UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR

PROYECTO FIN DE CARRERA

ELABORACIÓN DE UN SISTEMA PARA REALIZACIÓN DE AUDITORÍAS DE CONSUMO DE ENERGÍA ELÉCTRICA A TRAVÉS DE INTERNET

Javier Fernández Tapia

Julio 2015

ELABORACIÓN DE UN SISTEMA PARA REALIZACIÓN DE AUDITORÍAS DE CONSUMO DE ENERGÍA ELÉCTRICA A TRAVÉS DE INTERNET

AUTOR: Javier Fernández Tapia
TUTOR: Javier Garrido Salas

HCTLab
Departamento de Tecnología Electrónica y de las Comunicaciones
Escuela Politécnica Superior
Universidad Autónoma de Madrid
Julio 2015

Agradecimientos

En primer lugar a mis padres, por guiarme a lo largo de toda mi vida y porque gracias a ellos soy Ingeniero.

A mis amigos de la carrera Jaime, Fernando, Joseda, Gonzalo y Álvaro, por su apoyo durante estos años y por su amistad.

Gracias.

Resumen

En este proyecto se desarrolla un sistema que realiza mediciones detalladas de consumo energético y las muestra en tiempo real a través de internet. Este proyecto, junto con otros desarrollos futuros, constituirán un sistema domótico para automatización y control de viviendas.

Para que estas mediciones sean lo más detalladas posible, se ha diseñado y desarrollado un sistema que permite medir el consumo de cualquier aparato que se alimente de la red eléctrica doméstica utilizando un enchufe tradicional. Además, para visualizar las mediciones, el sistema cuenta con una aplicación web que muestra los datos en tiempo real. La aplicación web también almacena un historial de consumo, y permite calcular el consumo en intervalos de tiempo con precisión de segundos, así como visualizarlos mediante gráficas.

Para el desarrollo de este proyecto, se han empleado tecnologías como ZigBee para la creación de enlaces inalámbricos, sistemas embebidos como Raspberry Pi para el procesamiento de datos, y las tecnologías web como MeteorJs, HTML5 y MongoDB para el desarrollo de la aplicación web.

Abstract

This Project aims to develop a system that carries out detailed energy measurements and makes them available over the Internet. This Project and other future developements will be part of a complete domotic system for home control and automation.

This system has been designed to carry out these energy measurements in a detailed way, so it is capable of measuring the energy consumption of any electrical appliances supplied with energy from the city grid using a common outlet. In addition, the measurements data is available in real time thank to a web-based application. The web-based application provides a history of consumption and allows to calculate energy use in specific time intervals, as well as visualizing them using charts.

This project takes advance of technologies like ZigBee to create wireless sensor networks, embedded systems like Raspberry Pi for data processing and the latest technologies for web development like MeteorJs, HTML5 and MongoDB.

Índice

AGRADECIMIENTOS **5**

RESUMEN **7**

ABSTRACT **9**

ÍNDICE **11**

1. INTRODUCCIÓN. **15**

1.1 MOTIVACIÓN Y ANTECEDENTES. **15**

1.2 OBJETIVOS 15

1.3 ORGANIZACIÓN DE LA MEMORIA 16

CAPÍTULO 1 16

CAPÍTULO 2 16

CAPÍTULO 3 16

2. ESTADO DEL ARTE. **17**

2.1 INTRODUCCIÓN. 17

2.2 ESTUDIO DE MEDIDORES DE CONSUMO ELÉCTRICO COMERCIALIZADOS EN ESPAÑA 17

2.2.1 Soluciones para gestión de energía de Current Cost Iberia. *17*

2.2.2 Gestor de consumo Ecobox de Home Systems *19*

2.2.3 Medidores de consumo energético inteligentes de "The Owl" *20*

2.3 ESTUDIO DE TECNOLOGÍAS NECESARIAS PARA EL DISEÑO Y DESARROLLO DEL SISTEMA. 20

2.3 IDEA GENERAL DE DISEÑO GLOBAL DEL SISTEMA. 21

2.3 ENLACES DE RADIOFRECUENCIA DE CORTO ALCANCE. 22

2.3.1 ZigBee y IEEE 802.15.4 ..... *22*

2.3.2 Redes Zigbee *23*

2.3.2.1 Tipos de dispositivos en una red Zigbee [1] 23

2.3.2.2 Topologías de las redes ZigBee 23

2.3.2.3 Modulos Xbee. 25

2.3.2.4 Módulos Xbee serie 1 PRO 26

2.3.2.4.1 Diagrama esquemático de los pines de un Xbee Serie 1 27

2.3.2.5 Modos de operación. 28

2.3.2.5.1 Modo comando 28

2.3.2.5.2 Modo transparente 30

2.3.2.5.3 Modo API 30

2.3.3 Conclusiones sobre tecnología ZigBee *32*

2.4 HARDWARE MEDIDOR DE CONSUMO. 32

2.4.1 Potencia en alterna. *32*

2.4.2 Circuito integrado STPM10 ..... *33*

2.4.3 Medidor de consumo energético DSS5188 ..... *35*

2.4.4 Conclusiones sobre medidores de consumo estudiados *37*

2.5 HARDWARE DE CONTROL Y PROCESADO DE DATOS. 38

2.5.1 Descipción detallada de Raspberry Pi [3] *39*

2.5.1.2 Hardware. 39

2.5.1.2.1 Alimentación. 39

2.5.1.2.2 Puerto USB. 40

2.5.1.2.3 GPIO. 40

2.5.1.3 Software. 43

2.6 APLICACIÓN WEB. 45

2.6.1 Meteor.js [4][5] *45*

2.6.1.2 Estructura de datos de MeteorJs. 46

2.6.1.3 Reactividad en MeteorJS 47

2.6.1.4 Plantillas HTML. 48

2.6.1.5 Helpers 49

2.6.1.6 SpaceBars 51

2.6.1.7 onRendered 51

2.6.1.8 La variable Session ... 52
2.6.1.9 Tracker ... 53
2.6.1.10 Almacenamiento de datos en aplicaciones Meteor. ... 53
2.6.1.11 Colecciones ... 54
2.6.1.11.1 MongoDB ... 54
2.6.1.11.2 Crear una colección ... 54
2.6.1.11.3 Querys en Meteor. ... 55
2.6.1.11.4 Insertar elementos en colecciones. ... 55
2.5.1.11.5 Otros métodos para manejo de colecciones. ... 56

3. DISEÑO Y DESARROLLO. ... 57
3.1 OPCIONES DE DISEÑO HARDWARE CONTEMPLADAS. ... 57
3.1.2 Opción elegida. ... 59
3.2 RED DE COMUNICACIÓN ENTRE NODOS Y UNIDAD CENTRAL. ... 60
3.2.1 Topología de red. ... 60
3.2.2 Configuración de los módulos transmisores y receptores. ... 61
3.3 UNIDAD REMOTA. ... 64
Descripción general ... 64
3.3.1 Toma de medidas de consumo. ... 64
3.3.2 Recogida de medidas y transmisión a la unidad central. ... 66
3.3.2.1 Toma de medidas ... 66
3.3.2.2 PCB interfaz DSS5188 módulo XBee ... 66
3.3.2.1.1 Zócalos hembra-macho 2mm para Xbee ... 68
3.2.2.1.2 Regulador de tensión ... 68
3.2.2.1.2 Conversor de voltaje bi-direccional TXB0104 [7] ... 68
3.2.2.1.3 Zócalos hembra-macho 2,54mm ... 69
3.2.2.1.4 Condensadores de desacoplo ... 69
3.2.2.1.4 Conector para fuente de 5V ... 69
3.4 UNIDAD CENTRAL. ... 69
3.4.1 Interfaz hardware XBee-Raspberry Pi. ... 70
3.4.1.1 Conexiones implementadas. ... 70
3.4.1.2 PCB en Raspberry Pi. ... 71
3.4.2 Software en Raspberry Pi. ... 74
3.4.2.1 Recepción y envío de tramas mediante comunicación serie ... 74
3.4.2.1.1 UART (Universal Asynchronous Receiver-Transmitter) ... 74
3.4.2.1.2 Software para envío y recepción de tramas API ... 75
3.4.2.1.3 Configuración remota de los módulos XBee. ... 75
3.4.2.1.4 Recepción de tramas API. ... 81
3.4.2.2 Algoritmo global de recepción y procesado. ... 83
3.5 APLICACIÓN WEB. ... 84
3.5.1 Tecnologías empleadas ... 84
3.5.2 Estructura de la aplicación web. ... 84
3.5.2.1 client/ ... 84
3.5.2.1.1 graficas_rendered.js ... 85
3.5.2.1.2 graficas_helpers.js ... 87
3.5.2.1.3 graficas_events.js ... 89
3.5.2.1.4 Router.js ... 90
3.5.2.1.5 graficas.html ... 91
3.5.2.1.6 graficas.css ... 92
3.5.2.2 /deploy. ... 92
3.5.2.3 lib ... 92

4. PRUEBAS Y RESULTADOS ... 93
4.1 UNIDAD REMOTA ... 93
4.1.2 Pruebas al medidor de consumo ... 93
4.1.3 Pruebas sistema medidor de consumo más PCB ... 95
4.2 UNIDAD CENTRAL ... 96
4.2.1 Pruebas al PCB de la unidad central ... 96
4.2.1 Pruebas al software de procesado y aplicación web ... 96
4.4 PRUEBAS SISTEMA COMPLETO ... 98

5. CONCLUSIONES Y TRABAJO FUTURO. 101
5.1 CONCLUSIONES. 101
5.2 TRABAJO FUTURO 101
BIBLIOGRAFÍA. 103
APÉNDICES 104
A. PRESUPUESTO. 104
B. PLIEGO DE CONDICIONES 105

1. Introducción.

1.1 Motivación y antecedentes.

Desde hace varios años los continuos avances tecnológicos han traído consigo la aparición de innumerables tecnologías que hacen que hacen nuestro día a día cada vez más fácil.

Este proyecto surge del deseo de querer emplear estas tecnologías en el campo de la domótica y contribuir así al desarrollo de viviendas cada vez más cómodas, modernas y eficientes. En concreto, este proyecto consiste en el desarrollo de un sistema de gestión energética, que proporcionará a sus usuarios información de consumo eléctrico mucho más detallada que un consumo mensual o trimestral. De esta manera, los usuarios podrán conocer el consumo eléctrico detallado de todos y cada uno de los electrodomésticos y aparatos eléctricos usados día a día, y así poder eliminar consumos innecesarios que muchas veces pasan desapercibidos.

Dada la reciente expansión y desarrollo en los últimos años de internet, los ordenadores y los dispositivos móviles, el sistema se desarrolla de manera que la información proporcionada acerca del consumo se pueda monitorizar a través de internet de una manera fácil, cómoda y a su vez detallada.

1.2 Objetivos

El problema mencionado en el apartado anterior ha sido resuelto gracias al desarrollo de un sistema que involucre distintas herramientas hardware y software.

Se ha diseñado y desarrollado un dispositivo que se intercala entre el enchufe de la pared y el aparato cuyo consumo eléctrico se desee medir. De esta manera el aparato no se conectará directamente al enchufe en la pared, sino que irá conectado al dispositivo y a su vez el dispositivo se conectará al enchufe en la pared. El dispositivo actuará de “puente eléctrico”, permitiéndole así realizar mediciones acerca del consumo eléctrico del aparato al que está suministrando la electricidad.

Los datos de consumo recogidos por el dispositivo citado en el párrafo anterior son mostrados en una página web, proporcionando al usuario la capacidad de monitorización en tiempo real de su consumo eléctrico.

1.3 Organización de la memoria

Capítulo 1

En este capítulo se habla de las motivaciones que llevan a la realización del proyecto y los objetivos planteados.

Capítulo 2

En este capítulo se hace un estudio de las diferentes posibilidades de desarrollo del proyecto. Para ello se plantean diferentes enfoques para el proyecto, haciendo hincapié en las tecnologías que se podrían usar para resolver el problema planteado.

Capítulo 3

En este capítulo se habla en detalle de la solución adoptada, así como de las tecnologías elegidas, del funcionamiento del sistema y del proceso de desarrollo.

2. Estado del arte.

2.1 Introducción.

La domótica se podría definir en pocas palabras como el conjunto de técnicas orientadas a automatizar y modernizar la vivienda. Con ese propósito, la domótica busca integrar tecnología para proporcionar al usuario mejores prestaciones de seguridad, confort y gestión energética.

Con este proyecto se ha querido realizar una aportación a la domótica, en concreto al ámbito relacionado con la gestión energética. Desde el principio se pretendía dotar al usuario de aplicaciones domóticas de un sistema que le permitiera monitorizar su consumo eléctrico detalladamente. Actualmente en la mayoría de los hogares, la información acerca del consumo eléctrico cotidiano de la que se dispone se caracteriza por una notable falta de detalle, es decir, se dispone de una factura eléctrica mensual o trimestral, pero; ¿Realmente un dato de consumo mensual nos proporciona una información útil acerca de nuestro consumo energético? ¿Y si se pudiera disponer de los datos de consumo eléctrico por franjas de tiempo? ¿Y si se pudiera conocer el consumo individual de cada electrodoméstico o aparato que requiera corriente eléctrico? ¿Y si esta información fuera accesible en todo momento?

Además, debido a la gran expansión de internet, los ordenadores y los dispositivos móviles, se consideró adecuado, o más bien indispensable (a la vez que beneficioso), considerar estas tecnologías tan familiares, aceptadas y de fácil acceso como parte de la solución.

2.2 Estudio de medidores de consumo eléctrico comercializados en España

Antes de comenzar con el diseño de la solución que proporcionará este proyecto, se estudiaron algunas de las soluciones que otros fabricantes de ofrecen actualmente en España.

2.2.1 Soluciones para gestión de energía de Current Cost Iberia.

La compañía Current Cost Iberia dispone de sistemas de monitorización de consumo energético a la venta en España. La solución que esta compañía proporciona para la gestión de consumo energético, se sirve de varios sensores o medidores de consumo energético, enfocados a diferentes usos. Estos sensores

van acompañados de un transmisor inalámbrico, que manda los datos a otro aparato que sirve de monitor para visualizar los datos. Disponen de los siguientes tipos de medidores:

- Sensores ópticos: no son capaces de realizar una medición de consumo energético por sí solos, si no que se valen de los contadores de consumo energético proporcionados por las compañías eléctricas. Estos contadores suelen disponer de un led que parpadea cada vez que cierta cantidad de energía se ha consumido. En este caso, el sensor óptico se encarga de contar los pulsos luminosos proporcionados por estos medidores, gracias a los cuales se puede saber el consumo.

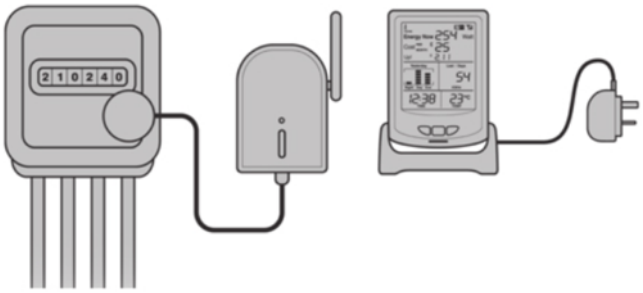


Figura 2.0.1. Representación de sistema contador-sensor óptico-transmisor-receptor (de izquierda a derecha)

- Sensor de energía eléctrica: se instala en el cuadro eléctrico de la casa, siendo capaz de medir autónomamente el consumo eléctrico. Sirve para monitorizar el consumo global de la casa.

Figura 2.0.2: Sensor eléctrico acompañado de transmisor inalámbrico. A la derecha se muestra la unidad central que recibe las mediciones y las muestra al usuario

- Sensor de enchufe (figura 2.0.3): se intercala entre el enchufe de la pared y el aparato cuyo consumo se desea medir. Sirve para proporcionar mediciones más detalladas de consumo. Permite medir el consumo de aparatos que como mucho consuman 3 KW de potencia.

Figura 2.0.3. Sensor de enchufe.

Las mediciones recogidas por los sensores de enchufe, al igual que con los sensores mencionados anteriormente, serán enviadas al monitor de consumo.

Así mismo, Current Cost Iberia, también proporciona un módulo de internet, que se conecta entre el router de la vivienda y el monitor, y hace posible el acceso a los datos de consumo a través de internet.

2.2.2 Gestor de consumo Ecobox de Home Systems

El sistema Ecobox consiste en un medidor de consumo energético que se instala en el cuadro eléctrico. El medidor mostrado en la figura 2.0.4 se conecta al cuadro eléctrico de la vivienda, y es capaz de proporcionar al usuario datos detallados acerca del consumo eléctrico de la vivienda.

Figura 2.0.4. Gestor de energía Ecobox.

Este gestor de energía, también proporciona la posibilidad de monitorizar el consumo (por franjas horarias) y acceder a una interfaz más detallada desde smartphones y ordenadores a través de Bluetooth.

Además, Ecobox dispone de otras funciones como la incorporación de actuadores para manejo de relés y otros actuadores compatibles con la tecnología X10 (comunicación por cable a través de el cableado de la red eléctrica).

2.2.3 Medidores de consumo energético inteligentes de "The Owl"

El sistema de gestión de consumo energético de la compañía The Owl, es capaz de proporcionar monitorización de consumo global de una vivienda a través de internet.

Para ello, se instala un medidor y un transmisor inalámbrico en cuadro eléctrico de la vivienda. El medidor realiza las mediciones, y el transmisor las envía a una unidad de red (también proporcionada por la compañía) con conexión a internet, que hará que los datos de las mediciones estén disponibles a través de internet.

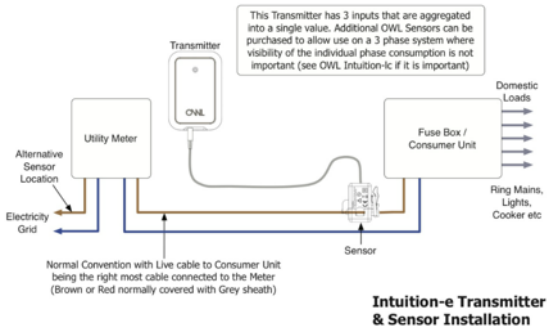


Figura 2.0.4. Conexiones necesarias para la instalación de el medidor y el transmisor en el cuadro eléctrico de la vivienda.

2.3 Estudio de tecnologías necesarias para el diseño y desarrollo del sistema.

Una vez estudiadas los sistemas de gestión energéticas disponibles en el mercado actualmente, se buscó elaborar un sistema de características similares. Este proyecto consiste en crear un sistema para la realización de auditorías de consumo energético, pero perteneciente al HCTLab. El sistema se desarrolla utilizando tecnologías modernas y flexibles que permitirán que se sigua trabajando en este proyecto, mejorándolo y añadiéndole funcionalidades para crear un sistema domótico completo.

En este apartado se estudiarán las tecnologías que influirán en el diseño y en el desarrollo del sistema.

2.3 Idea general de diseño global del sistema.

Como se ha dicho en la introducción, se pretendía obtener datos de consumo eléctrico lo más detallado posible, por este motivo se llegó a la conclusión de que la medición del consumo se llevaría a cabo en cada enchufe, proporcionando así tanta individualización a la hora de realizar las mediciones como el usuario desee.

La mera idea de proporcionar mediciones individualizadas por si sola trajo varias consecuencias que afectarían al diseño global, y por lo tanto a la elección de las tecnologías que se iban a utilizar. Entre las consecuencias más importantes se pueden resaltar:

- El sistema contará con tantas unidades hardware medidoras de consumo (a partir de ahora nodo) como se desee incorporar (en función de la cantidad de consumos de diferentes electrodomésticos que se deseen medir).
- El punto anterior trae consigo la replicación de componentes hardware en cada nodo, y por lo tanto, se debe buscar un diseño óptimo que intente replicar en cada nodo únicamente el hardware imprescindible. Dadas las necesidades del proyecto, las funcionalidades hardware imprescindibles son:
    - Llevar a cabo la medición del consumo: Imprescindible en cada nodo.
    - Procesar los datos de consumo y hacerlos visibles desde un dispositivo con acceso a internet.

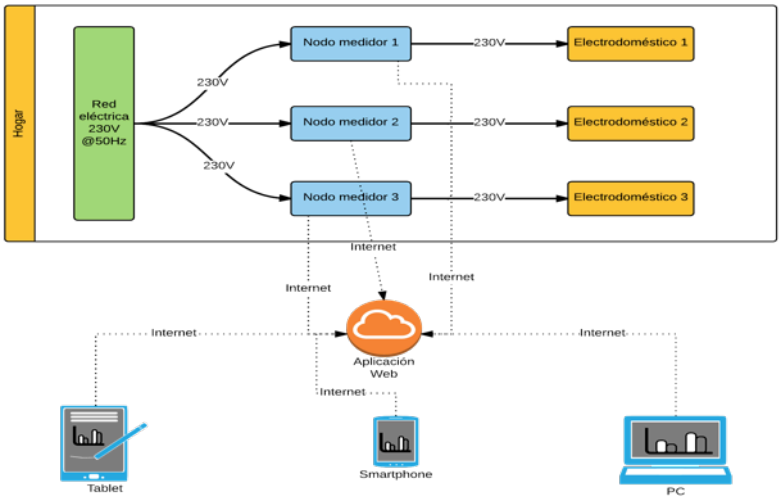


Figura 2.1.Representación del sistema completo.

Aunque el esquema general está claro, aun falta por decidir qué tecnologías utilizar exactamente para proporcionar la conexión a internet. Esta conexión puede ser conseguida incorporando tecnologías de comunicación móviles en cada nodo o incorporando conexión inalámbrica por wifi a un router con salida

a internet. No obstante, estas tecnologías serían demasiado caras si se requiere replicarlas en cada nodo. Por ello, se estudió una tecnología de comunicación inalámbrica de bajo coste y corto alcance, ZigBee.

2.3 Enlaces de radiofrecuencia de corto alcance.

Las mediciones llevadas a cabo por cada nodo deben estar disponibles en una plataforma web, lo que implica que cada nodo debe tener conexión a internet, directa o indirectamente.
Dado que dotar a cada nodo de la tecnología necesaria para que puedan conectarse a internet directamente sería demasiado costoso, se optó por estudiar la tecnología ZigBee. Esta tecnología sirve para la creación de enlaces inalámbricos de corto alcance y bajo coste, y ya ha sido utilizada anteriormente en HTCLab. Ya que el HCTLab dispone de hardware capaz de crear enlaces inalámbricos mediante ZigBee, se realizó un estudio detallado de esta tecnología. Aunque esta tecnología no es capaz de proporcionar un acceso directo a internet, sí puede servir para comunicar cada nodo con otro dispositivo que sí disponga de conexión a internet.

2.3.1 ZigBee y IEEE 802.15.4

El protocolo ZigBee fue estudiado en vistas a encontrar una tecnología que pudiera transmitir las mediciones de consumo recogidas en cada nodo. Se buscaba una tecnología de bajo coste, con facilidad de despliegue y apropiada para la creación de enlaces de radiofrecuencia de corta distancia.

Zigbee es un protocolo de comunicaciones inalámbrico basado en el estándar 802.15.4. El propósito general del estándar IEEE 802.15.4 es el de definir el nivel físico y control de acceso al medio de redes inalámbricas de área personal con bajas tasas de transmisión, y tiene como característica fundamental la reducción de los costes de fabricación por medio de la sencillez tecnológica.

El estándar 802.15.4 está ampliamente extendido, y este opera en las siguientes bandas de frecuencia:

- Banda de 2,4 GHz con DSSS (Direct Sequence Spread Spectrum) como método de codificación de canal y O-QPSK (Oset Quadrature Phase Shift) como técnica de modulación. En esta banda se consiguen 16 canales con una tasa de transferencia de 250 Kbps.
- Bandas de 868 MHz. con DSSS (Direct Sequence Spread Spectrum) como método de codificación de canal y BPSK (Binary Phase Shift Keying) como técnica de modulación. Se consiguen 11 canales a 20 Kbps.

- Banda de 915MHz usa las mismas técnicas de codificación de canal y de modulación que en 868 MHz, también se dispone de 11 canales, pero las tasas de transferencia son de 40 Kbps.

Zigbee es una aplicación del estándar 802.15.4 y fue diseñado para crear redes inalámbricas con las siguientes características:

- Bajo coste y facilidad de despliegue y mantenimiento.
- Bajas tasas de transferencia de datos.
- Bajo consumo.
- Alta capacidad y flexibilidad de ampliación.

2.3.2 Redes Zigbee

2.3.2.1 Tipos de dispositivos en una red Zigbee [1]

Una red Zigbee se compone de varios elementos que serán clasificados según su funcionalidad:

- Coordinador: Debe existir uno y sólo uno en cada red. Es el encargado de crear, inicializar y controlar la red, estableciendo aspectos como el canal de comunicaciones y asignando parámetros como las direcciones de red. Además, también tienen funcionalidad de router, con capacidad para encaminar mensajes que no van dirigidos expresamente a él.
- Routers: se encargan de interconectar dispositivos, enrutando mensajes hacia el nodo correspondiente. Gracias a estos dispositivos se pueden ampliar las redes y crear nuevas rutas de comunicación creando así redes más robustas.
- Dispositivos finales: son los encargados de llevar a cabo las acciones requeridas y para comunicarse usan la red proporcionada por los routers y el coordinador. Estos dispositivos no enrutan mensajes a otros dispositivos, ya que su funcionalidad está más enfocada a la reducción de recursos energéticos y hardware. Debido a que no tienen la capacidad de enrutar, los dispositivos finales siempre deben contar con un enrutador o un coordinador para comunicarse.

2.3.2.2 Topologías de las redes ZigBee

La estructura lógica en la que los elementos de una red Zigbee se encuentran conectados puede ser de diferentes maneras:

- Estrella (figura 2.2): Los dispositivos finales no se comunican entre sí directamente, todos los mensajes en la red deben pasar por el coordinador, que los enrutará al nodo destino.

Figura 2.2. Topología en estrella: Rojo: coordinador, Beige: dispositivo final

Malla(figura 2.3): en este tipo de configuración se emplean enrutadores aparte del coordinador. Sin embargo, el coordinador, aparte de gestionar la red, también actúa como enrutador.

Figura 2.3. Topología en Malla: Beige: dispositivos finales, Rojo: dispositivo coordinador, Azul: dispositivo enrutador

- Árbol(figura 2.4): Los componentes de la red se organizan jerárquicamente. De esta manera, cada enrutador tiene asignados unos hijos, y a su vez cada hijo un único enrutador. Los dispositivos finales no pueden tener hijos.

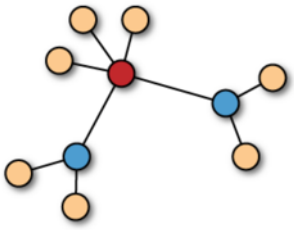


Figura 2.4. Topología en árbol: Beige: dispositivo final, Rojo: disositivo coordinador, Azul: dispositivo enroutador

2.3.2.3 Modulos Xbee.

Xbee es una marca de componentes de radiofrecuencia que soporta una variedad de protocolos de radio incluyendo Zigbee. Los módulos Xbee se pueden clasificar en dos series cuyas características se ven reflejadas en la tabla 2.1.

	Serie 1	Serie 2
Alcance interior	30 m	40 m
Alcance exterior	100 m	120 m
Potencia de transmisión	1mW (0dBm)	2mW (3dBm)
Tensión de alimentación	2.8 – 3.4 V	2.8 – 3.6 V
Sensibilidad del receptor	-92 dBm	-98 dBm
Corriente de apagado		
Topología de red	Punto a punto, estrella.	Punto a punto, estrella, malla.
Rango de temperatura	-40 a 85ºC	-40 a 85ºC
Entradas/Salidas digitales	8	10
Entradas/Salidas analógicas	7/2	4/Ninguna

Tabla 2.1. Características técnicas de Xbee Serie 1 y Serie 2

Aparte, de cada serie existe una versión más potente (versión PRO), con las mismas características pero con mayor potencia de transmisión. Con estos módulos PRO, se puede llegar a doblar el alcance del enlace de radio.
Independientemente de su serie, los módulos XBee también permiten incorporar diferentes tipos de antenas dependiendo de la necesidad:

- Chip antena(figura 2.5): con diagrama de radiación unidireccional en forma de cardioide.

Figura 2.5. Módulo Xbee con antena tipo chip

- Wire antena (figura 2.6): con diagrama de radiación omnidireccional.

Figura 2.6. Módulo XBee con antena omnidireccional

- Conector U.FL (figura 2.7): Recomendable para aplicaciones en las que se requiere un diagrama de radiación específico. Utilizado para conectar el módulo a una antena externa por medio de un cable.

Figura 2.7. Módulo XBee con conector U.FL

- Conector RPSMA (figura 2.8): utilizado para conectar una antena directamente al módulo.

Figura 2.8. Módulo XBee con conector RPSMA

En el laboratorio se dispone de módulos Xbee Serie 1 PRO, por ello se estudiaron sus características técnicas.

2.3.2.4 Módulos Xbee serie 1 PRO.

Estos módulos operan en la banda de 2,4 GHz, con una tasa de transferencia de 256 Kbps, un alcance de hasta 100m en interiores y una sensibilidad de -100 dBm, además, también incorporan un pequeño microcontrolador de 8 bits

que les permite llevar a cabo otras funciones complementarias de utilidad. Aparte de crear enlaces radioeléctricos, los pines de entrada y salida disponen de funcionalidades como:

- Entradas/salidas digitales.
- Entradas/salidas analógicos con conversores analógico-digitales.
- Interfaz para comunicación serie (UART).

Estas funcionalidades permiten encomendar a estos dispositivos tareas sencillas como lectura sensores analógicos y accionamiento de actuadores que normalmente es llevado acabo por otros sistemas embebidos.

2.3.2.4.1 Diagrama esquemático de los pines de un Xbee Serie 1

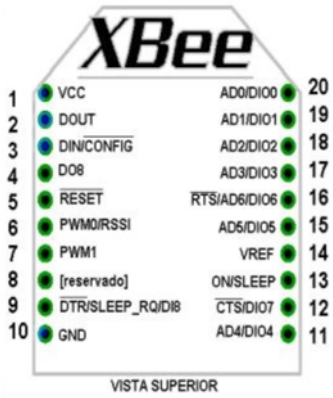


Figura 2.9. Diagrama de conexiones en módulo XBee serie 1 y Serie 1 PRO

Pin #	Name	Direction	Description
1	VCC	-	Power supply
2	DOUT	Output	UART Data Out
3	DIN / **CONFIG**	Input	UART Data In
4	DO8*	Output	Digital Output 8
5	**RESET**	Input	Module Reset (reset pulse must be at least 200 ns)
6	PWM0 / RSSI	Output	PWM Output 0 / RX Signal Strength Indicator
7	PWM1	Output	PWM Output 1
8	[reserved]	-	Do not connect
9	DTR / SLEEP_RQ / DI8	Input	Pin Sleep Control Line or Digital Input 8
10	GND	-	Ground
11	AD4 / DIO4	Either	Analog Input 4 or Digital I/O 4
12	**CTS** / DIO7	Either	Clear-to-Send Flow Control or Digital I/O 7
13	ON / **SLEEP**	Output	Module Status Indicator
14	VREF	Input	Voltage Reference for A/D Inputs
15	Associate / AD5 / DIO5	Either	Associated Indicator, Analog Input 5 or Digital I/O 5
16	**RTS** / AD6 / DIO6	Either	Request-to-Send Flow Control, Analog Input 6 or Digital I/O 6
17	AD3 / DIO3	Either	Analog Input 3 or Digital I/O 3
18	AD2 / DIO2	Either	Analog Input 2 or Digital I/O 2
19	AD1 / DIO1	Either	Analog Input 1 or Digital I/O 1
20	AD0 / DIO0	Either	Analog Input 0 or Digital I/O 0

Tabla 2.2. Descripción detallada de pines en esquemático de conexiones de módulo XBee Serie 1

Cuyas características eléctricas quedan reflejadas en la tabla 2.3.

	Mínimo	Máximo	Unidad
Tensión de alimentación	2.8	3.4	V
Tensión de entrada nivel bajo (VIL)	-	0.35*VCC	V
Tensión de entrada nivel alto (VIH)	0.7*VCC	-	V
Tensión de salida nivel bajo (VOL)	-	0.5	V
Tensión de salida nivel alto (VOH)	VCC-0.5	-	V
Tensión de referencia para ADC(VREF)	2.08	VCC	V
Corriente en la transmisión RF (TX)	215	215	mA
Corriente en la recepción RF (RX)	55	55	mA

Tabla 2.3. Caracteísticas eléctricas módulo XBee Serie 1

2.3.2.5 Modos de operación.

2.3.2.5.1 Modo comando.

El modo comando permite comunicar directamente un ordenador o sistema embebido con el módulo XBee por el puerto serie (UART). Gracias a este modo de operación, se puede acceder a la información de configuración del módulo XBee para consultarla o para modificarla.

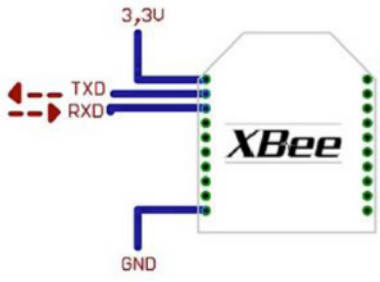


Figura 2.10. Diagrama de conexiones necesarias para comunicación con módulo XBee Serie 1 mediante UART (comunicación serie)

Para acceder a este modo de funcionamiento, se debe conectar el módulo XBee por UART (figura 2.10), una vez conectado, desde una consola serie y gracias a los comandos AT (attention) se puede interactuar con el módulo para configurarlo. Los comandos AT más relevantes son los siguientes:

- AT: utilizado para comprobar que existe comunicación entre el XBee y el ordenador. Al introducir "AT" en la consola serie, el XBee debe responder simplemente "OK" para indicar que existe comunicación y que está funcionando correctamente.
- ATID: En las redes ZigBee, cada módulo XBee tiene un identificador de red personal asignado. Este comando permite averiguar el identificador o bien modificarlo.

- ATSH/ATSL (figura 2.11): Cada XBee también tiene asignado un número de serie de 64 bits único y no puede ser cambiado. De esta forma, el comando ATSH permite acceder a los 32 bits más significativos de esta dirección, y el comando ATSL a los menos significativos. Esta dirección también se puede ver impresa en la parte trasera de cada módulo.

Figura 2.11. Número de serie de 64 bits único para módulo XBee

- ATDH/ATDL: asigna o muestra la el identificador de 64 bits del módulo al que enviará información. Al igual que en el comando anterior, los dos comandos sirven para acceder a los 32 bits más significativos y a los 32 bit menos significativos respectivamente.

- ATWR: este comando sirve para hacer permanente la configuración actual del XBee. Vía firmware esta configuración queda guardada de tal manera que aunque se interrumpa la alimentación del módulo, esta no se perderá.

- ATD0-D7: Este comando sirve para configurar los pines del 0 al 7 como entradas o salidas. El número de después de la D indica el pin que se esta configurando, y otro número después de este comando deberá ser incluido para indicar si el pin será configurado como E/S digital, entrada analógica (sólo de pines de 0 a 3). Estos son las opciones que pueden seguir a cada comando ATDx (tabla 2.4).

0	Desactivar I/O pin.
1	Función especial de cada pin.
2	Configurar como entrada analógica.
3	Configurar entrada digital.
4	Configurar como salida digital nivel bajo.
5	Configurar como salida digital nivel alto.

Tabla 2.4. Configuración de comandos ATD

- ATIC: sirve para monitorizar el cambio de estado en un pin configurado como entrada. Cuando se registra un cambio el módulo XBee envía un mensaje alertando del evento.

2.3.2.5.2 Modo transparente.

Es el modo más sencillo, el módulo XBee se comporta de la siguiente manera:

- La información que recibe por radio frecuencia es transmitida vía UART tal cual se recibe.
- La información recibida por UART es transmitida por radio frecuencia a la dirección destino. La dirección destino deberá haber sido configurada previamente por comandos AT.

Es modo se utiliza frecuentemente para reemplazar la comunicación serie por cable, también recibe el nombre de cable virtual.

2.3.2.5.3 Modo API

El modo API, a diferencia del modo transparente, empaqueta los datos en tramas a fin de conseguir una comunicación más estructurada y que aproveche más las capacidades de una red ZigBee. El modo API, permite la configuración remota de los nodos XBee por medio de comandos AT sin necesidad de acceder al modo comando ni de usar una consola serial, permite enviar información a diferentes dispositivos sin necesidad de pre-configurar la dirección de envío y también es posible identificar la procedencia de las tramas de datos que se reciben, ya que cada trama posee la dirección de el nodo remitente y del destinatario.

En modo API hay 5 tramas diferentes:

- Trama de envío de datos.
- Trama de comando AT.
- Trama de recepción de datos.

- Trama de respuesta a comando AT.
- Trama de notificación de eventos.

Figura 2.12 Estructura general de una trama API

Start delimiter.
Sirve para identificar comienzo de la trama y su valor debe ser siempre "0x7E" cuando se reciben datos por medio de UART.
Lenght bytes.
Indican el tamaño del campo "Frame Data", donde se encuentra la información útil de la trama.
Frame data.
En esta parte de la trama es se encuentra el comando API específico (pudiendo ser datos, comandos AT, respuestas a comandos AT…). Esta parte de la trama a su vez tiene una estructura que dependiendo de tipo de comando API, hay más de una docena de tipos de frame data (tabla 2.5)

0x17	Comando AT remoto.
0x97	Respuesta comando AT remoto.
0x08	Comando AT.
0x09	Comando AT (En cola de espera)
0x88	Respuesta comando AT.
0x01	Transmisión de datos. (Direccionamiento 16 bits)
0x81	Recepción de datos. (Direccionamiento 16 bits)
0x00	Transmisión de datos. (Direccionamiento 64 bits)
0x08	Recepción de datos. (Direccionamiento 64 bits)
0x83	Recepción de datos de entradas digitales y analógicas (Direccionamiento de 16 bits)
0x82	Recepción de datos de entradas digitales y analógicas (Direccionamiento de 64 bits)
0x89	Resultado de la transmission.
0x8A	Estado del modem.

Tabla 2.5. Tipos de frame data.

Checksum
Sirve para comprobar que la trama recibida no está dañada, en tal caso la trama será descartada.

2.3.3 Conclusiones sobre tecnología ZigBee

Después de un estudio detallado de la tecnología ZigBee y los módulos XBee, se ha llegado a la conclusión de que estos satisfacen las necesidades de comunicación inalámbrica de cada nodo. De ahora en adelante, teniendo en cuenta las capacidades de los módulos XBee, se intentará buscar hardware que sea compatible con las características de estos módulos.

2.4 Hardware medidor de consumo.

2.4.1 Potencia en alterna.

Antes de buscar el hardware a emplear en el diseño, se hizo un breve estudio de las características de la potencia en alterna para determinar qué hardware se debía escoger para llevar a cabo estas mediciones.
En corriente alterna, a un circuito se le aplica una tensión sinusoidal:

$$v(t) = V_0 * \sin(w * t)$$

Que provocará una corriente alterna desfasada $\emptyset$ grados respecto al voltaje:

$$i(t) = I_0 * \sin(w * t - \emptyset)$$

La potencia será el producto de las expresiones de la intensidad y de la corriente:

$$p(t) = I_0 * V_0 * \sin(w * t) * \sin(w * t - \emptyset)$$

Aplicando la identidad trigonométrica se puede analizar más a fondo las componentes de la potencia:

$$\sin(x) * \cos(y) = \frac{1}{2} * [\cos(x - y) + \cos(x + y)]$$

Por lo tanto:

$$p(t) = V_o * I_0 * \frac{1}{2} * \cos(\emptyset) - V_o * I_0 * \frac{1}{2} * \cos(2 * w * t - \emptyset)$$

De la anterior expresión se puede extraer una primera componente no dependiente del tiempo y una segunda componente, que sí depende del

tiempo. A la primera parte se la conoce como potencia reactiva o potencia real, y a la segunda parte como potencia imaginaria o potencia fluctuante.
En el caso de este proyecto, la potencia que marcará el consumo que se desea medir será la potencia reactiva o potencia real. A continuación se describen algunos de los componentes estudiados para la medición del consumo.

2.4.2 Circuito integrado STPM10

El circuito integrado STPM10 [2] es fabricado por ST Microelectronics cuyo objetivo es medir energía activa, reactiva y aparente.
Este circuito integrado está diseñado para ser utilizado en sistemas que cuenten con un microprocesador. El circuito implementado en el STPM se puede dividir en dos partes, una analógica y una digital.
La parte analógica cuenta con conversores analógico-digitales y reguladores de voltaje que permitan acondicionar la señal obtenida de la red eléctrica para poder ser tratada por la parte digital.
La parte digital la forman el sistema de control y procesado de señales, un oscilador así como una interfaz de salida de datos SPI (Serial Peripheral Interface).
Este circuito integrado también cuenta con una pequeña memoria interna volátil, controlada a través de la interfaz SPI cuyo propósito es almacenar parámetros de configuración y calibración del dispositivo. STPM10 se fabrica utilizando el encapsulado TSSOP20.
Sin embargo este dispositivo no es autónomo, ya que no puede ser conectado directamente a las líneas de la que se tomarán datos de corriente y voltaje. En lugar de ello, necesita de la incorporación de sensores externos de corriente y de voltaje. A continuación (figura 2.13) se muestra una aplicación típica que implemente el circuito de acondicionamiento necesario para el correcto funcionamiento del STPM10 así como los sensores de corriente y voltaje necesarios.

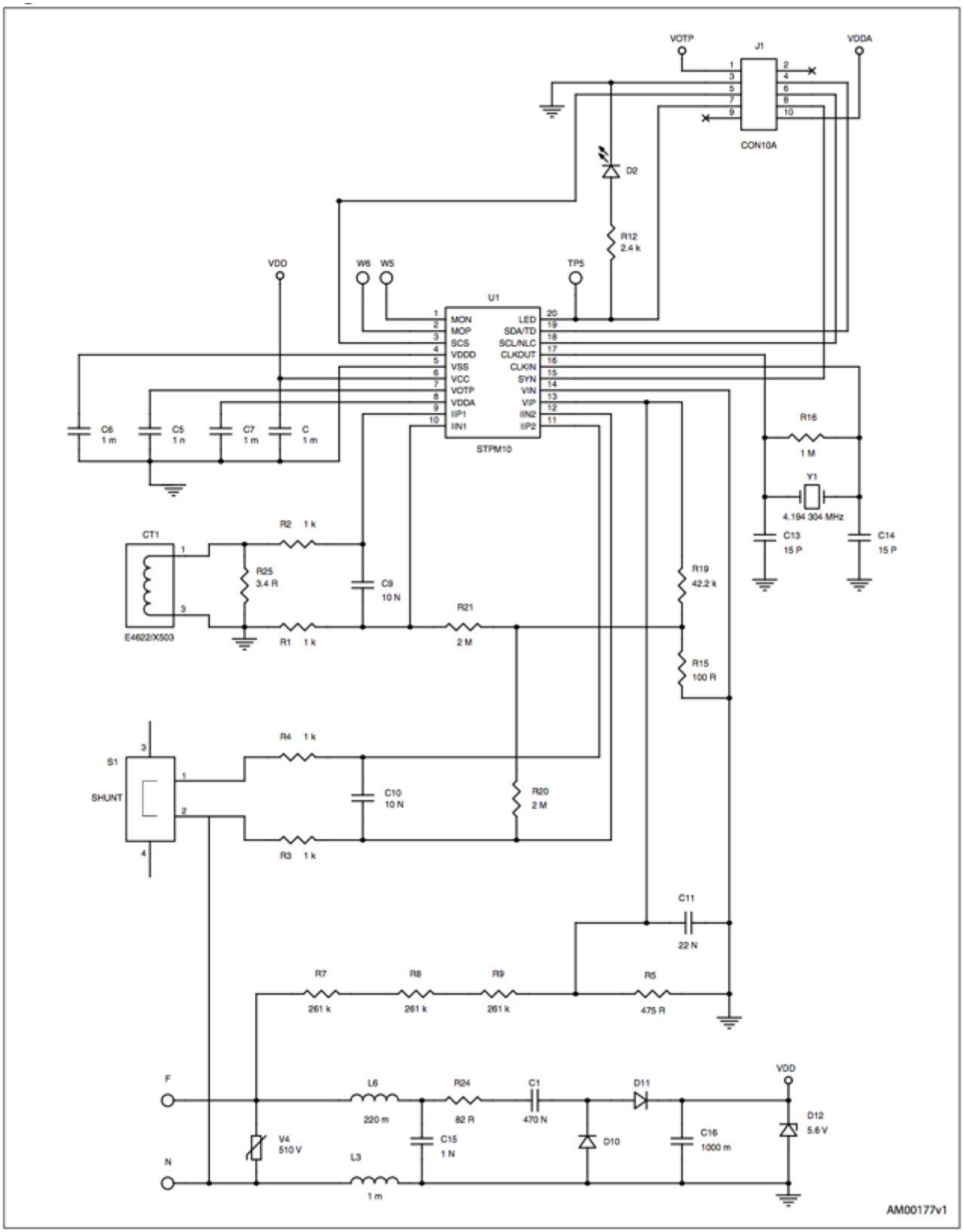


Figura 2.13. Aplicación del circuito integrado STPM10 (obtenida de la hoja de datos)

El STPM10 posee unos registros internos volátiles en los que almacena la información de las mediciones realizadas. Dispone de registros que actúan como contadores de energía activa, reactiva y aparente. En la figura 2.14 se muestra el formato de estos registros.

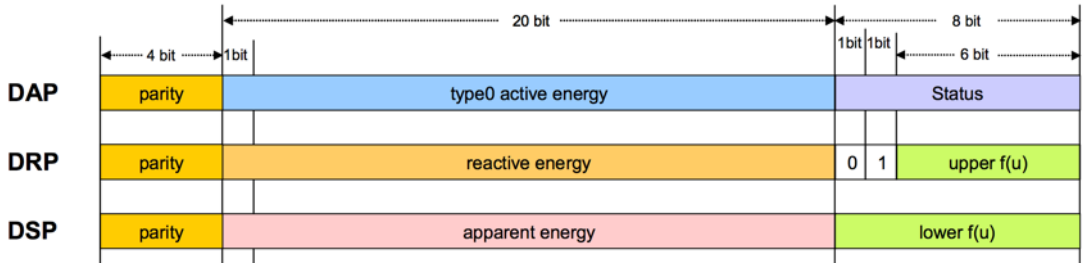


Figura 2.14. Formato de registros para almacenamiento de datos en STPM10

Para acceder a estos registros hay que implementar un protocolo de lectura que debería implementar las señales SCS, SYN, SCL y SDA (figura 2.15)

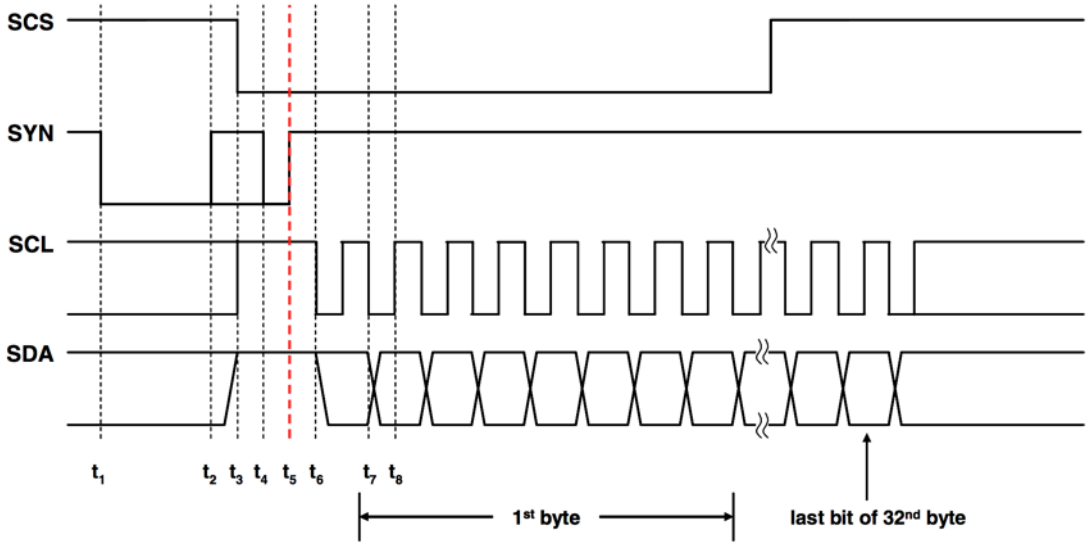


Figura 2.15. Diagrama temporal de protocolo de lectura de registros internos de STPM10

SCS, SYN y SCL son las señales de control que actúan sobre el STPM10 y servirían para indicar cuál de los registros se quiere leer. Después de haber seleccionado a qué registro se quiere acceder el STPM10 devolverá los bits que componen dicho registro al ritmo del reloj SCL por el puerto de salida que contiene la señal SDA.

2.4.3 Medidor de consumo energético DSS5188

Se trata de un medidor de energía reactiva con interfaz de salida de pulsos en función de la energía medida. El atractivo de este dispositivo reside en que es totalmente autónomo. Con este medidor no sería necesaria la implementación de un circuito de acondicionamiento, no sería necesario añadir sensores de corriente externos y tampoco sería necesario añadir un microprocesador que implementara el protocolo de comunicación para la extracción de datos con las

medidas de consumo. La información acerca del consumo energético viene dada con una interfaz de pulsos de salida en función de la energía consumida. En concreto, este dispositivo produce un pulso de salida cada vez que se consuman 0,5Wh. En la tabla 2.6 se muestran sus principales características técnicas:

Temperatura de funcionamiento	-10 a +50ºC
Temperatura de almacenamiento	-30 a +70ºC
Voltaje de funcionamiento	161 a 300V
Corriente de funcionamiento	0 a 30A
Pulsos de salida	1 imp/ 0.5Wh

Tabla 2.6. Características técnicas de medidor de consumo DSS5188

A continuación (figura 2.16) se muestran los puertos de conexión del DSS5188

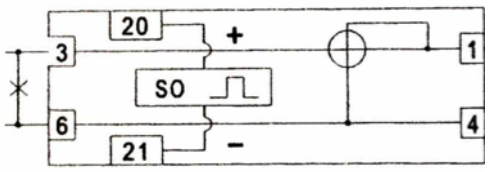


Figura 2.16. Vista superior de DSS5188. Puetos de conexión.

En la siguiente figura (2.17) se muestra un diagrama con las conexiones requeridas para que este dispositivo pueda llevar a cabo mediciones de consumo energético:

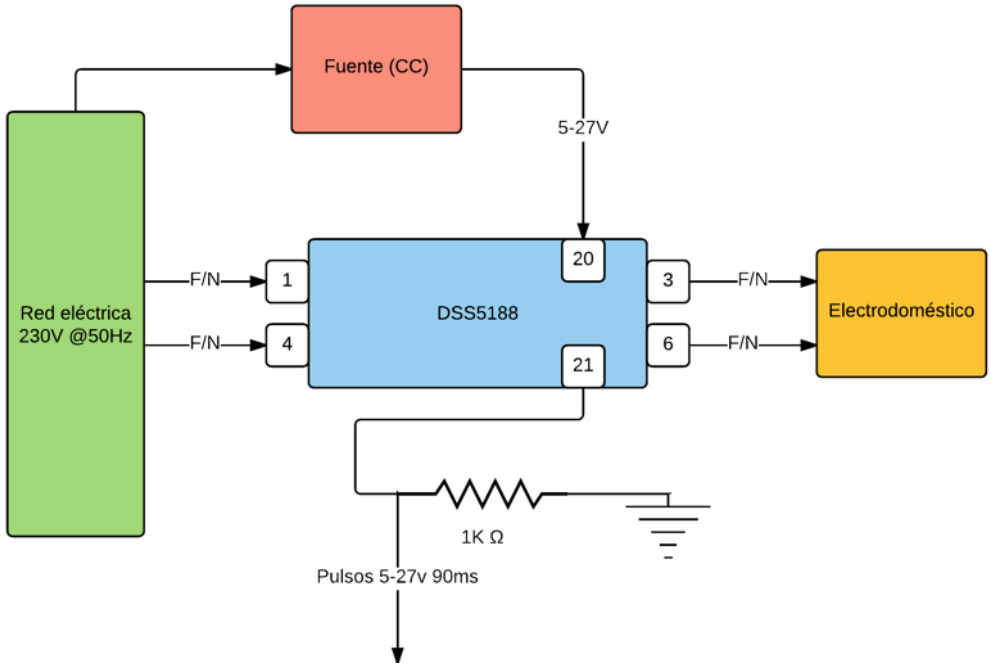


Figura 2.17. Diagrama de conexiones necesario para medir consumo de un electrodoméstico utilizando el DSS5188 (vista superior)

Se requiere de la apertura de los cables fase y neutro para intercalar el dispositivo. Una vez el dispositivo se haya intercalado, la energía eléctrica fluirá a través de él y este será capaz de realizar mediciones. Los cables fase y neutro que provienen directamente de la red eléctrica se deben conectar a los puertos 1 y 4. Los cables de salida del dispositivo, que irán a la carga que se desea alimentar saldrán de los puertos 3 y 6.

Para comunicar las medidas, el DSS5188 cuenta con un circuito que proporciona una interfaz de salida de pulsos. Este circuito es totalmente independiente del resto del dispositivo y debe alimentarse aparte. Para el correcto funcionamiento de la interfaz de salida de pulsos, se debe conectar una fuente de corriente continua de 5 a 27 V en los puertos 20 y 21. Los pulsos se obtendrán a partir del puerto 21, al que se deberá conectar una entrada digital que sea capaz de detectarlos, y una resistencia de pull down.

2.4.4 Conclusiones sobre medidores de consumo estudiados

Después de haber estudiado en detalle las características y funcionamiento de los medidores STPM10 y DSS5188, se llega a la conclusión de que el DSS5188 finalmente será el hardware que se utilizará para medir el consumo energético en cada nodo.
Aunque el STPM10 es un hardware más sofisticado y puede proporcionar datos sobre el consumo energético más detallados que el DSS5188, estos datos en el caso de este proyecto son innecesarios. Además, el DSS5188 resulta ser mucho más integrable en un diseño que contará con un módulo XBee. Esto se debe principalmente a que el STPM10 requiere de un sistema embebido con un microprocesador que sea capaz de ejecutar un script que implemente el protocolo de lectura para acceder a los registros internos. Sin embargo, el DSS5188, no requiere de tanta potencia de procesamiento para comunicarse con el exterior, ya que su interfaz de salida de pulsos puede ser interpretada por un módulo XBee (configurando una de sus E/S digitales en modo ATIC).

2.5 Hardware de control y procesado de datos.

Hasta ahora se ha decidido qué tecnologías y dispositivos serán los utilizados para desarrollar los nodos remotos. En cada nodo remoto habrá un medidor de consumo DSS5188 cuyas lecturas serán recogidas por un módulo XBee, que también se encargará de transmitir inalámbricamente. En esta sección se estudiarán sistemas embebidos que sean capaz de recibir las medidas enviadas por un módulo XBee, procesarlas y enviarlas a la plataforma web que se encargará de mostrarlas. Básicamente, este sistema deberá cumplir los siguientes requisitos:

- Debe ser capaz de comunicarse por UART con un módulo XBee receptor.
- Debe ser lo suficientemente potente para ejecutar un script de procesado de datos.
- Debe tener conexión a internet para enviar los datos procesados a la base de datos de la aplicación web.

Una vez establecidos los requisitos de la plataforma, se han estudiado dos sistemas embebidos: BeagleBone y Raspberry Pi.

Las características hardware más relevantes de estos dos sistemas embebidos se pueden resumir en la siguiente tabla (tabla 2.7) comparativa:

	BeableBone	Raspberry
Procesador	1GHz TI Sitara AM3359 ARM Cortex A8	700 MHz ARM1176JZFS
RAM	512 MB DDR3L @ 400 MHz	512 MB SDRAM @ 400 MHz
Almacenamiento	2 GB on-board eMMC, MicroSD	SD
Salidas de vídeo	1 Micro-HDMI	1 HDMI, 1 Composite
Salidas de audio	Stereo over HDMI	Stereo over HDMI, Stereo from 3.5 mm jack
Sistemas operativos	Angstrom(Default), buntu, Android, ArchLinux, Gentoo, Minix, RISC OS, others…	Raspbian (Recommended), Ubuntu, Android, ArchLinux, FreeBSD, Fedora, RISC OS, others…
Consumo	210-460 mA @ 5V under varying conditions	150-350 mA @ 5V under varying conditions
Puertos	1 USB Host, 1 Mini-USB Client, 1 10/100 Mbps Ethernet	2 USB Hosts, 1 Micro-USB Power, 1 10/100 Mbps Ethernet, RPi camera connector

Tabla2.7. Comparativa entre BeagleBone Black y Raspberry Pi B

Aunque la BeagleBone es en general, más potente que la Raspberry Pi en términos de procesador y memoria, la Raspberry Pi satisface los requerimientos del sistema. Es más, Raspberry Pi dispone de una versión más

barata llamada Raspberry Pi A que también cumple los requerimientos. Dado que se dispone de Raspberry Pi A en el HCTLab, se decidió utilizar este sistema como unidad central. A continuación se hace un estudio detallado de las capacidades de Raspberry Pi A.

2.5.1 Descipción detallada de Raspberry Pi [3]

2.5.1.2 Hardware.

La Raspberry Pi A (figura 2.18) es un ordenador de placa reducida con las siguientes características:

Figura 2.18. Raspberry Pi A

- CPU ARM 1176JZF-S a 700 MHz.
- Juego de instrucciones RISC de 32 bits.
- GPU (Unidad de procesamiento gráfico) Broadcom VideoCore IV
- Memoria (SDRAM) de 256 MB (compartidos con la GPU)
- 1 puerto USB
- Entradas de vídeo: conector MIPI CSI que permite instalar un módulo de cámara desarrollado por la RPF
- Salidas de vídeo: Conector RCA (PAL y NTSC), HDMI e interfaz DSI para un panel LCD
- Salidas de audio: conector de 3,5mm, HDMI
- Almacenamiento: tarjetas SD/MMC/ranura para SDIO
- Periféricos de bajo nivel: 8 GPIO, SPI, I2C y UART.
- Sistemas operativos: GNU/Linux: Raspbian (debían hecho a medida), Fedora (Pidora), Arch Linux.

2.5.1.2.1 Alimentación.

Raspberry se alimenta con 5V mediante un conector micro USB. La corriente que la Raspberry requerirá dependerá de qué periféricos se le conecten. Si no

se conecta ningún periférico, sólo requerirá 500mA, siendo la máxima corriente admitida de 1A.
Si se necesitara conectar algún dispositivo USB que requiera más de 1A, se deberá conectar a un USB hub con alimentación propia.
Los periféricos de bajo nivel GPIO sólo podrán entregar 500mA distribuidos entre todos los pines para evitar dañar la Raspberry, un solo GPIO como máximo puede entregar 16mA.

2.5.1.2.2 Puerto USB.

El puerto USB 2.0 se encuentra directamente conectado al microprocesador y permite la conexión de periféricos como teclados, ratones o webcams.
Sin embargo, hay algunas diferencias entre el puerto USB de una Raspberry Pi y el USB de un ordenador convencional, siendo el de la Raspberry Pi un "OTG" (on-the-go), una versión más simplificada enfocada a dispositivos móviles, que tiene algunas limitaciones que en ocasiones deberán ser compensadas mediante software.
El USB de Raspberry juega un papel clave en este proyecto ya que gracias a él se puede dotar a la Raspberry de conexión a internet por medio de un adaptador WiFi USB.

2.5.1.2.3 GPIO.

Raspberry Pi ofrece interfaces físicas de bajo nivel que pretenden proporcionar una conexión más directa con chips y otros circuitos más simples. Los GPIO proporcionan:

- SPI
- I2C
- UART
- Señales digitales de nivel alto a 3,3V y nivel bajo a 0V.
- Canales de alimentación a 3,3V y 5V.

Los GPIO se encuentran distribuidos en la placa en diferentes conectores de expansión.

Conector 1

La distribución de los pines se muestra en la figura 2.19.

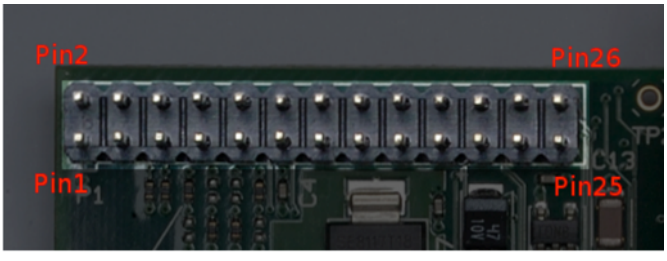


Figura 2.19. Vista superior de conector 1.

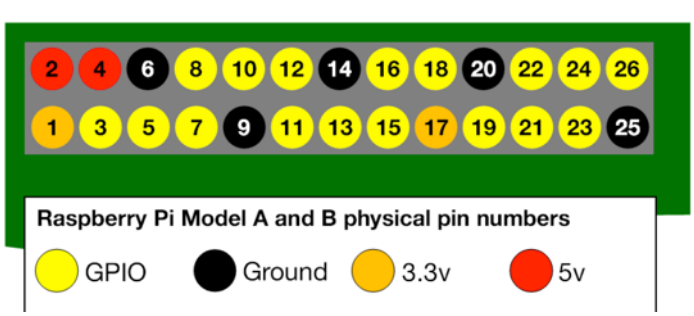


Figura 2.20. Funcionalidades de cada pin en conector 1

A la hora de utilizar los GPIO, hay dos posibles numeraciones para referirse a ellos:

- GPIO.BOARD: numeración física (orden en el cabezal, indicado con serigrafías en la placa)
- GPIO.BCM: numeración relacionada con el canal SOC del microprocesador.

En la tabla 2.8 se muestran las funciones que pueden proporcionar cada pin:

GPIO BOARD	GPIO BCM	Función alternativa
1	3,3V (max 50mA)	
2	5V	
3	2 (1K8 pull up)	I2C1_SDA
4	5V	
5	3 (1K8 pull up)	I2C1_SCL
6	GND	
7	4	
8	14	UART_TXD
9	GND	
10	15	UART_RXD
11	17	
12	18	
13	27	
14	GND	
15	22	
16	23	
17	3.3V	
18	24	
19	10	SPI0_MOSI
20	GND	
21	9	SPI0_MISO
22	25	
23	11	SPI0_SCLK
24	8	SPI_CE0
25	GND	
26	7	SPI_CE1

Tabla 2.8. Funciones de cada pin en conector 1

Conector 2.

Este conector (figura 2.21) es utilizado como VideoCore JTAG durante la producción de la placa. No puede ser usado como ARM JTAG y proporciona únicamente tres pines útiles (los demás se encuentran deshabilitados)

- Pin 1: salida de 3,3V
- Pin 7 & 8: GND.
-

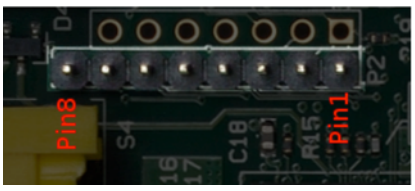


Figura 2.21. Vista superior de conector 2

Conector 3.

Este conector (figura 2.22) también se utiliza durante producción (es el LAN9512 JTAG), y todos sus pines están deshabilitados menos el Pin 7 que proporciona una conexión a GND.

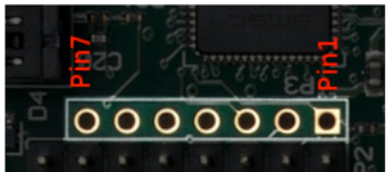


Figura 2.22. Vista superior conector 3

Conector 5.

GPIO BOARD	GPIO BCM	Otras funciones
1	5 V	
2	3.3V	
3	28	I2C0_SDA
4	29	I2C0_SCL
5	30	
6	31	
7	GND	
8	GND	

Tabla 2.9. Descripción de pines en conector 5

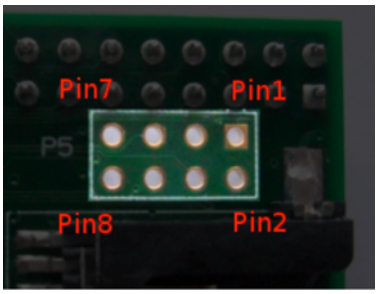


Figura 2.23. Vista superior conector 5

2.5.1.3 Software.

A continuación se habla de los sistemas operativos compatibles con la Raspberry.

- Raspbian: es un sistema operativo de licencia libre basado en Debian y optimizado para las carácterstica hardware de la Raspberry. Gracias a la comunidad de desarrolladores cuenta con 35.000 paquetes software disponibles para instalar gratuitamente. Dispone de una interfaz gráfica para facilitar la configuración del sistema operativo en su instalación sin tener que modificar los archivos de configuración manualmente. Esta herramienta gráfica incorpora un gestor de particiones, permite la configuración del teclado, aplicar overclock, etc.
- Fedora remix: distribución Linux para Raspberry Pi.
- OpenELEC
- RaspBMC: versión adaptada de XBMC para Raspberry. Es utilizado en otros sistemas reducidos como Vero y Apple TV.

- Risc OS: a diferencia que todos los anteriores, Risc OS no es una versión de Linux (ni de Windows).
- ArchLinux.

2.6 Aplicación web.

Para el desarrollo de la aplicación web se estudió la tecnología Meteor Js. Esta tecnología proporciona una solución completa para el desarrollo de aplicaciones web (incluyendo integración de bases de datos no relacionales).

2.6.1 Meteor.js [4][5]

Las aplicaciones web estaban inicialmente diseñadas de manera que el servidor enviaba una pantalla al cliente cada vez que tenía lugar alguna interacción con la web en el lado del cliente. Hoy en día, las aplicaciones web funcionan con JavaScript en el cliente para evitar esto. Las aplicaciones son reactivas, evitan refrescar la página al completo cada vez que se quiera realizar un cambio y cualquier cambio realizado en cualquier cliente se reflejará en el servidor y en todos los ordenadores que estén accediendo a la misma aplicación web.

Meteor es una plataforma para crear aplicaciones web en tiempo real construida sobre Node.js. Se localiza entre la base de datos de la aplicación y su interfaz de usuario, y se asegura de que ambas partes estén sincronizadas.
Entre las principales ventajas de Meteor, cabe destacar que permite una muy rápida creación de aplicaciones web altamente compatibles con las principales plataformas (web, Android, iOS).
Está altamente integrado con MongoDB y usa el protocolo DDP (distributed data protocol) para actualizar la base de datos en el lado del servidor y a su vez sincronizar dichos cambios entre los clientes.
Las principales características de Meteor son:

- "Data on the wire": en vez de enviar HTML al cliente, Meteor sólo envía los datos necesarios para cambiar la parte de la web que ha cambiado. Gracias a evitar recargar la web entera, se pueden construir webs con una muy baja latencia de carga.
- "One lenguage": tanto en el servidor como en el cliente se utiliza JavaScript, con el cual también se puede acceder a las bases de datos. Con un único lenguaje se programa completamente la parte lógica de la web.
- "Database Everywhere": a fin de agilizar el acceso a la base de datos, ésta no solo se encuentra en el servidor, si no que Meteor crea y envía al cliente una caché con la parte necesaria de la base de datos para dicho cliente.
- "Full stack reactivity": todas las capas, desde la base de datos a la plantilla HTML, se actualiza automáticamente cuando es necesario sin necesidad de refrescar la página web.

- Simplicity equals productivity: Meteor fue diseñado para ser fácil de aprender, incluso para principiantes en desarrollo web.

2.6.1.2 Estructura de datos de MeteorJs.

Meteor es una mezcla de código JavaScript en el lado del cliente que se ejecuta en el navegador, código JavaScript que se ejecuta en el lado del servidor sobre Node.js, y los demás recursos estáticos necesarios para construir una aplicación web como las plantillas HTML, las hojas de estilo CSS y otros recursos estáticos como archivos o fotos.

Meteor es muy flexible a la hora de la estructuración de los archivos. Automáticamente carga todos los archivos contenidos en la carpeta de la aplicación, por lo que no es necesario incluir etiquetas en el HTML como <script> ó <link> para incluir JavaScript o CSS.

No es preciso una estructuración de directorios concreta, siendo únicamente necesario incluir todos los archivos de la aplicación en la carpeta de la aplicación, sin embargo, los nombres de los archivos y directorios dentro del proyecto puede afectar al orden en que se cargan, dónde se cargan y otras características. Los siguientes directorios reciben un trato especial

- /client: contiene todos los datos necesarios en el cliente (HTML, CSS y código JavaScript).
- /server: cualquier archivo contenido en esta carpeta no será enviado al cliente. Sirve para ocultar al cliente información como datos de contraseñas o mecanismos de autentificación.
- /public: los datos contenidos en esta carpeta serán enviados tal cual al cliente, se utiliza para recursos como imágenes.
- /private: sólo se puede acceder a los recursos de esta carpeta mediante código en el servidor.

```
1  if (Meteor.isClient)
2   {
3
4      //código JavaScript cliente
5
6   };
7
8  if (Meteor.isServer)
9  {
10
11     //código JavaScript servidor
12
13 };
```

Figura 2.24: Código necesario para distinguir qué se ejecutará en el cliente y qué se ejecutará en el servidor si no se estructura la aplicación por carpetas.

Este sistema de directorios no es obligatorio, no obstante, si se desea omitir la creación de las carpetas /client y /server, el código JavaScript debe estar estructurado de la siguiente manera:

2.6.1.3 Reactividad en MeteorJS

Meteor es un framework reactivo ya que puede crear aplicaciones web dinámicas que cambian en tiempo real sin necesidad de refrescar la página web. No obstante, para evitar tener que ejecutar todo el código de la aplicación cada vez que se produce un cambio en alguno de los datos de los que esta depende, la reactividad se limita a áreas específicas llamadas computaciones. Las computaciones son partes del código que se ejecutan cada vez que cambia una de las fuentes de datos reactivas de las que depende esa parte de código.
Las fuentes de datos reactivas a su vez son monitorizan todas las computaciones de las que dependen, para avisarlas cuando su valor cambie y estas puedan volver a ejecutarse. Para esto usa la función `invalitate ()`.

Gracias a las computaciones y a las fuentes de datos reactivas, las aplicaciones desarrolladas con MeteorJS son altamente dinámicas, ya que si una parte del código depende de una variable o datos de una base de datos, el resultado que esta parte de código produce será automáticamente recalculado si los datos de los que depende cambian. A continuación se muestra un ejemplo de programación reactiva (figura 2.25):

```
1  Tracker.autorun(function ()
2  {
3    Meteor.subscribe("messages", Session.get("currentRoomId"));
4  });
```

Figura 2.25. Ejemplo de programación reactiva.

En este ejemplo, se crea una computación encerrando un fragmento de código en el bloque `Tracker.autorun`. En este caso, la fuente de datos reactiva de la que depende la computación es `Session.get ("currentRoomId")`, que podrá provocar que la computación se ejecute de nuevo al cambiar su valor.
Meteor dispone de varias funciones que permiten crear computaciones. Estas funciones son:

- `Template`
- `Tracker.autorun`
- `Blaze.render` y `Blaze.renderWithData`

Y como fuentes de datos reactivas pueden servir los siguientes elementos:

- Variables de Session
- Queries en Collections
- Meteor.status
- El método ready()
- Meteor.user
- Meteor.userId
- Meteor.loggingIn

2.6.1.4 Plantillas HTML.

Meteor emplea la tecnología Blaze para permitir que las plantillas HTML se actualicen automáticamente atendiendo a cambios en los datos que estas contienen. Gracias a esta tecnología con simplemente desarrollar las plantillas HTML es suficiente, pues Meteor se encargará de actualizar los datos contenidos en ellas cuando sea necesario.
Para crear una plantilla html, simplemente se debe de incluir en un archivo con extensión .html y dentro de la etiqueta <template>. Se puede insertar plantillas dentro de plantillas usando el operador {{> myTemplate }}.

Mediante la sentencia {{> myTemplate }} se inserta la pantilla "myTemplate" dentro de otra plantilla. Las plantillas pueden ser incluidas más de una vez, de hecho, uno de los propósitos generales de las plantillas es tener que evitar replicar código HTML.

La manera más fácil de introducir datos reactivos dentro de una plantilla es utilizando "helpers". A continuación se muestra un ejemplo sobre las plantillas (figura 2.26):

```
1
2  <head>
3    <title>My website!</title>
4  </head>
5
6
7  <body>
8    <h1>Hello!</h1>
9    {{> welcomePage}}
10 </body>
11
12
13 <template name="welcomePage">
14   <p>Welcome to my website!</p>
15 </template>
```

Figura 2.26. Ejemplo de uso de datos reactivos en html

En la línea 6 se crea una plantilla de nombre "welcomePage", la cual se inserta en la plantilla "body" (nombre reservado para plantilla por defecto) en la línea 9.

La sintaxis {{…}} en la que se encuentra el nombre de la plantilla, es parte de un lenguaje denominado Spacebars que meteor usa para añadir funcionalidad a el HTML. Haciendo uso de los SpaceBars, se pueden incluir plantillas y también acceder a datos obtenidos por medio de los helpers. Los helpers se escriben en JavaScript y pueden ser tanto variables como funciones.

2.6.1.5 Helpers

Los helpers proporcionan datos reactivos para dotar de un carácter dinámico a las plantillas HTML. Sólo se encuentran disponibles en el lado del cliente.

Por ejemplo, para definir un helper que consista en una variable llamada "name" para la plantilla llamada "nametag" se haría de la siguiente manera (figura 2.27):

```
1  Template.nametag.helpers({
2    name: "Ben Bitdiddle"
3  });
```

Figura 2.27. Ejemplo de creación de un helper.

Y podría ser accedido desde el HTML haciendo uso de los SpaceBars (figura 2.28).

```
1  <template name="nametag">
2    <p>My name is {{name}}.</p>
3  </template>
```

Figura 2.28: Ejemplo de uso de un helper en html usando Spacebars ({{}})

Los helpers también pueden ser funciones que reciben argumentos, como en el siguiente caso (figura 2.29):

```
Template.nametag.helpers({
  name: function (id)
  {
    if (id==1)
    {
        return "Ben Bitdiddle"
    }
    if (id==2)
    {
        return "Javier Fernández"
    }

  }
});
```

Figura 2.30. Ejemplo de creación de helpers que reciben parámetros por argumento

Cuya llamada desde la plantilla HTML sería (figura 2.31):

```
<template name="nametag">
  <p>My name is {{name 2}}.</p>
</template>
```

Figura 2.31. Ejemplo de uso de helper que recibe parámetros por argumento en el html

Siendo el número dos el argumento que el helper espera recibir.

2.6.1.6 SpaceBars

Los SpaceBars, aparte de para introducir plantillas en HTML, proporcionan un conjunto de estructuras de control que pueden ser usadas para acceder a los datos procedentes de los helpers. Algunas de estas son:

- {{#each data}} … {{each}}: se usa para iterar sobre los elementos de data y mostrar el bloque de código HTML que se encuentra dentro de la sentencia
- {{#if data}} … {{else}} …{{/if}}: Si se data es verdadero, se muestra el primer bloque, si es falso, se muestra el segundo bloque.

Cada bloque tiene su propio contexto de datos, siendo este un objeto a cuyas propiedades se puede acceder mediante SpaceBars. Dependiendo de la sentencia utilizada el contexto de datos será diferente. Si se usa #each para iterar un array, el contexto de datos sería los elementos de dicho array. Por ejemplo, si se tuviera un helper llamado "gente" (figura 2.32) que consiste en un array:

```
Template.nametag.helpers({

gente: [{nombre:"Javier"},{nombre:"Álvaro"}]

});
```

Figura 2.32. Creación de un helper con un array

Gracias a los SpaceBars podríamos acceder a sus elementos mediante la sentencia {{nombre}} (figura 2.33).

```
1  {{#each gente}}
2      <p> Nombre: {{nombre}} </p>
3  {{/each}}
```

Figura 2.33. Acceso a un helper compuesto por un array utilizando el elemento iterativo #each

2.6.1.7 onRendered

Las funciones añadidas con este método son llamadas una vez por cada

Template.myTemplate.onRendered *Client*

Register a function to be called when an instance of this template is inserted into the DOM.

Arguments

callback Function

A function to be added as a callback.

instancia de myTemplate cuando esta es insertada por primera vez en la web.

Figure 1

Este método es usado para integrar librerías externas que no son compatibles con la carga automática de meteor y necesitan ser inicializadas cada vez que el HTML de una plantilla es insertado en la página.

2.6.1.8 La variable Session

La variable Session es un objeto global en el cliente usada para almacenar datos de varios tipos asociados a claves (nombre de la variable). La variable Session es reactiva, así que si esta es insertada dentro de una computación como un template helper o dentro de Tracker.autorun, la parte de la plantilla que dependa de su valor se actualizará automáticamente cada vez que su valor cambie.

Las variables de Session son almacenas en la memoria del navegador y se pierden cuando se cierra la ventana.

Existe un método para dar valor a una clave (Session.set), y otro para obtener el valor de una clave (Session.get).

Session.set(key, value) *Client*

Set a variable in the session. Notify any listeners that the value has changed (eg: redraw templates, and rerun any `Tracker.autorun` computations, that called `Session.get` on this `key`.)

Arguments

key String

The key to set, eg, `selectedItem`

value EJSON-able Object or undefined

The new value for `key`

Session.get(key) *Client*

Get the value of a session variable. If inside a reactive computation, invalidate the computation the next time the value of the variable is changed by `Session.set`. This returns a clone of the session value, so if it's an object or an array, mutating the returned value has no effect on the value stored in the session.

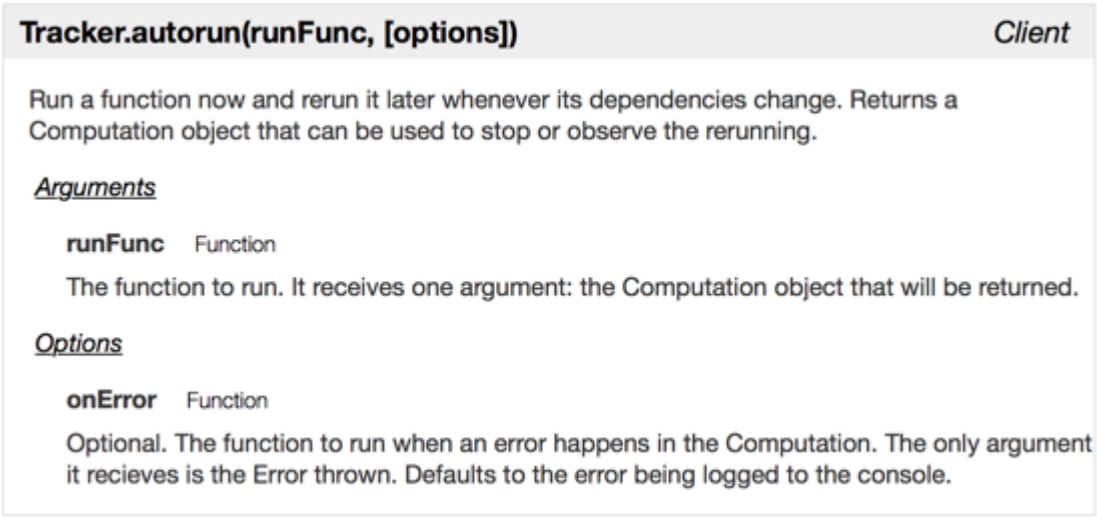
Arguments

key String

The name of the session variable to return

2.6.1.9 Tracker.

El método tracker se utiliza para crear computaciones que dependen de datos reactivos. Gracias al Tracker se pueden crear plantillas y funciones reactivas que se ejecutarán automáticamente cuando los datos reactivos de los que dependen cambien. Las dependencias entre las computaciones y los datos reactivos no tienen que ser creadas manualmente, una vez creada una computación con Tracker.autorun, cada vez que se llama a una función que



Tracker.autorun(runFunc, [options]) *Client*

Run a function now and rerun it later whenever its dependencies change. Returns a Computation object that can be used to stop or observe the rerunning.

Arguments

runFunc Function

The function to run. It receives one argument: the Computation object that will be returned.

Options

onError Function

Optional. The function to run when an error happens in the Computation. The only argument it recieves is the Error thrown. Defaults to the error being logged to the console.

devuelve datos, Tracker registra automáticamente estas dependencias. Este es el mismo mecanismo que utilizan las plantillas para cambiar automáticamente cuando alguno de los datos de sus helpers cambia. Se suele usar para producir cambios a nivel de variables, bases de datos que no se muestran directamente en la interfaz web (a diferencia de los helpers) pero que deben ser modificadas.

2.6.1.10 Almacenamiento de datos en aplicaciones Meteor.

Meteor hace uso de tres formas diferentes para almacenar los datos de las aplicaciones:

- Memoria del navegador: para almacenar datos no permanentes como variables en JavaScript, que se borrarán cuando se cierre el navegador.
- Almacén del navegador: los navegadores pueden almacenar datos de forma permanente utilizando cookies. Sin embargo, aunque sean permanentes, no es sencillo compartirlo con otros usuarios.
- Base de datos en el servidor: sistema de datos más eficaz para almacenar datos de forma permanente y hacerlos disponibles para más de un usuario.

Meteor hace uso de las tres formas, a veces sincronizando datos de un lugar a otro, pero la fuente de datos que contiene la copia maestra es la base de datos en el servidor.

2.6.1.11 Colecciones.

Meteor hace uso de una estructura de datos especial llamada colecciones para almacenar los datos de sus aplicaciones. Las colecciones son el eje central de las aplicaciones Meteor, ya que permiten guardar los datos de forma permanente en el servidor mediante una base de datos MongoDB y sincronizar los datos con el navegador de cada usuario conectado en tiempo real.

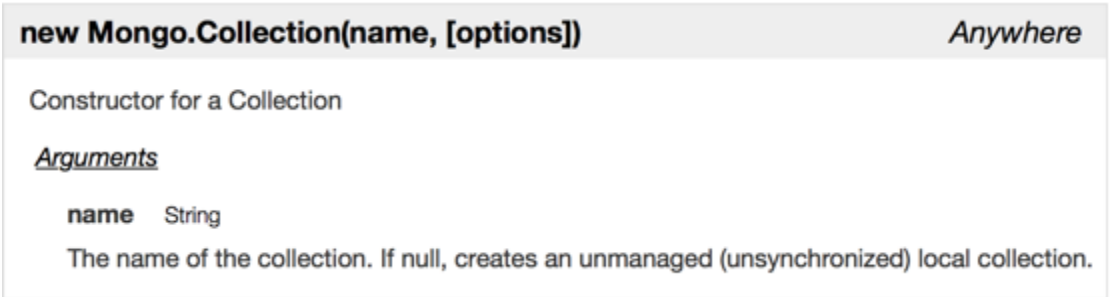
Colecciones en el lado del cliente.

Cuando se declara una colección en el cliente, se crea una caché local de la colección real de Mongo dentro del navegador. Ya que lo normal es que cada cliente sólo tenga acceso a cierta parte de la base de datos, esta caché contiene un subconjunto de datos de la colección Mongo y ofrece un acceso muy rápido. Esto es debido a que ya que esta caché está almacenada en la memoria del ordenador, y cuando se accede a ella no hay que conectar con la base de datos en el servidor ya que los datos ya están precargados. Después de que se inserte un dato en una colección local en la memoria de un navegador, Meteor se encarga de almacenarlo automáticamente en la colección del servidor para que pueda ser distribuido a todos los demás clientes que requieran acceso a ese dato.

2.6.1.11.1 MongoDB

MongoDB es un sistema de base de datos NoSQL orientado a documentos desarrollado bajo el concepto de código abierto. A diferencia de las bases de datos SQL, que guardan datos en tablas, MongoDB guarda estructuras de datos en archivos de tipo JSON con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.
Meteor incluye entre sus herramientas un conjunto de métodos que permite acceder a estas bases de datos utilizando JavaScript.

2.6.1.11.2 Crear una colección

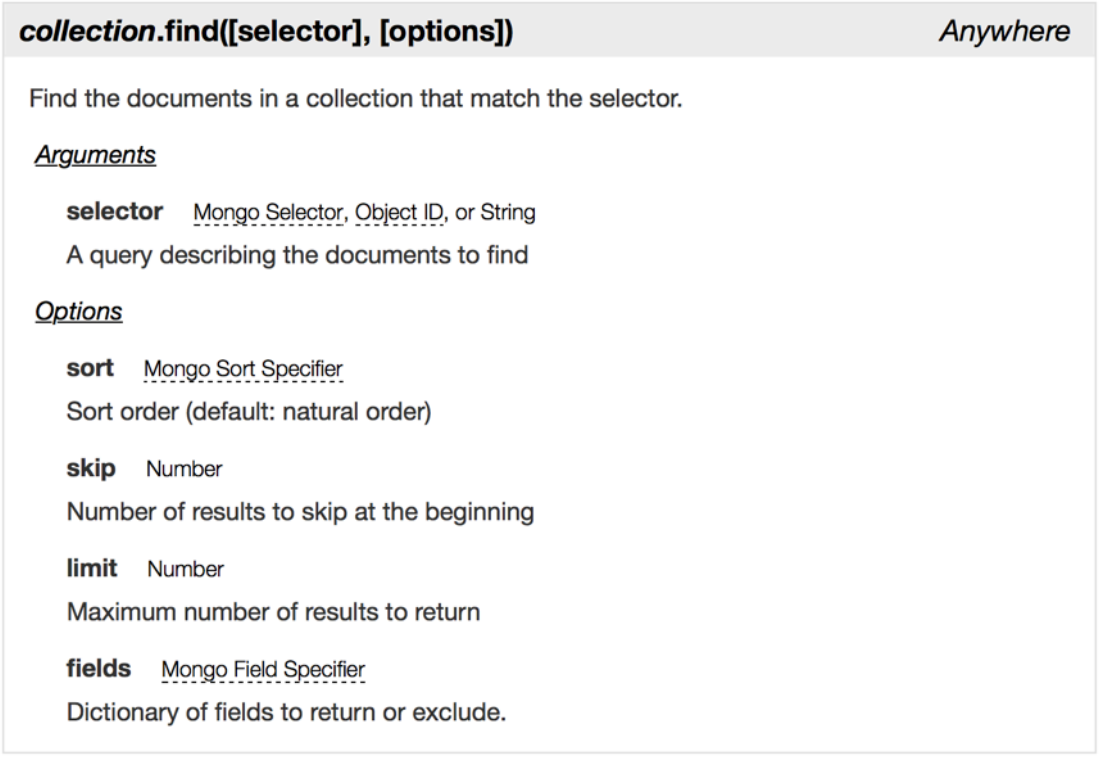


Al llamar al constructor `Mongo.Collection` se crea un objeto que actúa como una colección MongoDb. Si se le pasa un nombre se creará una colección

(persistente) en la base de datos, y si no se creará una colección en la memoria del navegador que se perderá al cerrar el navegador.

2.6.1.11.3 Querys en Meteor.

Meteor dispone de métodos como "`collection.find`" que permiten realizar querys en la base de datos MongoDB.



***collection*.find([selector], [options])** *Anywhere*

Find the documents in a collection that match the selector.

Arguments

selector Mongo Selector, Object ID, or String

A query describing the documents to find

Options

sort Mongo Sort Specifier

Sort order (default: natural order)

skip Number

Number of results to skip at the beginning

limit Number

Maximum number of results to return

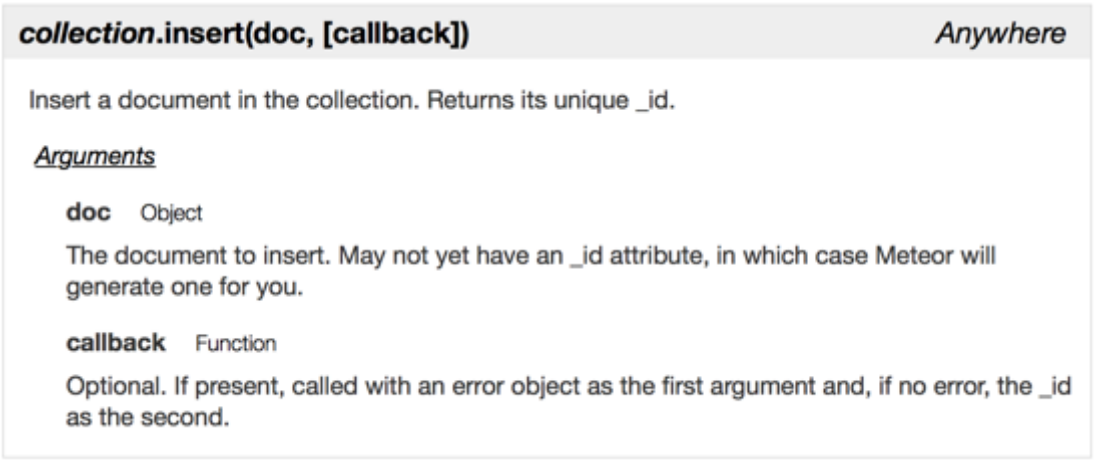
fields Mongo Field Specifier

Dictionary of fields to return or exclude.

Este método devuelve un objeto de tipo MongoDB *cursor* que contiene una lista de documentos con los resultados de la query. Este objeto puede ser accesible en las plantillas mediante Helpers.

2.6.1.11.4 Insertar elementos en colecciones.

Meteor proporciona un método para insertar elementos:



collection.insert(doc, [callback]) *Anywhere*

Insert a document in the collection. Returns its unique _id.

Arguments

doc Object

The document to insert. May not yet have an _id attribute, in which case Meteor will generate one for you.

callback Function

Optional. If present, called with an error object as the first argument and, if no error, the _id as the second.

Todos los documentos en una colección MongoDB cuentan con un campo "id" que es único y automáticamente generado si no se proporciona.

2.5.1.11.5 Otros métodos para manejo de colecciones.

Además existen otros métodos de relevancia como pueden ser:

- *collection.findOne*
- *collection.update*
- *collection.remove*
- *collection.allow*
- *collection.deny*

3. Diseño y desarrollo.

3.1 Opciones de diseño hardware contempladas.

Dadas las dos necesidades funcionales dependientes del hardware, se contemplaron dos posibles diseños para el desarrollo de los nodos:

1. Dotar a cada nodo de una funcionalidad completa, es decir, que cada nodo sea capaz de llevar a cabo la medición del consumo y de procesar y hacer accesibles los datos a través de internet (figura 3.1).
    a. Ventajas: absoluta independencia del cada nodo, siendo capaz por sí sólo de alojar los datos recogidos en una base de datos.
    b. Desventajas: para conseguir que el propio nodo sea capaz de realizar el procesado y el volcado de las medidas en una base de datos se precisaría de la incorporación de un sistema embebido con conexión a internet y con recursos necesarios para ejecutar el programa encargado del procesado de datos.

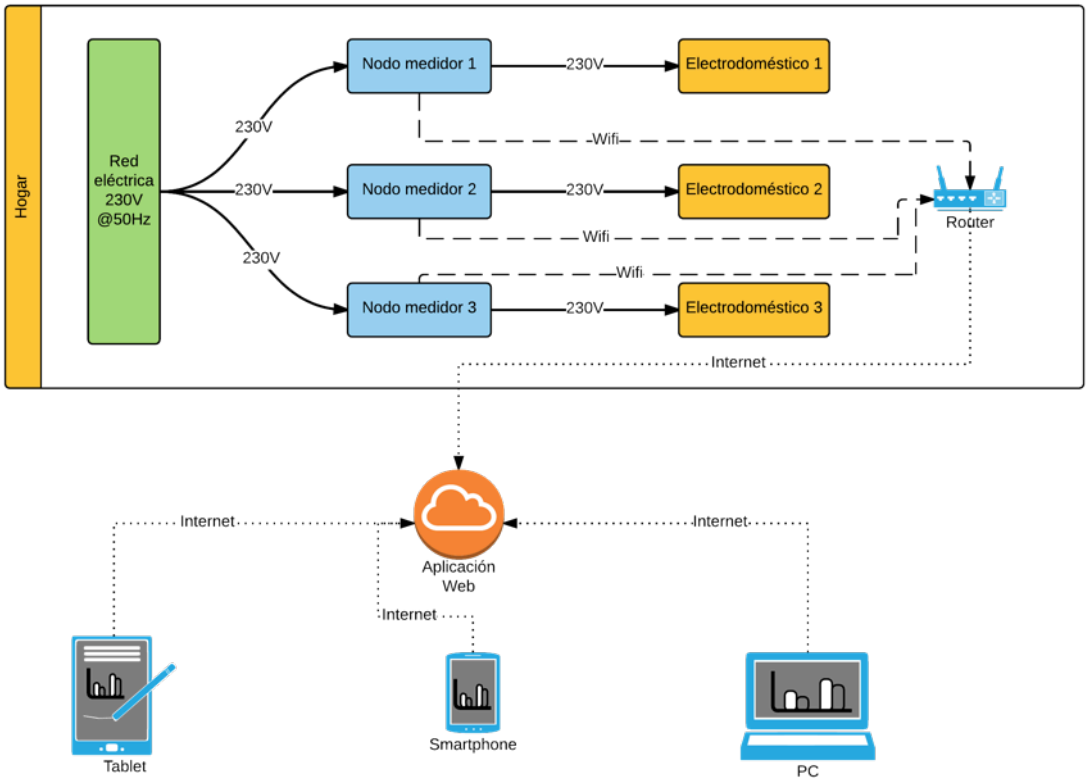


Figura 3.1.Diagrama solución 1

2. Reducir al máximo posible las funcionalidades de cada nodo. Para ello, analizando las funcionalidades hardware imprescindibles se llegaron a las siguientes conclusiones:
    a. Llevar a cabo el proceso de medición de consumo: Dada la naturaleza física del problema, se requiere contacto físico con

los cables de fase y neutro de la red eléctrica, por lo tanto, esta funcionalidad no puede ser separada de cada nodo.

b. Procesar los datos de consumo y hacerlos visibles desde un dispositivo con acceso a internet: Una vez obtenidos los datos, todo el proceso que sigue puede ser llevado a cabo en el propio nodo, pero también fuera de él, ya que no hay ningún motivo físico por el cual este proceso necesariamente tenga que ser llevado a cabo en el propio enchufe. Por lo tanto, se llegó a la conclusión de que esta funcionalidad podría ser abstraída de cada nodo. Sin embargo, adoptar esta solución implicaría dejar a cada nodo aislado ya que este ya no dispondría de conexión a internet. De esta problemática surgió la necesidad de incorporar un nuevo componente a cada nodo que fuera capaz de establecer un enlace inalámbrico para así poder acceder a los datos de las mediciones.

Las ventajas e inconvenientes de esta solución (figura 3.2) son las siguientes:

a. Ventajas: reducción del coste y la complejidad hardware de cada nodo.

b. Inconvenientes: todos los nodos son dependientes de la unidad central. Si la unidad central falla, los nodos quedan aislados.

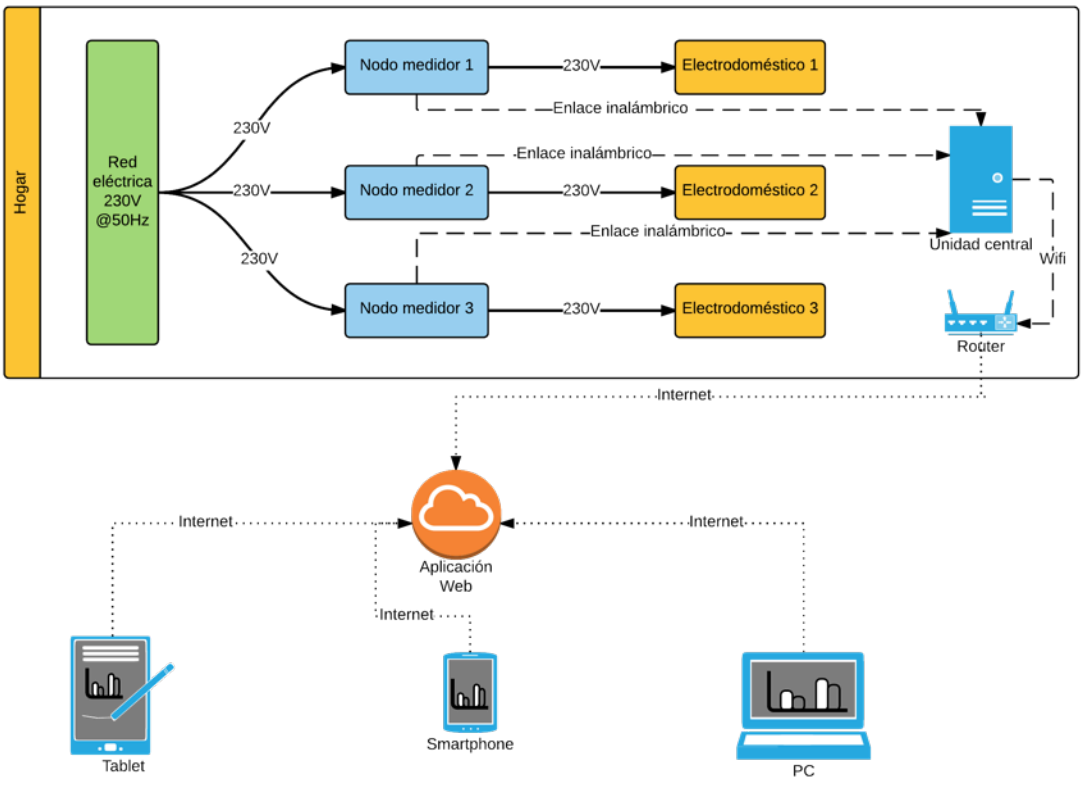


Figura 3.2. Diagrama solución 2

3.1.2 Opción elegida.

Dado que dotar a cada nodo de un sistema embebido que sea capaz de procesar los datos y hacerlos accesibles a través de internet sería sobre equipar cada nodo y aumentar notablemente su coste, se decidió sustituir el sistema embebido por un componente que fuera capaz de crear un enlace inalámbrico para comunicar al nodo con el exterior (figura 3.3). De esta manera, cada nodo enviaría sus datos a una unidad central, que se encargaría de procesar los datos y hacerlos accesibles a través de internet. Por lo tanto, la estructura del sistema completo sería de la siguiente forma:

- Unidad remota ó nodo: compuesto de el hardware encargado de llevar a cabo las medidas de consumo y de un módulo inalámbrico encargado de recoger estas medidas y transmitirlas fuera del nodo a una unidad central.
- Unidad central: compuesto de un sistema embebido capaz de comunicarse con los nodos medidores y de hacer accesible a través de internet los datos de consumo.

Dada la necesidad de incorporar una tecnología que hiciera de mediadora entre el hardware medidor de consumo y la unidad de procesamiento central (nodo central), se estudió la tecnología inalámbrica que desde un principio pareció adecuada para este propósito (ZigBee).

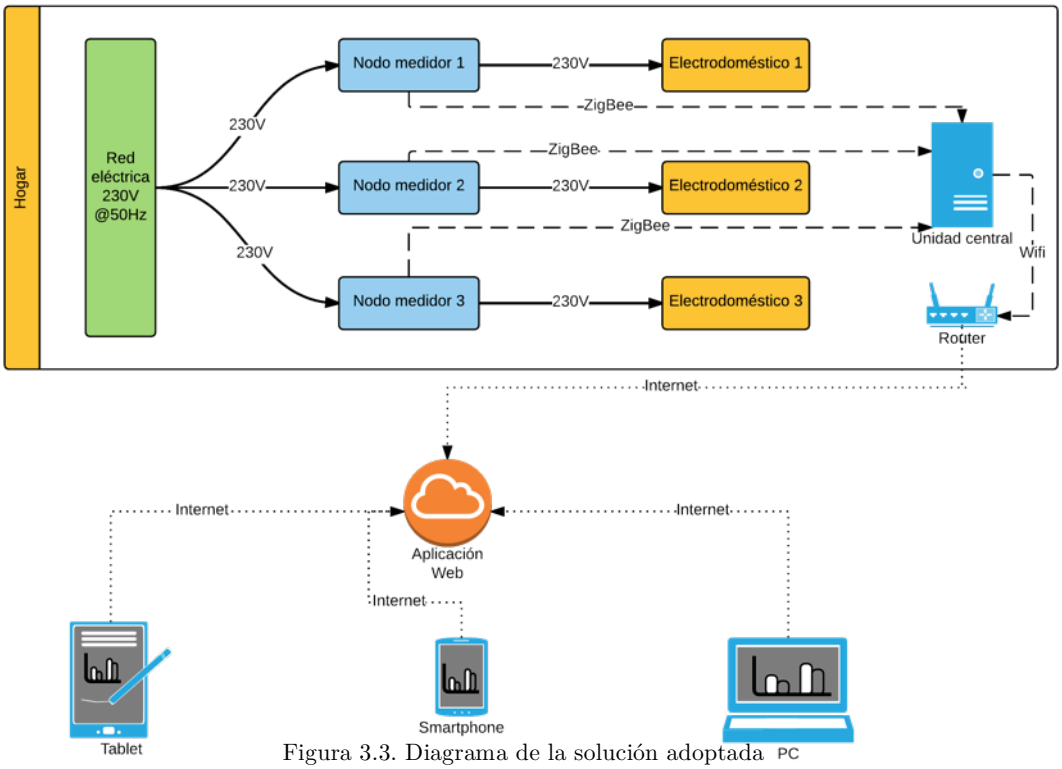


Figura 3.3. Diagrama de la solución adoptada

3.2 Red de comunicación entre nodos y unidad central.

3.2.1 Topología de red.

La comunicación de la unidad central con los nodos remotos se basa en la creación de una red ZigBee. A la hora de elegir la topología de red más adecuada, se tuvieron en cuenta los siguientes aspectos:

- Los dos únicos casos de flujo de información son de un nodo remoto a la unidad central y viceversa, y nunca será necesaria una comunicación nodo a nodo.
- Dadas las características del problema a resolver, en el que se pretende conocer el consumo de tantos dispositivos eléctricos como se desee, la red debe ser totalmente flexible a la hora añadir o eliminar nodos.

La topología que mejor satisface las necesidades planteadas es la topología en estrella (figura 3.4), cuya principal característica es la dependencia de los nodos de la unidad central. Esta dependencia tiene el principal inconveniente de que el alcance total del despliegue se encuentra condicionado por el alcance del nodo central, por el que deben pasar, y al que irán destinadas todas las tramas de la red. Sin embargo, dado el ámbito doméstico en el que se plantea el sistema, y las características de potencia de emisión y sensibilidad de los módulos XBee, esto no será un problema. Por el contrario, es ésta única dependencia de los nodos remotos respecto de la unidad central, la que proporciona a la red la versatilidad necesaria a la hora de añadir ó quitar nodos.

Figura 3.4. Topología en estrella. En beige los nodos finales, en rojo la unidad central.

3.2.2 Configuración de los módulos transmisores y receptores.

Para el despliegue de la red descrita en el apartado anterior se han utilizado módulos XBee Serie 1 Pro. Dada la topología en estrella de la red, con una única unidad central, y tantos nodos remotos como se desee, se necesita que la unidad central sea capaz de diferenciar entre cada uno de los nodos de la red.
Por ello, los dispositivos XBee que conforman la red de la que se sirve este proyecto están configurados en modo API.
El flujo de información se produce en un único sentido, de los nodos remotos a la unidad central. Los nodos remotos tomarán medidas de consumo cuando sea necesario, y las enviarán instantáneamente a la unidad central, que se encontrará en estado de espera preparada para recibir y procesar tramas API. Gracias a la estructura de las tramas API, la unidad central sabe quién envía las tramas y puede tenerlo en cuenta a la hora de proporcionar datos de consumo más detallados.
Lo primero que hay que hacer para desplegar una red ZigBee en modo API es establecer las funciones que cada módulo XBee tendrá dentro de la red. Por la topología en estrella elegida, esto se reducirá a elegir un único dispositivo coordinador de la red, y los demás como dispositivos finales.
Dado que los módulos XBee también ofrecen un direccionamiento de 16 bits, aparte de la dirección física de 64 bits, para poder usarlo necesitamos configurar esta dirección individualmente en cada nodo antes de incorporarlo a la red.
Una vez establecido el modo API como modo operativo en la red, se puede sacar partido a su sistema de tramas, que permitirán configurar los módulos XBee remotamente mediante comandos AT, así como recibir información en tramas acerca de cualquier información que la unidad central requiera.
Como ya se ha dicho, para usar el modo API primero hacer falta definir los papeles de coordinador o dispositivo final. Para ello como interfaz física para comunicarse con los módulos XBee se usaron las tarjetas de desarrollo XBIB-R-DEV y XBIB-U-DEV disponibles en el laboratorio, y como interfaz software el X-CTU

Tarjetas de desarrollo XBIB-R-DEV y XBIB-U-DEV

Ambas sirven para el mismo propósito, y aunque hay algunas pequeñas diferencias entre ambas, el principal distintivo es que la tarjeta XBIB-R-DEV (figura 3.5) cuenta con una interfaz RS-232, mientras que la XBIB-U-DEV (figura 3.6) dispone de una interfaz USB.

Figura 3.5. Tarjeta XBIB-R-DEV

Figura 3.6. Tarjeta XBIB-U_DEV

Estas tarjetas, aparte de proporcionar una interfaz física entre los módulo XBee y el ordenador, también proporcionan otros elementos como reguladores de tensión para alimentar los módulos, pulsadores para reiniciar el módulo ó LEDs que permiten ver el estado de los pines configurados como E/S digitales, mostrar la actividad de los pines utilizados como UART y otras funciones.

Software X-CTU

El software X-CTU proporciona una interfaz de usuario fácil y orientada al usuario para configurar los módulos XBee de manera fácil y rápida.

Una vez conectada la tarjeta de desarrollo, en la pantalla principal del X-CTU se muestran las posibles conexiones USB-Serial.

Con el botón Test / Query el X-CTU comprueba que la conexión funciona correctamente (figura 3.7).

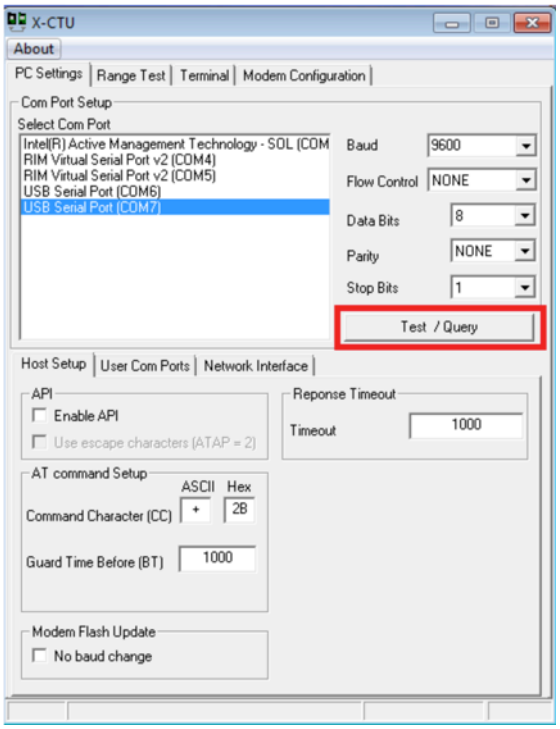


Figura 3.7. Interfaz de X-CTU para configuración de los módulos

En la pestaña "modem configuration" (figuras 3.8 y 3.9) se muestra la configuración actual del módulo XBee conectado. Es en esta pestaña donde podremos establecer el coordinador de la red y los dispositivos finales, así como fijar el direccionamiento de 16 bits de cada módulo XBee.

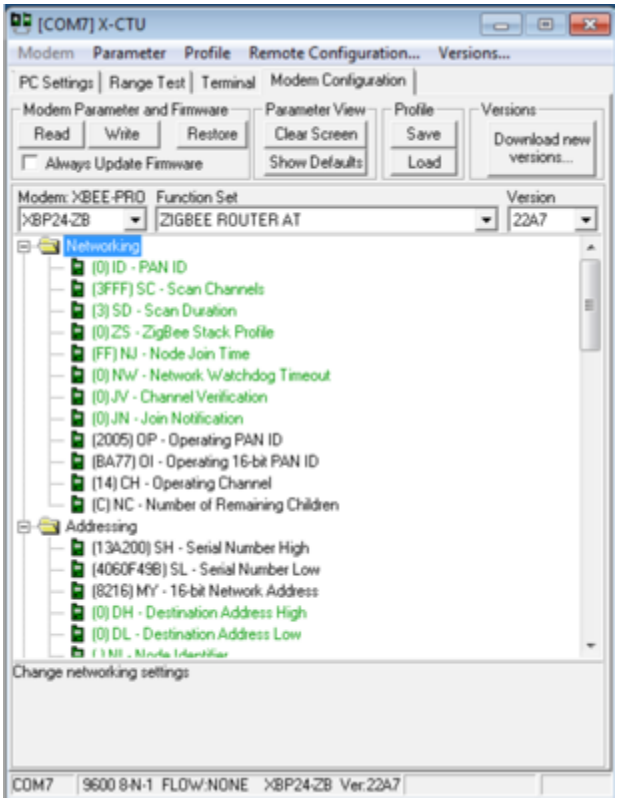


Figura 3.8. Vista de la pestaña "modem configuration"

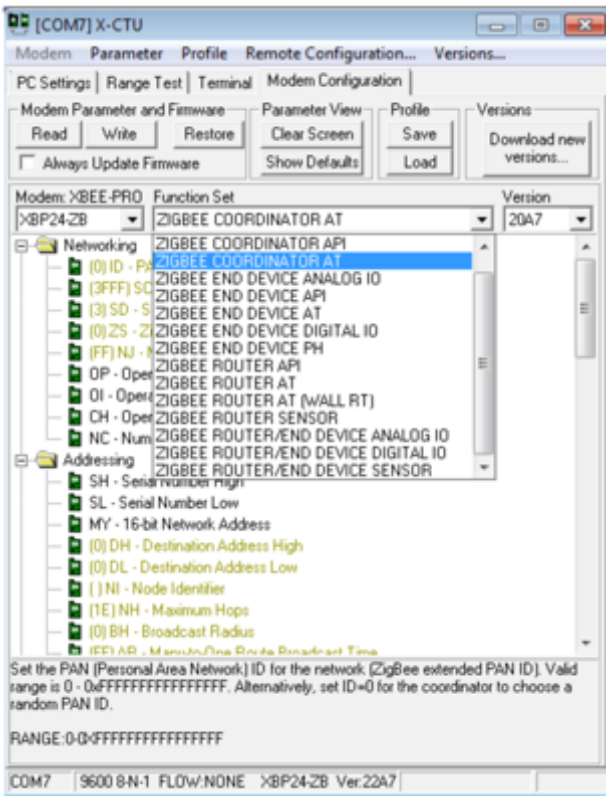


Figura 3.9. Configuración del módulo como coordinador

3.3 Unidad remota.

Después del estudio de la tecnología ZigBee se consideró que esta era muy adecuada para el proyecto, ya que es capaz de proporcionar a cada nodo las funcionalidades hardware requeridas después de decidir suprimir el sistema embebido del que en un principio se pensó que cada nodo dispondría. En concreto:

- El módulo XBee soluciona la problemática de la no disponibilidad de conexión a internet en cada nodo, ya que crea un enlace radioeléctrico para comunicarse con una unidad central que sí dispone de conexión a internet.
- Gracias a su pequeño procesador de 8 bits los módulos XBee tienen funcionalidades añadidas aparte de la creación de un enlace radioeléctrico, que le permitirán comunicarse con el hardware medidor. En concreto, el la función ATIC (monitorización de cambio de estado de una entrada digital) fue desde el principio de especial interés para el desarrollo del medidor.

Por otra parte, aunque la eliminación del sistema embebido haya traído consigo la simplificación y abaratamiento de cada nodo remoto, también implica ciertas limitaciones como la imposibilidad de usar algunos circuitos encapsulados diseñados para medir energía, ya que se necesita un sistema embebido que sea capaz de implementar el protocolo necesario para comunicarse con la mayoría de los circuitos. No obstante, gracias a la función ATIC (monitorización de los cambios de estado de una determinada entrada digital), otro segmento de medidores, cuya interfaz de comunicación se limita a la producción de pulsos, resultaron altamente integrables con el diseño planteado hasta ahora. Por ello se estudió el DDS5188.

Descripción general

El funcionamiento de las unidades remotas se podría resumir en los siguientes puntos:

- Toma de medidas de consumo.
- Recogida de datos de las mediciones y transmisión a la unidad central.

3.3.1 Toma de medidas de consumo.

Las medidas de consumo son llevadas a cabo por el medidor DDS5188. El DDS5188 fue elegido para este proyecto ya que se trata de un dispositivo totalmente autónomo, que se conecta directamente a la red eléctrica, ya que no requiere de la incorporación de ningún tipo de circuito de

acondicionamiento. Además, cuenta un una interfaz de salida de pulsos fácilmente integrable con el modo ATIC de los módulos XBee.

A continuación se muestra un diagrama con las conexiones (figura 3.10) requeridas para que este dispositivo pueda llevar a cabo mediciones de consumo energético:

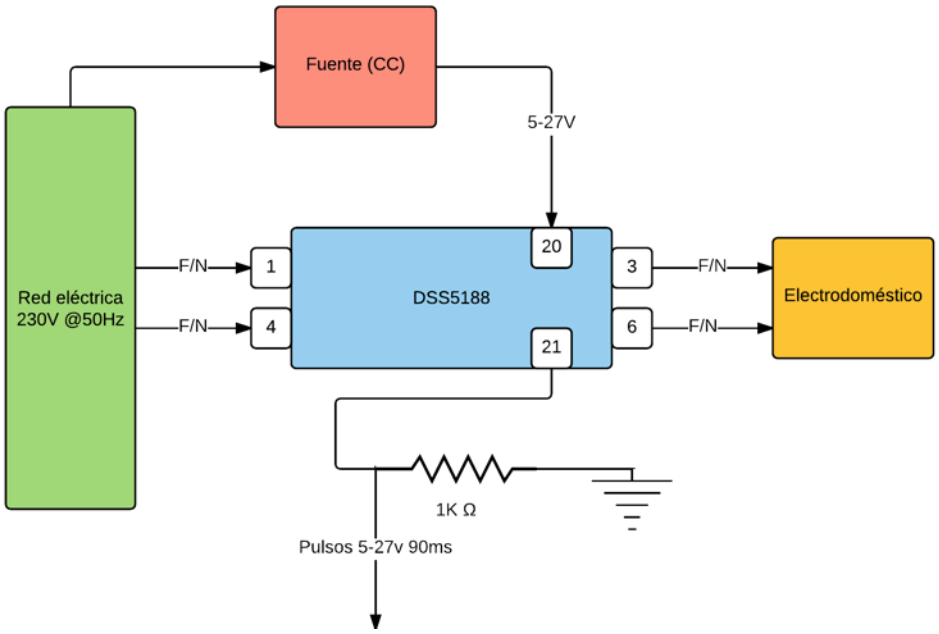


Figura 3.10. Diagrama de conexiones necesario para medir consumo de un electrodoméstico utilizando el DSS5188 (vista superior)

Se requiere de la apertura de los cables fase y neutro para intercalar el dispositivo. Una vez el dispositivo se haya intercalado, la energía eléctrica fluirá a través de él y este será capaz de realizar mediciones. Los cables fase y neutro que provienen directamente de la red eléctrica se deben conectar a los puertos 1 y 4. Los cables de salida del dispositivo, que irán a la carga que se desea alimentar saldrán de los puertos 3 y 6.

Para comunicar las medidas, el DSS5188 cuenta con un circuito que proporciona una interfaz de salida de pulsos. Este circuito es totalmente independiente del resto del dispositivo y debe alimentarse aparte. Para el correcto funcionamiento de la interfaz de salida de pulsos, se debe conectar una fuente de corriente continua de 5 a 27 V en los puertos 20 y 21. Los pulsos se obtendrán a partir del puerto 21, al que se deberá conectar una entrada digital que sea capaz de detectarlos, y una resistencia de pull down.
Los pulsos serán de la misma amplitud que la señal de voltaje con la que se alimenta el DSS5188.

El DSS5188 producirá un pulso de duración 90ms cada vez que 0,5Wh hayan sido consumidos.

3.3.2 Recogida de medidas y transmisión a la unidad central.

3.3.2.1 Toma de medidas

La detección de los pulsos que comunicarán la energía consumida, así como la transmisión de esta información a la unidad central, se llevará a cabo por un módulo XBee.

Para que el módulo XBee cumpla con este cometido, habrá que configurar una de sus pines de E/S digitales como entrada digital. En el caso de éste proyecto, el pin elegido ha sido el pin 0.

Una vez configurado el pin 0 como entrada digital, bastará con configurar el módulo XBee para que opere en modo ATIC sobre este pin en concreto.

El modo ATIC sirve para llevar a cabo monitorización de pines configurados como entrada digital. En concreto, este modo creará una trama de tipo API, que enviará al nodo coordinador de la red cuando se detecte un cambio de estado en el pin seleccionado.

De esta manera, la entrada digital se encontrará a cero lógico por norma, excepto cuando el DSS5188 emita un pulso. Cuando esto ocurra, el modo ATIC detectará dos cambios de estado. El primero se dará en el flanco de subida del pulso, y el segundo 90ms después, en el flanco de bajada. De esta manera, el XBee emitirá 2 tramas al coordinador de la red, una cuando se produzca el flanco de subida y otra cuando se produzca el flanco de bajada.

3.3.2.2 PCB interfaz DSS5188 módulo XBee

La función principal de este PCB es servir como interfaz hardware entre el DSS5188 y el módulo XBee. Para ello, incorpora el circuito y los componentes necesarios para llevar a cabo las siguientes funciones:

- Proporcionar 5V de alimentación al DSS5188.
- Proporcionar 3.3V de alimentación al módulo XBee.
- Transformar los pulsos de salida del DSS5188 de amplitud 5V a pulsos de 3,3V de amplitud que no dañen las entradas digitales del módulo XBee. A continuación se muestra el esquema de conexiones del PCB (figura 3.12) y el diagrama de conexiones implementado (figura 3.13).

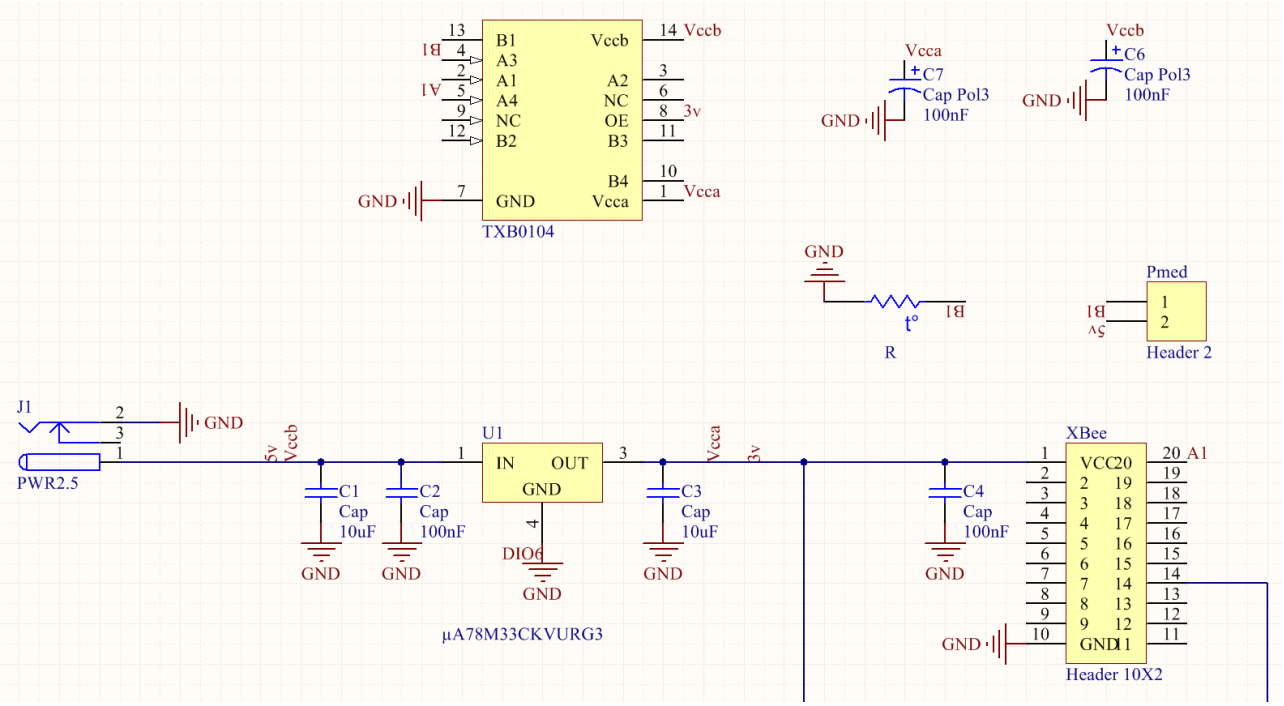


Figura 3.11. Esquema del circuito implementado en el PCB

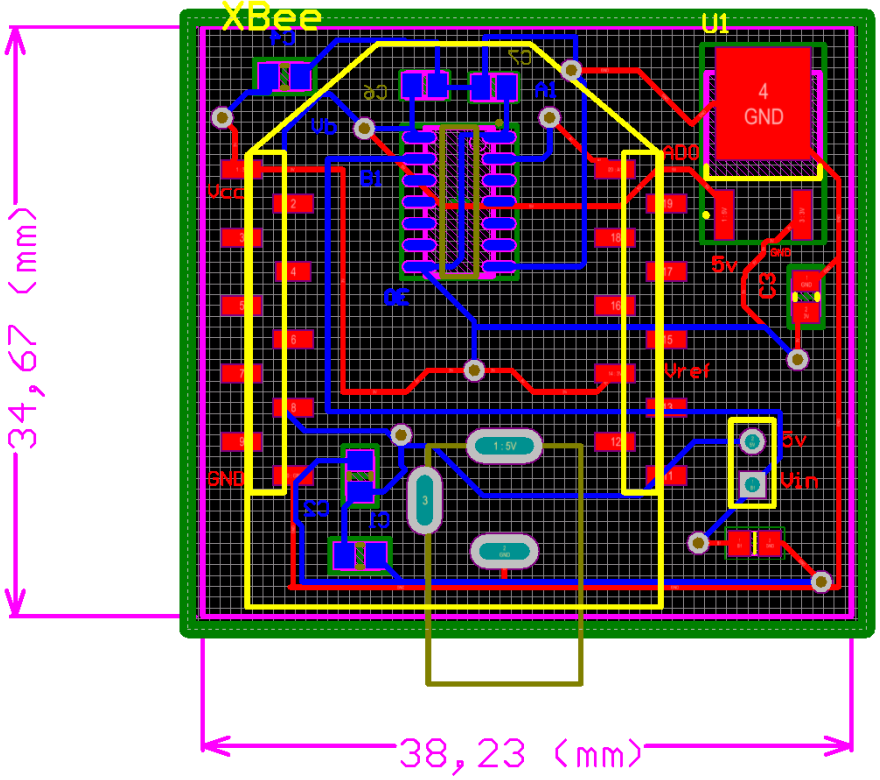


Figura 3.12. Esquema de conexiones del PCB

3.3.2.1.1 Zócalos hembra-macho 2mm para Xbee

Zócalos macho-hembra 2 mm: Estos zócalos están preparados para la inserción de un componente con separación de 2 mm entre sus patas. Gracias a ellos se evita tener que soldar directamente en el PCB el módulo Xbee, y ofrecen la posibilidad de sustituir el módulo por otro en caso de avería.

3.2.2.1.2 Regulador de tensión

Dado que la fuente que alimenta el PCB es de 5V, y el módulo XBee tiene que ser alimentado con 3.3V como máximo, se ha incorporado un regulador de tensión que transforme 5V a 3.3V para este propósito. En concreto, se ha utilizado el regulador de tensión uA78M33C[6] de Texas Instruments. Este regulador admite una tensión de entrada de 5V-25V, proporciona hasta 500mA de corriente y viene en el encapsulado TO-220.

3.2.2.1.2 Conversor de voltaje bi-direccional TXB0104 [7]

Este conversor de voltaje bi-direccional (figura 3.13) es el encargado de convertir los pulsos de duración 90ms y amplitud 5V a pulsos de amplitud 3.3V.

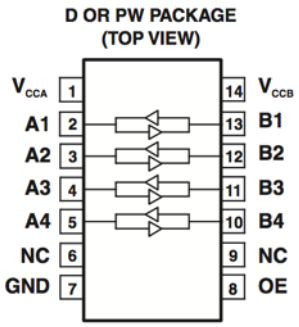


Figura 3.13. Encapsulado de conversor txb0104

Este conversor dispone de 4 canales de conversión, las señales conectadas a los canales A serán convertidas a sus respectivos canales B con su misma numeración y viceversa. El nivel de conversión que se desea tener en cada lado debe ser indicado en Vcca conectando este pin a dicho voltaje, y los voltajes de los canales A siempre deben ser menores que los de los canales B. El rango de funcionamiento de los canales es de 1.2V a 3.3V para los canales A, y de 1,65V a 5,5V para los canales B.

Aunque este dispositivo dispone de funcionalidades que no son necesarias para el tipo de conversión, como la bi-direccionalidad y la disposición de 4 canales, se ha empleado ya que se disponía en el laboratorio de varias unidades y ya se había probado satisfactoriamente con anterioridad. Además, dado que el Xbee dispone de muchas E/S digitales que no se están usando, este conversor es adecuado en vistas a ampliaciones futuras que requieran el uso de más E/S digitales.

3.2.2.1.3 Zócalos hembra-macho 2,54mm

Empleados para conectar las entradas 20 y 21 del DSS5188 a 5V y al canal B1 del conversor de voltaje respectivamente.

3.2.2.1.4 Condensadores de desacoplo

A fin de minimizar el ruido generado por la conmutación de las salidas del circuito integrado y otros efectos no deseados, se introducen condensadores de desacoplo. Los condensadores de desacoplo deben estar colocados entre la entrada de alimentación del circuito integrado y la tierra, lo más cerca posible del circuito integrado para minimizar el área del circuito recorrido por las corrientes parásitas.

3.2.2.1.4 Conector para fuente de 5V

Adapta la salida de la fuente de 5V que alimenta el PCB.

3.4 Unidad central.

La unidad central tiene como tarea recoger las medidas enviadas por cada nodo por ZigBee hacerlas visibles al usuario a través de internet. Desde un primer momento se pensó en alojar las medidas en una base de datos que sería accesible por una aplicación web que mostraría los datos de forma conveniente. Las opciones planteadas fueron:

- La unidad central actuaría como servidor alojando a la aplicación web y a la base de datos.
- La aplicación web y la base de datos se alojarían en un servidor externo, y la unidad central simplemente actuaría como intermediario entre los nodos independientes y el servidor remoto.

Se decidió que la solución implementaría la segunda opción. Ésta se consideró más adecuada por dos razones:

- Si se desea alojar un servidor en la unidad central, sería conveniente que esta dispusiera de suficientes recursos hardware para manejar los procesos software involucrados en esta tarea. Esto obligaría a descartar la mayoría de sistemas embebidos de pequeño tamaño que hasta ahora se habían pensado usar para este motivo.
- Si la unidad central también actúa como servidor, supondría crear una aplicación web totalmente dependiente de la unidad central, ya que cuando esta no se encontrara disponible, la aplicación web tampoco lo estaría. Por el contrario, si la unidad central únicamente actuara como intermediario entre la base de datos de la aplicación web y los nodos remotos, aunque ésta no

estuviera encendida, o no dispusiera de conexión a internet, la aplicación web seguiría estando disponible para consultar el histórico de datos de consumo.

Una vez decidido que la unidad central no actuaría como servidor web, el sistema embebido estudiado elegido para servir como unidad central fue una Raspberry Pi.

3.4.1 Interfaz hardware XBee-Raspberry Pi.

3.4.1.1 Conexiones implementadas.

Como ya se ha mencionado en apartados anteriores, el objetivo es desplegar una red ZigBee con topología en estrella operando en modo API. En este apartado se detallarán las conexiones físicas necesarias para que el módulo XBee sea capaz de comunicarse con la Raspberry Pi.

El módulo XBee recibe por radiofrecuencia tramas API y las transmite mediante UART a la Raspberry Pi. Para este cometido, las conexiones físicas requeridas (figura 3.15) son las siguientes:

- Pin 1: Alimentación del módulo con 3,3v.
- Pin 10: GND
- Pin 2: pin de salida de información UART (tx)
- Pin 3: pin entrada de información UART (rx)

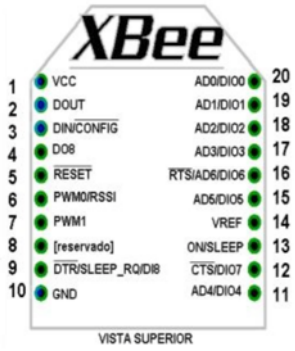


Figura 3.14. Diagrama de pines módulo XBee

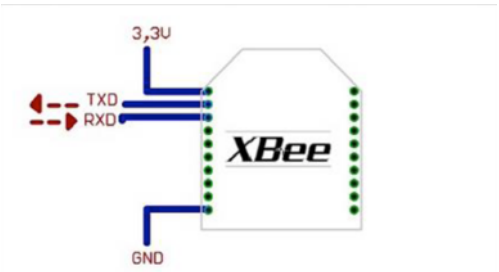


Figura 3.15.Conexiones necesarias para implementar una comunicación por UART

Así mismo, dado el diagrama de pines del conector 1 de la Raspberry Pi, se requiere el uso de los siguientes pines:

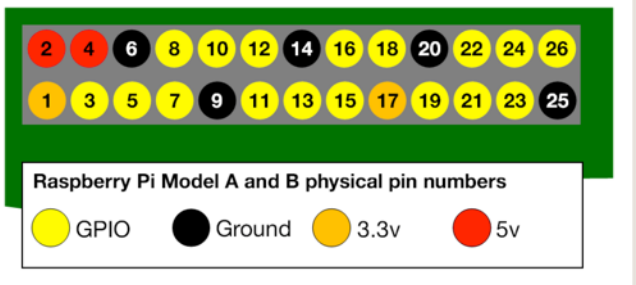


Figura 3.16. Diagrama de pines en conector 1 de Raspberry Pi

- Pin 1: proporcionará la alimentación de 3.3V a la Raspberry.
- Pin 6: GND
- Pin 8: Pin multipropósito, puede servir de GPIO convencional así como de interfaz física para el canal de salida de datos de una comunicación UART, por lo tanto, este pin servirá de UART TX
- Pin 10: al igual que el pin 8, se trata de un pin multipropósito con función de GPIO y de entrada de datos en comunicaciones UART. En este caso servirá de UART RX.

Las conexiones entre la Raspberry Pi y el módulo XBee se resumen en la tabla 3.1.

Pines conectados	
Pin en Raspberry	Pin en XBee
1	1
6	10
10	2
8	3

Tabla 3.1. Conexiones implementadas entre conector 1 de Raspberry Pi y módulo XBee

3.4.1.2 PCB en Raspberry Pi.

Todas las conexiones explicadas en el apartado anterior se implementan mediante un PCB (Printed Circuit Board) diseñado con el softare de diseño de circuitos impreso Altium Designer, que sirve de interfaz física entre el módulo Xbee y la Raspberry Pi.

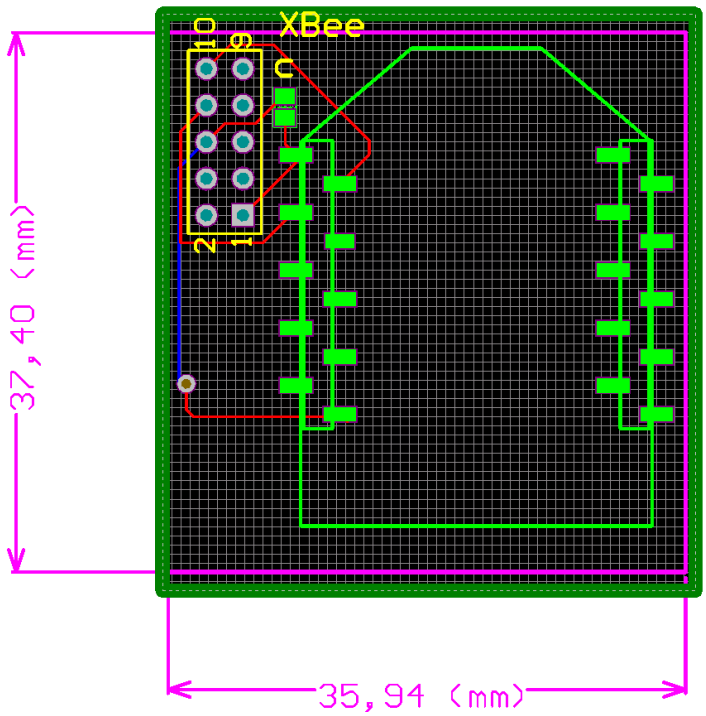


Figura 3.17. Diseño de PCB conectado en Raspberry Pi

El PCB está diseñado en dos capas, estando situadas las pistas rojas en la cara superior, y las pistas azules en la cara inferior. Los zócalos para el XBee que se encuentran marcados en verde también se encuentran situados en la cara superior (figura 3.18).

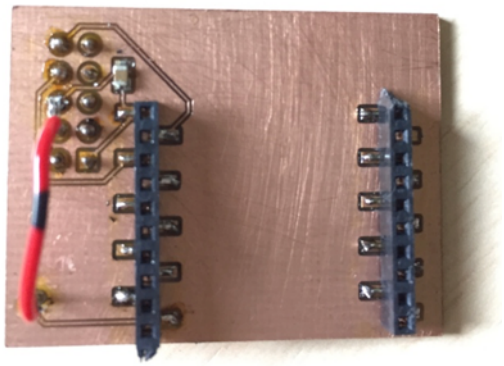


Figura 3.18. Cara superior del PCB en Raspberry Pi.

En la cara superior del PCB (figura 3.18) se montan los siguientes componentes:

- Condensadores de desacoplo: A fin de minimizar el ruido generado por la conmutación de las salidas del circuito integrado y otros efectos no deseados, se introducen condensadores de desacoplo. Los condensadores de desacoplo deben estar colocados entre la entrada de alimentación del circuito integrado y la tierra, lo más cerca posible del circuito integrado para minimizar el área del circuito recorrido por las corrientes parásitas.

- Zócalos macho-hembra 2 mm: Estos zócalos están preparados para la inserción de un componente con separación de 2 mm entre sus patas. Gracias a ellos se evita tener que soldar directamente en el PCB el módulo Xbee, y ofrecen la posibilidad de sustituir el módulo por otro en caso de avería.

En la cara posterior (figura 3.19) únicamente se encuentra un zócalo macho hembra para acoplar el PCB a los GPIO de la Raspberry Pi, que también se emplean para evitar soldar componentes directamente a la placa. El papel de estos zócalos es la de proporcionar una conexión física con los GPIO de la Raspberry Pi. A diferencia de los zócalos para los módulos XBee, estos zócalos están preparados para insertar pines con separación de 2,54mm.

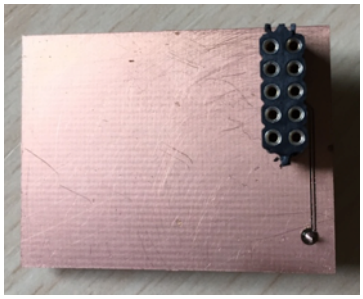


Figura 3.19. Cara posterior del PCB en Raspberry Pi

El conjunto PCB, con el módulo XBee incluido, acoplado a la Raspberry se muestra en las siguientes imágenes (figuras 3.20 y 3.21):

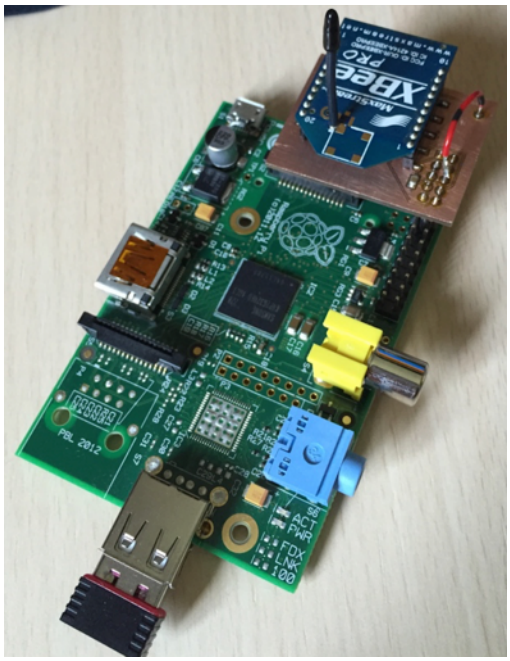


Figura 3.20 y 3.21. Vista superior de Raspberry Pi con el PCB conectado en conector 1.

3.4.2 Software en Raspberry Pi.

Hasta ahora, los módulos XBee instalados en los nodos remotos envían información por la red ZigBee empaquetada en tramas API. Una vez enviadas, estas son recibidas por el módulo XBee de la unidad central y transformadas en impulsos eléctricos y transmitidas a través de los GPIO de la Raspberry Pi mediante UART. A partir de aquí, estos impulsos eléctricos son procesados por un software específicamente desarrollado para este propósito. A continuación se describe en líneas generales el algoritmo de operación de este software:

0. Configuración de los módulos remotos.
1. Recepción de las tramas API mediante el puerto serie.
2. Procesado de las tramas.
3. Volcado de información en base de datos externa.

3.4.2.1 Recepción y envío de tramas mediante comunicación serie.

3.4.2.1.1 UART (Universal Asynchronous Receiver-Transmitter)

El módulo XBee recibe las tramas por radiofrecuencia y las transforma en señales eléctricas que se transmiten a la Raspberry Pi por los GPIO. Los bits que componen las tramas son transmitidos en serie y de manera asíncrona. No es necesario el empleo de una señal de reloj ya que la información de sincronismo va incluida en la propia trama, sin embargo, para que la comunicación serie sea posible, es necesario configurar al transmisor y al receptor con los mismos parámetros:

- Velocidad de transmisión (baudios)
- Bits de datos.
- Bits de stop.
- Control de flujo
- Paridad.

En el caso de la comunicación que existe entre la Raspberry Pi y los módulos XBee, se emplean 8 bits de datos, 1 bit de stop y 9600 baudios. La información se dividirá en secuencias de 10 bits, de los cuales 8 contendrán datos. En la figura 3.22 se muestra el diagrama temporal necesario para transmitir el byte 0xF1.

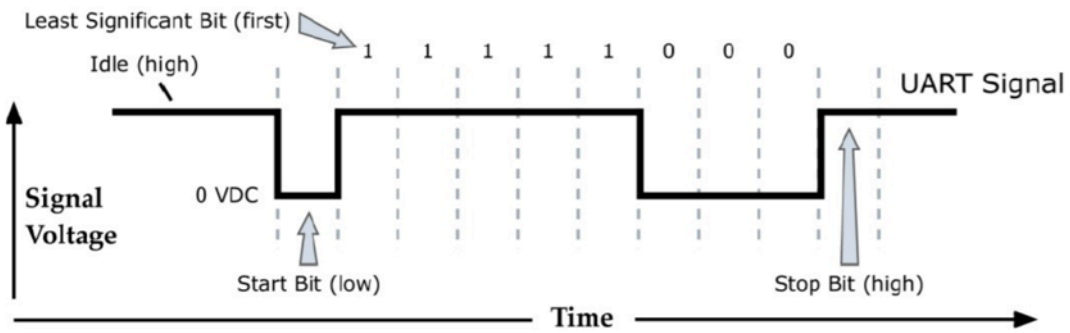


Figura 3.22. Diagrama temporal para transmisión de 11111000 mediante UART

Para ser capaz de interpretar este flujo de comunicación, la Raspberry Pi cuenta con un Transmisor-Receptor Asíncrono Universal (UART) que controla los puertos y dispositivos serie. La función principal de del chip UART es manejar las interrupciones de los dispositivos conectados al puerto serie, convertir los datos al formato paralelo para poder ser transmitidos por el bus del sistema y puedan ser utilizados.
Gracias a la comunicación asíncrona se pueden conseguir velocidades de hasta 921,6 Kbps full dúplex. Para ello, emplea canales separados para el envío y para la recepción de datos.

3.4.2.1.2 Software para envío y recepción de tramas API. Python.

El lenguaje de programación empleado para gestionar la recepción y envío de tramas API es Python. Python es un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y programación funcional. Es un lenguaje interpretado (ejecuta las instrucciones directamente, sin una previa compilación a lenguaje máquina), usa tipado dinámico (una misma variable puede tomar valores de distinto tipo en momentos diferentes) y es multiplataforma.
Las principal ventaja del lenguaje Python es la facilidad de su sintaxis, además, la existencia de librerías específicas para tratar con módulos XBee y con bases de datos MongoDB resulta especialmente útil para este proyecto.
La librería XBee de Python implementa la comunicación serie para tratar con tramas API en un Xbee.

3.4.2.1.3 Configuración remota de los módulos XBee.

Tanto para la configuración de los módulos XBee, así como para la recepción de la información de consumo se emplean tramas API.

Cuando se incorpora un nuevo nodo XBee a la red ZigBee, una vez que el modo API está activado en el módulo, éste es configurado remotamente mediante comandos AT.

El modo ATIC.
El modo ATIC configura un pin de E/S como entrada digital y monitoriza los cambios de estado que se producen en ese pin. Cuando un módulo XBee es opera en modo ATIC para alguno de sus pines, éste se mantiene a la espera observando cualquier cambio en el voltaje de entrada de dicho pin. Cuando se produzca un cambio de estado, de activo bajo a activo alto ó viceversa, el módulo enviará una trama para notificar de éste cambio

Formato de tramas API utilizadas.
Lo primero que se hará será enviar comandos AT remotos mediante tramas API para configurar el funcionamiento de los módulos. La estructura de estas tramas será de la siguiente manera (tabla 3.2)

Campo I		Offset	Ejemplo	Descripción
Delimitador inicial		0	0x7E	
Longitud		1-2	0x0010	Número de bytes entre el campo longitud y el checksum
Campo de datos	Tipo de trama	3	0x17	
	Identificador de trama	4	0x01	
	Id 64-bits destino	5-12	0x124FAD423435 HFF1	*0x0000000000000000 reservado para el coordinador. *0x000000000000FFFF broadcast
	Id 16-bits destino	13-14		Si no se sabe, introducir 0xFFFE
	Opciones de comando remoto	15	0x02	Opciones soportadas: * 0x01 deshabilitar ACK *0x02 habilitar cambios *0x40 utilizar tiempo de espera ampliado

	Comando AT	16-17	0x4248	Introducir las letras del comando AT en ASCII.
	Parámetros comando AT	18	0x01	
Checksum		19	0xF5	

Tabla 3.2. Formato genérico de estructura de tramas API

Una vez configurados lo módulos en modo ATIC, estos enviarán tramas con la información de las mediciones. En este caso, la unidad central recibirá tramas con el siguiente formato (tabla 3.3)

Campo I		Offset	Ejemplo	Descripción
Delimitador inicial		0	0x7E	
Longitud		1-2	0x0014	Número de bytes entre el campo longitud y el checksum
Campo de datos	Tipo de trama	3	0x92	
	Id 64-bits destino	4-11	0x124FAD423435HFF1	*0x0000000000000000 reservado para el coordinador. *0x000000000000FFFF broadcast
	Id 16-bits destino	12-13	0x7D84	Si no se sabe, introducir 0xFFFE
	Opciones de recepción	14	0x02	Opciones soportadas: * 0x01 *0x02 trama tipo broadcast
	Número de muestras.	15	0x01	Siempre debe ser 1.
	Máscara de canales digitales	16-17	0x001C	Indica qué E/S digitales están enviando los datos

	Máscara de pines analógicos	18	0x02	Indica qué E/S digitales están enviando datos
	Muestras digitales	19-20	0x0011	Uno por cada pin de E digital de la máscara de canales digitales. Las muestras correspondientes a aquellos pines que no se estén utilizando aparecerán a cero.
	Muestras analógicas	21-22	0x0225	Por cada canal analógico activado en la máscara se recibirán 2 bytes con el nivel de tensión que marque el pin. Los bytes aparecerán concatenados empezando desde AD0 a AD3.
Checksum		19	0xF5	

Tabla 3.3. Formato de trama enviada por un módulo XBee configurado en modo API

Como las tramas recibidas en la unidad central no contendrán información de tipo analógico, la máscara de los canales analógicos estará a cero y no habrá muestras analógicas en la trama recibida.

Tramas API para configuración remota.

Antes de que cada nodo remoto pueda operar de la manera deseada, es necesario configurarlo mediante una trama API desde la unidad central. Para que los nodos puedan funcionar en modo ATIC primero se ha de enviar una trama que configure el puerto que se desea monitorizar como entrada digital. Para ello se envía la siguiente trama: (tabla 3.4):

Valor	Significado
0x7E	Comienzo de trama
0x0010	Longitud (en bytes) del campo de datos. Este parámetro debe ser acorde con el introducido en X-CTU.
0x17	Id comando AT. Para indicar de qué tipo es la trama, que en este caso corresponde con un comando AT remoto.
0xXX	Id de trama. A elección del usuario.
0x0000000000000000	Dirección destino 64 bits. Se encuentra impreso físicamente debajo de cada módulo. Si se quisiera usar el direccionamiento de 16 bits (como en este caso) habría que ponerlo a cero.

0x0001	Dirección destino 16 bits. La dirección 0x0001 corresponde con el nodo remoto de prueba empleado en el proyecto.
0x02	Bits de opciones. En este caso, con 0x02 se evita tener que enviar una segunda trama para indicar al módulo que aplique los cambios.
0x4430	Opciones comando ATD0. 0x44 y 0x30 representan los equivalentes en ASCII de los caracteres D y 0. Con el comando ATD0 se indica al módulo XBee que las propiedades del siguiente campo deben ser aplicadas al DIO0.
0x03	Indica que el pin seleccionado en el comando AT debe ser configurado como entrada digital.
0xXX	Checksum de la trama API (calculado automáticamente por la librería de Python).

Tabla 3.4: Trama de configuración de E/S número 0 de módulo XBee como entrada digital

Una vez configurada una de las E/S digitales como entrada digital, concretamente la E/S 0, ya se puede configurar el XBee para que monitorice los cambios de estado de dicha entrada digital. Para ello se envía la siguiente trama (tabla 3.5), que contiene el comando ATIC.

Valor	Significado
0x7E	Comienzo de trama
0x0010	Longitud (en bytes) del campo de datos. Este parámetro debe ser acorde con el introducido en X-CTU.
0x17	Id comando AT. Para indicar de qué tipo es la trama, que en este caso corresponde con un comando AT remoto.
0xXX	Id de trama. A elección del usuario.
0x0000000000000000	Dirección destino 64 bits. Se encuentra impreso físicamente debajo de cada módulo. Si se quisiera usar el direccionamiento de 16 bits (como en este caso) habría que ponerlo a cero.
0x0001	Dirección destino 16 bits. La dirección 0x0001 corresponde con el nodo remoto de prueba empleado en el proyecto.
0x02	Bits de opciones de comando remoto. En este caso, con 0x02 se evita tener que enviar una segunda trama para indicar al módulo que aplique los cambios.
0x4943	Comando AT. En este caso, 0x49 significa “I” en código ASCII y 0x43 “C” en código ASCII también. De esta manera se introduce el comando “IC”.
0x0001	Opciones de comando ATIC. En este caso se debe introducir una máscara de 16 bits indicando qué pines se deben monitorizar. Esta máscara asigna un bit a cada pin de manera que el menos significativo corresponde con el DIO0 y el bit 11 con el DIO11. El bit del DIO que se desee monitorizar debe estar a 1. En este caso esta máscara será 00000000000000001 = 0x0001.
0xXX	Checksum de la trama API (calculado automáticamente por la librería de Python).

Tabla 3.5. Trama de configuración de entrada digital número 0 en modo ATIC

Por último, una vez que el pin 0 ha sido configurado como entrada digital, y se ha configurado el módulo XBee para que monitorice los cambios de estado de dicho pin, se guardará esta configuración en la memoria no volátil que incorpora el módulo XBee. De esta manera, aunque se interrumpa la alimentación del módulo, no se perderá la configuración. Para ello se enviará otra trama API (tabla 3.6) con el comando ATWR remoto, que guardará la configuración en la memoria no volátil.

Valor	Significado
0x7E	Comienzo de trama
0x0010	Longitud (en bytes) del campo de datos. Este parámetro debe ser acorde con el introducido en X-CTU.
0x17	Id comando AT. Para indicar de qué tipo es la trama, que en este caso corresponde con un comando AT remoto.
0xXX	Id de trama. A elección del usuario.
0x0000000000000000	Dirección destino 64 bits. Se encuentra impreso físicamente debajo de cada módulo. Si se quisiera usar el direccionamiento de 16 bits (como en este caso) habría que ponerlo a cero.
0x0001	Dirección destino 16 bits. La dirección 0x0001 corresponde con el nodo remoto de prueba empleado en el proyecto.
0x02	Bits de opciones de comando remoto. En este caso, con 0x02 se evita tener que enviar una segunda trama para indicar al módulo que aplique los cambios.
0x5752	Comando AT. En este caso, 0x57 significa “W” en código ASCII y 0x53 “R” en código ASCII también. De esta manera se introduce el comando “WR”.
0xXX	Checksum de la trama API (calculado automáticamente por la librería de Python).

Tabla 3.6. Trama para almacenado de configuración en memoria no volátil

El código necesario para enviar estas tres tramas de configuración inicial en Python es el siguiente (figura 3.23):

```
"""
Configuracion de el puerto serie que se va a emplear, y Configuracion
de la misma velocidad de transmision ya configurada mediante x-ctu en
los modulos Xbee.
"""
ser = serial.Serial('/dev/ttyAMA0', 9600)
xbee = XBee(ser)


xbee.send('remote_at',
          frame_id='A',
          dest_addr_long='\x00\x00\x00\x00\x00\x00\x00\x00',#direccionamiento 64 bits no utilizado
          dest_addr='\x00\x01',#Direccion corta de xbee receptor.
          options='\x02',#Aplicar el comando nada mas recibirlo.
          command='D0', #Seleccionar pin DIO 0.
          parameter='\x03') #Configurar como entrada digital.

xbee.send('remote_at',
          frame_id='B',
          dest_addr_long='\x00\x00\x00\x00\x00\x00\x00\x00',#direccionamiento 64 bits no utilizado
          dest_addr='\x00\x01',#Direccion corta de xbee receptor.
          options='\x02',#Aplicar el comando nada mas recibirlo.
          command='IC',#Avisar de los flancos de subida o de bajada.
          parameter='\x00\x01')#En el pin DIO0

xbee.send('remote_at',
          frame_id='B',
          dest_addr_long='\x00\x00\x00\x00\x00\x00\x00\x00',#direccionamiento 64 bits no utilizado
          dest_addr='\x00\x01',#Direccion corta de xbee receptor.
          options='\x02',#Aplicar el comando nada mas recibirlo.
          command='WR')#Guardar las instrucciones en memoria no volatil.
```

Figura 3.23. Código Python para enviar 3 tramas de configuración: configuración de E/S número 0 como entrada digital, configuración de E/S número 0 en modo ATIC, almacenamiento de la nueva configuración en memoria no volátil [8][9]

Como se puede ver, el Python facilita mucho la tarea de envío de tramas gracias a sus librerías específicas para la comunicación con módulos XBee. Gracias al Python, sólo es necesario incluir el destino y las opciones de el comando remoto en hexadecimal. Todos los códigos hexadecimales y ASCII indicados en las tablas de arriba, así como el cálculo de el checksum, pueden ser introducidos con el nombre del propio comando.

3.4.2.1.4 Recepción de tramas API.

Una vez configurado un módulo para que opere en modo ATIC, monitorizando los pines indicados en la máscara digital, cada vez que se detecta un cambio de estado en alguno de estos, el módulo remoto enviará una trama a la unidad central que será de la siguiente manera (tabla 3.7):

Campo I		Valor	Significado
Delimitador inicial		0x7E	
Longitud		0x0014	Número de bytes entre el campo longitud y el checksum
Campo de datos	Tipo de trama	0x92	
	Id 64-bits destino	0x124FAD423435H	*0x0000000000000000 reservado para el

		FF1	coordinador. *0x000000000000FFFF broadcast
	Id 16-bits destino	0x7D84	Si no se sabe, introducir 0xFFFE
	Opciones de recepción	0x01	Opciones soportadas: * 0x01 *0x02 trama tipo broadcast
	Número de muestras.	0x01	Siempre debe ser 1.
	Máscara de canales digitales	0x0001	Indica qué E/S digitales están enviando los datos
	Máscara de pines analógicos	0x00	Indica qué E/S digitales están enviando datos
	Muestras digitales	0x000X	Uno por cada pin de E/S digital de la máscara de canales digitales. Las muestras correspondientes a aquellos pines que no se estén utilizando aparecerán a cero.
Checksum		0xF5	

Tabla 3.7. Trama enviada por módulo XBee en nodo remoto a módulo Xbee en unidad central cuando se detecta un cambio de estado en la entrada digital que está siendo monitorizada

3.4.2.2 Algoritmo global de recepción y procesado.

En el siguiente diagrama (figura 3.24) se describe el algoritmo de operación del software que se ejecuta en la unidad central y que gobierna todo el proceso de recepción, procesado y volcado de datos en la base de datos.

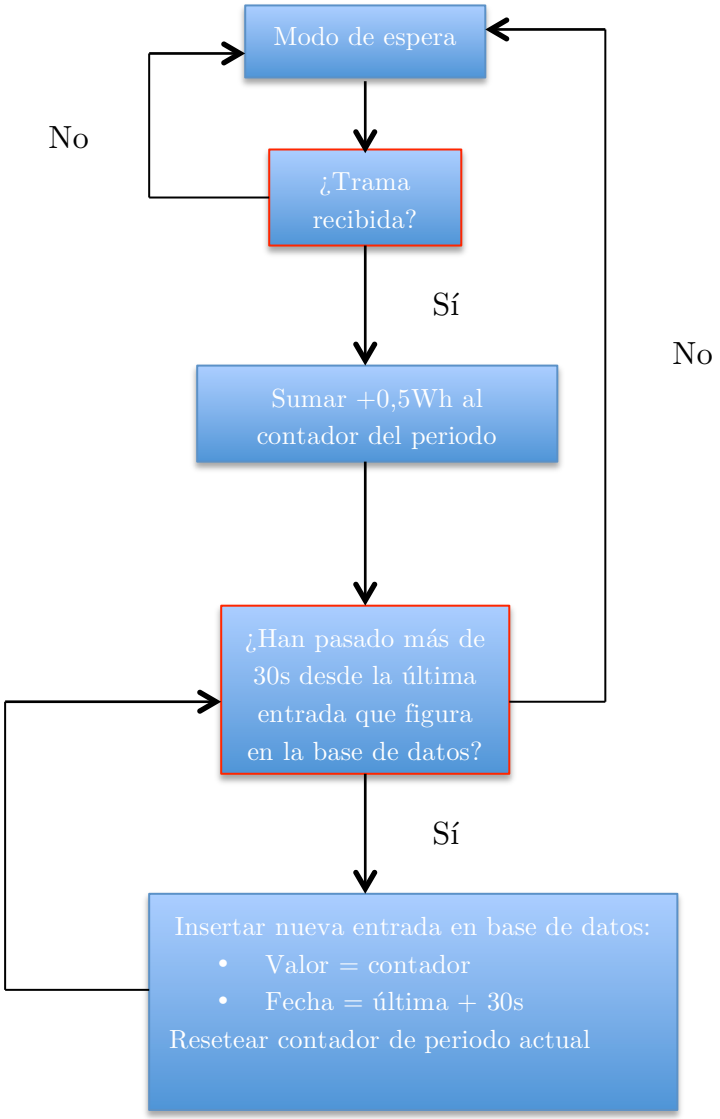


Figura 3.24. Algoritmo de operación de código Python ejecutado en la unidad central

3.5 Aplicación web.

3.5.1 Tecnologías empleadas

La plataforma encargada de mostrar al usuario las mediciones de consumo energético se ha implementado con las siguientes tecnologías.

- MeteorJS: software encargado de manejar la base de datos. Se trata de un framework de Javascript que permite crear webs dinámica de alto rendimiento.
- La base de datos donde se almacenan los datos de consumo se implementan con la tecnología MongoDB. Entre los principales motivos de esta elección se encuentran la integración con MeteorJS y la existencia de librerías específicas para Python. Gracias a MeteorJS, los elementos de esta base de datos pueden insertarse directamente en el HTML 5.
- La parte visual de la web se decidió implementar usando una combinación de HTML 5 y otros paquetes de desarrollo web como Chart.js y Bootstrap 3.0.

3.5.2 Estructura de la aplicación web.

El código y los recursos que usa la aplicación están estructuradas de la siguiente manera:

```
~ client/       graficas_helpers.js
                graficas_rendered.js
                graficas.css
                graficas.html
                router.js

~ deploy/       mup.json
                mup.json.save
                settings.json

~ lib/  db.js
```

3.5.2.1 client/

En esta carpeta se encuentra todo el código que se ejecutará en el cliente. Cuando se accede mediante un navegador a la aplicación web, el servidor envía los scripts que contenga esta carpeta. Los scripts contenidos en esta carpeta se ejecutarán únicamente en el cliente.

3.5.2.1.1 graficas_rendered.js

Este script contiene las siguientes funciones:

- leerPotencias:
  - Recibe: Duración del intervalo en segundos.
  - Devuelve un array con las últimas mediciones registradas correspondientes a un intervalo de tiempo de tamaño indicado en el argumento recibido. Calculará la potencia consumida desde un comienzo con un offset hasta que se alcance la duración del intervalo indicada por argumento. Esta función se utiliza para obtener los datos de la Collection para la primera gráfica. La única diferencia con leerPotencias2 es que el offset que indica el comienzo del intervalo en ambas es diferente.
- generarData:
  - Recibe: array devuelto por leerPotencias.
  - Procesa las mediciones recibidas por argumento, y devuelve una estructura de datos que contiene los datos necesarios para la gráfica. Entre se incluyen los datos del eje y, los del eje x y algunas opciones de configuración para la gráfica. Esta función es utilizada para crear los datos de la primera gráfica. Análogamente, para generar los datos de la segunda gráfica existe otra función llamada generarDataSegundaGrafica.
- pintarGrafica:
  - Recibe: la estructura devuelta por generarData y un id.
  - Devuelve: un objeto de tipo chart que mostrará los datos contenidos en la estructura recibida por argumento y la dará nombre con el identificador también recibido por argumento. Esta gráfica será accesible desde el html mediante este id.
- hora_en_string:
  - Recibe: hora desde epoch (1970) en milisegundos.
  - Devuelve: hora en formato hora:minutos:segundos.
- Fecha_en_string:
  - Recibe: hora desde epoch (1970) en milisegundos.
  - Devuelve: fecha en formato Día mes
- calcularTotalPower:
  - Recibe un array con todos los elementos de la Collection MongoDB.

- Devuelve la potencia total consumida en un intervalo de tiempo. Este intervalo de tiempo estará determinado por los datos que contengan el array que recibe como argumento.

- Template.graficas.rendered: función principal que hace las llamadas necesarias a las funciones mencionadas hasta ahora. Ya que ésta función contiene los datos reactivos que modificarán el html continuamente, ésta se ejecuta por primera vez cuando el html se ha cargado, y a partir de entonces, se ejecutará automáticamente modificando el html de la aplicación cuando cualquiera de los datos reactivos de los que depende sean modificados. A continuación se incluye el código de esta función (figura 3.25).

```
Template.graficas.rendered = function()
{  //Esta funcion esperara el tiempo indicado por setTimeout para arrancar.
  //Esto se hace para dar tiempo de sobra a que cargue la base de datos
  Meteor.setTimeout(function()
  {
    //Gracias a Tracker.autorun se crea una computacion, que se ejecutara
    //siempre que cualquiera que los datos reactivos de los que depende cambien
    Tracker.autorun(function ()
    {
      //Devuelve un array con las ultimas 20 mediciones almacenadas en la base de datos.
      //Estas mediciones corresponden a los ultimos 20 invervalos de 30s.
      var potenciasLeidasPrimeraGrafica = leerPotencias(20);

      //Devuelve una estructura con los datos necesarios para la primera grafica.
      var dataPrimeraGrafica = generarData(potenciasLeidasPrimeraGrafica);

      var comienzo_10min = potenciasLeidasPrimeraGrafica[19].epoch*1000;

      Session.set("comienzo_10min", hora_en_string(comienzo_10min));

      var final_10min = potenciasLeidasPrimeraGrafica[0].epoch*1000+30000;

      Session.set("final_10min", hora_en_string(final_10min));

      //Una vez se tienen los datos en dataPrimeraGrafica, se le pasa a la funcion
      //que creara un elemento de tipo chart accesible desde el html
      //bajo el identificador "myChart1"
      pintarGrafica("myChart1", dataPrimeraGrafica);

      //repetir el procedimiento pero para la segunda grafica.
      var potenciasLeidasSegundaGrafica = leerPotencias2(2880);
      var dataSegundaGrafica = generarDataSegundaGrafica(potenciasLeidasSegundaGrafica, 120);
      var comienzo_24h = potenciasLeidasSegundaGrafica[30].epoch*1000;

      Session.set("comienzo_24h", fecha_en_string(comienzo_24h));
      pintarGrafica("myChart2", dataSegundaGrafica);
      Session.set("potencia_primera_grafica", calcularTotalPower(potenciasLeidasPrimeraGrafica));
      Session.set("potencia_segunda_grafica", calcularTotalPower(potenciasLeidasSegundaGrafica));
    });
    //tiempo de de Meteor.setTimeout en milisegundos
  }, 6000);

}
```

Figura 3.25. Función principal de la aplicación web

3.5.2.1.2 graficas_helpers.js

Este script contiene el código necesario para poder incrustar en el html variables almacenadas en la memoria del navegador o en la base de datos. Para ello se crean varios helpers, que modificarán el html cuando los datos reactivos de los que dependen cambien. Los helpers, a diferencia de la función onRendered, ya son computaciones de por sí, por lo que no será necesario añadir Tracker.autorun. En la figura 3.26 se muestra el código de graficas_helpers.js.

```
1  Template.graficas.helpers
2  (
3  {
4    // Permitiran incrustar datos reactivos directamente en el html.
5    // Estos datos son calculados y modificados en onRendered().
6    comienzo_10min: function ()
7    {
8      return Session.get("comienzo_10min");
9    },
10   final_10min: function ()
11   {
12     return Session.get("final_10min");
13   },
14   comienzo_24h: function ()
15   {
16     return Session.get("comienzo_24h");
17   },
18   potencia_primera_grafica: function()
19   {
20     return Session.get("potencia_primera_grafica");
21   },
22   potencia_segunda_grafica: function()
23   {
24     return Session.get("potencia_segunda_grafica");
25   },
26   energia_intervalo: function ()
27   {
28     return Session.get("energia_intervalo");
29   }
30 // });
```

Figura 3.26. Código de los helpers. Los helpers permiten la insterción de elementos dinámicos en el HTML

Estos helpers son accesibles directamente desde el html de la siguiente manera (figuras 3.27, 3.28 y 3.29).

```
<div class="jumbotron">
    <div class="chartContainer">
        <h3>Consumo {{comienzo_10min}} - {{final_10min}}:
        {{potencia_primera_grafica}} (Wh)</h3>
        <canvas id="myChart1" style="width:100%"></canvas>
        <button id="buttom_1" type="button" class="btn btn-primary btn-sm
        opacity">
            <span class="glyphicon glyphicon-circle-arrow-left" aria-hidden="
            true"></span>
        </button>
        <button id="buttom_2" type="button" class="btn btn-primary btn-sm
        opacity">
            <span class="glyphicon glyphicon-circle-arrow-right" aria-hidden="
            true"></span>
        </button>
    </div>

</div>
```

Figura 3.27. Uso de los helpers desde el HTML (código resaltado). Para usar un helper en el HTML se usan los spacebars ({{}})

```
<div class="jumbotron">
    <div class="chartContainer">
        <h3>Consumo {{comienzo_24h}}: {{potencia_segunda_grafica}}(Wh)</h3>
        <canvas id="myChart2" style="width:100%"></canvas>
        <button id="buttom_3" type="button" class="btn btn-primary btn-sm
        opacity">
            <span class="glyphicon glyphicon-circle-arrow-left" aria-hidden="
            true"></span>
        </button>
        <button id="buttom_4" type="button" class="btn btn-primary btn-sm
        opacity">
            <span class="glyphicon glyphicon-circle-arrow-right" aria-hidden="
            true"></span>
        </button>
    </div>
</div>
```

Figura 3.28. Uso de los helpers desde el HTML (código resaltado). Para usar un helper en el HTML se usan los spacebars ({{}})

```
<div class="row">
    <div class="col-xs-12">
        <h2>Consumo: {{energia_intervalo}} (Wh)</h2>
    </div>
</div>
```

Figura 3.29. Uso de los helpers desde el HTML (código resaltado). Para usar un helper en el HTML se usan los spacebars ({{}})

3.5.2.1.3 graficas_events.js

El código incluido en este fichero ayuda a manejar los eventos que se deben disparar cuando el usuario interactúa con la aplicación web. Incluye toda la parte lógica que se debe de ejecutar en javascript para modificar automáticamente el código html que debe mostrar al usuario. Se incluyen las siguientes funciones:

- date_en_epoch:
    - Recibe: fechas introducidas por el usuario en las casillas cuando se desea calcular el consumo en un intervalo dado.
    - Devuelve: esa fecha en milisegundos desde 1970
- leerTodasPotencias:
    - Devuelve: un array con los datos de todas las mediciones de potencia almacenadas en la Collection. Este array será usada para calcular el consumo en un intervalo de tiempo determinado.
- Power_intervalo:
    - Recibe: el comienzo y el fin del intervalo en tiempo en milisegundos desde 1970.
    - Devuelve: un número con el consumo de energía en el intervalo marcado por el comienzo y fin recibidos por argumento
- Template.graficas.events:
    - Contiene las funciones que se deben ejecutar cuando el usuario interactúa con cualquiera de los elementos de la aplicación web.

3.5.2.1.4 Router.js

El código explicado hasta ahora se corresponde con el proceso de ejecución para mostrar las mediciones correspondientes a un único nodo remoto. Sin embargo, se pretende que en la aplicación web se pueda ver el consumo de tantos nodos como haya instalados en la red ZigBee.

Es aquí donde cobra más sentido el uso de MeteorJS. Para mostrar la información relativa a otros nodos medidores, lo único que habría que hacer es distinguir las mediciones de cada nodo en la base de datos, y nutrir a la aplicación hasta ahora explicada con los datos específicos de cada nodo.

Para facilitar esta tarea, evitando replicar ni una sola línea de código, se ha sustituido el sistema de rutado estático tradicional, en el que un archivo en la raíz de la aplicación web se corresponde con una referencia url única.

En lugar de ello, sólo existirá el sistema de plantillas explicado hasta ahora, que se nutrirá de los distintos datos correspondientes a cada nodo remoto, y será asociado con link dinámico, creado en función del número de nodos que se quieran añadir. De esta manera, este único sistema de plantillas será reutilizado varias veces con diferentes datos, y será asociado a tantas url dinámicas como sea necesario. Para ello, se ha incorporado el paquete Iron Router para Meteor, que nos permitirá realizar un rutado dinámico.

En la figura 3.30 se muestra un ejemplo de rutado para la plantilla principal de las gráficas, en el que se asocia una template con una url. En este caso, dado que sólo hay un nodo instalado, habrá una correspondencia estática plantilla-url.

router.js

```
Router.map(function ()
  {
    this.route('graficas',
    {
      path: '/',
      template: 'graficas'
    });

  });
```

Figura 3.30. Código de configuración del Router de MeteorJS (para un solo nodo)

A continuación se muestra el código (figura 3.31) que se utilizaría realizar una asociación plantilla-multiples url. Así mismo, Iron Router permite recibir información por medio de la url, de esta manera, iron router puede extraer esos datos, y enviárselos a otra función para saber por ejemplo, con qué id tiene que buscar en la base de datos para extraer los datos (correspondiendes a un nodo en concreto) de los que se nutrirá la plantilla.

router_dinamico.js

```
Router.map(function ()
  {
    this.route ('graficas',
     {
        //El router sabra, que cuando se le pide dirigir la aplicacion a una
        //url concreta, la parte de detras de la "/" corresponde con el id del
        //nodo en concreto cuyos datos se van a mostrar.
        path: '/:id_del_nodo',
        //Aunque hay tantas url como nodos, no se replica codigo, ya que la app
        //utiliza siempre la misma plantilla.
        template: 'graficas',
        data: function()
        {
            //Para realizar la extraccion de datos de la base de datos correctamente,
            // y nutrir el sistema de plantillas ya explicado con los datos correctos
            //correspondientes a un nodo en concreto, iron router devuelve despues
            //de su llamada el id del nodo que ha extraido de la url.
            return
            {
                id_nodo : this.params.id_del_nodo;
            };
        }
    });

  });
```

Figura 3.31. Código para configuración del Router de MeteorJS para dar soporte a varios nodos medidores

3.5.2.1.5 graficas.html

Contiene el código html que proporciona la parte visual de la web. Desde él se accede a toda la lógica de procesado llevada a cabo por JavaScript para calcular los datos de consumo.

3.5.2.1.6 graficas.css

Proporciona características de diseño específicas a la estructura visual del HTML. En este proyecto se ha hecho uso del framework Bootstrap para proporcionar las bases css y js del diseño de la web. Además, se ha añadido unas características css mediante la hoja de estilos graficas.css

3.5.2.2 /deploy.

Este directorio contiene los archivos con los datos necesarios para desplegar la aplicación en un servidor externo fácil y rápidamente.
Estos archivos contienen los datos necesarios para que la herramienta Meteor Up, despliegue de forma correcta y rápida la aplicación Meteor en un servidor externo. Para que Meteor Up pueda desplegar éste proyecto únicamente ha sido necesario modificar el archivo "mup.json". En él se incluye información acerca del servidor donde se alojará la aplicación:

- IP del servidor para poder acceder a él a través de ssh.
- Usuario bajo el que Meteor Up accederá al servidor para instalar la aplicación web.
- Contraseña del usuario.
- Puerto de acceso a la aplicación web.
- Directorio raíz para la aplicación web.
- Directorio raíz para la base de datos que nutrirá a la aplicación web y puerto a través del cuál se accederá.

3.5.2.3 lib

Contiene los elementos comunes entre el servidor y el cliente. Además, los archivos contenidos en esta carpeta serán los primeros en ser cargados. De esta manera, cuando la web arranque, la base de datos ya estará disponible.
Ésta carpeta incluye únicamente el archivo db.js para la creación de la colección Mongo que nutrirá a la aplicación web.
En el archivo db.js, se crea la variable que se asociará con una colección de una base de datos MongoDB. Aunque este código sea común para cliente y servidor, en el servidor esta variable se asociará a toda la base de datos, y en el cliente únicamente a una parte restringida de esta (aunque en este caso, como sólo hay un nodo y un usuario, el cliente tiene acceso a toda la base de datos).

4. Pruebas y resultados

Gracias al diseño modular que se ha desarrollado para este sistema, primero se probaron por separado cada una de sus partes (unidad remota, unidad central y aplicación web), y después se integraron todas y se hizo una prueba general del sistema. A continuación se describen las pruebas realizadas a cada uno de los bloques así como al sistema completo.

4.1 Unidad remota

Las pruebas de la unidad remota se separaron en dos bloques, primero se probó que el medidor de consumo, y después la interfaz hardware que adapta la salida del pulsos del medidor al módulo Xbee.

4.1.2 Pruebas al medidor de consumo

Con esta prueba se pretendía comprobar que la conexión red eléctrica, medidor de consumo, aparato eléctrico estaban bien implementadas y el medidor de consumo funcionaba completamente. En la figura 4.1 se muestra un esquema resumen de las conexiones implementadas para esta prueba.

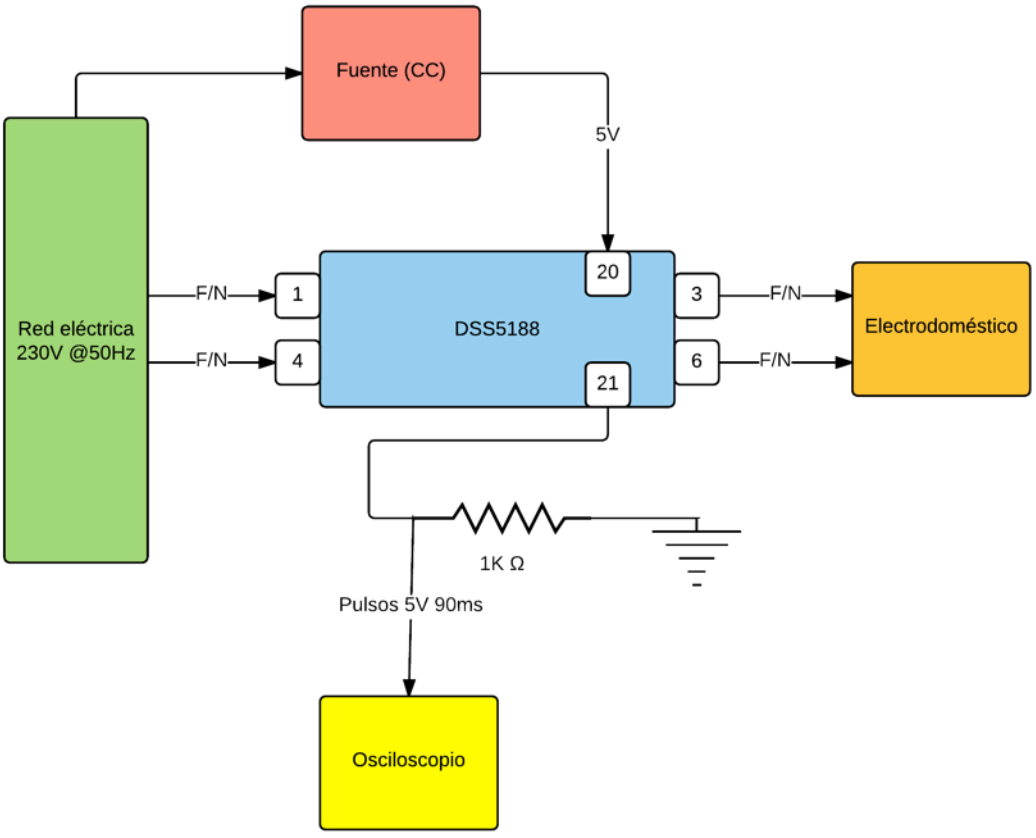


Figura 4.1. Conexiones implementadas para la prueba

Se pretendía comprobar que los pulsos de la salida correspondían con el indicador led rojo que parpadea cada vez que se consuman 0,5Wh, y que estos tenían 5V de amplitud y su duración era 90ms. Los resultados obtenidos con el osciloscopio fueron satisfactorios (figura 4.2).

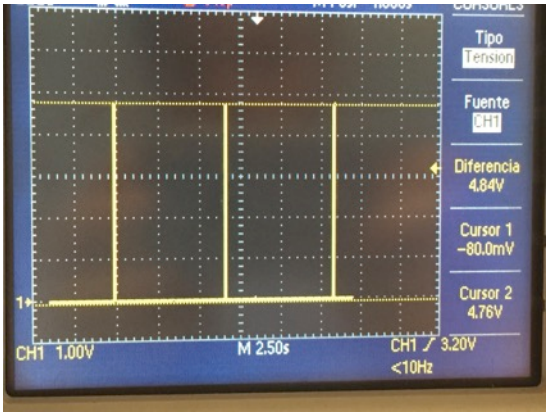


Figura 4.2. Gráfica de salida de pulsos del medidor de consumo mostrada en la pantalla del osciloscopio

En la anterior imagen (figura 4.2) se muestran 3 pulsos del sistema, que representan un consumo de 1,5Wh. En la figura 4.3, se muestra un pulso ampliado.

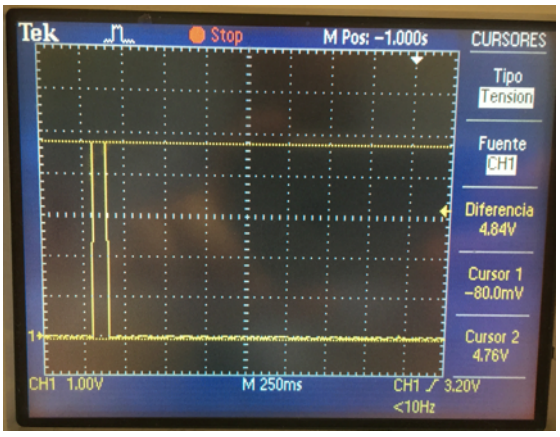


Figura 4.3. Detalle temporal de uno de los pulsos de la gráfica mostrada en la imagen anterior

4.1.3 Pruebas sistema medidor de consumo más PCB

Con esta prueba se pretendía comprobar que el sistema que compone el medidor más la interfaz hardware PCB (figura 4.4) para XBee funcionaba correctamente.

A la salida del siguiente sistema, se pretendían obtener pulsos de amplitud 3,3V y duración 90ms.

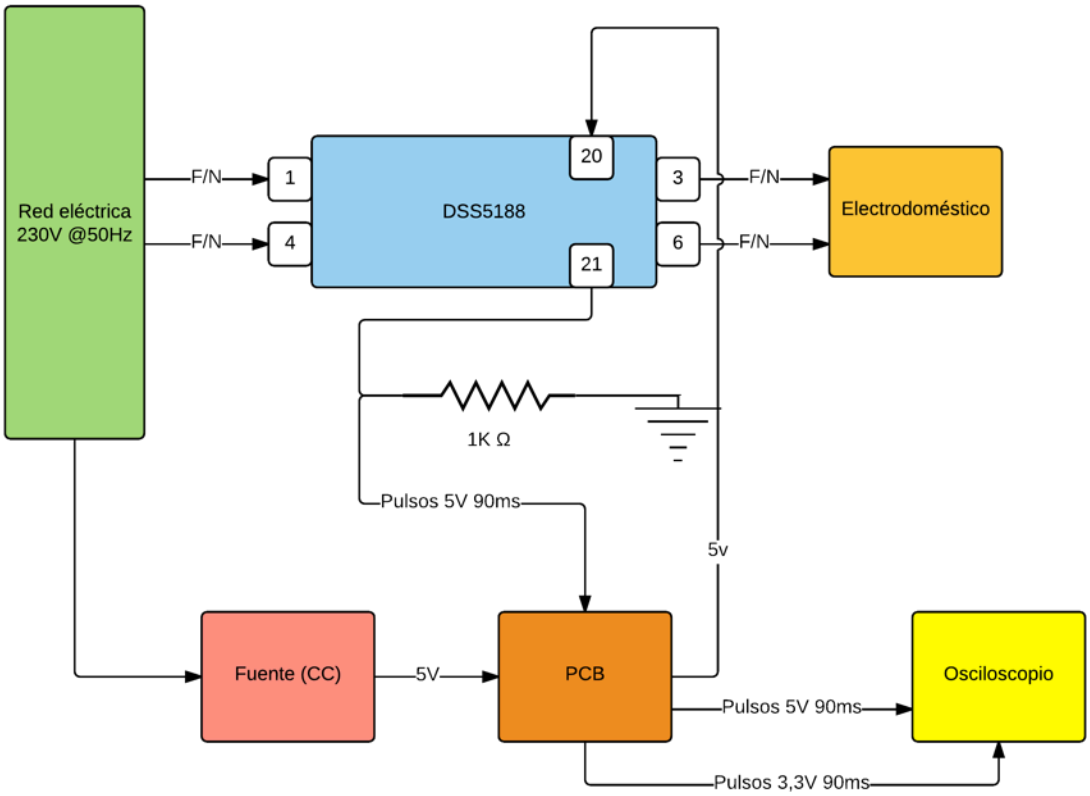


Figura 4.4. Sistema implementado para realizar prueba sistema medidor de consumo más PCB

Los resultados, que se muestran en la figura 4.5, fueron satisfactorios:

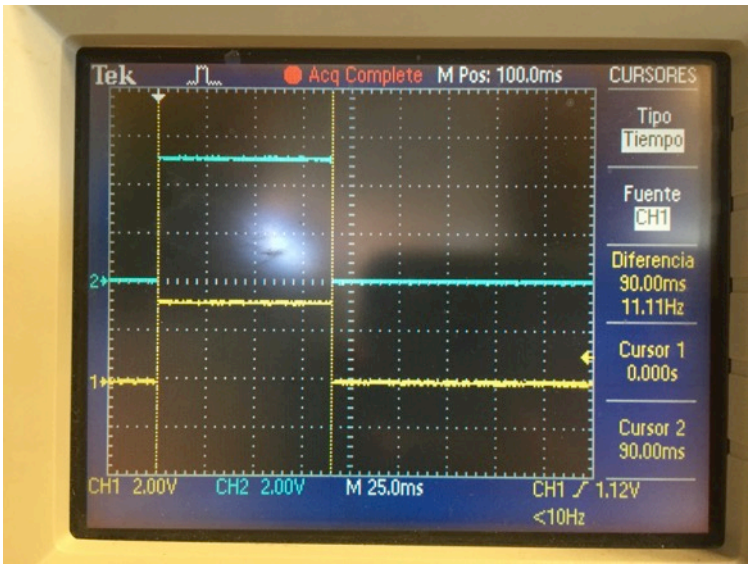


Figura 4.5. Canal amarillo: pulso a 3,3V. Canal azul: pulso a 5V

4.2 Unidad central

Las pruebas realizadas a la unidad central consistieron en dos partes:

- Comprobar que el módulo XBee funciona correctamente
- Comprobar que el software de procesado de datos procesa correctamente los datos de las mediciones y se comunica correctamente con la base de datos de la aplicación web.

4.2.1 Pruebas al PCB de la unidad central

La prueba consistió en conectar el módulo XBee al PCB realizar las siguientes comprobaciones:

- Las tensiones en los pines del XBee eran correctas.
- Probar mediante una terminal serie que la comunicación serie con el módulo también funciona correctamente.

4.2.1 Pruebas al software de procesado y aplicación web

Con esta prueba se pretendía comprobar que el software procesaba adecuadamente los pulsos del medidor, generando datos coherentes de mediciones e insertándolos correctamente en la base de datos de la aplicación web.

Para probar de manera aislada el software, se sustituyó la función que detecta los pulsos enviados por el módulo XBee del nodo remoto por una simple pulsación de la tecla “intro”. De esta manera, cada vez que se pulsara esta tecla, correspondería con un pulso que indica que 0,5Wh han sido consumidos en el nodo remoto.

A fin de comprobar la conexión con la base de datos, y el correcto funcionamiento de la aplicación web, los datos se insertarían directamente en la base de datos alojada en el servidor remoto.

A continuación se detalla (tabla 4.1) la cantidad de veces que se pulsó la tecla “intro” en cada intervalo de tiempo y el consumo asociado a estas pulsaciones que la aplicación web debería mostrar.

Hora de comienzo del intervalo de 30s	19:23h	19:26h	19:26:30h	19:27h	19:27:30h	19:28h
Pulsos	1	1	7	5	2	1
Energía consumida (Wh)	0,5	0,5	3,5	2,5	1	0,5

Tabla 4.1

A continuación se muestran (figuras 4.6 y 4.7) los resultados mostrados gráficamente en la web:

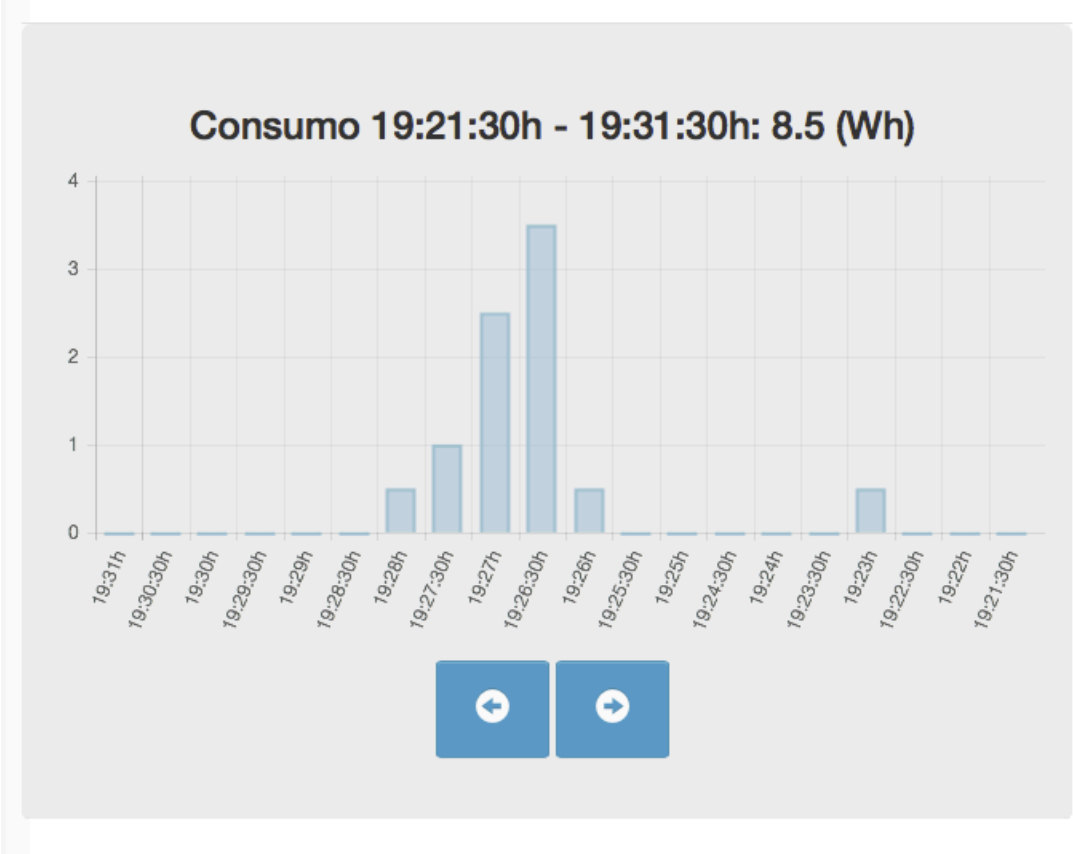


Figura 4.6. Gráfica correspondiente con los pulsos introducidos a modo de prueba (indicados en la tabla de la página anterior)

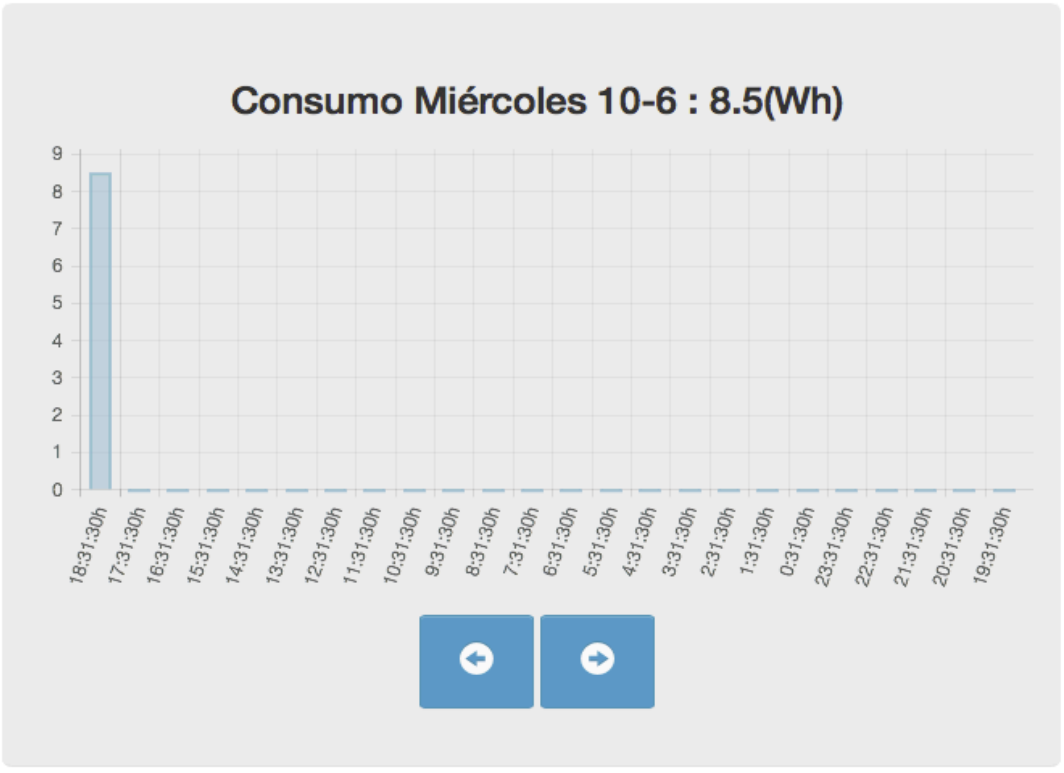


Figura 4.7. Gráfica de consumo en las últimas 24h. 8,5 Wh corresponde con la suma de todos los pulsos introducidos a modo de prueba (indicados en la tabla de la página anterior)

Para probar que el cálculo de consumo en un intervalo concreto funcionaba correctamente, se calculó el consumo del miércoles 10 de Junio entre las 19:22h y las 19:27h, que debería ser 7Wh (figura 4.8).

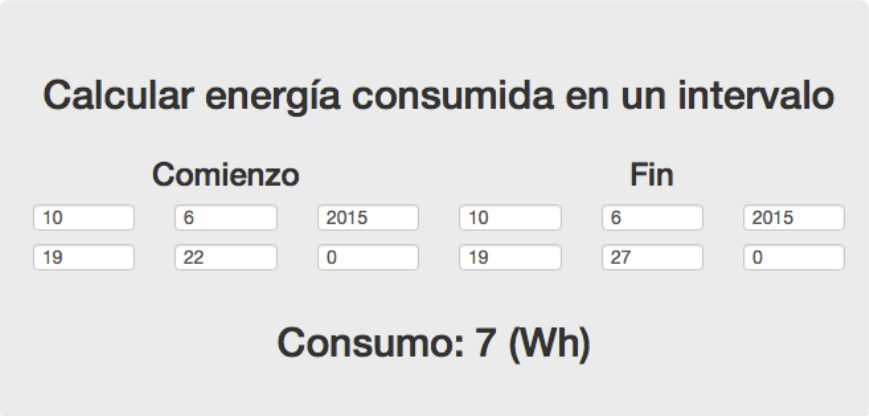


Figura 4.8. Cálculo de consumo energérito entre las 19:22h y las 19:27h

4.4 Pruebas sistema completo

Para probar el sistema completo, se conectó una plancha cuyo consumo se deseaba medir. Gracias al led que incorpora el medidor, podemos saber cuántos pulsos (correspondientes a 0,5Wh de consumo cada uno) se han generado.

Al conectar la plancha, hay un pico de consumo hasta que ésta alcanza la temperatura óptima, y después el consumo disminuye considerablemente ya que sólo tiene que mantener esta temperatura. En la figura 4.9 se puede apreciar un pico de consumo cuando la plancha se está calentando (11:18h-11:19h), un intervalo de consumo constante cuando la plancha ya ha alcanzado la temperatura óptima y ésta no se encuentra en contacto con nada (11:19:30h-11:21:30h), otro pico de consumo más alto cuando hacemos perder calor a la plancha al moviéndola por una superficie fría como el suelo (11:22h-11:23h), y otro intervalo de consumo constante cuando dejamos la plancha quieta en el suelo (11:24h-11:25:30h).

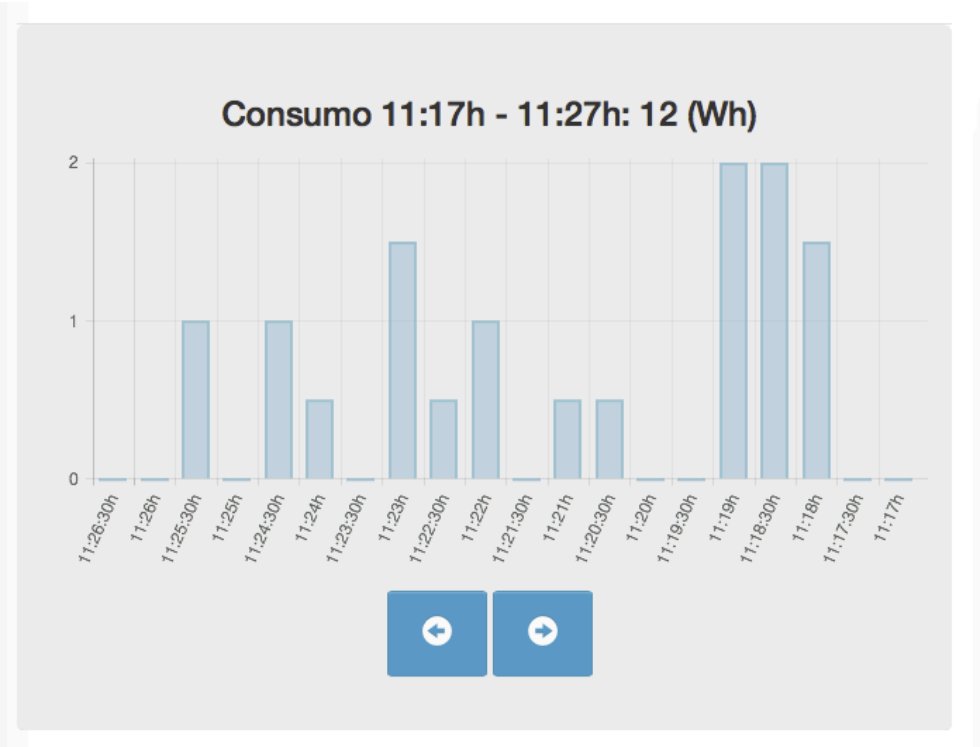


Figura 4.9

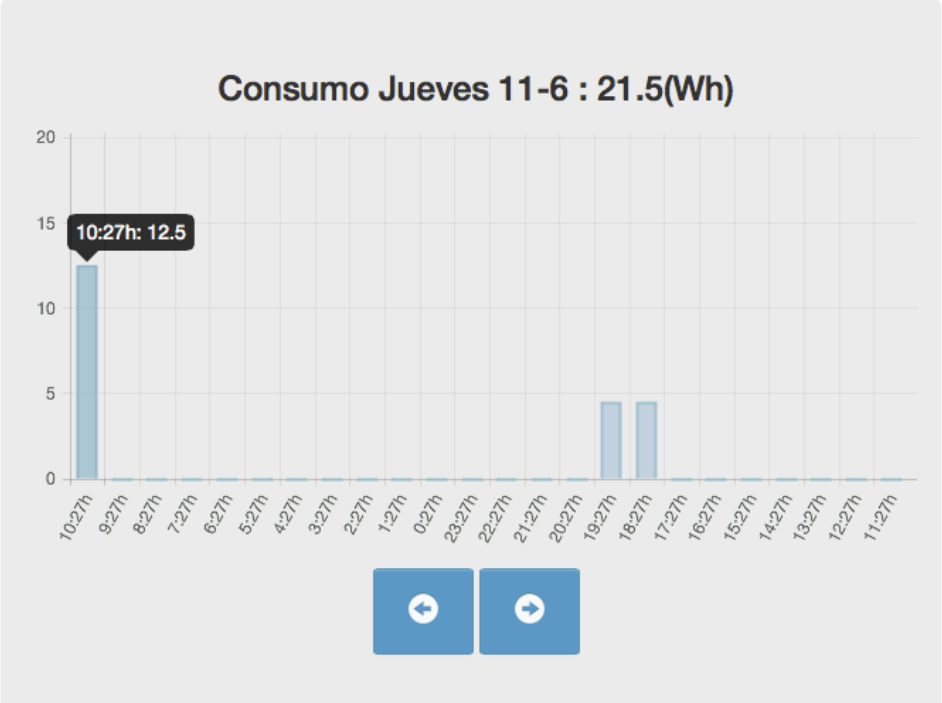


Figura 4.10. Consumo en las últimas 24h. Los datos de las 10:27h corresponden con la suma de consumo de la gráfica 4.4

5. Conclusiones y trabajo futuro.

5.1 Conclusiones.

Para desarrollar este sistema de medición de consumo energético ha sido necesario el estudio de múltiples tecnologías que pudieran llevar a cabo las funciones requeridas para este proyecto. No ha sido fácil, ya que para desarrollar este proyecto es necesaria la integración de tecnologías con fines tan diferentes como procesado de señal, procesado de datos, creación de enlaces inalámbricos ó desarrollo de plataformas web dinámicas.
Aun así, después de mucho trabajo el objetivo del proyecto se ha cumplido satisfactoriamente.

5.2 Trabajo futuro

En vistas a dar continuidad a este proyecto, teniendo en cuenta las grandes posibilidades de la domótica, y que las tecnologías utilizadas para el desarrollo de este proyecto proporcionan grandes posibilidades de expansión del mismo, me gustaría dejar algunas propuestas de trabajo futuro para dar continuidad a este proyecto:

- Incorporación de más nodos al sistema desarrollado. En este proyecto se ha desarrollado un sistema de un solo nodo remoto, una unidad central y una aplicación web. No obstante, como se ha comentado a lo largo de esta memoria, tanto la estructura del sistema, como las tecnologías empleadas en ella, fueron elegidas con vistas a la incorporación de numerosos nodos.
  Para la incorporación de varios nodos, habría que ajustar el código de la unidad central:
  - Tendría que insertar las entradas de la base de datos, aparte de los campos de tiempo y consumo, otro con el identificador del nodo remoto que ha llevado a cabo la medición. Este identificador puede ser obtenido fácilmente, ya que aunque actualmente no se ha usado, éste se encuentra en uno de los campos de la trama API recibida.
  - En la aplicación web, habría que replicar la plantilla que aparece para un solo nodo, e insertar los datos en función del nuevo campo que distingue las mediciones de cada nodo.
- Incorporación de funcionalidades a cada nodo. Dadas las capacidades de cada módulo XBee que va a estar presente en cada nodo (consta de E/S digitales y conversores A/D que pueden ser utilizados remotamente por tramas API), las capacidades de cada nodo no están

siendo totalmente explotadas, siendo posible añadir funcionalidades fácilmente, entre ellas:

- Añadir sensores analógicos, como por ejemplo de luz y de humedad, ya que los módulos Xbee disponen de conversores A/D que no se están usando.

- Añadir actuadores: ya que el ZigBee tiene muchas E/S digitales que no se están usando, se podrían añadir actuadores que actuaran en función de los datos recogidos por los sensores.

Bibliografía.

[1] Robert Faludi. *Building Wireless Sensor Networks*. Editorial O´Reilly

[2] STPM10 Datasheet, *http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/CD00278872.pdf*

[3] *Documentación oficial de Raspberry Pi,* https://www.raspberrypi.org/documentation/

[4] Tom Coleman & Sacha Greif. *Discover Meteor. Building Real-Time Javascript Web Apps.*

[5] Meteor Official Documentation: http://docs.meteor.com/#/full/

[6]uA78m00 datasheet

[7] txb0104 Datasheet, *http://www.ti.com/lit/ds/symlink/txb0104.pdf*

[8] Wolfram Donat. *Learn Raspberry Pi Programming with Python.* Editorial: Technology in action.

[9] David Beazley & Brian K. Jones. *Python Cookbook,* Editorial O´Reilly

[10] Mario Rodriguez Cerezo. Proyecto de fin de carrera: *Sistema de control remoto para aplicaciones domóticas a través de internet.*

https://www.westfloridacomponents.com/mm5/graphics/datasheets/2/UA78M33CDCYRG3.pdf

Apéndices

A. Presupuesto.

1. Ejecución material

- Ordenador personal.....2249€
- Raspberry Pi Starter Kit.....60€
- Kit de desarrollo ZigBee y módulos Xbee.....209€
- Fabricación PCB y componentes.....50€
- Medidor de consumo.....15€
- Total ejecución material.....2583€

2. Gastos generales

- 16% sobre ejecución material.....413€

3. Beneficio industrial

- 640 horas a 15€ / h.....9600€

4. Subtotal del presupuesto .....12595€

5. I.V.A aplicable

- 21% del subtotal del presupuesto.....2645€

6. Total.....15240€

Madrid, Julio 2015
El ingeniero jefe del proyecto
Fdo.: Javier Fernández Tapia
Ingeniero de Telecomunicación

B. Pliego de condiciones

Este documento contiene las condiciones legales que guiarán la realización, en este proyecto, de "Elaboración de un sistema para realización de auditorías de consume de energía eléctrica a través de internet". En lo que sigue, se supondrá que el proyecto ha sido encargado por una empresa cliente a una empresa consultora con la finalidad de realizar dicho sistema. Dicha empresa ha debido desarrollar una línea de investigación con objeto de elaborar el proyecto. Esta línea de investigación, junto con el posterior desarrollo de los programas está amparada por las condiciones particulares del siguiente pliego.
Supuesto que la utilización industrial de los métodos recogidos en el presente proyecto ha sido decidida por parte de la empresa cliente o de otras, la obra a realizar se regulará por las siguientes:

Condiciones generales

1. La modalidad de contratación será el concurso. La adjudicación se hará, por tanto, a la proposición más favorable sin atender exclusivamente al valor econó- mico, dependiendo de las mayores garantías ofrecidas. La empresa que somete el proyecto a concurso se reserva el derecho a declararlo desierto.

2. El montaje y mecanización completa de los equipos que intervengan será realizado totalmente por la empresa licitadora.

3. En la oferta, se hará constar el precio total por el que se compromete a realizar la obra y el tanto por ciento de baja que supone este precio en relación con un importe límite si este se hubiera fijado.

4. La obra se realizará bajo la dirección técnica de un Ingeniero de Telecomunicación, auxiliado por el número de Ingenieros Técnicos y Programadores que se estime preciso para el desarrollo de la misma.

5. Aparte del Ingeniero Director, el contratista tendrá derecho a contratar al resto del personal, pudiendo ceder esta prerrogativa a favor del Ingeniero Director, quien no estará obligado a aceptarla.

6. El contratista tiene derecho a sacar copias a su costa de los planos, pliego de condiciones y presupuestos. El Ingeniero autor del proyecto autorizará con su firma las copias solicitadas por el contratista después de confrontarlas.

7. Se abonará al contratista la obra que realmente ejecute con sujeción al proyecto

que sirvió de base para la contratación, a las modificaciones autorizadas por la superioridad o a las órdenes que con arreglo a sus facultades le hayan comunicado por escrito al Ingeniero Director de obras siempre que dicha obra se haya ajustado a los preceptos de los pliegos de condiciones, con arreglo a los cuales, se harán las modificaciones y la valoración de las diversas unidades sin que el importe total pueda exceder de los presupuestos aprobados. Por consiguiente, el número de unidades que se consignan en el proyecto o en el presupuesto, no podrá servirle de fundamento para entablar reclamaciones de ninguna clase, salvo en los casos de rescisión.

8. Tanto en las certificaciones de obras como en la liquidación final, se abonarán los trabajos realizados por el contratista a los precios de ejecución material que figuran en el presupuesto para cada unidad de la obra.

9. Si excepcionalmente se hubiera ejecutado algún trabajo que no se ajustase a las condiciones de la contrata pero que sin embargo es admisible a juicio del Ingeniero Director de obras, se dará conocimiento a la Dirección, proponiendo a la vez la rebaja de precios que el Ingeniero estime justa y si la Dirección resolviera aceptar la obra, quedará el contratista obligado a conformarse con la rebaja acordada.

10. Cuando se juzgue necesario emplear materiales o ejecutar obras que no figuren en el presupuesto de la contrata, se evaluará su importe a los precios asignados a otras obras o materiales análogos si los hubiere y cuando no, se discutirán entre el Ingeniero Director y el contratista, sometiéndolos a la aprobación de la Dirección. Los nuevos precios convenidos por uno u otro procedimiento, se sujetarán siempre al establecido en el punto anterior.

11. Cuando el contratista, con autorización del Ingeniero Director de obras, emplee materiales de calidad más elevada o de mayores dimensiones de lo estipulado en el proyecto, o sustituya una clase de fabricación por otra que tenga asignado mayor precio o ejecute con mayores dimensiones cualquier otra parte de las obras, o en general, introduzca en ellas cualquier modificación que sea benefi- ciosa a juicio del Ingeniero Director de obras, no tendrá derecho sin embargo, sino a lo que le correspondería si hubiera realizado la obra con estricta sujeción a lo proyectado y contratado.

12. Las cantidades calculadas para obras accesorias, aunque figuren por partida alzada en el presupuesto final (general), no serán abonadas sino a los precios de la contrata, según las condiciones de la misma y los proyectos particulares que para ellas se formen, o en su defecto, por lo que resulte de su medición final.

13. El contratista queda obligado a abonar al Ingeniero autor del proyecto y direc- tor de obras así como a los Ingenieros Técnicos, el importe de sus respectivos honorarios facultativos por formación del proyecto, dirección técnica y admi-

nistración en su caso, con arreglo a las tarifas y honorarios vigentes.

14. Concluida la ejecución de la obra, será reconocida por el Ingeniero Director que a tal efecto designe la empresa.

15. La garantía definitiva será del 4 % del presupuesto y la provisional del 2 %.

16. La forma de pago será por certificaciones mensuales de la obra ejecutada, de acuerdo con los precios del presupuesto, deducida la baja si la hubiera.

17. La fecha de comienzo de las obras será a partir de los 15 días naturales del replanteo oficial de las mismas y la definitiva, al año de haber ejecutado la provisional, procediéndose si no existe reclamación alguna, a la reclamación de la fianza.

18. Si el contratista al efectuar el replanteo, observase algún error en el proyecto, deberá comunicarlo en el plazo de quince días al Ingeniero Director de obras, pues transcurrido ese plazo será responsable de la exactitud del proyecto.

19. El contratista está obligado a designar una persona responsable que se entende- rá con el Ingeniero Director de obras, o con el delegado que éste designe, para todo relacionado con ella. Al ser el Ingeniero Director de obras el que interpre- ta el proyecto, el contratista deberá consultarle cualquier duda que surja en su realización.

20. Durante la realización de la obra, se girarán visitas de inspección por personal facultativo de la empresa cliente, para hacer las comprobaciones que se crean oportunas. Es obligación del contratista, la conservación de la obra ya ejecutada hasta la recepción de la misma, por lo que el deterioro parcial o total de ella, aunque sea por agentes atmosféricos u otras causas, deberá ser reparado o reconstruido por su cuenta.

21. El contratista, deberá realizar la obra en el plazo mencionado a partir de la fecha del contrato, incurriendo en multa, por retraso de la ejecución siempre que éste no sea debido a causas de fuerza mayor. A la terminación de la obra, se hará una recepción provisional previo reconocimiento y examen por la dirección técnica, el depositario de efectos, el interventor y el jefe de servicio o un representante, estampando su conformidad el contratista.

22. Hecha la recepción provisional, se certificará al contratista el resto de la obra, reservándose la administración el importe de los gastos de conservación de la misma hasta su recepción definitiva y la fianza durante el tiempo señalado como plazo de garantía. La recepción definitiva se hará en las mismas condiciones que la provisional, extendiéndose el acta correspondiente. El Director Técnico propondrá

a la Junta Económica la devolución de la fianza al contratista de acuerdo con las condiciones económicas legales establecidas.

23. Las tarifas para la determinación de honorarios, reguladas por orden de la Presidencia del Gobierno el 19 de Octubre de 1961, se aplicarán sobre el denomi- nado en la actualidad “Presupuesto de Ejecución de Contrata” y anteriormente llamado ”Presupuesto de Ejecución Material” que hoy designa otro concepto.

Condiciones particulares

La empresa consultora, que ha desarrollado el presente proyecto, lo entregará a la empresa cliente bajo las condiciones generales ya formuladas, debiendo añadirse las siguientes condiciones particulares:

1. La propiedad intelectual de los procesos descritos y analizados en el presen- te trabajo, pertenece por entero a la empresa consultora representada por el Ingeniero Director del Proyecto.

2. La empresa consultora se reserva el derecho a la utilización total o parcial de los resultados de la investigación realizada para desarrollar el siguiente proyecto, bien para su publicación o bien para su uso en trabajos o proyectos posteriores, para la misma empresa cliente o para otra.

3. Cualquier tipo de reproducción aparte de las reseñadas en las condiciones generales, bien sea para uso particular de la empresa cliente, o para cualquier otra aplicación, contará con autorización expresa y por escrito del Ingeniero Director del Proyecto, que actuará en representación de la empresa consultora.

4. En la autorización se ha de hacer constar la aplicación a que se destinan sus reproducciones así como su cantidad.

5. En todas las reproducciones se indicará su procedencia, explicitando el nombre del proyecto, nombre del Ingeniero Director y de la empresa consultora.

6. Si el proyecto pasa la etapa de desarrollo, cualquier modificación que se realice sobre él, deberá ser notificada al Ingeniero Director del Proyecto y a criterio de éste, la empresa consultora decidirá aceptar o no la modificación propuesta.

7. Si la modificación se acepta, la empresa consultora se hará responsable al mismo nivel que el proyecto inicial del que resulta el añadirla.

8. Si la modificación no es aceptada, por el contrario, la empresa consultora de-

clinará toda responsabilidad que se derive de la aplicación o influencia de la misma.

9. Si la empresa cliente decide desarrollar industrialmente uno o varios productos en los que resulte parcial o totalmente aplicable el estudio de este proyecto, deberá comunicarlo a la empresa consultora.

10. La empresa consultora no se responsabiliza de los efectos laterales que se puedan producir en el momento en que se utilice la herramienta objeto del presente proyecto para la realización de otras aplicaciones.

11. La empresa consultora tendrá prioridad respecto a otras en la elaboración de los proyectos auxiliares que fuese necesario desarrollar para dicha aplicación industrial, siempre que no haga explícita renuncia a este hecho. En este caso, deberá autorizar expresamente los proyectos presentados por otros.

12. El Ingeniero Director del presente proyecto, será el responsable de la dirección de la aplicación industrial siempre que la empresa consultora lo estime opor- tuno. En caso contrario, la persona designada deberá contar con la autorización del mismo, quien delegará en él las responsabilidades que ostente.