de función de la trama API, descrita en los siguientes apartados.

- Bytes n + 1: estos bytes formarán el checksum, el cual será un método de seguridad con el objetivo de saber si se transmitieron todos los bytes. El procedimiento para el cálculo del checksum se realizará en el apartado "checksum".

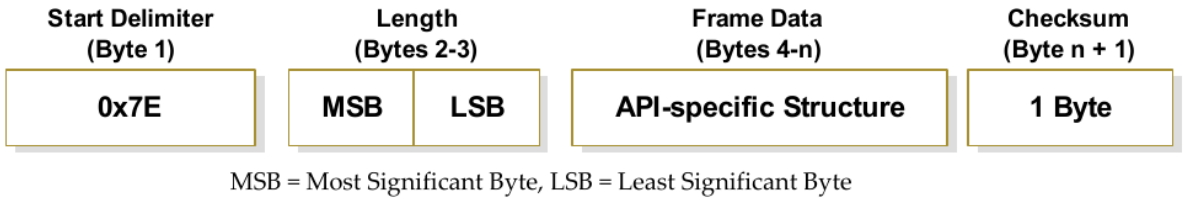


Figura 17.

7.6.2.5.4.2 Checksum

El checksum como se mencionó anteriormente será un método de seguridad para la correcta recepción de datos. El procedimiento para elaborar el cálculo del checksum consistirá en realizar la "y" lógica entre el número de 8 bits 0xFF y los "bytes 4 - n" descrito en el apartado anterior. Posteriormente el resultado de 8 bits se restará al mismo número de 8 bits 0xFF y el resultado será el checksum que necesita la trama API.

7.6.2.5.4.3 Modo API 1 y API 2

El modo API tendrá a su vez dos modos de funcionamiento, el modo API 1 y API 2. La diferencia entre los dos es que el modo API 2 tendrá bytes de "escape". Este modo es creado con el objetivo de evitar que dentro de la trama API los bytes "reservados" no se confundan con la información que se quiera transmitir. Los bytes "reservados" serán los bytes destinados a señalar determinadas acciones como podría ser el byte 0x7E, comenzando de nuevo la trama en el caso de que este byte se encuentre en el interior de una trama. Por lo tanto, el modo API 2 empleará unas determinadas acciones para diferenciar estos bytes. Los bytes "reservados" serán los siguientes:

- 0x7E: byte de inicio de la trama.
- 0x7D: byte de escape
- 0x11: señalizador XON
- 0x13: señalizador XOFF

En el modo API 2 en el momento que se quiera "escapar" un byte se colocará el byte de escape 0x7D seguido del resultado de la operación lógica "XOR" del el bytes a transmitir y el byte 0x20.

7.6.2.5.4.4 Tipos de tramas API

Los tipos de las tramas API se especificarán en el mensaje de datos de la propia trama. Los bytes que se representan en las diferentes tramas estarán en formato hexadecimal. Posteriormente se describirán las funciones más destacadas exponiendo su estructura en las imágenes correspondientes.

- Modo API: Configuración del módulo local XBee

La configuración de un módulo local en modo API se realizará con una trama API, de un tipo específico. La trama que se utilizará será de tipo AT command (0x08) y se tendrá una respuesta del módulo receptor del tipo AT command response (0x88).

La estructura de esta trama API será similar a la que se encuentra en la figura 18. Ambas tramas AT command y AT command response tendrán una estructura similar diferenciando el tipo de trama (0x08 o 0x88).

Figura 18.

- Envío: AT Command (0x08)
- Respuesta: AT Command Response (0x88)

- Modo API: Configuración remota de un módulo XBee

A partir de esta trama se configura un módulo XBee de forma remota con otro módulo XBee sin necesidad de configurarlo con el software X CTU. La trama API a enviar será del tipo Remote AT Command (0x17) siendo posible configurar todos los comandos AT disponibles. La trama de respuesta será Remote AT command response (0x97), ofreciendo el resultado del comando configurado.

La estructura de la trama para enviar el comando AT de forma remota será como la estructura representada en la figura 19.

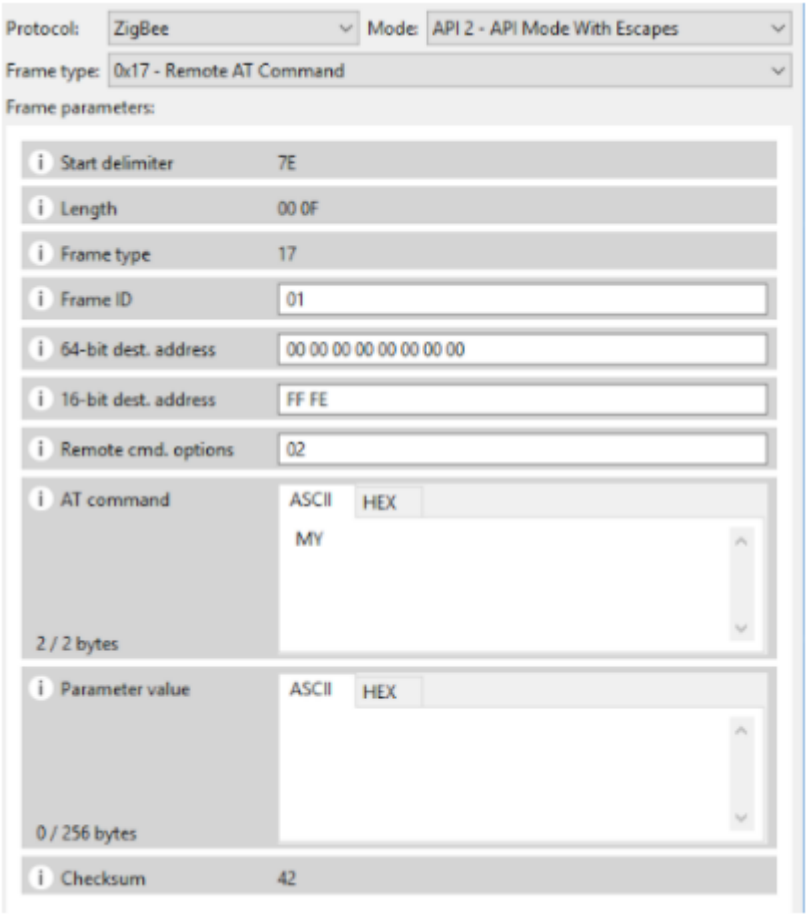


Figura 19.

La figura 20 especifica la estructura de la trama API para la respuesta del módulo receptor diferenciándola de la anterior con command status, valorando los distintos estados de la trama enviada.

- Envío: Remote AT command (0x17)

- Respuesta: Remonte AT command response (0x97)

Figura 20.

- Modo Api: Transmitir y recibir datos

Este tipo de tramas se podrá enviar una serie de datos que se deseen. El tipo de trama que se utilizará en este caso será Transmit request (0x10) donde el módulo receptor enviará una trama de recepción de los datos enviados, la cual será Receive packet (0x90).

La estructura de la trama Transmit request será como la mostrada en la siguiente imagen.

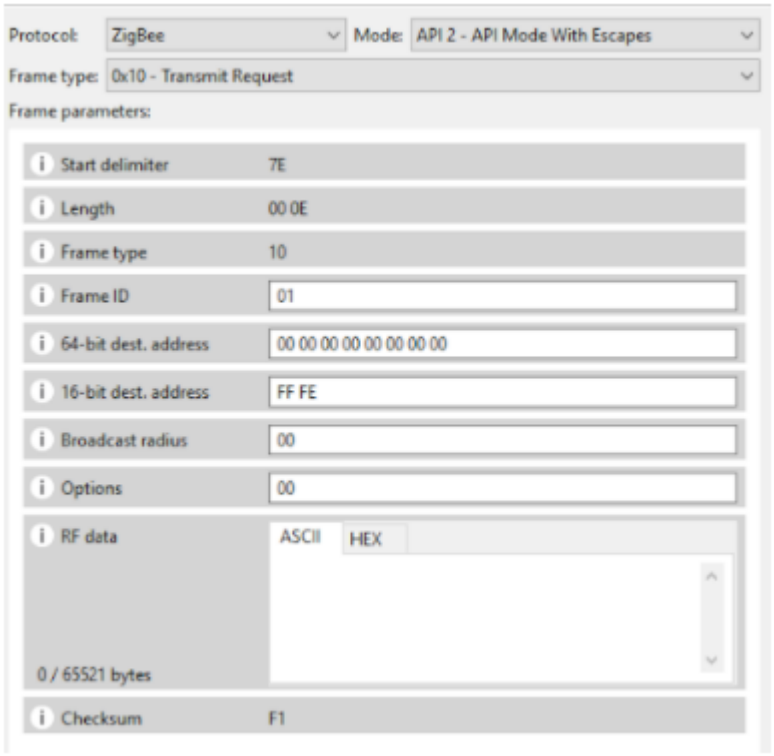


Figura 21.

Posteriormente se muestra la estructura de la trama Receive packet en la figura 22.

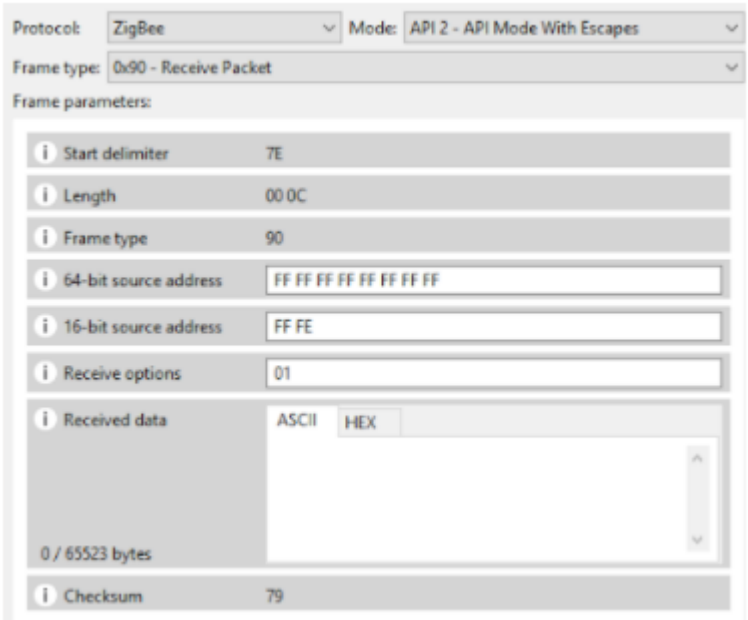


Figura 22.

- Envío: Transmite request (0x10)

- Respuesta: Receive packet (0x90)

7.6.2.5 Seguridad

Los módulos XBee establecerá el nivel de seguridad de la red especificados en el protocolo ZigBee. Los XBee deberán activar la seguridad de la red en su configuración estableciendo la clave maestra que se obtendrá por los métodos de obtención de clave, preinstalada o mediante un centro de confianza y clave de red para la introducción de nuevos dispositivos. Por lo tanto, los datos transmitidos entre los dispositivos de la red se encontrarán encriptados con la clave de red y también por la clave de enlace obtenida por los diferentes métodos de obtención de claves.

7.6.2.5.1 Seguridad: configuración

Los XBee deberán ser configurados con el comando AT EE, con el valor 1 para activar la encriptación de los datos estableciendo la seguridad de la red. Posteriormente, se deberá configurar la clave de red con el comando NK, por defecto este comando recibirá el valor 0 esto quiere decir que la clave de red se obtendrá de forma aleatoria. Otra posibilidad sería que se especifique un valor al comando NK y la clave de red tendría un número específico.

La configuración que se deberá establecer en los módulos XBee para disponer de una red segura se especificarán en los siguientes apartados. La configuración del dispositivo estará representada por medio de los comandos AT.

7.6.2.5.1.1 Activar seguridad

Los XBee establecerán la seguridad de la red mediante el comando AT EE. A partir, de este comando con el valor 1 se activará la encriptación de los datos, asegurando la red.

7.6.2.5.2 Clave de red

En este apartado se especificará que la clave de red deberá establecerse en el coordinador de la red. Esta clave se representará con el comando NK siendo su valor por defecto 0, donde establecerá un número aleatorio para determinar esta clave. También será posible especificar un número concreto introduciendo dicho valor en el comando NK, el cual solo será de escritura y no de lectura. Los dispositivos router o dispositivos finales se unen a la red a través del coordinador que les transmitirá la clave de red encriptada.

7.6.2.5.3 Clave de enlace

El coordinador será el dispositivo que funcionará como centro de confianza distribuyendo las diferentes claves que se establecen en la red, necesitando una clave de enlace para aumentar la seguridad de la comunicación cuando repartan las claves a los demás dispositivos. La clave de enlace se establece con el comando KY siendo su valor por defecto 0, es decir, obtendrá la clave de forma aleatoria del centro de confianza. Sí la clave de enlace se deja con el valor por defecto los dispositivos que se unan a la red se les enviará la clave de red sin que esté encriptada, por lo tanto, no será recomendable para la seguridad de la red. Sí la clave de enlace es preinstalada en los diferentes dispositivos, la clave de red será encriptada y transmitida a los dispositivos que se unan a la red.

7.6.2.5.4 Centro de confianza

El centro de confianza se establecerá con el comando EO. El centro de confianza será capaz de actualizar la clave de red donde esta clave se podrá modificar mediante el centro de confianza o sin el centro de confianza.

La actualización de la clave de red con un centro de confianza se realizará mediante el propio centro de confianza sin necesidad de cambiar de canal de transmisión ni dirección PAN, actualizando los demás dispositivos la clase de red.

Por otra parte, la actualización de la clave de red sin centro de confianza se realizará con el comando NK haciendo que los dispositivos abandonen la red cambiando

el canal de transmisión y su dirección PAN. Posteriormente el coordinador creará una nueva red restableciendo la clave de red.

8. Resultados finales

La instalación de los sensores y actuadores se realizará en un formato modular, es decir, creando módulos que realicen diferentes funciones que pueden ser de utilidad para realizar el control que se requiere en el hogar. La gran ventaja de esta configuración es la de ajustarse a las necesidades del cliente domotizando ciertas zonas de la casa o bien realizando la domotización de la vivienda en su totalidad. Además, será muy sencillo de esta forma mejorar el sistema incluyendo nuevos módulos sin la necesidad de realizar grandes cambios en los demás dispositivos.

A partir, de los requisitos del usuario y los planos del domicilio se han creado los siguientes módulos implementando diferentes sensores en su interior para conseguir la función que se le pretende dar al módulo en concreto. A continuación, se presentan los módulos diseñados para la presente vivienda.

- Módulo Habitación
- Módulo Pulsador
- Módulo Actuador
- Módulo de movimiento
- Módulo Casa

8.1 Sistema de comunicación

En cada uno de los módulos que se han diseñado se ha instalado un XBee para realizar la comunicación entre los módulos. Cada uno de ellos, tendrá que trabajar en el

mismo canal y con la misma dirección PAN con el objetivo de crear una red de dispositivos en todo el domicilio.

Posteriormente, se ha activado la seguridad de cada uno de los XBee creando una red de dispositivos segura previniendo ataques externos al sistema. La seguridad, se ha activado configurando los siguientes comandos:

- **Comando AT EE:** este comando activará la encriptación de datos. Los datos se transmiten en paquetes que se encuentren encriptados por el estándar AES que establece el protocolo ZigBee siendo desencriptados en el lugar de destino.

- **Comando AT EO:** a partir de este comando, se activarán las diferentes opciones de encriptación del protocolo ZigBee. Este comando estará compuesto por 7 bits activando cada opción de encriptación dependiendo de los bits que se activen.

  En primer lugar, se excitará el bit 1 el cual activará el uso del centro de confianza transmitiendo las claves de seguridad que establece el protocolo ZigBee, para el envío de datos entre dispositivos. Posteriormente, se activará el bit 3 accionando la autenticación de los datos que confirmará que los datos han llegado a su destino correctamente y sin sufrir cambios.

- **Comando KY:** este comando establece la clave de enlace. Esta clave será una de las condiciones para la transmisión de datos siendo instalada en cada uno de los dispositivos de la red. Esta configuración se ha realizado para que cada XBee tenga preinstalada esta clave con el objetivo de que no sea transmitida, disminuyendo las probabilidades de que esta clave pueda ser captada por dispositivos externos.

- **Comando NK:** en este comando se establecerá la clave de red, siendo una condición más para conseguir una comunicación segura. En este caso, se colocará el valor por defecto de 0 siendo el recomendado por el fabricante. A partir de esta configuración la clave de red será generada por el coordinador de forma aleatoria.

8.2 Módulo habitación

Este módulo será el más complejo de los que se han diseñado el cual procesa todos los datos de los sensores que se encuentren en su interior y los que estén en módulos remotos, activando los actuadores correspondientes a los sistemas de control que se instalen en él. Dependiendo del control que se quiera hacer en cada zona este módulo tendrá una configuración diferente junto con los sensores que necesite cada sistema. A continuación, se describirán los sistemas de control diseñados, para cubrir las necesidades del usuario.

8.2.1 Sistema de luminarias

El sistema de luminarias instalado en este domicilio será un ON-OFF ejecutando un control sobre un actuador. En este sistema se han empleado los sensores de movimiento y luz, incluyendo un reloj de tiempo real que se utilizará para determinadas acciones del sistema. También, se han diseñado diferentes modos de funcionamiento para activar y desactivar las luminarias con ciertas condiciones. Después, este control se podrá combinar con un módulo que lleve instalado un pulsador para activar los diferentes modos de funcionamiento.

8.2.1.1 Sensores

El sensor de movimiento resolverá el apartado de visión del módulo indicando la presencia de personas en un lugar determinado. En cada uno de los módulos que necesite este sensor será necesario instalar dos dispositivos de este tipo. El propósito de este sensor adicional será aumentar la probabilidad de detestación de un cuerpo debido a que estos dispositivos carecen de una alta sensibilidad, necesitando movimientos amplios para que pueda detectar los cambios de temperatura que generan las personas. Por lo tanto, se instalarán dos sensores de este tipo cuando se requiera.

Por otra parte, el sensor de luz detectará el nivel de iluminación que se encuentra en el área en el que esté instalado. Al conocer el nivel de luz cuando sobrepase un cierto umbral de oscuridad el microcontrolador conocerá si la habitación está poco iluminada. El umbral de iluminación se ajustará a cada una de las habitaciones y al usuario.

Por último, el reloj de tiempo real ha sido instalado para conocer la hora del día siendo necesario para activar ciertos modos de funcionamiento que se explicarán en el apartado siguiente. Arduino tiene relojes internos que pueden realizar esta función, pero a su vez tiene ciertos inconvenientes no deseados. Este reloj interno, se reiniciará cada 50 días aproximadamente y además cuando el Arduino se desconecta de la corriente el reloj se inicia en cero el uno de enero de 1970. Esta serie de inconvenientes no se tendrán con el reloj de tiempo real teniendo una pila interna, que servirá para no perder la hora en la que se encuentra cuando se desconecta este dispositivo de la corriente. Además, aportará datos interesantes como el día, la semana o el mes en que se encuentra.

8.2.1.2 Modos de funcionamiento

Los modos de funcionamiento se tendrán en cuenta cuando se coordine este sistema con un pulsador siendo los siguientes:

- **Modo Automático:** esta será la configuración por defecto, activando las luces cuando exista movimiento y poco iluminada en la en la zona que se desee. Posteriormente, para su desconexión será necesario que no se detecte movimiento durante un determinado tiempo. Este periodo, deberá de ser del orden de minutos debido a la poca sensibilidad de los sensores asumiendo que un tiempo con estas unidades, el usuario realice movimientos suficientes para que sea detectado por los sensores de movimiento.

- **Modo Manual:** la habitación pasará a este modo cuando se presione un pulsador enlazado con este sistema. Este modo se ha diseñado con el objetivo de que el usuario pueda apagar y encender la luz mediante un pulsador cuando él lo determine. Después, este modo se cambiará al modo automático cuando no se detecte movimiento en un determinado tiempo.

- **Modo Nocturno:** este modo realiza la misma función que el modo manual, interactuando con uno o varios pulsadores. La diferencia que tendrá será que solo funcionará a unas determinadas horas, descritas en este proyecto como “horas nocturnas” y “horas de mañana” y solo se podrá modificar cuando se supere la

"hora de mañana", al cual será el modo automático. Esta configuración se ha diseñado por las horas de sueño del usuario, con la intención de que no se active la luz si el usuario realice un movimiento cuando este durmiendo. Por otra parte, en este modo de funcionamiento será cuando se utilice el reloj de tiempo real para determinar las horas que activan este modo.

8.2.2 Sistema de climatización

El sistema de climatización consistirá en un sistema ON-OFF aplicando este control a un conjunto de actuadores que formarán el sistema de temperatura de la zona en el que se instale. El sistema de climatización regulará la temperatura a un valor determinado formado este sistema por un aire acondicionado y un calefactor.

El control de esta magnitud se llevará a cabo mediante el sensor de temperatura LM35, conociendo la temperatura de la zona siendo modificada por el sistema de climatización. Dependiendo de la consigna que se establezca en el sistema se activarán o desconectarán los actuadores correspondientes. Existirán dos casos posibles, que el valor del sensor esté por encima de la consigna en cuyo caso se activará el aire acondicionado y que dicho valor esté por debajo de la consigna, activando la calefacción.

8.2.2.1 Tolerancia de temperatura

El sensor que se ha elegido para desarrollar este sistema de control tendrá un error de lectura de ±0,5 ºC teniendo la necesidad de establecer una tolerancia entre la consigna establecida de ±1 ºC. Por lo tanto, el sistema de climatización se activará cuando se encuentre por fuera de este rango. Además, con este método se evitará que los actuadores se estén activando y desactivando continuamente.

8.2.2.2 Modos de funcionamiento

El sistema presente se ha creado para trabajar únicamente cuando está enlazado con un "Módulo Pulsador". Por lo tanto, el usuario modificará la consigna mediante el potenciómetro que se encuentre en este dispositivo.

8.2.2.3 Mostrar consigna

El sistema de temperatura mostrará el valor de la consigna en el módulo Habitación con el propósito de enseñar al usuario la temperatura del área en la que se encuentre. Los componentes encargados de realizar esta aplicación serán dos displays siete segmentos. Estos dispositivos mostrarán un cierto número dependiendo del estado de sus entradas teniendo una entrada por cada led que posee.

Entre el Arduino y cada uno de los displays se colocará un decodificador de BCD a siete segmentos, 74LS48. Este circuito integrado, a partir de un número en BCD aportará siete salidas que modificarán su estado dependiendo del número introducido. Por lo tanto, se utilizarán cuatro salidas del Arduino en vez de siete que sería conectando el display directamente al Arduino por cada uno de los displays.

En el caso del primer display, el número máximo que se mostrará será el 3. Por lo tanto, simplemente se conectarán dos salidas del Arduino a las dos primeras entradas del integrado 74LS48 y conectando sus dos últimas a tierra, consiguiendo enseñar los tres números mencionados, sin necesidad de conectar las cuatro entradas de este integrado.

8.2.3 Sistema de Persianas

El sistema de persianas que se colocará en este hogar realizará las acciones de apertura y cierre de este elemento teniendo en cuenta dos modos de funcionamiento. En primer lugar, cuando no se enlace ningún pulsador con este sistema las acciones de apertura y cierre se realizarán de forma automática activando estas acciones a unas horas establecidas, en este caso se utilizarán las horas mencionadas anteriormente “hora de mañana” y “hora nocturna”.

Después, existirá otra posibilidad acoplando dos pulsadores a este sistema, uno para activar la apertura y otra para activar el cierre. El usuario tendrá que pulsar uno de los dos pulsadores activando la acción correspondiente y para detenerla deberá presionar de nuevo a ese botón. De esta forma, el sistema interactúa con el usuario colocando la persiana en la posición que pretenda.

Además, el sistema se ha diseñado para que los dos actuadores no se activen al mismo tiempo con el objetivo de no forzar el motor. Por lo tanto, cuando se esté realizando una de estas acciones y se presione el pulsador para accionar la operación contraria, el actuador que estaba en funcionamiento se desactiva y el actuador que corresponda con el pulsador que se ha presionado en último lugar se activa.

8.2.4 Sistema de ahorro de energía: XBee

El sistema de ahorro de energía de los dispositivos XBee se ha diseñado para conseguir el menor consumo de energía posible cuando los sistemas de control no precisen el uso de estos componentes. Este instante se producirá cuando no se detecte presencia en la zona transcurrido un periodo de tiempo después de que el sensor de movimiento no detecte presencia, es decir, que esta zona no está en uso y que los dispositivos XBee se deberán configurar en modo de ahorro de energía.

Este modo de funcionamiento será el "modo sleep" configurando este comando como "cyclic sleep", desactivando los dispositivos durante un periodo de tiempo y activándolo durante otro espacio de tiempo diferente. En este caso, el dispositivo estará desactivado durante 2 segundos configurados en el comando "SP" y activado en un periodo de 3 segundos establecidos en el comando "ST". De esta forma, se realizará un ahorro de energía cuando el área se encuentre vacía.

En este caso, se han configurado en este modo los dispositivos que se encuentran enlazados con este elemento excepto el propio módulo Habitación que debe de estar preparado para desactivar el "modo sleep" de los dispositivos cuando el usuario entre en el área que esté instalado este módulo.

Además, se ha añadido una función para conseguir un ahorro de energía en los módulos Habitación. El procedimiento se realizará en el momento que se activa el modo de ahorro de energía de los demás dispositivos, enviando una trama API al módulo Casa indicando que este módulo está preparado para entrar en "modo sleep". Este dispositivo entrará en este modo dependiendo de una serie de condiciones configuradas en el módulo Casa.

8.2.4.1 Configuración XBee

El XBee en este caso cumplirá la función de recibir y transmitir datos al tener conectado a él un Arduino que se encargará de leer los datos de los sensores que se encuentren en el módulo remotos y procesarlos. La comunicación entre el Arduino y el XBee se realizará mediante el puerto serial que tienen instalado ambos elementos. El puerto serial será un protocolo de comunicación que permitirá comunicar dos dispositivos transmitiendo la información en serie, bit a bit. Por lo tanto, a partir de esta comunicación se podrán enviar y recibir tramas API que será el formato que entienda el XBee para realizar el intercambio de información con otro dispositivo de este tipo.

La conexión que se tendrá que realizar para utilizar este tipo de comunicación será conectando los respectivos pines “Tx” y “Rx” de cada dispositivo con el opuesto del dispositivo contrario.

8.3 Módulo Pulsador

El módulo pulsador ha sido diseñado para que el usuario pueda interactuar con los diferentes sistemas de control. Los sensores que se implementarán en este módulo dependerán de los sistemas instalados.

En primer lugar, el sistema de luminarias y de persianas se podrá interactuar con ellos a partir de varios pulsadores táctiles instalado en este módulo. En segundo lugar, para el sistema de climatización se utilizará un potenciómetro para modificar su consigna.

El objetivo que se ha tenido en el proyecto ha sido crear esta serie de módulo con el menor coste posible, por tanto, en este módulo se ha prescindido de instalar un microcontrolador. La función que realiza este dispositivo se reemplazará con los recursos que pueden aportar los XBee capaces de leer los estados de los sensores y enviar esta información a un módulo concreto.

8.3.1 Configuración XBee

Este dispositivo será configurado como router teniendo la oportunidad de realizar las labores de enrutamiento entre dispositivos que esté fuera de su rango de transmisión creando una red más robusta.

Los XBee tendrán la posibilidad de leer tanto sensores digitales como sensores analógicos enviando una trama API conteniendo los estados de sus entradas. Existen dos métodos para realizar esta acción mediante dos comandos diferentes. Por una parte, el comando AT IR (IO sampling rate) enviando una trama API "IO Sample" cada cierto periodo de tiempo, indicando este tiempo en el valor del comando. Por lo tanto, este comando será necesario cuando se quieran transmitir valores de lecturas analógicas debido a que los sensores de este tipo se necesita conocer su valor constantemente. Por otra parte, el comando IC (digital IO change detection) detectará los cambios que se produzcan en sus entradas transmitiendo una trama API del mismo tipo que la anterior, pero en vez de enviarla de una manera periódica se realizará cada vez que una de sus entradas cambie de estado.

A continuación, se indicará la configuración de cada comando AT utilizado y donde se colocará cada sensor siendo siempre la misma ubicación independientemente si se implementan todos los sistemas de control o uno solamente.

- **Comando AT D0:** este comando tendrá el valor 3 indicando que este pin será una entrada digital. En esta ubicación se instalará un pulsador para ser enlazado con el sistema de luminarias.

- **Comando AT D1:** este pin se configura para ser una entrada digital siendo su valor 3. En este pin se conectará el pulsador correspondiente a la acción de apertura del sistema de persianas.

- **Comando AT D2:** en pin D2 se indicará que es una entrada digital. Este pin se complementará con el anterior conectando el pulsador que realice la acción de cierre de la persiana.

- **Comando AT D3:** en esta entrada se conectará el potenciómetro para interactuar con el sistema de temperatura. Por lo tanto, se configurará como entrada analógica siendo el valor de este comando 2.

- **Comando AT D5:** en la entrada siguiente se obtendrá la posibilidad de conectar un relé con la intención de unificar este módulo con un módulo actuador que se mostrará a continuación. Por lo tanto, este pin se configura desde un módulo remoto como salida digital a nivel bajo o alto.

- **Comando AT IR:** en caso de instalar de instalar el sistema de climatización será necesario activar este comando para conocer el valor del potenciómetro constantemente. La trama API se enviará cada 300 ms introduciendo un valor de 0x12C, obteniendo así una respuesta suficientemente rápida para que el usuario no detecte retrasos cuando utiliza los dispositivos del módulo.

- **Comando AT IC:** este comando, se utilizará cuando no se implemente la interacción con el sistema de temperatura teniendo únicamente sensores digitales. Este comando, se activará colocando un valor superior a 0.

8.4 Módulo Actuador

El siguiente módulo se ha planteado para colocar los actuadores que realizarán el control de los sistemas de luminarias, persianas y climatización debido que los tres sistemas emplearán el mismo actuador, pero en cantidades diferentes. En su interior, se instalarán uno o dos relés dependiendo de si se realiza simplemente para el sistema de luces que requiere uno o en cambio, para los sistemas de persianas y temperatura que ambos métodos llevan instalados dos actuadores.

En el sistema de luces se realizará el control directo sobre las luminarias que se quieren controlar conectándolas al relé que se instale en este módulo. Por otra parte, la automatización de las persianas requerirá dos actuadores, uno por cada sentido de giro del motor que se ubicará en este elemento. Por último, las funciones de temperatura se realizarán haciendo el control sobre el conjunto de actuadores que forman el sistema de climatización, necesitando dos relés instalados para el aire acondicionado y el calefactor.

Posteriormente, este módulo carecerá de microcontrolador siendo el XBee el encargado de realizar el control de los relés.

8.4.1 Configuración XBee

El XBee en este caso, estará configurado como un dispositivo final, con el fin de tener el menor consumo de energía posible eliminando las tareas de enrutamiento del protocolo ZigBee. Los relés irán conectados a las salidas digitales de este dispositivo variando su valor para activar y desactivar el relé correspondiente. El módulo habitación, será el encargado de modificar las salidas del XBee mediante una trama API, "Remote AT command". A continuación, se muestran los comandos configurados en el presente módulo:

- **Comando AT D0:** este comando corresponderá con el pin número 20 del XBee conectando el relé determinado para el sistema de luminarias.

- **Comando AT D1:** el siguiente comando se empleará para los sistemas de persianas y climatización. En el primer caso se conectará el relé que determine la acción de apertura de las personas. En segundo lugar, el aire acondicionado será la alternativa a la anterior aplicación.

- **Comando AT D2:** semejante al comando previo se realizarán las mismas aplicaciones, pero con el propósito de aplicar las acciones opuestas de los sistemas de persianas y climatización.

8.4.2 Conexiones: Carga

Además, es necesario especificar las conexiones que tendrán estos actuadores con las cargas correspondientes. En primer lugar, las luminarias del hogar requerirán tres conexiones, fase, neutro y tierra para ser accionadas. Estos elementos, se activarán de forma convencional mediante un interruptor manual que se situará entre dos puntos de la fase que se conecte a la luminaria cortando o cerrando el circuito para activarla. Por lo

tanto, para el control de luminarias se sustituirá el interruptor convencional por un relé conectando entre dos puntos de la fase.

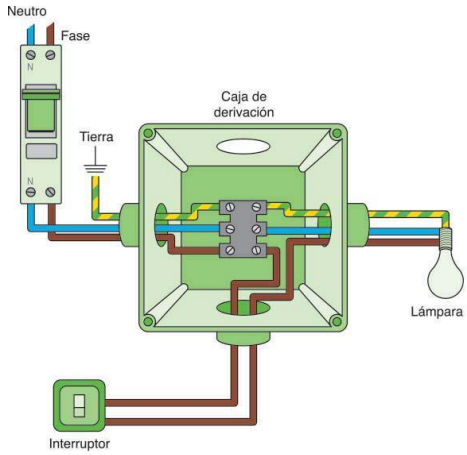


Figura 23.

En segundo lugar, los dispositivos del sistema de climatización, los cuales irán conectados a los circuitos de fuerza del hogar. Estos circuitos, estarán compuesto por las conexiones de fase, neutro y tierra en enchufes en formato “schuko”. El control, se realizará colocando un relé entre dos puntos de la fase haciendo la función de un interruptor.

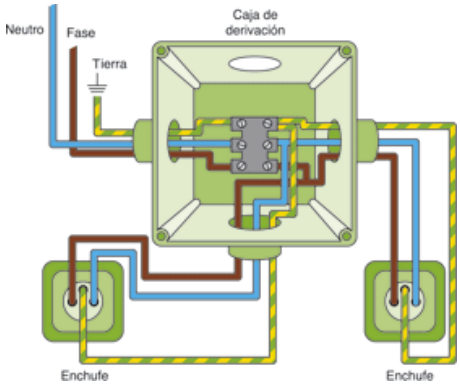


Figura 24.

8.5 Módulo de movimiento

El módulo de movimiento ha sido creado por las dimensiones de la sala. Al tener unas dimensiones críticas para los sensores de movimiento elegidos se ha optado por

colocar un sensor de movimiento en otra zona de la sala y así abarcar todos los espacios que pueden ser no detectados al poner un único módulo con este tipo de sensores.

En este dispositivo se instalará un sensor de movimiento conectado a un XBee sin necesidad de colocar un microcontrolador para realizar la lectura del sensor. El XBee se encargará de leer el estado de este instrumento y enviar una trama API cuando el sensor de movimiento detecte presencia.

8.5.1 Configuración XBee

Este módulo se considerará como dispositivo final, para eliminar las tareas de enrutamiento que tiene implementado el protocolo ZigBee obteniendo un menor consumo.

La función del XBee en este caso será la de enviar un dato digital hacia otro dispositivo. Esta acción se realizará mediante el comando IC, que dispone este instrumento de comunicación enviando una trama API cuando el estado del sensor de movimiento varía tanto a nivel alto como a nivel bajo.

Las entradas y comandos que se han utilizado para realizar esta tarea han sido los siguientes:

- **Comando AT D0:** este pin será configurado como entrada digital conectando el sensor de movimiento.

- **Comando AT IC:** esta instrucción será necesario activarla para enviar la trama "IO Sample" cuando el sensor cambie de estado. La activación se realizará colocando un valor superior a 0 en este comando.

8.6 Módulo casa

El módulo siguiente se ha proyectado para realizar el sistema de ahorro de energía de los módulos XBee de toda la casa. La intención de este este método será activar el

modo sleep de los módulos Habitación de todo el hogar cuando la casa se encuentre vacía y desactivando dicho modo cuando accedan los usuarios al domicilio.

Este dispositivo, estará compuesto por un Arduino y un XBee. El microcontrolador será necesario para decidir cuándo se active y desactive el modo sleep en los módulos mencionados. Los módulos habitación, enviarán una trama API "transmit request" precisando que no hay personas en la zona en la que estén instalados. En el momento, que todos los módulos habitación transmitan esta información la casa se considerará vacía activando el modo sleep de estos dispositivos.

La ubicación de este módulo será en la entrada del domicilio, con el objetivo de detectar a los usuarios en el momento que accedan a la vivienda. Por lo tanto, en este módulo se volverá a instalar un sensor de movimiento para realizar la aplicación que se ha descrito.

8.6.1 Configuración XBee

Este XBee será configurado como coordinador siendo el encargado de crear la red de comunicación entre los distintos XBee y conceder el permiso a los nuevos dispositivos para unirse a la red.

La conexión con el Arduino se realizará, mediante la comunicación serial que tienen ambos dispositivos conectando sus respectivos pines "Tx" y "Rx" con los opuestos en el dispositivo contrario. De esta forma, el Arduino será capaz de procesar las tramas API que se recibirán de los módulos habitación.

8.7 Módulos instalados

A continuación, se mostrarán los módulos instalados en cada una de las zonas del domicilio, plano número 2. Los diferentes sectores serán descritos indicando las configuraciones correspondientes a los requisitos del cliente. En las especificaciones de cada módulo mostradas anteriormente se han expuesto todas las posibilidades que puede tener cada uno de ellos. Dependiendo de las especificaciones que quiera el usuario en

cada zona se adaptará cada módulo a estas condiciones para reducir a su vez el precio de cada uno y con la intención de que no contengan recursos que no utilicen.

8.7.1 Entrada

En la entrada del domicilio se instalarán dos tipos de módulos con el objetivo de realizar un control de luminarias automático. En primer lugar, en este espacio se ubicará el módulo Casa el cual será necesario por las características de este módulo. El esquema eléctrico corresponderá con el plano número 3, implementando únicamente los sensores necesarios para el sistema de luminarias.

En segundo lugar, el módulo actuador complementa al primer módulo para realizar el control especificado. El esquema eléctrico contemplará los dispositivos básicos de este módulo con un relé para realizar el sistema de iluminación, plano número 4.

8.7.2 Cocina

En el siguiente espacio, se colocará un módulo Habitación para procesar los sensores y actuadores que complementen la automatización de este espacio, mostrando su configuración en el plano número 5.

Después, en la zona presente se instalará un módulo Pulsador con el objetivo ejecutar el control de luminarias mediante un relé que se encuentren en su interior, además de dos pulsadores necesarios para manipular las persianas.

Además, la distancia entre el actuador de luminarias y persianas requiere implementar un módulo Actuador con el objetivo de completar el sistema de personas, compuesto por los componentes que se describen en el plano número 7.

8.7.3 Sala

En esta área de la vivienda, ha sido necesario un módulo Habitación con el propósito de implementar los tres sistemas de control especificados.

A los actuadores se refiera se han instalado dos módulos Actuadores con el objetivo de realizar el control de los diferentes sistemas. El módulo correspondiente al plano número 11, se ha configurado para el control de luminarias y persianas mientras que el módulo Actuador restante completa el sistema de climatización de esta zona, especificado en el plano número 12.

Por otra parte, se colocará un módulo Pulsador para interactuar con los sistemas. En su interior, se encontrarán tres pulsadores con la propiedad de manipular las luminarias y persianas de la sala. Además, se ha implementado un potenciómetro con la intención de que el usuario sea capaz de modificar la temperatura de esta zona.

También, por las dimensiones de la sala ha sido necesario instalar un sensor de movimiento de forma remota para poder detectar todos los espacios de esta área. Los esquemas eléctricos de los módulos correspondientes están representados en el plano número 10.

8.7.4 Zonas comunes

Las zonas comunes de la casa estarán constituidas por el balcón, pasillos y baños. En estas zonas se instalarán dos tipos de módulos con las mismas configuraciones. En primer lugar, se instalarán los módulos Habitación para ejecutar el sistema de luminarias que ha requerido el usuario.

En segundo lugar, el módulo Actuador necesario para cerrar la automatización de las luminarias de estas zonas sustituyendo los interruptores que controlan las luminarias de cada zona por estos módulos. En el plano número 14, se indican los esquemas eléctricos correspondientes.

8.7.5 Habitaciones

En las tres habitaciones se implementará un módulo Habitación para procesar los sistemas de control diseñados. En este caso, el módulo Habitación estará compuesto por los sensores necesarios para realizar todos los sistemas diseñados.

Posteriormente, los módulos Pulsadores completarán los sistemas de control para que el usuario puede interactuar con ello, manipulando las magnitudes correspondientes. Además, se ha incluido en su interior un relé para realizar el control de las luminarias. Por lo tanto, es necesario ubicar este módulo en los interruptores de cada habitación.

También, se instalarán dos tipos de módulos Actuadores. Por un lado, para realizar el control de las persianas a través de los dos relés conectados al módulo XBee. Por otro lado, será necesario colocar otro módulo de este tipo en cada uno de los sectores para completar el sistema de climatización, debido a la distancia que existe entre los instrumentos a controlar.

8.8 Programación

El desarrollo del sistema domótico se ha realizado en el entorno de programación IDE Arduino. En este ámbito se redactarán las instrucciones necesarias para la configuración de los pines del Arduino y desarrollar la lógica de un programa concreto.

En este caso, con la intención de tener un programa más sencillo y estructurado se han diseñado una serie de librerías en el lenguaje de programación C++, que se implementarán en el IDE Arduino para ser guardadas en la memoria del microcontrolador.

Las librerías se han creado con una programación orientada a objetos con el objetivo de tener una programación más estructurada y a su vez sencilla. En ellas, se diseña cada objeto almacenado en el módulo Habitación y el módulo Casa obteniendo las funciones de cada sensor implementado.

8.8.1 Clases

Las clases que se han proyectado son provenientes de los sensores que se ubican en los módulos, creando métodos para conocer cierta información de estos dispositivos. Además, se habrán creado una clase “Habitación” y “Casa”, donde utilizarán estos objetos para desarrollar la lógica de cada uno de los sistemas de control diseñados.

También, han sido diseñadas ciertas clases auxiliares para la utilización de las herencias que ofrece la programación orientada a objetos. La herencia acogerá los atributos y métodos de la clase superior aportando la posibilidad a los objetos heredados a utilizar estas propiedades.

8.8.2 Sensores

Las primeras clases que se han creado corresponderán con los sensores de luz, movimiento y temperatura. En este caso, los sensores necesitarán saber en qué pin del Arduino están conectados y a su vez configurar este pin como entrada. Por lo tanto, se ha decidido crear una clase madre, "Sensor", con el objetivo de almacenar el valor de un pin y además tenga la posibilidad de configurarlo a través de un método. Los sensores del módulo heredarán de esta clase "Sensor" obteniendo sus atributos y métodos.

- **Atributos:**

uint8_t _Pin: Almacena el número del pin donde esté conectado el sensor.

- **Métodos**

void Add (uint8_t Pin): este método asigna el valor introducido como "Pin" al atributo "_Pin" y lo configura como entrada. La función pinMode, será la encargada de configurar un pin tanto como entrada como salida dependiendo de los valores que se introduzcan a la función.

- Clase LDR

El sensor de luz se ha implementado mediante la clase "LDR". Este instrumento, dependerá de la luz que incide sobre la fotorresistencia produciéndose picos de tensión en ciertos instantes de tiempo, por las características de la resistencia que hacen inestable el sistema. La solución que se ha planteado ha sido realizar una media con un número determinado de muestras y trabajar con los valores de esta media. Esta solución,

despreciará los valores elevados que se han comentado obteniendo un valor constante cada número de muestras establecido.

- **Atributo:**

double _Suma: en este atributo se acumularán los valores leídos por el pin del sensor de luz. La función capaz de leer sensores analógicos será analogRead (uint8_t Pin) introduciendo como parámetro el pin que se pretende leer.

uint8_t _Muestras: será un contador que aumente su cuenta cada muestra adquirida y se resetea en el momento que se alcancen las 50 muestras.

uint16_t _Media: esta variable será la encargada de almacenar el valor calculado de la media siendo actualizado cada 50 muestras.

- **Métodos**

bool Oscuridad (double Um): el siguiente método devolverá un valor booleano dependiendo de la variable Um, introducida como parámetro. Esta variable indicará el umbral de luz que muestre si la zona donde esté el sensor se encuentra en un ambiente oscuro o por lo contrario el área esta iluminada. En este caso, si el atributo "_Media", está por encima de este umbral se devolverá un 0 señalando que existe iluminación, pero si está por debajo se devolverá un 1 advirtiendo que la zona necesita ser iluminada, figura 25.

```
14  bool LDR::Oscuridad(double Um){
15
16  uint16_t Lectura = analogRead( _Pin );
17
18  _Suma = _Suma + Lectura;
19
20  _Muestras++;
21
22  if ( _Muestras >= Muestras ){
23
24      _Media = _Suma / Muestras;
25
26      _Muestras = 0;
27
28      _Suma = 0;
29
30    }
31
32  if ( _Media < Um ){
33
34      return 1;
35
36    } else{
37
38      return 0;
39
40      }
41
42
43
44  }
```

Figura 25.

- Clase SenMovi

Por otra parte, el sensor de movimiento se verá reflejado en la clase “SenMovi”. Esta clase carece de atributos al heredar la variable necesario de la clase “Sensor”, utilizando simplemente un método que sea capaz de mostrar la salida del sensor.

- **Métodos:**

bool Salida (): este método, leerá los valores del sensor digital obteniendo el estado del instrumento. El valor 0 señala que no se ha detectado presencia y el valor 1 si ha detectado presencia.

- Clase Temperatura

Después, el sensor de temperatura se representará mediante la clase “Temperatura”. El Arduino tendrá un conversor analógico - digital de 10 bits con una

tensión de referencia de 5 voltios. Por lo tanto, la lectura de este sensor analógico se deberá escalar a partir del número de valores que tenga el CAD en este caso 1024 y la tensión de referencia.

El valor de temperatura se obtendrá mediante la sensibilidad del sensor en conjunto a la lectura obtenida por el Arduino siendo de 10 mV / ºC.

- Métodos:

double Valor (): el siguiente método devolverá la temperatura a la que se encuentra el sensor realizando la conversión necesaria para obtener este parámetro, figura 26.

```
#include "Temperatura.h"


Temperatura::Temperatura(){

}

double Temperatura::Valor(){

  return ( ( analogRead(_Pin) / 1023 ) * 5000 ) ; //Sensibilidad 10 mV

}

```

Figura 26.

8.8.3 Reloj

La clase “Reloj”, se ha decidido crear por la necesidad de contabilizar el tiempo en el interior del microcontrolador y además el objeto será capaz de conocer la hora y minutos del día a través del reloj de tiempo real.

Arduino tendrá una función capaz de contabilizar el tiempo, millis (), devolviendo esta función el tiempo en milisegundo desde que se alimente el Arduino, desbordándose este contador a los 50 días. Entonces, se utilizará esta función en el interior de esta clase, con el objetivo de realizar la temporización de un tiempo determinado.

- **Atributos:**

bool _TempActivo: está variable indicará si el temporizador está activado.

double _TiempoFinal: se almacena la suma del valor millis () en un determinado tiempo y el tiempo que se quiere temporizar.

RTC_DS3231 RTC: este atributo, será una instancia de un objeto definido específicamente, para el reloj de tiempo real que se está utilizando en este proyecto. El objeto inicializará la conexión del reloj con el Arduino, además aportará los datos horarios que se necesitan de este sensor.

- **Métodos:**

bool Temporizador (double tiempo): el método contabilizará el tiempo determinado por el parámetro que se le introduce a la función, "tiempo", devolviendo un 1 cuando se cumpla esta cuenta.

En el momento que se utiliza este método el atributo de este objeto, "_TiempoFinal", se asignará el valor de la función millis () en ese instante, y se le sumará la variable que se le ha pasado a la función, "tiempo". Posteriormente, se compara "_TiempoFinal" con el valor de la función millis (), que irá aumentando a medida que pase el tiempo. En el instante que el valor de millis (), sea superior a "_TiempoFinal", el temporizador se resetea y este método devolverá un 1. En cualquier otro caso, la función devolverá un 0.

void TempReset (): la función se resetea el Temporizador, fijando un nuevo valor a "_TiempoFinal".

bool GetTempActivo (): devolverá un 0 si el temporizador está desactivado y un 1 si el temporizador está activado.

bool Horario (uint8_t hora, uint8_t minuto): el siguiente método utilizará el objeto instanciado RTC_DS3231, para conocer la hora y los minutos en el momento que se llame

a la función. Esta función, devolverá un 1 cuando la hora y minutos actuales, sean mayores a la "hora" y "minutos" incluidas a la función. En cualquier otro caso, la función devolverá un 0.

8.8.4 Display

La siguiente clase ha sido diseñada para las ocasiones que se quiera instalar en el módulo Habitación dos displays para mostrar la temperatura a la se quiere regular la zona.

- **Atributos:**

uint8_t _Pines [4]: este atributo, estará compuesto de un array de 4 elementos. En este array se almacenarán los pines correspondientes a las 4 salidas que se necesitan para activar el display mediante el decodificador 74LS48.

- **Métodos:**

void Add (uint8_t A, uint8_t B, uint8_t C = 0, uint8_t D = 0): en este método, se indicarán los pines de cada display que se instancia. Las variables C y D utilizarán una de las mejoras que se implementaron cuando se creó el lenguaje C++ respecto a C, inicializando estas variables a cero en el caso de que no se le asignen ningún valor cuando se llame a esta función.

La razón por el que se ha creado esta función con este formato es debido a que en el proyecto se instalará un display controlado solamente con dos pines, conectando los dos restantes a tierra. Por lo tanto, esta función será más flexible teniendo la posibilidad de conectar dos o cuatro pines indistintamente.

Entonces, en esta función se le asignará el Pin correspondiente al atributo donde se almacenan estas variables. Además, si se ha asignado un número a los pines distinto de 0 se configurarán como salidas digitales.

```
#include "Display.h"

Display::Display(){

}

void Display::Add( uint8_t A, uint8_t B, uint8_t C = 0, uint8_t D = 0 ){

  _Pines[3] = A;
  _Pines[2] = B;
  _Pines[1] = C;
  _Pines[0] = D;

  for (size_t i = 0; i < sizeof( _Pines ) ; i++) {

    if ( _Pines[i] != 0 ) {

      pinMode( _Pines[i], OUTPUT );

    }

  }

}

```

Figura 27.

void MostrarNum (uint8_t Num): Esta función está preparada para mostrar el número que se le indique por parámetro (Num) en el display. En su interior, estará implementado una matriz 10x4 señalando cada número en formato BCD. Por lo tanto, se recorre la fila del número que se indique y se colocarán los niveles de tensión a los pines correspondientes.

```
25
26  void Display::MostrarNum( uint8_t Num ){
27
28    uint8_t  Digitos[10][4] =
29
30          {
31             { 0,0,0,0 },   // 0
32             { 0,0,0,1 },   // 1
33             { 0,0,1,0 },   // 2
34             { 0,0,1,1 },   // 3
35             { 0,1,0,0 },   // 4
36             { 0,1,0,1 },   // 5
37             { 0,1,1,0 },   // 6
38             { 0,1,1,1 },   // 7
39             { 1,0,0,0 },   // 8
40             { 1,0,0,1 }    // 9
41          };
42
43    //El siguiente for recorrera dos vectores:
44    //el vectores de pines y el vector del número que se Indica a la funcion (Num)
45
46    for (size_t i = 0; i < 4; i++) {
47
48      uint8_t Valor = Digitos[Num][i];
49
50      if ( _Pines[i] != 0 ) {
51
52        digitalWrite( _Pines[i], Valor );
53
54      }
55
56    }
57
58  }
```

Figura 28.

8.8.5 XBee

En el apartado de los XBee se utilizará una librería preparada para emplear estos dispositivos con Arduino. Las funciones que almacena en su interior como recibir y transmitir tramas API han sido de utilidad para diseñar estas nuevas librerías facilitando el trabajo en este aspecto se refiere.

En esta librería en concreto se deberá configurar el XBee con un parámetro en específico. El comando AT “AP” será el comando que se necesite modificar, configurando este parámetro en su segunda opción modo API 2.

A continuación, se mostrarán los objetos almacenados en la librería XBee que se utilizarán en las librerías que requieran este dispositivo.

- Clase XBee

El objeto XBee se encargará tanto para adaptar el XBee al Arduino como para la recepción y transmisión de tramas API. Las siguientes funciones serán las encargadas de realizar estas acciones:

void begin (stream &serial): se transmitirá al método un puerto serial del Arduino para comunicarse con el XBee a través de sus pines Rx y Tx.

void readPacket (): La siguiente función será la encargada de leer una trama API bit a bit. Esta función cada vez que lee una trama sobrescribirá a la anterior.

void readPacket (int timeout): similarmente a la anterior función se leerá una trama API, pero se espera el tiempo establecido como "timeout" para recibir una trama una vez cumplido este tiempo se saldrá de esta función sino se ha recibido ninguna trama.

void getResponse (XBeeResponse &response): En esta función se le cargará a la función la referencia de un objeto "XBeeResponse" introduciendo en este objeto las características de la trama que se ha recibido por medio de las dos funciones anteriores.

void send (XBeeRequest &request): el objetivo de este método será enviar la trama asignada por la variable "request" por el puerto serial del XBee.

- Clase XBeeAddress64

La siguiente clase, se encargará de guardar las direcciones de 64 bits que llevan implementado los XBee. Por lo tanto, se tendrán funciones tanto para introducir la dirección a este objeto como para leer dicho parámetro.

- Clase XBeeResponse

En esta clase se almacenarán todas las propiedades que tengan las tramas recibidas por el XBee. Esta clase se utilizará para cargar a las clases de las tramas API específicas, las características correspondientes a esas tramas.

void getZBRxIoSampleResponse (XBeeResponse &response): se le introducirá por parámetro la referencia de un objeto del tipo XBeeResponse con la intención de asignar en este objeto las propiedades que tiene una trama “IO Sample”.

void getZBTxStatusResponse (XBeeResponse &response): de forma similar a la anterior función trabajará este método, pero en este caso asignará las variables correspondientes de una trama “TxStatus”.

- Clase ZBRxIOSampleResponse

El objeto “ZBRxIOSampleResponse” almacenará las propiedades de una respuesta “IOSample” teniendo la posibilidad de emplear las siguientes características a través de estas funciones:

bool isDigitalOn (uint8_t pin): esta función será la encargada de conocer el estado de las entradas del XBee que ha enviado una trama “IO Sample”. Entonces, para conocer el estado de una entrada en concreto dependerá de la variable “pin”.

bool getAnalog (uint8_t pin): esta variable devolverá el valor de los pines analógicos que se encuentren activos en el XBee transmisor. La variable “pin” será la encargada de variar la entrada que se pretende conocer.

XBeeAddress getRemoteAddress64(): este método se heredará de la clase “ZBRxResponse”, obteniendo de él la dirección de 64 bits de la trama “IO Sample” que se quiere analizar.

- Clase ZBTxRequest

Esta clase contendrá las variables necesarias para enviar una trama API “Tx”, las cuales serán la dirección de destino y los bytes que se quieran transmitir en formato

hexadecimal. A su vez, este objeto se complementará con la función "send" enviando esta trama por el puerto serial del XBee.

- Clase RemoteAtCommandRequest

La siguiente función, hará referencia a las tramas "AT Remote Request". Este tipo de tramas modificarán los comandos de un XBee remoto. Esta clase almacenará la dirección de destino, el comando que se pretende modificar y el valor de este comando.

Las acciones mencionadas se realizarán por medio del constructor de esta clase.

RemoteAtCommandRequest (XBeeAddress64 &remoteAddress64, uint8_t *command, uint8_t *commandValue, uint8_t commandValueLength): en este constructor se asignan las características mencionadas anteriormente. La dirección de destino se asignará por referencia mientras que el comando y su valor se transmitirán a esta función por medio de punteros.

8.8.6 Dirección

La clase "dirección" ha sido creada para ser una clase madre de los objetos que se encuentren de forma remota respecto al módulo Habitación o módulo Casa, que necesitan una dirección para identificar el XBee que lleven instalado. Heredando, el atributo con la dirección específica y los métodos para poder modificar esta dirección.

En el objeto "Dirección" se instancia un objeto "XBeeAddress64" proveniente de la librería que se ha mencionado con anterioridad de los dispositivos XBee. En el objeto instanciado se podrá definir un número de 64 bits mediante un número compuesto por 64 bits o por dos números de 32 bits cada uno Msb y Lsb.

- Atributos:

XBeeAddress64 _address: este objeto almacenará el número de 64 bit que constituirá la dirección del XBee remoto.

- Métodos

void Add (uint32_t addressMsb, uint32_t addressLsb): este método será una de las posibilidades para añadir la dirección del XBee remoto. En esta función se le introducirá los bits msb y lsb, asignándoselo al atributo de esta clase.

void Add (uint64_t address): esta función será similar a la anterior, pero diferenciando las variables que se le pasan a la función, donde se le pasará una variable de 64 bits para formar el atributo correspondiente.

XBeeAddress64 getAddress64(): este método devolverá el atributo "_address" almacenando su dirección.

8.8.7 Actuador

La siguiente clase que se expone representa a los dispositivos remotos que activen los circuitos de potencia. En su interior se representará las características de un dispositivo de este tipo añadiendo determinadas características para modificar su estado.

El presente objeto al ser un componente que se instale a distancia del módulo Habitación se le asignará una dirección de destino con la posibilidad de modificar sus propiedades a distancia. Por lo tanto, este objeto hereda de la clase "Dirección" con el propósito de obtener sus atributos.

- Atributos:

bool _Estado: en esta variable se almacena el estado del actuador. El valor de 0 indica que esta desactivado y el valor de 1 que se encuentra accionado.

bool _Enviar: este atributo, se utilizará cuando se quiera modificar el estado del actuador de forma remota indicando esta variable que este actuador está preparado para enviar trama API para cambiar su estado.

uint8_t _cmd [2]: el array creado con dos posiciones determinará la entrada del XBee remoto al que esté conectado. Por lo tanto, almacenará el comando correspondiente a la entrada de este dispositivo, siendo más flexibles este instrumento para sus posibles modificaciones en los diferentes módulos.

- Métodos:

void setCmd (uint8_t cmd [2]): se coloca el comando adecuado para este actuador.

void Act (): le asigna el valor 1 a la variable "_Estado" indicando que el actuador está activado.

void DesAct (): asigna el valor 0 al atributo, "_Estado" señalando que el actuador está desconectado.

void CambiarEstado (): cambia su estado al valor contrario.

void Enviar (): indica que a este actuador se le puede enviar una trama API con el objetivo de modificar su estado. Asignándole un valor de 1, a la variable "_Enviar".

void NoEnviar (): se le coloca un valor de 0 a el atributo "_Enviar" señalando que no está que no es necesario modificar el estado de este actuador.

uint8_t GetCmd (bool iterator): devuelve valor uint8_t del comando correspondiente al iterador.

bool GetEstado (): devuelve la variable "_Estado".

bool GetEnviar (): retorna el atributo "_Enviar".

8.8.8 Persianas

Las persianas de una habitación, por ejemplo, serán un elemento remoto al módulo Habitación donde este objeto tendrá una serie de características. Al ser un módulo que se

encuentra a distancia requerirá un XBee, es decir, esta clase necesitará heredar el atributo y los métodos de la clase “Dirección”.

Este objeto, almacenará dos instancias de la clase Actuador, debido a que el control del módulo remoto correspondiente a la persiana se realizará por medio de dos relés. En el interior de esta clase existirán métodos que devuelvan estos “Actuadores” con la intención de trabajar con las funciones de este objeto independientemente.

```
#ifndef Persiana_h
#define Persiana_h

#include "Arduino.h"
#include "Reloj.h"

#include "Actuador.h"

#include "Direccion.h"

class Persiana : public Direccion {
private:

  //Actuadores para bajar y subir la Persiana
  Actuador _PerS;
  Actuador _PerB;

public:

  Persiana ();

  //Los dos métodos devuelven los actuaoderes
  Actuador& getASubir();
  Actuador& getABajar();

};

#endif
```

Figura 29.

- Atributos:

Actuador _PerS: este atributo corresponde a un “Actuador” realizando la función del relé que acciona el sentido de apertura del motor de la persiana.

Actuador _PerB: este actuador será similar al anterior, pero haciendo referencia al relé que ejecuta la acción de cierre de la persiana.

- Métodos:

Actuador& getASubir (): en este método se devolverá el actuador de subida (_PerS). El símbolo "&", será que se estará devolviendo la referencia del objeto, es decir, su posición de memoria.

Actuador& getABajar (): similar a la anterior función, devolviendo la referencia del actuador de bajada (_PerB).

8.8.9 Climatización

Los sistemas de climatización se reflejarán en esta clase donde almacenará los dos actuadores que necesita para realizar el control especificado. Al estar situado en un módulo remoto necesitará una dirección para comunicarse con este módulo, heredando como el objeto anterior la clase "Dirección".

A continuación, se mostrarán los atributos y métodos correspondientes a esta clase.

- Atributos

Actuador& _AFrio: el siguiente atributo hará referencia al relé que acciona el aire acondicionado.

Actuador& _ACalor: este atributo será el opuesto al anterior conectado la calefacción de este sistema.

- Métodos

Actuador& getAFrio (): esta función se encargará de devolver el atributo "_AFrio" devolviendo su posición de memoria mediante el símbolo "&" en vez de una copia del objeto.

Actuador& getACalor (): el método presente devolverá la posición de memoria del atributo "_ACalor".

8.8.10 ArduVector

ArduVector, será una librería que es capaz de trabajar con vectores en Arduino. Esta librería sustituirá a los vectores correspondientes de la librería estándar de C++. En las siguientes librerías los vectores de objetos serán necesarios para hacer al sistema más completo y robusto.

Las funciones que se implementarán en las demás librerías serán:

Push_back (elemento): añadirá a la última posición del vector el elemento que se le pase por parámetro.

Size (): devolverá el tamaño del vector.

Reserve (uint8_t tamaño): en esta función se le indicará al vector el tamaño máximo que puede tener a través de la variable "tamaño".

8.8.11 Habitación

La clase "Habitación" almacenará las instancias de los objetos que se han descrito anteriormente y los métodos que tendrán implementados la lógica para cada sistema de control.

La programación de los sistemas de luminarias y persianas se ha desarrollado para trabajar en dos modos diferentes, en cambio el sistema de climatización se ha creado con un único modo de funcionamiento.

En primer lugar, los sistemas de luminarias y persianas incluirán un modo automático y un modo manual, dependiendo si se ha añadido a estos sistemas un módulo

Pulsador. En segundo lugar, en sistema de climatización será necesario enlazarlo con un módulo Pulsador para modificar la consigna de este sistema.

A continuación, se mostrarán dos funciones diseñadas en la librería para enviar una trama API "Remote AT Command" y en la posterior una trama "Tx Status" con una serie de características. Después, se expondrán los atributos usados para cada sistema y posteriormente se analizarán los métodos diseñados para cada control.

Función:

bool EnviarTramaAT (XBee xbee, XBeeAddress64 address, Actuador A, uint8_t cmd [2]): la siguiente función ha sido diseñada para comprobar que se ha enviado con éxito una trama API "Remote AT Command" con unas ciertas características. Encapsulando las instrucciones necesarias de la librería "XBee" para enviar una trama API de este tipo haciendo el código más sencillo cuando necesite realizar esta acción.

La transmisión de una trama API se realizará a partir de un objeto existente de la librería "XBee", "RemoteAtCommandRequest". En esta clase, se asignará la dirección de destino, el comando a modificar y su valor, a partir de las variables que se transmiten a esta función. Después se enviará por el puerto serial del XBee esta trama mediante la función "send (XBeeRequest &request)", donde se tendrá que recibir una respuesta del XBee receptor, indicando si la trama se ha transmitido correctamente.

Entonces, la trama que se tendrá que recibir del XBee receptor se obtendrá por medio de la función readPacket (int timeout) descrita anteriormente asignando las propiedades de esta trama al objeto "RemoteAtCommandResponse".

En conclusión, esta función devolverá un 1 cuando la trama que se pretende enviar ha sido enviada con éxito. Esta acción se comprobará con la función "isOk" de la librería "XBee". En cualquier otro caso, se devolverá un 0.

```
bool EnviarTramaAT( XBee xbee, XBeeAddress64 address, uint8_t cmd[2],
                    uint8_t Value[1] ){

  RemoteAtCommandRequest remoteAtRequest;
  RemoteAtCommandResponse remoteAtResponse = RemoteAtCommandResponse();

  uint8_t value[1];

  value[0] = Value[0];

  remoteAtRequest = RemoteAtCommandRequest( address, cmd, value, sizeof(value));

  xbee.send(remoteAtRequest);

  if (xbee.readPacket(5000)) {

    if (xbee.getResponse().getApiId() == REMOTE_AT_COMMAND_RESPONSE) {

        xbee.getResponse().getRemoteAtCommandResponse(remoteAtResponse);


        if (remoteAtResponse.isOk()) {

          remoteAtRequest.clearCommandValue();

          return 1;

        } else {
          remoteAtRequest.clearCommandValue();

          return 0;

          }

      }

    }

}
```

Figura 30.

bool EnviarTramaTx (XBee xbee, XBeeAddress64 address, uint8_t payload [3]): la segunda función creada ha sido para enviar tramas “Tx Status”, necesitando implementar ciertos objetos de la clase “XBee”. Las clases que se han utilizado serán “ZBTxRequest” para almacenar la dirección de destino y los bytes que se quieren transmitir, y además “ZBTxStatusResponse” que corresponderá a la trama del XBee receptor cuando ha recibido esta trama enviada, conteniendo ella información sobre la transmisión de datos.

Por lo tanto, se crearán estos dos objetos mencionados con las variables que se introducen a la función, enviando esta trama a través de la función “send (XBeeRequest &request)”.

Después, la función “readPacket (int timeout)”, recibirá la trama almacenando en el objeto correspondiente a la respuesta de esta trama. Devolviendo un valor a nivel alto si la trama se ha enviado con éxito y un valor a nivel bajo si no se ha transmitido correctamente.

Atributos:

- Sistema de comunicación:

XBee xbee: este objeto será una instancia de la clase “XBee” obteniendo en él, las características de estos dispositivos para poder trabajar con ellos en Arduino.

ZBRxIoSampleResponse ioSample: el atributo será una instancia de un objeto “ZBRxIoSampleResponse”. En el momento que el XBee reciba una trama API “IO Sample”, se cargará a este objeto obteniendo de él todas las características de esta respuesta. Estas propiedades permiten conocer la dirección de 64 bits del XBee, además de los valores de sus entradas donde estarán los sensores que permitirán al usuario interactuar con los sistemas.

ArduVector<XBeeAddress64> _AddressModPulsador: la siguiente clase que se instancia es un vector que almacena direcciones de 64 bits (XBeeAddress64). En el interior de este vector se almacenan las direcciones de los “Módulo Pulsadores” que se quieran enlazar con este “Módulo Habitación”. En el momento que se reciba una trama API “IO Sample” se compara su dirección con las que están en este vector, en el caso de que coincida, habilitará a esa respuesta “IO Sample” a modificar las magnitudes de los sistemas que utilicen este módulo como enlace con el usuario.

De esta forma, se podrá duplicar las funciones de un módulo pulsador en lugares distantes en una misma zona. También se tendrá la posibilidad, si el usuario en algún futuro quiere instalar en alguna zona, más “Módulos Pulsadores” de los que existen el sistema estará preparado.

- Sistema de luz

ArduVector<XBeeAddress64> _AddressMovimiento: este objeto será un vector similar al anterior, pero almacena direcciones de XBee referidos a "Módulo de movimiento". Estos módulos se han creado en este proyecto para la sala del hogar, pero en la programación se ha diseñado para si en algún momento se quiere colocar un módulo de este tipo en cualquier otra zona, el sistema estará capacitado para añadirlo.

SenMovi _M1: el módulo habitación tendrá instalado dos sensores de movimiento, por lo tanto, se instancian dos objetos de la clase que se ha diseñado para este sensor "SenMovi". El atributo "_M1" será el primer sensor de este tipo que se implemente añadiendo posteriormente el pin a donde está conectado este sensor.

SenMovi _M2: de forma similar al anterior este objeto se referirá al segundo sensor de movimiento que tiene el "Módulo Habitación".

bool _Movimiento: esta variable indicará si existe movimiento en la habitación. Este atributo se utilizará en los sistemas que requieran la condición de presencia en la habitación para implementar su lógica.

LDR _L: el siguiente atributo corresponde a un objeto "LDR" creado para el sensor de luz que tiene instalado el "Módulo Habitación".

bool _Oscuridad: esta variable será similar a "_Movimiento" indicando esta vez si la habitación está poco iluminada. Esta variable será utilizada en los sistemas que necesiten condiciones lumínicas.

String _Modo: este atributo se ha colocado para identificar los diferentes modos en los que se puede encontrar a habitación. Este String se modificará para tener tres posibilidades, "Automático", "Manual" y "Nocturno", siendo el modo "Automático" el modo de funcionamiento por defecto.

Reloj _RModoF: el reloj se utilizará en un método que modifique los modos de funcionamiento que necesita condiciones temporales para su correcto funcionamiento.

bool _Encender: esta variable mostrará si al sistema si las luminarias se encuentran accionadas. Un valor de 0 señala que están apagadas, mientras que el valor contrario indica que están encendidas.

Reloj _ROnOff: el siguiente objeto se utilizará en el método que se encargue de modificar el estado de las luminarias, que necesita condiciones de tiempo.

bool Sample: la variable presentada será la que habilite al XBee en un método que recibirá tramas API "IO Sample", dando prioridad a otro tipo de tramas. Un valor de 0, no lo habilita, y un valor de 1, si.

bool _EstadoAnteriorLuz: esta variable guardará el estado de la entrada correspondiente al pulsador de Luz del "Módulo Pulsador". A partir de esta variable, se detectarán los flancos de subida de los pulsadores, ejecutando las acciones necesarias en un único ciclo.

bool _EstadoAnteriorPerS: similar a la función anterior, pero específico del pulsador de subir persianas.

bool _EstadoAnteriorPerS: semejante a las variables anteriores, pero haciendo referencia al pulsador de bajar persianas.

- Sistema de Temperatura

Temperatura _T: este objeto será una instancia de la clase creada para el sensor de temperatura, obteniendo las características de este sensor.

uint8_t _Consigna: en esta variable se guardará la consigna a la que se pretende regular el sistema de temperatura.

ArduVector<Climatización> Climatización: este vector almacenará objetos de la clase "Climatización". Este vector se ha diseñado para tener la posibilidad de añadir al sistema más sistemas de climatización en un mismo lugar. Orientado este vector, a instalar este

sistema en lugares espaciosos que necesiten, más de un sistema de climatización para regular la temperatura de una zona.

Display D1: el "Módulo Habitación" tendrá display en los casos que se quiera mostrar la consigna que habrá establecido el usuario. Por lo tanto, se aplicará una instancia al objeto creado para esta serie de aparatos. Este atributo corresponde al primer display.

Display D2: el siguiente objeto de forma similar al anterior, se crea para ejecutar las acciones del segundo display instalado.

char _Num1Anterior: este atributo de tipo char, se utilizará para comprobar el número anterior que se ha mostrado por el primer display.

char _Num2Anterior: similarmente a la variable anterior, pero respecto al segundo display.

- Sistema de persianas

ArduVector<Persiana> _Persianas: el vector almacenará objetos "Persianas" indicando todas las persianas que quieren que sea controlado por este módulo.

Reloj _RPer: este reloj se usará para contabilizar el tiempo, que esté activa la persiana y realizar las acciones correspondientes cuando se cumpla un determinado tiempo.

bool _EstadoAnteriroHoraN: esta variable se utilizará para detectar el flanco de subida, cuando se produzca una hora determinada del día, más concretamente, por la noche.

bool _EstadoAnteriroHoraM: esta variable tendrá la misma función que la anterior, pero detectará el flanco de subida de otra hora diferente, el cual será por la mañana.

- Ahorro de energía

Reloj _RPersonas: el reloj en conjunto al sensor de movimiento se utilizará para indicar si la zona no está ocupada por algún usuario.

bool _Personas: esta variable indicará, si la zona está ocupada o por el contrario se encuentra vacía.

bool _EstadoAnteriorPersonas: el atributo guardará el valor anterior a la variable anterior, "_Personas", detectando con ella los flancos de subida de esta variable.

Métodos:

- Inicializar objetos

Constructor: el constructor se invocará cuando se cree el objeto "Habitación" realizando las instrucciones necesarias para inicializar los atributos de esta clase. En este caso, se añadirán los pines de cada sensor que se encuentra en el módulo, con los métodos de la clase madre "Sensor". Posteriormente, a cada vector descrito anteriormente, se limitará su tamaño dependiendo del vector, con la intención de no saturar el sistema.

- XBee

void AddXBee (XBee Xbee): en la función siguiente, se añadirá al sistema el dispositivo XBee, copiando las características del objeto "Xbee", pasado por parámetro, a las del atributo de este mismo objeto.

void RecibirDatos (): esta función será la encargada de recibir las tramas API que lee el XBee. La función "readPacket" proveniente de la librería "XBee" obtendrá las tramas API, introduciéndolas en una respuesta que estará almacenada en el objeto XBee.

Posteriormente, se comprueba si coincide el tipo de trama que ha recibido el XBee, con una respuesta "IO Sample", cuyo ID es 0x92. En el caso ser semejantes, esta respuesta se introduce al atributo "ioSample" consiguiendo las características de este tipo de respuesta, figura 31.

```
157  /////////Método para recibir los datos de los módulos pulsadores///////////////
158
159  void Habitacion::RecibirDatos(){
160
161    if (Sample == 1) {
162
163      xbee.readPacket();
164
165      if ( xbee.getResponse().isAvailable() ) {
166
167        if ( xbee.getResponse().getApiId() == ZB_IO_SAMPLE_RESPONSE ) {
168
169          xbee.getResponse().getZBRxIoSampleResponse( ioSample );
170
171          }
172
173        }
174
175      }
176
177    }
178
179  }
180
```

Figura 31.

- Añadir Módulos remotos

void AddMPulsador (XBeeAddress64 address): este método añadirá una nueva "Módulo Pulsador" al vector que contenga las direcciones de esta serie de dispositivos. La acción se realizará por medio de la función "push_back ()", cuya función es agregar un nuevo elemento al vector.

void AddLuz (XBeeAddress64 address, uint8_t cmd [2]): la siguiente función incrementará los actuadores de luz que controle este módulo. Donde se deberá indicar su dirección remota y el comando, indicando el pin al que están conectados el XBee.

void AddMovmiento (XBeeAddress64 address): en este caso, se incluirá un nuevo "Módulo de Movimiento" incrementando el vector que contenga las direcciones de estos módulos en concreto.

void AddClimatizacion (XBeeAddress64 address, uint8_t cmdFrio[2], uint8_t cmdCalor[2]): el método siguiente incorporará un nuevo sistema de climatización al sistema, aumentando el vector correspondiente con esta nueva instancia.

void AddPersiana (XBeeAddress64 address, uint8_t cmdSubir [2], uint8_t cmdBajar [2]): de forma similar, incrementará el vector de "_Persianas" en un objeto más, añadiendo así una Persiana a la zona que se pretende instalar este sistema, siendo necesario incluir su dirección y los comandos donde estén conectados cada actuador.

```
208  //Método para añadir persianas a la Habitacion
209  void Habitacion::AddPersiana( XBeeAddress64 address, uint8_t cmdSubir[2],
210                                uint8_t cmdBajar[2]){
211
212    Persiana Per;
213    Per.Add( address.getMsb(), address.getLsb() );
214    Per.getASubir().setCmd( cmdSubir );
215    Per.getABajar().setCmd( cmdBajar );
216
217
218    _Persianas.push_back( Per );
219
220  }
```

Figura 32.

- Sistema de luz

void HMovimiento (): el siguiente método será el encargado de indicar a este módulo, si en el interior de la habitación o la zona en el que esté hay movimiento, modificando su atributo "_Movimiento".

Este procedimiento dependerá si hay un módulo de Movimiento remoto implementado en el sistema. En el caso de tener uno o varios módulos de este tipo, la condiciones para poder modificar el atributo "_Movimiento" serán las siguientes:

- El módulo remoto envía una trama API "IO Sample" analizando el estado de su entrada donde esté conectado el sensor. En el caso, de estar a nivel alto esta entrada modificará "_Movimiento" con un valor de 1 lógico.

- También, se podrá activar esta variable cuando la salida de los sensores de movimiento instalados en el módulo Habitación se active.

En el caso, de no tener ningún módulo de movimiento enlazado con este sistema solo dependerá de los estados de los sensores que se encuentran en el módulo Habitación.

void HOscuridad (): esta función será la encargada de modificar la variable, "_Oscuridad". Por medio, de la función "Oscuridad" de la clase "LDR" se conocerá el nivel de iluminación de la zona. En esta función, es necesario especificar un umbral de luz indicando si la zona está poco iluminada, si el nivel de luz está por debajo de este valor siendo ajustable este parámetro para cada zona.

```
281  void Habitacion::HOscuridad(){
282
283      uint16_t UmbralLuz = 600; //Indica el umbral de luz de la Habitacion
284
285      if ( _L.Oscuridad( UmbralLuz ) == 1 ) {
286
287        _Oscuridad = 1;
288
289      } else{
290
291          _Oscuridad = 0;
292
293        }
294
295        Serial.print( " Oscuridad:  ");
296        Serial.println( _Oscuridad );
297
298  }
```

Figura 33.

void ModoF (): el siguiente método se ocupará de variar los modos de funcionamiento del sistema de luz o lo que es lo mismo modificar el atributo "_Modo" siendo el modo de funcionamiento por defecto, "Automático".

En el modo "Automático" existirán dos posibilidades que lo fuercen a cambiar al modo "Manual" o al "Nocturno". Al primero de ellos la condición será que el usuario presione el pulsador correspondiente a encender las luminarias. Por otra parte, el modo "Nocturno", se activará a una cierta hora nocturna establecida por el usuario, figura 34.

```
300  void Habitacion::ModoF(){
301
302    double TiempoTemp = 180; // Indica el tiempo de Temporizador
303    bool Tiempo = 0;
304
305    if ( _Modo == "Automatico" ) {
306
307      for (int i = 0; i < _AddressModPulsador.size(); i++) {
308
309        if ( ( ioSample.getRemoteAddress64().getMsb() ==
310            _AddressModPulsador[i].getMsb() ) &&
311            ( ioSample.getRemoteAddress64().getLsb() ==
312            _AddressModPulsador[i].getLsb() )
313            ) {
314
315              if ( ioSample.isDigitalOn(0) == 1 ) {
316
317                if ( _EstadoAnteriorLuz == 0 ) {
318
319                  _Modo = "Manual";
320
321                }
322
323              }
324
325              _EstadoAnteriorLuz = ioSample.isDigitalOn(0);
326
327              }
328
329      }
330
331      if ( _RModoF.Horario( HoraN, MinutoN ) == 1 )  {
332
333          _Modo = "Nocturno";
334
335      }
336
337    }
```

Figura 34.

Posteriormente, en el modo “Manual” se modificará al modo “Automático” cuando la variable “_Movimiento” se encuentre a 0 durante un tiempo del orden de minutos, en este caso han sido 2 minutos utilizando para contabilizar este tiempo el reloj “_RModoF”. Por otro lado, también se podrá modificar al modo “Nocturno” de forma similar que sucedía cuando esté activo el modo “Automático”.

Luego, en el modo “Nocturno” sólo se podrá modificar esta variable al modo “Automático” cuando se supere una hora por la mañana siendo la hora cuando se despierta el usuario. Este procedimiento como los anteriores para ser modificados a este modo se realizará con la función “Horario” de la clase “Reloj” devolviendo un 1 cuando la hora actual es superior a la hora que se le introduzca a esta función.

void OnOFFLuz (): en esta función se decidirá si se activan o se desconectan las luminarias. Dependiendo de las variables, "_Movimiento", "_Oscuridad" y "_Modo" siendo modificadas en los métodos anteriores.

En primer lugar, se decidirá el estado de las luminarias. Posteriormente, habrá un proceso de enviar una trama API "Remote AT Request" a los módulos remotos que controlan las luminarias cuando la variable "Enviar" de los actuadores esté activada.

Entonces en este primer proceso se activará la variable "Enviar" de los actuadores que controlan las luminarias indicando que estos actuadores deben enviar una trama API para modificar su estado, teniendo en cuenta en qué modo se encuentra el sistema, "Automático", "Manual" o "Nocturno". En el primer caso, se activará "Enviar" para accionar las luminarias cuando se detecte movimiento y además el nivel de iluminación esté por debajo del umbral de luz establecido. Después, el proceso de desconexión surgirá cuando no se detecte movimiento durante un cierto tiempo para este caso se ha puesto un tiempo de 3 minutos, siendo esta otra condición para activar "Enviar".

Por otra parte, tanto en el modo "Manual" como el "Nocturno" el estado de las luminarias dependerá del pulsador correspondiente a modificar esta magnitud instalado en el módulo Pulsador. En el momento que se presione este sensor modificará las luminarias al estado contrario a las que se encuentra, figura 35.

```
440    if ( _Modo == "Manual"  || _Modo == "Nocturno") {
441
442        for (int i = 0; i < _AddressModPulsador.size(); i++) {
443
444          if ( ioSample.getRemoteAddress64().getMsb() ==
445              _AddressModPulsador[i].getMsb()
446              && ioSample.getRemoteAddress64().getLsb() ==
447              _AddressModPulsador[i].getLsb() ) {
448
449
450              if ( ioSample.isDigitalOn(0) == 1 ) {
451
452                if (_EstadoAnteriorLuz == 0) {
453
454                //_ALuz.Enviar();
455                  for (size_t i = 0; i < _ALuz.size(); i++) {
456
457                    if ( _ALuz[i].GetEstado() == 0 ) {
458
459                      LuzValue[0] = 0x5;
460
461                    }  else {
462
463                      LuzValue[0] = 0x4;
464
465                    }
466                    _ALuz[i].Enviar();
467
468                  }
469
470                }
471
472              }
473
474            _EstadoAnteriorLuz = ioSample.isDigitalOn(0);
475
476          }
477
478        }
479
480    }
```

Figura 35.

El siguiente proceso de esta función será enviar la trama API durante la variable “Enviar” este activada. La trama API enviada será del tipo “Remote AT Command” necesitando esta cadena de bytes la dirección de destino, el comando que se pretende modificar y el byte que indica el valor de este comando. La dirección de destino se decidirá mediante el actuador debido a que este objeto almacena su dirección de 64 bits. Después, el comando estará predeterminado por el actuador al que se le envíe esta trama teniendo este objeto su propio comando. Luego, el valor del comando estará definido en el proceso anterior siendo este un byte 0x4 (Salida digital a nivel bajo) o 0x5 (Salida digital nivel alto).

El envío de la trama se realizará con la función que se presentó al principio de este capítulo “EnvioTramaAT”. Entonces, mientras la función devuelva un 0, se enviará la trama hasta que se transmita correctamente. En el caso, de resultar con éxito esta transferencia de datos, se asignará al actuador al que se le está enviando esta trama que no es necesario que transmita más tramas API (“Enviar” = 0).

```
482   for (size_t i = 0; i < _ALuz.size(); i++) {
483
484     if ( _ALuz[i].GetEnviar() == 1 ) {
485
486       Sample = 0;
487
488       LuzCmd[0] = _ALuz[i].GetCmd( 0 );
489       LuzCmd[1] = _ALuz[i].GetCmd( 1 );
490
491       TramaEnviada = EnviarTramaAT( xbee, _ALuz[i].getAddress64(), LuzCmd,
492                                     LuzValue );
493
494       if ( TramaEnviada == 1 ) {
495
496          Sample = 1;
497          TramaEnviada = 0;
498          //No enviamos mas tramas
499          _ALuz[i].NoEnviar();
500
501         //Resetear variables para el modo Automatico
502         _ROnOff.TempReset();
503
504         //Modificacamos la variable que indica si esta encendida o no
505         //la Habitacion
506         if ( _Encender == 0 ) {
507
508           _Encender = 1;
509           _ALuz[i].Act();
510
511         } else {
512
513           _Encender = 0;
514           _ALuz[i].DesAct();
515
516         }
517
518       }
519
520     }
521
522   }
```

Figura 36

- Sistema de climatización

void SisTemperatura (): el siguiente método se corresponderá con el sistema de climatización. Este mecanismo se encargará de regular la temperatura de la zona. Por lo tanto, será necesario el sensor de temperatura para conocer la temperatura ambiente y una consigna para regular esta magnitud a un valor determinado, aplicándole una tolerancia de ± 1 grado. Entonces, cuando se detecte que la temperatura está por encima o por debajo de la consigna, se indicará que esta fuera de este rango y cuando se advierta que está en la zona de confort es que esta magnitud está entre estos valores (consigna ± 1).

El primer procedimiento que se debe realizar será la obtención de la consigna ("_Consigna"). La variable mencionada se modificará enlazando un "Módulo Pulsador" al sistema teniendo en él un potenciómetro cuyo valor modificará la consigna. El valor de este componente se obtendrá mediante la función "getAnalog (uint8_t pin)". Este valor oscila en un rango de 0 a 1024 debido a que el convertidor analógico - digital del XBee es de 10 bits y se pretende escalar entre 0 y 30 siendo los valores extremos que se pueden asignar a la consigna. La conversión de estos datos se realizará con la función "map" permitiendo escalar una determinada variable al rango que se desee, figura 37.

```
530  void Habitacion::SisTemperatura(){
531
532    uint8_t TemperaturaCmd[2];
533    uint8_t TemperaturaValue[1];
534
535    bool TramaEnviada;
536
537    uint8_t ToleranciaTemperatura = 1;
538
539      //En el siguiente for se obtendrá la consigna del modulo pulsador
540      //leida por la entrada 3 del xbee ( potenciometro )
541      for (int i = 0; i < _AddressModPulsador.size(); i++) {
542
543        if ( ( ioSample.getRemoteAddress64().getMsb() ==
544            _AddressModPulsador[i].getMsb() ) &&
545             ( ioSample.getRemoteAddress64().getLsb() ==
546            _AddressModPulsador[i].getLsb() ) ) {
547
548            //uint16_t EntradaAnalogica = ioSample.getAnalog(3);
549            _Consigna = map( ioSample.getAnalog(3) , 0, 1023, 0, 30 );
550
551            if ( ioSample.getAnalog(3) == 0 ) {
552
553              _Consigna = 25;
554
555            }
556
557        }
558
559      }
560
561    }
```

Figura 37.

El siguiente apartado de este sistema es decidir cuando los instrumentos que componen el sistema de climatización deben conectarse o desactivarse dependiendo a una serie de condiciones. En primer lugar, el aire acondicionado se deberá encender en el momento que el valor del sensor de temperatura esté por encima de la consigna indicando el valor del actuador (0x5) y activar la variable “Enviar” correspondiente a este dispositivo. Por el lado contrario, se deberá accionar el calefactor cuando el sensor advierta que la temperatura está por debajo de la consigna realizando las mismas acciones mencionadas para el caso anterior. Después, la desconexión de estos dispositivos se producirá cuando estén activados y el valor de la temperatura se introduzca en la zona de confort, figura 38.

```
566   //En el momento que la temepratura este por encima de  _Consigna +
567   //ToleranciaTemperatura se activa el sistema de Climatizacion
568   if ( _T.Valor() > ( _Consigna + ToleranciaTemperatura ) ) {
569
570     for (size_t i = 0; i < _Climatizacion.size(); i++) {
571
572       if ( _Climatizacion[i].getAFrio().GetEstado() == 0 ) {
573
574         TemperaturaValue[0] = 0x5;
575         _Climatizacion[i].getAFrio().Enviar();
576
577       }
578
579     }
580
581   }
582
583   //Que la temperatura esta por debajo de la consigna + la Tolerancia, activa
584   //todos los sisetmas de climatizacón
585   if ( _T.Valor() < ( _Consigna - ToleranciaTemperatura ) ) {
586
587     for (size_t i = 0; i < _Climatizacion.size(); i++) {
588
589       if ( _Climatizacion[i].getACalor().GetEstado() == 0 ) {
590
591         TemperaturaValue[0] = 0x5;
592         _Climatizacion[i].getACalor().Enviar();
593
594       }
595
596     }
597
598   }

600   //Que la temepratura se encuente entre los valores +- de la consigna
601   //comprobando que esta activado el actuador concreto se enviara una trama para
602   //apagarlo
603   if ( ( _Consigna + ToleranciaTemperatura ) > _T.Valor() >
604       ( _Consigna - ToleranciaTemperatura ) ) {
605
606     for (size_t i = 0; i < _Climatizacion.size(); i++) {
607
608       if ( _Climatizacion[i].getAFrio().GetEstado() == 1 ) {
609
610         TemperaturaValue[0] = 0x4;
611         _Climatizacion[i].getAFrio().Enviar();
612
613       }
614
615       if ( _Climatizacion[i].getACalor().GetEstado() == 1 ) {
616
617         TemperaturaValue[0] = 0x4;
618         _Climatizacion[i].getACalor().Enviar();
619
620       }
621
622     }
623
624   }
```

Figura 38.

La transmisión de la trama se realizará en el momento de que la variable “Enviar” del actuador correspondiente esté activa. Nuevamente, se utilizará la función

"EnvioTramaAT" especificando tanto la dirección como el comando por medio del objeto "Climatización" y el valor será determinado por los procesos anteriores, figura 39.

```
628    for (size_t i = 0; i < _Climatizacion.size(); i++) {
629
630      if ( _Climatizacion[i].getAFrio().GetEnviar() == 1 ) {
631
632        Sample = 0;
633
634        Serial.println(" Enviando Temperatura..... ") ;
635
636        direccion = _Climatizacion[i].getAddress64();
637
638        TemperaturaCmd[0] = _Climatizacion[i].getAFrio().GetCmd( 0 );
639        TemperaturaCmd[1] = _Climatizacion[i].getAFrio().GetCmd( 1 );
640
641        TramaEnviada = EnviarTramaAT( xbee, direccion, TemperaturaCmd,
642                                      TemperaturaValue );
643
644        if ( TramaEnviada == 1 ) {
645
646          _Climatizacion[i].getAFrio().NoEnviar();
647
648          _Climatizacion[i].getAFrio().CambiarEstado();
649
650        }
651
652      }
653
```

Figura 39.

void MostrarConsigna (): la visualización de la consigna en los dos displays que tenga instalados el "Módulo Habitación" se realizará a través de este método.

En primer lugar, se instancian dos objetos "display" para conseguir mostrar un cierto número en este dispositivo. El primer display mostrará el primer número de la consigna y el segundo número se visualizará por medio del display restante.

La acción para visualizar el número en el display se realizará por medio de la función diseñada en este objeto "MostrarNum (uint8_t Num) activando las salidas del Arduino que se necesitan para enseñar el número correspondiente a la variable "Num".

```
704  void Habitacion::MostrarConsigna(){
705
706    String StrConsigna = String( _Consigna );
707
708    uint8_t Num1 = StrConsigna[0] - '0';
709    uint8_t Num2 = StrConsigna[1] - '0' ;
710
711
712    if ( Num1 != _Num1Anterior ) {
713
714      switch ( Num1 ) {
715
716        case 0:
717
718          D1.MostrarNum( 0 );
719
720        break;
721
722        case 1:
723
724          D1.MostrarNum( 1 );
725
726        break;
727
728        case 2:
729
730          D1.MostrarNum( 2 );
731
732        break;
733
734        case 3:
735
736          D1.MostrarNum( 3 );
737
738        break;
739
740      }
741
742    }
743
744    _Num1Anterior = Num1;
```

Figura 40.

- Sistema de persianas

void SisPersianas (): en el próximo sistema se analizarán dos procesos posibles. Por un lado, si se une un “Módulo Pulsador” a este sistema las persianas se podrán activar tanto de forma automática como manual. Por otro lado, si este sistema carece del tipo de módulo mencionado, simplemente se accionará automáticamente.

El modo automático activará las persianas a unas horas determinadas establecidas como “Horas de mañana” y “Horas Nocturnas” cuando se produzca un flanco de subida por medio del método “Horario” que ofrece el reloj instanciado en este sistema. Después, el modo manual se realizará a través de los pulsadores predeterminados para activar las persianas que se encuentren en este módulo remoto.

En el momento que se produzcan las acciones anteriores, se activará la variable “Enviar”, del actuador correspondiente, indicando que se deberá enviar una trama API para modificar el estado de este actuador.

En segundo lugar, se procederá al envío de las tramas API, existiendo una condición crítica en este sistema. Esta condición será que en ningún momento se activen los dos actuadores de una misma persiana al mismo tiempo con la intención de no forzar el motor. La condición se realizará cuando se quiera activar un actuador en concreto, comprobando que el actuador contrario este desactivado por medio de la variable “_Estado” que almacena esta clase. Entonces, si se pretende activar un actuador y el opuesto está funcionando se enviará una trama API con el valor 0x4, apagando este actuador y cuando se reciba la trama indicando que está desactivado se transmitirá otra trama para activar el actuador que se desea, figura 41.

```
952   for (size_t i = 0; i < _Persianas.size(); i++) {
953
954     if ( _Persianas[i].getASubir().GetEnviar() == 1 ) {
955
956       Sample = 0;
957
958       //Si el Actuador de bajar esta activado o desactivado
959       if ( _Persianas[i].getABajar().GetEstado() == 0 ) {
960
961         if ( _Persianas[i].getASubir().GetEstado() == 0 ) {
962
963           PerValue[0] = 0x5;
964
965         } else {
966
967           PerValue[0] = 0x4;
968
969         }
970
971         Serial.println(" Enviando.......... ") ;
972
973         PerCmd[0] = _Persianas[i].getASubir().GetCmd( 0 );
974         PerCmd[1] = _Persianas[i].getASubir().GetCmd( 1 );
975
976         direccion = _Persianas[i].getAddress64();
977
978         TramaEnviada = EnviarTramaAT( xbee, direccion, PerCmd, PerValue );
979
980         if ( TramaEnviada == 1 ) {
981
982           Sample = 1;
983
984           TramaEnviada = 0;
985
986           _RPer.TempReset();
987
988           _Persianas[i].getASubir().NoEnviar();
989
990           //Modifica el estado que tiene al contrario
991           _Persianas[i].getASubir().CambiarEstado();
992
993         }
994
995       } else {
996
997         PerValue[0] = 0x4;
998
999         Serial.println(" Enviando.......... ") ;
1000
1001        direccion = _Persianas[i].getAddress64();
1002        PerCmd[0] = _Persianas[i].getABajar().GetCmd( 0 );
1003        PerCmd[1] = _Persianas[i].getABajar().GetCmd( 1 );
1004
1005        TramaEnviada = EnviarTramaAT( xbee, direccion, PerCmd, PerValue );
1006
1007        if ( TramaEnviada == 1 ) {
1008
1009          TramaEnviada = 0;
1010
1011          _Persianas[i].getABajar().DesAct();
1012
1013        }
1014
1015      }
1016
1017    }
```

Figura 41.

- Sistema de ahorro de energía

void DetecPersonas (): en este método se decidirá si la habitación está habitada activando el atributo "_Personas". Los sensores de movimiento junto a una instancia del objeto "Reloj", serán los encargados de modificar este atributo.

La zona estará habitada en el momento que el sensor de movimiento detecte presencia en una determinada zona. Posteriormente, para desactivar esta variable el sensor no debe detectar movimiento y el temporizador del reloj mencionado se debe activar.

void ModoSleep (): el modo sleep de los módulos que se encuentren dentro de este sistema, se realizará en este método cuando se produzca dos acciones determinadas.

La primera, será cuando exista un flanco de bajada en la variable "_Personas". Esta acción ejecutará el modo sleep de todos los dispositivos que estén interaccionando con este módulo Habitación. Este proceso se realizará mediante el envío de una trama API "Remote AT command" modificando el parámetro "SM" de los XBee a todos los vectores que almacenan las direcciones de los módulos remotos. El valor para activar este modo será el byte 0x4 (Cyclic sleep).

Posteriormente, el proceso para desactivar este modo será cuando en la variable "_Personas", se produzca un flanco de subida enviando la misma trama anterior, pero con el valor 0x0 desconectando este modo.

Finalmente, este módulo enviará una trama API "Tx Status" para advertir al módulo Casa que este dispositivo está preparado para activar el modo sleep. En la trama transmitida se enviarán una serie de bytes específicos para realizar esta acción. El mensaje se codifica enviando "011" indicando que se puede activar el modo sleep y en caso contrario, el mensaje transmitirá "012" teniendo que estar desactivado el modo sleep de este módulo.

```
1122  void Habitacion::ModoSleep(){
1123
1124    //se debe indicar la direccion del modulo casa
1125
1126    bool TramaEnviada;
1127
1128    bool Enviado = 0;
1129
1130    uint8_t payload[3];
1131
1132    uint8_t SleepCmd[] = "SM";
1133    uint8_t SleepValue[1];
1134
1135    XBeeAddress64 direccion;
1136
1137    if ( _Personas == 0 && _EstadoAnteriorPersonas == 1 || _Personas == 1
1138        && _EstadoAnteriorPersonas == 0 ) {
1139
1140      Sample = 0;
1141
1142      if ( _Personas == 0 ) {
1143
1144        SleepValue[0] = 0x4;
1145        payload[0] = 0x0;
1146        payload[1] = 0x1;
1147        payload[2] = 0x1;
1148
1149      }  else {
1150
1151        SleepValue[0] = 0x0;
1152        payload[0] = 0x0;
1153        payload[1] = 0x1;
1154        payload[2] = 0x2;
1155
1156      }
1158      //Modulos pulsadores
1159      for (size_t i = 0; i < _AddressModPulsador.size(); i++) {
1160
1161        Enviado = 0;
1162
1163        while ( Enviado == 0 ) {
1164
1165          Serial.println(" Enviando SLEEP.......... ") ;
1166
1167          TramaEnviada = EnviarTramaAT( xbee, _AddressModPulsador[i],
1168                                        SleepCmd, SleepValue );
1169
1170          if ( TramaEnviada == 1 ) {
1171
1172              Enviado = 1;
1173              TramaEnviada = 0;
1174        }
1175
1176      }
```

Figura 42.

8.8.12 Módulo Remoto

La siguiente clase ha sido diseñada con el objetivo de guardar dos tipos de variables, la dirección del dispositivo remoto y el valor de una variable booleana que señale si este módulo está preparado para dormir. Al necesitar una dirección, esta clase heredará de la clase "Dirección" obteniendo los atributos y métodos necesarios para modificar esta propiedad.

- Atributos

bool _Sleep: señala si el módulo puede activar su modo sleep. El valor 0 muestra que no se puede accionar mientras que el valor contrario señalará que si es posible activar este modo.

- Métodos

void setSleep (bool Sleep): asigna al atributo "_Sleep" el valor de la variable "Sleep".

bool getSleep (): devuelve el estado del atributo de esta clase.

8.8.13 Casa

El objeto "Casa" ha sido desarrollado para instalar en el módulo Casa. En esta clase se implementará la lógica para decidir si todos los dispositivos de la casa deben "dormir" o en cambio "despertar".

Los módulos Habitación enviarán una trama Tx con un mensaje concreto para señalar que no detecta presencia en su zona y que está preparado para dormir. Las dos condiciones para activar este módulo en toda la casa será que todos los módulos Habitación envían la trama correspondiente y que el sensor de movimiento que se encuentra en este dispositivo no detecta presencia.

- Atributo:

XBee xbee: la clase XBee será necesario para el enlace del sistema de comunicación y el Arduino.

ZBRxResponse RxData: objeto donde se almacenará la información cuando se reciba una trama API Rx.

ArduVector<ModuloRemoto> _MRemoto: vector que contiene instancia de "MóduloRemoto".

SenMovi _M: el sensor de movimiento estará reflejado en este atributo.

bool _CasaVacia: variable que muestra si la casa está vacía. El valor 0 no estará vacía y en el caso contrario si estará vacía.

bool _EstadoCasaVacia: el siguiente atributo guardará el último estado de la variable "_CasaVacia" para detectar los flancos tanto de subida como de bajada.

- Métodos:

Constructor: en el constructor se inicializan los atributos, las variables booleanas a 0, se añadirá el pin correspondiente al sensor de movimiento y se creará la respues Rx mediante el constructor de la clase "ZBRxResponse".

void Inicializar (XBee Xbee): en este método se copiará el objeto XBee pasado a la función, con las características necesarias para su correcto funcionamiento.

void AddRemoto (ModuloRemoto remoto): en el interior de esta función, se añadirá un nuevo módulo remoto por medio de la función "push_back" de la clase ArduVector.

void RecibirDatos (): el objetivo de este método es recibir las tramas API Rx entrantes al XBee. En la recepción de tramas se utilizará la función "readPacket" de la librería "XBee".

void DetecPersonas (): en la siguiente función se detectará si en el hogar permanecen personas en su interior o por lo contrario la casa está vacía. Este método utilizará el atributo "RxData" obteniendo de él las propiedades de las tramas API Rx que se reciben.

Después, dependiendo del mensaje de la trama recibida se realizarán ciertas acciones. En el caso de almacenar en el mensaje los bytes "011" se indicará que el módulo no tiene la posibilidad de dormir. Por el contrario, si se recibe una trama transmitiendo los bytes "012", la variable "_Sleep" de los módulos remotos se activa. La variable "_Personas" se encontrará a nivel alto cuando todos los módulos Habitaciones transmitan la trama "012" y el sensor de movimiento no esté activo.

Posteriormente, a partir de estas dos condiciones, si en el atributo "_Personas", se produce un flanco de subida se enviará una trama a cada módulo para activar su modo sleep. Por otra parte, si se detecta por medio del sensor de movimiento que existe presencia se desactiva este modo enviando una trama API con el valor 0x0 haciendo referencia al comando "SM".

```
93  void Casa::DetecPersonas(){
94
95    uint8_t NumHabSleep = 0; //Contador que cuenta las habitaciones que estan
96                             //preparadas para dormir
97
98    //En el siguiente for se realizaran dos tareas
99    //La primera, detectar si ha entrada una trama Rx de los modulo habitacion
100   //diciendo que la habitacion se puede dormir
101   //codigo 011 esta preparadas para dormir
102   //codigo 012 no esta preparada para dormir
103
104
105   //En segundo lugar se contara si todos los modulos habiatcion de la casa
106   //estan preparados para dormir, por lo tanto, se colocaran todos los módulos
107   //habiatcion en modo sleep
108
109   for (size_t i = 0; i < _MHabitacion.size(); i++) {
110
111     if ( RxData.getRemoteAddress64().getMsb() ==
112         _MHabitacion[i].getAddress64().getMsb() &&
113         RxData.getRemoteAddress64().getLsb() ==
114         _MHabitacion[i].getAddress64().getLsb() ) {
115
116       if ( RxData.getData(0) == 0 && RxData.getData(1) == 1 &&
117            RxData.getData(2) == 1 ) {
118
119         _MHabitacion[i].setSleep( true );
120
121       }
122
123       if ( RxData.getData(0) == 0 && RxData.getData(1) == 1 &&
124            RxData.getData(2) == 2 ) {
125
126         _MHabitacion[i].setSleep( false );
127
128       }
129
130     }
132     if ( _MHabitacion[i].getSleep() == true ) {
133
134       NumHabSleep++;
135
136     }
137
138   }
139
140   //Si el contador NumHabSleep es igual al número de habitaciones la casa estará
141   //preparada para dormir
142   if ( NumHabSleep != _MHabitacion.size() || _M.Salida() == 1) {
143
144     _CasaVacia = 0;
145
146   } else {
147
148     _CasaVacia = 1;
149
150   }
151
152
153 }
```

Figura 43.

void ModoSleep (): el método posterior elabora la lógica de envió de tramas API en unos determinados instantes. La función depende del atributo "_Personas" cuando se produce un flanco de subida o de baja en esta variable. En este instante, se procederá a transmitir una trama "Remote AT Command" con la intención de modificar el comando AT "SM", figura 44.

```
155  void Casa::ModoSleep(){
156
157    bool Enviado = 0;
158
159    bool TramaEnviada = 0;
160
161    uint8_t SleepCmd[] = "SM";
162    uint8_t SleepValue[1];
163
164
165    if ( _CasaVacia == 1 && _EstadoCasaVacia == 0 || _CasaVacia == 0
166        && _EstadoCasaVacia == 1 ) {
167
168      if ( _CasaVacia == 0 ) {
169
170        SleepValue[0] = 0x0;
171
172      } else {
173
174        SleepValue[0] = 0x4;
175
176      }
177
178      //Modulos habitaciones
179      for (size_t i = 0; i < _MHabitacion.size(); i++) {
180
181        Enviado = 0;
182
183        while ( Enviado == 0 ) {
184
185          Serial.println(" Enviando SLEEP..... ") ;
186
187          XBeeAddress64 direccion = _MHabitacion[i].getAddress64();
188
189          TramaEnviada = EnviarTrama( xbee, direccion, SleepCmd, SleepValue );
```

```
190
191           if ( TramaEnviada == 1 ) {
192
193             Enviado = 1;
194             TramaEnviada = 0;
195
196           }
197
198         }
199
200       }
201
202     }
203
204     _EstadoCasaVacia = _CasaVacia;
205
206   }
```

Figura 44.

8.8.14 Programa Arduino

Los módulos Habitación de cada sector necesitarán almacenar un programa en la memoria de su microcontrolador para ejecutar la lógica necesaria para cada sistema que requiera el módulo en concreto. En el anexo 13 se expondrán los programas diseñados a partir de las librerías creadas anteriormente.

Memoria justificativa

Índice

Índice de figuras

1. Rectificador de precisión

En los siguientes apartados, se describen los procesos de diseño para el desarrollo del rectificador de precisión de onda completa, figura 1. En primer lugar, se describen los efectos no ideales de los amplificadores operacionales, que pueda afectar al diseño. Posteriormente, el funcionamiento del circuito y el filtrado de la señal de salida.

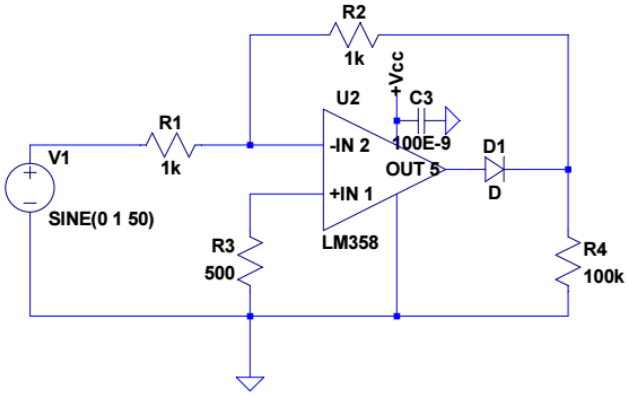


Figura 1.

1.1 Amplificador operacional real

Los amplificadores operacionales ideales tendrán una serie de características que no afectará a su salida. En cambio, cuando se trabaja con amplificadores operacionales reales, tienen una serie de características que pueden afectar a la salida del sistema. En el presente circuito, se presentarán principalmente los errores de la componente de continua de los amplificadores operacionales no ideales. Estos errores serán las corrientes de bias y la tensión de offset.

Las corrientes de bias serán las corrientes necesarias que se debe introducir en las entradas del amplificador operacional para polarizar correctamente los transistores que lo forman en su interior. Estas corrientes pueden producir modificaciones en la salida, por lo tanto, se deben anular. El rectificador de precisión elegido se encontrará en una configuración inversora en uno de los semiciclos de la señal de entrada. La anulación de estas corrientes se realizará conectando una resistencia entre la entrada no inversora del amplificador

operacional y tierra. Por lo tanto, la condición para eliminar las corrientes de bias, la resistencia mencionada deberá tener el siguiente valor.

$$R3 = \frac{R1 * R2}{R1 + R2}$$

Posteriormente, la tensión de offset será una tensión continua que presentan los amplificadores operacionales, en sus entradas. El orden de la tensión de offset es de mV, teniendo en cuenta que si se quiere amplificar una señal también se amplifica la señal de offset pudiendo ser muy elevado, modificando así la tensión de salida.

La tensión de offset en este circuito, $\boldsymbol{V^-}$, será amplificada por la relación de resistencias entre R2 y R1. Representada en la siguiente ecuación.

$$V_O = -\frac{R_2}{R_1} V_{IN} + \frac{R_2+R_1}{R_1} V^-$$

Las corrientes de bias y la tensión de offset están determinadas por el amplificador operacional, variando esta serie de parámetros en cada uno. En el circuito a diseñar, se ha realizado con un amplificador operacional LM358, teniendo como tensión de offset 3 mV, y como corriente de bias -20 nA.

En conclusión, los errores en continua que sufren los amplificadores operacionales, con una ganancia elevada, pueden producir cambios significativos en la entrada, debido a la amplificación de la tensión de offset. A su vez, las corrientes de bias si no son anuladas pueden disminuir la tensión de salida, si el amplificador operacional tiene unas corrientes de polarización lo suficientemente elevadas para que se reflejen en la salida. En este circuito, al no amplificar la señal, la tensión de offset no influye prácticamente en la tensión de salida. Por último, las corrientes de bias debido a la resistencia entre la entrada no inversora y tierra del diseño realizado, se anulan las corrientes mencionadas, por lo que no se reflejan en la salida del sistema.

1.2 Rendimiento del rectificador de precisión

En este apartado se presentan las ecuaciones para el cálculo del rendimiento de rectificador de precisión.

- Voltaje en continua: VDC

$$V_{DC} = \frac{1}{T} \int_0^T v_o(t)\, dt$$

- Voltaje eficaz: Vrms

$$V_{rms} = \sqrt{V_{rms} = \frac{1}{T} \int_0^T v_o^2\,(t)\, dt}$$

- Rendimiento: η

$$\eta = \left(\frac{V_{DC}}{V_{rms}}\right)^2$$

1.2.1 Rendimiento: Rectificador de onda completa

El rendimiento del rectificador de onda completa necesitará calcular primero la tensión de continua, VDC y posteriormente la tensión eficaz, Vrms.

Las tensiones a calcular dependen de la tensión de salida, la cual se modificará conforme a los semiciclo del periodo completo.

- Tensión de salida: $\boldsymbol{V_{IN} > 0}$

$$V_O = \frac{R_4}{R_1+R_2+R_4} * V_p * sen(wt)$$

- Tensión de salida: $\boldsymbol{V_{IN} < 0}$

$$V_O = -\frac{R_2}{R_1} * V_p * sen(wt) + \frac{R_2+R_1}{R_1} V^-$$

Por lo tanto, ambas tensiones deberán reflejar ambos semiciclos donde se mostrarán a continuación.

- Tensión continua

$$V_{DC} = \frac{1}{T} \int_0^T v_o(t)\, dt$$

$$V_{DC} = \frac{1}{T} \int_0^{T/2} \frac{R_4}{R_1+R_2+R_4} * V_p * sen(wt)\, dt + \frac{1}{T} \int_{T/2}^{T} -\frac{R_2}{R_1} * V_p * sen(wt) + \frac{R_2+R_1}{R_1} V^-\, dt$$

$$V_{DC} = \left(\frac{R_4}{R_1+R_2+R_4} + \frac{R_2}{R_1}\right) * \frac{V_p}{\pi}$$

- Voltaje eficaz: Vrms

$$V_{rms} = \sqrt{\frac{1}{T} \int_0^T v_o^2(t)\, dt}$$

$$V_{rms} = \sqrt{\frac{1}{T}\int_0^{T/2} \left(\frac{R_4}{R_1+R_2+R_4} * V_p * sen(wt)\right)^2 dt + -\frac{R_2}{R_1} * V_p * sen(wt)\, dt}$$

$$V_{rms} = \sqrt{\left(\left(\frac{R_4}{R_1+R_2+R_4}\right)^2 + \left(\frac{R_2}{R_1}\right)^2\right)} * \frac{V_p}{2}$$

- Rendimiento: η

$$\eta = \left(\frac{V_{DC}}{V_{rms}}\right)^2$$

$$\eta = \frac{\left(\frac{R_4}{R_1+R_2+R_4} + \frac{R_2}{R_1}\right)^2 * 4}{\left(\left(\frac{R_4}{R_1+R_2+R_4}\right)^2 + \left(\frac{R_2}{R_1}\right)^2\right) * \pi^4}$$

- $\boldsymbol{R_1 = 1K, R_2 = 1K, R_4 = 1K}$

$$\eta = 81,06\%$$

1.3 Rectificador de precisión: Un AO

El diseño del rectificador de precisión se realizará haciendo referencia del circuito mostrado en la figura 12. En primer lugar, se demuestra el funcionamiento del diodo y posteriormente el comportamiento de la tensión de salida respecto a la entrada.

1.3.1 Comportamiento del diodo

El diodo será el encargado de modificar el comportamiento del amplificador operacional. El análisis del funcionamiento del diodo se realizará a partir del comportamiento en tensión de los extremos del diodo, figura 24. Hay que tomar en cuenta, el comportamiento del amplificador operacional en este circuito equivalente funcionado como un comparador. El diodo se polariza directamente, si entre el ánodo y el cátodo hay una tensión positiva de aproximadamente 0,6 o 0,7 voltios, por lo contrario, el diodo se encontrará polarizado inversamente.

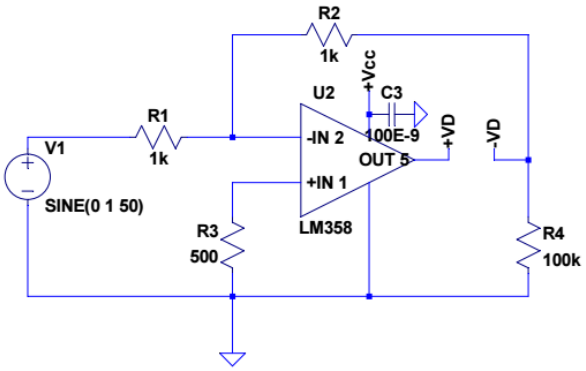


Figura 2.

En primer lugar, el ánodo del diodo estará conectado a la salida del amplificador operacional siendo la tensión en este punto la saturación positiva del amplificador operacional o cero voltios, debido a que su alimentación negativa está conectada a tierra. La tensión de salida del operacional dependerá de la diferencia de tensión que exista en las entradas de este dispositivo.

- Tensión de salida del amplificador operacional: $\boldsymbol{V_{AO}}$
- Ganancia del amplificador operacional, tiende a infinito: $\boldsymbol{A}$

$$V_{AO} = A\,(\,V^{+} - V^{-}\,)$$

- La tensión en la entrada no inversora: $\boldsymbol{V^{-}}$
- Tensión de entrada: $\boldsymbol{V_{IN}}$

$$V^- = \frac{R_2+R_4}{R_1+R_2+R_4} V_{IN}$$

Entonces, en el momento que la tensión de entrada se encuentra en el semiciclo positivo, la tensión de la entrada inversora es positiva y la tensión de salida del operacional se encuentra a nivel bajo. En el caso contrario, en el semiciclo negativo de la señal de entrada la tensión en la entrada inversora será negativa, cambiando así la tensión de salida del amplificador operacional de nivel bajo a nivel alto.

Por lo tanto, la tensión del ánodo del diodo (VD+), dependiendo de la entrada será:

- $\boldsymbol{V_{IN} > 0}$: $\boldsymbol{V_{D+} = 0\ V}$
- $\boldsymbol{V_{IN} < 0}$: $\boldsymbol{V_{D+} = V_{SAT+}}$

Posteriormente, se analiza el estado del cátodo coincidiendo con la tensión de salida.

- Tensión de salida: $\boldsymbol{V_o}$

$$V_O = \frac{R_4}{R_1+R_2+R_4} V_{IN}$$

La tensión de salida será positiva cuando la señal de entrada este en el semiciclo positivo y el caso opuesto cuando la señal de entrada se encuentre en el semiciclo negativo.

- $\boldsymbol{V_{IN} > 0}$: $\boldsymbol{V_o > 0\ V}$
- $\boldsymbol{V_{IN} < 0}$: $\boldsymbol{V_o < 0\ V}$

Finalmente, analizando el comportamiento de ambos extremos del diodo, donde su polarización dependerá de la tensión de entrada:

- Polarizado directamente: $\boldsymbol{V_{IN} < 0}$:

- Polarizado inversamente: $\boldsymbol{V_{IN} > 0}$:

En conclusión, el amplificador operacional se comportará como un inversor, en los semiciclos negativos y co mo un comparador en los semiciclos positivos.

1.3.1.1 Selección: Diodo

El presente apartado tendrá como objetivo, el proceso de diseño para la elección del diodo a instalar en el rectificador de precisión. La decisión dependerá de la cantidad de corriente máxima que circulará a través del diodo y las tensiones que deberán soportar ambos extremos de este elemento.

El diodo se conectará en la salida del amplificador operacional, aportando una corriente máxima de 20 mA. Por lo tanto, el diodo deberá soportar como mínimo una corriente de 20 mA. Posteriormente, la tensión máxima que soportará el diodo será cuando encontrará polarizado inversamente siendo dicha tensión de 3,5 voltios aproximadamente (Tensión de saturación). Entonces, la diferencia de tensión mínima que debe de soportar será de 3,5 voltios, cuando se encuentre polarizado inversamente.

A partir, de las condiciones de diseño descritas, el diodo escogido será el 1N4001, anexo 4.

1.4 Tensión de salida

A continuación, teniendo en cuenta el funcionamiento del diodo en el apartado anterior, se determinará el voltaje de salida del rectificador, dependiendo de la tensión de entrada. El amplificador se comportará como un comparador cuando la tensión de entrada se encuentre en el semiciclo positivo, por lo tanto, la tensión de salida será un divisor de tensión.

- Tensión de salida, $\boldsymbol{V_{IN} > 0}$:

$$V_O = \frac{R_4}{R_1+R_2+R_4} V_{IN}$$

En el semiciclo negativo el amplificador operacional se encuentra realimentado trabajando como un inversor.

- Tensión de salida, $\boldsymbol{V_{IN} < 0}$

$$V_O = -\frac{R_2}{R_1} V_{IN} + \frac{R_2+R_1}{R_1} V^-$$

El voltaje de salida se encontrará invertida respecto a la tensión de entrada, debido al primer miembro de la ecuación anterior. A su vez, la tensión de salida también dependerá de la tensión de offset del amplificador operacional siendo amplificada por la relación de resistencias que le acompaña.

Por otra parte, se decidirá el valor de las resistencias condicionado por el objetivo de que el circuito electrónico diseñado únicamente rectifique la señal y no la amplifique. En cada semiciclo se tendrá una relación de resistencias diferentes, pero en ambos casos deben de ser igual a la unidad.

A continuación, se muestra la relación de resistencias en ambos casos de la tensión de salida.

- Tensión de salida, $\boldsymbol{V_{IN} > 0}$

$$\frac{R_4}{R_1+R_2+R_4} = 1$$

- Tensión de salida, $\boldsymbol{V_{IN} < 0}$

$$\frac{R_2}{R_1} = 1$$

El diseño del rectificador tendrá que tener en cuenta estas dos condiciones. En el semiciclo negativo para no amplificar la tensión ambas resistencias deben de ser iguales,

estableciendo este valor en 1 KΩ. En cambio, en el caso de la tensión de entrada positiva, será necesario hacer la siguiente aproximación.

- $\boldsymbol{R_1 + R_2 << R_4}$

Por lo tanto, si se cumple esta aproximación el valor amplificado será semejante a la unidad. El valor de la resistencia R4 para cumplir la condición establecida, será de 100 KΩ.

- $\boldsymbol{\frac{R_4}{R_4} = 1}$

En el anexo 7 se observará la tensión de salida del rectificador diseñado con los valores de las resistencias asignados anteriormente. En conclusión, este dispositivo realizará correctamente la función de rectificador con un cierto error en los instantes que la señal de entrada se encuentra en los semiciclos positivo. En la figura 47 se observa el error que se produce en los ciclos positivos atenuando la señal. El error se producirá por la primera aproximación realiza para el cálculo del valor de las resistencias.

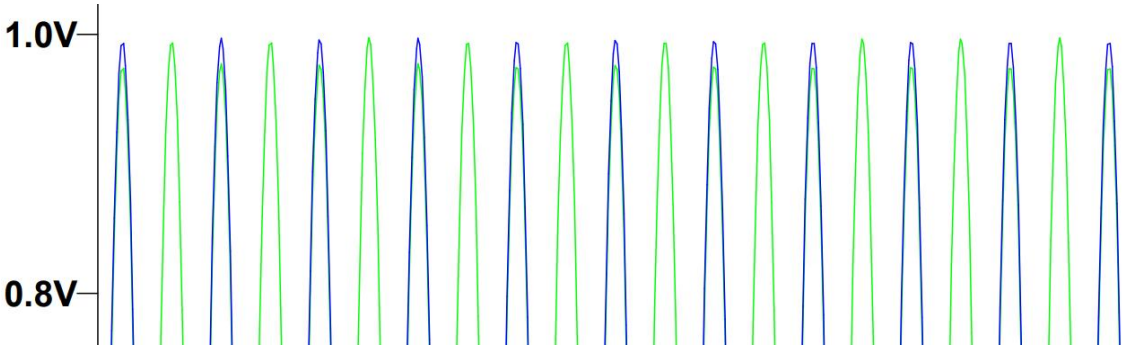


Figura 3.

2 Divisor de tensión

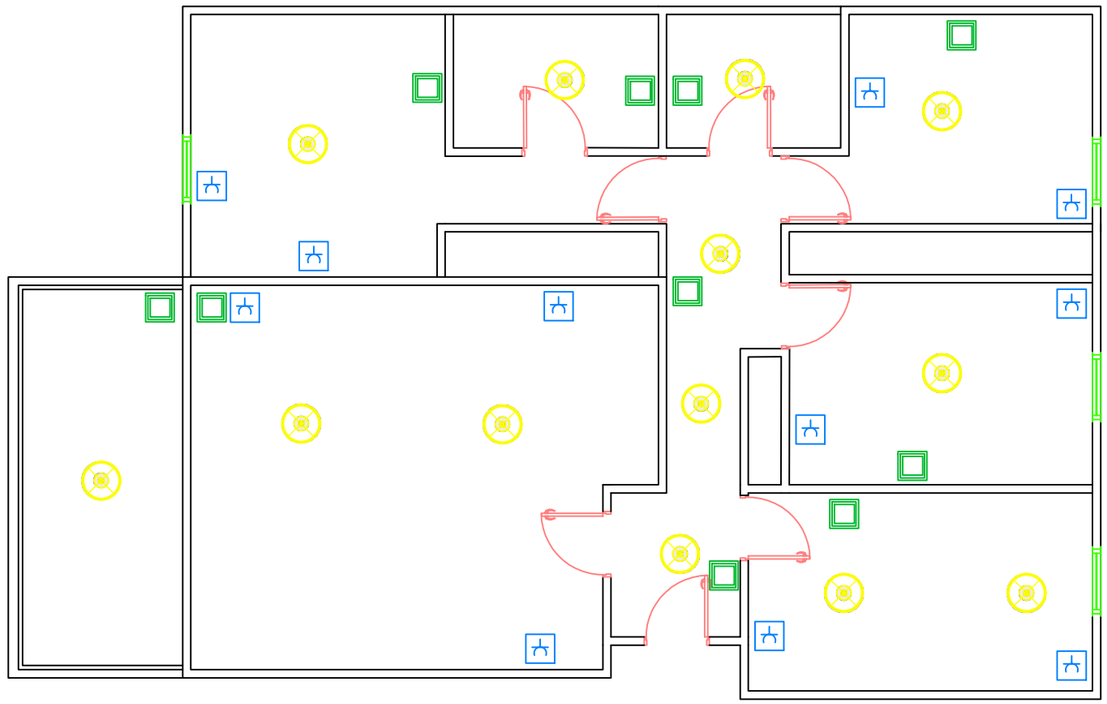
El divisor de tensión constará de dos resistencias conectadas en serie, alimentadas entre una tensión, Vcc, y tierra, figura 14. La resistencia R2, actuará como LDR, aumentado su valor, cuando se encuentre en un ambiente oscuro y cuando incide luz sobre la resistencia, su valor disminuirá. La tensión de salida corresponderá a la siguiente ecuación.

$$V_O = \frac{R_2}{R_1+R_2} V_{cc}$$

En conclusión, la tensión de salida aumentará a medida que la resistencia detecte menos luz en su entorno. Destacando, que la salida del divisor de tensión será la salida del sensor de temperatura que se conectará al microcontrolador, realizando la lectura del sensor de temperatura, y estableciendo a partir de esta lectura, un umbral de oscuridad para el control de las luminarias.

Planos

Índice



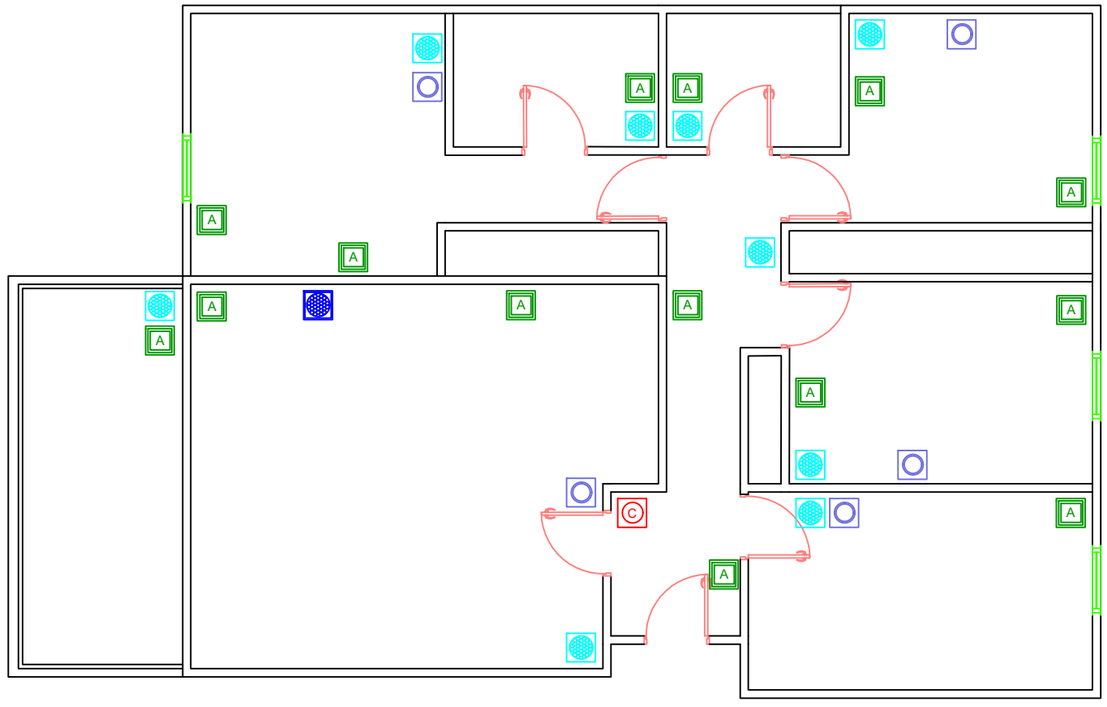
Actuador Luz

Fuerza

Luminaria

Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar

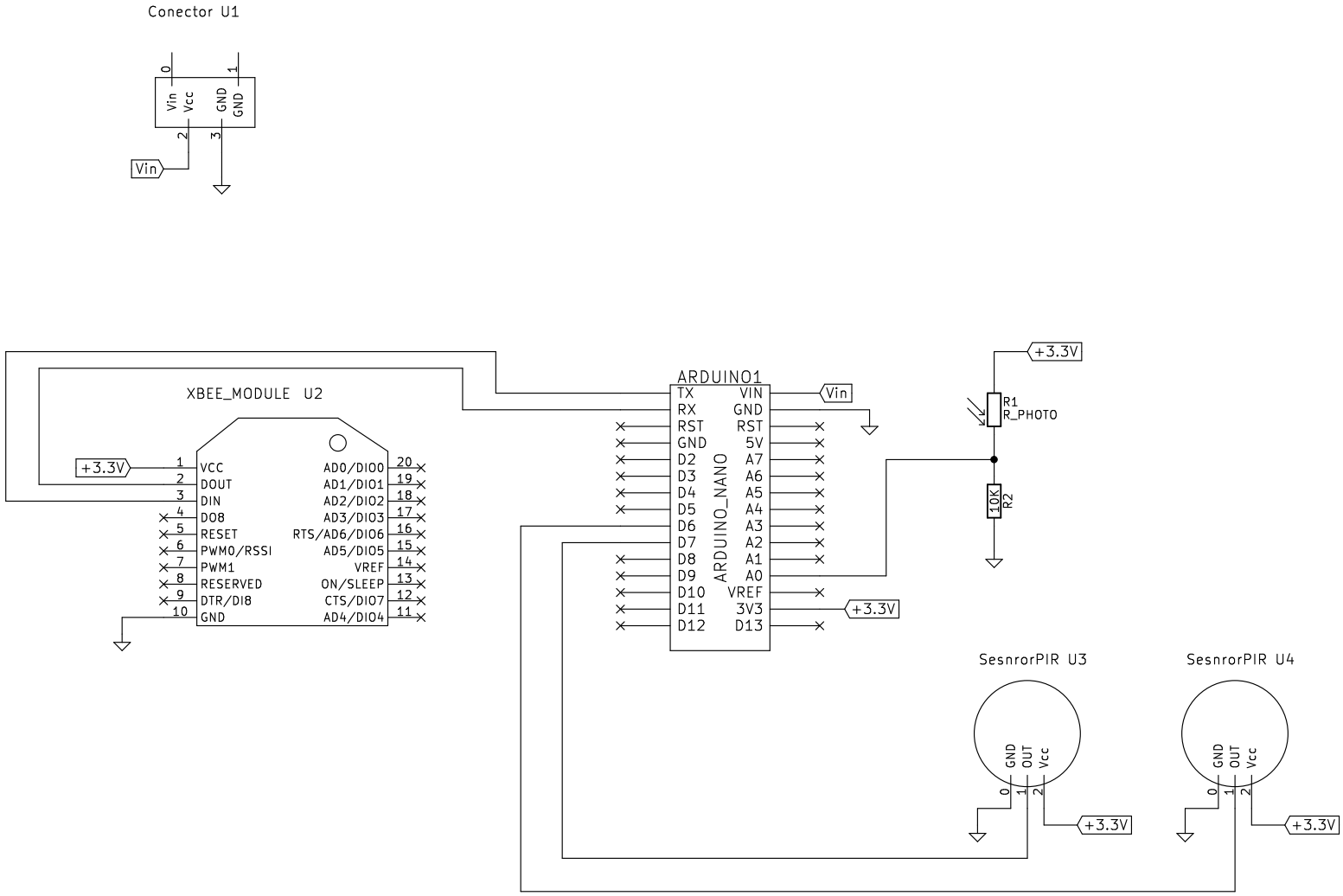
	Fecha	Autor		Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	26/06/2017	Julio Alejandro	ULL Universidad de La Laguna	Escuela Técnica Superior de Ingeniería Civil e Industrial Universidad de La Laguna
Comprobado	26/06/2017	Marichal Lorenzo		
Id. s. normas	UNE-EN-DIN			Nº plano:
Escala: 1:75	Electricidad			1



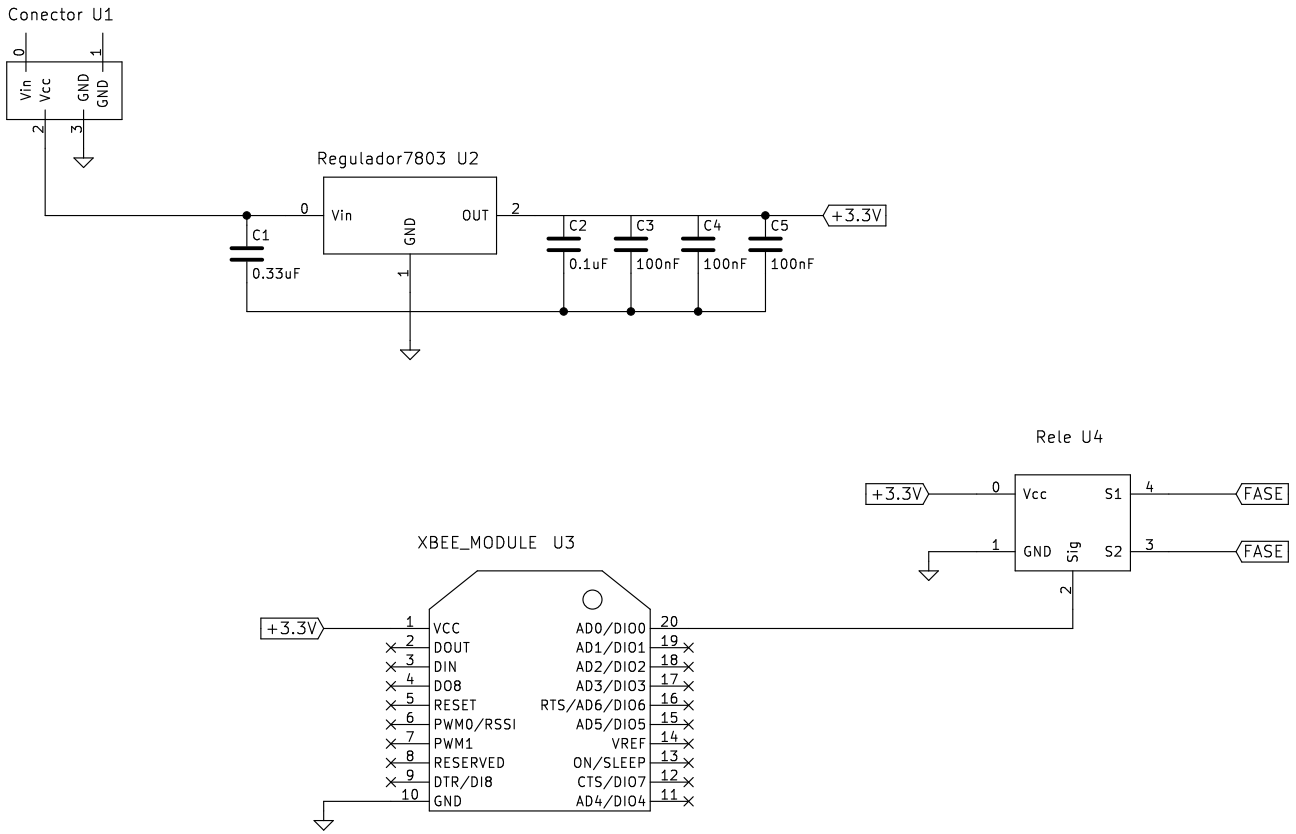
- Módulo Casa
- Módulo Habitación
- Módulo de movimiento
- Módulo Pulsador
- Módulo Actuador

Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar

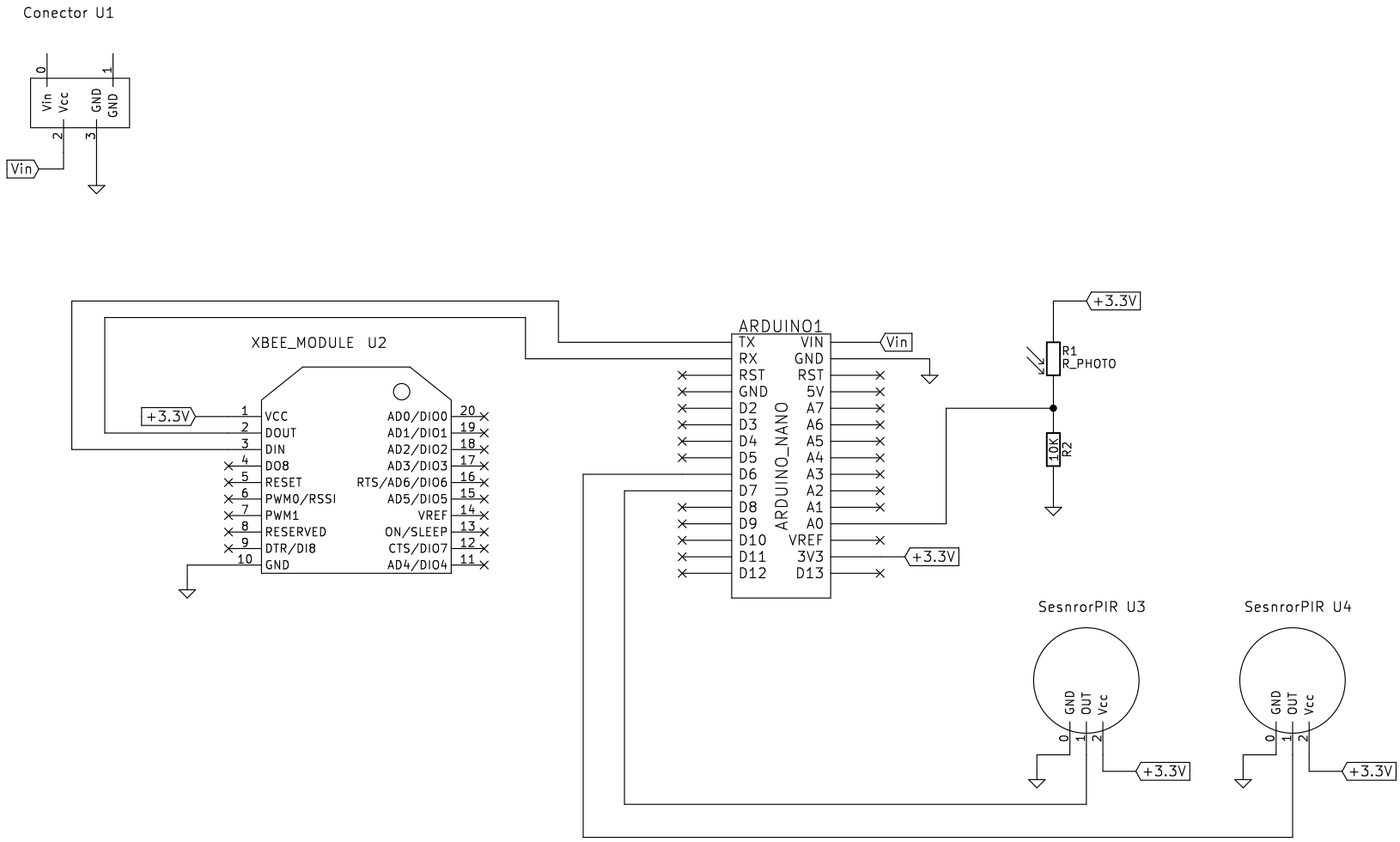
	Fecha	Autor		Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	26/06/2017	Julio Alejandro	ULL Universidad de La Laguna	Escuela Técnica Superior de Ingeniería Civil e Industrial Universidad de La Laguna
Comprobado	26/06/2017	Marichal Lorenzo		
Id. s. normas	UNE-EN-DIN			Nº plano:
Escala: 1:75	Módulos instalados			2



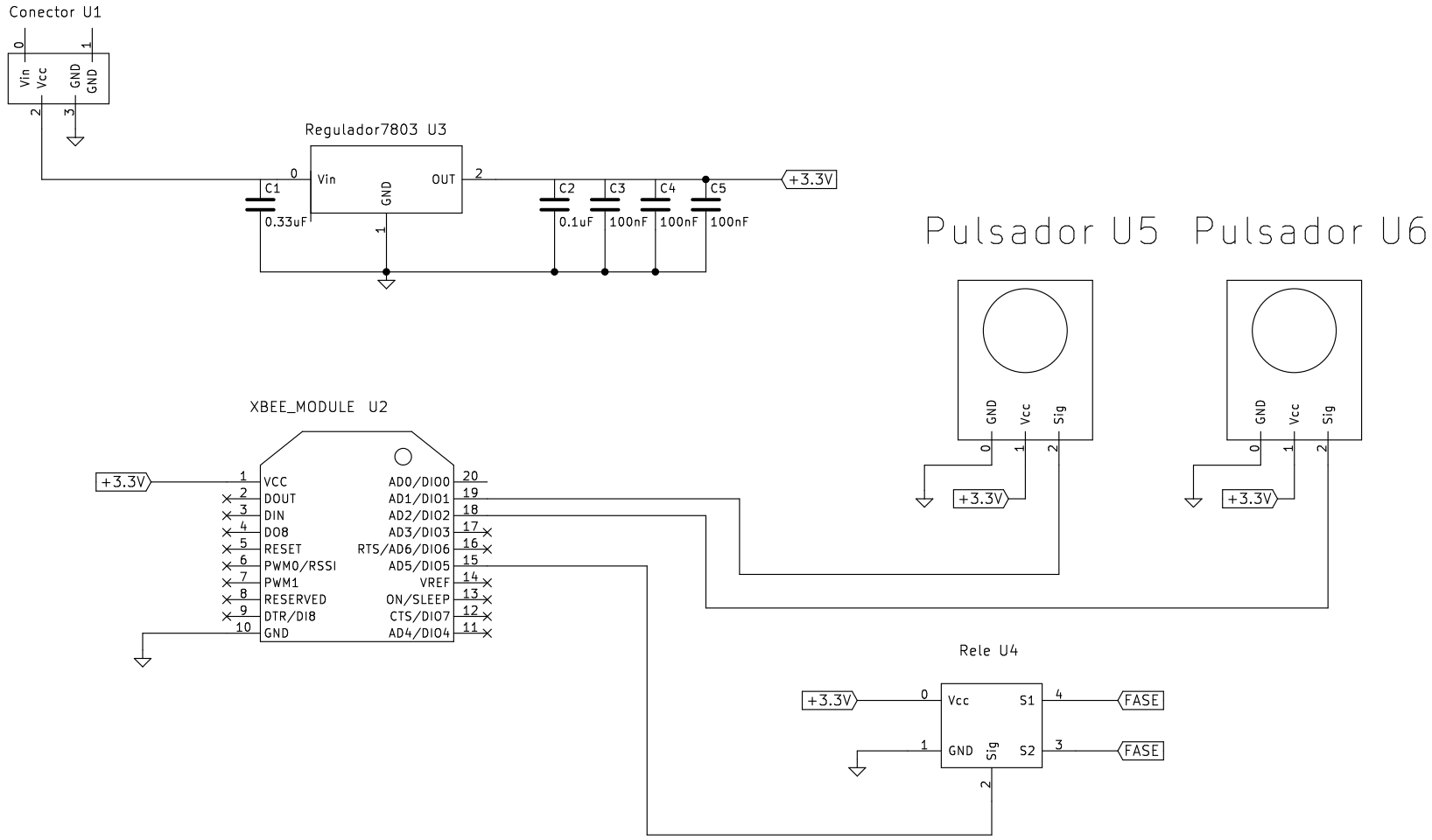
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Entrada – Módulo Casa			3



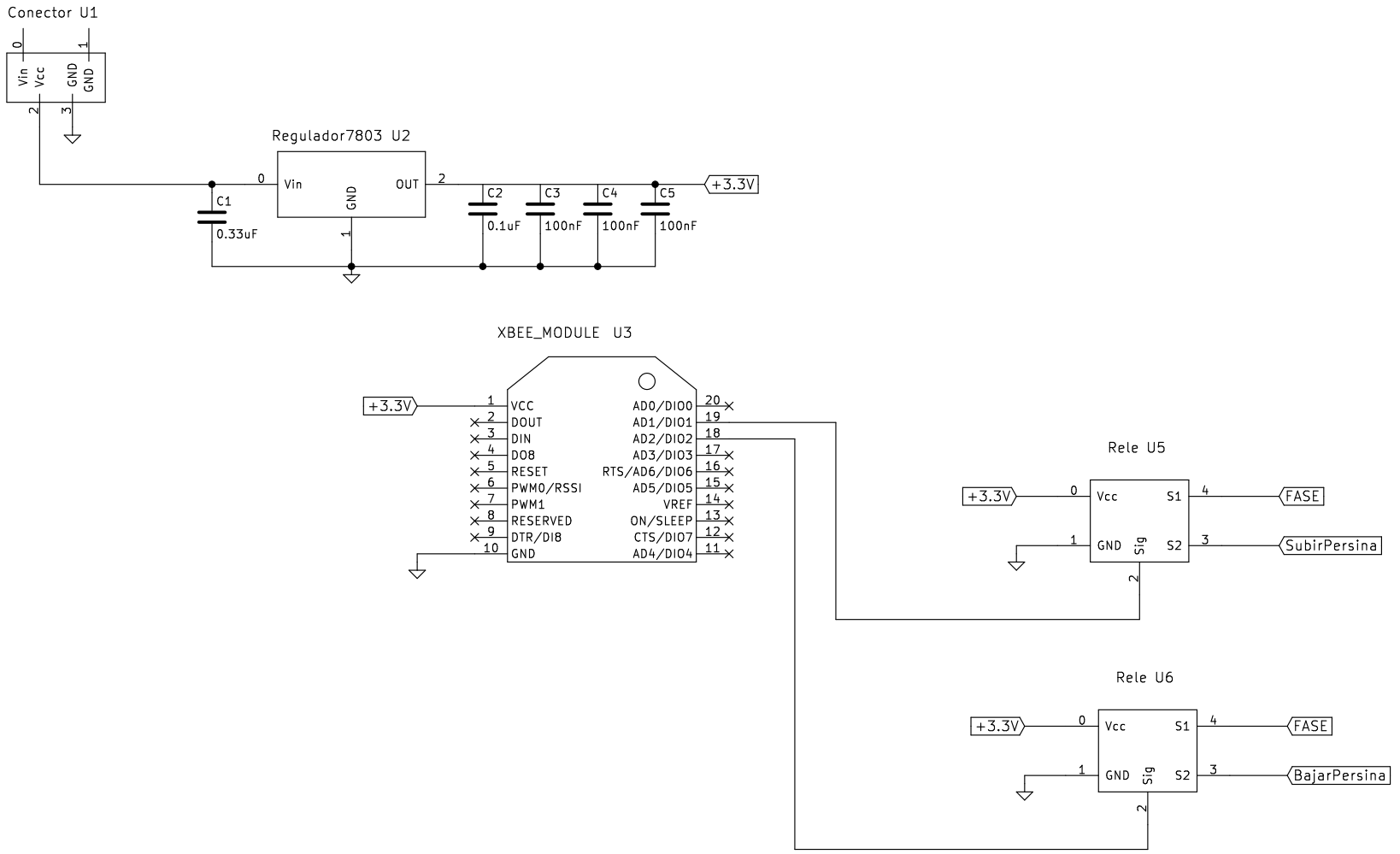
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Entrada – Módulo Actuador			4



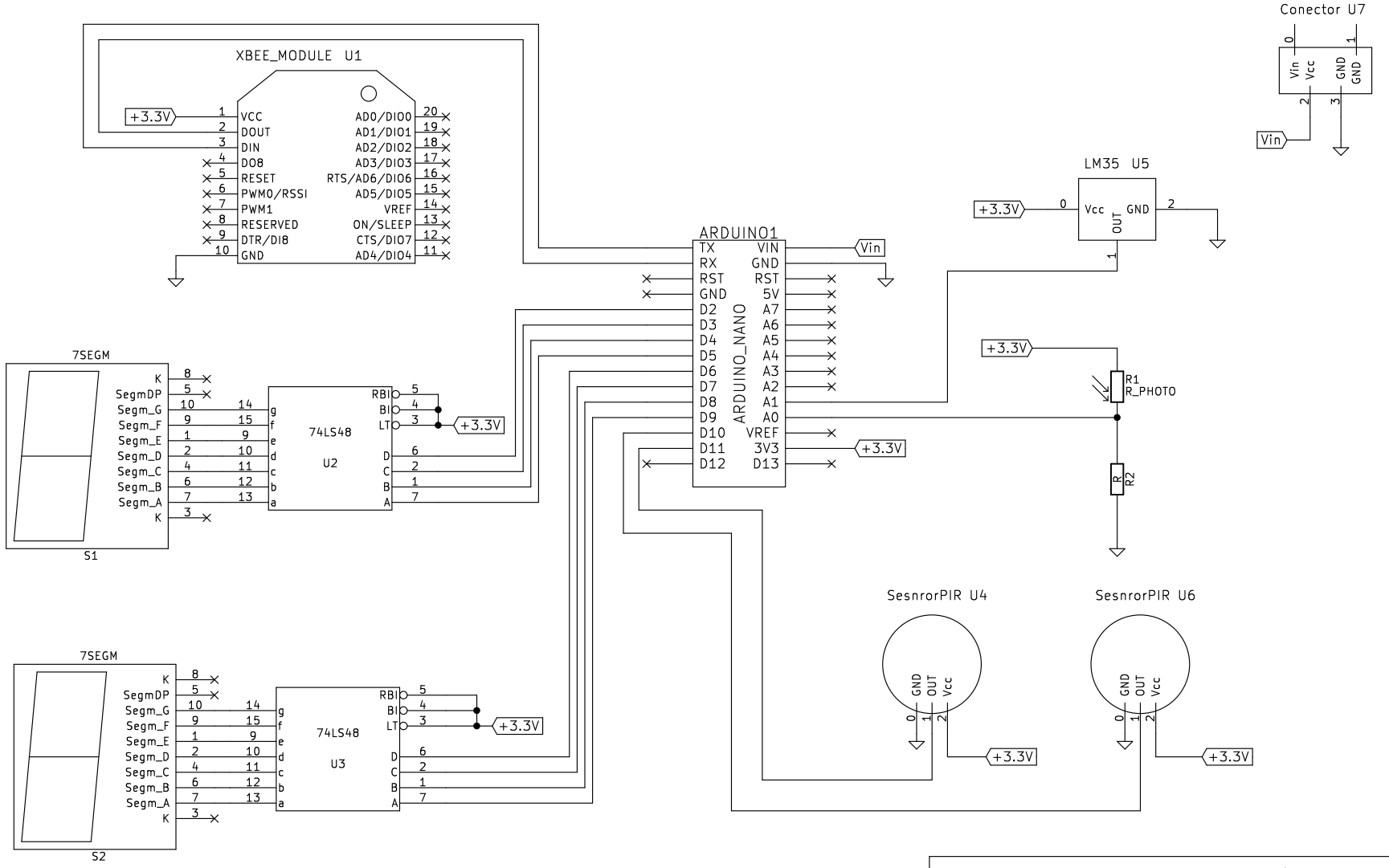
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Cocina – Módulo Habitación			5



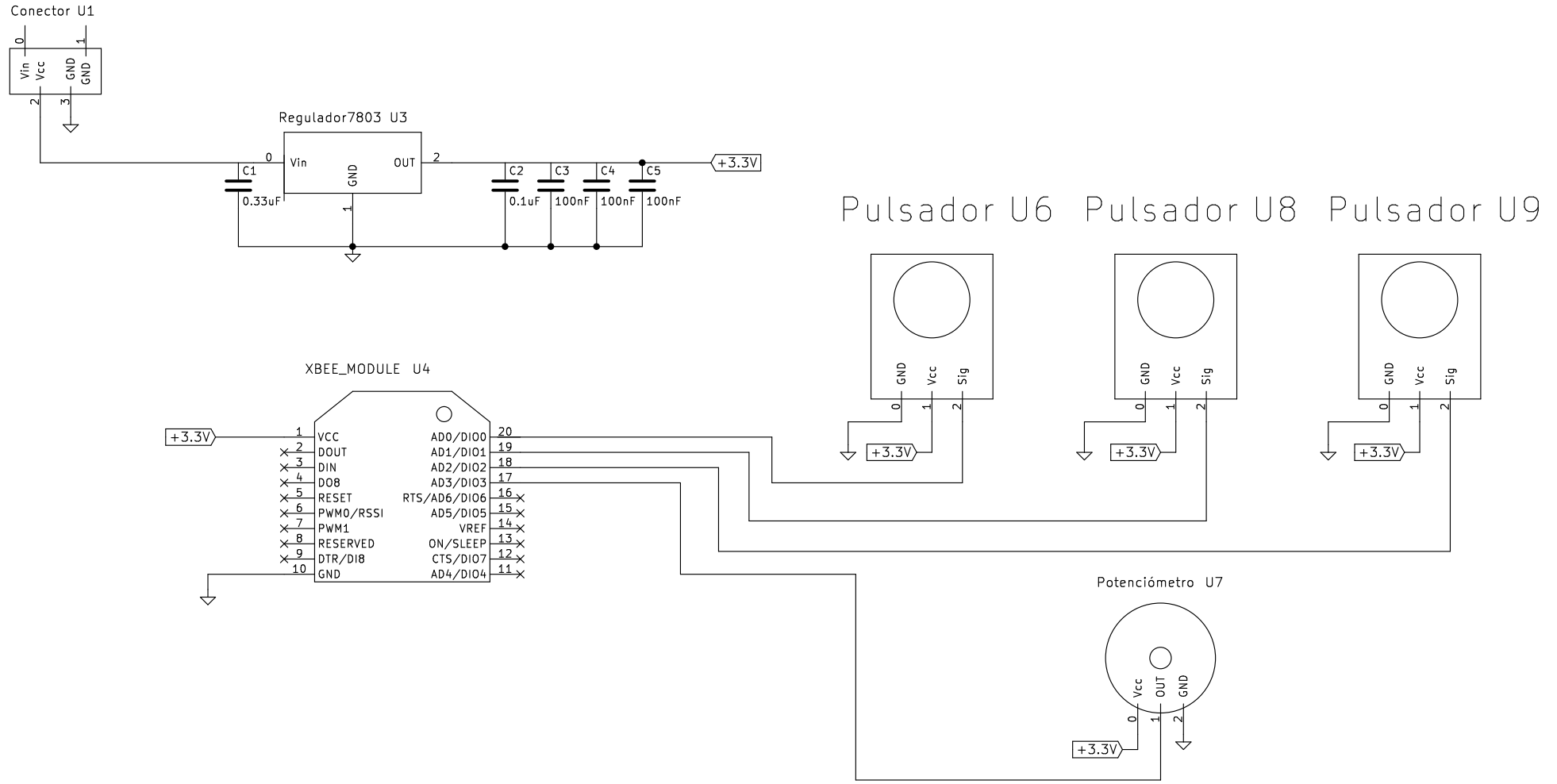
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE-EN-DIN			Nº Plano
Escala	Cocina – Módulo Pulsador			6



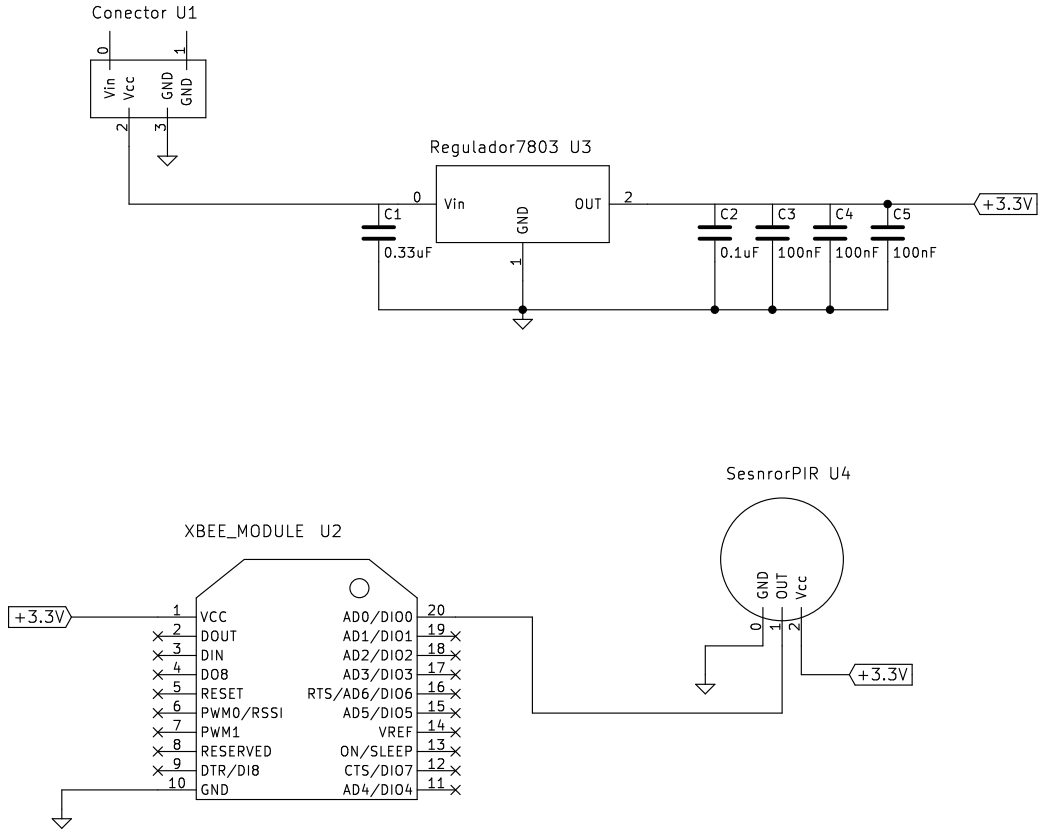
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Cocina – Módulo Actuador			7



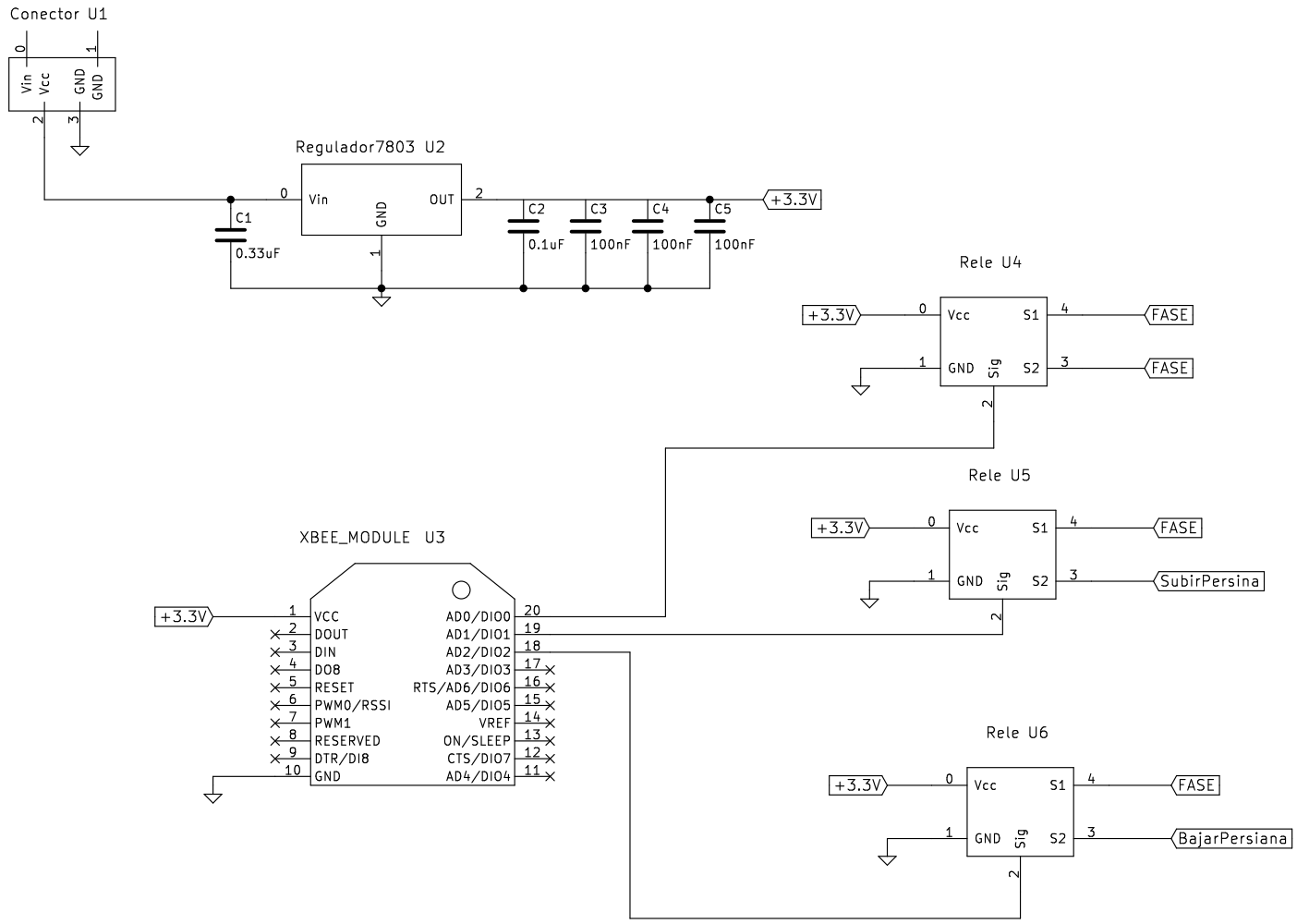
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Sala – Módulo Habitación			8



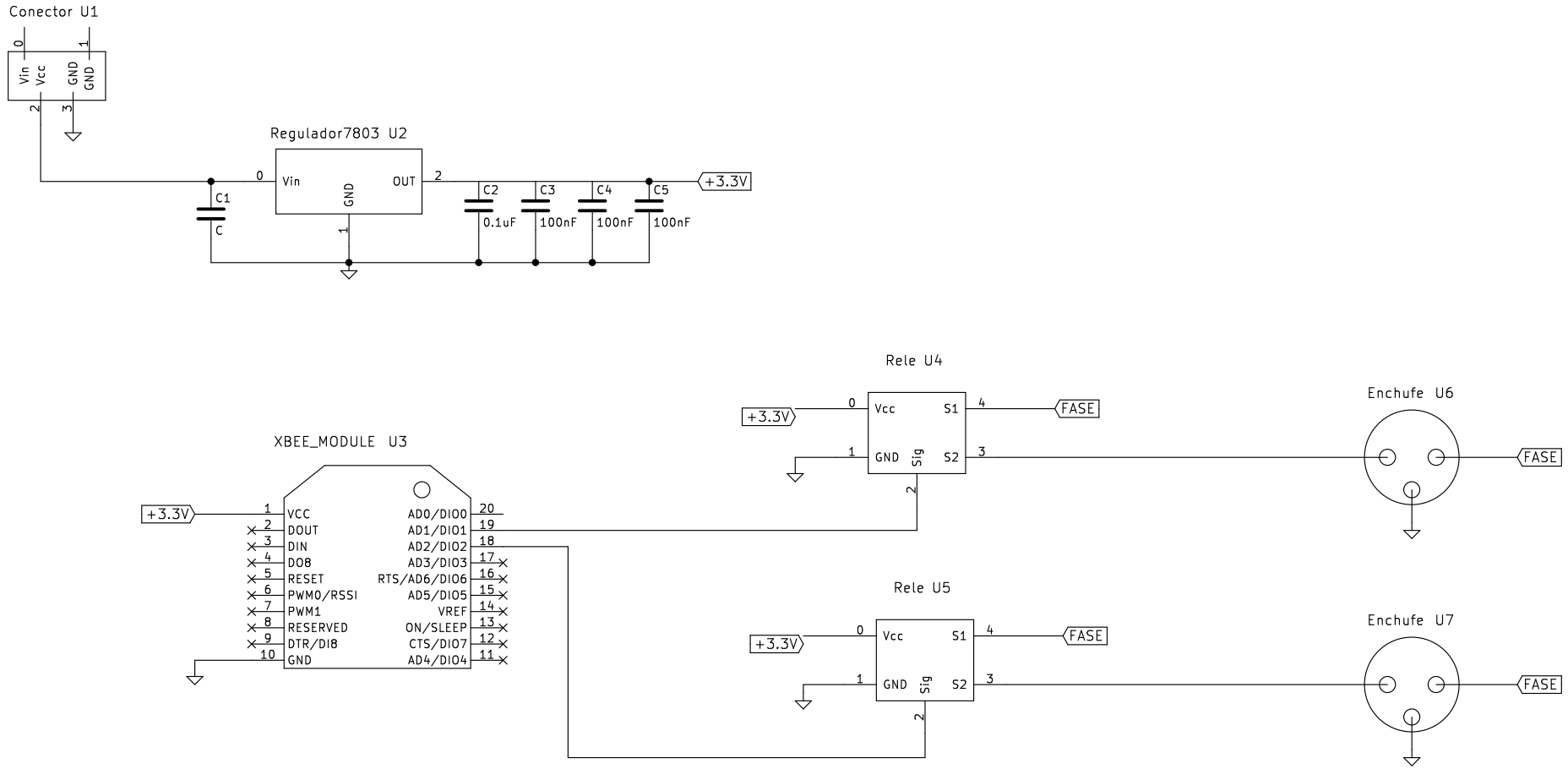
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE-EN-DIN			Nº Plano
Escala	Sala – Módulo Pulsador			9



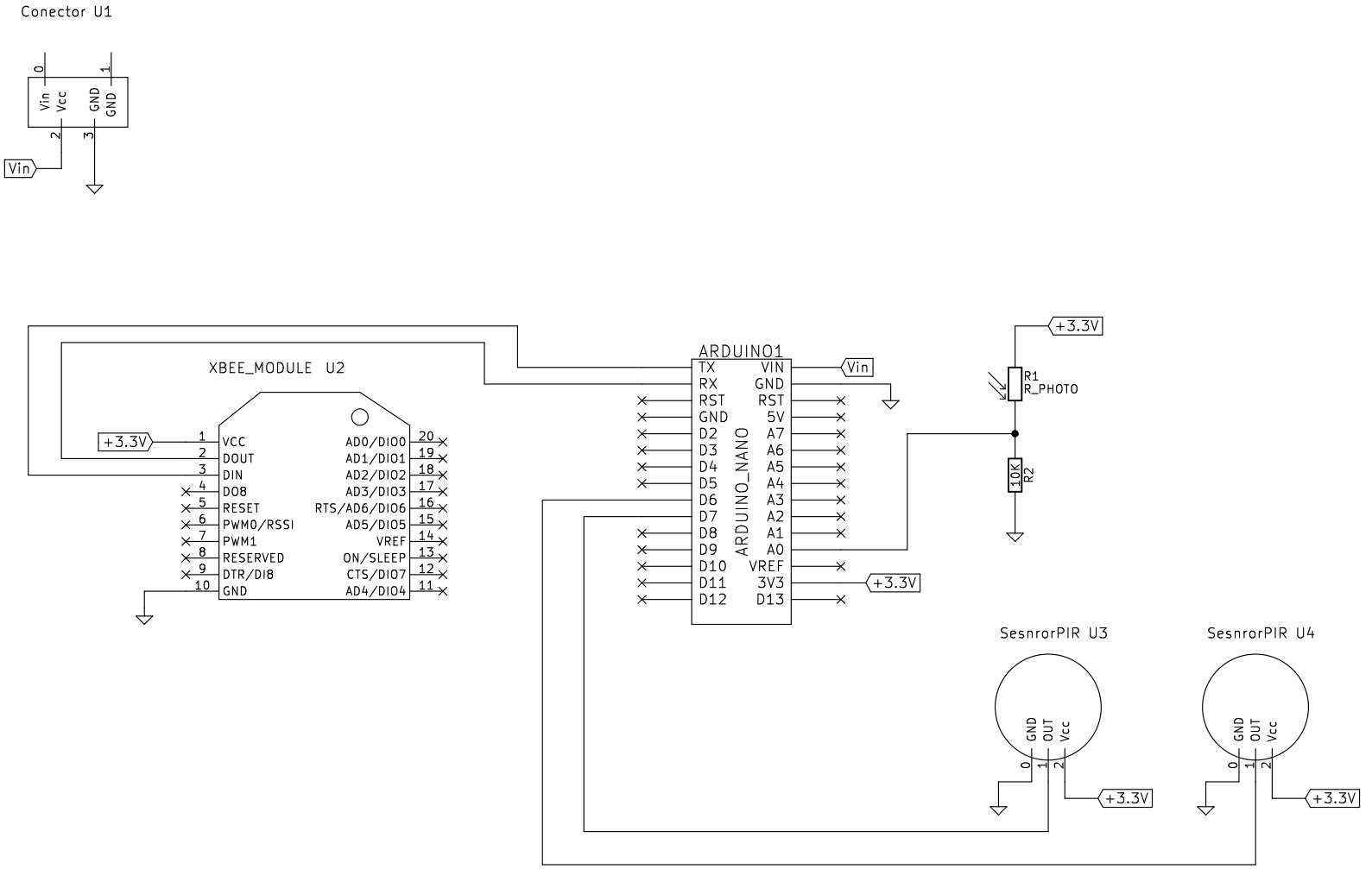
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Sala – Módulo Movimiento			10



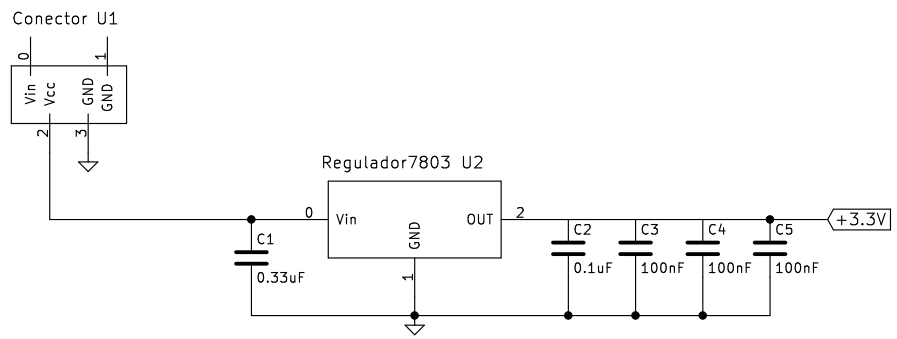
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Sala – Módulo Actuador			11

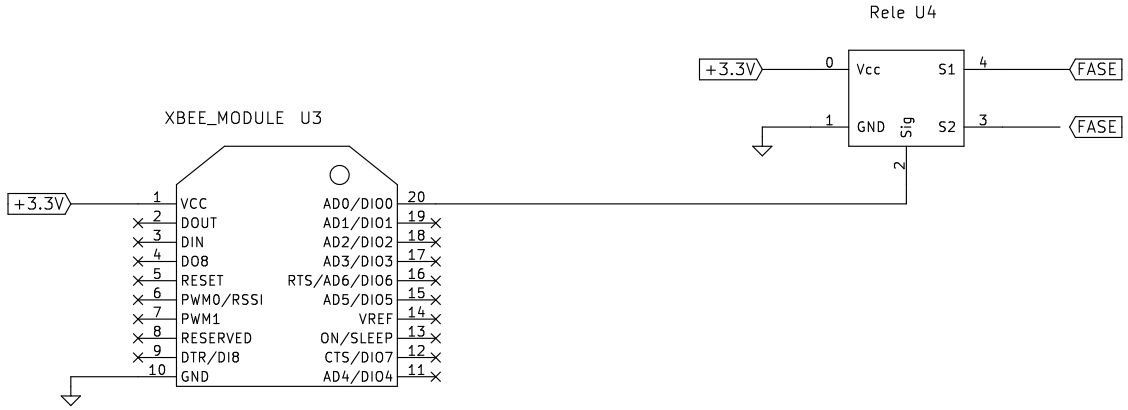


Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE-EN-DIN			Nº Plano
Escala	Sala- Módulo Actuador 2			12

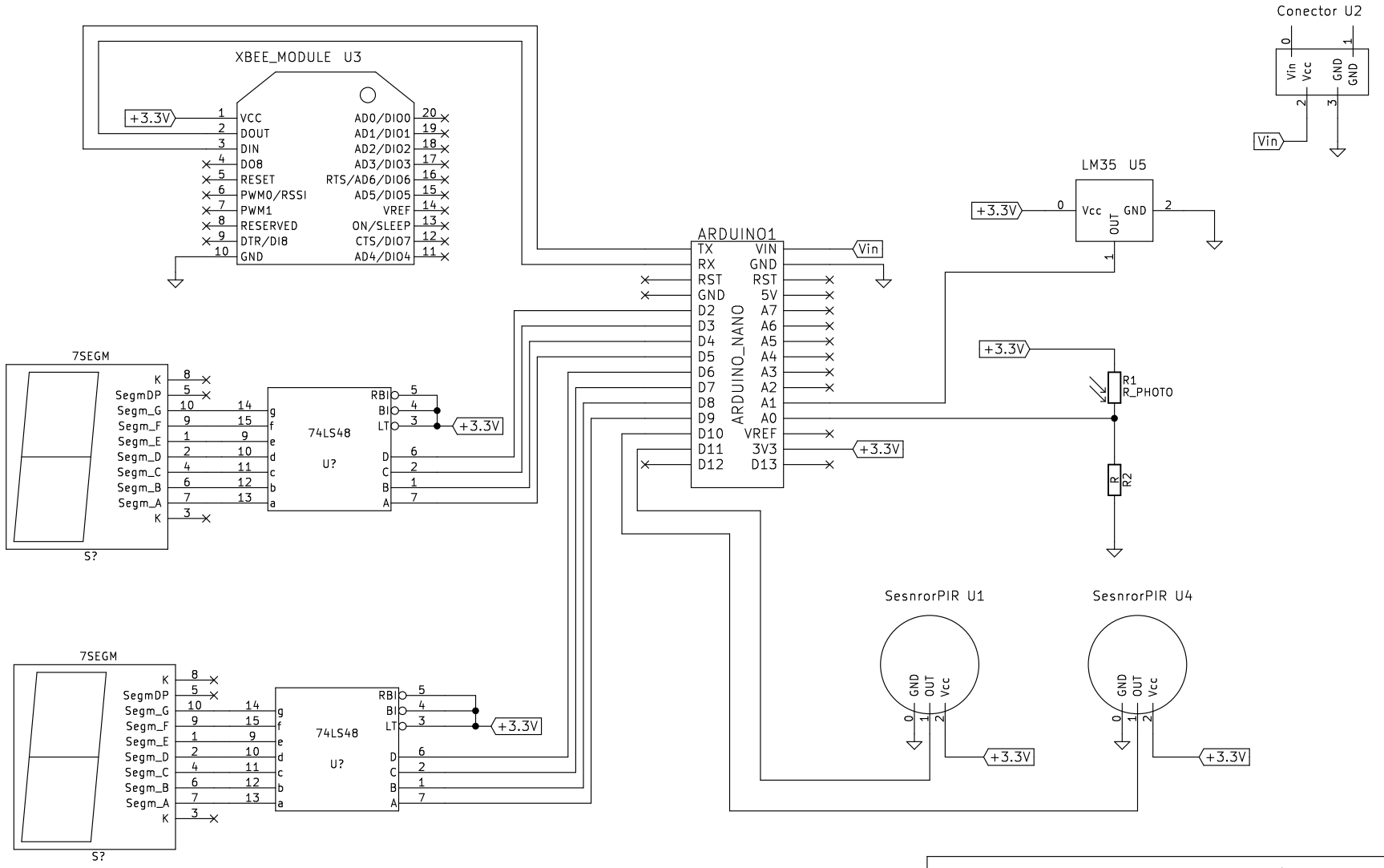


Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Zonas Comunes – Módulo Habitación			13

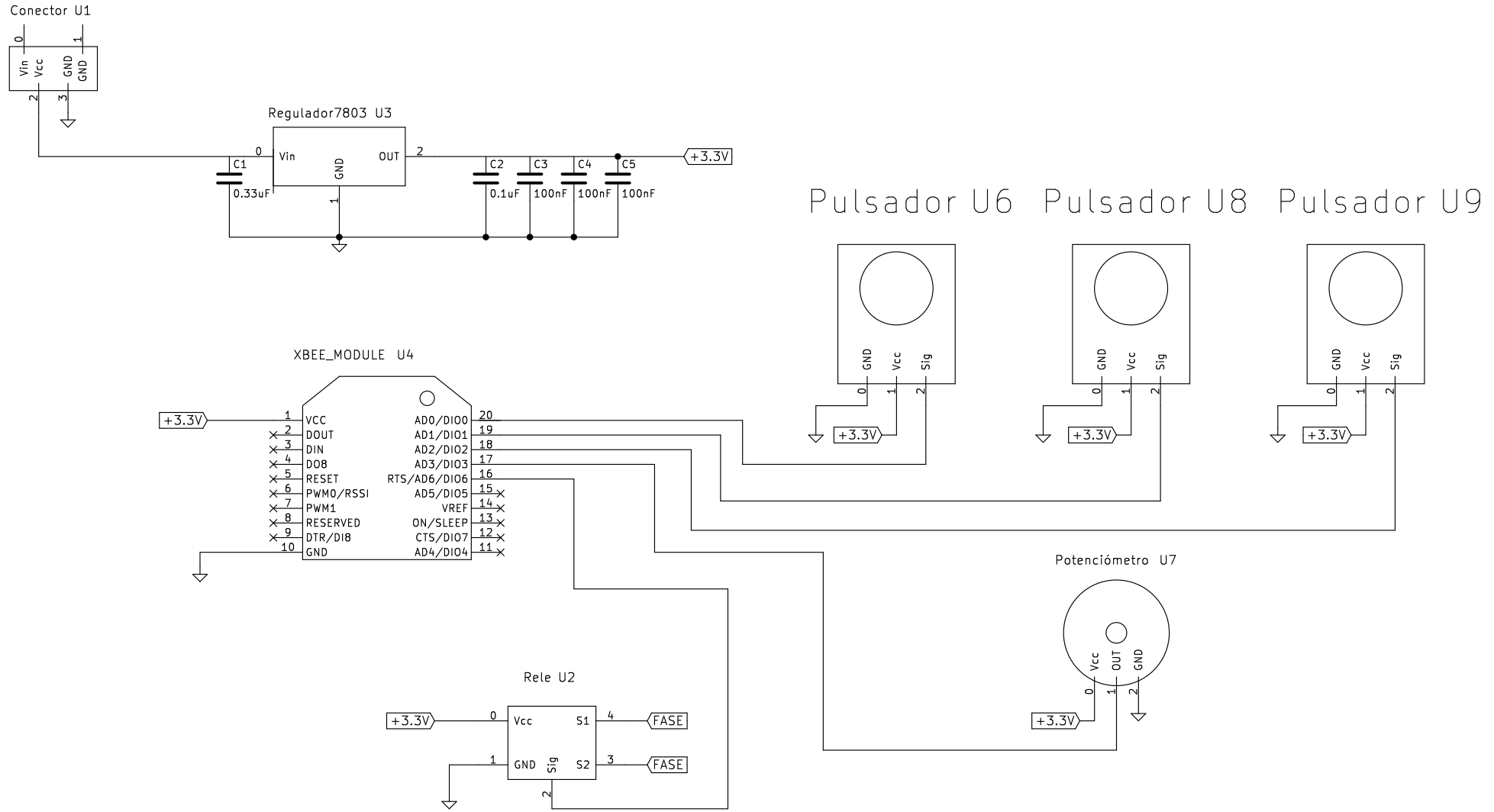




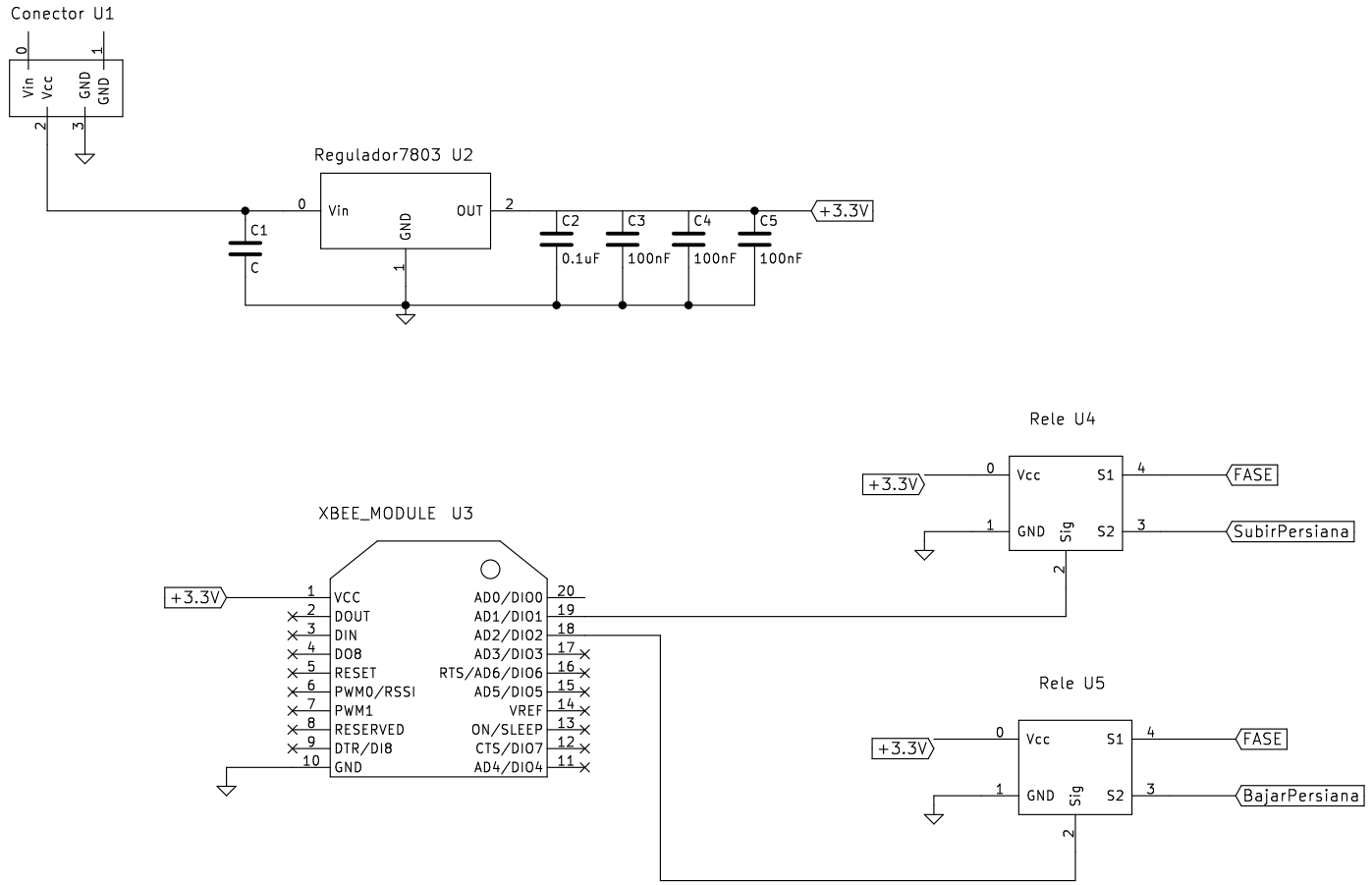
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE-EN-DIN			Nº Plano
Escala	Zonas Comunes – Módulo Actuador			14



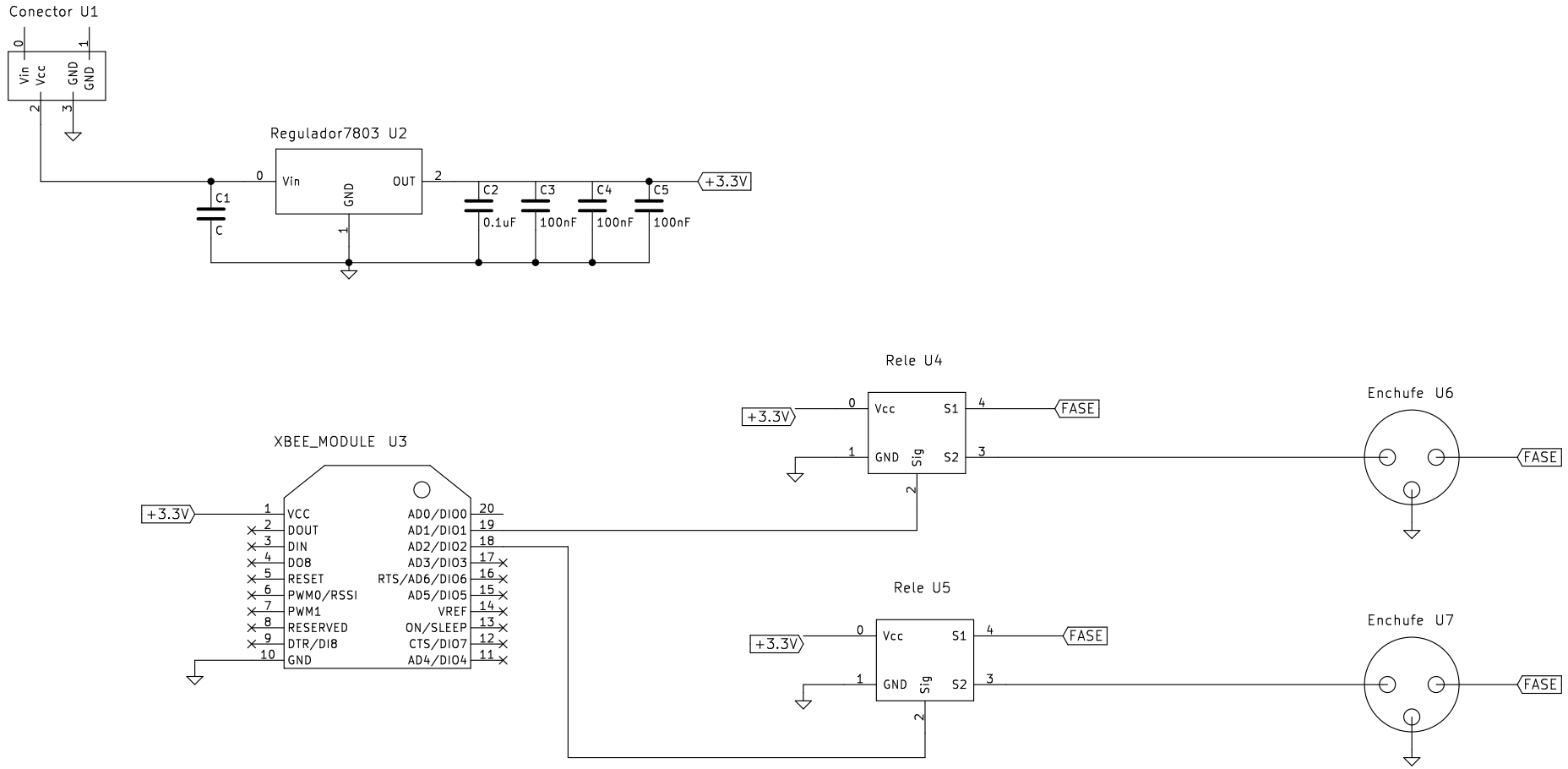
Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Habitación – Módulo Habitación			15



Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE-EN-DIN			Nº Plano
Escala	Habitación – Módulo Pulsador			16



Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE–EN–DIN			Nº Plano
Escala	Habitación – Módulo Actuador 1			17



Diseño de un sistema domótico de bajo coste para la gestión energética de una vivienda unifamiliar				
	Fecha	Autor	Universidad de La Laguna	Grado en Ingeniería Electrónica Industrial y Automática
Dibujado	02/07/2017	Julio Alejandro		
Comprobado	02/07/2017	Marichal Lorenzo		
Id. s. normas	UNE-EN-DIN			Nº Plano
Escala	Habitación – Módulo Actuador 2			18

Presupuesto

Cocina				
Material	Modelo	Unidad	Precio(euros)	Total
Módulo Habitación				
Arduino	Arduino Nano	1	22	22
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Sensor de movimiento	HC-SR501	2	11,5	23
Sensor de luz	LDR	1	1,2	1,2
Resistencia	10 kΩ	1	0,08	0,08
Conector	2 pines	1	0,5	0,5
			Precio Total	67,51
Módulo Pulsador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Pulsador	Sensor capacitivo	2	2	4
Relé	-	1	3,88	3,88
Conector	2 pines	1	0,5	0,5
			Precio Total	31,93
Módulo Actuador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Relé	5V 30A	2	3,88	7,76
Conector	2 pines	1	0,5	0,5
			Precio Total	31,81
Persianas				
Motor	ES35-10	1	39,75	39,75
		Precio Total Cocina		171

Entrada				
Material	Modelo	Unidad	Precio(euros)	Total
Módulo Casa				
Arduino	Arduino Nano	1	22	22
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Sensor de movimiento	HC-SR501	2	11,5	23
Sensor de luz	LDR	1	1,2	1,2
Resistencia	10 kΩ	1	0,08	0,08
Conector	2 pines	1	0,5	0,5
			Precio Total	67,51
Módulo Actuador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Relé	5V 30A	1	3,88	3,88
Conector	2 pines	1	0,5	0,5
			Precio Total	27,93
		Precio Total Entrada		95,44

Zonas Comunes				
Material	Modelo	Unidad	Precio(euros)	Total
Módulo Habitación				
Arduino	Arduino Nano	1	22	22
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Sensor de movimiento	HC-SR501	2	11,5	23
Sensor de luz	LDR	1	1,2	1,2
Resistencia	10 kΩ	1	0,08	0,08
Conector	2 pines	1	0,5	0,5
			Precio Total	67,51

Módulo Actuador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Relé	5V 30A	1	3,88	3,88
Conector	2 pines	1	0,5	0,5
			Precio Total	27,93
		Precio Total Zonas Comunes		381,76

Sala				
Material	Modelo	Unidad	Precio(euros)	Total
Módulo Habitación				
Arduino	Arduino Nano	1	22	22
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Sensor de movimiento	HC-SR501	2	11,5	23
Sensor de luz	LDR	1	1,2	1,2
Resistencia	10 kΩ	1	0,08	0,08
Sensor de temperatura	LM35	1	1,36	1,36
Display	7-Segmentos	2	1,05	2,1
Decodificador BCD - 7 Segmentos	74LS48	2	0,95	1,9
Conector	2 pines	1	0,5	0,5
			Precio Total	72,87

Módulo Pulsador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Pulsador	Sensor capacitivo	3	2	6
Potenciómetro	10 kΩ	1	1,69	1,69
Conector	2 pines	1	0,5	0,5
			Precio Total	31,74

Módulo de movimiento				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Sensor de movimiento	HC-SR501	1	11,5	11,5
Conector	2 pines	1	0,5	0,5
			Precio Total	35,55

Módulo Actuador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Relé	-	2	3,88	7,76
Conector	2 pines	1	0,5	0,5
			Precio Total	31,81

Módulo Actuador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Relé	5V 30A	3	3,88	11,64
Conector	2 pines	1	0,5	0,5
			Precio Total	35,69
Persianas				
Motor	G-CO-Cable	1	157	157
		Precio Total Sala		364,66

Habitación				
Material	Modelo	Unidad	Precio	Total
Módulo Habitación				
Arduino	Arduino Nano	1	22	22
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Sensor de movimiento	HC-SR501	2	11,5	23
Sensor de luz	LDR	1	1,2	1,2
Resistencia	10 kΩ	1	0,08	0,08
Sensor de temperatura	LM35	1	1,36	1,36
Display	7-Segmentos	2	1,05	2,1
Decodificador BCD - 7 Segmentos	74LS48	2	0,95	1,9
Conector	2 pines	1	0,5	0,5
			Precio Total	72,87

Módulo Pulsador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Pulsador	Sensor capacitivo	3	2	6
Potenciómetro	10 kΩ	1	1,69	1,69
Conector	2 pines	1	0,5	0,5
			Precio Total	31,74
Módulo Actuador				
Módulo XBee	Programmable XBee ZigBee Through-Hole (Wire Antenna)	1	20,73	20,73
Regulador de tensión	7803	1	1	1
Condensador	100 pF	3	0,16	0,48
Condensador	0,33 uF	1	0,16	1,2
Condensador	0,1 uF	1	0,14	0,14
Relé	5V 30A	2	3,88	7,76
Conector	2 pines	1	0,5	0,5
			Precio Total	31,81
Persianas				
Motor	ES35-10	1	39,75	39,75
		Precio Total Habitación		207,98

Precio Total en la vivienda	1636,80

Conclusiones

El presente proyecto ha finalizado con las siguientes conclusiones:

En primer lugar, el sistema domótico diseñado cumplirá con los objetivos principales que se han presentado en este proyecto.

En segundo lugar, el sistema creado ofrecerá al usuario una amplia flexibilidad para adaptarse a las necesidades de cada vivienda, al ser diseñado en un formato modular. El sistema aportará la automatización de luminarias y persianas a partir de mecanismos automáticos y manuales. Además, concederá la posibilidad de instalar un sistema de climatización capaz de regular la temperatura de una zona en particular.

En tercer lugar, el sistema se caracteriza por implementar entre sus dispositivos una conexión inalámbrica establecida por el protocolo de comunicación ZigBee, a través de los módulos XBee. Este protocolo ha aportado una alta fiabilidad de transmisión de datos, además de un alto nivel de seguridad ante ataques externos a la red.

En conclusión, el sistema domótico diseñado tendrá la posibilidad de adaptarse a diferentes medios, a través de la implementación de los módulos que componen este sistema. También tendrá la capacidad de ser mejorado incluyendo módulos con diferentes funcionalidades, mejorando los sistemas diseñados o incluyendo nuevos sistemas de control de la vivienda.

Anexos

Índice

1. Datasheet Sensor de temperatura: LM35

LM35 Precision Centigrade Temperature Sensors

1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates from 4 V to 30 V
- Less than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

2 Applications

- Power Supplies
- Battery Management
- HVAC
- Appliances

3 Description

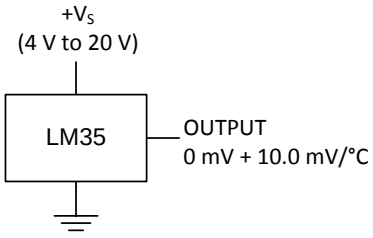
The LM35 series are precision integrated-circuit temperature devices with an output voltage linearly-proportional to the Centigrade temperature. The LM35 device has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from the output to obtain convenient Centigrade scaling. The LM35 device does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55°C to 150°C temperature range. Lower cost is assured by trimming and calibration at the wafer level. The low-output impedance, linear output, and precise inherent calibration of the LM35 device makes interfacing to readout or control circuitry especially easy. The device is used with single power supplies, or with plus and minus supplies. As the LM35 device draws only 60 μA from the supply, it has very low self-heating of less than 0.1°C in still air. The LM35 device is rated to operate over a −55°C to 150°C temperature range, while the LM35C device is rated for a −40°C to 110°C range (−10° with improved accuracy). The LM35-series devices are available packaged in hermetic TO transistor packages, while the LM35C, LM35CA, and LM35D devices are available in the plastic TO-92 transistor package. The LM35D device is available in an 8-lead surface-mount small-outline package and a plastic TO-220 package.

Device Information(1)

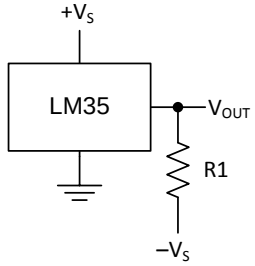
PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM35	TO-CAN (3)	4.699 mm × 4.699 mm
	TO-92 (3)	4.30 mm × 4.30 mm
	SOIC (8)	4.90 mm × 3.91 mm
	TO-220 (3)	14.986 mm × 10.16 mm

(1) For all available packages, see the orderable addendum at the end of the datasheet.

Basic Centigrade Temperature Sensor (2°C to 150°C)



Full-Range Centigrade Temperature Sensor



Choose $R_1 = -V_S / 50\ \mu A$
$V_{OUT} = 1500$ mV at 150°C
$V_{OUT} = 250$ mV at 25°C
$V_{OUT} = -550$ mV at −55°C

6 Specifications

6.1 Absolute Maximum Ratings

over operating free-air temperature range (unless otherwise noted)(1)(2)

		MIN	MAX	UNIT
Supply voltage		–0.2	35	V
Output voltage		–1	6	V
Output current			10	mA
Maximum Junction Temperature, $T_J$max			150	°C
Storage Temperature, T_{stg}	TO-CAN, TO-92 Package	–60	150	°C
	TO-220, SOIC Package	–65	150	

(1) If Military/Aerospace specified devices are required, please contact the Texas Instruments Sales Office/ Distributors for availability and specifications.
(2) Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions.

6.2 ESD Ratings

			VALUE	UNIT
$V_{(ESD)}$	Electrostatic discharge	Human-body model (HBM), per ANSI/ESDA/JEDEC JS-001(1)	±2500	V

(1) JEDEC document JEP155 states that 500-V HBM allows safe manufacturing with a standard ESD control process.

6.3 Recommended Operating Conditions

over operating free-air temperature range (unless otherwise noted)

		MIN	MAX	UNIT
Specified operating temperature: T_{MIN} to T_{MAX}	LM35, LM35A	–55	150	°C
	LM35C, LM35CA	–40	110	
	LM35D	0	100	
Supply Voltage (+V_S)		4	30	V

6.4 Thermal Information

THERMAL METRIC(1)(2)		LM35				UNIT
		NDV	LP	D	NEB	
		3 PINS		8 PINS	3 PINS	
$R_{\theta JA}$	Junction-to-ambient thermal resistance	400	180	220	90	°C/W
$R_{\theta JC(top)}$	Junction-to-case (top) thermal resistance	24	—	—	—	

(1) For more information about traditional and new thermal metrics, see the *IC Package Thermal Metrics* application report, SPRA953.
(2) For additional thermal resistance information, see *Typical Application*.

2. Datasheet resistencia LDR

CdS Photoconductive Cell

GL5-05AP02(1)

Coating type

◆ Absolute Maximum Ratings

Ta=25℃

Parameter	Value	Unit
Power Dissipation at 25℃	100	mW
Impressed Voltage applied at 0lx	150	V
Ambient Temperature	-30 ~ +70	℃
Soldering Temperature**	260	℃

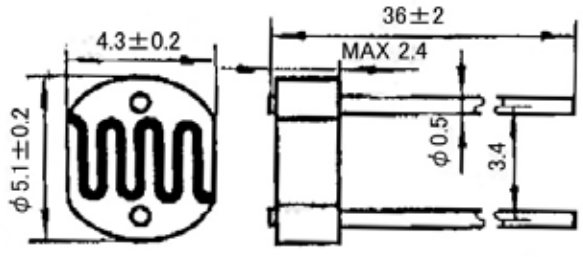
** Soldering time should be within 3 sec and the position should be min. 5.0mm from the lead root.

◆ Electro-optical Characteristics

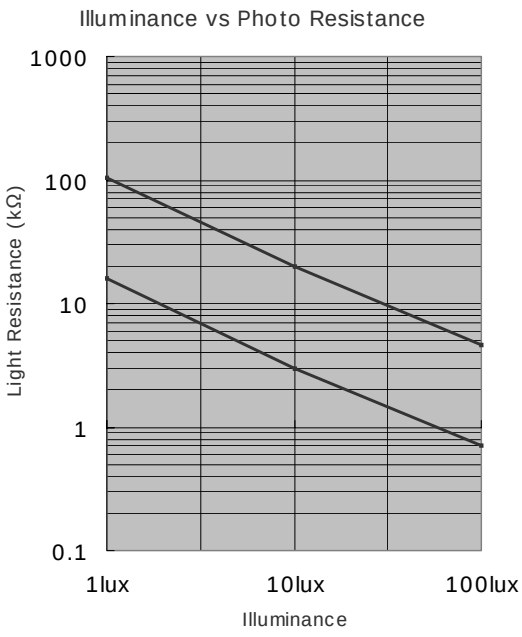
Ta = 25℃

Parameter	Symbol	Value	Unit	Condition
Light Resistance at 10 lx	R_{10}	3 ~ 20	kΩ	2856K
Dark Resistance	Rd	Min.0.5	MΩ	20 sec. after blocking 100lx light.
Peak Spectral sensitivity	λ_P	560	nm	
Gamma (10-100)	$\gamma_{(100/10)}$	0.6 $^{\pm 0.1}$		
Rise Time	T_R	20	ms	
Decay Time	T_F	30	ms	

◆ Outline and Feature



Unit: mm



Revision: (1) Model number is changed [2004.09.14]

3. Datasheet Sesnor de consumo eléctrico: SCT – 013

Model: SCT-013 **Rated input current: 5A/100A**

Characteristics: Openning size: 13mm*13mm,
Non-linerity±3%（10%—120% of rated input current）
1m leading wire, standard Φ3.5 three core plug output.
Current output type and voltage output type (voltage output type built-in sampling resistor)

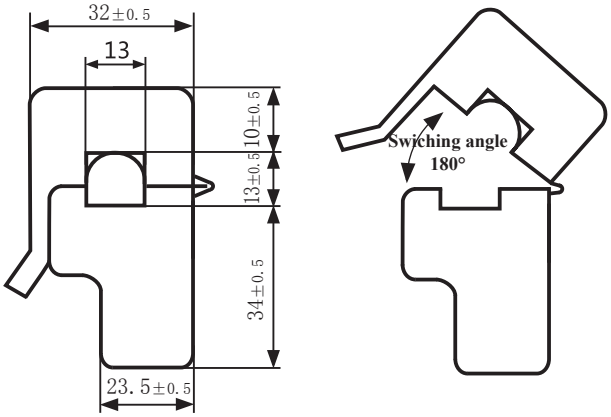
Purpose: Used for current measurement, monitor and protection for AC motor, lighting equipment, air compressor etc

Core material: ferrite

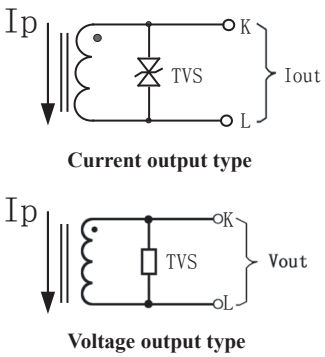
Mechanical strength: the number of switching is not less than 1000 times(test at 25℃)

Safty index: Dielectric strength(between shell and output)1000V AC/1min
Fire resistance property: In accordance with UL94-Vo
Work temperature: -25℃～+70℃

Outline size diagram: (in mm)



Front View **Side View**



Schematic diagram

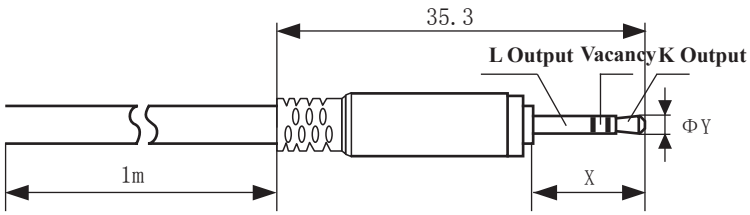


Diagram for standard three-core plug

Three-core plug size

	X	Y	
2.5mm Audio Plug	11.9	2.5	Optional
3.5mm Audio Plug	15.0	3.5	standard

Table of technical parameter:

Model	SCT-013-000	SCT-013-005	SCT-013-010	SCT-013-015	SCT-013-020
Input current	0-100A	0-5A	0-10A	0-15A	0-20A
Output type	0-50mA	0-1V	0-1V	0-1V	0-1V
Model	SCT-013-025	SCT-013-030	SCT-013-050	SCT-013-060	SCT-013-000V
Input current	0-25A	0-30A	0-50A	0-60A	0-100A
Output type	0-1V	0-1V	0-1V	0-1V	0-1V

※ **Output type: voltage output type built-in sampling resistor, current output type built-in protective diode.**

Tel: 86-13933609279 Fax: 86-7929499-804 skype:macymeng1
Web (China): www.yhdc.com Web (Latvia): www.yhdc.lv

4. Datasheet diodo 1N4001

Rectifiers 1N4001G to 1N4007G

FEATURES

- Glass passivated
- High maximum operating temperature
- Low leakage current
- Excellent stability
- Available in ammo-pack.

DESCRIPTION

Rugged glass package, using a high temperature alloyed construction.

This package is hermetically sealed and fatigue free as coefficients of expansion of all used parts are matched.

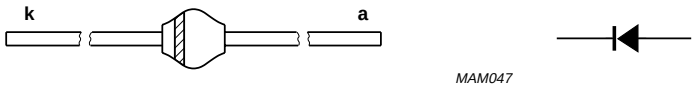


Fig.1 Simplified outline (SOD57) and symbol.

LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	CONDITIONS	MIN.	MAX.	UNIT
V_{RRM}	repetitive peak reverse voltage				
	1N4001G		–	50	V
	1N4002G		–	100	V
	1N4003G		–	200	V
	1N4004G		–	400	V
	1N4005G		–	600	V
	1N4006G		–	800	V
	1N4007G		–	1000	V
V_R	continuous reverse voltage				
	1N4001G		–	50	V
	1N4002G		–	100	V
	1N4003G		–	200	V
	1N4004G		–	400	V
	1N4005G		–	600	V
	1N4006G		–	800	V
	1N4007G		–	1000	V
$I_{F(AV)}$	average forward current	averaged over any 20 ms period; T_{amb} = 75 °C; see Fig.2	–	1.00	A
		averaged over any 20 ms period; T_{amb} = 100 °C; see Fig.2	–	0.75	A
I_F	continuous forward current	T_{amb} = 75 °C; see Fig.2	–	1.00	A
I_{FRM}	repetitive peak forward current		–	10	A
I_{FSM}	non-repetitive peak forward current	half sinewave; 60 Hz	–	30	A
T_{stg}	storage temperature		–65	+175	°C
T_j	junction temperature		–65	+175	°C

5. Motor Persianas Domesticas

Motor de persiana de 35mm 10Nm (25kg)

Descripción del producto.

El motor tubular estándar de 35mm 10Nm es un motor que permite controlar el proceso de subida y bajada de una **persiana (mini), cortina enrollable o toldo**, facilitando así su apertura y su cierre, y, haciendo así, más cómodo su uso. Son capaces de levantar hasta **25Kg**.

Motores para persianas adecuados para usar también en toldos, estores, pantallas de proyección, screens, foscurits, cortinas enrollables y persianas MINI de PVC, o persianas de aluminio con eje metálico, generalmente de 40mm (puede variar).

Características técnicas.

Par motor (Nm)	10
Velocidad (rpm)	17
Diámetro (mm)	35
Voltaje (V)	230
Frecuencia (Hz)	50
Potencia (W)	121
Intensidad (A)	0,53
Tiempo de funcionamiento continuado	4
Protección (IP)	IP44
Número máximo de vueltas	36
Longitud (L1/L2) (mm)	457/432

6. Motor de cortina

Motor de cortina

Advertencias:

- Lea este manual atentamente antes de realizar cualquier conexión o modificación en la instalación.
- No sumerja el motor en agua.
- No taladre el motor.
- No golpear el motor.
- Mantener alejado de productos corrosivos.

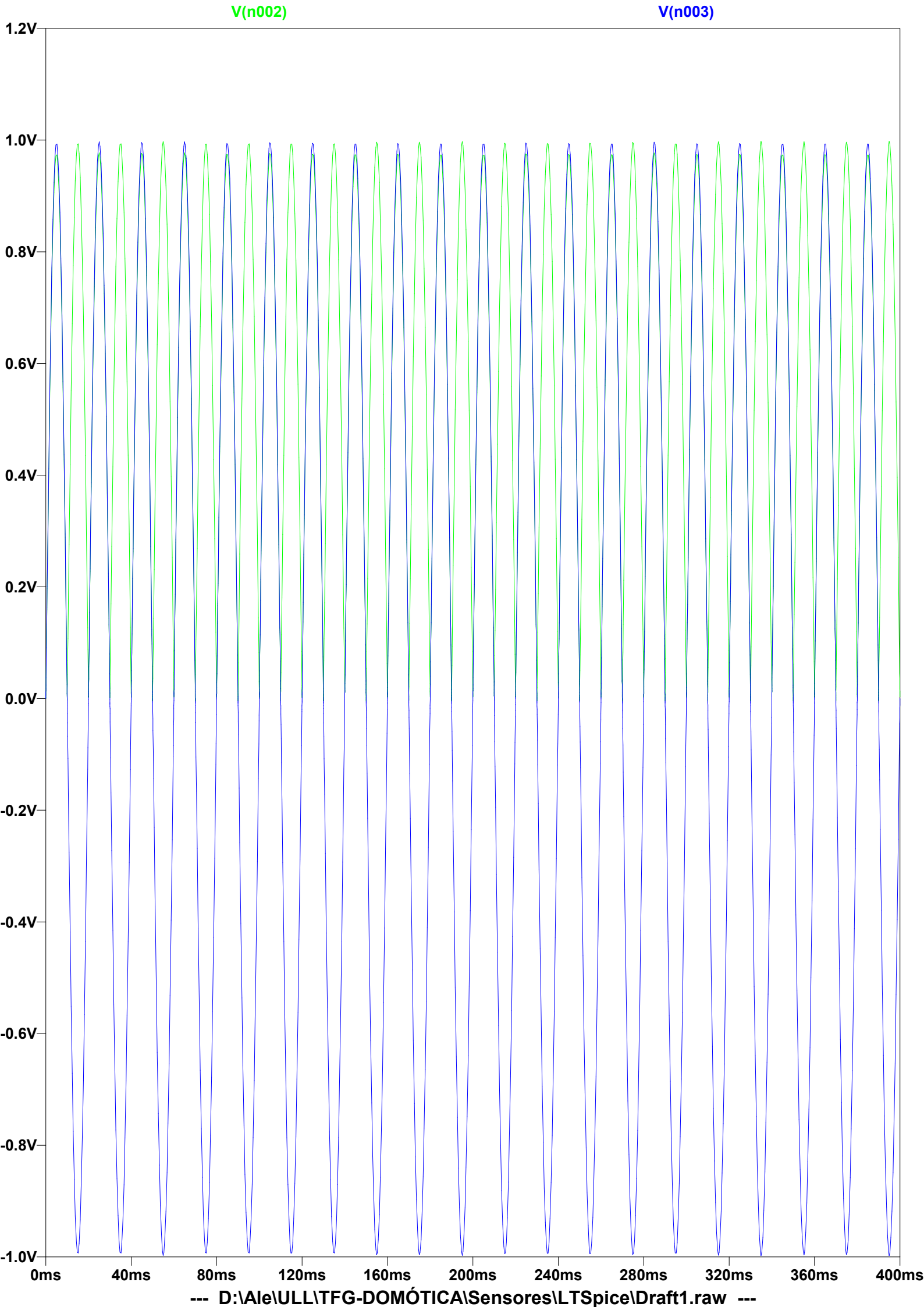
Descripción del producto.

El motor de cortinas de la gama estándar, le permitirá controlar su cortina de forma automática, utilizando un simple control remoto, como por ejemplo: un mando.

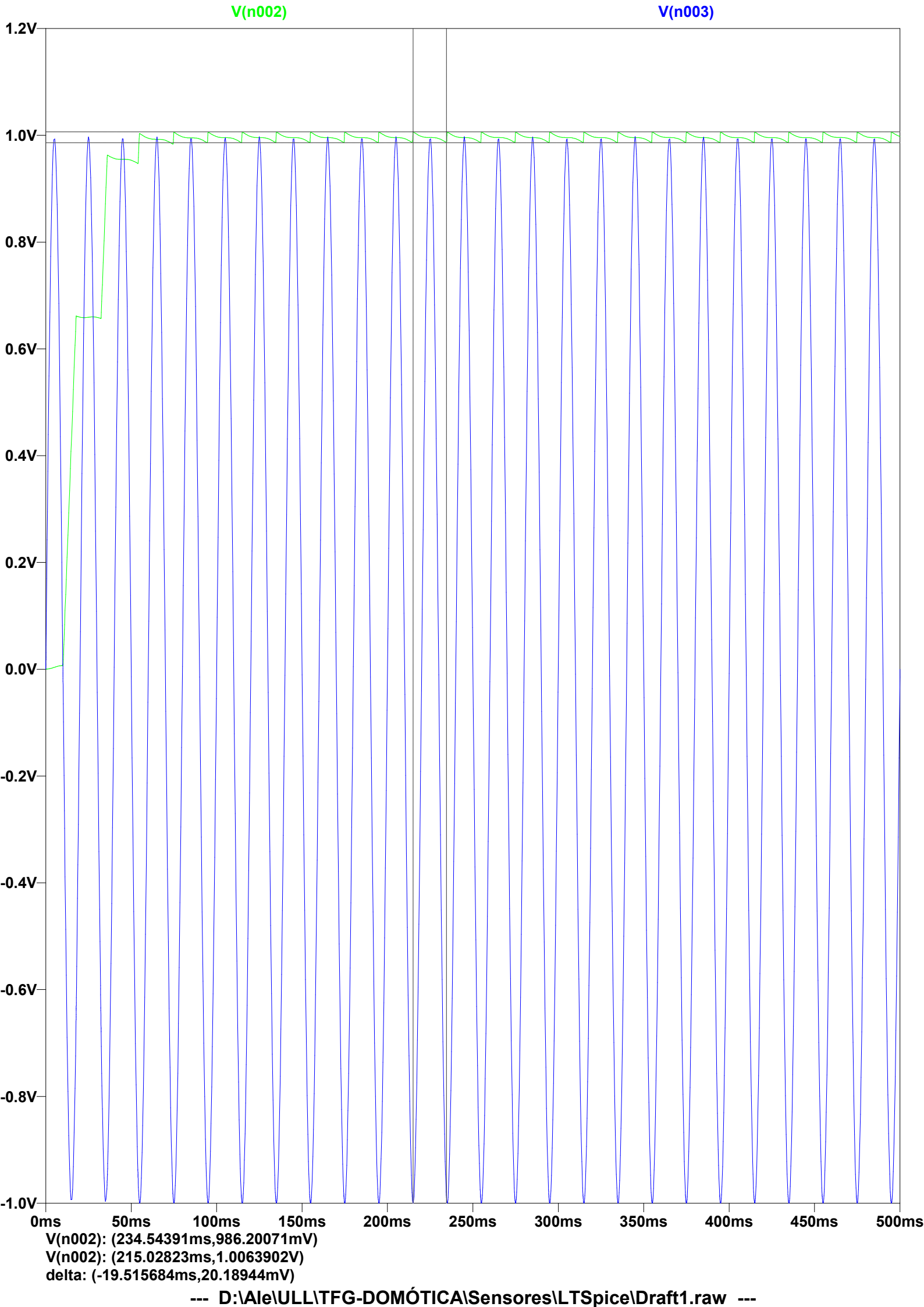
Características técnicas.

Par motor (Nm)	1.2
Velocidad (cm/s)	20
Voltaje (V)	230
Frecuencia (Hz)	50
Potencia (W)	75
Intensidad (A)	0.34
Frecuencia	433MHz
Protección (IP)	IP20

7. Rectificador de precisión: Señal rectificada


V(n002)
V(n003)
1.2V
1.0V
0.8V
0.6V
0.4V
0.2V
0.0V
-0.2V
-0.4V
-0.6V
-0.8V
-1.0V
0ms
40ms
80ms
120ms
160ms
200ms
240ms
280ms
320ms
360ms
400ms
--- D:\Ale\ULL\TFG-DOMÓTICA\Sensores\LTSpice\Draft1.raw ---

8. Rectificador de precisión: Señal rectificada con filtro

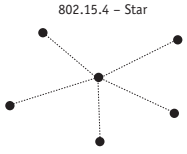

V(n002)
V(n003)
1.2V
1.0V
0.8V
0.6V
0.4V
0.2V
0.0V
-0.2V
-0.4V
-0.6V
-0.8V
-1.0V
0ms
50ms
100ms
150ms
200ms
250ms
300ms
350ms
400ms
450ms
500ms
V(n002): (234.54391ms,986.20071mV)
V(n002): (215.02823ms,1.0063902V)
delta: (-19.515684ms,20.18944mV)
--- D:\Ale\ULL\TFG-DOMÓTICA\Sensores\LTSpice\Draft1.raw ---

9. XBee Serie 1 características

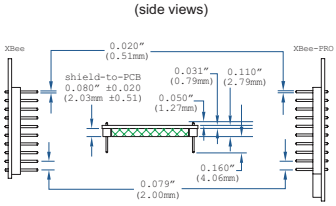
Platform	XBee® 802.15.4 (Series 1)	XBee-PRO® 802.15.4 (Series 1)	XBee-PRO® XSC
Performance			
RF Data Rate	250 kbps	250 kbps	10 kbps / 9.6 kbps
Indor/Urban Range	100 ft (30 m)	300 ft (100 m)	Up to 1200 ft (370 m)
Outdoor/RF Line-of-Sight Range	300 ft (100 m)	1 mi (1.6 km)	Up to 6 mi (9.6 km)
Transmit Power	1 mW (+0 dBm)	60 mW (+18 dBm)*	100 mW (+20 dBm)
Receiver Sensitivity (1% PER)	-92 dBm	-100 dBm	-106 dBm
Features			
Serial Data Interface	3.3V CMOS UART	3.3V CMOS UART	3.3V CMOS UART (5V Tolerant)
Configuration Method	API or AT Commands, local or over-the-air	API or AT Commands, local or over-the-air	AT Commands
Frequency Band	2.4 GHz	2.4 GHz	902 MHz to 928 MHz
Interference Immunity	DSSS (Direct Sequence Spread Spectrum)	DSSS (Direct Sequence Spread Spectrum)	FHSS (Frequency Hopping Spread Spectrum)
Serial Data Rate	1200 bps - 250 kbps	1200 bps - 250 kbps	1200 bps - 57.6 kbps
ADC Inputs	(6) 10-bit ADC inputs	(6) 10-bit ADC inputs	None
Digital I/O	8	8	None
Antenna Options	Chip, Wire Whip, U.FL, & RPSMA	Chip, Wire Whip, U.FL, & RPSMA	Wire Whip, U.FL, RPSMA
Networking & Security			
Encryption	128-bit AES	128-bit AES	No
Reliable Packet Delivery	Retries/Acknowledgments	Retries/Acknowledgments	Retries/Acknowledgements
IDs and Channels	PAN ID, 64-bit IEEE MAC, 16 Channels	PAN ID, 64-bit IEEE MAC, 12 Channels	PAN ID, 32-bit Address, 7 Channels
Power Requirements			
Supply Voltage	2.8 - 3.4VDC	2.8 - 3.4VDC	3.0 - 3.6VDC
Transmit Current	45 mA @ 3.3VDC	215 mA @ 3.3VDC	265 mA typical
Receive Current	50 mA @ 3.3VDC	55 mA @ 3.3VDC	65 mA typical
Power-Down Current	<10 uA @ 25° C	<10 uA @ 25° C	45 uA pin Sleep
Regulatory Approvals			
FCC (USA)	OUR-XBEE	OUR-XBEEPRO	MCQ-XBEEXSC
IC (Canada)	4214A-XBEE	4214A-XBEEPRO	1846A-XBEEXSC
ETSI (Europe)	Yes	Yes* Max TX 10 mW	No
C-TICK Australia	Yes	Yes	No
Telec (Japan)	Yes	Yes*	No

* XBee-PRO 802.15.4 TX Power restricted to 10 mW in Europe and Japan.

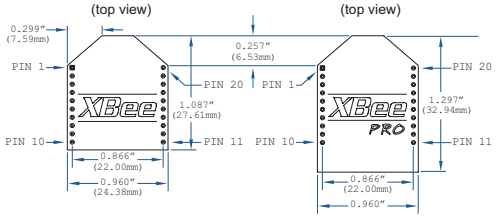
802.15.4 – Star



(side views)



(top view) (top view)



Please visit www.digi.com for part numbers.

DIGI SERVICE AND SUPPORT - You can purchase with confidence knowing that Digi is here to support you with expert technical support and a one-year warranty. www.digi.com/support

WHEN RELIABILITY MATTERS™

Digi International
11001 Bren Road E.
Minnetonka, MN 55343
U.S.A.
PH: 877-912-3444
952-912-3444
FX: 952-912-4952
email: info@digi.com

Digi International France
31 rue des Poissonniers
92200 Neuilly sur Seine
PH: +33-1-55-61-98-98
FX: +33-1-55-61-98-99
www.digi.fr

Digi International KK
NES Building South 8F
22-14 Sakuragaoka-cho,
Shibuya-ku
Tokyo 150-0031, Japan
PH: +81-3-5428-0261
FX: +81-3-5428-0262
www.digi-intl.co.jp

Digi International (HK) Limited
Suite 1703-05, 17/F.,
K Wah Centre
191 Java Road
North Point, Hong Kong
PH: +852-2833-1008
FX: +852-2572-9989
www.digi.cn

Digi International, the leader in device networking for business, develops reliable products and technologies to connect and securely manage local or remote electronic devices over the network or via the web. With over 20 million ports shipped worldwide since 1985, Digi offers the highest levels of performance, flexibility and quality.

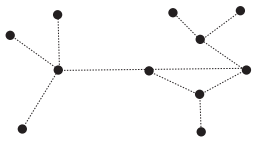
www.digi.com

91001412
B1/308

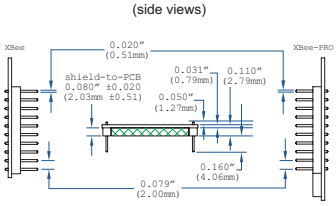
10. XBee Serie 2 ZB Características

Platform	XBee® ZB	XBee-PRO® ZB
Performance		
RF Data Rate	250 Kbps	250 Kbps
Indoor/Urban Range	133 ft (40 m)	300 ft (90 m)
Outdoor/RF Line-of-Sight Range	400 ft (120 m)	1 mi (1.6 km)
Transmit Power	1.25 mW (+1 dBm) / 2 mW (+3 dBm) boost mode	50 mW (+17 dBm) / Int'l 10 mW (+10 dBm)
Receiver Sensitivity (1% PER)	-96 dBm in boost mode	-102 dBm
Features		
Serial Data Interface	3.3V CMOS UART	3.3V CMOS UART
Configuration Method	API or AT commands, local or over-the-air	API or AT commands, local or over-the-air
Frequency Band	2.4 GHz	2.4 GHz
Interference Immunity	DSSS (Direct Sequence Spread Spectrum)	DSSS (Direct Sequence Spread Spectrum)
Serial Data Rate	1200 bps - 1 Mbps	1200 bps - 1 Mbps
ADC Inputs	(4) 10-bit ADC inputs	(4) 10-bit ADC inputs
Digital I/O	10	10
Antenna Options	Chip, Wire Whip, U.FL, RPSMA	Chip, Wire Whip, U.FL, RPSMA
Networking & Security		
Encryption	128-bit AES	128-bit AES
Reliable Packet Delivery	Retries/Acknowledgments	Retries/Acknowledgments
IDs and Channels	PAN ID, 64-bit IEEE MAC, 16 channels	PAN ID, 64-bit IEEE MAC, 13 channels
Power Requirements		
Supply Voltage	2.1 - 3.6VDC	3.0 - 3.4VDC
Transmit Current	35 mA / 45 mA boost mode @ 3.3VDC	295 mA @ 3.3VDC
Receive Current	38 mA / 40 mA boost mode @ 3.3VDC	45 mA @ 3.3VDC
Power-Down Current	<1 uA @ 25° C	<10 uA @ 25° C
Regulatory Approvals		
FCC (USA)	Yes	Yes
IC (Canada)	Yes	Yes
ETSI (Europe)	Yes	Yes (int'l unit only)
C-TICK (Australia)	Yes	Pending
Telec (Japan)	Yes	Pending (int'l unit only)

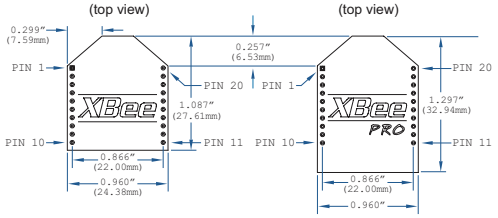
ZigBee – Mesh



(side views)



(top view) (top view)



Please visit www.digi.com for part numbers.

DIGI SERVICE AND SUPPORT - You can purchase with confidence knowing that Digi is here to support you with expert technical support and a one-year warranty. www.digi.com/support

WHEN RELIABILITY MATTERS™

Digi International
11001 Bren Road E.
Minnetonka, MN 55343
U.S.A.
PH: 877-912-3444
952-912-3444
FX: 952-912-4952
email: info@digi.com

Digi International France
31 rue des Poissonniers
92200 Neuilly sur Seine
PH: +33-1-55-61-98-98
FX: +33-1-55-61-98-99
www.digi.fr

Digi International KK
NES Building South 8F
22-14 Sakuragaoka-cho,
Shibuya-ku
Tokyo 150-0031, Japan
PH: +81-3-5428-0261
FX: +81-3-5428-0262
www.digi-intl.co.jp

Digi International (HK) Limited
Suite 1703-05, 17/F.,
K Wah Centre
191 Java Road
North Point, Hong Kong
PH: +852-2833-1008
FX: +852-2572-9989
www.digi.cn

Digi International, the leader in device networking for business, develops reliable products and technologies to connect and securely manage local or remote electronic devices over the network or via the web. With over 20 million ports shipped worldwide since 1985, Digi offers the highest levels of performance, flexibility and quality.

www.digi.com

91001471
A1/608

11. Lista de comandos AT

Addressing commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
DH	**Destination Address High**.Set/Get the upper 32 bits of the 64-bit destination address. When combined with DL, it defines the 64-bit destination address for data transmission. Special definitions for DH and DL include 0x000000000000FFFF (broadcast) and 0x0000000000000000 (coordinator).	CRE	0 - 0xFFFFFFFF	0
DL	**Destination Address Low**. Set/Get the lower 32 bits of the 64-bit destination address. When combined with DH, it defines the 64-bit destination address for data transmissions. Special definitions for DH and DL include 0x000000000000FFFF (broadcast) and 0x0000000000000000 (coordinator).	CRE	0 - 0xFFFFFFFF	0xFFFF (Coordinator) 0 (Router/End Device)
MY	**16-bit Network Address**. Read the 16-bit network address of the module. A value of 0xFFFE means the module has not joined a ZigBee network.	CRE	0 - 0xFFFE [read-only]	0xFFFE
MP	1**6-bit Parent Network Address**. Read the 16-bit network address of the module's parent. A value of 0xFFFE means the module does not have a parent.	E	0 - 0xFFFE [read-only]	0xFFFE
NC	**Number of Remaining Children**. Read the number of end device children that can join the device. If NC returns 0, then the device cannot allow any more end device children to join.	CR	0 - MAX_CHILDREN (maximum varies)	read-only
SH	**Serial Number High**. Read the high 32 bits of the module's unique 64-bit address.	CRE	0 - 0xFFFFFFFF [read-only]	factory-set
SL	**Serial Number Low**. Read the low 32 bits of the module's unique 64-bit address.	CRE	0 - 0xFFFFFFFF [read-only]	factory-set
NI	**Node Identifier.** Set/read a string identifier. The register only accepts printable ASCII data. In AT Command Mode, a string cannot start with a space. A carriage return ends the command. A command will automatically end when maximum bytes for the string have been entered. This string is returned as part of the ND (Node Discover) command. This identifier is also used with the DN (Destination Node) command. In AT command mode, an ASCII comma (0x2C) cannot be used in the NI string.	CRE	20-Byte printable ASCII string	ASCII space character (0x20)
SE	**Source Endpoint**. Set/read the ZigBee application layer source endpoint value. This value will be used as the source endpoint for all data transmissions. SE is only supported in AT firmware.The default value 0xE8 (Data endpoint) is the Digi data endpoint.	CRE	0 - 0xFF	0xE8

Security commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
EE	**Encryption Enable**. Set/Read the encryption enable setting.	CRE	0 - Encryption disabled 1 - Encryption enabled	0
EO	**Encryption Options.** Configure options for encryption. Unused option bits should be set to 0. Options include: 0x01 - Send the security key unsecured over-the-air during joins 0x02 - Use trust center (coordinator only	CRE	0 - 0xFF	
NK	**Network Encryption Key**. Set the 128-bit AES network encryption key. This command is write-only; NK cannot be read. If set to 0 (default), the module will select a random network key.	C	128-bit value	0
KY	**Link Key**. Set the 128-bit AES link key. This command is write only; KY cannot be read. Setting KY to 0 will cause the coordinator to transmit the network key in the clear to joining devices, and will cause joining devices to acquire the network key in the clear when joining.	CRE	128-bit value	0

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
D7	**DIO7 Configuration**. Select/Read options for the DIO7 line of the RF module.	CRE	0 = Disabled 1 = CTS Flow Control 3 = Digital input 4 = Digital output, low 5 = Digital output, high 6 = RS-485 transmit enable (low enable) 7 = RS-485 transmit enable (high enable)	1
D6	**DIO6 Configuration**. Configure options for the DIO6 line of the RF module.	CRE	0 = Disabled 1 = RTS flow control 3 = Digital input 4 = Digital output, low 5 = Digital output, high	0

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

I/O commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
IR	**IO Sample Rate**. Set/Read the IO sample rate to enable periodic sampling. For periodic sampling to be enabled, IR must be set to a non-zero value, and at least one module pin must have analog or digital IO functionality enabled (see D0-D8, P0-P2 commands). The sample rate is measured in milliseconds.	CRE	0, 0x32:0xFFFF (ms)	0
IC	I**O Digital Change Detection**. Set/Read the digital IO pins to monitor for changes in the IO state. IC works with the individual pin configuration commands (D0-D8, P0-P2). If a pin is enabled as a digital input/output, the IC command can be used to force an immediate IO sample transmission when the DIO state changes. IC is a bitmask that can be used to enable or disable edge detection on individual channels. Unused bits should be set to 0. Bit (IO pin): 0 (DIO0 4 (DIO4) 8 (DIO8) 1 (DIO1) 5 (DIO5) 9 (DIO9) 2 (DIO2) 6 (DIO6) 10 (DIO10) 3 (DIO3) 7 (DIO7) 11 (DIO11)	CRE	: 0 - 0xFFFF	0

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
D7	**DIO7 Configuration**. Select/Read options for the DIO7 line of the RF module.	CRE	0 = Disabled 1 = CTS Flow Control 3 = Digital input 4 = Digital output, low 5 = Digital output, high 6 = RS-485 transmit enable (low enable) 7 = RS-485 transmit enable (high enable)	1
D6	**DIO6 Configuration**. Configure options for the DIO6 line of the RF module.	CRE	0 = Disabled 1 = RTS flow control 3 = Digital input 4 = Digital output, low 5 = Digital output, high	0

1. Node types that support the command: C = Coordinator, R = Router, E = End Device

I/O commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
IR	**IO Sample Rate**. Set/Read the IO sample rate to enable periodic sampling. For periodic sampling to be enabled, IR must be set to a non-zero value, and at least one module pin must have analog or digital IO functionality enabled (see D0-D8, P0-P2 commands). The sample rate is measured in milliseconds.	CRE	0, 0x32:0xFFFF (ms)	0
IC	I**O Digital Change Detection**. Set/Read the digital IO pins to monitor for changes in the IO state. IC works with the individual pin configuration commands (D0-D8, P0-P2). If a pin is enabled as a digital input/output, the IC command can be used to force an immediate IO sample transmission when the DIO state changes. IC is a bitmask that can be used to enable or disable edge detection on individual channels. Unused bits should be set to 0. Bit (IO pin): 0 (DIO0 4 (DIO4) 8 (DIO8) 1 (DIO1) 5 (DIO5) 9 (DIO9) 2 (DIO2) 6 (DIO6) 10 (DIO10) 3 (DIO3) 7 (DIO7) 11 (DIO11)	CRE	: 0 - 0xFFFF	0

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
P0	**PWM0 Configuration**. Select/Read function for PWM0.	CRE	0 = Disabled 1 = RSSI PWM 3 - Digital input, monitored 4 - Digital output, default low 5 - Digital output, default high	1
P1	**DIO11 Configuration**. Configure options for the DIO11 line of the RF module.	CRE	0 - Unmonitored digital input 3- Digital input, monitored 4- Digital output, default low 5- Digital output, default high	0
P2	**DIO12 Configuration**. Configure options for the DIO12 line of the RF module.	CRE	0 - Unmonitored digital input 3- Digital input, monitored 4- Digital output, default low 5- Digital output, default high	0
P3	**DIO13 Configuration**. Set/Read function for DIO13. This command is not yet supported.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	
D0	**AD0/DIO0 Configuration**. Select/Read function for AD0/DIO0.	CRE	1 - Commissioning button enabled 2 - Analog input, single ended 3 - Digital input 4 - Digital output, low 5 - Digital output, high	1
D1	**AD1/DIO1 Configuration**. Select/Read function for AD1/DIO1.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
D2	**AD2/DIO2 Configuration**. Select/Read function for AD2/ DIO2.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D3	**AD3/DIO3 Configuration**. Select/Read function for AD3/ DIO3.	CRE	0, 2-5 0 – Disabled 2 - Analog input, single ended 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D4	**DIO4 Configuration**. Select/Read function for DIO4.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	0
D5	**DIO5 Configuration**. Configure options for the DIO5 line of the RF module.	CRE	0 = Disabled 1 = Associated indication LED 3 = Digital input 4 = Digital output, default low 5 = Digital output, default high	1
D8	**DIO8 Configuration**. Set/Read function for DIO8. This command is not yet supported.	CRE	0, 3-5 0 – Disabled 3 – Digital input 4 – Digital output, low 5 – Digital output, high	
LT	**Assoc LED Blink Time**. Set/Read the Associate LED blink time. If the Associate LED functionality is enabled (D5 command), this value determines the on and off blink times for the LED when the module has joined a network. If LT=0, the default blink rate will be used (500ms coordinator, 250ms router/end device). For all other LT values, LT is measured in 10ms.	CRE	0, 0x0A - 0xFF (100 - 2550 ms)	0

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
CC	**Command Sequence Character**. Set/Read the ASCII character value to be used between Guard Times of the AT Command Mode Sequence (GT + CC + GT). The AT Command Mode Sequence enters the RF module into AT Command Mode. The CC command is only supported when using AT firmware: 20xx (AT coordinator), 22xx (AT router), 28xx (AT end device).	CRE	0 - 0xFF	0x2B ('+' ASCII)
1. Node types that support the command: C = Coordinator, R = Router, E = End Device				

Sleep commands

AT Command	Name and Description	Node Type[1]	Parameter Range	Default
SM	**Sleep Mode** Sets the sleep mode on the RF module. An XBee loaded with router firmware can be configured as either a router (SM set to 0) or an end device (SM > 0). Changing a device from a router to an end device (or vice versa) forces the device to leave the network and attempt to join as the new device type when changes are applied.	RE	0-Sleep disabled (router) 1-Pin sleep enabled 4-Cyclic sleep enabled 5 - Cyclic sleep, pin wake	0 - Router 4 - End Device
SN	**Number of Sleep Periods.** Sets the number of sleep periods to not assert the On/Sleep pin on wakeup if no RF data is waiting for the end device. This command allows a host application to sleep for an extended time if no RF data is present.	CRE	1 - 0xFFFF	1
SP	**Sleep Period.** This value determines how long the end device will sleep at a time, up to 28 seconds. (The sleep time can effectively be extended past 28 seconds using the SN command.) On the parent, this value determines how long the parent will buffer a message for the sleeping end device. It should be set at least equal to the longest SP time of any child end device.	CRE	0x20 - 0xAF0 x 10ms (Quarter second resolution)	0x20
ST	**Time Before Sleep** Sets the time before sleep timer on an end device. The timer is reset each time serial or RF data is received. Once the timer expires, an end device may enter low power operation. Applicable for cyclic sleep end devices only.	E	1 - 0xFFFE (x 1ms)	0x1388 (5 seconds)
SO Command	**Sleep Options**. Configure options for sleep. Unused option bits should be set to 0. Sleep options include: 0x02 - Always wake for ST time 0x04 - Sleep entire SN * SP time 0x06 - Enabled extended sleep and wake for ST time Sleep options should not be used for most applications. See Network commissioning and diagnostics on page 82 for more information.	E	0 - 0xFF	0

12. Programación

12.1 Sensor

12.1.1 Archivo hpp

```
#ifndef Sensor_
#define Sensor_

#include "Arduino.h"

class Sensor {

protected:

  uint8_t _Pin;



public:
  Sensor ();

  void Add( uint8_t Pin );

};

#endif
```

12.1.2 Archivo Cpp

```
#include "Sensor.h"
```

```
Sensor::Sensor(){

}

void Sensor::Add( uint8_t Pin ){

 _Pin = Pin;

 pinMode( _Pin, INPUT );

}
```

12.2 Clase LDR

12.2.1 Archivo hpp

```
#ifndef LDR_h
#define LDR_h

#include "Arduino.h"
#include "Sensor.h"

class LDR : public Sensor {

private:

 //Media de los datos del sensor y contador para saber el Nº de muestras en
 // cada ciclo

 //Se han modificado el tipo de variables para ahorrar almacenamiento
 //utilizando las variables adecuadas para cada una de ellas.
```

```
 uint16_t _Media;
 uint8_t _Muestras; //Cuenta las muestras que se han obtenido
 double _Suma;



public:

 LDR();

 //Metodo para informar si hay Oscuridad
 //Realizará la media del numero de muestras que se le indique al Metodo
 // y si el umbral está por debajo de la media devolverá un 1, informando
 //que la habitaciín esta oscura.
 bool Oscuridad( double Um);

 uint16_t GetMedia();

};

#endif
```

12.2.2 Archivo cpp

```
#include "LDR.h"

#define Muestras 50

LDR::LDR(){

 //Inicialización de los atributos a 0
 _Media = 0;
 _Muestras = 0;
```

```
  _Suma = 0;

}

bool LDR::Oscuridad(double Um){

uint16_t Lectura = analogRead( _Pin );

_Suma = _Suma + Lectura;

_Muestras++;

if ( _Muestras >= Muestras ){

   _Media = _Suma / Muestras;

   _Muestras = 0;

   _Suma = 0;

  }

if ( _Media < Um ){

   return 1;

  } else{

   return 0;

   }
```

```
}

uint16_t LDR::GetMedia(){

 return _Media;

}
```

12.3 Clase SenMovi

12.3.1 Archivo hpp

```
#ifndef SenMovi_h
#define SenMovi_h

#include "Arduino.h"
#include "Sensor.h"

class SenMovi : public Sensor {
private:

public:
 //Constructor
 SenMovi();

 //Metodo de Salida: Sí el sensor detecta movmiento su salida se encontrará
 // a 1 durante un periodo de tiempo (Propiedad del sensor) y cuenda no detecte
 //movimiento o despueés de haber transcurrido este periodo de tiempo su Salida
```

```
 //será 0
 bool Salida();

};

#endif
```

12.3.2 Archuivo cpp

```
#include "SenMovi.h"

SenMovi::SenMovi(){
}

bool SenMovi::Salida(){

 bool Lectura;

 Lectura = digitalRead( _Pin );

 if (Lectura == 1) {

  return 1;

 } else {

    return 0;

  }

}
```

12.3 Clase Temperatura

12.3.1 Archivo hpp

```
#ifndef Temperatura_
#define Temperatura_

#include "Arduino.h"
#include "Sensor.h"

class Temperatura : public Sensor {

private:

public:

  Temperatura ();

  double Valor();

};

#endif
```

12.3.2 Archuivo cpp

```
#include "Temperatura.h"



Temperatura::Temperatura(){

}
```

```
double Temperatura::Valor(){

 return ( ( analogRead(_Pin) / 1023 ) * 5000 ) ; //Sensibilidad 10 mV

}
```

12.4 Clase Reloj

12.4.1 Archivo hpp

```
#ifndef Reloj_
#define Reloj_

#include "Arduino.h"
#include "Wire.h"
#include "RTClib.h"

class Reloj {
private:

 //Variable para activar el Temporizador
 bool _TempActivo;

 //Variables Para controlar el Tiempo
 double _TiempoFinal;

 //Metodo Horario
 RTC_DS1307 RTC;
 //RTC_DS3231 RTC;


public:
```

```
 Reloj ();

 bool Temporizador(double tiempo);

 void TempReset();

 bool GetTempActivo();

 //Devuelve un 1 si la Hora y Minutos pasados a la funcion es mayor
 //a las horas y minutos actuales, sino un 0
 bool Horario( double Hora, double Minuto );



};

#endif
```

12.4.2 Archivo cpp

```
#ifndef Reloj_
#define Reloj_

#include "Arduino.h"
#include "Wire.h"
#include "RTClib.h"

class Reloj {
private:

 //Variable para activar el Temporizador
 bool _TempActivo;
```

```
 //Variables Para controlar el Tiempo
 double _TiempoFinal;

 //Metodo Horario
 RTC_DS1307 RTC;
 //RTC_DS3231 RTC;


public:

 Reloj ();

 bool Temporizador(double tiempo);

 void TempReset();

 bool GetTempActivo();

 //Devuelve un 1 si la Hora y Minutos pasados a la funcion es mayor
 //a las horas y minutos actuales, sino un 0
 bool Horario( double Hora, double Minuto );


};

#endif
```

12.5 Clase Display

12.5.1 Archivo hpp

```
#ifndef Display_
#define Display_
```

```
#include "Arduino.h"

class Display {
private:

 uint8_t _Pines[4]; //Almacena los pines del Display


public:

 Display ();

 void Add( uint8_t A, uint8_t B, uint8_t C = 0, uint8_t D = 0 );

 void MostrarNum( uint8_t Num );

};

#endif
```

12.5.2 Archivo cpp

```
#include "Display.h"

Display::Display(){

}

void Display::Add( uint8_t A, uint8_t B, uint8_t C = 0, uint8_t D = 0 ){

 _Pines[3] = A;
 _Pines[2] = B;
```

```
 _Pines[1] = C;
 _Pines[0] = D;

 for (size_t i = 0; i < sizeof( _Pines ) ; i++) {

   if ( _Pines[i] != 0 ) {

     pinMode( _Pines[i], OUTPUT );

   }

 }

}

void Display::MostrarNum( uint8_t Num ){

 uint8_t  Digitos[10][4] =

      {
        { 0,0,0,0 },   // 0
        { 0,0,0,1 },   // 1
        { 0,0,1,0 },   // 2
        { 0,0,1,1 },   // 3
        { 0,1,0,0 },   // 4
        { 0,1,0,1 },   // 5
        { 0,1,1,0 },   // 6
        { 0,1,1,1 },   // 7
        { 1,0,0,0 },   // 8
        { 1,0,0,1 }    // 9
      };
```

```
 //El siguiente for recorrera dos vectores:
 //el vectores de pines y el vector del número que se Indica a la funcion (Num)

 for (size_t i = 0; i < 4; i++) {

   uint8_t Valor = Digitos[Num][i];

   if ( _Pines[i] != 0 ) {

     digitalWrite( _Pines[i], Valor );

   }

 }

}
```

12.6 Clase Dirección

12.6.1 Archivo hpp

```
#ifndef Direccion_
#define Direccion_

#include "XBee.h"

class Direccion {
protected:

 XBeeAddress64 _address;
```

```
public:

 Direccion ();

 void Add( uint32_t addressMSB, uint32_t addressLSB );
 void Add( uint64_t address );

 XBeeAddress64 getAddress64();



};

#endif
```

12.6.2 Archivo cpp

```
#include "Direccion.h"

Direccion::Direccion(){}

void Direccion::Add( uint32_t addressMSB, uint32_t addressLSB ){

 _address.setMsb( addressMSB);
        _address.setLsb(addressLSB);

}
void Direccion::Add( uint64_t address ){

 _address.set( address );

}

XBeeAddress64 Direccion::getAddress64(){
```

```
 return _address;

}
```

12.7 Clase Actuador

12.7.1 Archivo hpp

```
#ifndef Actuador_h
#define Actuador_h

#include "Arduino.h"

#include "Direccion.h"

class Actuador : public Direccion {
private:

 //Atributo que indica el estado del actuador
 bool _Estado;

 //Atributo que indica si se puede enviar el estado del actuador
 bool _Enviar;

 uint8_t _cmd[2];

public:

 Actuador();

 void setCmd( uint8_t cmd[2] );
```

```
 //Este estado activara o desactivara el actuador dependiendo de "Estado"
 // Sí es un 0 el actuador estara desactivado y si es un 1 lo activa
 void Act();
 void DesAct();

 void CambiarEstado();

 void Enviar();
 void NoEnviar();

 uint8_t GetCmd( bool iterador );

 bool GetEnviar();

 //Devuelve el estado en el que se encuentra el actuador
 bool GetEstado();

};

#endif
```

12.7.2 Archivo cpp

```
#include "Actuador.h"

Actuador::Actuador(){

 _Estado = 0;
 _Enviar = 0;

}
```

```
void Actuador::setCmd( uint8_t cmd[2] ){

 _cmd[0] = cmd[0];
 _cmd[1] = cmd[1];



}

void Actuador::Act(){

 _Estado = 1;

}

void Actuador::DesAct(){

 _Estado = 0;

}

void Actuador::CambiarEstado(){

 if ( _Estado == 0 ) {

   _Estado = 1;

 } else {

   _Estado = 0;

 }
```

```
}

void Actuador::Enviar() {

 _Enviar = 1;

}

void Actuador::NoEnviar() {

 _Enviar = 0;

}

uint8_t Actuador::GetCmd( bool iterador ){

 return _cmd[iterador];

}

bool Actuador::GetEnviar() {

 return _Enviar;

}


bool Actuador::GetEstado(){

 return _Estado;
```

```
}
```

12.8 Clase Persiana

12.8.1 Archivo hpp

```
#ifndef Persiana_h
#define Persiana_h

#include "Arduino.h"
#include "Reloj.h"

#include "Actuador.h"

#include "Direccion.h"

class Persiana : public Direccion {
private:

  //Actuadores para bajar y subir la Persiana
  Actuador _PerS;
  Actuador _PerB;

public:

  Persiana ();

  //Los dos métodos devuelven los actuaoderes
  Actuador& getASubir();
  Actuador& getABajar();
```

```
};

#endif
```

12.8.2 Archivo cpp

```
#include "Persiana.h"

Persiana::Persiana(){

}

Actuador& Persiana::getASubir(){

  return _PerS;

}

Actuador& Persiana::getABajar(){

  return _PerB;

}
```

12.9 Clase Climatización

12.9.1 Archivo hpp

```
#ifndef Climatizacion_
#define Climatizacion_

#include "Arduino.h"

#include "Actuador.h"
```

```
#include "Direccion.h"

class Climatizacion : public Direccion {
private:
  //Actuadores para bajar y subir la Persiana
  Actuador _AFrio;
  Actuador _ACalor;



public:

  Climatizacion ();

  //Los dos métodos devuelven los actuaoderes
  Actuador& getAFrio();
  Actuador& getACalor();

};

#endif
```

12.9.2 Archivo cpp

```
#ifndef Climatizacion_
#define Climatizacion_

#include "Arduino.h"

#include "Actuador.h"

#include "Direccion.h"
```

```
class Climatizacion : public Direccion {
private:
  //Actuadores para bajar y subir la Persiana
  Actuador _AFrio;
  Actuador _ACalor;


public:

  Climatizacion ();

  //Los dos métodos devuelven los actuaoderes
  Actuador& getAFrio();
  Actuador& getACalor();

};

#endif
```

12.10 Clase Habitación

12.10.1 Archivo hpp

```
#ifndef Habitacion_h
#define Habitacion_h

//XBee
#include "XBee.h"

#include "SoftwareSerial.h"

#include "ArduVector.h"
```

```
#include "Actuador.h"

#include "Reloj.h"

#include "SenMovi.h"

#include "LDR.h"

#include "Temperatura.h"

#include "Climatizacion.h"

#include "Persiana.h"

#include "String.h"

#include "Display.h"

class Habitacion {
private:

////////////////////////////////////Xbee/////////////////////////////////////////

XBee xbee;

//Objeto de la respuesta IOSample, utilizada para los módulos pulsadores
ZBRxIoSampleResponse ioSample = ZBRxIoSampleResponse();

//Vector con las direcciones de los módulos pulsadores
ArduVector<XBeeAddress64> _AddressModPulsador;
```

```
//////////////////////////Añadir Dispostitivos////////////////////////////////


//////////////////////////////Sistema de Luz/////////////////////////////////////

//Variables -> Necesarias para que los pulsadores se activen una vez sola.
bool _EstadoAnteriorLuz;
bool _EstadoAnteriorPerS;
bool _EstadoAnteriorPerB;

//Variable para conocer el modo de funcionamiento de la habitación
//ModoF
String _Modo;
Reloj _RModoF; //Reloj para modificar el modo de funcionamiento

//HMovimiento
ArduVector<XBeeAddress64> _AddressMovimiento;
SenMovi _M1;
SenMovi _M2;
bool _Movimiento;

//HOscuridad
LDR _L;
bool _Oscuridad;

//OnOFFLuz
bool _Encender;
Reloj _ROnOff;
bool Sample;
//Actuadores de la Habitacion
```

```
//Actuador _ALuz;
ArduVector<Actuador> _ALuz;



//////////////////////////Sistema de Temepratura//////////////////////////////

//SisTemperatura
Temperatura _T;
uint8_t _Consigna; //Valor al que tiene que estar la Habitacion

//El siguiente vector tendrá las direcciones de los actuadores de Temperatura
//de la Habitacion
ArduVector<Climatizacion> _Climatizacion;

//MostrarConsigna
Display D1;
Display D2;

char _Num1Anterior;
char _Num2Anterior;

////////////////////////////SisPersianas/////////////////////////////////////

//En el siguiente vector se almacenarán los objetos de persianas, añadiendo
//estos objetos con el metodo AddPersiana
ArduVector<Persiana> _Persianas;

Reloj _RPer;

//_EstadoPer si esta a 0 será que la persina esta cerrada y si es 1 la
```

```
//persina estará abierta
bool _EstadoAnteriroHoraN;
bool _EstadoAnteriroHoraM;

//////////////////////////Ahorro de energia////////////////////////////////

//DetecPersonas
Reloj _RPersonas;
bool _Personas;

//ModoSleep
bool _EstadoAnteriorPersonas;

public:

 Habitacion();

 void AddXBee( XBee Xbee);

//////////////////////////Recibir datos////////////////////////////////////

 void RecibirDatos();


//////////////////////////Añadir Dispostitivos////////////////////////////////

 //Añadimos Direcciones de los modulos pulsadores
 void AddMPulsador( XBeeAddress64 address );

 void AddLuz( XBeeAddress64 address, uint8_t cmd[2]);
```

```
//Añadimos direcciones de los módulos de movmiento
void AddMovmiento( XBeeAddress64 address );

void AddPersiana( XBeeAddress64 address, uint8_t cmdSubir[2], uint8_t cmdBajar[2] );

void AddClimatizacion( XBeeAddress64 address, uint8_t cmdFrio[2], uint8_t cmdCalor[2] );



//////////////////////Sistema de Luz/////////////////////////////////////

//Indica si hay movmiento en la habitación
void HMovimiento();

//Indica si hay movmiento en la habitación
void HOscuridad();

//Modo de funcionamiento
void ModoF();

void OnOFFLuz();

//////////////////////Sistema de Temperatura//////////////////////////////

void SisTemperatura();

void MostrarConsigna();

//////////////////////////Sistema de perianas//////////////////////////////

void SisPersianas();
```

```
 ////////////////////////////Ahorro de Energia////////////////////////////////

 void DetecPersonas();

 void ModoSleep();




};

#endif
```

12.10.2 Archivo Cpp

```
#include "Habitacion.h"

//Añadir en el sistema de temperatura para mostrar la Consigna

//Condicion de el sistema de persianas se activan todas las persianas y se
//desactivan todas, se podria modifica para que un modulo actuador afecte
//solo a una persianas.

//metodo para añadir el actuador de Luz

//Cuando se envia una trama AT commmand eliminar los array de cmd y introducir
//el valor del comando directamente en cada trama

//Cambiar pin del sensor de movmiento ahora esta en el pin
8!!!!!!!!!/////////////////////////////////////////////////

//Horario Nocturno
```

```
#define HoraN 23
#define MinutoN 55

//Horario por la mañana
#define HoraM 7
#define MinutoM 30



bool EnviarTramaAT( XBee xbee, XBeeAddress64 address, uint8_t cmd[2],
                uint8_t Value[1] ){

 RemoteAtCommandRequest remoteAtRequest;
 RemoteAtCommandResponse remoteAtResponse = RemoteAtCommandResponse();

 uint8_t value[1];

 value[0] = Value[0];

 remoteAtRequest = RemoteAtCommandRequest( address, cmd, value, sizeof(value));

 xbee.send(remoteAtRequest);

 if (xbee.readPacket(5000)) {

  if (xbee.getResponse().getApiId() == REMOTE_AT_COMMAND_RESPONSE) {

     xbee.getResponse().getRemoteAtCommandResponse(remoteAtResponse);


     if (remoteAtResponse.isOk()) {
```

```
        remoteAtRequest.clearCommandValue();

        return 1;

      } else {
        remoteAtRequest.clearCommandValue();

        return 0;

        }

    }

  }

}

bool EnviarTramaTx( XBee xbee, XBeeAddress64 address, uint8_t payload[3] ){

 ZBTxRequest zbTx = ZBTxRequest( address , payload, sizeof(payload));
 ZBTxStatusResponse txStatus = ZBTxStatusResponse();

 xbee.send(zbTx);

 if (xbee.readPacket(500)) {

  if (xbee.getResponse().getApiId() == ZB_TX_STATUS_RESPONSE) {

    xbee.getResponse().getZBTxStatusResponse(txStatus);

    if (txStatus.getDeliveryStatus() == SUCCESS) {
```

```
      Serial.println("Success!!");

      return 1;

    } else {

      return 0;

    }

  }

 }

}

Habitacion::Habitacion() {

 //Direccion de modulos pulsadores
 _AddressModPulsador.reserve(3);

 //Variables
 _EstadoAnteriorLuz = 0;
 _EstadoAnteriorPerS= 0;
 _EstadoAnteriorPerB = 0;

 //Modo de funcionamiento
 _Modo = "Automatico";

 //HMovimiento
```

```
_M1.Add(7);
_M2.Add(9);
_Movimiento = 0;
_AddressMovimiento.reserve(3);

//HOscuridad
_L.Add(A0);

//OnOFFLuz
_Encender = 0;
Sample = 1;

//Direcciones del Actuador de Luz
//_ALuz.Add( 0x0013A200, 0x4154DCFC);
_ALuz.reserve(3);

//Temperatura
_T.Add(A1);
_Consigna = 0;
_Climatizacion.reserve(2);

//MostrarConsigna
D1.Add( 2, 3 ); //( A, B )
D2.Add( 4, 5, 6, 7 ); //( A, B, C, D )



//Persianas
_Persianas.reserve(3);

//DetecPersonas
_Personas = 0;
```

```
 _EstadoAnteriroHoraN = 0;
 _EstadoAnteriroHoraM = 0;


 //ModoSleep
 _EstadoAnteriorPersonas = 0;

}

void Habitacion::AddXBee( XBee Xbee ){

   xbee = Xbee;

}

/////////Método para recibir los datos de los módulos pulsadores///////////////

void Habitacion::RecibirDatos(){

 if (Sample == 1) {

   xbee.readPacket();

   if ( xbee.getResponse().isAvailable() ) {

     if ( xbee.getResponse().getApiId() == ZB_IO_SAMPLE_RESPONSE ) {

       xbee.getResponse().getZBRxIoSampleResponse( ioSample );

       }
```

```
    }

   }

 }

}

////////////////////Métodos para añadir Dispostitivos//////////////////////

//Añadir Módulo pulsadores
void Habitacion::AddMPulsador( XBeeAddress64 address ){

 _AddressModPulsador.push_back( address );

}

void Habitacion::AddLuz( XBeeAddress64 address, uint8_t cmd[2] ){

 Actuador A;
 A.Add( address.getMsb(), address.getLsb() );
 A.setCmd( cmd );
 _ALuz.push_back( A );

}


//Añadir módulos de movmiento
void Habitacion::AddMovmiento( XBeeAddress64 address ){
```

```
  _AddressMovimiento.push_back( address );

}

//Método para añadir persianas a la Habitacion
void Habitacion::AddPersiana( XBeeAddress64 address, uint8_t cmdSubir[2],
                        uint8_t cmdBajar[2]){

  Persiana Per;
  Per.Add( address.getMsb(), address.getLsb() );
  Per.getASubir().setCmd( cmdSubir );
  Per.getABajar().setCmd( cmdBajar );



  _Persianas.push_back( Per );

}

//Añadir módulos de Temperatura
void Habitacion::AddClimatizacion( XBeeAddress64 address, uint8_t cmdFrio[2], uint8_t
cmdCalor[2] ){

  Climatizacion C;
  C.Add( address.getMsb() , address.getLsb() );
  C.getAFrio().setCmd( cmdFrio );
  C.getACalor().setCmd( cmdCalor )

  _Climatizacion.push_back( C );

}
```

```
//////////////////////////Sistema de Luz//////////////////////////////////////
void Habitacion::HMovimiento(){

 //Módulo remoto y modulo local

 //En el caso de que exista algun módulo de movimiento remoto
 //influirá en la variable _Movimiento de la habitación
 //En el caso contrario solo afectarán los sensores de movimiento
 //del módulo Habitacion
 if ( _AddressMovimiento.size() > 0 ) {

  for (int i = 0; i < _AddressMovimiento.size(); i++) {
  //Sensor de movmiento local
   if ( _M1.Salida() == 1 || _M2.Salida() == 1 || ( ( ioSample.getRemoteAddress64().getMsb()
      == _AddressMovimiento[i].getMsb() && ioSample.getRemoteAddress64().getLsb()
      == _AddressMovimiento[i].getLsb() ) && ioSample.isDigitalOn(0) == 1 )
      ) {

    _Movimiento = 1;

   }

    else {

     _Movimiento = 0;

    }

  }
```

```
 } else {
   if ( _M1.Salida() == 1 || _M2.Salida() == 1 ) {
     _Movimiento = 1;
   } else {
     _Movimiento = 0;
   }
 }
}
void Habitacion::HOscuridad(){
   uint16_t UmbralLuz = 600; //Indica el umbral de luz de la Habitacion
   if ( _L.Oscuridad( UmbralLuz ) == 1 ) {
     _Oscuridad = 1;
   } else{
       _Oscuridad = 0;
     }
```

```
    Serial.print( " Oscuridad:  ");
    Serial.println( _Oscuridad );

}

void Habitacion::ModoF(){

 double TiempoTemp = 180; // Indica el tiempo de Temporizador
 bool Tiempo = 0;

 if ( _Modo == "Automatico" ) {

  for (int i = 0; i < _AddressModPulsador.size(); i++) {

   if ( ( ioSample.getRemoteAddress64().getMsb() ==
      _AddressModPulsador[i].getMsb() ) &&
      ( ioSample.getRemoteAddress64().getLsb() ==
      _AddressModPulsador[i].getLsb() )
      ) {

       if ( ioSample.isDigitalOn(0) == 1 ) {

         if ( _EstadoAnteriorLuz == 0 ) {

           _Modo = "Manual";

         }

       }

       _EstadoAnteriorLuz = ioSample.isDigitalOn(0);
```

```
        }

  }

  if ( _RModoF.Horario( HoraN, MinutoN ) == 1 )  {

    _Modo = "Nocturno";

  }

}

if ( _Modo == "Manual" ) { //-----------------------------------------

  Tiempo = _RModoF.Temporizador( TiempoTemp ); //1 Minuto

  if ( _Movimiento == 1 && _RModoF.GetTempActivo() == 1 ) {

    _RModoF.TempReset();

  }

  if ( Tiempo == 1 && _Movimiento == 0 ) {

    _Modo = "Automatico";
    _RModoF.TempReset();

  }

  if ( _RModoF.Horario( HoraN, MinutoN ) == 1 ) {
```

```
      _Modo = "Nocturno";

    }

 }

 if ( _Modo == "Nocturno" ) { //----------------------------------------

   if ( ( _RModoF.Horario( HoraM , MinutoM) == 1 ) && ( _RModoF.Horario( HoraN , MinutoN)
== 0 ) ) {

    _Modo = "Automatico";

   }

 }

 Serial.print( " Modo:  ");
 Serial.println( _Modo );
 Serial.println("");

}

void Habitacion::OnOFFLuz(){

 double TiempoTemp = 30; // Indica el tiempo de Temporizador
 bool TiempoLuz = 0;

 bool TramaEnviada;
```

```
//bool TramaEnviada;

uint8_t LuzCmd[2];
uint8_t LuzValue[1];



if ( _Modo == "Automatico" ) {

 if ( _Encender == 0 ) {

  if ( _Oscuridad == 1 ) {

   if ( _Movimiento == 1 ) {

    for (size_t i = 0; i < _ALuz.size(); i++) {
     LuzValue[0] = 0x5;
     _ALuz[i].Enviar();

    }

   }

  }

 }

 if ( _Encender == 1 ) {

  if ( _Movimiento == 0 ) {

   TiempoLuz = _ROnOff.Temporizador( TiempoTemp );
```

```
    } else {

      TiempoLuz = 0;
      _ROnOff.TempReset();

     }

    if ( TiempoLuz == 1 ) {

     //_ALuz.Enviar();
     for (size_t i = 0; i < _ALuz.size(); i++) {
      LuzValue[0] = 0x4;
      _ALuz[i].Enviar();

     }

    }

  }


}

if ( _Modo == "Manual"  || _Modo == "Nocturno") {

   for (int i = 0; i < _AddressModPulsador.size(); i++) {

     if ( ioSample.getRemoteAddress64().getMsb() ==
        _AddressModPulsador[i].getMsb()
        && ioSample.getRemoteAddress64().getLsb() ==
```

```
        _AddressModPulsador[i].getLsb() ) {



       if ( ioSample.isDigitalOn(0) == 1 ) {

        if (_EstadoAnteriorLuz == 0) {

        //_ALuz.Enviar();
          for (size_t i = 0; i < _ALuz.size(); i++) {

            if ( _ALuz[i].GetEstado() == 0 ) {

              LuzValue[0] = 0x5;

            }  else {

              LuzValue[0] = 0x4;

            }
            _ALuz[i].Enviar();

          }

        }

       }

      _EstadoAnteriorLuz = ioSample.isDigitalOn(0);

    }
```

```
    }

 }

 for (size_t i = 0; i < _ALuz.size(); i++) {

  if ( _ALuz[i].GetEnviar() == 1 ) {

    Sample = 0;

    LuzCmd[0] = _ALuz[i].GetCmd( 0 );
    LuzCmd[1] = _ALuz[i].GetCmd( 1 );

    TramaEnviada = EnviarTramaAT( xbee, _ALuz[i].getAddress64(), LuzCmd,
                          LuzValue );

    if ( TramaEnviada == 1 ) {

      Sample = 1;
      TramaEnviada = 0;
      //No enviamos mas tramas
      _ALuz[i].NoEnviar();

     //Resetear variables para el modo Automatico
     _ROnOff.TempReset();

     //Modificacamos la variable que indica si esta encendida o no
     la Habitacion
     if ( _Encender == 0 ) {

       _Encender = 1;
```

```
        _ALuz[i].Act();

      } else {

        _Encender = 0;
        _ALuz[i].DesAct();

      }

    }

  }

}

//////////////////////////Sistema de Temperatura//////////////////////////////

void Habitacion::SisTemperatura(){

  uint8_t TemperaturaCmd[2];
  uint8_t TemperaturaValue[1];

  bool TramaEnviada;

  uint8_t ToleranciaTemperatura = 1;

   //En el siguiente for se obtendrá la consigna del modulo pulsador
   //leida por la entrada 3 del xbee ( potenciometro )
   for (int i = 0; i < _AddressModPulsador.size(); i++) {
```

```
    if ( ( ioSample.getRemoteAddress64().getMsb() ==
        _AddressModPulsador[i].getMsb() ) &&
         ( ioSample.getRemoteAddress64().getLsb() ==
        _AddressModPulsador[i].getLsb() ) ) {

        //uint16_t EntradaAnalogica = ioSample.getAnalog(3);
        _Consigna = map( ioSample.getAnalog(3) , 0, 1023, 0, 30 );

        if ( ioSample.getAnalog(3) == 0 ) {

          _Consigna = 25;

        }

    }

  }

}

//En los siguientes tres if se especificarán las 3 posibilidades que pueden
//suceder

//En el momento que la temepratura este por encima de  _Consigna +
//ToleranciaTemperatura se activa el sistema de Climatizacion
if ( _T.Valor() > ( _Consigna + ToleranciaTemperatura ) ) {

  for (size_t i = 0; i < _Climatizacion.size(); i++) {

    if ( _Climatizacion[i].getAFrio().GetEstado() == 0 ) {
```

```
      TemperaturaValue[0] = 0x5;
      _Climatizacion[i].getAFrio().Enviar();

    }

  }

}

//Que la temperatura esta por debajo de la consigna + la Tolerancia, activa
//todos los sisetmas de climatizacón
if ( _T.Valor() < ( _Consigna - ToleranciaTemperatura ) ) {

  for (size_t i = 0; i < _Climatizacion.size(); i++) {

    if ( _Climatizacion[i].getACalor().GetEstado() == 0 ) {

      TemperaturaValue[0] = 0x5;
      _Climatizacion[i].getACalor().Enviar();

    }

  }

}

//Que la temepratura se encuente entre los valores +- de la consigna
//comprobando que esta activado el actuador concreto se enviara una trama para
//apagarlo
if ( ( _Consigna + ToleranciaTemperatura ) > _T.Valor() >
```

```
    ( _Consigna - ToleranciaTemperatura ) ) {

  for (size_t i = 0; i < _Climatizacion.size(); i++) {

    if ( _Climatizacion[i].getAFrio().GetEstado() == 1 ) {

      TemperaturaValue[0] = 0x4;
      _Climatizacion[i].getAFrio().Enviar();

    }

    if ( _Climatizacion[i].getACalor().GetEstado() == 1 ) {

      TemperaturaValue[0] = 0x4;
      _Climatizacion[i].getACalor().Enviar();

    }

  }

}

//En el siguiente for se enviará a todos los actuadores de Temepratura de la
//Habitacion, activarse o desactivarse, todos estarán activados o desactivados
for (size_t i = 0; i < _Climatizacion.size(); i++) {

  if ( _Climatizacion[i].getAFrio().GetEnviar() == 1 ) {

    Sample = 0;

    Serial.println(" Enviando Temperatura..... ") ;
```

```
    direccion = _Climatizacion[i].getAddress64();

    TemperaturaCmd[0] = _Climatizacion[i].getAFrio().GetCmd( 0 );
    TemperaturaCmd[1] = _Climatizacion[i].getAFrio().GetCmd( 1 );

    TramaEnviada = EnviarTramaAT( xbee, direccion, TemperaturaCmd,
                          TemperaturaValue );

    if ( TramaEnviada == 1 ) {

     _Climatizacion[i].getAFrio().NoEnviar();

      if ( _Climatizacion[i].getAFrio().GetEstado() == 0 ) {

        _Climatizacion[i].getAFrio().Act();

      } else {

        _Climatizacion[i].getAFrio().DesAct();

      }

    }

  }

  if ( _Climatizacion[i].getACalor().GetEnviar() == 1 ) {

    Sample = 0;
```

```
    Serial.println(" Enviando Temperatura..... ") ;

    direccion = _Climatizacion[i].getAddress64();

    TemperaturaCmd[0] = _Climatizacion[i].getACalor().GetCmd( 0 );
    TemperaturaCmd[1] = _Climatizacion[i].getACalor().GetCmd( 1 );

    TramaEnviada = EnviarTramaAT( xbee, direccion, TemperaturaCmd, TemperaturaValue );

    if ( TramaEnviada == 1 ) {

     _Climatizacion[i].getACalor().NoEnviar();

     if ( _Climatizacion[i].getACalor().GetEstado() == 0 ) {

       _Climatizacion[i].getACalor().Act();

     } else {

       _Climatizacion[i].getACalor().DesAct();

     }

    }

  }

}

 Serial.print( " Sistema de Temperatura:  ");
 Serial.println(  );
```

```
  Serial.println("");

  Serial.print( " Consigna de Temperatura:  ");
  Serial.println( _Consigna );
  Serial.println("");

}

void Habitacion::MostrarConsigna(){

  String StrConsigna = String( _Consigna );

  uint8_t Num1 = StrConsigna[0] - '0';
  uint8_t Num2 = StrConsigna[1] - '0' ;


  if ( Num1 != _Num1Anterior ) {

    switch ( Num1 ) {

      case 0:

        D1.MostrarNum( 0 );

      break;

      case 1:

        D1.MostrarNum( 1 );

      break;
```

```
    case 2:
      D1.MostrarNum( 2 );
    break;
    case 3:
      D1.MostrarNum( 3 );
    break;
  }
}
_Num1Anterior = Num1;
if ( Num2 != _Num2Anterior ) {
  switch ( Num2 ) {
    case 0:
      D2.MostrarNum( 0 );
    break;
    case 1:
```

```
 D2.MostrarNum( 1 );

break;

case 2:

 D2.MostrarNum( 2 );

break;

case 3:

 D2.MostrarNum( 3 );

break;

case 4:

 D2.MostrarNum( 4 );

break;

case 5:

 D2.MostrarNum( 5 );

break;

case 6:

 D2.MostrarNum( 6 );
```

```
    break;

    case 7:

      D2.MostrarNum( 7 );

    break;

    case 8:

      D2.MostrarNum( 8 );

    break;

    case 9:

      D2.MostrarNum( 9 );

    break;

   }

 }

 _Num2Anterior = Num2;


}
```

```
//////////////////////////Sistema de Persiaas////////////////////////////////

void Habitacion::SisPersianas(){

  uint8_t TiempoPer = 10; //10 segundos para que se cierre o se abra la persiana
  bool Tiempo = 0;

  uint8_t PerCmd[2];

  uint8_t PerValue[1];

  bool TramaEnviada;

  XBeeAddress64 direccion;

  //Proceso de decidir si se envia una trama API
  if ( _AddressModPulsador.size() > 0 ) {
    //Se obtiene las lecturas de los pulsadores
    for (int i = 0; i < _AddressModPulsador.size(); i++) {

     if ( ioSample.getRemoteAddress64().getMsb() == _AddressModPulsador[i].getMsb() &&
ioSample.getRemoteAddress64().getLsb() == _AddressModPulsador[i].getLsb() ) {

       if ( ioSample.isDigitalOn(1) == 1 ) {

         if ( _EstadoAnteriorPerS== 0 ) {

           for (size_t i = 0; i < _Persianas.size(); i++) {

             _Persianas[i].getASubir().Enviar();
```

```
        }

       }

      }

      _EstadoAnteriorPerS= ioSample.isDigitalOn(1);

      if ( ioSample.isDigitalOn(2) == 1 ) {

       if ( _EstadoAnteriorPerB == 0 ) {

         for (size_t i = 0; i < _Persianas.size(); i++) {

           _Persianas[i].getABajar().Enviar();

         }

       }

      }

      _EstadoAnteriorPerB = ioSample.isDigitalOn(2);

    }

  }

}

 //Cuando sea la hora de despertar las persianas se subiran -> Hora de mañana
```

```
 if ( ( _RModoF.Horario( HoraM , MinutoM) == 1 ) && ( _RModoF.Horario( HoraN , MinutoN)
== 0 ) && _EstadoAnteriroHoraM == 0 ) {

    for (size_t i = 0; i < _Persianas.size(); i++) {

      _Persianas[i].getASubir().Enviar();

    }

  }

 _EstadoAnteriroHoraM = _RModoF.Horario( HoraM , MinutoM);

 //Para apagar la persiana cuando se esta subiendo en el horario de mañana

 //Cuando sea la hora de dormir las persianas se cerraran -> Hora Nocturna
 if ( ( _RModoF.Horario( HoraN , MinutoN) == 1 ) && _EstadoAnteriroHoraN == 0 ) {

    for (size_t i = 0; i < _Persianas.size(); i++) {

      _Persianas[i].getABajar().Enviar();

    }

  }

 _EstadoAnteriroHoraN = _RModoF.Horario( HoraN , MinutoN);

//Para apagar la persiana cuando se esta subiendo transcurrido un Tiempo
 for (size_t i = 0; i < _Persianas.size(); i++) {
```

```
    if ( _Persianas[i].getASubir().GetEstado() == 1 ) {

      Tiempo = _RPer.Temporizador( TiempoPer );

      if ( Tiempo == 1 ) {

        _Persianas[i].getASubir().Enviar();

      }

    }

  }

//Para apagar la persiana cuando se esta bajando pasado un Tiempo
 for (size_t i = 0; i < _Persianas.size(); i++) {

    if ( _Persianas[i].getABajar().GetEstado() == 1 ) {

      Tiempo = _RPer.Temporizador( TiempoPer );

      if ( Tiempo == 1 ) {

        _Persianas[i].getASubir().Enviar();

      }

    }

  }
```

```
//En el siguiente for se recorre el vector de persianas(Todas las persianas
//de la Habitacion)
//Posteriormente, se obtienen los actuadores de las persianas de subida y de
//bajada, y se trabaja con los métodos de la clase actuador

for (size_t i = 0; i < _Persianas.size(); i++) {

  if ( _Persianas[i].getASubir().GetEnviar() == 1 ) {

    Sample = 0;

    //Si el Actuador de bajar esta activado o desactivado
    if ( _Persianas[i].getABajar().GetEstado() == 0 ) {

      if ( _Persianas[i].getASubir().GetEstado() == 0 ) {

        PerValue[0] = 0x5;

      } else {

        PerValue[0] = 0x4;

      }

      Serial.println(" Enviando..... ") ;

      PerCmd[0] = _Persianas[i].getASubir().GetCmd( 0 );
      PerCmd[1] = _Persianas[i].getASubir().GetCmd( 1 );
```

```
    direccion = _Persianas[i].getAddress64();

    TramaEnviada = EnviarTramaAT( xbee, direccion, PerCmd, PerValue );

    if ( TramaEnviada == 1 ) {

     Sample = 1;

     TramaEnviada = 0;

     _RPer.TempReset();

     _Persianas[i].getASubir().NoEnviar();

     //Modifica el estado que tiene al contrario
     _Persianas[i].getASubir().CambiarEstado();

    }

   } else {

    PerValue[0] = 0x4;

    Serial.println(" Enviando..... ") ;

    direccion = _Persianas[i].getAddress64();
    PerCmd[0] = _Persianas[i].getABajar().GetCmd( 0 );
    PerCmd[1] = _Persianas[i].getABajar().GetCmd( 1 );

    TramaEnviada = EnviarTramaAT( xbee, direccion, PerCmd, PerValue );
```

```
      if ( TramaEnviada == 1 ) {

        TramaEnviada = 0;

        _Persianas[i].getABajar().DesAct();

      }

    }

  }

  //El caso del actuador de bajar persianas será similar pero la comprobación
  //es a partir del actuador de subir persianas.
  if ( _Persianas[i].getABajar().GetEnviar() == 1 ) {

    Sample = 0;

    if ( _Persianas[i].getASubir().GetEstado() == 0 ) {

      if ( _Persianas[i].getABajar().GetEstado() == 0 ) {

        PerValue[0] = 0x5;

      } else {

        PerValue[0] = 0x4;

      }

      Serial.println(" Enviando..... ") ;
```

```
PerCmd[0] = _Persianas[i].getABajar().GetCmd( 0 );
PerCmd[1] = _Persianas[i].getABajar().GetCmd( 1 );

direccion = _Persianas[i].getAddress64();

TramaEnviada = EnviarTramaAT( xbee, direccion, PerCmd, PerValue );

if ( TramaEnviada == 1 ) {

  Sample = 1;

  TramaEnviada = 0;

  _RPer.TempReset();

  _Persianas[i].getABajar().NoEnviar();

  _Persianas[i].getABajar().CambiarEstado();

 }

} else {

  PerValue[0] = 0x4;

  Serial.println(" Enviando..... ") ;

  PerCmd[0] = _Persianas[i].getASubir().GetCmd( 0 );
  PerCmd[1] = _Persianas[i].getASubir().GetCmd( 1 );
```

```
        direccion = _Persianas[i].getAddress64();

        TramaEnviada = EnviarTramaAT( xbee, direccion, PerCmd, PerValue );

        if ( TramaEnviada == 1 ) {

          TramaEnviada = 0;

          _Persianas[i].getASubir().DesAct();

        }

     }

   }

 }

}



//////////////////////////Ahorro de energia////////////////////////////////

void Habitacion::DetecPersonas(){
  //Si el sensor de movimiento no detecta movimiento en la habitación
  //durante 5 minutos, significara que no hay personas en la habitación

  double TemporizacionPersonas = 10; //Tiempo que decide cuando no hay personas
                                     //en la Habitacion
  bool TiempoPersonas = 0;
```

```
 if ( _Movimiento == 0 ) {

   TiempoPersonas = _RPersonas.Temporizador( TemporizacionPersonas );

 } else {

   _Personas = 1;
   _RPersonas.TempReset();

 }

 if ( TiempoPersonas == 1 ) {

   _Personas = 0;

 }



}

void Habitacion::ModoSleep(){

 //se debe indicar la direccion del modulo casa

 bool TramaEnviada;

 bool Enviado = 0;

 uint8_t payload[3];
```

```
uint8_t SleepCmd[] = "SM";
uint8_t SleepValue[1];

XBeeAddress64 direccion;

if ( _Personas == 0 && _EstadoAnteriorPersonas == 1 || _Personas == 1
   && _EstadoAnteriorPersonas == 0 ) {

 Sample = 0;

 if ( _Personas == 0 ) {

  SleepValue[0] = 0x4;
  payload[0] = 0x0;
  payload[1] = 0x1;
  payload[2] = 0x1;

 }  else {

  SleepValue[0] = 0x0;
  payload[0] = 0x0;
  payload[1] = 0x1;
  payload[2] = 0x2;

 }

 //Modulos pulsadores
 for (size_t i = 0; i < _AddressModPulsador.size(); i++) {

  Enviado = 0;
```

```
    while ( Enviado == 0 ) {

      Serial.println(" Enviando SLEEP..... ") ;

      TramaEnviada = EnviarTramaAT( xbee, _AddressModPulsador[i],
                              SleepCmd, SleepValue );

      if ( TramaEnviada == 1 ) {

         Enviado = 1;
         TramaEnviada = 0;
     }

   }

   //Modulos de movimiento
   for (size_t i = 0; i < _AddressMovimiento.size(); i++) {

     Enviado = 0;

     while ( Enviado == 0 ) {

      Serial.println(" Enviando SLEEP..... ") ;

      TramaEnviada = EnviarTramaAT( xbee, _AddressMovimiento[i], SleepCmd, SleepValue );

      if ( TramaEnviada == 1 ) {

        Enviado = 1;
        TramaEnviada = 0;
      }
```

```
  }

}

//Módulo de persianas
for (size_t i = 0; i < _Persianas.size(); i++) {

   Enviado = 0;

   while ( Enviado == 0 ) {

    Serial.println(" Enviando SLEEP..... ") ;

    //Se crea una variable con el valor de la direccion de la persiana
    //porque el método setRemoteAddress64 pasa la variable por referencia
    //por lo tanto debe ser una variable y los que nos devuelve el método
    //_Persianas[i].getAddress64() es un valor.

    XBeeAddress64 direccion = _Persianas[i].getAddress64();

    TramaEnviada = EnviarTramaAT( xbee, direccion, SleepCmd, SleepValue );

    if ( TramaEnviada == 1 ) {

     Enviado = 1;
     TramaEnviada = 0;

    }

   }
```

```
  }

  //EL siguiente while indica al modulo casa que la habitacion esta preparada o no para ser dormida
  Enviado = 0;

  while ( Enviado == 0 ) {

    //Direccion del modulo casa
    XBeeAddress64 addressCasa = XBeeAddress64(0x0013A200, 0x4163D942);

    TramaEnviada = EnviarTramaTx( xbee, addressCasa, payload)

    if ( TramaEnviada == 1 ) {

      Enviado = 1;

      TramaEnviada = 0;

    }

  }

  Sample = 1;

 }

 _EstadoAnteriorPersonas = _Personas;

 Serial.print("Personas: ");
```

```
  Serial.println( _Personas );

}
```

12.11 Clase MóduloRemoto

12.11.1 Archivo hpp

```
#ifndef ModuloRemoto_
#define ModuloRemoto_

#include "Direccion.h"

class ModuloRemoto : public Direccion{
private:

  bool _Sleep;



public:
  ModuloRemoto ();

  void setSleep( bool Sleep );

  bool getSleep();

};

#endif
```

12.11.2 Archivo cpp

```
#include "ModuloRemoto.h"
```

```
ModuloRemoto::ModuloRemoto(){

 _Sleep = 0;

}

void ModuloRemoto::setSleep( bool Sleep ){

 _Sleep = Sleep;

}

bool ModuloRemoto::getSleep(){

 return _Sleep;

}
```

12.12 Clase Casa

12.12.1 Archivo hpp

```
#ifndef Casa_
#define Casa_

#include "ArduVector.h"

#include "XBee.h"

#include "SoftwareSerial.h"

#include "ModuloRemoto.h"
```

```
#include "SenMovi.h"

class Casa {
private:

XBee xbee;

//////////////////////////Añadir Dispostitivos////////////////////////////////

ZBRxResponse RxData;

//

//En el siguiente vector se almacenan las direcciones de los modulos Habitacion
ArduVector<ModuloRemoto> _MRemoto;

SenMovi _M;

//ModoSleep

bool _CasaVacia;

bool _EstadoCasaVacia;

public:

  Casa ();

  void Inicializar( XBee Xbee);
```

```
//////////////////////////Añadir Dispostitivos////////////////////////////////
 //void Add( XBeeAddress64 address );

 void AddRemoto( ModuloRemoto remoto );

 void RecibirDatos();

 void DetecPersonas();

 void ModoSleep();

};

#endif
```

12.12.2 Archivo cpp

```
#include "Casa.h"

bool EnviarTrama( XBee xbee, XBeeAddress64 address, uint8_t cmd[2], uint8_t Value[1] ){

 RemoteAtCommandRequest remoteAtRequest;
 RemoteAtCommandResponse remoteAtResponse = RemoteAtCommandResponse();

 uint8_t value[1];

 value[0] = Value[0];

 remoteAtRequest = RemoteAtCommandRequest( address, cmd, value, sizeof(value));

 xbee.send(remoteAtRequest);
```

```
 if (xbee.readPacket(5000)) {

   if (xbee.getResponse().getApiId() == REMOTE_AT_COMMAND_RESPONSE) {

      xbee.getResponse().getRemoteAtCommandResponse(remoteAtResponse);


      if (remoteAtResponse.isOk()) {

        remoteAtRequest.clearCommandValue();

        return 1;

      } else {
        remoteAtRequest.clearCommandValue();

        return 0;

        }

    }

  }

}


Casa::Casa(){

  RxData = ZBRxResponse();
```

```
   _MRemoto.reserve(10);

   _M.Add(2);

   _CasaVacia = 0;

   _EstadoCasaVacia = 0;

}

void Casa::Inicializar( XBee Xbee ){

   xbee = Xbee;

}

void Casa::AddRemoto( ModuloRemoto remoto ){

 _MRemoto.push_back( remoto );

}

void Casa::RecibirDatos(){

   xbee.readPacket();

   if ( xbee.getResponse().isAvailable() ) {

     if (xbee.getResponse().getApiId() == ZB_RX_RESPONSE) {

       xbee.getResponse().getZBRxResponse( RxData );
```

```
    }

  }

  else {

    Serial.print("Expected Remote AT response but got ");
    Serial.println(xbee.getResponse().getApiId(), HEX);

  }

}


void Casa::DetecPersonas(){

 uint8_t NumHabSleep = 0; //Contador que cuenta las habitaciones que estan
                          //preparadas para dormir

 //En el siguiente for se realizaran dos tareas
 //la primera, detectar si ha entrada una trama Rx de los modulo habitacion
 //diciendo que la habitacion se puede dormir
 //codigo 011 esta preparadas para dormir
 //codigo 012 no esta preparada para dormir


 //En segundo lugar se contara si todos los modulos habiatcion de la casa
 //estan preparados para dormir, por lo tanto, se colocaran todos los módulos
 //habiatcion en modo sleep
```

```
 for (size_t i = 0; i < _MHabitacion.size(); i++) {

   if ( RxData.getRemoteAddress64().getMsb() ==
      _MHabitacion[i].getAddress64().getMsb() &&
      RxData.getRemoteAddress64().getLsb() ==
      _MHabitacion[i].getAddress64().getLsb() ) {

    if ( RxData.getData(0) == 0 && RxData.getData(1) == 1 &&
        RxData.getData(2) == 1 ) {

      _MHabitacion[i].setSleep( true );

    }

    if ( RxData.getData(0) == 0 && RxData.getData(1) == 1 &&
        RxData.getData(2) == 2 ) {

      _MHabitacion[i].setSleep( false );

    }

   }

   if ( _MHabitacion[i].getSleep() == true ) {

    NumHabSleep++;

   }

 }
```

```
  //Si el contador NumHabSleep es igual al número de habitaciones la casa estará
  //preparada para dormir
  if ( NumHabSleep != _MHabitacion.size() || _M.Salida() == 1) {

    _CasaVacia = 0;

  } else {

    _CasaVacia = 1;

  }


}

void Casa::ModoSleep(){

  bool Enviado = 0;

  bool TramaEnviada = 0;

  uint8_t SleepCmd[] = "SM";
  uint8_t SleepValue[1];


  if ( _CasaVacia == 1 && _EstadoCasaVacia == 0 || _CasaVacia == 0
      && _EstadoCasaVacia == 1 ) {

    if ( _CasaVacia == 0 ) {

      SleepValue[0] = 0x0;
```

```
  } else {

    SleepValue[0] = 0x4;

  }

  //Modulos habitaciones
  for (size_t i = 0; i < _MHabitacion.size(); i++) {

    Enviado = 0;

    while ( Enviado == 0 ) {

      Serial.println(" Enviando SLEEP..... ") ;

      XBeeAddress64 direccion = _MHabitacion[i].getAddress64();

      TramaEnviada = EnviarTrama( xbee, direccion, SleepCmd, SleepValue );

      if ( TramaEnviada == 1 ) {

        Enviado = 1;
        TramaEnviada = 0;

      }

    }

  }
```

```
 }

 _EstadoCasaVacia = _CasaVacia;

}
```

13. Programas Arduino

13.1 Entrada

```
#include "Casa.h"



XBee Xbee = XBee();

//Declaración del objeto XBee
Casa C;



//Declaración de las direcciones de los módulos Remotos
ModuloRemoto Habitacion1;
ModuloRemoto Habitacion2;
ModuloRemoto Habitacion3;

ModuloRemoto Sala;
ModuloRemoto Cocina;
ModuloRemoto Entrada;
ModuloRemoto Pasillo;
ModuloRemoto Balcon;
ModuloRemoto Bano1;
ModuloRemoto Bano2;
```

```
void setup() {

Serial.begin(9600);
Xbee.setSerial(Serial);

//Se introducie las características del XBee al atributo xbee del objeto Casa
C.Inicializar( Xbee );

//Se añaden las direcciones de 64 bits, en este ejemplo todas tendrán la misma
Habitacion1.Add( 0x0013A2004154DCFC );
Habitacion2.Add( 0x0013A2004154DCFC );
Habitacion3.Add( 0x0013A2004154DCFC );

Sala.Add(0x0013A2004154DCFC);
Cocina.Add(0x0013A2004154DCFC);
Entrada.Add(0x0013A2004154DCFC);
Pasillo.Add(0x0013A2004154DCFC);
Balcon.Add(0x0013A2004154DCFC);
Bano1.Add(0x0013A2004154DCFC);
Bano2.Add(0x0013A2004154DCFC);

//Se introduce cada módulo remoto en la clase Casa

C.AddRemote( Habitacion1 );
C.AddRemote( Habitacion2 );
C.AddRemote( Habitacion3 );
C.AddRemote( Sala );
C.AddRemote( Cocina );
C.AddRemote( Entrada );
```

```
C.AddRemote( Pasillo );
C.AddRemote( Balcon );
C.AddRemote( Bano1 );
C.AddRemote( Bano2 );



}

void loop() {

  C.RecibirDatos();

  C.DetecPersonas();

  C.ModoSleep();



}
```

13.2 Cocina

```
#include "Habitacion.h"

XBee Xbee = XBee();

//Declaración de las direcciones de los módulos Remotos
XBeeAddress64 AddressPulsador;
XBeeAddress64 AddressActuadorPersiana;



Habitacion H;
```

```
void setup() {

Serial.begin(9600);

//Se introduce el puerto serial al Arduino al dispositivo XBee
Xbee.setSerial(Serial);
H.AddXBee( Xbee );

//Direcciones de los módulo remotos, en este ejemplo son iguales
AddressPulsador = XBeeAddress64( 0x0013A200, 0x4154DCFC );
AddressActuadorPersiana = XBeeAddress64( 0x0013A200, 0x4154DCFC );

//Añadir modulos remotos a la clase Habitacion
H.AddMPulsador( AddressPulsador );
H.AddLuz( AddressPulsador, "D4" );
void AddPersiana( AddressActuadorPersiana , "D2", "D3" );

}

void loop() {

  H.RecibirDatos();

  H.HMovimiento();

  H.HOscuridad();

  H.OnOFFLuz();
```

```
  H.SisPersinas();

  H.DetecPersonas();
  H.ModoSleep();

}
```

13.3 Sala

```
#include "Habitacion.h"

XBee Xbee = XBee();

//Declaración de las direcciones de los módulos Remotos
XBeeAddress64 AddressPulsador;
XBeeAddress64 AddressMovimiento;
XBeeAddress64 AddressPersiana;
XBeeAddress64 AddressClimatizacion;


Habitacion H;


void setup() {

Serial.begin(9600);

//Se introduce el puerto serial al Arduino al dispositivo XBee
Xbee.setSerial(Serial);
H.AddXBee( Xbee );
```

```
//Direcciones de los módulo remotos, en este ejemplo son iguales
AddressPulsador = XBeeAddress64( 0x0013A200, 0x4154DCFC );
AddressPersiana = XBeeAddress64( 0x0013A200, 0x4154DCFC );
AddressMovimiento = XBeeAddress64( 0x0013A200, 0x4154DCFC );
AddressClimatizacion = XBeeAddress64( 0x0013A200, 0x4154DCFC );



//Añadir modulos remotos a la clase Habitacion
H.AddMPulsador( AddressPulsador );
H.AddLuz( AddressPersiana, "D0" );
H.AddPersiana( AddressPersiana , "D1", "D2" );
H.AddClimatizacion( AddressClimatizacion , "D1", "D2" );
H.AddMovmiento(AddressMovimiento);

}

void loop() {

  H.RecibirDatos();

  H.HMovimiento();

  H.HOscuridad();

  H.ModoF();

  H.OnOFFLuz();

  H.SisTemperatura();

  H.MostrarConsigna();
```

```
 H.SisPersianas();

 H.DetecPersonas();
 H.ModoSleep();

}
```

13.4 Zonas Comunes

```
#include "Habitacion.h"

XBee Xbee = XBee();

//Declaración de las direcciones de los módulos Remotos
XBeeAddress64 AddressLuz;

Habitacion H;


void setup() {

Serial.begin(9600);

//Se introduce el puerto serial al Arduino al dispositivo XBee
Xbee.setSerial(Serial);
H.AddXBee( Xbee );

//Direcciones de los módulo remotos, en este ejemplo son iguales
AddressLuz = XBeeAddress64( 0x0013A200, 0x4154DCFC );
```

```
//Añadir modulos remotos a la clase Habitacion
H.AddLuz( AddressPulsador, "D0" );



}

void loop() {

  H.RecibirDatos();

  H.HMovimiento();

  H.HOscuridad();

  H.OnOFFLuz();

  H.DetecPersonas();
  H.ModoSleep();

}
```

13.5 Habitaciones

```
#include "Habitacion.h"

XBee Xbee = XBee();

//Declaración de las direcciones de los módulos Remotos
XBeeAddress64 AddressPulsador;
```

```
XBeeAddress64 AddressPersiana;


Habitacion H;


void setup() {

Serial.begin(9600);

//Se introduce el puerto serial al Arduino al dispositivo XBee
Xbee.setSerial(Serial);
H.AddXBee( Xbee );

//Direcciones de los módulo remotos, en este ejemplo son iguales
AddressPulsador = XBeeAddress64( 0x0013A200, 0x4154DCFC );
AddressPersiana = XBeeAddress64( 0x0013A200, 0x4154DCFC );


//Añadir modulos remotos a la clase Habitacion
H.AddMPulsador( AddressPulsador );
H.AddLuz( AddressPulsador, "D4" );
H.AddPersiana( AddressPersiana , "D1", "D2" );
H.AddClimatizacion( AddressClimatizacion , "D1", "D2" );


}

void loop() {

  H.RecibirDatos();
```

```
  H.HMovimiento();

  H.HOscuridad();

  H.ModoF();

  H.OnOFFLuz();

  H.SisTemperatura();

  H.MostrarConsigna();

  H.SisPersianas();

  H.DetecPersonas();
  H.ModoSleep();

}
```