Trabajo de Fin de Grado

GRADO EN INGENIERÍA ELECTRÓNICA INDUSTRIAL Y AUTOMÁTICA

Estudio y Análisis de Sensores y Actuadores Orientados a Aplicaciones Domóticas Inalámbricas

José David González de la Guardia.
Feliciano Javier Mikó Mba Nchama

Tutor:
Alberto Francisco Hamilton Castro.

La Laguna, a 13 de septiembre de 2017

D. **Alberto Francisco Hamilton Castro**, con N.I.F. 43773884-D Profesor del Departamento de Ingeniería Informática y Sistemas de la Universidad de la Laguna.

I N F O R M A:

Que el presente trabajo de fin de grado, titulado "*Estudio y Análisis de Sensores y Actuadores Orientados a Aplicaciones Domóticas Inalámbricas"*, ha sido realizado bajo su supervisión por D. **José David González de la Guardia**, con N.I.F. 78647198-D y por D. **Feliciano Javier Mikó Mba Nchama** con N. I. E. Y2126767-V, y autorizo la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en La Laguna a 13 de septiembre de 2017.

Agradecimientos

Quisiéramos tener en consideración y agradecer a varias personas la ayuda que hemos recibido directa e indirectamente durante la realización de este Trabajo de Fin de Grado.

En primer lugar a nuestro tutor Alberto Francisco Hamilton Castro por su disposición a ayudarnos a superar los obstáculos que se iban interponiendo. Sobretodo agradecer su paciencia, sus consejos y haber compartido sus conocimientos con nosotros. Al profesor Alejando Ayala por habernos ayudado a resolver algunas dudas de electrónica.

Por otro lado a nuestros familiares y amigos que han sido un pilar fundamental en nuestras vidas, dándonos todo y apoyándonos día a día. Dándonos fuerzas para seguir hacia delante y a alcanzar nuestras metas.

"La recompensa se encuentra en el esfuerzo y no en el resultado.

Un esfuerzo total es una victoria completa"

MAHATMA GANDHI

Resumen:

Este proyecto realiza la implementación de un sistema domótico, para el control remoto de la Temperatura, Iluminación, Consumo Eléctrico, etc., mediante el uso de dispositivos XBee con acceso a la red inalámbrica utilizando para ello el protocolo ZigBee. Se realiza un análisis exhaustivo sobre los sensores y actuadores para desarrollar las configuraciones, calibraciones, y el diseño de acondicionadores de señales pertinentes. Toda esta la información es necesaria para el desarrollo de un prototipo de red domótica inalámbrica que permita el sensado y control del sistema a muy bajo coste económico y energético.

Los principales aspectos que podemos destacar son los siguientes:

- La integración de varios dispositivos para una intercomunicación entre los sensores y los módulos XBee. Configurando la red ZigBee para su adaptación a cualquier entorno, ofreciendo flexibilidad al usuario.
- Mediante la comunicación entre el nodo coordinador y el nodo dispositivo final se permite visualizar los datos sensados en tiempo real mediante una computadora

Palabras clave: Domótica, ZigBee, Inalámbrica, XBee, Sensores, Actuadores.

Abstract:

This project realizes the implementation of an automation system, for remote control of Temperature, Brightness, Electricity Consumption, etc., through the use of XBee devices connected to wireless network using the ZigBee protocol. A Comprehensive analysis is made using recommended sources about sensors and actuators to perform the configurations, calibrations, and design of relevant signal conditioner circuits. All information would be necessary in order to developing a wireless home automation network prototype, that allows the sensing and control of the system at a lower economic and energy cost.

The main ones that can highlight the following:

- The integration of several devices for an intercommunication between sensors and XBee modules. Configuring the ZigBee network to adapt to any environment, offering flexibility for any user.
- Through the communication between the ZigBee Coordinator and the ZigBee End Device allows to view the sensed data at real time using a computer

Key words: Home Automation, ZigBee, Wireless, XBee, Sensors, Actuators.

Índice General

Índice Figuras

Índice Tablas

CAPÍTULO 1: INTRODUCCIÓN

1.1 MOTIVACIONES Y ANTECEDENTES

En el transcurso de los años, las instalaciones de las viviendas han evolucionado. Antiguamente, los usuarios se conformaban con disponer de iluminación, calefacción y puntos de acceso (tomas de corriente) para poder conectar sus electrodomésticos a la red eléctrica. Pero las características que deben perdurar son aquellas que los usuarios soliciten, como pueden ser: una mayor facilidad, flexibilidad y a la vez una mayor interconectividad. Para incorporar todos los aspectos mencionados, suponía un incremento del volumen de cableado en la vivienda y, además, contar con las intervenciones de los instaladores eléctricos para solventar posibles problemas.

Gracias al desarrollo tecnológico, se produce el hecho de pensar en controlar dispositivos remotamente, ya sea desde internet o desde una computadora personal y con una infinidad de dispositivos que existen en nuestro día a día. Entonces se podrían controlar algunos aspectos de la vivienda como la iluminación, la temperatura o la humedad de forma remota

En la actualidad, los escenarios donde los seres humanos interactúan son del tipo inteligente. Estas nuevas viviendas están dotadas de un sistema nervioso donde la palabra inteligente se ha empezado a posar sobre todo lo que la constituye, desde los electrodomésticos hasta la arquitectura en sí misma. Todo esto con el único fin de integrar la nueva era tecnológica al hogar tradicional para convertirlo en el hogar inteligente [1].

1.2. OBJETIVOS

El principal motivo por el que se ha realizado la puesta en marcha de este proyecto, es con el fin de integrar en un sistema domótico varios dispositivos (sensores y actuadores) interconectados para controlar diferentes aspectos de una vivienda domótica (Temperatura, Humedad, Luminosidad, etc.). Además de optimizar y fomentar el ahorro energético y económico a nivel domótico.

CAPÍTULO 2: ESTADO DEL ARTE. CONCEPTOS

2.1. DOMÓTICA.

La domótica es la instalación e integración de varias redes y dispositivos electrónicos en el hogar, permitiendo la automatización de actividades cotidianas, el control local o remoto, de la vivienda o de un edificio inteligente. Se compone de las palabras: **domus** (vivienda en latín) y **tica** (de automática).

Debido a la expansión de la palabra Domótica, se distinguen tres campos diferentes:

- **Inmótica**: para el sector terciario, industrial (edificios, colegios, etc.).
- **Urbótica**: Utilizada en las ciudades.
- **Domótica**: destinado al sector doméstico.

Pero para definir una vivienda domótica o automatizada, hay que tener en cuenta dos puntos de vista: del **usuario** y el **técnico**.

- Desde el **usuario**, una vivienda domótica es una integración de soluciones para el hogar compuesta de muchos productos que el usuario final puede utilizar para el beneficio de su día a día.
- Desde lo **técnico**, una vivienda domótica sería aquella en la que se integran los distintos aparatos domésticos que tienen la capacidad de comunicarse entre sí, para sustituir las tareas que hasta el momento se realizaban de forma manual [2].

2.1.1. Aplicaciones de la Domótica

Los servicios que ofrece la domótica se dividen en los siguientes ámbitos principales:

- **Ahorro energético**: No es algo tangible. En muchos casos no es necesario sustituir los sistemas del hogar por otros que consuman menos, sino una gestión eficiente de los mismos: Climatización, Gestión eléctrica, sistemas acumuladores.
- **Confort**: Conlleva las actuaciones que permiten la mejora de la comodidad en una vivienda. Pudiendo ser tanto activas, pasivas o mixtas: Iluminación, Control vía internet.
- **Seguridad**: Se encarga de proteger tanto los Bienes Patrimoniales como a la seguridad personal: Simulación de presencia, Cerramiento de persianas, Acceso a cámaras.
- **Comunicaciones**: Sistemas o infraestructuras de comunicaciones del que dispone el hogar: Control remoto desde internet, Mandos inalámbricos [2], [3].

2.1.2. Medios de transmisión en Redes Domóticas

Son la manera por el que se transmite la información entre los diferentes dispositivos en la vivienda. Los tipos de transmisión son los siguientes:

- **Transmisión por conductores:** Transmite las órdenes y el estado de los sistemas en la Red. Ejemplo: Cable coaxial, Fibra óptica, etc.
- **Transmisión sin conductores:** Se basa en el envío y la recepción de ondas electromagnéticas que utilizan nuestro entorno para su propagación.
  Ejemplo: Infrarrojos y Radiofrecuencia.

Dentro de la transmisión sin conductores, la **transmisión por Radiofrecuencia** (RF) permite el envío de información entre dos puntos distantes. La transmisión se realiza mezclando los datos que se deben transmitir o recibir con la onda portadora. Por ello son empleados los siguientes sistemas en la configuración de redes inalámbricas: Bluetooth, Home RF, Z-Wave, **Zigbee** y Wi-Fi [3].

2.2. INTRODUCCIÓN A LA COMUNICACIÓN INALÁMBRICA

En un sentido amplio y general, se entienden por comunicaciones inalámbricas aquellas comunicaciones entre dispositivos que intercambian información utilizando el espectro electromagnético. Favorece que el usuario pueda mantenerse conectado cuando se desplaza dentro de una zona geográfica, facilitando el acceso a recursos en lugares donde la utilización de cableado es limitada. Permiten que los dispositivos remotos se conecten con facilidad, sin necesidad de realizar cambios en la estructura del lugar donde se va a instalar [4].

2.2.1. Redes de Comunicaciones

Las redes inalámbricas se clasifican en varias categorías, de acuerdo al área geográfica desde la que el usuario se conecta a la red: WLAN, WPAN, WWAN, WMAN.

- **Redes de Área Local (WLAN).** Son redes privadas que se ubican en un edificio o a pocos kilómetros de longitud. Se puede aplicar en la interconexión de las estaciones de trabajo o para compartir recursos e información.

- **Redes de Área Personal (WPAN).** Son redes de comunicación de uso privado, y presentan características similares a las WLAN. El nombre WPAN proviene que están orientadas a la conexión inalámbrica de terminales portátiles. Su rango de cobertura es de 10m, con velocidades de hasta 1Mbps. Las normas para la WPAN son: Bluetooth, UWB, **Zigbee** e IrDA.

- **Redes de área extendida inalámbricas (WWAN).** Una red de área extendida, es una red que intercomunica equipos en un área geográfica muy amplia. Las transmisiones en esta Red se realizan a través de líneas públicas. Esta capacidad de transmisión suele ser menor que las utilizadas en las redes de área local. Además, son compartidas por muchos usuarios a la vez.

- **Redes de área metropolitana inalámbricas (WMAN).** Es una red de distribución para un área geográfica en el entorno de una ciudad. Es apropiado para la distribución de televisión por cable, internet, la telefonía tradicional, etc.

Pero la categoría de red que utilizan los dispositivos ZigBee es la WPAN por una transmisión basada en la norma IEEE 802.15.4. [4]

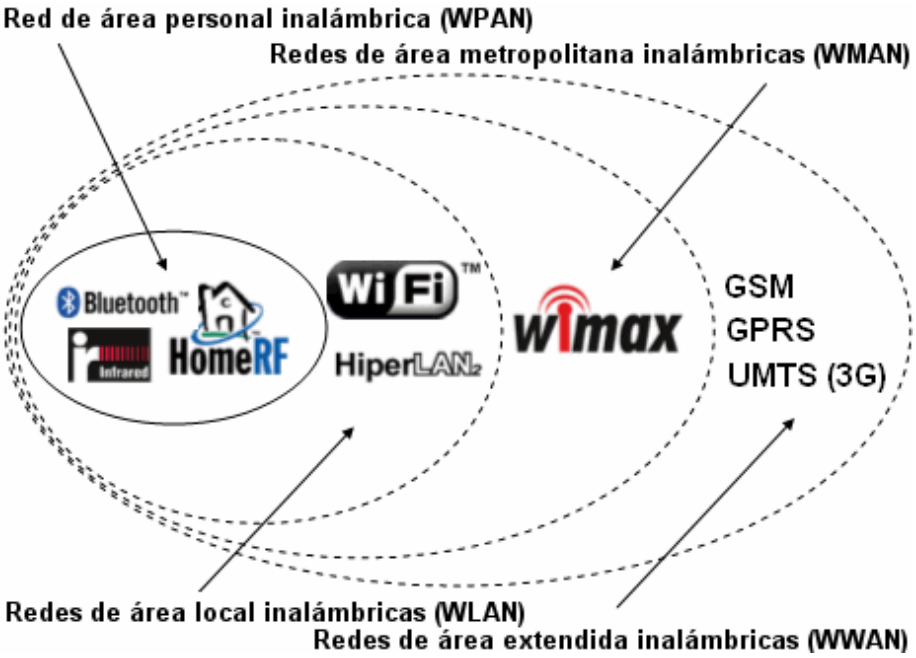

Figura 1. *Redes de Comunicación Inalámbrica [16]*

2.3. IEEE 802.15.4 Y PROTOCOLO ZIGBEE

2.3.1 Introducción

ZigBee es un sistema de comunicación inalámbrica centrado en la comunicación entre dispositivos a corta distancia, con la finalidad de conseguir optimizar la comunicación con el menor consumo energético posible. Esta tecnología inalámbrica de es implementada para controlar y monitorizar aplicaciones o procesos dentro de un rango desde 10 a 100m. Además, es un conjunto de protocolos de alto nivel de comunicación inalámbrica, regulado por el estándar IEEE 802.15.4 de redes WPAN [3].

Adopta al estándar IEEE 802.15.4 para sus dos primeras capas, es decir, la Capa Física (FL) y la subcapa de Acceso al Medio (MAC) para soportar muchos dispositivos con una baja tasa de transmisión de datos, y agrega las Capas de Red y de Aplicación.

2.3.2. Características.

- Pueden transferir datos a velocidades desde 25 Kbps hasta 250 Kbps (empleado en aplicaciones que requieran baja tasa de transmisión de datos). Esto le permite coexistir a la misma frecuencia con otras tecnologías inalámbricas como Bluetooth o Wifi.
- Formada por un máximo de 255 nodos.
- Puede ser utilizado en diferentes configuraciones de red (maestro – maestro y maestro - esclavo).
- Las redes ZigBee pueden ser ampliables a través del uso de Routers.

- Actúa en las bandas libres ISM (Industrial, Scientific & Medical) de 2,4 GHz, 868 MHz (Europa) y 915 MHz (Estados Unidos).
- Utiliza un protocolo asíncrono y estandarizado, que permite a dispositivos diferentes fabricantes operar conjuntamente.
- Capacidad de operar en redes de gran densidad. Cuantos más nodos haya dentro de la red, mayor número de rutas alternativas existen para que el paquete llegue a su destino.
- Cada red ZigBee tiene un identificador de red único, lo que permite que varias redes puedan coexistir en un mismo canal.
- Evita las interferencias, garantizando un funcionamiento libre de preocupaciones. Permite el control de energía, encender o apagar los dispositivos de forma remota
- Los dispositivos ZigBee tienen la posibilidad de funcionar en un modo de bajo consumo, lo que supone prolongar la vida de sus baterías [3].

Principales aplicaciones:

- Control remoto.
- Productos dependientes de la batería.
- Sensores.

En el ámbito de la domótica:

- Climatización.
- Utilización de energías renovables.
- Apagado general de la iluminación de la vivienda. Encendido/Apagado de zonas concretas. Regulación de la iluminación.
- Automatización de distintos sistemas.
- Control de alarmas, detección de presencia, video vigilancia, cerramiento de persianas.

En el ámbito de la salud:

- Seguimiento de enfermedades crónicas.
- Supervisión del bienestar personal.
- Personal fitnes

En el ámbito de las comunicaciones:

- Control remoto desde internet, desde una computadora o mandos a distancia.
- Intercomunicaciones

También puede utilizarse en el control industrial, detección de humo, etc., o como en nuestro caso para desarrollar redes de sensores inalámbricos para uso en sistemas domó ticos [3], [6].

El protocolo ZigBee posee tres funcionalidades importantes dentro de una red como son:

- **Ruteo:** Reenvío de mensajes entre módulos distantes.
- **Creación de redes Ad Hoc:** Redes que se crean automáticamente de forma espontánea.
- **Auto Regeneración:** Identificación de nodos y rutas para que la red siga funcionando.

En la siguiente imagen, se destacan las principales empresas encargadas de su fabricación y distribución:

Figura 2. *Empresas que componen ZigBee Alliance [17]*

2.3.3. Arquitectura del Protocolo ZigBee

La arquitectura del protocolo ZigBee consiste en un conjunto de capas donde el estándar IEEE 802.15.4, definido por la capa FL y por la subcapa MAC. ZigBee amplía el estándar introduciendo las Capas de Red y de Aplicación. De esta forma, los estándares IEEE 802.15.4 y ZigBee se complementan proporcionando un conjunto completo de protocolos que permite la comunicación entre multitud de dispositivos de una forma eficiente y sencilla:

- **Capa Física** (IEEE 802.15.4): Esta capa realiza operaciones de modulación y demodulación en la transmisión y recepción de señales.

- **Capa MAC** (IEEE 802.15.4): Es la responsable de la transmisión segura de datos accediendo a diferentes redes.

- **Capa de Red** (ZigBee Alliance): Encargada de todas las operaciones relacionadas con la gestión de la red como son: la configuración de la red, conexión y desconexión del nodo final a la red.

- **Capa de soporte a la aplicación** (ZigBee Alliance): Establece una interfaz para la capa de red y los objetos de los dispositivos.

- **Capa de Aplicación**: Esta capa consta de la subcapa de aplicación, del ZDO (ZigBee Device Objects) que se encarga de definir el papel del dispositivo en la red.

Estos protocolos proveen un alto rendimiento en cuanto a la transmisión de paquetes por radio frecuencia y una alta inmunidad, por lo que los dispositivos ZigBee son más robustos frente a interferencias que los que siguen los estándares Bluetooth o Wi-Fi. [3]

2.3.4. ¿Por qué ZigBee y no otro sistema de comunicación Inalámbrico?

A continuación, se realizará una comparativa de las tecnologías inalámbricas que existen en el mercado. En donde se demostrará que utilizamos ZigBee porque se adapta mejor a las condiciones que nuestro trabajo requiere.

Comparación de Tecnologías Inalámbricas			
	Wifi	**Bluetooth**	**ZigBee**
Estándar	802.11.X	802.15.1	802.15.4
Medio	Radio Frecuencia	Radio Frecuencia	Radio Frecuencia
Banda de Frecuencias	2.4 GHz	2.4 GHz	2.4 GHz, 868 MHz, 915 MHz
Velocidad de transmisión	54 Mbps	1 Mbps	250 Kbps
Alcance	120 m	10 m	100m
Tipos de Datos	Digital	Digital	Digital/Analógica
Número de dispositivos	32	8	255/65535
Arquitecturas	Estrella	Estrella	Estrella, Árbol, Punto a Punto, Maya
Coste	Alto	Bajo	Bajo
Complejidad	Complejo	Complejo	Simple
Consumo	Medio	Bajo	Bajo

Tabla 1. *Comparativa entre ZigBee y otras tecnologías inalámbricas.*

Entrando en detalle, cada uno de los sistemas es más apropiado que el otro dependiendo de las aplicaciones en los que se va a utilizar. Por tanto, en comparación con las redes Bluetooth y Wi-Fi, este sistema de comunicación es más barato y simple. Mientras que la velocidad de transmisión empleada por ZigBee hace que no sea óptima como por ejemplo para implementar aplicaciones de teléfonos móviles, sin embargo, es empleado en aplicaciones domóticas, productos dependientes de baterías, sensores y en juguetes en los cuales la transferencia de datos es menor. Por otro lado, ZigBee tiene un menor consumo eléctrico que el resto de sistemas. Concretamente ZigBee consume 30mA transmitiendo y 3 uA en reposo, frente a los 40mA transmitiendo y 0,2 mA en reposo que tiene el Bluetooth, gracias a que el sistema ZigBee se queda la mayor parte del tiempo dormido [3].

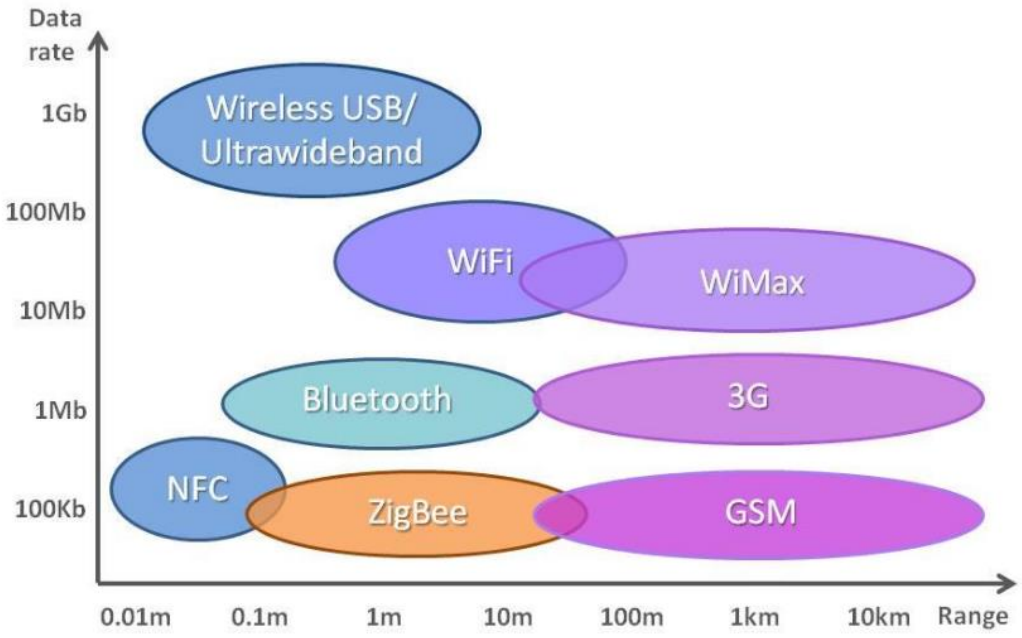

Figura 3. *Comparativa Tecnologías Inalámbricas. Velocidad de transmisión de datos – Alcance [18]*

2.3.5. Dispositivos ZigBee

ZigBee define dos tipos de nodos en la red categorizado según su funcionalidad:

- **Dispositivos de Funcionalidad Completa (FFD):** Estos dispositivos Implementan de forma completa la capa MAC. Pueden ser utilizados en cualquier topología. Estos pueden ser el Coordinador y el Router.

- **Dispositivos de Funcionalidad Reducida (RFD):** Dispositivos de funcionalidad limitada, como es el caso de los Dispositivos Finales, End Devices (ED). Tienen implementada de forma parcial la capa MAC y su uso se restringe solo a topología en estrella. Son los sensores/actuadores de la red.

Con lo que ZigBee define tres tipos de elementos con los dos nodos anteriores: El Coordinador, El Router y el Dispositivo Final.

- **Coordinador:** Toda red ZigBee debe tener al menos un coordinador responsable de iniciar, controlar y formar la red. También es el encargado de seleccionar el canal de comunicaciones y el Identificador de red PAN ID (Personal Area Network Identifier). Se encarga de la seguridad y la autoregeneración de la red., además puede recibir y transmitir datos.

- **Router:** Se une a una red previamente formada. Tienen la capacidad de rutear o enrutar el tráfico entre nodos distantes, es decir, puede servir de intermediario entre diferentes dispositivos que componen la red. También se emplean para ampliar las dimensiones de la red facilitando la creación de nuevos caminos para el envío de datos para que, en caso de bloqueo, tengan otras alternativas para llegar al nodo de destino.

- **End Devices:** Son capaces de transmitir y recibir información sólo del nodo padre, ya sea un coordinador o un router. Carecen de la función de enrutamiento de datos, al igual que la capacidad de introducir otros dispositivos a la red. Si necesitan comunicarse con otros ED no lo puede hacer directamente sino a través de un dispositivo padre. Tienen la posibilidad de operar en **Modo Sleep**, lo cual reduce el consumo energético alargando la vida de las baterías [3], [5].

Respecto a nuestro proyecto, los elementos de ZigBee que vamos a utilizar son:

- Un coordinador y Un Dispositivo Final.

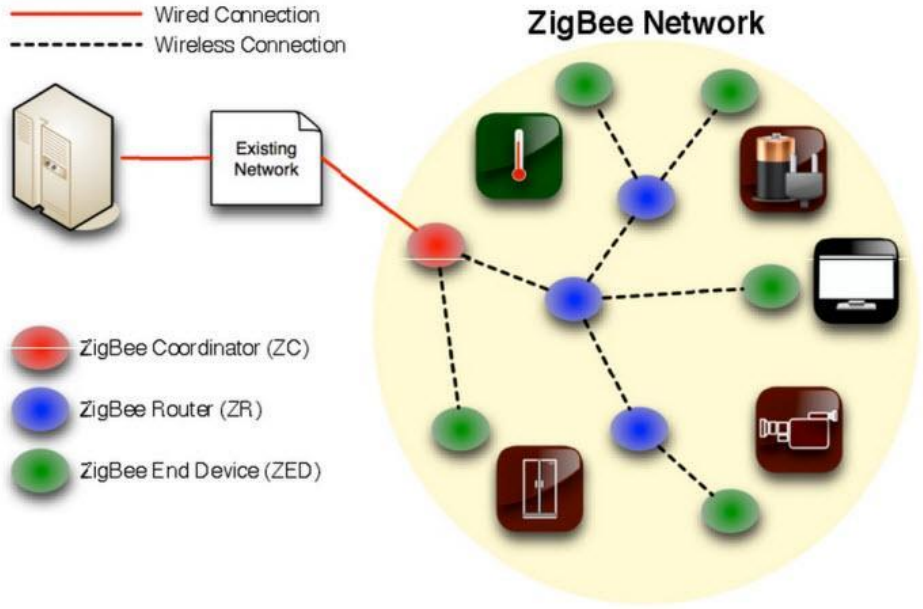

Figura 4. *Dispositivos de una red ZigBee [19]*

2.3.6. Topologías de Red:

La tecnología ZigBee tiene la capacidad de implementar diferentes tipos de topologías, aunque las más utilizadas son estrella, malla y árbol:

- **Topología Estrella (Star):** Formadas por un conjunto de ED que se comunican directamente con el coordinador central. El número de ED está limitado y es sobretodo utilizado en entornos industriales donde los nodos finales necesitan comunicarse con el controlador central (HUB). Consume muy poco por lo que se alarga la vida de las baterías de los dispositivos.

- **Topología Malla (Mesh):** En esta estructura, cualquier dispositivo tiene la posibilidad de comunicarse con cualquier otro a través de los Routers que actúan

como puentes de comunicación. La complejidad de esta red hace posible establecer una multitud de rutas alternativas para el envío de datos al nodo de destino, sin depender de un único nodo. En caso de que cualquier nodo falle, la información es ruteada a través de otro dispositivo. Es también utilizada en entornos industriales donde la redundancia es un factor muy importante en cuanto al control y seguridad de los procesos.

- **Topología Árbol (Cluster Tree):** Esta topología tiene los beneficios de las dos topologías anteriormente nombradas, pues consume poco y tiene la capacidad de enrutar paquetes de datos estableciendo rutas alternativas. Tiene una estructura jerárquica compuesta por el coordinador que inicializa la red, del cual se conectan routers o coordinadores que a su vez pueden conectar otros routers o ED. Como los dispositivos finale ED no tienen la capacidad de introducir nuevos dispositivos a la red, sólo se pueden comunicar con un dispositivo padre (coordinador o router).

  La red Zigbee tiene la posibilidad de organizar sus nodos de forma automática. En caso de que se suprima o añada cualquier nodo, éstos se reconfiguran nuevamente. [5]

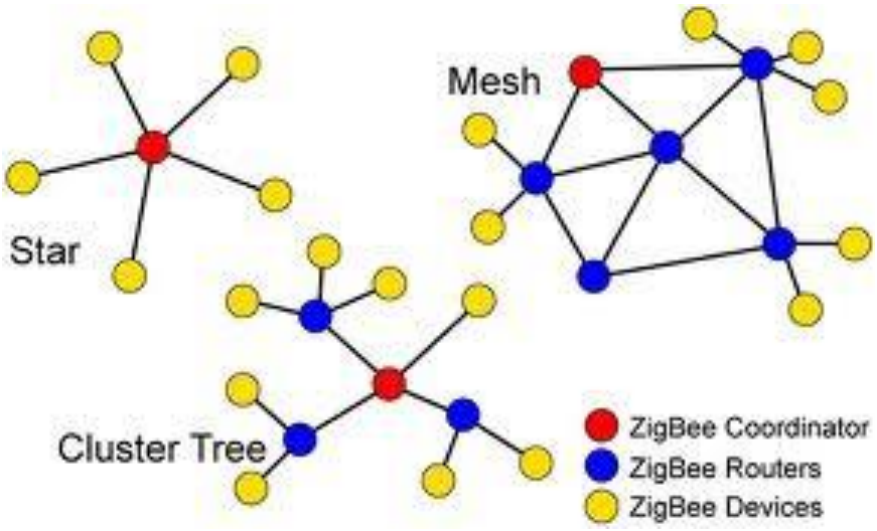

Figura 5. *Tipos de redes ZigBee [20]*

2.4. MÓDULOS XBEE

2.4.1. Introducción.

Los módulos XBee son pequeños módulos electrónicos de radiofrecuencia fabricados por Digi, utilizados para la trasmisión y recepción fiable de datos. Su funcionamiento está basado en el protocolo ZigBee y el estándar IEEE 802.15.4. Son fáciles de usar y pueden ser instalados en lugares donde es difícil la comunicación por cable. [5]

Son flexibles en cuanto a su configuración y fundamentales para la construcción de redes ZigBee (Punto a Punto o Punto a Multipunto), ya que constituyen los nodos de la red.

2.4.2. Series y Características:

- **XBee Series 1** (XBee y XBee PRO)**:** También llamados XBee 802.15.4. Son la serie más fácil para trabajar, no necesitan ser configurados ya que traen incorporado el firmware 802.15.4. Pensados para comunicaciones Punto-a-Punto o en Estrella. Trabajan tan bien como los de la Serie 2, pero sin todo el trabajo de la pre-configuración.

- **XBee Series 2** (XBee Znet 2.5, XBee Znet 2.5 Pro, XBee ZB ZigBee, XBee Pro ZB ZigBee): Fueron diseñados para ser utilizados en aplicaciones de redes con topología de malla. Dentro de este grupo, tenemos los módulos ZB y los módulos 2B, que permiten reducir el consumo de potencia e incorporan un microcontrolador.

    - **Xbee Znet 2.5 (Formalmente Series 2):** Los módulos Series 2 deben ser configurados antes de ser usados. Pueden funcionar en modo Transparente o con comandos API. Pueden funcionar en una red Mesh, creando unos módulos totalmente configurables. Son más difíciles de usar que los de la Serie 1. Los módulos Znet 2.5 ya no se venden, pero han sido remplazados con los módulos ZB, que son más compatibles [7].

    - **ZB (el actual módulo Series 2):** Es el módulo Znet 2.5, pero con un nuevo firmware. Funciona en modo transparente o por medio de comandos API. También funcionan en redes Mesh.

Una de las desventajas de los módulos es que el hardware de las Series 1 y las Series 2 no es compatible, por lo tanto, no se pueden comunicar entre ellos. Los dispositivos de la serie 2 tienen la capacidad de ser convertidos de una plataforma a otra cargando el firmware que se desee usar ya que comparten el mismo hardware. Según las necesidades del usuario final, se elige el firmware que mejor se adapte a la aplicación. En la siguiente tabla se muestra una comparativa entre los dispositivos de ambas series:

Característica	Modelo de XBee						
	Serie 1		Serie 2				
	XBee	XBee Pro	XBee Znet 2.5	XBee Pro Znet 2.5	XBee ZB ZigBee	XBee Pro ZB ZigBee	XBee Pro XSC
Potencia de salida en transmisión	1 mW	63 mW	2 mW	50 mW	2 mW	50 mW	100 mW
Alcance en interiores	30 m	100 m	40 m	120 m	40 m	120 m	370 m
Alcance en línea de vista	100 m	1.6 Km	120 m	1.6 Km	120 m	1.6 Km	24 Km
Régimen de RF en datos: 250 Kbps	SI	SI	SI	SI	SI	SI	9.6 Kbps
Baud Rate de la UART	Hasta 115.2 Kbps	Hasta 115.2 Kbps	Hasta 1 Mbps	Hasta 1 Mbps	Hasta 1 Mbps	Hasta 1 Mbps	Hasta 57.6 Kbps
Frecuencia de operación: 2.4 GHz	SI	SI	SI	SI	SI	SI	900 MHz
Opciones de antena: Conector U. FL, antena de chip, dipolo	SI	SI	SI	SI	SI	SI	Conector U. FL, antena de chip
Topologías de red soportadas	Estrella	Estrella	Estrella Malla	Estrella Malla	Estrella Malla	Estrella Malla	Estrella
Número de canales	16	12	16	16	16	13	7
Terminales de entrada-salida digital	8	8	10	10	10	10	-
Terminales de entrada analógica (ADC 10 bits)	7	7	4	4	4	4	0

Tabla 2. *Comparativa series de XBee [3]*

Estos módulos son capaces de realizar lecturas de sensores tanto analógicos como digitales a través de sus pines y transmitirlas de forma inalámbrica a otros nodos, en nuestro caso desde el dispositivo final al coordinador. Estos pines también pueden ser configurados como salidas para la manipulación de actuadores. Por esta razón se ha optado por la utilización de estos módulos, en concreto disponemos del módulo XBee Znet 2.5 de la serie 2 en nuestro proyecto para la implementación de una red de sensores inalámbricos. Se ha incluido en el apartado de anexos su tabla de características.

De entre sus características más relevantes se encuentran:

- Alto rendimiento a bajo coste. Su alcance se extiende hasta 40m en entornos urbanos o interiores, e incluso hasta 120m en entornos exteriores donde alcanza una visión directa, es decir, no hay ningún obstáculo intermedio. Refleja el bajo consumo con una potencia de salida de 2mW y una tasa de transmisión de datos de 250 Kbps a 2,4 GHz en banda ISM.

- Internamente tienen un convertidor Analógico Digital (ADC). Esto permite la conexión de sensores analógicos sin necesidad de utilizar un ADC externo.

Figura 6. *Módulo XBee Series 2 [21]*

2.4.3. Pines del módulo

En el siguiente diagrama esquemático se muestra la distribución de los pines en el dispositivo y a continuación de describirán en detalle las funcionalidades de cada uno de ellos.

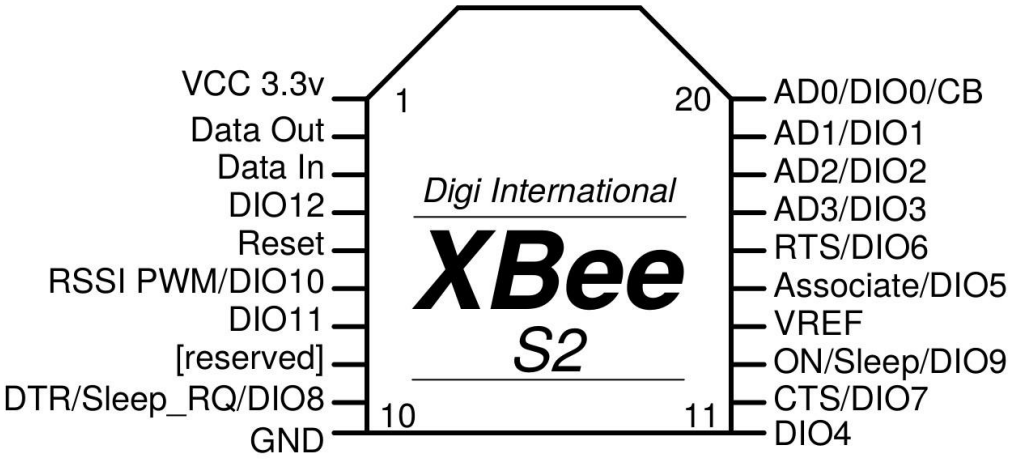

Figura 7. *Distribución de los pines del módulo XBee Series 2*

Pin	Nombre	Dirección	Descripción
1	VCC	-	Alimentación
2	DOUT	Salida	Salida de datos UART
3	DIN/CONFIG	Entrada	Entrada de datos UART
4	DIO12	Entrada/Salida	E/S digital 12
5	RESET	Entrada	Reseteo del módulo
6	PWM0/RSSI/DIO10	Entrada/Salida	PWM salida 0 / Indicador fuera señal RSSI / E/S digital 10
7	PWM/DIO11	Entrada/Salida	E/S digital 11
8	RESERVADA	-	No conectar
9	DTR/SLEEP_RQ/DIO8	Entrada/Salida	Control de hibernación por pin / E/S digital 8
10	GND	-	Tierra
11	DIO4	Entrada/Salida	E/S digital 4
12	CTS/DIO7	Entrada/Salida	Control de flujo Clear-to-send / E/S digital 7
13	ON/SLEEP/DIO9	Salida	Indicador de estado del módulo / E/S digital 9
14	RESERVADA	-	No conectar
15	ASSOCIATE/DIO5	Entrada/Salida	Indicador asociado / E/S digital 5
16	RTS/DIO6	Entrada/Salida	Control de flujo Request-to-send /E/S digital 6
17	AD3/DIO3	Entrada/Salida	Entrada analógica 3 / E/S digital 3
18	AD2/DIO2	Entrada/Salida	Entrada analógica 2 / E/S digital 2
19	AD1/DIO1	Entrada/Salida	Entrada analógica 1 / E/S digital 1
20	AD0/DIO0/	Entrada/Salida	Entrada analógica 0 / E/S digital 0 / Botón de comisionado

Tabla 3. *Descripción de los pines del módulo [5]*

Los terminales de entradas o salidas digitales son 13 en total que van de (DIO0 - DIO12), aunque el pin DIO8 y el DIO9 no son configurables, dejándolo en un total de 11. Cuatro de ellos pueden ser configurados como entradas analógicas (DIO0 - DIO3). Tanto las entradas y salidas digitales como las entradas analógicas comparten los mismos terminales, por lo que es necesario configurarlos de acuerdo con los requisitos establecidos.

2.4.4. XBee. Tipos de Antena:

Cada módulo XBee permite utilizar varios tipos de antena para la transmisión y recepción de señales. Se dispone de los siguientes:

- **Wire antena**: Es un cable que sale por encima del módulo y que permite una radiación omnidireccional si está recta y perpendicular al módulo.

- **Conector U. FL**: Se utiliza para que el usuario pueda conectar su propia antena. Su inconveniente es que requiere de un cable entre conector y antena.

- **Chip Antena**: Es un robusto y pequeño chip que permite una radiación en forma de corazón. Dicha antena capta señal desde la parte frontal, teniendo una mínima sensibilidad en su parte posterior, donde se produce la atenuación. El módulo XBee Znet 2.5 dispone de este chip Antena.

- **Conector RPSMA**: Es una versión más grande del conector “U. FL”, y permite montar la antena sobre el módulo sin necesidad de usar un cable de conexión.

2.5. MODOS DE COMUNICACIÓN

Los módulos XBee ofrecen varios modos para realizar la comunicación entre dispositivos: Modo Transparente, Modo Comando AT y Modo API (Application Programming Interface). Los módulos permiten la elección de uno de ellos, por medio de parámetros de comunicación, según la aplicación donde se vaya a utilizar. [5]

2.5.1. Modo Transparente.

En este modo, la comunicación resulta sencilla y fácil de configurar. Se forma una conexión entre dos nodos y ambos pueden enviar o recibir información mediante su puerto serial. La comunicación inalámbrica viene a sustituir al cable serial. El flujo de datos es recibido por el pin DIN y pasa a la cola del buffer de entrada para luego ser transmitidos mediante conexión inalámbrica al nodo de destino. Si los datos llegan al módulo de destino a través de RF, son enviados por el terminal DOUT y entonces estarían disponibles en el puerto serial del mismo. Este modo de comunicación se emplea en comunicaciones tipo: **Punto a Punto**, es la conexión idónea para reemplazar la comunicación serial por cable; y **Punto a Multipunto**, permite la transmisión de información desde la entrada serial DIN de un módulo a uno o varios módulos que se conectan a la misma red.

2.5.2. Modo Comando AT.

La información recibida por el puerto serial no es transmitida, sino es interpretada por el módulo local. En este caso lo que el módulo recibe por el puerto serial son comandos y no datos. Cuando un comando es recibido por DIN, el módulo lo analiza y lo ejecuta. Si la ejecución es correcta envía o devuelve un mensaje de confirmación. Este es un modo que permite la configuración y control del módulo.

2.5.3. Modo API.

Tiene la función de transmitir datos de forma segura y estructurada a través de tramas (frames). Ofrece una mayor configurabilidad para el aprovechamiento y control de las capacidades del XBee. El flujo de información que entra y sale del XBee viene en una estructura específica empaquetada en frames que definen operaciones y eventos como, por ejemplo, el muestreo. De este modo todos los datos enviados y recibidos a través de la UART (Universal Asynchronous Receiver Transmitter) deben ser tramas de comando API. La UART, en los XBee, sirve para establecer la comunicación y la trasmisión en serie de datos (de manera secuencial) con otros dispositivos como PCs o microcontroladores.

Pueden contener datos o comandos AT. Las tramas que contienen datos se transmiten a los módulos remotos. Si las tramas contienen comandos AT pueden tener 2 destinos: estar dirigidos al módulo local o al módulo remoto, es decir a otros módulos localizados en la red. Esto permite reprogramar y configurar módulos remotamente.

Este modo se emplea en topologías donde interactúan múltiples nodos, como en la configuración malla, y permite el acceso remoto a cualquiera de ellos. No solo facilita la configuración remota sino también la lectura y escritura de sus pines I/O.

Ventajas del modo API son:

- La identificación del origen y destino de las tramas,
- Los datos pueden contener comandos AT.
- Permite la configuración de los módulos de forma remota.
- Facilita la comunicación.
- Permite recibir la confirmación de cada paquete transmitido.

Inconvenientes:

- Mayor complejidad por la estructuración de las tramas.
- Se emplean más bits para enviar la información.

Todas las características mencionadas han supuesto la elección de este modo de comunicación en nuestro proyecto. [6]

2.5.3.1. Tipos de comandos API.

Hay diferentes tipos de comandos API que permiten mandar y recibir comandos AT normales y remotos, saber el estado del módulo, recibir lecturas de entradas digitales y analógicas, enviar y recibir mensajes de datos, etc. El tipo de comando viene definido por el byte identificador (cmdID) que indica la función de la trama y cómo será la estructura de ésta. Permite seleccionar el modo correcto de extraer la información, es decir, ofrece la posibilidad de interpretar correctamente los bytes y los campos que forman la estructura específica, los cuales cambian en función del tipo de comando API.

2.5.3.2. Tramas/Comandos API.

Las tramas API deben cumplir una serie de especificaciones y tener una estructura determinada: [6]

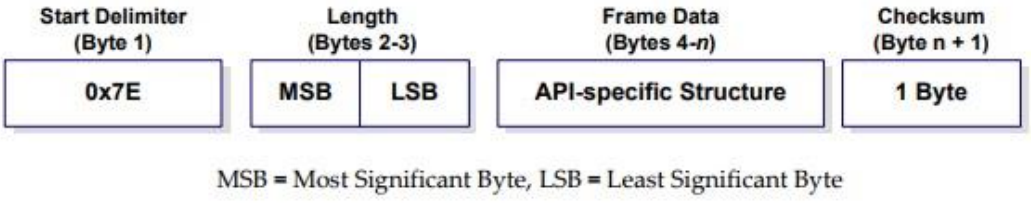

Figura 8. *Estructura básica de las tramas API [5l]*

- **Start Delimiter:** El Byte de inicio indica el comienzo de cada trama. Por defecto su valor es siempre “0x7E”. Cuando un frame es recibido por el puerto serie del XBee lo primero que se busca es el byte de inicio para reconocer el inicio del frame.

- **Length:** Indica el tamaño del campo de Datos (Frame Data). Facilita la información sobre la cantidad de bytes a leer para obtener la información.

- **Frame Data:** Aquí se encuentra contenida la información, concretamente, la carga de la trama. Tiene una estructura específica según el tipo de mensaje enviado - recibido por el XBee.

- **Checksum:** Chequea la integración de los datos de la trama.

En la siguiente tabla se exponen los comandos API que soportan los módulos XBee pertenecientes a la serie 2, concretamente al XBee ZNet 2.5:

API Frame Names	Values
Modem Status	0x8A
AT Command	0x08
AT Command - Queue Parameter Value	0x09
AT Command Response	0x88
Remote Command Request	0x17
Remote Command Response	0x97
ZigBee Transmit Request	0x10
Explicit Addressing ZigBee Command Frame	0x11
ZigBee Transmit Status	0x8B
ZigBee Receive Packet (AO=0)	0x90
ZigBee Explicit Rx Indicator (AO=1)	0x91
ZigBee IO Data Sample Rx Indicator	(0x92)
XBee Sensor Read Indicator (AO=0)	0x94
Node Identification Indicator (AO=0)	0x95

Tabla 4. *Tipos de tramas API [5]*

Los comandos API más utilizados en nuestro proyecto son: **0x97** (Respuesta a Comando AT Remoto) y el **0x92** (Recepción IO remota).

2.5.3.1.1. Respuesta a Comando Remoto.

Se trata de un mensaje de confirmación tras recepción. En este caso, si el módulo recibe una trama de respuesta remota, como respuesta a una solicitud de comando AT remoto, el módulo enviará un mensaje de Respuesta de Comando AT Remoto a la UART.

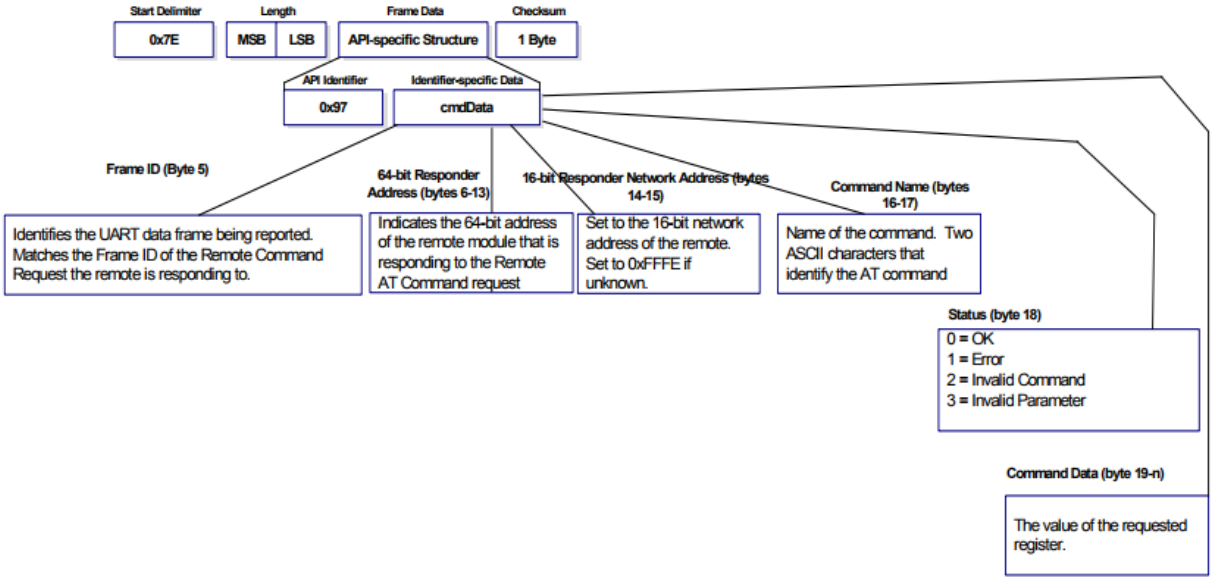

Figura 9. *Estructura de la trama 0x97 [5]*

2.5.3.1.2. Recepción IO Remota.

Tiene la función de verificar el estado de las entradas y salidas del nodo remoto. En nuestro caso utilizaremos el comando AT específico para el muestreo periódico. Concretamente cuando el coordinador recibe una trama de muestreo procedente del nodo remoto, envía la muestra a través de la UART.

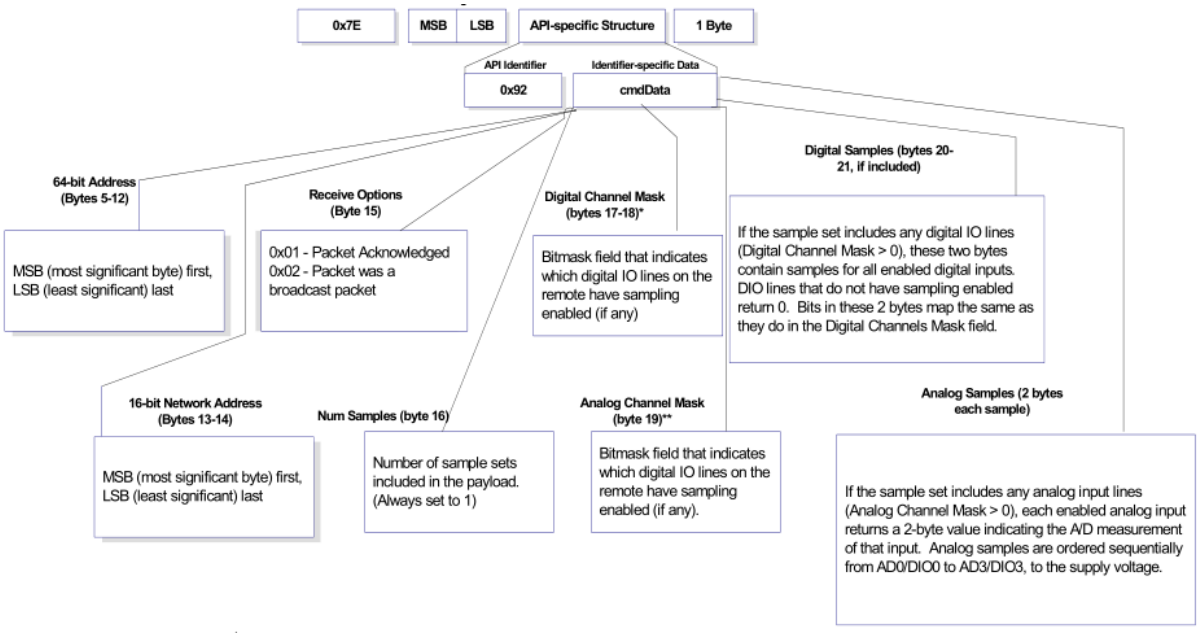

Figura 10. *Estructura de la trama 0x92 [5]*

En la estructura de la trama, concretamente el frame Data contiene varios bytes donde se explican las operaciones de muestreo de los pines del nodo remoto:

- *Máscara de Canales Digitales (Digital Channel Masck)* bytes 17 -18: Indican cuáles de los pines del dispositivo remoto tienen el muestreo habilitado.

- *Máscara de Canales Analógicos (Analog Channel Mask),* byte 19*.* Nos dice cuáles de los pines del dispositivo remoto tienen el muestreo habilitado.

- *Muestras Digitales* (Digital Samples), bytes 20 – 21: Contiene los valores como resultado de las operaciones de muestreo para todas aquellos pines configurados como entradas digitales.

- *Muestras Analógicas (Analog Samples), (2 bytes para cada muestra).* Incluye los valores analógicos muestreados de las entradas configuradas como analógicas [5].

2.5.4. Comandos AT

Para establecer la comunicación entre los módulos XBee se ha hecho uso de diversos comandos AT dependiendo de su funcionalidad. A continuación se detallan los más utilizados:

- **Comandos AT para configuración de la Red.**

Comando	Descripción
ND	Es útil para inicialización. Identifica todos los nodos.
NI	Almacena la cadena identificadora de cada módulo.
CH	Lee el número de canal que se está usando para la transmisión entre los módulos.
SH	Lee los 32 bits superiores de la dirección de 64 bits de cada módulo.
SL	Lee los 32 bits inferiores de la dirección de 64 bits de cada módulo.
VR	Devuelve la versión del firmware de cada módulo.
AI	Devuelve la información relativa de la última petición de unión de cada módulo a la red.
OP	Devuelve el identificador de la red PAN al que está asociado cada módulo.

Tabla 5. *Comandos AT para la configuración de la red [5]*

- **Comandos AT de muestro de datos.**

Comando	Descripción
IS	Realiza una única lectura de todas las entradas analógicas y digitales activadas. Recibimos un paquete tipo trama 97 donde está la información.
IR[x]	Realiza una lectura periódica de todas las entradas analógicas y digitales activadas. El período de tiempo se específica en milisegundos junto al comando. Cada cierto tiempo el dispositivo remoto genera automáticamente un mensaje de recepción.
IC[x]	Se enviará un mensaje cada vez que se produzca un cambio en las entradas digitales que se están monitoreando. El comando usa una máscara de bits hexadecimal para distinguir los diferentes canales de los que se requiere tener información.

Tabla 6. *Comandos AT de muestreo de datos [5]*

Los programas deben ejecutarse independientemente del comando AT de muestreo que se haya enviado: IS, IR o IC. Si sólo tenemos entradas digitales, el comando más recomendable sería el IC, ya que ahorra energía y reduce el número de mensajes. Sin embargo, en caso de que tengamos entradas analógicas, procedentes de las salidas de sensores, es importante que cada cierto tiempo sean muestreadas para obtener información del entorno. Lo más idóneo es utilizar el comando IR, pues al no producirse cambios bruscos en las señales de los sensores no se dispararían mensajes automáticamente.

Centrándonos en el comando IR, se puede configurar el tiempo de muestreo. Para ello se le tiene que pasar el tiempo en milisegundos y en base hexadecimal. Por ejemplo, para configurar IR para un tiempo de muestreo de un segundo: 1s = 1000ms -> 0011 1110 1000_2 -> $3E8_{16}$. En el programa se implementaría de la siguiente forma *"IR3E8"* para que muestree las entradas cada segundo.

Respecto al comando AT IC, hay que indicarle de qué entradas configuradas como digitales queremos obtener información. En este caso se le ha de pasar, en hexadecimal, aquellas entradas habilitadas. Por ejemplo dadas las entradas del XBee:

	DIO12	DIO11	DIO10	DIO9	DIO8	DIO7	DIO6	DIO5	DIO4	DIO3	DIO2	DIO1	DIO0
A	1	0	1	0	0	0	0	1	0	1	0	1	0
B	1	0	0	0	0	0	1	1	0	0	1	0	0

Tabla 7. *Ejemplo utilización del comando AT IC.*

Para el caso A, las entradas habilitadas son: DIO (1,3, 5, 10 y 12):
0001 0100 0010 1010_2 -> $142A_{16}$. Entonces el comando a enviar sería: *IC142A.*

Para el caso B, DIO (2, 5, 6 y 12):
0001 0000 0110 0100_2 -> 1064_{16}. Entonces el comando a enviar sería: *IC1064.*

- **Comandos AT de configuración de los pines.**

PIN	Comando	Modo de configuración
4	**P2[x]** **DIO12**	0 = Señal digital de entrada no monitorizada 3 = Entrada digital monitorizada 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
6	**P0[x]** **DIO10**	0 = Desactivado 1 = RSSI PWM 3 = Entrada digital monitorizada 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
7	**P1[x]** **DIO11**	0 = Señal digital de entrada no monitorizada 3 = Entrada digital monitorizada 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
9	**D8[x]** **DIO8**	0 = Desactivado 3 = Entrada digital 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
11	**D4[x]** **DIO4**	0 = Desactivado 3 = Entrada digital 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
12	**D7[x]** **DIO7**	0 = Desactivado 1 = Control del flujo CTS 3 = Entrada digital 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”) 6 = transmisión RS-485 activada (activada a baja) 7 = transmisión RS-485 activada (activada a alta)
15	**D5[x]** **DIO5**	0 = Desactivado 1 = Led de indicación asociado 3 = Entrada digital 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
16	**D6[x]**	0 = Desactivado 1 = Control del flujo CTS
17	**D3[x]** **DIO3**	0 = Desactivado 2 = Entrada analógica, acabado simple 3 = Entrada digital 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
18	**D2[x]** **DIO2**	0 = Desactivado 2 = Entrada analógica, acabado simple 3 = Entrada digital 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
19	**D1[x]** **DIO1**	0 = Desactivado 1 = Botón de identificación de nodos habilitado 2 = Entrada analógica, acabado simple 3 = Entrada digital 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)
20	**D0[x]** **DIO0**	0 = Desactivado 1 = Botón de identificación de nodos habilitado 2 = Entrada analógica, acabado simple 3 = Entrada digital 4 = Salida digital, con valor en baja (0 “Desactivado”) 5 = Salida digital, con valor en alta (1 “Activada”)

Tabla 8. *Tabla de comandos AT para configuración de pines del módulo XBee [5]*

Hay que tener en cuenta que los pines del XBee configurados como entradas analógicas son capaces de muestrear un rango de valores de tensión desde 0V a 1,2V como máximo. Esto hace necesario la adaptación de las señales de salida de algunos sensores.

2.7. INFORMÁTICA. CONCEPTOS

2.7.1. Lenguaje de Programación.

Un lenguaje de programación se podría definir como una notación formal para describir algoritmos o funciones que serán ejecutadas por el ordenador.

Los lenguajes de programación se pueden clasificar en cinco grupos según su grado de independencia con la máquina: El lenguaje máquina, El lenguaje ensamblador, Los lenguajes de medio nivel, Los lenguajes de alto nivel y Los lenguajes orientados a problemas concretos.

También se pueden clasificar según la forma de sus instrucciones: Lenguajes procedimentales, Lenguajes declarativos, Lenguajes concurrentes, Lenguajes orientados a objetos. En el que destacamos los Lenguajes Orientados a Objetos (LOO) por su utilización en la programación de nuestro sistema.

Un lenguaje de programación es considerado un lenguaje orientado a objetos cuando soporta directamente los **Tipos Abstractos de Datos (TAD)** y **Clases**, pero añade mecanismos para soportar la **herencia** y el **polimorfismo**.

Finalmente, C++ ha sido el lenguaje de programación orientado a objetos implementado debido a sus múltiples características ampliadas del Lenguaje de programación C. Sus principales características son:

- Programación estructurada.
- Abundancia de operadores y tipos de datos.
- Codificación en alto y bajo nivel simultáneamente.
- No está orientado a ningún área en especial.
- Facilidad de aprendizaje.

2.7.2. Sistema Operativo. GNU/Linux. XUbuntu.

Un **Sistema Operativo** es un programa para administrar el hardware y software de un PC para el usuario.

GNU/Linux es una versión para PC del sistema operativo **UNIX,** elegido para servidores de red y estaciones de trabajo. Ofreciendo servicios de internet y herramientas de desarrollo extensas, interfaces de usuario (GUIs) funcionales. Pero se distingue por su flexibilidad, además de su disponibilidad gratuita.

Actualmente GNU/Linux se ha vuelto la plataforma principal para software de fuente abierta, en gran medida creado por el proyecto **GNU/LINUX** de Free Software Foundation. Las distribuciones principales de Linux son: Red Hat, Ubuntu, Fedora, Debian, etc. En nuestro caso utilizamos **XUbuntu** en su versión 16.04 LTS Xenial Xerus, basada en Ubuntu. [7]

XUbuntu es un sistema operativo ligero, estable y configurable. Va destinado a aquellos usuarios que utilicen una computadora con pocos recursos o que busquen un entorno de escritorio muy eficiente.

2.7.3. GIT.

Es un sistema de control de versiones que nos registra los cambios realizados en un archivo o conjunto de archivos. Dispone de tres estados bien diferenciados:

- **Working Directory**: Donde editamos y trabajamos con nuestros proyectos.
- **Staging Area**: Donde escogemos qué archivos están listos para pasar al tercer estado, al igual que decidimos que archivos no están listos por el momento.
- **Repository**: Aquí es el registro de todo nuestro proyecto.

GitHub es un repositorio remoto donde podemos guardar nuestros proyectos, usando Git para su gestión y administración. Otro repositorio remoto es el "**BitBucket"**, que te permite tener repositorios privados en su versión gratuita [20]. Pero se optó por utilizar el "GitHub" como nuestro repositorio remoto, ya que permite el manejo para el control de versiones de un archivo.

Algunos de los comandos utilizados son:

- **git clone**: No sirve para clonar un proyecto, sino para descargar proyectos.
- **git remote**: Vincula nuestro proyecto local con nuestro proyecto remoto.
- **git push**: Para pasar lo que tenemos en la computadora (mediante "**commits**") a repositorio remoto.
- **git pull**: Pasar lo que tenemos en el repositorio remoto a nuestra computadora después de haber hecho previamente un "**Git clone**".
- **git add .** : Prepara la última versión delos archivos.
- **git commit**: Para guardar todos los cambios realizados en el programa con un mensaje para identificarlos.

2.7.4. Librería Proporcionada por el tutor:

La librería que nos proporcionó el profesor/tutor contiene diversas clases necesarias para el desarrollo de la comunicación (**Conexion, Dialogo_API, frame, frame88/8A/92/95/97/AT/IO/Remoto, Serial_Conexion, TablaDirecciones y VectorByte)**. *Dialogo_API* es la clase principal y utiliza, a su vez, la clase *Conexión, mediante* un puntero a *Conexión,* para los datos que vienen por la serial. Cuando recibe un paquete procedente del coordinador, se invoca el método *recibePaquete*, que estudia, trocea y clasifica el paquete.

Una vez que la clase *Dialogo_API* recibe un paquete, procede a determinar el tipo de frame. Si se trata, por ejemplo de un *frame 92*; lo prepara, lo trocea, introduce la información en un objeto tipo *frame 92* e invoca a un método propio de Dialogo_API (*recepciónIOremota)*. *RecepciónIOremota()* en *Dialogo_API* muestra por pantalla la información del paquete*. Por ejemplo "El paquete recibido es de tipo 92 y tiene la siguiente información".*

Cuando se hereda de una Clase, se pueden añadir o modificar tanto atributos como métodos. Concretamente la clase *Gestión1*, proporcionada por el tutor, hereda

de la clase Dialogo_API. Particularmente lo que hace *Gestión1* es modificar los métodos **respuestaATremota()** y ***recepciónIOremota()***, que es donde se desarrolla la lógica implementada. El resto de métodos heredados mantienen la misma funcionalidad ya que no han sido modificados en *Gestion1*. Algunos de estos métodos son: r*eparteMensaje() o recibePaquete()*, etc.

2.7.4.1. Clases de la librería proporcionada por el tutor.

- **Conexión:** Clase Abstracta para conexión Genérica.
- **Dialogo_API:** Clase que implementa la comunicación API.
- **frame:** Frame general que contendrá el VectorByte completo.
- **frame88:** Deriva de frameAT y a su vez de frame.
- **frame8A:** Deriva de frame.
- **frame92:** Deriva de frame, frameRemoto y FrameIO.
- **frame95:** Deriva de frame y frameRemoto.
- **frame97:** Deriva de frame remoto y de frameAT y a su vez de frame.
- **frameAT:** Estructura parte común de respuesta a comando AT (local o remoto).
- **frameIO:** Estructura para la información de muestreo I/O.
- **frameRemoto:** Estructura para las direcciones recibidas en paquete remoto.
- **Serial_Conexion:** Clase para la gestión de la comunicación a través de la serial.
- **TablaDirecciones:** Clase Auxiliar para gestionar la tabla de direcciones.
- **VectorByte:** Clase Auxiliar para agrupar los bytes de los frames.

2.7.4.2. Algunos métodos de la clase Dialogo_API:

Algunos métodos Públicos:

- ***Dialogo_API (Conexión*cp):*** Constructor
- *enviaPaquete ():* Envía el paquete por la conexión creando el frame necesario antes de envío
- *recibePaquete ():* Lee de la conexión hasta completar un frame válido, re el paquete y lo devuelve.
- *comandoATlocal ():* Con y sin parámetro. Envía comandos AT local a través del API usando API 0x08
- ***comandoATremoto ():*** Con o sin parámetro. Envía comando AT remoto a través del API utilizando 0x17.
- *reparteMensaje ():* Reparte un paquete invocando al método correspondiente.
- *tablaDir ():* Devuelve una copia de la tabla de direcciones.

Algunos de los métodos Protegidos:

- ***respuestaATremota (frame 97 f97):*** Envía un mensaje de configuración por el dispositivo remoto.
- ***recepcionIORemota (frame 92 f92):*** Método que se invoca cada vez que se recibe un paquete con información de los pines de entrada/salida del módulo remoto. Muestra por pantalla la información del paquete.
- *_recPaquete ():* Lee de la conexión hasta completar un frame valido, extrae el paquete y lo devuelve.
- *respuestaATlocal(frame 88 f88):* Modo para gestionar paquetes de respuesta AT remota.

2.7.4.3. Clase Gestion1:

Los métodos implementados en esta clase son los siguientes:

- **Gestion1 (conexión*cp):** Constructor de la clase.
- **Configura(std::string _nombreNodo).** Envía comandos para la configuración de los pines del nodo remoto.
- ***respuestaATremota (frame 97 f97):*** Confirmación de la configuración.
- ***recepcionIORemota (frame 92 f92):*** Método modificado por la clase donde se desarrolla la lógica a implementar.

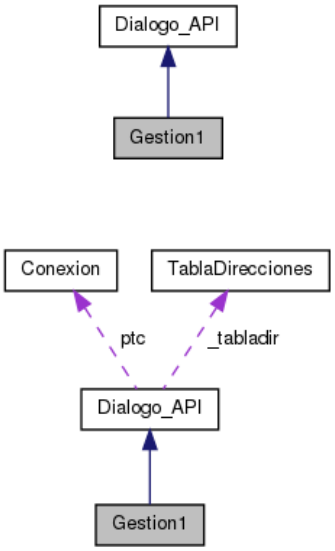

Figura 11. *Herencia y dependencia de clases. Gestion1*

CAPITULO 3. DISEÑO DEL HARDWARE.

Partimos de la documentación de todos y cada uno de los dispositivos (Datasheets, etc.) y posteriormente diseñar, analizar e implementar los circuitos necesarios de acuerdo a los resultados obtenidos tras los análisis.

3.1. IMPLEMENTACIÓN DE DISPOSITIVOS. FASE ELECTRÓNICA.

En esta Primera Fase, en donde se tratará sobre los componentes (Sensores y Actuadores) utilizados para la implementación del Sistema Completo, teniendo en cuenta la documentación (Datasheets) proporcionada por los fabricantes de los mismos. Debido a que los módulos Xbee operan con rangos de voltaje de hasta 3.3V, se necesitó de un acondicionamiento de las señales de entrada al XBee con la finalidad de facilitar la interconexión de los sensores y actuadores; para esto se tuvo que manipular las señales de entradas digitales y analógicas. El acondicionador de señales es uno de los elementos más relevantes de un sistema de adquisición de datos ya que sin la adaptación de las señales, no se podría confiar en la exactitud de la medida.

También se hace uso de las Calibraciones de los sensores. La calibración pretende mantener y verificar el buen funcionamiento de los equipos, responder los requisitos de calidad y de fiabilidad de las medidas.

3.1.1. Calibración de las Fuentes.

Las fuentes de alimentación son elementos a monitorear por su gran importancia para evitar la reconfiguración de los dispositivos si se producen fluctuaciones.

Tras realizar una revisión de todos los circuitos que interpretan las señales procedentes de los sensores, se han detectado anomalías relacionadas con la fuente de alimentación de nuestro circuito. Pues, cada vez que la alimentación oscila entre 4V y 5V procedentes del puerto USB de la computadora, hay que recalibrar cada uno de los circuitos acondicionadores.

Así pues, se decidió sustituir la alimentación del estándar USB de la computadora por la de una **Fuente Externa**. Tanto la alimentación proporcionada por el estándar USB, como la alimentación del integrado uA741 procedente de la Fuente Externa (de +/- 15 V) y la tensión de referencia de offset, han servido para asegurar que el ruido en la alimentación producida sea ínfimo. A continuación, se mostrarán los valores ideales y reales:

Figura 12: *Power Supply FAC-363B*

POWER SUPPLY FAC-363B	**Valor Ideal (V)**	**Valor Real (V)**
Fuente regulable	0,600	0,594
Fuente estática (+)	+15,00	+15,04
Fuente estática (-)	-15,00	-14,79
Fuente estática	5,00	4,87

Tabla 9. Tabla de valores ideales - valores reales de la fuente FAC-363B.

3.1.2. Sensores

Un sensor o transductor es un dispositivo de entrada que provee una salida manipulable de la variable física medida. Nos centraremos en los sensores eléctricos, es decir, aquellos cuya salida nos ofrece una señal eléctrica de tensión o corriente. Dicha salida puede ser **digital** (si funcionan como elementos de detección), o **analógica** (cuando se trata de una medición).

- **Transductores digitales**: Son aquellos que dan como salida una señal codificada en forma de pulsos o de una palabra digital codificada en binario.
- **Transductores analógicos**: Son aquellos que dan como salida un valor de tensión o de corriente que es función continua de la magnitud física medida.

Los sensores pueden clasificarse según el tipo de variable medida: de posición, velocidad; de nivel y proximidad; de humedad y temperatura; de corriente; de flujo y presión, etc. Y algunas de las características son: Sensibilidad, Rango, Precisión, Exactitud, Offset, etc.

Se dispone en el mercado comercial de numerosos tipos de sensores dependiendo de su utilidad. Por ello, se han dispuesto los sensores que han tenido cabida en nuestro proyecto, elegidos por las características propias que nos ofrecen para hacer al sistema cumplir los requerimientos mínimos de usabilidad. Dichos sensores son los siguientes: **Sensor PIR** (HC - SR501), **Sensor corriente** (ACS712),

Sensor de temperatura y humedad (AMT1001), **Fotorresistor LDR** y el **Termistor NTC**.

3.1.2.1. Sensor PIR (HC - SR501).

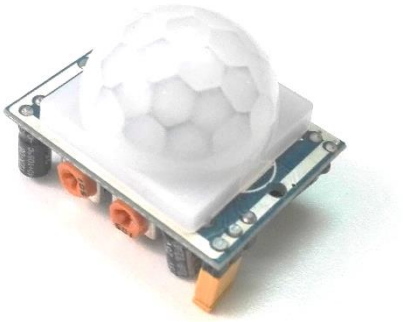

Figura 13.A*: Sensor PIR.*

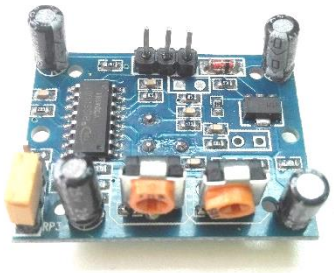

Figura 13.B: *Sensor PIR. Vista de planta*

Es un sensor de movimiento infrarrojo que permite detectar si existe movimiento dentro de su área de funcionamiento. Puede configurarse para detectar únicamente en un rango específico de radiación infrarroja. Además, dispone de una óptica, que es una cúpula de plástico formado por lentes de fresnel, que divide el espacio en zonas, y enfoca la radiación a cada uno de los campos del sensor PIR. [9]

Especificaciones:

- Lente fresnel de 19 zonas, ángulo < 100º.
- Salida activa alta a 3.3 V.
- Consumo de corriente en reposo: < 50 µA.
- Voltaje de alimentación: 4.5 VDC a 20 VDC.

Características:

- Es barato, tiene un tamaño pequeño, baja potencia y facilidad de uso.
- Tiene la capacidad de rechazar falsas detecciones con precisión.
- Permite configurar el tiempo que permanece la salida en alto nivel.

Descripción de sus pines:

- **Vcc:** voltaje de alimentación de 5v.
- **GND:** voltaje de referencia (tierra).
- **OUT:** Señal de salida digital.
- **Resistencia izquierda (Tx):** mediante su variación, se puede establecer con ella el tiempo de activación de la señal, disponiendo de un tiempo mínimo de 3 seg.
- **Resistencia derecha (Sx):** permite variar la distancia del rango de detección del sensor, entre 3 y 7 metros.

3.1.2.1.1. Acondicionador de señal.

No fue necesario ningún acondicionador de señal para este sensor, ya que no fue configurado como entrada analógica para el dispositivo XBee. Su conexión se realizó directamente sobre el módulo, obteniendo de esta manera el funcionamiento deseado.

3.1.2.2. Sensor de corriente (ACS712).

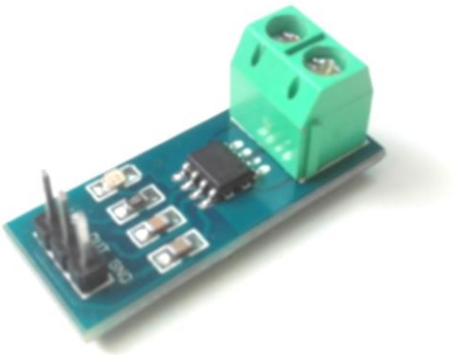

Figura 14: *Sensor ACS712*

Este módulo basado en el circuito integrado ACS712 de Allegro MicroSystems permite medir la cantidad de corriente que fluye a través de un circuito de corriente alterna (AC) o corriente continua (DC). El método de sensado es a través de un sensor de efecto hall que proporciona un voltaje de salida proporcional a la corriente que fluye por el circuito.

Existen tres modelos dependiendo del rango de corriente a medir: (ACS712ELECTR-05B-T), (ACS712ELECTR-20A-T) y (ACS712ELECTR-30A-T). En nuestro caso, disponemos del (**ACS712ELECTR-30A-T)**, el cual tiene un rango de corriente de (-30A a +30A) y una sensibilidad de 66 mV/A [10].

Especificaciones:

- **Tensión de alimentación (Vcc):** 5V.
- **Rango de medida de Corriente (Pico):** - 30A a 30A.
- **Sensibilidad:** 66 mV/A
- **Tiempo de Respuesta:** 3,5us.
- **Ruido:** 7mV.
- **Rango de Tensión de salida:** 0.5V (-30A) - 4,5V (30A); Vout = 2,5V (0A).
- **Ancho de Banda:** 80KHz.
- **Resistencia Interna:** 1,2mΩ
- **La tensión de salida es estable.**

Descripción de sus pines:

- **Vcc:** Tensión de Alimentación (5V).
- **GND:** Tierra (0V).
- **Vout:** Señal de Salida (0.5V a 4,5V).
- **Bornes:** Conexión de la línea a medir.

Comportamiento:

La tensión de salida del dispositivo representa una ecuación lineal con pendiente positiva. La resistencia interna es de 1,2 mΩ proporcionando unas pérdidas muy pequeñas. En las siguientes gráficas se explicará el comportamiento del voltaje de Salida del sensor (Vout) frente a la corriente a medir (Ip), además de la variación de la sensibilidad respecto a la corriente para una alimentación de 5V y Tª = 25ºC.

En las gráficas que facilita el fabricante, se puede notar que la sensibilidad permanece constante (66 mV/A) para cualquier valor de corriente suponiendo Tª = 25ºC.

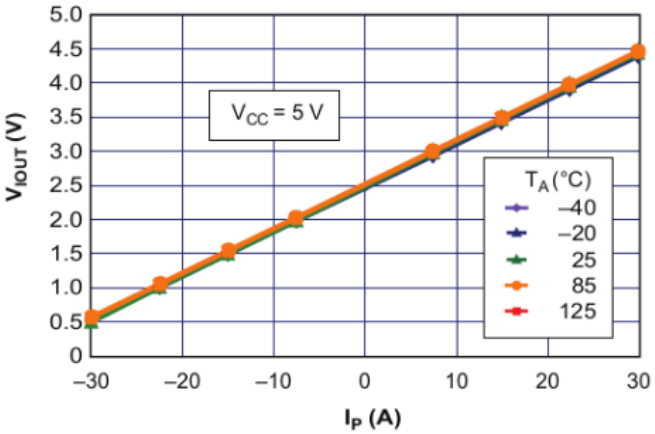

Figura 15A: *Tensión/corriente pico [10]*

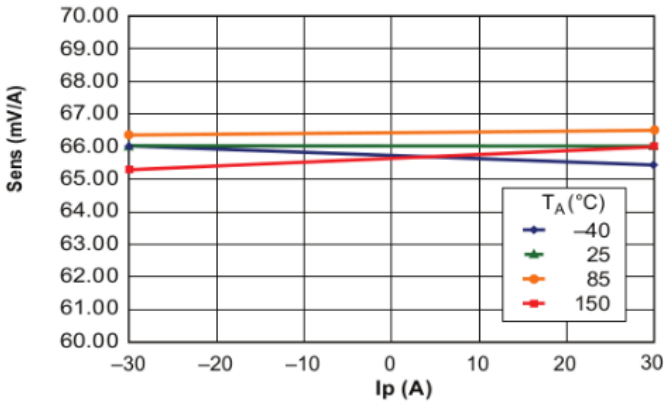

Figura 15B: *Sensibilidad/corriente [10]*

La ecuación que rige su comportamiento sería tal que:

V= Vo + sensibilidad (I - Io)

$$Vout = 2.5 + 0.066 * I \quad ; \quad I\,(medir) = (\frac{Vout-2.5}{Sensibilidad})$$

De esta forma se podría obtener la corriente que circula por el dispositivo a conectar tanto en continua como en alterna:

- Para la corriente continua (CC):

  $\boldsymbol{Icc} = (\frac{Vout - 2.5}{0.066})$

  P = V * Icc

- Para la corriente alterna (AC):

  $\boldsymbol{Ip} = (\frac{Vout-2.5}{0.066}) \quad ; \quad Irms = \frac{Ip}{\sqrt{2}}$

  P = Vrms * Irms = 220 * Irms.

Nota: Hay que tener cuidado con la posición del sensor pues los valores de corriente podrían resultar negativos.

3.1.2.2.1. Acondicionador de señal.

El sensor de corriente presenta a la salida un valor máximo de tensión (Vomax) de 4,5V para un valor de corriente máxima medible de 30A. Por ello es necesario diseñar el acondicionador de señal tal que la corriente máxima medible represente los 1,2Vmáx a la entrada del XBee.
El acondicionador de señal elegido para tal fin ha sido un divisor de tensión, capaz de repartir la tensión de salida del sensor entre una o más impedancias conectadas en serie de la siguiente forma:

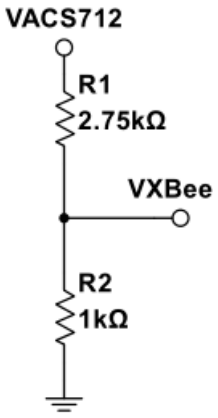

Figura 16: *Acondicionador Sensor de Corriente.*

Tenemos que: **VACS712** representa la salida del sensor (VinM).

VXBee la entrada al XBee (VoutM).

Luego, considerando los valores máximos, según la ecuación siguiente:

$\boldsymbol{VXBee = VACS712 \times (\frac{R2}{R1+R2})}$ y la relación entre las resistencias es: $\boldsymbol{(\frac{R1}{R2}) = (\frac{Vacs712}{VXBee} - 1)}$

que sustituyendo, nos queda: $\boldsymbol{(\frac{R1}{R2}) = (\frac{4.5}{1.2}) - 1 = 2.75}$

$$\boldsymbol{R1 = 2.75 \times R2}$$

Por último, hay que encontrar alguna relación de resistencias que cumpla con lo previsto. Las resistencias han sido R1 = 2750 Ω y R2 = 1000 Ω. Comprobando los valores $\boldsymbol{\frac{R1}{R2} = 2.75}$. Por tanto, se cumple con los requisitos de diseño.

3.1.2.2.2. Calibración

Se han realizado las conexiones necesarias para la obtención de la corriente que circula por distintos dispositivos (cargas) y sus valores a la salida. La calibración se realizó con una alimentación de 5V en continua con una fuente externa, conectando en serie tanto el multímetro configurado como amperímetro, el sensor de corriente y la carga en cuestión. Teniendo lo siguiente:

- **Corriente continua**: Se introdujo unas resistencias de potencia de valores (R1= 15 ohm; R2 = 1 ohm y R3 = 10 ohm). Y pudo notarse que con la corriente inicial

de 0A, el voltaje de salida es **2.49V**. Y a medida que se ha ido aumentando la corriente para las distintas resistencias antes mencionadas, el voltaje a la salida resultaba ir disminuyendo. Con lo que el comportamiento descrito en las especificaciones del sensor no se aproxima al de las pruebas realizadas.

Para el desarrollo de las pruebas se han alimentado el sensor y las resistencias con fuentes independientes para que no influya el consumo de las resistencias en la medición.

- **Corriente alterna**: Se ha realizado la conexión inicial con cargas aisladas (secador, bombilla, una fuente externa de alimentación) por separado, se ha conseguido observar que el voltaje de salida se incrementa respecto al aumento progresivo de la corriente. El problema es que la tensión de salida del sensor no se correspondía del todo al descrito en las especificaciones. El consumo seguía resultando inferior a los valores indicados de consumo por parte del sensor.

Por último, se eligió una regleta para conectar al mismo tiempo varias cargas para que el consumo fuera más elevado. Para ello se conectó un Secador (2000W) y una Aspiradora (1500W); consiguiendo un consumo de 13,3 A. Con lo que el voltaje de salida no superó los 2.6V. Resultando que dicho voltaje no está más allá de los 3V que debería resultar según las especificaciones.

Tras las pruebas realizadas, se procede a determinar la ecuación de la recta mediante la herramienta informática "TableCurve 2D, versión 13". La función de esta herramienta es el análisis matemático, para evaluar datos y obtener las ecuaciones asociadas al comportamiento de los mismos, y tiene una fiabilidad de un 99%. En este caso representa el comportamiento real del sensor a través del cálculo de la Corriente. Dicha ecuación proporcionada será la utilizada en el código de programación. La ecuación de partida es obtenida a partir de la hoja de características:

- Para la corriente continua (CC):
  $\boldsymbol{Icc} = (\frac{Vout - 2.5}{0.066})$
  P = V * Icc

- Para la corriente alterna (AC):
  $\boldsymbol{Ip} = (\frac{Vout-2.5}{0.066})$; $Irms = \frac{Ip}{\sqrt{2}}$
  P = Vrms * Irms = 220 * Irms.

Tras probar con dos módulos distintos, pero con las mismas características se ha observado que el comportamiento en continua y en alterna son diferentes:

- En continua, a medida que aumenta la corriente, el voltaje de salida del sensor va disminuyendo a partir de 2,5 V. Sin embargo, las gráficas que nos muestra las hojas características del sensor describen un comportamiento lineal (Voltaje de salida del sensor proporcional a la intensidad a medir).

- En Alterna, la corriente parte de 0A y el voltaje de salida aumenta al conectar diferentes cargas. El incremento no es tan notable, pero nos sirve para determinar que aumenta con la corriente medible.

En definitiva, se ha decidido implementar en el programa sólo el comportamiento para medir corriente continua, ya que en ambos casos no se pude reflejar el comportamiento esperado.

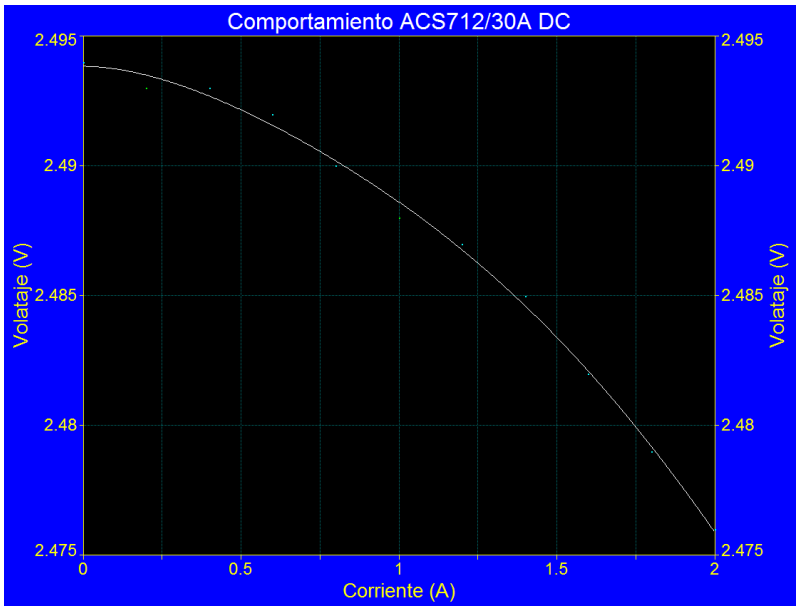

Figura 17: *Sensor ACS712. Curva de calibración Tensión de salida – Corriente DC*

3.1.2.3. Sensor de temperatura y humedad (AMT1001)

Figura 18A: *Sensor AMT1001*

Figura 18B: *Sensor AMT1001. Parte Trasera*

Este módulo ofrece dos sensores en uno, capaz de medir humedad relativa y temperatura. Las medidas que realiza el módulo vienen a ser calculadas a través de un nivel de tensión de salida. Esta tensión es proporcional a la humedad relativa o temperatura según la variable que se desee medir.

Realiza mediciones con alta precisión, de muy buena confiabilidad, consistencia y compensando la temperatura para garantizar estabilidad a largo plazo. [11]

Especificaciones:

- **Alimentación:** 4 - 5,5 Vdc
- **Consumo:** 2mA.
- **Tensión a la salida:** 0.6V - 2,7V.
- **Tiempo de Respuesta:** 10s.
- **Rango de temperatura:** 0 - 60ºC
- **Rango de humedad relativa:** 20 - 90% HR.
- **Sensibilidad:** Humedad +-5%HR y Temperatura +- 1ºC.

Descripción de los Pines:

- **Vin (Rojo):** Tensión de alimentación (5V).
- **Hout (Amarillo):** Señal de salida medida de Humedad (0V - 3V).
- **GND (Negro):** Tierra (0V).
- **Tout (Blanco):** Señal de salida medida de Temperatura (0V - 0,8V).

3.1.2.3.1. Acondicionador de señal.

Con el sensor de temperatura hemos realizado un estudio de su comportamiento ante estímulos externos, aumentando la temperatura considerablemente para observar cambios a la salida.

A pesar de elevar la temperatura hasta el límite de derretir la carcasa externa, ni tan siquiera el voltaje de salida varió unos milivoltios. Sin embargo, el sensor de humedad funciona correctamente.

Se utilizaron otros dos módulos iguales y tras realizar las mismas pruebas nos ha dado el mismo problema con el sensor de temperatura. Por tanto, se deduce que el problema viene de fábrica y es algo que no se pudo corregir. Como el reemplazo para la medición de la temperatura se ha utilizado un termistor NTC que se desarrolla posteriormente.

Así pues, el módulo finalmente se ha montado para las mediciones respecto a la humedad relativa, aunque para el diseño del acondicionador de señal no ha sido nada trivial.

Según las especificaciones, la salida de este sensor presenta un comportamiento lineal que va desde 0,6V (correspondiente a un 20% RH) hasta 2,7V (90% RH), por tanto, hay que adaptar estos valores a la entrada analógica del XBee para que correspondan a 0V y 1,2V respectivamente.

El diseño consiste en enlazar varias etapas. En primer lugar, disponemos del Amplificador Operacional en Configuración Restador para corregir la tensión de Offset de 0,6V, con lo que se reduce el rango de medida de 0V a 2,7V.

Después le sigue un Amplificador Operacional en Configuración Inversora con ganancia inferior a la unidad, reduciendo el rango de medida de “0V” a “- 1,2V”.

Por último, como la salida es invertida se añade otro Amplificador Operacional en Configuración Inversora de Ganancia Unitaria que transforma el rango de medida.

Los resultados obtenidos corresponden a los previstos para garantizar el buen funcionamiento del sensor de cara a su implementación posterior. Se ha conseguido lo siguiente:

- Para una entrada de 0,6V (mínimo valor del sensor) tenemos a la salida 0V.
- Para 3V a la entrada (máximo valor del sensor) tendremos a la salida 1,2V

De cara a la implementación se ha simplificado el circuito combinando los operacionales anteriores en un Amplificador Diferencial con ganancia adaptada a los requerimientos del XBee. Para profundizar en la elaboración de las pruebas para conseguir los resultados anteriores.

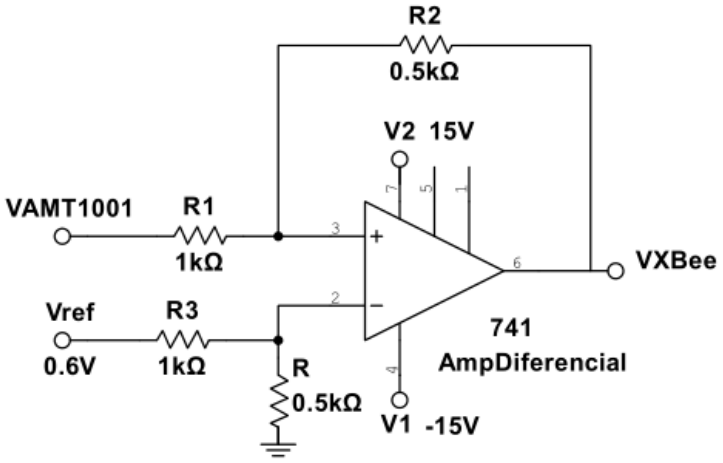

Figura 19: *Acondicionador de Señal. Sensor AMT1001*

3.1.2.3.2. Calibración

Para la calibración del sensor se ha utilizado el Amplificador Operacional (UA741) en Configuración Diferencial con el objetivo de obtener a la salida un máximo de 1,2V.

Se ha dispuesto de la siguiente forma:

- El valor a partir del cual el sensor comienza a medir es 0,6V, que corresponde a un porcentaje de humedad del 20% RH, que corresponde al mínimo valor a la salida 0,6V. Este valor equivale a la tensión “VOffset”.

- El máximo valor a la salida (Vs,max) es 3V, que corresponde a un porcentaje en humedad del 100% RH.

- Rango de valores muestreados por el XBee: (0V - 1,2 V).

Una vez implementado correctamente el amplificador diferencial con su corrección de offset, se procede a establecer una ecuación para la obtención del valor de humedad a partir del valor analógico leído en la entrada. La ecuación que rige el

comportamiento del sensor de humedad es extraída de la hoja de características y es la siguiente:

Vsensor = Voffset + m * (RH - 20) -----> Vsensor = 0,6 + 0,03 * (RH - 20).

Como el comportamiento del sensor está acondicionado por amplificadores y divisores de tensión, hay que establecer una relación entre la tensión a la entrada del XBee y la humedad medida por el sensor.
La expresión de la tensión a la entrada del XBee es: ***VXBee = A (Vsensor - Vref).***

Sustituyendo la ecuación de la curva de calibración en la expresión a la entrada del XBee se tiene lo siguiente:

VXBee = A ((Vofftet + m * (RH - 20)) - Vref) ⇨ ***VXBee = A * (m * (RH - 20))***

Y despejando la variable "RH" tenemos que:

RH = $\frac{VXBee}{A * m} + 20$ ⇨ ***RH =*** $\frac{VXBee}{0.0015} + 20$

Donde:

- **VSensor:** Tensión a la salida del sensor (V).
- **m:** Pendiente de la curva de calibración (0,03).
- **RH:** Humedad Relativa (%RH).
- **A:** Ganancia del amplificador diferencial (0,5)
- **VXBee:** Tensión a la entrada del XBee (V).
- **Vref:** Valor de referencia para compensar el Offset (0,6V).
- **Voffset:** Tensión a partir de la cual el sensor comienza a medir (0,6V).

Esta es la ecuación que nos permite determinar el valor más aproximado de la humedad relativa real del ambiente. Posteriormente en la parte de programación se profundizará un poco más en cuanto a la utilización de esta ecuación.

3.1.2.4. Fotoresistor LDR.

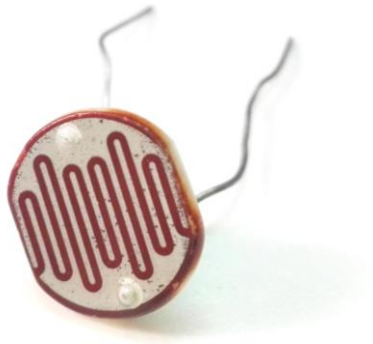

Figura 20: *Sensor LDR*

Son semiconductores que varían su resistencia de acuerdo con las radiaciones luminosas que inciden sobre su superficie. A medida que la intensidad luminosa que incide sobre la LDR aumenta, su resistencia eléctrica disminuye de valor [12].

Especificaciones:

- **Resistencia a la luz (>10 Lux):** 36KΩ (10Lux) - 170Ω (1000Lux).
- **Resistencia en la Oscuridad (0Lux):** 4MΩ.
- **Potencia:** 100mW.
- **Tensión de Alimentación:** 150 Vmax.
- **Rango de Operación (Tª):** -30 - 70 ºC.

3.1.2.4.1. Acondicionador de señal

En este caso, el acondicionador de señal probado fue el Divisor de tensión para conseguir que a la salida del sensor haya un voltaje proporcional a la luz que incide sobre él. Tras el montaje, en que se ha dispuesto una Resistencia en la parte superior del Divisor, mientras que la LDR se ha colocado en la parte inferior. Con ello, se ha conseguido que dicho sensor funcionase de manera correcta, limitando la tensión a su salida.

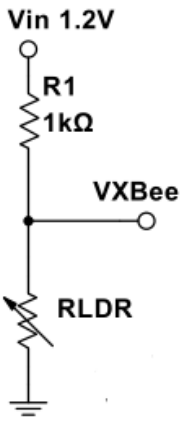

Figura 21: *Acondicionador de Señal. Sensor*

3.1.2.4.1. Calibración

Para realizar el proceso hemos necesitado un luxómetro, pero al no haber este tipo de aparato de medida en los laboratorios, hemos optado por utilizar el sensor de luz y proximidad de un dispositivo móvil.

Como instrumento de medida hemos utilizado una App de Play Store denominado "Luxómetro" vinculada al sensor del Smartphone. Esta aplicación calcula los valores máximos, mínimos y promedio del nivel de luminancia en el ambiente. De esta forma se obtiene la variación de la resistencia según la iluminación y posteriormente se calcula cómo influye al voltaje de salida del Divisor de tensión. Obteniéndose una relación entre el nivel de luminosidad y la tensión de entrada al XBee.

Una vez obtenidas varias muestras se ha optado por utilizar el programa tableCurve 2D para representar la curva del comportamiento y además obtener la ecuación más adecuada para su posterior calibración. La curva que hemos elegido es:

$$L = \frac{a + b}{VXBee} + \frac{c}{VXBee\hat{}\ 2} + \frac{d}{VXBee\hat{}\ 3} + \frac{e}{VXBee\hat{}\ 4}$$

Donde:

- **L:** Nivel de Iluminación (Lux).
- **VXBee:** Tensión a la entrada del XBee (V).
- **a, b, c, d, e:** Constantes.

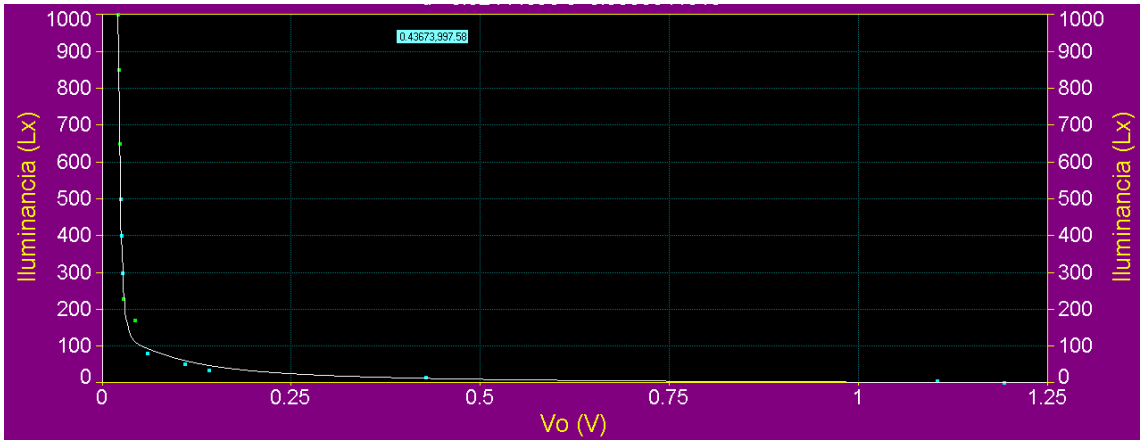

Figura 22: *Sensor LDR. Curva de calibración Iluminancia - Vo*

3.1.2.5. Termistor NTC.

Figura 23: *Sensor NTC.*

Los termistores (Thermally sensitive resistor) son dispositivos basados en la variación de la resistividad de los semiconductores con la temperatura. Al aumentar la temperatura, lo hace también el número de portadores, reduciéndose la resistencia. De ahí que presenten coeficiente de temperatura negativo (NTC). Ofrecen una alta velocidad de respuesta, son de bajo costo y tienen una alta sensibilidad. [13]

Respecto a la sensibilidad, tiene la capacidad de detectar pequeños cambios de temperatura que no se podrían detectar tanto por una RTD como por un Termopar. Dicho incremento de sensibilidad, produce una pérdida de linealidad.

Especificaciones:

- **Resistencia a 25ºC:** 10kΩ.
- **Bandas de color:** Marrón Negro Naranja.
- **Tiempo de respuesta:** 1.2 segundos.
- **Constante A:** 0.01618Ω.
- **Constante B:** 3977 ºK.

3.1.2.5.1. Acondicionador de señal

Respecto a la Calibración de la NTC, una manera de lineal izar su comportamiento a costa de la pérdida de sensibilidad, consiste en disponer el termistor en paralelo con una resistencia auxiliar, considerando al conjunto como un único elemento. Por contra, aunque el comportamiento sigue siendo no lineal, su variación con la temperatura es menor [23].

Se han realizado dos montajes para estudiar el comportamiento y analizar cuál de ellas representa mejor la relación (R-T) y (V-T). Para ello, han sido necesarios: Dos termistores medidos a temperatura ambiente con las resistencias R1 y R2 (1K2 Ohm y 10K Ohm) respectivamente y dos multímetros digitales. La alimentación (Vin) se ha limitado a 1,2V, de esta forma difícilmente sobrepasaremos este umbral.

En un primer momento, al colocar al sensor NTC en la parte inferior del Divisor de tensión, se ha producido el efecto no deseado, aumentando la tensión de salida al disminuir la temperatura. Por lo que se ha decantado por la configuración del NTC colocado en la parte superior del Divisor de tensión.

Una segunda prueba ha consistido en utilizar un Divisor de tensión para que esa variación exponencial de resistencia se vea reflejada a la salida en forma de tensión. Para ello, se ha colocado al sensor NTC en la parte inferior del Divisor de tensión y otra

resistencia auxiliar en la parte superior del Divisor. Se ha conseguido comprobar el buen funcionamiento del sensor, en el que la resistencia disminuye cuando se va calentando el sensor y la tensión de salida del sensor se ha incrementado al calentarlo.

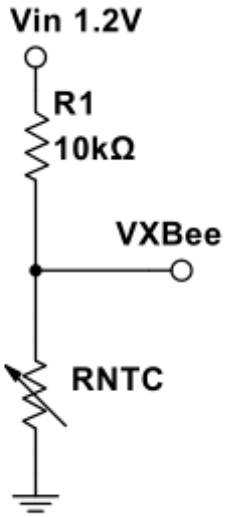

Figura 24*: Acondicionador de señal. Sensor*

3.1.2.5.1. Calibración

Para la medición de la temperatura se dispuso del Termistor NTC. Dicho sensor ha sido sometido ante estímulos externos de temperatura para visualizar su funcionamiento.

Para ello, utilizamos un multímetro digital a través de una sonda que incorpora un termopar. Siendo un Termopar la unión de dos metales diferentes en un extremo, generando en otro extremo una diferencia de potencial que es proporcional a la temperatura. Para probar su precisión comprobamos la medición de la temperatura corporal, y el valor que salía reflejado en el multímetro era entre (36 - 37 ºC), lo cual era óptimo para su utilización.

Como el comportamiento es de tipo exponencial, se optó por obtener una tabla de valores que relacionarán temperatura frente a la tensión de salida (T-Vo) para implementarlo en nuestro programa. Para llevarlo a cabo de forma experimental, utilizamos una pistola de calor para el incremento de temperatura y un bloque de hielo, dispuesto en el interior de una caja de cartón con ventilación forzada, para evaluar la reducción de la temperatura.

Una vez obtenida la tabla de valores, utilizamos un programa que nos calculase la ecuación más próxima al comportamiento de la tensión de salida frente a la temperatura (Vo-T).

La ecuación que regirá el comportamiento del sensor será la siguiente:

T = (a + c*VXBee +e*VXBee^2 + g*VXBee^3 + i*VXBee^4) / (1 + b*VXBee + d*VXBee^2 + f*VXBee^3 + h*VXBee^4)

Donde:

- **T:** Temperatura a Calcular (ºC).
- **VXBee:** Tensión a la entrada del XBee (V).
- **a, b, c, d, e, f, g, h, i:** Constantes.

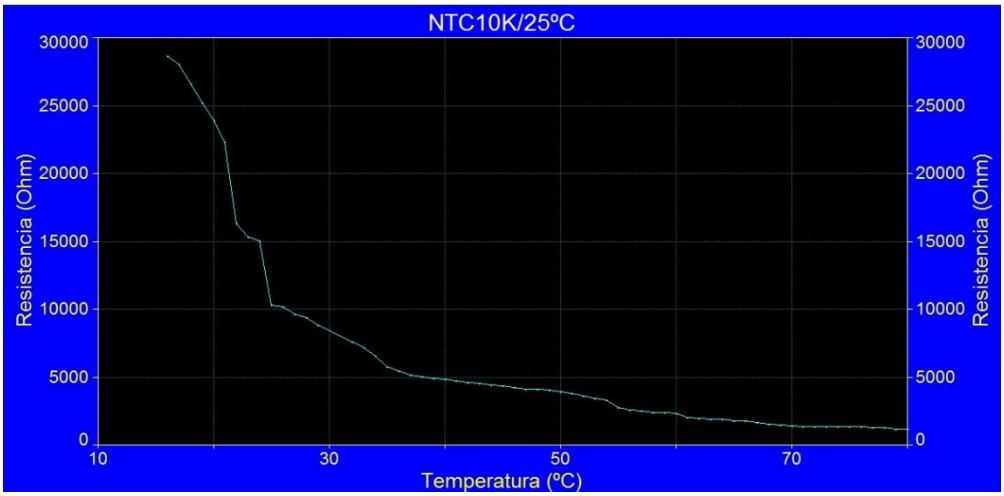

Figura 25A: *Sensor NTC. Curva de calibración Resistencia - Temperatura*

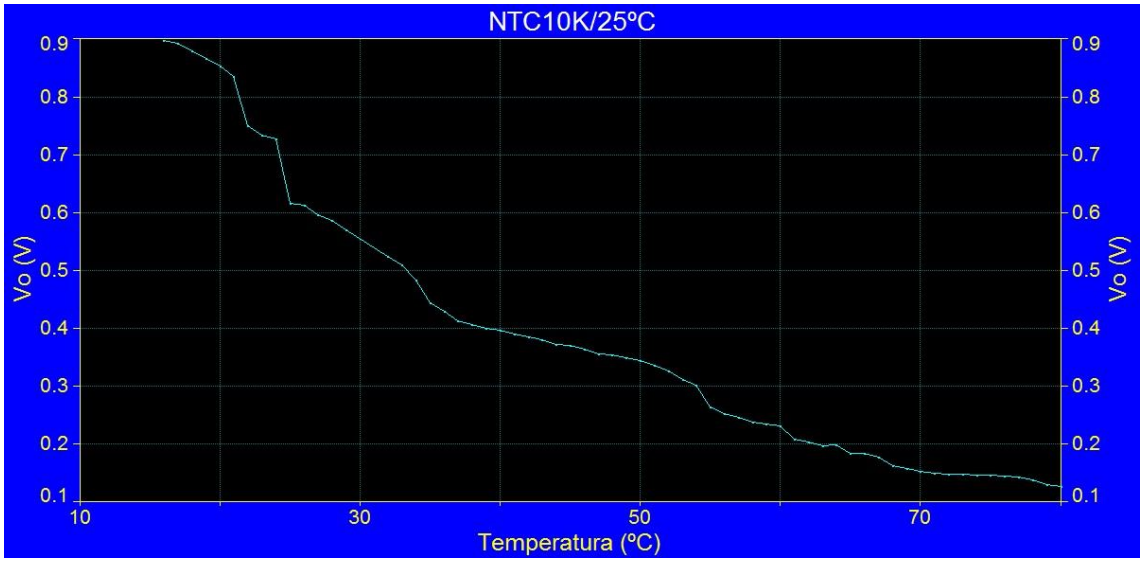

Figura 25B: *Sensor NTC. Curva de calibración Voltaje de salida - Temperatura*

3.1.3. Actuadores

Un actuador es un dispositivo con la capacidad de transformar energía (en nuestro caso, energía eléctrica) en la activación de un proceso con la finalidad de generar un efecto sobre un proceso automatizado. Su función en un sistema domótico será la de cambiar la situación de una vivienda tras un evento, ocasionado al realizar una lectura de un sensor que debe ser tratado. Los utilizados en nuestro proyecto son:

3.1.3.1. Relé de dos canales.

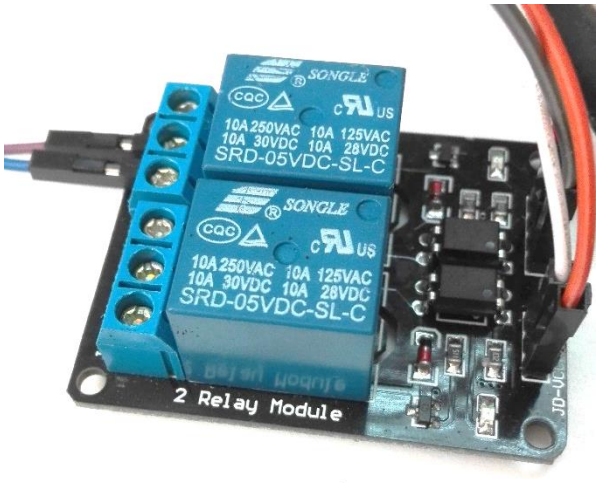

Figura 26: *Módulo de 2 Relés*

Se trata de un módulo que contiene dos relés capaces de soportar tanto 10A a 250V AC como 10A a 30V DC opto acoplados a un circuito de control.

Características y acondicionamiento:

Puede ser utilizado para controlar varios aparatos que requieren alto consumo de corriente. Se controla mediante señales digitales de 5V o 3,3V a través de un microcontrolador, en este caso el XBee. Tiene la capacidad de aislar eléctricamente la etapa de potencia de la etapa de control, tal que, cualquier tipo de anomalía que se produzca en la etapa de potencia no se transmitirá a la etapa de control.
Normalmente, entre VCC y JD-VCC existe un puente “jumper” para alimentar el circuito de control e inductores de los relés a la misma tensión. En nuestro caso, hemos aprovechado esta capacidad para aislar eléctricamente ambas etapas, alimentando los inductores a 5V y el circuito de control a 3,3V. [14]

Especificaciones:

- Relé soporta alto consumo, AC250V 10A, DC30V 10A
- 2 testigos LED para indicar cuando los relés están encendidos
- Funciona con señales de nivel lógico de dispositivos de 3,3 V o 5V.
- Circuito de aislamiento óptico.

Conector de 4 pines:

- **GND**: Tierra (0V).
- **N1:** Controla el relé 1. El relé se accionará cuando esta entrada esté por debajo de 2.0V.
- **IN2**: Controla el relé 1. El relé se accionará cuando esta entrada se encuentre por debajo de 2.0V.
- **VCC:** Alimentación de acopladores ópticos (5V - Misma fuente o fuente aislada).

Conector de 3 pines.

- **JD-VCC**: Alimentación Relés (5V).
- **VCC:** Alimentación de acopladores ópticos. Conexión común al pin Vcc del conector de 4 pines.
- **GND**: Tierra (0V). Conexión común al pin GND del conector 4 pines.

Los relés disponen de 3 conectores de salida cada uno (Interruptor):

- **NC:** Normalmente cerrado.
- **COM:** Común (Siempre se utiliza).
- **NO:** Normalmente Abierto.

3.1.3.2. Leds.

Los LEDs son dispositivos opto electrónicos (Light Emitting Diode) que se comportan igual que un diodo de silicio o germanio. Se produce la emisión de luz cuando es atravesado por una corriente.

Pueden ser construidos con diversos tipos de materiales, lo que produce la radiación de energía en un amplio espectro de longitud de onda: Luz roja, verde, amarilla, azul, naranja o infrarroja (no visible). Tienen un voltaje de operación desde 1,5V a 2,2V aproximadamente, y la gama de corrientes va de 10 mA a 40 mA dependiendo del color del Led. [24]

Especificaciones:

- **Tensión Umbral:** 1.65V.
- **Voltaje disrupción:** 5V.
- **Capacitancia:** 35 pF.
- **Potencia disipada:** 180 mW.

Los Leds representan actuadores en el montaje del sistema. En cuanto a la programación se han implementado para indicar un estado en concreto dependiendo del sensor y de diferentes colores para evitar una posible confusión en los diferentes programas.

3.1.3.3. Ventilador DC FAN

Figura 27: *Ventilador.*

Un ventilador es una máquina hidráulica que incrementa la energía cinética del aire. Dispone de un rotor con aspas accionado habitualmente por un motor eléctrico. En nuestro sistema, ha sido implementado como actuador para la disipación de calor y pueden ser controlados desde un procesador Xbee.

Especificaciones:

- **Voltaje alimentación:** 12V DC.
- **Consumo**: 0.1A

3.1.4. Asignación de Pines. XBee

En la tabla 10 se muestra como se han conectado los diferentes componentes electrónicos del sistema a los pines del módulo Xbee remoto:

Pin	Nombre	Descripción	Conexionado
1	VCC	Alimentación	Fuente de Tensión
2	DOUT	Salida de datos UART	USB
3	DIN/CONFIG	Entrada de datos UART	USB
4	DIO12	Entrada digital 12	Sensor PIR
5	RESET	Reseteo del módulo	Sin asignar
6	PWM0/RSSI/DIO10	Salida digital 10	Led Verificación (PIR y LDR)
7	PWM/DIO11	Salida digital 11	Led Lámpara (LDR) y led Detección (PIR)
8	RESERVADA	No conectar	Sin asignar
9	DTR/SLEEP_RQ/DIO8	Control de hibernación por pin / E/S digital 8	Sin asignar
10	GND	Tierra	Fuente de tensión
11	DIO4	Salida digital 4	Ventilador
12	CTS/DIO7	Salida digital 7	Led Calefactor
13	ON/SLEEP/DIO9	Indicador de estado del módulo / E/S digital 9	Sin asignar
14	RESERVADA	No conectar	Sin asignar
15	ASSOCIATE/DIO5	Salida digital 5	Led Confort (AMT1001 y NTC)
16	RTS/DIO6	Control de flujo Request-to-send / E/S digital 6	Sin asignar
17	AD3/DIO3	Entrada analógica 3	Sensor NTC
18	AD2/DIO2	Entrada analógica 2	Sensor LDR
19	AD1/DIO1	Entrada analógica 1	Sensor AMT1001
20	AD0/DIO0/	Entrada analógica 0	Sensor ACS712

Tabla 10. *Conexiones de los dispositivos en el XBee.*

En la Figura 28 se puede ver el esquema completo del sistema y el conexionado de los diferentes componentes electrónicos (sensores, actuadores, módulos de comunicación inalámbricos, fuente de alimentación,…). En la Figura 29 se puede observar su distribución física.

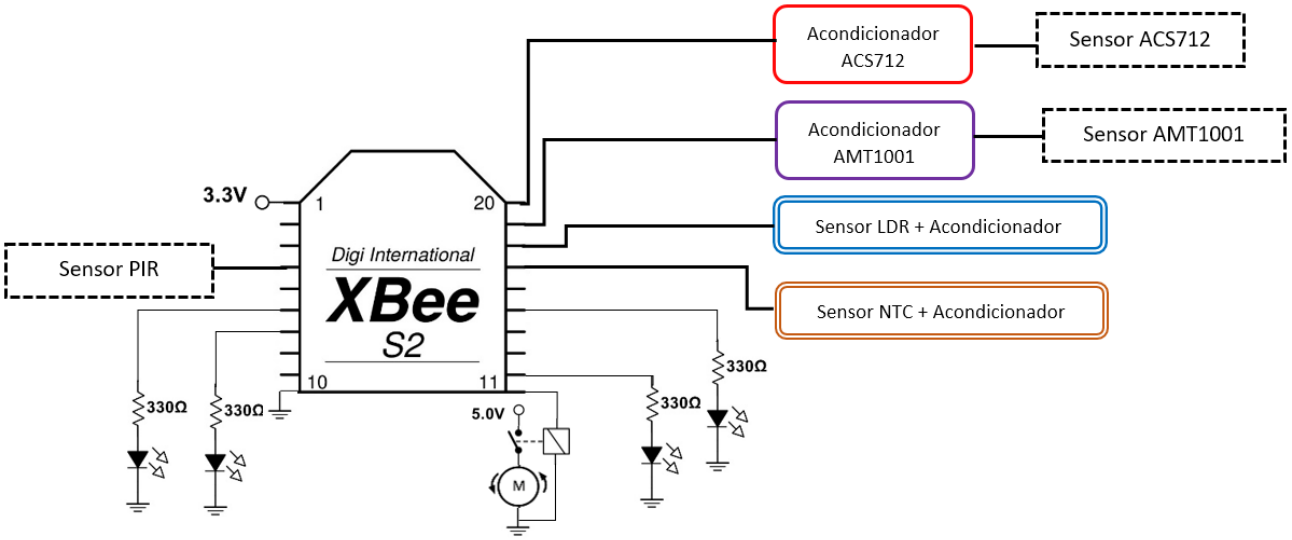

Figura 28. *Esquema completo del Sistema Implementado*

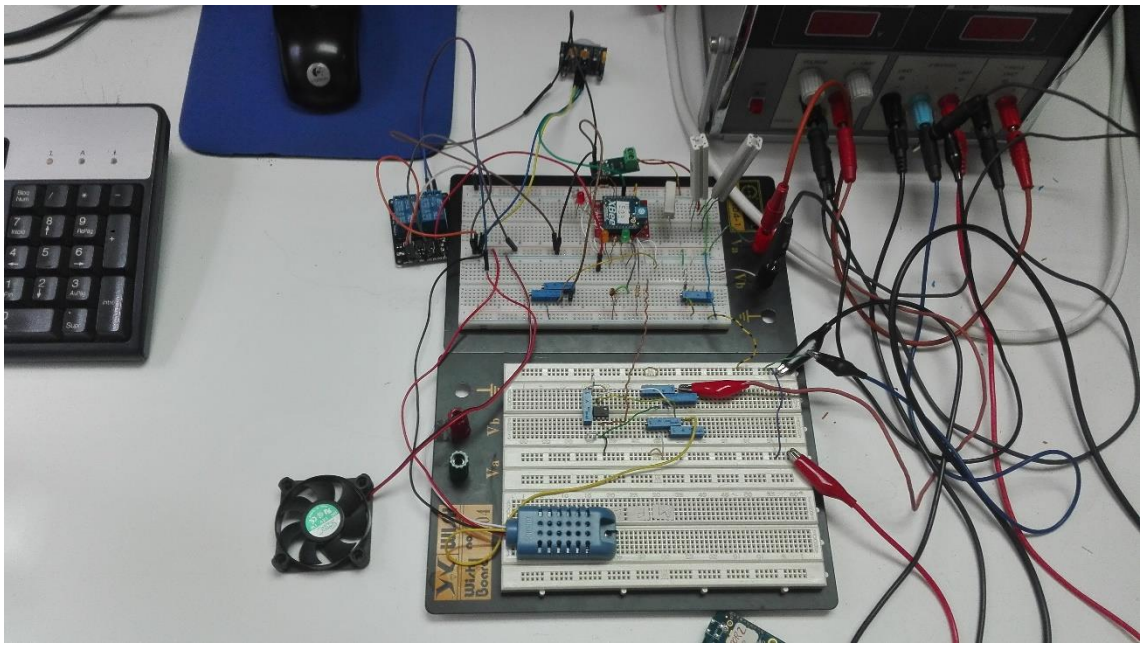

Figura 29. *Montaje final del sistema*

CAPITULO 4. PROGRAMACIÓN.

4.1. INTRODUCCIÓN

En esta fase se describe la forma en que se comunican el hardware y software del sistema, especialmente los módulos XBee (Coordinador y Dispositivo final), los sensores (PIR, NTC, ATM1001, etc.) y los actuadores (Leds y relés).

Nuestro sistema está dividido en programas que tratan de interpretar las señales que proceden de los sensores. Cada clase la hemos vinculado con cada sensor tratando de ver el funcionamiento de cada uno de forma individual. En conjunto todos nos proporcionan información sobre el entorno que, dependiendo de la respuesta, se actuará sobre el mismo ya sea controlando la temperatura o indicando la detección de una presencia, entre otras funciones.

Cada clase, compuesta por los ficheros (.hpp y .cpp), contiene los atributos y métodos necesarios que intervendrán en la configuración y desarrollo de los subsistemas (sensor + actuador). Estas clases instanciarán objetos durante la ejecución de los programas.

4.1.1 Programa y Estructura

Para todas las clases que han sido implementadas con los sensores (*ControlIliminacion, LDRsensor, PIRsensor, HumiditySensor, NTCsensor*), hemos tenido como referencia la clase *Gestion1* para su desarrollo. Las clases son creadas a partir de varios ficheros (.hpp y. cpp) y heredan de *Dialogo_API* todos los atributos y métodos necesarios. En ellas se modificarán los métodos *respuestaATremota()* y *recepcionIOremota*(), donde se desarrollará el código implementado.

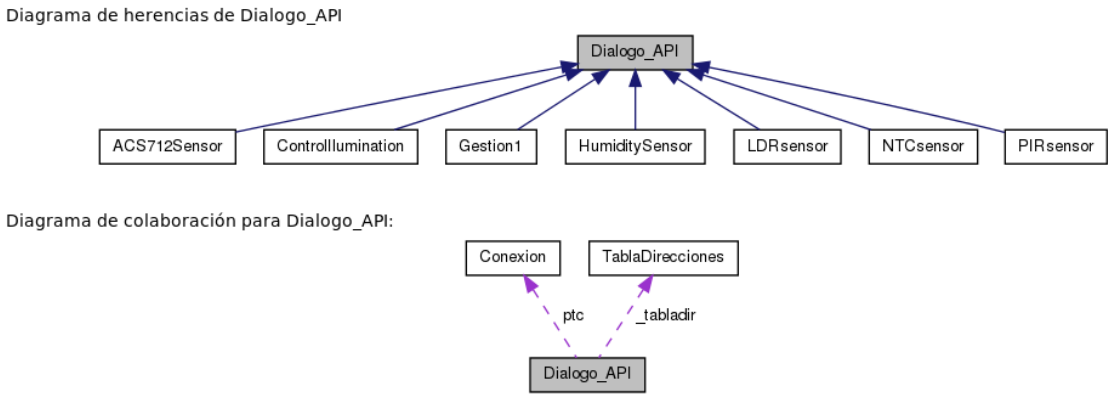

Figura 30. *Herencia. Dialogo_API*

Los ficheros utilizados en las clases son los siguientes:

fichero.hpp

Contiene la declaración de las clases cuyos atributos y métodos podrán ser públicos, privados o protegidos. Al comienzo del fichero se incluirán una breve descripción del sensor y condiciones que implementan.

fichero.cpp

Contiene la definición de los métodos declarados en el (.hpp), es decir, desarrolla cada uno de los métodos de acuerdo al programa en cuestión. También se han añadido algunas funciones complementarias que han sido necesarias de cara a la ejecución del programa. Se ha dividido en zonas muy bien diferenciadas que posteriormente han sido empleadas en el resto de clases. Básicamente la estructura su es la siguiente:

- Inclusión de las Declaraciones de otro fichero y Definición de Macros.
- Constructor del sensor (Inicialización de Atributos).
- Configuración de Pines del Dispositivo Remoto.
- Confirmación de la Lógica Implementada.

test_fichero.cpp

Para poder crear un programa ejecutable se requiere la función *main()*. Los ficheros *test_programa.cpp* contienen esta función necesaria para crear el programa en sí. Tiene la misma estructura para cada una de las clases implementadas. Por ejemplo en *test_gestion1.cpp* se crea la clase conexión y se conforma utilizando los métodos que tiene, se configura el puerto USB y los baudios.

Además implementa varios hilos, concretamente dos, que se utilizan para la recepción y envío de datos. En el *main ()* se desarrolla el hilo principal y una vez que tiene configurada la clase en cuestión, concluye su ejecución. Sin embargo el hilo que realiza todo el trabajo es el de recepción. Siempre está escuchando lo que venga de conexión y en base a este se pueden enviar datos. Cuando recibe algo, llama a su método y toma una serie de decisiones. Envía mensajes y como consecuencia se recibe otro paquete.

En cuanto a los nodos, para establecer la conexión, envía un mensaje para saber los nodos que *Dialogo_API* va identificando y anota sus direcciones en la *Tabla de Direcciones*. Envía el mensaje y transcurrido un tiempo, en caso de no encontrar algún nodo, se queda en un bucle mientras la tabla esté vacía. Cuando deje de estar vacía sale del bucle y establece la comunicación con el nodo que ha encontrado, en nuestro caso, el E09.

Una vez teniendo todos estos ficheros el ***makefile*** utiliza los ficheros de la clase en cuestión (*clase.hpp, clase.cpp y test_clase.cpp*), la clase *Dialogo_API* y la clase *Conexion,* y los compila creando un programa ejecutable.

En cuanto al contenido de las clases, los métodos y atributos heredados de *Dialogo_API*, son comunes a todas sus clases derivadas. La clase de la que partimos, *Gestion1,* tiene los siguientes:

- ***_nameNode:*** Almacena el nombre del nodo remoto.

- ***Gestion1 (Conexion *cp):*** Constructor de la clase a partir del puntero *cp.

- ***configura(std::string nombreNodo):*** Configura los pines del módulo XBee remoto para la conexión de los actuadores y sensores. Estos pueden ser configurados como entradas o salidas digitales, o como entradas analógicas.

- ***respuestaATremota(Frame97 f97):*** Método que realiza la confirmación de la configuración tras el envío de comandos AT para configuración al nodo remoto.

- ***recepcionIORemota(Frame92 f92):*** En caso de que el frame que reciba es de tipo 92 se ejecuta este método. En él se evalúa si hay datos disponibles en los pines configurados como entrada y además, se desarrolla la lógica del programa.

Cabe destacar que hay una función utilizada por algunos de los métodos anteriormente mencionados:

- **ComandoATremoto ()**. Cuando se ejecuta esta función enviamos al nodo remoto comandos AT para la configuración de los pines. Como respuesta el hilo de recepción va a recibir un mensaje de confirmación por parte del nodo remota para la confirmación de la configuración.

De cara a optimizar el código fuente de las clases hemos diseñado algunas funciones auxiliares que ayudan a reducir el volumen de código y mejoran su funcionamiento:

- ***LogicChange ():*** Su función es optimizar el envío de comandos AT al nodo de destino. Los parámetros que se le pasan son comandos AT para la configuración y ejecutan la función *comandoATremoto ().*

- ***infoLed():*** Nos da información acerca del estado de algunos sensores y de los actuadores importantes que intervienen en el subsistema. Básicamente muestra por pantalla un mensaje con el estado de las salidas y entradas digitales.

Otra forma de hacer más entendible y práctico el código fuente de las clases fue la utilización de MAC*ROS.* Una de sus funcionalidades relevantes es que ayudan a automatizar tareas o procesos que se realizan con frecuencia. Por ejemplo en la función *comandoATremoto ()* observamos que por cada pin que se configuraba se tenía que ejecutar esta función. Teníamos que pasarle como parámetros comandos AT, sacados de la "tabla de comandos AT" para la configuración y esto era un proceso tedioso. Entonces se ha decidido sustituir algunos de estos comandos AT por MACROS como *ENCIENDE, APAGA, ACTIVA_INV, DESACTIVA_INV, CALEFACTOR, VENTILADOR*,… A continuación se definirán algunas:

```
#define LED_CONFORT "D5"
#define SAMPLE_N "IR7D0"
#define ENCIENDE(X) (X "5")
#define APAGA(X) (X "4")
```

Luego cuando se invoca a la función *comandoATremoto ():*

```
ComandoATremoto (APAGA (LED_CONFORT), nodoRemoto, -1, true);
```

Que equivale a:

ComandoATremoto (“D54”nodoRemoto, -1, true);

Por otra parte, como se ha podido deducir todos los ficheros de nuestro sistema se han realizado en **C++.** Para su desarrollo hemos utilizado un IDE (Entorno de Desarrollo Integrado) llamado **QT creator**. Un entorno de programación multiplataforma que permite búsqueda rápida por el código, resalta la sintaxis, tiene la función de autocompletado y el plegado del código, entre otros.

4.1.2. Primer Programa.

Para entender la comunicación y el funcionamiento de los XBee hemos decidido realizar un programa para el control de iluminación de un establecimiento. Básicamente se trata de realizar un programa que permita apagar o encender las luminarias de un establecimiento dependiendo del tiempo de pulsación de los pulsadores. Estas pulsaciones pueden ser clasificadas en:

- **Pulsación corta:** Cada vez que se realice esta acción se van encendiendo las luminarias (Leds) de forma secuencial hasta que se alcance todo el encendido del establecimiento. Una vez se haya iluminado todo, si se sigue pulsando se van apagando de forma secuencial igualmente.

- **Pulsación larga**: En este caso se procederá al encendido o apagado total de las luminarias.

- **Pulsación fuera de rango**: Se producirá en caso de que se exceda del tiempo de pulsación larga, apangando las luminarias principales y encendiéndose a su vez un Led auxiliar de testigo.

Para implementar esta lógica se disponen de leds y pulsadores que simulan el comportamiento de las luminarias e interruptores respectivamente**.** Todos ellos se conectarán al XBee remoto para observar el comportamiento de forma visual.

CONEXIONES			
Componentes	**Pines XBee Asociados**	**Descripción**	**Representa**
Pulsador	D0	Entrada Digital	Pulsador
Led Amarillo	D1	Salida Digital	Lámpara
Led Verde	D2	Salida Digital	Lámpara
Led Rojo	D3	Salida Digital	Testigo de error

Tabla 11: *ControlIlumination. Conexiones y descripción de componentes*

Posteriormente se prosigue con la creación del programa que implementa la clase **“ControlIlumination”**. Los atributos propios de la clase son los siguientes:

- ***_actualstate D1, D2 y D4:*** Almacenan el estado de los leds (Encendido o apagado). Necesario para saber en todo momento como se encuentran las salidas.
- ***_VinXBeeAct:*** Almacena el estado actual de la entrada D0 al XBee.
- ***_VinXBeePre:*** *Almacena el estado anterior de la entrada D0.*
- ***_t_inicial****:* Almacena el instante inicial en el que empieza el flanco de subida.
- ***_t_final****:* Guarda el instante final en el que se produce el flanco de bajada.

- ***_timepulse:*** Almacena el tiempo de duración del pulso.

La lógica final que se ha implementado en el programa es la siguiente. Cada vez que se accione el pulsador se producirá un pulso invertido que configurará los pines del XBee dependiendo del tiempo de pulsación. El comando IC nos advierte de cambios en el pin configurado (IC1 -> $0x01_{16}$ -> entrada D0 del XBee). Luego si el estado anterior estaba en alta y pasa a baja, se ha accionado el pulsador y se registra el tiempo. Cuando se suelte, se vuelve a registrar el tiempo y se calcula el tiempo de pulsación.

El tiempo de pulsación se ha deducido a partir de la función *gettimeofday()*[1] *que,* como se puede ver en la Figura 31, registra el tiempo cuando se pulsa y cuando se deja de presionar el pulsador. De esta forma se calcula la duración del pulso, que es la diferencia de tiempos (***_t_inicial y _t_final****),* y se clasifica el tipo de pulsación.

- **Pulsación corta:** (tiempo menor de 500ms). Pueden darse diversos casos:

	Apaga	***Enciende***
D1 y D2 apagados	D2	D1
D1 encendido y D2 apagado	ninguno	D1 y D2
D1 y D2 encendidos	D2	D1
D1 apagado y D2 encendido	D1, D2	ninguno

Tabla 12: *ControlIlumination. Lógica Implementada Pulsación Corta.*

- **Pulsación larga**: Se produce cuando el tiempo de pulsación esta entre 500ms y 1,5s. La lógica implementada muestra si ambos leds están apagados, ambos se encenderán. En otro caso (alguno de los leds encendidos) se apagarán todos.

- **Pulsación fuera de rango**: Se producirá en caso de que se exceda el tiempo de pulsación larga, decir, cuando el tiempo supere 1,5s. En tal caso se encenderá D3, que indicará *"FUERA DE RANGO*".

En cualquier tipo de pulsación la lógica implementada en la programación es muy parecida**.** Como se ve en la Figura 32, se actualizarán los estados de las variables, se encenderán o apagarán los leds correspondientes y se mostrará por pantalla la información más relevante (Tiempo de pulso, estado de las variables, etc.).

En la Figura 33, se muestra un diagrama en Grafcet que explica de forma visual la lógica llevada a cabo por este programa. Aunque el nombre de las variables es diferente la lógica es la misma.

[1] ***gettimeofday()*** *es una función que se utiliza para medir el tiempo que tarda en ejecutarse cierta operación. Esta función se encuentra en la librería* ***sys/time.h****.*

```
if ((_VinXBeePre == true) && (_VinXBeeAct == false)) { //FLANCO

  std::cout << " ---------------------------------" << std::endl;
  std::cout << " FLANCO DE BAJADA (TRUE -> FALSE). " << std::endl;
  std::cout << " ---------------------------------" << std::endl;

  std::cout << "[Gestion1::recepcionIORemota] Se ha accionado el pulsador "
               "por primera vez. " << std::endl;

  //Registro del tiempo.
  gettimeofday(&t_inicial,NULL);
  std::cout << "Guardamos el tiempo inicial ... " << std::endl;

}
 else if ((_VinXBeePre == false) && (_VinXBeeAct == true)){  // FLANCO

  std::cout << " ---------------------------------" << std::endl;
  std::cout << " FLANCO DE SUBIDA (FALSE -> TRUE). " << std::endl;
  std::cout << "          PULSO COMPLETADO         " << std::endl;
  std::cout << " ---------------------------------" << std::endl;

  std::cout << "[Gestion1::recepcionIORemota] Ha pasado de: "
           << _VinXBeePre << " a " <<_VinXBeeAct << ". Se ha"
          " producido un flanco." << std::endl;


  //Registro del tiempo. Calculo del tiempo de pulso.
  gettimeofday(&t_final, NULL);
  std::cout << "Guardamos el tiempo final ... " << std::endl;
  std::cout << "Calculando el tiempo de pulso ... " << std::endl;
```

Figura 31: *ControlIlumination Cálculo del tiempo de pulso.*

```
if ((_timepulse > 0) && (_timepulse <= 500) ){ // PULSACIÓN CORTA

    std::cout << " ---------------------------------" << std::endl;
    std::cout << " HA ENTRADO EN  PULSACIÓN CORTA. " << std::endl;
    std::cout << " Encendemos LED_AMARILLO.        " << std::endl;
    std::cout << " ---------------------------------" << std::endl;

  if (_actualstateD1 == false && _actualstateD2 == false &&
      _actualstateD4 == false) { // D1 y D2 Apagados

    _actualstateD1 = ! _actualstateD1; // Se invierte el estado
    _actualstateD2 = false;

    LogicChange(ENCIENDE(LED_AMARILLO) ,APAGA(LED_VERDE) ,
                APAGA(LED_ROJO) ,nodoRemoto);

    //InfoLed();
```

Figura 32: *ControlIlumination. Lógica Pulsación Corta.*

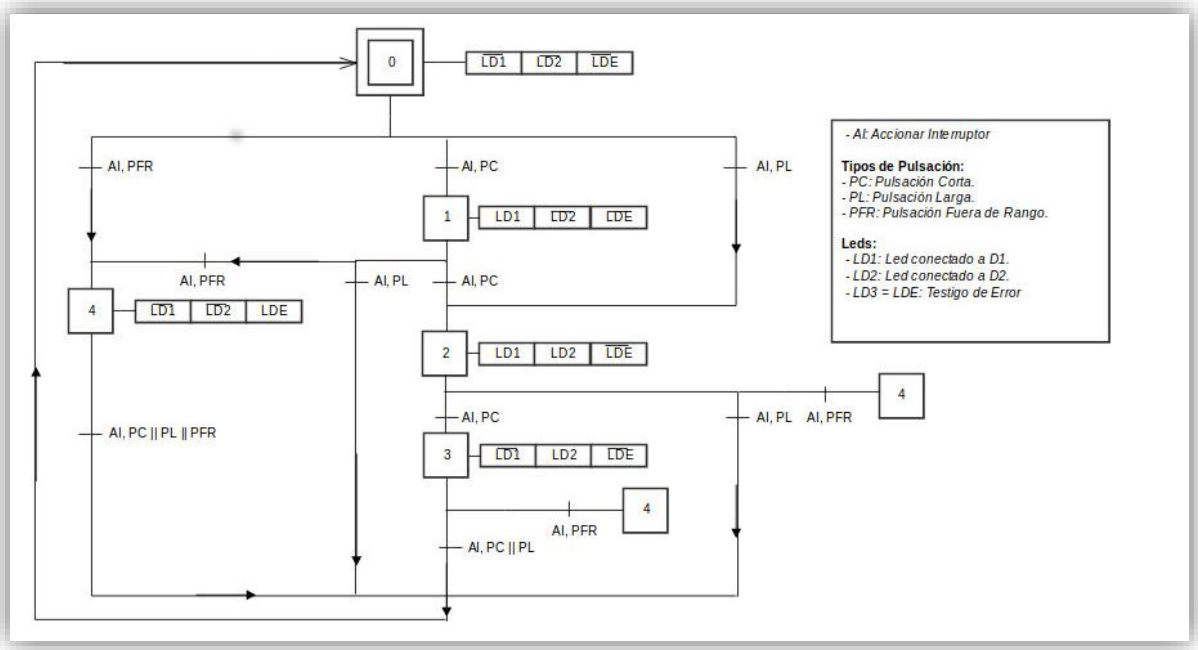

Figura 33: *ControlIlumination. Lógica del programa en Grafcet.*

4.1.3. Sensor PIR (HC - SR501).

Después de realizar la prueba anterior con el control de la iluminación, se ha pasado de lleno a programar el resto de los sensores utilizados en el proyecto. El primer sensor programado ha sido precisamente el sensor de presencia PIR, cuyo funcionamiento se explica en una sección del capítulo anterior. El objeto del programa es la activación automática de luminarias cada vez que detecte la presencia de una persona u objeto al entrar en una vivienda o local.

El subsistema ha requerido el uso del Sensor HC-SR501, y dos leds (naranja y Violeta). Se ha creado una clase particular denominada "**PIRsensor**". Este programa ha configurado tres de los pines del XBee que se describen a continuación:

CONEXIONES PIRsensor			
Componentes	**Pines XBee Asociados**	**Descripción**	**Representa**
Sensor PIR	D12	Entrada Digital	Sensor de presencia
Led Naranja	D11	Salida Digital	Verificación de Presencia
Led Violeta	D10	Salida Digital	Luminaria

Tabla 13: *PIRsensor. Conexiones de los componentes.*

Los atributos propios de esta clase son:

- ***_VinXBeePre:*** Indica el estado anterior de la salida del sensor PIR.
- ***_VinXBeeAct***: Verifica el estado actual de la salida del sensor PIR.
- ***_LedVerif:*** Indica cuando se produce la detección.

- ***_LedLamp:*** Indica el estado del led que representa la luminaria.
- ***_Tpulse:*** Almacena el tiempo en el que salida del sensor está a alta.
- ***_TDelay;*** Tiempo de retardo a la desconexión del led luminaria.
- **t_inicial**: Registra el tiempo inicial cuando el sensor detecta presencia (flanco subida).
- ***t_final***; Registra el tiempo final (flanco bajada).

La lógica que se pretende implementar es la que se puede ver en la Figura 35. Cada vez que algún cuerpo entre dentro del espectro del PIR, este activará el led de verificación de presencia (***_LedVerif***) y, al mismo tiempo, un led que represente la luminaria (***_LedLamp***) durante un tiempo (configurado de forma manual). Una vez se cumpla este tiempo se apaga el led de verificación y después de un tiempo prudencial, el Led que representa una luminaria.

Como se puede ver en la figura 34, si un cuerpo entra en el rango de detección del PIR, el pin D12 del XBee pasa de false a true (flanco de subida), el tiempo se registra, se actualizan las variables y se activan D11 y D10. Una vez concluido el tiempo de activación D12 pasa de true a false (flanco de bajada) y se vuelve a registrar el tiempo. El pulso se ha completado y se apaga D11. Después de un tiempo prudencial, se apagará D10.

Para el desarrollo del programa hemos utilizado las funciones auxiliares "**gettimeofday()"** para el cálculo del tiempo de activación. Además se ha utilizado el comando AT **"IC"** que nos da información de cuando se producen cambios a la salida del sensor, decir, cuando se detecta presencia (**IC1C00** -> $0x1C00_{16}$ -> pines D12, D10 y D11 del XBee).

```
if((_VinXBeePre == false) && (_VinXBeeAct == true)) {

  std::cout << " ----------------------" << std::endl;
  std::cout << "SE DETECTA UNA PRESENCIA" << std::endl;
  std::cout << " ----------------------" << std::endl;

  //_LedVerif = !_LedVerif;
  //_LedLamp = !_LedLamp;

  gettimeofday(&t_inicial,NULL); //Se Obtiene el tiempo

  LogicChange(ACTIVA(DETECT), ACTIVA(VERIFICACION), nodoRemoto );

  //infoLed();

  }
}

//Se ha producido un flanco,
else if ((_VinXBeePre == true) && (_VinXBeeAct == false)) {

  std::cout << "[PIRsensor::recepcionIORemota] Ha pasado de:"
            << _VinXBeePre << "a" << _VinXBeeAct
            << ".Se ha producido un flanco" << std::endl;

  //_estadoLD = !_estadoLD;
  //_LedVerif = !_LedVerif;

  gettimeofday(&t_final, NULL); //Se Obtiene el tiempo
```

Figura 34: *PIRsensor. Control de pulsos del PIR*

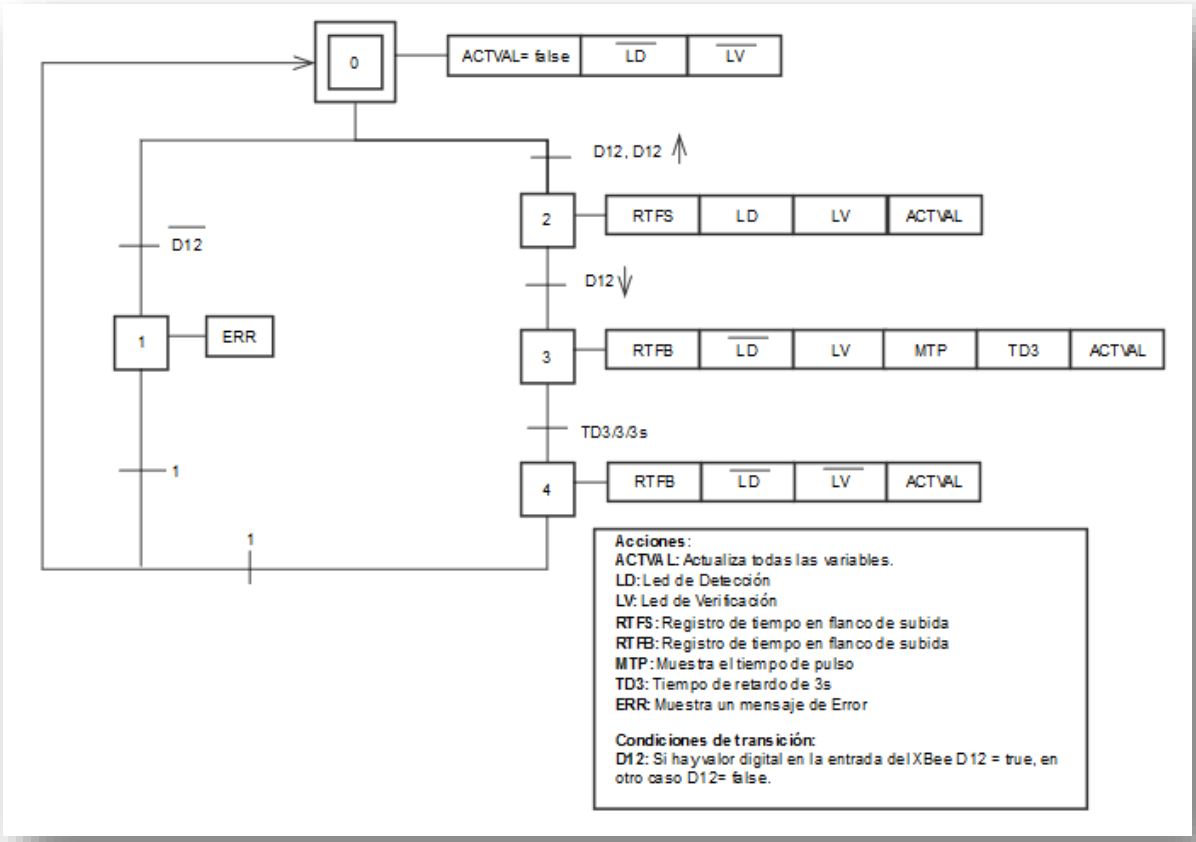

Figura 35: *PIRsensor. Lógica del programa en Grafcet.*

4.1.4. Sensor de corriente (ACS712)

Con el sensor de corriente se pretende estimar el consumo medio de cualquier dispositivo que se conecte a la red, así como el cálculo del consumo medio de un establecimiento o local. Además con esta aplicación se podrá saber cuándo un dispositivo estará conectado o estropeado, o cuándo consumirá más, entre otras funciones.

Para tal fin se ha creado una clase denominada **"ACS712Sensor"** en la que se ha utilizado sólo uno de los pines del XBee:

CONEXIONES ACS712Sensor			
Componentes	**Pines XBee Asociados**	**Descripción**	**Representa**
Sensor ACS712	D0	Entrada Analógica	Amperímetro

Tabla 14: *ACS712Sensor. Conexiones de los componentes.*

Atributos y métodos propios de la clase ACS712Sensor:

Atributos:

- _**Current;** Almacena el valor de la corriente que circula por el dispositivo a medir.
- _**Power;** Guarda la potencia que consume el dispositivo a medir.
- _**VoltajeSensor;** Almacena lo que hay a la entrada del XBee.

Métodos:

- **get_current(double VinXbee):** Función que devuelve el valor de la corriente (A).
- **get_power(double VinXBee):** Función que devuelve el valor de la potencia (W).

A groso modo, como se observa en la Figura 37, el programa calcula el consumo en corriente y en potencia del dispositivo conectado al sensor realizando la llamada a funciones auxiliares para el cálculo de los valores. Tras hacer diversas pruebas con el sensor, éste no funciona de acuerdo al comportamiento descrito en las hojas características. Por tanto nos dificulta el proceso de medición del consumo.

Hemos decidido implementar sólo el comportamiento en continua. Para ello se han utilizado dos funciones auxiliares, Figura 36: *get_current*() para la obtención de la corriente que circula por el dispositivo eléctrico conectado y *get_power()* para el cálculo de la potencia consumida. Esta última depende de la anterior pues para el cálculo de la potencia es preciso tener el valor de la corriente instantánea, es decir, la corriente de pico obtenida a partir de la función *get_current*() .

Además, como se trata de una señal analógica, para obtener la intensidad en cada instante de tiempo se precisa la utilización de una función de muestreo. Esto se hace a través el comando AT **"IR"**, configurado para un tiempo de muestreo de dos segundos (IR7D0 -> $0x7D0_{16}$ -> 2000ms = 2s).

Tanto para dispositivos que operen en corriente alterna, como en corriente continua no se han obtenido los valores deseados con este sensor. Por tanto no es fiable su uso para tal fin.

```
// Observamos si hay información en D
if (f92.haveAnalog(0)) {

  _VinXBeeC = f92.getAnalog(0); // Guarda lo que hay en D0.

  _Current = get_current(_VinXBeeC); // Intensidad en Amperios (A)

  _Power = get_power(_VinXBeeC); //Potencia en Watios (W)

  std::cout << "El consumo del dispositivo es el siguiente: " << std::endl;
  std::cout << "+ Intensidad: " << _Current << " A." << std::endl;
  std::cout << "+ Potencia: " << _Power << " W." << std::endl;

}
```

Figura 36: *ACS712Sensor. Lógica implementada.*

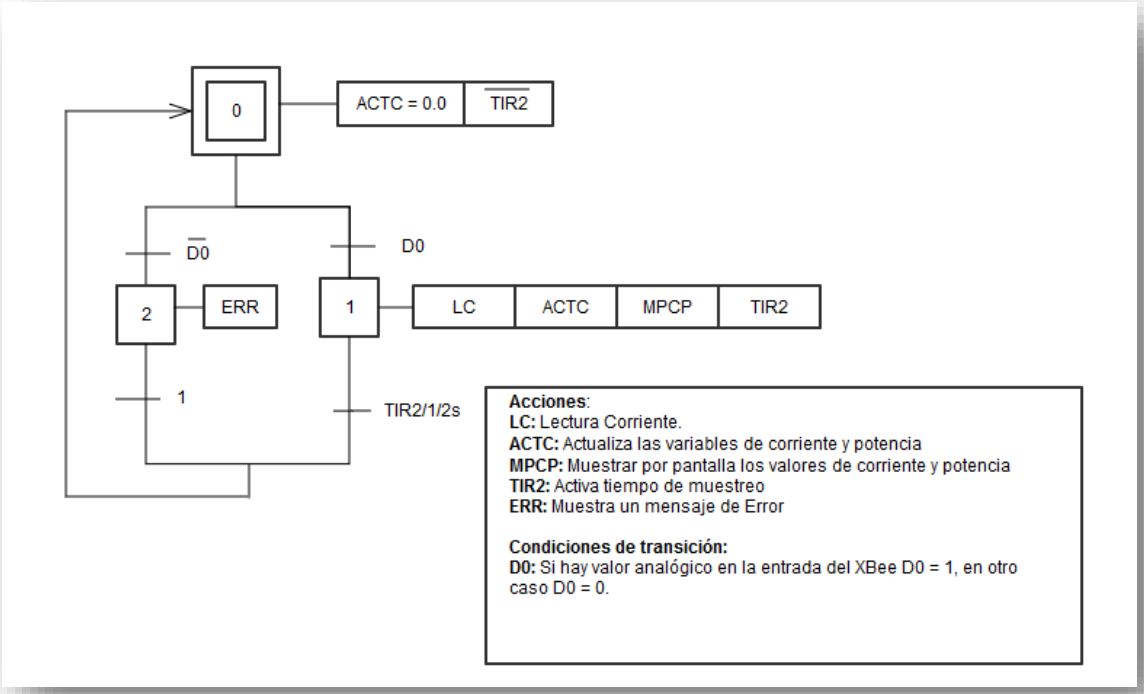

Figura 37: *ACS712Sensor. Lógica del programa en Grafcet.*

4.1.5. Sensor de temperatura y humedad (AMT1001)

Desde el punto de vista del hogar, el sensor ATM1001 es utilizado para monitorizar la temperatura y humedad del recinto. Como consecuencia trata de alcanzar el valor óptimo deseado en caso de que el valor real se desvíe considerablemente. El sensor ha sido configurado únicamente como sensor de Humedad.

Los elementos que se han utilizado para la construcción del circuito han sido un ventilador, el sensor AMT1001 y un led. En el programa implementado denominado como "**HumiditySensor**" ha sido necesaria la configuración de tres pines en el XBee:

CONEXIONES HumiditySensor			
Componentes	**Pines XBee Asociados**	**Descripción**	**Representa**
Sensor AMT1001	D1	Entrada Analógica	Humistor
Ventilador	D4	Salida Digital	Deshumificador
Led Verde	D5	Salida Digital	Testigo de Confort

Tabla 15: *HumiditySensor. Conexiones de los componentes.*

Los métodos y atributos que se han utilizado, a parte de los comunes a todas las clases anteriores son los siguientes:

Atributos:

- ***_ActualHum:*** Almacena la humedad actual del ambiente.
- ***_VinXbeeH:*** Almacena el voltaje de entrada al XBee.
- ***_StateVent:*** Almacena el estado actual del ventilador.
- ***_LEDConfort:*** Nos dice si el nivel de humedad es la apropiada.

Métodos:

- ***get_humidity(float VXBee);*** Función que calcula la humedad a partir del voltaje de entrada al XBee.

La lógica que se desea implementar en el programa, se puede ver en el grafcet de la Figura 39, es la obtención del valor de humedad relativa del recinto y dependiendo del valor preestablecido, tratar de alcanzar ese nivel de humedad. Luego para la obtención de la humedad relativa en sí, la función auxiliar implementada se denomina *get_Humidity()*. Esta función establece una relación entre la humedad ambiental y la tensión a la entrada del XBee.

La lógica implementada en el programa se reproduce en la Figura 38. Si existe un valor analógico en el pin D2, se llama a la función y se transforma el valor de tensión en humedad relativa. Posteriormente se clasifica según las condiciones establecidas y así actuar sobre el medio. Por ejemplo en caso de que la humedad esté comprendida entre el 21% y 50%, la humedad relativa es la óptima para el recinto, luego se activa D5 (***_LEDConfort***). Sin embargo, si el valor está comprendido entre el 51% y el 80% la humedad es elevada y por tanto se activará el pin D4 (***_StateVent***) para tratar de

reducirla. En ambos casos las variables de estado serán actualizadas para imprimir por pantalla mostrar por pantalla el estado actual en todo momento.

Además, hay que tener en cuenta que al ser una señal analógica es necesario realizar una lectura del pin D2 cada cierto tiempo. Para tal fin se ha hecho uso del comando AT IR que permite el muestreo de señales analógicas. IR se ha configurado para que cada tres segundos lea y actualice el valor del pin D2 (IRBB8 -> $0xBB8_{16}$ -> 3000ms = 3s).

```
if (f92.haveAnalog(3)) { //vemos si hay informacion de D3

  std::cout << " ----------------------------" << std::endl;
  std::cout << " SE OBTIENE EL VALOR ANALÓGICO. " << std::endl;
  std::cout << " ----------------------------" << std::endl;

  _VinXBeeT = f92.getAnalog(3); // Se registra el valor de la tensión

  _ActualTemp = get_temperature(_VinXBeeT); //Se obtiene la temperatura

  std::cout << " ------------------------------------------" << std::endl;
  std::cout << "La temperatura ambiente es: " << _ActualTemp
            << " ºC; Valor de tensión asociado: " << _VinXBeeT << " V."
            << std::endl;
  std::cout << " ------------------------------------------" << std::endl;

  //! ======== CONDICIONES PARA LA TEMPERATURA (SIN HISTÉRESIS) =============

  if ((_ActualTemp > 0.0) && (_ActualTemp < CONSIGNA)) {
    std::cout << " ------------------------------------" << std::endl;
    std::cout << " La temperatura por DEBAJO del CONFORT." << std::endl;
    std::cout << " Se acciona el Calefactor." << std::endl;
    std::cout << " ------------------------------------" << std::endl;

    //_actualWormLED = !_actualWormLED;
    //_actualRelVent = false;
    //_actualLConf = false;
    //_Disminuye = false;

    LogicChange(ENCIENDE(CALEFACTOR), DESACTIVA_INV(VENTILADOR),
                APAGA(LED_CONFORT), nodoRemoto);
```

Figura 38: *HumiditySensor. Lógica Implementada.*

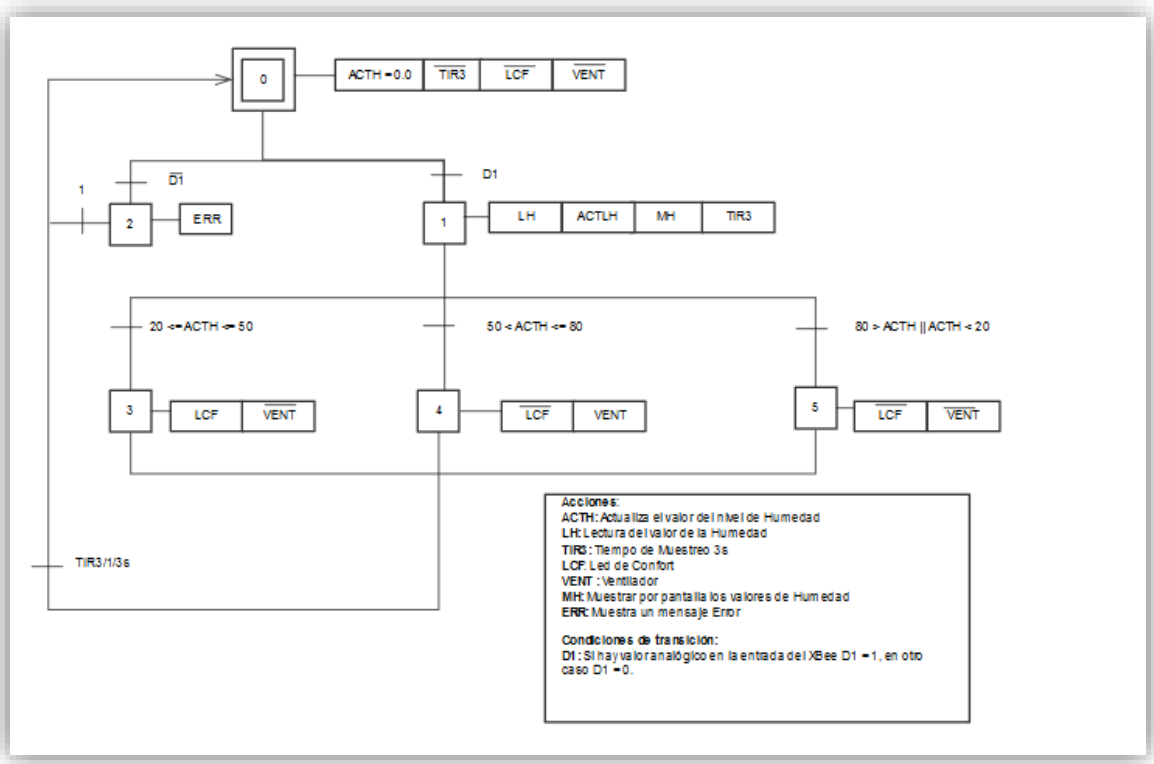

Figura 39: *HumiditySensor. Lógica del programa en Grafcet.*

4.1.6. Fotoresistor LDR.

El sensor LDR es muy utilizado en aplicaciones relacionadas con la iluminación. Es nuestro caso se utilizará para evaluar el nivel de iluminación del ambiente. De esta forma se observará si la luminosidad del ambiente es la adecuada o no para el fin que se busca.

En el subsistema se han utilizado diversos elementos como la NTC y varios leds (Naranja y Violeta). Se ha construido la clase **"LDRsensor"** para programar este sensor. En cuanto a las conexiones se han reservado tres pines del XBee:

CONEXIONES LDRsensor			
Componentes	**Pines XBee Asociados**	**Descripción**	**Representa**
Sensor LDR	D2	Entrada Analógica	Sensor de Luminosidad
Led Naranja	D10	Salida Digital	Led de Verificación
Led Violeta	D11	Salida Digital	Lámpara

Tabla 16: *LDRsensor. Conexiones de los componentes.*

La clase LDRsensor contiene los atributos y métodos necesarios para crear un objeto de clase. A continuación, se exponen los propios de la clase:

Atributos:

- _***VinXBeeL:*** Almacena la tensión de entrada al XBee.
- _***LUX*****:** Almacena la cantidad de luxes que hay en el ambiente.
- _***verificacion:*** Indica el estado del led de verificación.
- _**lampara;** Indica el estado del led que representa la lámpara.
- _***ledConfort:*** Indica el estado del led de confort.

Métodos:

- ***get_lux(float VXBee*****):** Función que calcula la cantidad de luxes medidos.

La implementación de la lógica que se definió anteriormente se ha realizado según se explica en la Figura 41. Se lee si hay algún valor analógico en la entrada D2 del XBee. En tal caso se realiza la llamada a la función *get_lux()* que calcula la cantidad de luxes del entorno a través de una ecuación obtenida a partir del estudio de la LDR. Una vez obtenido el valor se clasifica según las condiciones predeterminadas. En caso de que el valor esté comprendido entre 750lux y 200lux, el nivel de luminosidad será óptimo y se activará D5. EN caso de que esté por debajo de 750lux se activará D10 y D11. En caso contrario, es decir, que el nivel de luminosidad sea excesivo para el local se activará solamente D10.

Como se puede ver en la Figura 40, en cualquiera de los casos los valores de las variables que almacenan el estado de los pines se actualizarán. Se mostrarán los valores por pantalla cada vez que se realice una lectura del sensor. Como se trata de extraer información a partir de una señal analógica, hemos tenido que utilizar la función de muestreo con el uso del comando AT **"**IR**",** configurado para muestrear la señal cada segundo (IR3E8 -> $0x3E8_{16}$ -> 1000ms = 1s).

Como tan sólo se está utilizando un nodo final, la cantidad de pines del XBee se hace insuficiente para la construcción independiente de todos los subsistemas. Por tanto, nos vemos obligados a aprovechar ciertos pines (LEDs) que han sido reservados en otros subsistemas para la ejecución del programa.

```
if (f92.haveAnalog(2)) { //vemos si hay informacion de D1

  std::cout << " ------------------------------" << std::endl;
  std::cout << " SE OBTIENE EL VALOR ANALÓGICO. " << std::endl;
  std::cout << " ------------------------------" << std::endl;

  _VinXBeeL = f92.getAnalog(2); // Se registra el valor de la tensión

  _LUX = get_lux(_VinXBeeL); //Se obtiene el valor en luxes

  std::cout << "El nivel de iluminación es: " << _LUX << "Lux." << std::endl;

  //! ================== CONDICIONES DE ACTUACIÓN ========================

  if (_LUX > 750 && _LUX < 5000){

    std::cout << " ---------------------------------" << std::endl;
    std::cout << " EL NIVEL DE LUMINOSIDAD ES ÓPTIMO " << std::endl;
    std::cout << " ---------------------------------" << std::endl;

    //_verificacion = false;
    //_lampara = false;
    //_ledConfort = !_ledConfort;

    LogicChange(APAGA(LED_VERIFICACION),APAGA(LED_LAMP)
                ,ENCIENDE(LED_CONFORT),nodoRemoto);

    //infoLed();
```

Figura 40: *LDRsensor. Lógica del programa.*

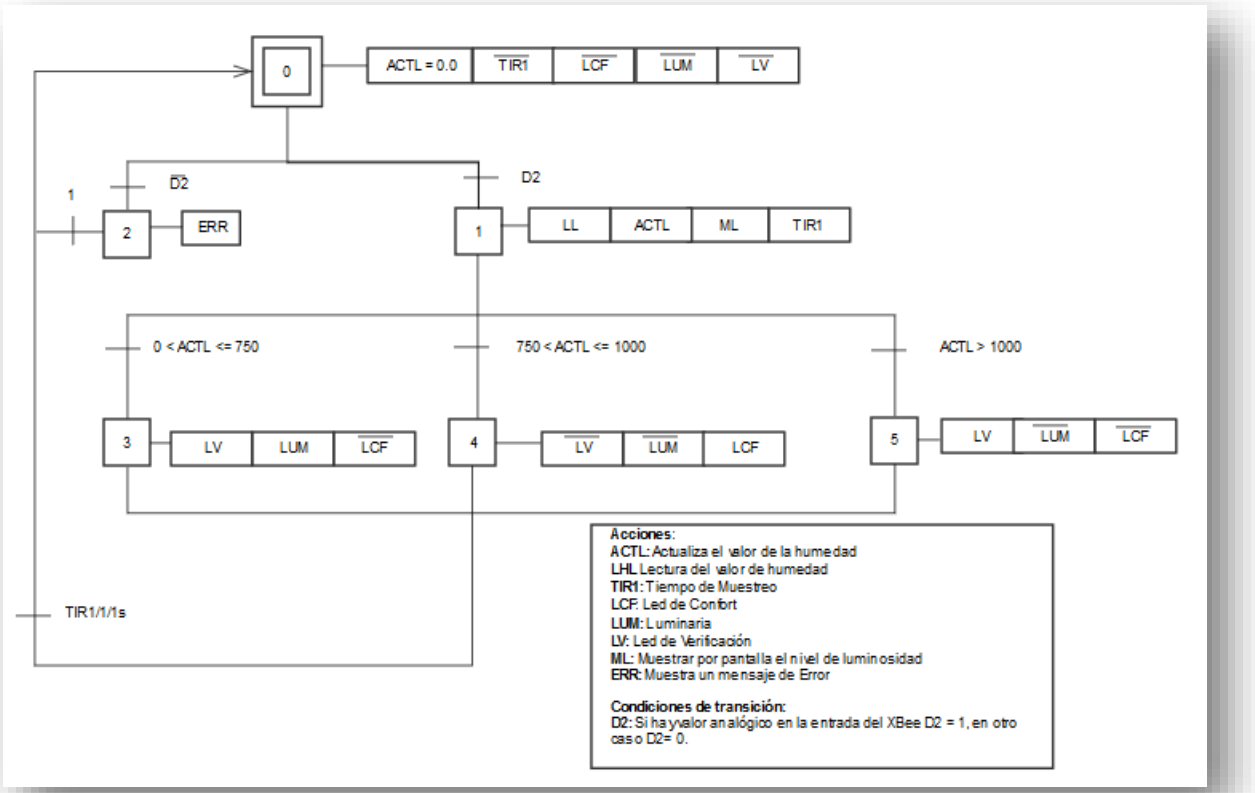

Figura 41: *LDRsensor. Lógica del programa en Grafcet.*

4.1.7. Termistor NTC.

Con el objetivo de controlar la temperatura de un establecimiento o local se ha utilizado el termistor NTC, cuya función es obtener y controlar la temperatura del entorno. Preferiblemente hubiera sido más fácil implementar el sensor de temperatura del módulo AMT1001 (Comportamiento aproximadamente lineal) pero las circunstancias descritas con anterioridad han dado lugar a ser sustituido por la NTC.

En el subsistema se han utilizado diversos elementos como la NTC, Bloque de relés y varios leds (Verde y rojo). La Clase que describe el comportamiento de la NTC se denomina "**NTCsensor**". Se han configurado en este caso 4 pines del XBee:

CONEXIONES NTCsensor			
Componentes	**Pines XBee Asociados**	**Descripción**	**Representa**
Sensor NTC	D3	Entrada Analógica	Termómetro
Entrada IN2 Rele	D4	Salida Digital	Ventilador
Led Verde	D5	Salida Digital	Testigo de Confort
Led Rojo	D7	Salida Digital	Calefactor

Tabla 17: *NTCsensor. Conexiones de los componentes.*

A continuación, se mostrarán los atributos y métodos propios de la clase NTCsensor:

Atributos:

- _***ActualTemp:*** Almacena la temperatura actual del sistema.
- _***VinXBeeT:*** Almacena el valor de tensión del pin del XBee.
- _***actualRelVent:*** Guarda el estado actual del ventilador.
- _***actualWormLED:*** Guarda el estado actual del calefactor.
- _***actualLConf:*** Guarda el estado actual del testigo de confort.

Métodos:

- ***get_temperature(float VTG):*** Función que halla el valor de la temperatura a partir de la tensión de entrada al XBee.

Respecto a la lógica diseñada para este sensor, Figura 43, se han establecido unas condiciones de actuación en caso de que la temperatura se aleje del valor de consigna preestablecido. Como se observa en la Figura 42, en primer lugar se hace una lectura de la entrada D3 del XBee para saber si existe un valor analógico. Esta acción se realiza a través del comando AT IR configurado para un tiempo de muestreo determinado, en este caso dos segundos. En caso afirmativo, se procede al cálculo de la temperatura ambiente haciendo uso de la función auxiliar *get_temperature()* . Posteriormente el valor de temperatura será clasificado según ciertas condiciones:

a) Si la temperatura es inferior al valor de consigna, se activa D7, hasta que la temperatura no alcance la zona de confort (calefactor).
b) Si la temperatura está dentro de los márgenes del confort, se activa D5 (testigo de confort).

c) En caso de que la temperatura sea superior al límite del confort, se activará D4 (ventilador).

Cabe destacar que, en la programación de este sensor, se estimó oportuno añadir la influencia de la histéresis cuando se produjeran transiciones que afectaran a los actuadores. De esta forma, en caso de que la temperatura oscilara entre los valores de consigna predefinidos, la actuación de los sensores fuera más estable [Ver Anexo 7.1.2.].

Tras realizar varias pruebas no se pudo implementar el comportamiento de la histéresis en los programas y por tanto lo hemos dejado para posibles desarrollos futuros.

```
if (f92.haveAnalog(3)) { //vemos si hay informacion de D3

  std::cout << " ------------------------------" << std::endl;
  std::cout << " SE OBTIENE EL VALOR ANALÓGICO. " << std::endl;
  std::cout << " ------------------------------" << std::endl;

  _VinXBeeT = f92.getAnalog(3); // Se registra el valor de la tensión

  _ActualTemp = get_temperature(_VinXBeeT); //Se obtiene la temperatura

  std::cout << " ---------------------------------------------" << std::endl;
  std::cout << "La temperatura ambiente es: " << _ActualTemp
            << " ºC; Valor de tensión asociado: " << _VinXBeeT << " V."
            << std::endl;
  std::cout << " ---------------------------------------------" << std::endl;

  //! ======== CONDICIONES PARA LA TEMPERATURA (SIN HISTÉRESIS) =============

  if ((_ActualTemp > 0.0) && (_ActualTemp < CONSIGNA)) {
    std::cout << " -------------------------------------" << std::endl;
    std::cout << " La temperatura por DEBAJO del CONFORT." << std::endl;
    std::cout << " Se acciona el Calefactor." << std::endl;
    std::cout << " -------------------------------------" << std::endl;

    //_actualWormLED = !_actualWormLED;
    //_actualRelVent = false;
    //_actualLConf = false;
    //_Disminuye = false;

    LogicChange(ENCIENDE(CALEFACTOR), DESACTIVA_INV(VENTILADOR),
                APAGA(LED_CONFORT), nodoRemoto);
```

Figura 42: *NTCsensor. Ejemplo implementación de la lógica.*

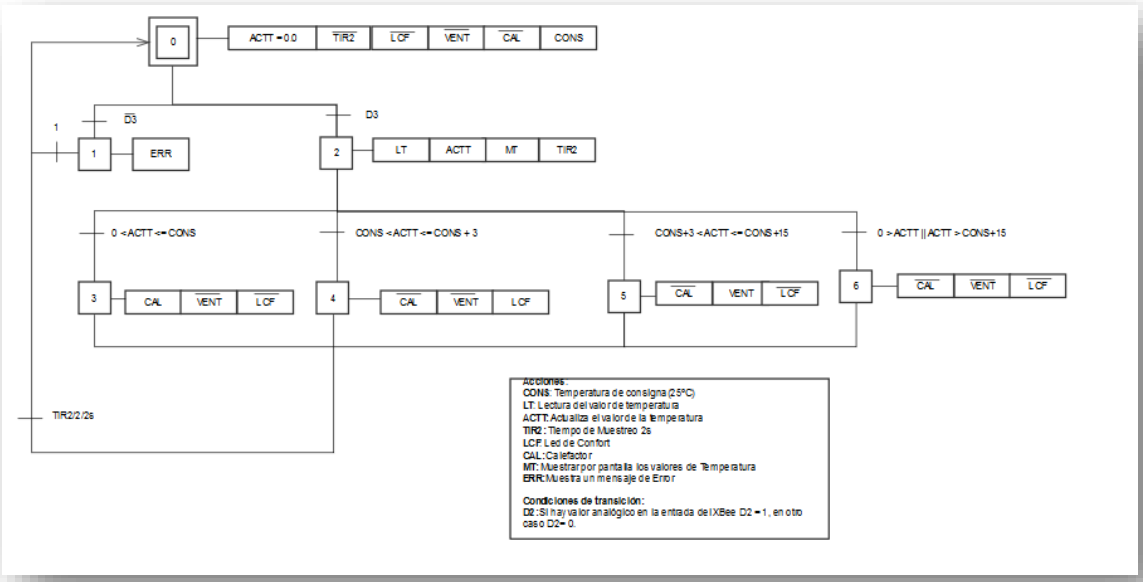

Figura 43: *NTCsensor. Lógica del programa en Grafcet.*

4.2 ACCESO AL REPOSITORIO

Para acceder a la carpeta del repositorio que contiene todas las clases está disponible en el repositorio remoto “GitHub” [25]

CAPÍTULO 5: CONCLUSIONES Y LÍNEAS FUTURAS

5.1. CONCLUSIONES.

- Se ha realizado un montaje electrónico en dos placas de pruebas (Protoboard) integrando varios dispositivos: Sensor PIR, Sensor de Corriente ACS712, Sensor de temperatura y Humedad ATM1001, Sensor de iluminación LDR, Sensor de temperatura NTC, Relé, etc.

- Se ha realizado un estudio exhaustivo previo de cada uno de los dispositivos mediante la calibración y el acondicionamiento de sus señales.

- Todos los sensores han tenido un funcionamiento eficiente, excepto el sensor de corriente ACS712, cuya respuesta no ha sido la idónea según lo indicado en su hoja de características, y el sensor de temperatura y humedad AMT1001 ha servido únicamente para la medición de la humedad.

- Se manejaron diferentes herramientas informáticas para el control de versiones, realización de gráficas, control para procesos secuenciales, la programación en sí, etc.

- Se han desarrollado todos los programas en Lenguaje C++ para modelar el funcionamiento de cada uno de los sensores por separado.

- Se ha implementado un sistema completo de un conjunto de dispositivos configurados e interconectados para el control de una vivienda automatizada (domótica), una sostenibilidad energética y sobretodo económica.

5.2. LÍNEAS FUTURAS

- Sustituir el termistor NTC por el sensor LM35 ya que tiene una salida aproximadamente lineal con la temperatura. Calibrado para cambios de 1ºC, su rango de medición va desde los -55ºC hasta los 150ºC.

- Integrar la influencia de la Histéresis en los códigos de programación de los sensores para mejorar las transiciones lógicas y proteger a los actuadores (Relé, etc.).

- Posibilidad de introducir un segundo módulo XBee como dispositivo remoto con los actuadores y sensores, permitiendo una intercomunicación entre dos dispositivos remotos con el módulo XBee Coordinador; aunque dicho cambio implique realizar algunas variaciones en el código de programación principal.

- Aumentar la red ZigBee introduciendo dispositivos routers que permitieran el envío de comandos AT a distancias más lejanas.

- Integrar en un mismo sistema tanto la tecnología Zigbee como la WI-FI para establecer conexiones inalámbricas a distancias más amplias y aprovechar las características económicas y de consumo de ambas.

- La creación de una aplicación móvil (para Smartphone, tablets y ordenadores) y la inclusión de otros sensores para hacer un sistema domótico mucho más complejo y a la vez completo para mejorar el control de una casa inteligente (Smart Home). Y así establecer una unión entre las nuevas tecnologías y los electrodomésticos.

- Posibilidad de integrar en un mismo sistema, tanto los dispositivos Xbee como por ejemplo la Raspberry pi y el módulo Arduino, para la comunicación.

CAPÍTULO 6: PRESUPUESTO

Para el presupuesto, se realizará un presupuesto particular para cada circuito de cada sensor y sumarlo con otros elementos para una sumatoria neta. Las siguientes tablas muestran los precios:

DESCRIPCIÓN	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
Sensor ACS712	1	13.6 €/u	13.6 €
RESISTENCIAS	2	0.14 €/u	0.28 €
PRECIO TOTAL (EUROS)	13.88 €		

Tabla 18: *Calibración y acondicionamiento del sensor de corriente ACS712.*

DESCRIPCIÓN	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
Sensor ATM1001	1	1.75 €/u	1.75 €
Amplificador Operacional UA741	1	0.562 €/u	0.562 €
Amplificador operacional LM324	1	0.482 €/u	0.482 €
Potenciómetros	5	0.34 €/u	1.7 €
PRECIO TOTAL (EUROS)	4,49 €		

Tabla 19: *Calibración y acondicionamiento del sensor de temperatura y humedad ATM1001*.

DESCRIPCIÓN	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
Sensor LDR	1	1.80 €/u	1.80 €
RESISTENCIAS	2	0.14 €/u	0.28 €
PRECIO TOTAL	2.08 €		

Tabla 20: *Calibración y acondicionamiento de la Fotorresistencia LDR.*

DESCRIPCIÓN	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
Sensor NTC	2	0.03 €/u	1.80 €
RESISTENCIAS	2	0.14 €/u	0.28 €
PRECIO TOTAL	2.08 €		

Tabla 21: *Calibración y acondicionamiento del sensor de temperatura NTC.*

DESCRIPCIÓN	CANTIDAD	PRECIO UNITARIO	PRECIO TOTAL
Leds	4	0.30 €/u	1.20 €
XBee ZNet 2.5	2	21.20 €/u	42.4 €
Ventilador DC 12V	1	3.99 €/u	3.99 €
Sensor PIR	1	0.80 €/u	0.80 €
Relé 2 canales	1	5.90 €/u	5.90 €
Placa Protoboard	2	5.15 €/u	10.30 €
PRECIO TOTAL	64,59 €		

Tabla 22: *Otros elementos*

DESCRIPCIÓN	NÚMERO	TRABAJO/INGENIERO JUNIOR (HORAS)	TARIFA/ HORA aproximado	PRECIO
Mano de Obra	2	200 h	15 €/h	6000 €
Base Imponible I.G.I.C (7%)				420 €
PRECIO TOTAL				6420 €

Tabla 23: *recursos humanos*

Nota: No se han mencionado algunos elementos auxiliares como (Fuentes de alimentación, Multímetros, Cableado, entre otros).

El Presupuesto Final para este proyecto es: **6.507,12 €.**

CAPÍTULO 7: ANEXOS Y BIBLIOGRAFÍA

7.1 ANEXOS:

7.1.1. Estudio del sensor de corriente ACS712.

Comprobación del funcionamiento del sensor de corriente tanto para corriente continua como para corriente alterna.

7.1.1.1. Corriente Continua:

Consiste en realizar mediciones de corriente continua en resistencias de valores diferentes, alimentadas con una tensión de (Valor teórico → Vcc = 5V; Valor real → Vcc = 4,98 V). Esta fuente será la misma que alimenta a todo el sistema (F). Por otro lado, se realizarán las mismas mediciones cuando el sensor de corriente utiliza como alimentación una fuente independiente[2] (FF) a 5V.

Para medir la corriente se utilizará un multímetro[3] en serie con las resistencias, seguido del sensor de corriente ASC712 para así verificar que la Intensidad que atraviesa dichas resistencias.

Además se medirá la tensión a la salida del sensor (Vout) y la tensión a la entrada del XBee (VXbee). Los valores son los siguientes:

	R teórica (Ohm)	R Real (Ohm)	I teórica (A)	I Amperímetro (A)	Vo(F) V	Vo(FF) V
R0	Infinita	Infinita	0	0	*2,478*	-
R1	1	1.2	4.150	2.02	1.471	2.364
R2	10	9.6	0.519	0.48	2.477	2.321
R3	15	15.6	0.319	0.32	2.394	2.489

Tabla 24: *ACS712. Estudio en Continua para diferentes cargas*

Según los resultados obtenidos se puede observar que el comportamiento del sensor no es el idóneo teniendo en cuenta las especificaciones del fabricante. Al probar diferentes valores de resistencias, se producen cambios inversamente proporcionales

[2] *Respecto a la **fuente**, es conveniente que al encenderla esté desconectada del circuito para evitar que los picos de corriente puedan dañar los componentes electrónicos. Pasados un par de segundos se puede conectar.*

[3] *De cara a las mediciones hay que tener un cuidado especial con el **amperímetro**. Si estimamos que las mediciones serán sobre 1A, hay que configurarlo para una escala mayor antes de medir. En caso de que no se sepa siempre es mejor partir de una escala mayor y luego ir disminuyendo hasta una escala adecuada. De esta forma se alarga la vida del fusible.*

en la corriente como en la ley de Ohm se indica, pero la tensión de salida del sensor no aumenta de forma proporcional. Para 0A la salida es aproximadamente 2,5V, pero a medida que aumenta el valor de la corriente va disminuyendo el voltaje de salida a partir de ese valor.

Cabe destacar que las medidas obtenidas cuando el sensor ACS712 está alimentado con la misma fuente que todo el sistema (F) no son del todo fiables. Las Intensidades del orden de amperios que circulan por cargas pequeñas producen un efecto negativo sobre la tensión de alimentación del sistema ya que disminuye su valor.

Para que las medidas fueran más fiables se optó por alimentar al sensor con una fuente independiente del sistema electrónico (FF) de tal forma que, aunque el valor de la intensidad fuera alto no influyera sobre la alimentación del sensor. Aun así, la tensión a la salida del sensor sigue siendo inferior a 2,5V cuando incrementa la intensidad.

Otro factor importante es el **Efecto Joule.** Ya que cuando la intensidad aumenta, las resistencias de valores pequeños tienden a calentarse mucho, y por eso es necesario utilizar resistencias de potencia en este estudio. Este efecto repercute al valor propio de las resistencias ya que tienden a aumentar su valor con la temperatura. Este fenómeno además, influye en la medición de tal forma que la intensidad va disminuyendo por el calentamiento de las resistencias. Para que la medida sea fiable, la intensidad debe alcanzar un valor estable.

7.1.2. Ciclo de Histéresis.

El ciclo de histéresis se tendrá en cuenta únicamente para el sensor de temperatura (NTC). El control de temperatura depende del grado de control requerido por la aplicación o sistema. La solución más sencilla y más económica es utilizar lo que se llama como "**Control Sí-No (ON-OFF)**". Dicho control trabaja como el termostato del hogar, en el cual la salida es 100% Sí o 100% No.

La sensibilidad del sensor, también llamado "Histéresis" o "banda muerta", es la diferencia máxima en los valores de salida del sensor.
Se diseña de tal manera que la salida no cambie demasiado rápido. Si el rango de histéresis es muy pequeño, habrá una conmutación muy rápida. Entonces, la histéresis deberá ajustarse para que haya un retardo suficiente. Debido a la histéresis necesaria, la oscilación de temperatura estará siempre presente.

- Posible comportamiento en la programación.

Hay que incluir en nuestro programa una variable que nos indique en qué posición nos encontramos. Teniendo un control ON - OFF (Gráfica salida frente a entrada) por lo cual el ciclo de histéresis es rectangular y no se consideran las pendientes. Respecto al programa debemos considerar 3 rangos y una variable de desviación (**Delta**).

Por ejemplo: Si el salto de produce en **C** (valor establecido de consigna), debemos considerar el rango (valores anteriores," C – Delta"], valores [C + Delta, posterior) y el intervalo cerrado entre [C – Delta y C + Delta]). El valor de delta debe ser mayor que el ruido del sensor (Sensibilidad). Hay que tener en cuenta dos variables booleanas (Bajando y subiendo). Ejemplo: En caso de que la Tª esté aumentando

(Subiendo = True y Bajando = False), y no se ha superado el C+Delta, el valor lógico será “false”. En caso de superarlo se permutan los valores, cambio lógico (**Subiendo** = False y **Bajando** = True).

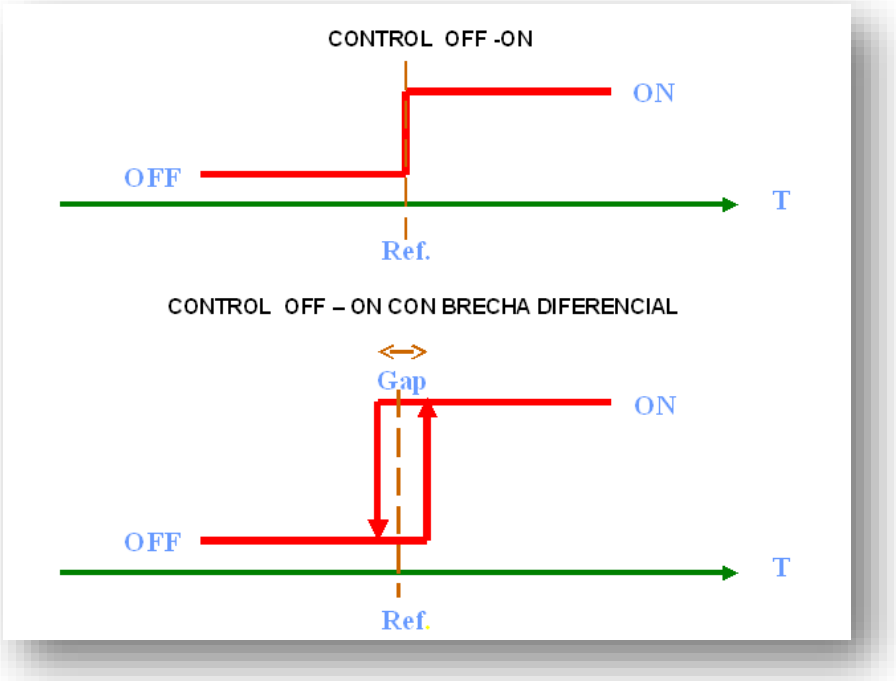

Figura 44: *Control ON-OFF. Ciclo de Histéresis [24].*

7.2 REFERENCIAS Y BIBLIOGRAFÍA

[1] http://www.scielo.org.co/pdf/iei/v25n2/v25n2a06.pdf, visitada el 08/05/ 2017.

[2] http://www.nebrija.es/%7Ejmaestro/ATA018/Domotica.pdf, visitada el 10/06/2017.

[3] Dispositivos ZigBee, aplicaciones, modos de transmisión y Protocolo IEEE 802.15.4

- ZigBee y sus aplicaciones, Ignacio Vidri Salgado, Universidad Pontificia Comillas, 2012.
- Informe Técnico: Protocolo Zigbee (IEEE 802.15.4), J. Martin Moreno y D. Ruiz Ferandez, junio 2007
- https://es.wikipedia.org/wiki/IEEE_802.15.4, visitada el 12/05/2017
- http://www.redalyc.org/pdf/5075/507550790007.pdf , visitada el 16/06/2017
- http://teinnova.aprendiendoarduino.com/2016/07/25/xbee/, visitada el 13/06/2017
- Tesis: Diseño y Construcción de un módulo… 802.15.4 (Zigbee), *J. Santiago Paz, Universidad tecnológica de Mixteca, octubre 2008*
- Proyecto fin de carrera: Sistema de Control Remoto para aplicaciones domóticas a través de Internet, M. Rodriguez Cerezo, octubre 2014

[4] Introducción a la comunicación Inalámbrica:

- http://es.ccm.net/contents/818-redes-inalambricas, visitada el 10/05/2017
- https://es.scribd.com/document/258092435/Redes-de-area-local-Espana-McGraw-Hill-pdf, visitada el 06/09/2017

[5] Tipología de Red y XBee ZNet 2.5:

- Manual Xbee : http://ftp1.digi.com/support/documentation/90000866_G.pdf, visitada el 13/05/2017
- http://teinnova.aprendiendoarduino.com/2016/07/25/xbee/, visitada el 14/06/2017
- http://xbee.cl/que-es-xbee/ , visitada el 14/05/2017

[6] Modos de comunicación.

- http://www.domodesk.com/a-fondo-zigbee, visitada por última vez en 07/06/2017
- https://www.ombushop.com/blog/diseno-web/git-para-disenadores.html, visitada el 08/05/ 2017

[7] Enlaces XUbuntu:

- https://es.wikipedia.org/wiki/Xubuntu, visitada el 13/05/2017

[8] Configuración pines IO.

- https://riull.ull.es/xmlui/bitstream/handle/915/5848/Desarrollo%20de%20un%20sistema%20controlador%20para%20red%20domotica%20inalambrica%20basa

da%20en%20protocolo%20ZigBee..pdf?sequence=1 , páginas 18, 19, 20; Visitada el 28/06/2017

[9] Sensor de presencia PIR:

- https://www.luisllamas.es/detector-de-movimiento-con-arduino-y-sensor-pir/ , visitada el 14/05/2017

[10] Sensor de Corriente CS712:

- http://www.naylampmechatronics.com/blog/48_tutorial-sensor-de-corriente-acs712.html., visitada el 15/06/2017
- https://www.sparkfun.com/datasheets/BreakoutBoards/0712.pdf , visitada el 15/08/2017

[11] Sensor de Temperatura y Humedad AMT1001

- http://www.electrodragon.com/w/AMT1001 , visitada el 15/05/2017

[12] Sensor de luminosidad LDR

- https://www.sparkfun.com/datasheets/Sensors/Imaging/SEN-09088-datasheet.pdf , visitada el 16/04/2017
- https://play.google.com/store/apps/details?id=crunchybytebox.lightmeter , visitada el 16/05/2017

[13] Sensor de temperatura NTC:

- http://www.picuino.com/es/uno-doc/sensor-temp.html , visitada el 16/06/2017

[14] Relés de 2 Canales:

- http://modtronix.com/mod-rly2-5v.html , visitada el 17/05/2017

[15] Leds:

- http://unicrom.com/diodo-led/ , visitada el 16/06/2017
- Libro de Principios de Electrónica (Albert Malvino y David J. Bates)
- https://www.alliedelec.com/m/d/6355b8aba0b01578df0bb7b871ceefd7.pdf, visitada el 16/05/2017

[22] Anexo ciclo de histéresis:

- http://serverpruebas.com.ar/news13/nota08.htm, visitada el 27/08/2017

[23] Acondicionador de señal NTC:

- Libro Instrumentación Electrónica, O. González, S. Hernández, 2013

[24] Control ON OFF. Ciclo de histéresis

- http://www.monografias.com/trabajos105/controladores/controladores.shtml

[25] Enlace al repositorio en GITHUB
https://github.com/ULL-InformaticaIndustrial-Empotrados/TFG_David_Feliciano,
consultado por última vez 12/09/2017