INICIACIÓN A LA INVESTIGACIÓN
REVISTA ELECTRÓNICA
UNIVERSIDAD DE JAÉN
Ini Inv, e5: a12 (2013)

Construyendo la Web de las Cosas: Ahorro energético y gestión de iluminación en ciudades

Francisco J. Estévez, Vicente Ruiz y Jesús González

Programa de Máster en Arquitectura de Computadores y Redes. Departamento de Arquitectura y Tecnología de Computadores. Universidad de Granada. Granada, España.
fjestevez@ieee.org

Resumen

Se están realizando muchos esfuerzos para acercar aquellos objetos de la vida cotidiana a Internet. A lo largo de estos últimos años han emergido una gran cantidad de tecnologías como comunicación por proximidad -*RFID* o *NFC*-, localización en tiempo real o sensores embebidos. Además la computación distribuida, las tecnologías móviles y la conectividad en cualquier lugar están cobrando una mayor relevancia. La unión de todo esto nos permite transformar objetos de cada día en objetos inteligentes que pueden entender y reaccionar ante su entorno.

En este artículo vamos a presentar como llevar toda esta tecnología a la vida real. Vamos a describir cómo implantar dispositivos en farolas y cómo hemos realizado diversos demostradores, de forma que se ha controlado, regulado y monitorizado remotamente su consumo energético. A lo largo del trabajo describiremos las pautas a seguir para llevar estas propuestas a nuestras ciudades.

INTRODUCCIÓN

Internet de las Cosas, también llamado *IoT*, hace referencia a objetos identificables de forma única y a su representación virtual en Internet. *Kevin Ashton* fue el primero en utilizar este término en 1999. La identificación a través de radio-frecuencia o *RFID*, a menudo se ve como algo necesario para el *Internet de las Cosas*. Un sistema de identificación es indispensable para que cualquier equipo sea capaz de reconocer y procesar aquellos objetos del mundo que nos rodean.

En los últimos años, especialmente en este último, el término *IoT* está empezando a centrar numerosos estudios. Hay muchas propuestas y numerosos artículos al respecto, en los que se esta tratando de conseguir heterogeneidad. Se está comenzando a hablar acerca de dispositivos empotrados, no sólo sensores sino también actuadores, incluso de integración con *SmartPhones*, de interacción natural o de información ubicua, todas estas áreas amplían el área del *IoT* aunando cada vez más corrientes y afianzando su expansión.

Cuando hablamos de *Internet de las Cosas*, no sólo estamos diciendo que todas las cosas tienen presencia en la red, sino más importante aún, hay que ver todas las posibilidades que ello ofrece. El hecho de disponer de conexión en cada objeto nos permite crear una red inteligente, una mina de conocimiento y capacidad de cómputo. Por tanto, lo interesante no es que los dispositivos estén conectados a Internet, sino todo lo que se puede hacer teniendo todos estos dispositivos conectados.

Por otro lado, dado el contexto económico actual encontramos muchas investigaciones enfocadas al ahorro energético y a la mejora en la explotación de los recursos. La energía y su precio son puntos clave en tiempos de crisis, ya que pueden ayudar o perjudicar seriamente una rápida recuperación económica. Algo tan básico como la iluminación de todas las calles de una ciudad es muy costoso. Esto necesita una inversión de miles o cientos de miles de euros cada día, dependiendo de la ciudad y su expansión geográfica.

¿Realmente es necesario tener todas las farolas de una ciudad funcionando a su máxima potencia durante toda la noche? La respuesta, evidentemente, es no. Es posible jugar con la intensidad de las bombillas para realizar un alumbrado inteligente, que además nos permita un ahorro significativo. Cuando empieza a anochecer, no es necesario que una bombilla esté a su máxima potencia. Tenemos datos que nos permiten saber cuándo podemos subir o bajar la intensidad de la luz, con el ahorro que esto conlleva.

Cuando empieza a anochecer, se puede establecer una intensidad mínima que compense la falta de luz y que se irá incrementando en intervalos de tiempo prefijados hasta alcanzar la máxima intensidad, una vez entrada la noche. Además, sabemos que en invierno a partir de media noche, hay poco tránsito en las calles, por lo que a partir de una hora umbral prefijada, podemos comenzar a decrementar la intensidad progresivamente hasta alcanzar unos mínimos de seguridad y volverla a incrementar, por ejemplo, antes de las horas de mayor tránsito o del amanecer.

En definitiva, esta área de investigación abarca numerosos campos, desde el económico o el psicológico, hasta el de la electrónica o la informática. Dado el reto, y en este escenario en particular, conociendo los hábitos de los ciudadanos en las distintas épocas del año, el gasto energético se puede ajustar y reducir al mínimo. Incluso se puede explorar el uso de otros dispositivos de detección, en adición a la infraestructura de control y comunicación, de forma que se permita el apagado completo de la iluminación.

A continuación se analiza el estado actual de las tecnologías aplicadas, así como algunos de los problemas que surgen de su uso, para, a continuación describir la propuesta de arquitectura en detalle y su implementación. Finalmente se presenta una justificación a partir de la puesta en marcha de un demostrador.

ESTADO DEL ARTE

En los últimos años Internet ha arrastrado una gran problemática: las direcciones IP, tal y como las conocemos hoy día, se están agotando. Esto supondría una gran barrera para incorporar una gran cantidad de dispositivos a la red, ya que no dispondríamos de suficientes direcciones. Sólo hay que pensar que cada objeto que nos rodea necesite de una IP que le permita conectarse para darse cuenta de la cantidad ingente de direcciones que necesitamos.

En Junio de 2011 se celebró el día mundial de *IPv6*, con lo que este problema, al menos por unos años, se ha solucionado. Con respecto a la versión anterior del protocolo IP, *IPv4*, la longitud de las direcciones ha pasado de 32 bits a 128 bits, o lo que es lo mismo, se pasa de tener 2^{32} direcciones IP a tener 2^{128}. Con esto se puede garantizar que habrá suficientes direcciones IP para todos los dispositivos. Desde 2011 y durante estos años, Internet está migrando hacía esta nueva versión. Por este motivo, en este estudio, hemos apostado directamente por utilizar *IPv6*.

Cuando hablamos de crear redes de cientos o miles de dispositivos, el cableado necesario para la comunicación es un gran problema. Esto puede encarecer el despliegue, o incluso hacer el proyecto inviable. Por este motivo es necesario utilizar tecnologías que permitan la comunicación inalámbrica.

Hoy en día vivimos rodeados de tecnologías de comunicaciones inalámbricas: *Wi-Fi*, *Bluetooth*, *3G*, *HSDPA*, *GPS*, etc. Estas tecnologías disponibles a nivel doméstico y de uso cotidiano no son del todo ideales para dispositivos empotrados. Esto se debe a que no se ajustan a las necesidades que hemos considerado como básicas. Es importante recordar en qué punto nos encontramos: estamos hablando de dispositivos que tienen una gran cantidad de restricciones que debemos de tener en cuenta. En general, todas estos dispositivos o motas suelen tener fuertes restricciones en el consumo energético, el tamaño y la potencia de cómputo.

Tecnologías como *Wi-Fi*, que cumple con el estándar *IEEE 802.11* o *Bluetooth*, con el estándar *IEEE 802.15.1*, se diseñaron para transmitir grandes cantidades de información, lo cual no es de interés en los escenarios como el aquí propuesto. No necesitamos altas tasas de intercambio de información para poder controlar un actuador o leer la medición de un sensor. *Wi-Fi* se diseñó pensando en computadores, donde el consumo energético no es una restricción y *Bluetooth* para dispositivos portátiles. Por ejemplo, *Wi-Fi* tiene un consumo superior a 700mW, tanto en emisión como en recepción, y *Bluetooth* tiene un consumo máximo de 100mW.

Puede parecer interesante utilizar *Bluetooth*, que además ha sido adoptado por un buen número de dispositivos de consumo, pero esta tecnología también tiene sus limitaciones. Para empezar, las conexiones deben de ser explícitas y aceptadas por los dos dispositivos antes de empezar la transmisión. Pero el principal inconveniente es que el número máximo de dispositivos conectados a una red Bluetooth es 8 (7 esclavos más un maestro). Si nuestra intención es conectar decenas, cientos o incluso miles de dispositivos a la misma red, automáticamente podemos descartarlo.

En este trabajo hemos apostado por el estándar *IEEE 802.15.4*, el cuál define las capas física y de control de acceso al medio para redes inalámbricas de área personal de baja potencia, *LoWPANs* o *low-power wireless personal area networks*, por sus siglas en inglés. En este estándar se basan algunas tecnologías como *ZigBee* o *6LowPAN*, las cuales fueron concebidas para dispositivos empotrados. *ZigBee* especifica un conjunto de protocolos de comunicación de alto nivel y se encuentra mantenido por la *ZigBee Alliance*. *ZigBee* es actualmente, objeto de numerosos estudios, a lo largo de todas sus propiedades. Una de las formas que tiene *ZigBee* de conseguir interoperatibilidad es a través de los conocidos como perfiles de aplicación. Estos perfiles son definidos por la *ZigBee Alliance*, y proveen una descripción de los dispositivos soportados para una aplicación concreta junto a un esquema de mensajes utilizado en la comunicación de los dispositivos.

La especificación *ZigBee* diferencia tres tipos de dispositivos distintos que se encuentran normalmente en una red:

- *Coordinator (ZC)*: organiza la red y mantiene las tablas de enrutamiento.
- *Routers (ZR)*: pueden hablar con el *Coordinator*, con otros *Routers* y con *End devices*.
- *End devices (ZED)*: pueden hablar con *Routers* y con el *Coordinator*, pero no con otros *End devices*.

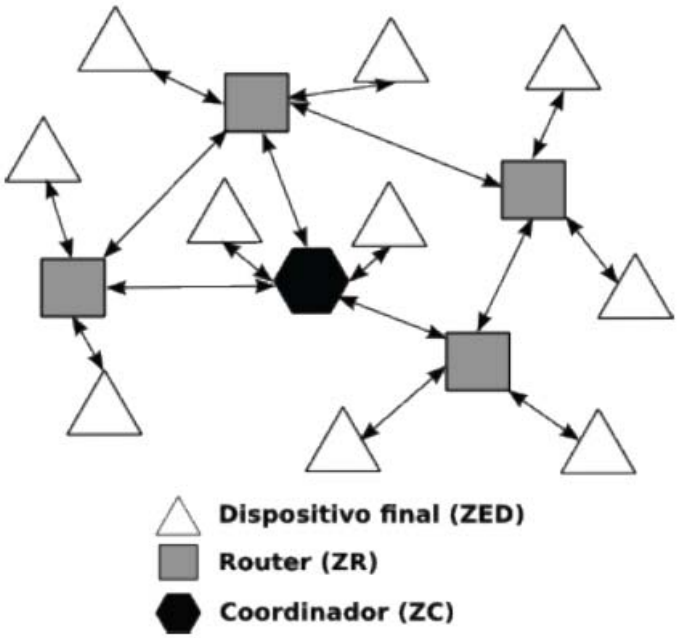

Figura 1. Red *ZigBee* típica

Esto nos conlleva una serie de problemas. En una red *Zigbee* es necesario tener un *Coordinator*, el cuál inicia y mantiene la red, o lo que es lo mismo, si este dispositivo falla, la red entera caerá y quedará inoperativa. Aunque *ZigBee* soporta múltiples topologías, todas ellas necesitan del *Coordinator*. Esto hace que la arquitectura no sea muy robusta. Para el escenario propuesto, necesitamos una red en malla capaz de autogestionarse con una topología en malla que sea totalmente robusta y pueda recuperarse ante eventuales fallos.

Por otro lado, la *ZigBee Alliance* ha definido únicamente cinco perfiles desde 2004, fecha en la que apareció la especificación 1.0 de *ZigBee*, por lo que si queremos implementar una aplicación distinta necesitaremos crear nuestro propio perfil, con lo que habremos perdido interoperabilidad con el resto de dispositivos. Además, la necesidad de definir perfiles no se ajusta mucho al hecho de tener cualquier dispositivo conectado a Internet, ya que hay una enorme cantidad de dispositivos distintos.

Estos motivos nos han hecho decantarnos por el uso de *6LowPAN* [1]. Aunque esta tecnología inalámbrica aún no es un estándar [2], su desarrollo está muy avanzado y se está comenzando a utilizar en multitud de dispositivos. *6LowPAN* define el formato de las tramas para la transmisión de paquetes *IPv6* a través de *IEEE 802.15.4*. La mejor forma de trabajar con *Internet de las Cosas* es utilizando el protocolo *IP*, así podremos llegar directamente a cada dispositivo sin necesidad de una puerta de enlace o *Gateway*, que haga traducciones entre distintos protocolos.

No sólo disponemos de todas las ventajas ya comentadas, sino además de las ventajas que *IPv6* nos ofrece por defecto. En *IPv4* es necesario utilizar un servidor *DHCP* para evitar la configuración manual de todos y cada uno de los dispositivos de la red. En *IPv6* aparece un nuevo método de configuración autónoma. De aquí en adelante, nos referiremos a *IPv6* como *IP* por comodidad.

Un punto realmente importante, es que utilizando *IP* aseguramos la interoperabilidad de cualquier dispositivo con el resto de dispositivos de la red. Pero aún nos queda un último detalle por solventar: no hay un protocolo de aplicación estandarizado para que los dispositivos puedan hablar entre ellos. Tener

comunicación a través de *IP* no garantiza el entendimiento si cada dispositivo utiliza el protocolo que mejor se adapta a sus necesidades, o se diseña uno a medida. Esto impide que dos dispositivos de distintos fabricantes sean capaces de comunicarse entre sí.

Por tanto, existe la necesidad de utilizar un estándar lo suficientemente extendido para eliminar los problemas de interoperabilidad. En base a esto, aparece una nueva idea: la *Web de las Cosas*. En lugar de trabajar sobre *IP* + cualquier otro protocolo, la propuesta más interesante es trabajar sobre *HTTP*. Seguramente y con diferencia, *HTTP* sea el protocolo más utilizado, además de tener otras ventajas. La mayoría de *routers* y *firewalls* comerciales bloquean casi todos los puertos, excepto el puerto 80, utilizado en la navegación web, por ello resulta de interés la utilización de esta combinación de protocolos.

DETALLE DE LA ARQUITECTURA

Como se comentó anteriormente, la intención es construir una implementación de la *Web de las Cosas*. Para ello es necesario que cada bloque básico disponga de presencia en la red. El hecho de disponer de acceso a la nube en cada dispositivo puede encarecer enormemente el despliegue, lo cual, casi con toda seguridad, hará que un proyecto de estas características sea inviable. Para solventar esto, hemos optado por introducir una serie de *Routers* que dispondrán de conexión a Internet. Éstos *Routers* permitirán el acceso a la nube a todos los dispositivos empotrados que se conecten a ellos.

Podemos ver un esquema global en la siguiente figura:

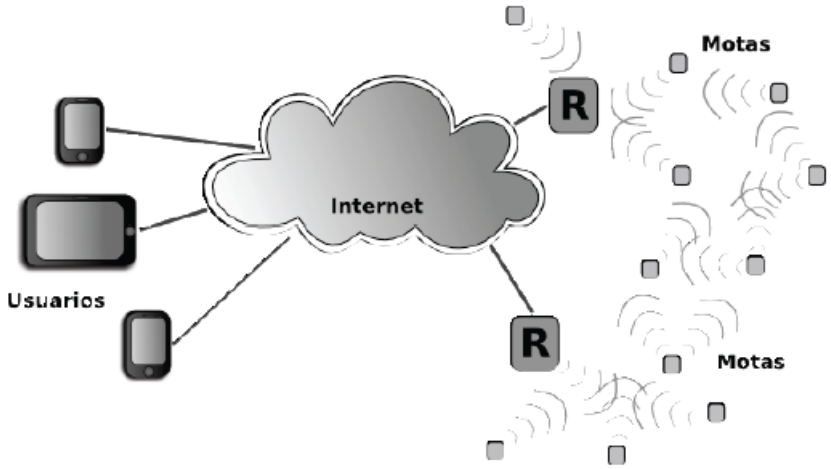

Figura 2. Esquema de la arquitectura

Cómo podemos ver, el acceso a la plataforma por parte del usuario se realiza a través de Internet desde cualquier dispositivo final, como puede ser un teléfono móvil, una tableta, un ordenador portátil, etc. Así, se ha desarrollado una interfaz que permita a cualquier usuario ver información acerca de la red e interactuar en tiempo real con cada uno de los dispositivos.

Dentro de esta arquitectura, los *Routers* son aquellos dispositivos marcados con una R y dirigen todo el tráfico hacia y desde Internet. Así, la petición de cualquier dispositivo de nuestra red a un servicio o recurso externo, será redirigida

por un *Router*; De la misma forma, se encaminará cualquier petición realizada desde Internet a una de nuestras motas.

Los *Routers* no tienen por qué ser iguales ni utilizar el mismo método de conexión a Internet, lo que nos garantiza el acceso a la red. Podemos disponer de un *Router* con conexión *DSL*, otro con conexión *3G/HSDPA* y otro con *WiMAX*, todos ellos en la misma red. Así, si se cae el acceso a Internet de uno de ellos, la red de motas continuará con acceso a Internet a través de otro. Además, siguiendo esta misma filosofía, la arquitectura admite distintas redes de dispositivos ubicadas en lugares diferentes y que interactúen entre ellas. Con esto se consigue tener una red distribuida.

Por último, las piezas clave de la arquitectura: las motas. Cada uno de estos dispositivos representan un bloque básico con presencia en Internet. Estos bloques básicos crean y mantienen la red de forma autónoma y automática, estableciendo rutas óptimas y rutas alternativas, para conseguir tolerancia a fallos y garantizar la comunicación.

A partir de este punto empezamos a hablar sobre distintos tipos de dispositivos, desde recolectores de información, como sensores de humedad, temperatura o presión, hasta actuadores, es decir, aquellos que actúan modificando el entorno con acciones tales como activar un motor que abra o cierre una ventana, encender o apagar una luz, etc.

IMPLEMENTACIÓN DE LA ARQUITECTURA

Una vez determinada la arquitectura vamos a exponer los detalles necesarios para su implementación. Para ello, volveremos a repasar cada parte de la arquitectura indicando qué es necesario y cómo llevarlo a la práctica. Partimos de que la intención es realizar una implementación de la Web de las Cosas, por lo que es lógico, que la interfaz de usuario sea una página web. Con esto no sólo nos beneficiamos de todo lo comentado anteriormente, sino que tenemos otras ventajas adicionales.

```
GET / ligth / status / HTTP / 1.1
Host : device1
User−Agent : NotGiven / 1.0
Authorization : Basic ZGFwAnk6d2VjZXJldZ==
Accept: application/json
```

Figura 3. Petición RESTful

Los dispositivos que puede utilizar el usuario final para acceder a la plataforma son muy distintos. Si se optase por una aplicación de escritorio, debería desarrollarse una versión para *GNU/Linux*, *MacOSX* y *Windows*, pero también para *Android* e *iPhone*. Si utilizamos una plataforma web, los usuarios de todos estos sistemas tendrán acceso directamente.

Por tanto, hemos utilizado un servidor disponible en Internet, que sin ser parte necesaria de la arquitectura nos simplifica la puesta en marcha. Además, este servidor recolecta la información de todas las redes disponibles en la plataforma, la procesa y la ofrece al usuario. De esta forma se evita que un usuario final necesite conocer la dirección *IP* de cada uno de los *Routers* para establecer la comunicación.

A su misma vez, los *Routers* recogen la información de todos los dispositivos que hay conectados a la red, e informan al servidor central. Así evitamos que el servidor esté constantemente preguntando a cada uno de los dispositivos, con lo que se reduce considerablemente el tráfico de red. Los únicos mensajes directos del servidor al dispositivo serán las peticiones de actuación, es decir, cuando se quiera enviar una orden.

Durante todo el documento hemos hablado de comunicación entre las motas. Para este cometido se ha elegido utilizar el protocolo *RESTful*, que cumple con la arquitectura *REST*. *REST* es un estilo arquitectónico que fue desarrollado en paralelo con *HTTP/1.1* y basado en el diseño existente de *HTTP/1.0*. Por tanto, haciendo uso de *WebServices* garantizamos la interoperabilidad, no solo entre dispositivos distintos, sino también entre aplicaciones diferentes. Además, este hecho permite independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro.

Cada una de las motas será un dispositivo empotrado con una radio que cumpla el estándar *IEEE 802.15.4*, sobre la que se realizarán todas las comunicaciones. Además, cada dispositivo tendrá implementado un servidor *RESTful*, a través del cual se podrá obtener información acerca de la mota o interactuar con ella. Por ejemplo:

Para facilitar la gestión de la red utilizaremos el método de configuración autónoma -*stateless autoconfiguration*- que ofrece *IP*. De esta forma no será necesario añadir servidores *DHCPv6* [3] en la red. Como podemos observar en la **Figura 4**, se ha optado por una topología en malla. Para ello hemos utilizado el protocolo de enrutamiento *RPL* [4], ideado para redes de bajo consumo y con pérdidas de paquetes.

```
HTTP/1.1 200 Ok
Date: Wed, 29 Jun 2011 17:17:17 GMT
Content−Length : nnn
Content−Type: application / json
{
"intensity": "20",
"consumption": "10W"
}
```

Figura 4. Respuesta RESTful

PUESTA EN MARCHA

En este punto tenemos especificada la arquitectura y disponemos de suficientes detalles acerca de su implementación. Ahora toca aplicar todo esto a nuestro problema: el ahorro energético y la gestión de iluminación en ciudades.

Una parte muy interesante para el escenario que proponemos, es que se puedan distinguir varios perfiles de usuario dentro de la plataforma. Como nuestro objetivo es la gestión de la iluminación en una ciudad, no podemos permitir que cualquier usuario apague o encienda una farola. Para este cometido debemos filtrar a los usuarios que pueden hacer uso de la infraestructura aquí propuesta.

Pero se ofrece la posibilidad de que todos los usuarios tengan acceso, con diferentes roles. Los usuarios podrán ver los consumos y el gasto del alumbrado, y hacer un seguimiento de toda esta información.

Para el desarrollo de la aplicación de usuario hemos utilizado *Django* [5]. *Django* es un *framework* web de alto nivel desarrollado en *Python* que fomenta el desarrollo rápido y un diseño limpio y pragmático. La ventaja de utilizar *Python* es la velocidad de desarrollo y la facilidad a la hora de realizar pruebas. Esta web queda alojada en un servidor en Internet. A parte del servidor web, será también necesario un servidor de bases de datos para almacenar la información de todos los dispositivos.

Para poder controlar cada una de las farolas de forma independiente, hemos introducido un dispositivo empotrado en cada una. Debido a la abundancia de farolas, es factible montar una red con topología en malla, tal y como la que se ha explicado con anterioridad. Así, las propias farolas mantendrán información sobre el estado de la red y los consumos energéticos. Todo esto se puede monitorizar a través de la página web. Además, el hecho de disponer de esta información, nos permite la posibilidad de generar un sistema de alertas automáticas, por ejemplo, cuando se funda una bombilla o cuando una farola tenga un consumo mayor del esperado.

Contiki [6] es un sistema operativo diseñado para dispositivos empotrados, en especial con micros de 8 y 16 bits. Además, este sistema tiene soporte nativo para 6LowPAN y es *IPv6 Ready* desde hace ya algún tiempo, por lo que está completamente preparado para dar el salto a la nueva generación de direcciones *IP*. *Contiki* también ofrece una pila *TCP/IP*, llamada *uIP* o micro IP, capaz de funcionar en microcontroladores de 8 bits. Por estos motivos, se ha elegido utilizar *Contiki* para las motas.

Las motas pueden realizar cualquier función, desde el control de la calefacción o el riego automático hasta la gestión de la iluminación. Para ello, los dispositivos tienen un bus de comunicaciones en el cuál hay disponibles un *SPI*, dos *PWMs*, dos conversores Analógico-Digital, dos conversores Digital-Analógico, una señal de reloj, una señal de *reset* y dos líneas para la alimentación externa.

Realmente, para este trabajo es necesario el control de las bombillas de las farolas, por lo tanto, a este bus tendremos conectada una etapa de potencia de amplio rango. Este hardware adicional nos permitirá el control de la intensidad de las bombillas ubicadas en las farolas. El funcionamiento es muy sencillo, utilizando un tren de pulsos o *PWM,* se consigue que la bombilla no esté encendida de forma continua. Variando este tren de pulsos conseguimos variar el tiempo que está encendida la farola. De esta forma se produce una sensación de intensidad variable. Aquí podemos ver unos ejemplos:

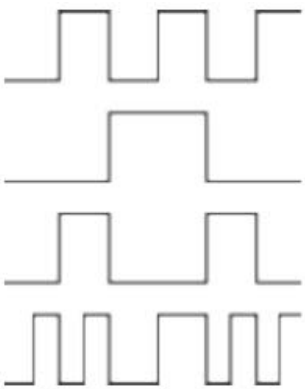

Figura 5. Ejemplos de trenes de pulsos

ROUTER

En este apartado vamos a presentar el dispositivo que hemos empleado como pasarela. Este dispositivo es clave para el correcto funcionamiento de la plataforma, ya que se encarga del acceso a Internet de todas las motas.

Figura 6. Pasarela ALIX 6E1

La máquina utilizada como *Router* es mucho más potente que los dispositivos empotrados que se utilizan como motas y por eso es factible utilizarla para recoger y almacenar información acerca del estado de la red y de todos los dispositivos conectados. Además, aquí podemos implementar una *Business Inteligence* que nos permita decidir qué motas pueden conectarse o quién tiene acceso a los dispositivos a través de Internet.

Sobre el *router* se ha instalado la distribución *Debian GNU/Linux* [7] ya que es un sistema muy estable y escalable. Además el hardware está perfectamente soportado, por lo que no existen problemas de compatibilidad. *Debian* dispone desde hace ya unos años soporte para *IPv6*, por lo que no presenta ningún problema de compatibilidad. Aunque ya se ha comentado que el mundo completo está migrando hacia *IPv6*, no todos los *ISP* y proveedores de servicios han realizado dicha migración. Debian nos permite encapsular y encaminar sobre *IPv4*.

Sólo queda un último detalle que solucionar. El *router* no dispone de un dispositivo de radio *IEEE 802.15.4*, ni están disponibles para *miniPCI* ó como dispositivos USB externos. Esto se ha solucionado utilizando una mota, la cual estará conectada por USB al *router*. De esta forma, se puede utilizar un driver para *Debian* que trabaje con el dispositivo integrado de tal forma que parezca una interfaz de red.

FIREFLY

Y por último vamos a otra parte fundamental del sistema, los dispositivos empotrados que formarán las motas. *FireFly* es un prototipo desarrollado por la empresa *Cilab*, sobre el que hemos basado todo este trabajo.

Este dispositivo dispone de un microprocesador *Jennic JN5148*, el cual lleva un transmisor de radio *IEEE 802.15.4* integrado, una antena de radio, una memoria Flash de 4Mbits y un bus de comunicaciones *MikeBus*. Aquí podemos ver el dispositivo:

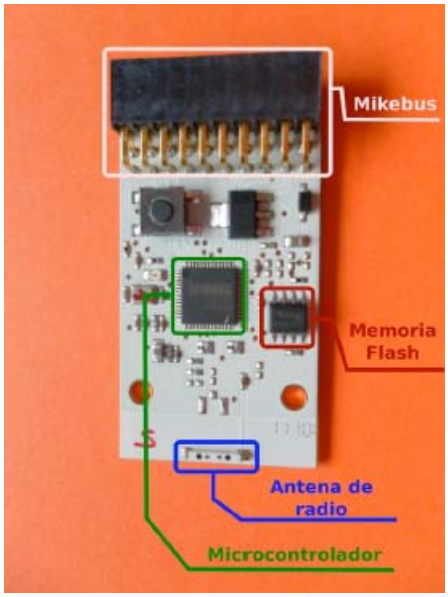

Figura 7. Imagen en detalle del dispositivo *Cilab Firefly*

Contiki no soporta de forma nativa los procesadores de la familia *Jennic*. Por ello ha sido necesario adaptar este sistema operativo a *FireFly*.

AGRADECIMIENTOS

Todo este trabajo lo he desarrollado en *Cilab*, quien ha puesto a nuestra disposición todo el material que hemos necesitado. Nos gustaría agradecer a todo el equipo de *Cilab* la ayuda prestada, ya que ellos han hecho posible este trabajo.

REFERENCIAS

[1] RFC 4944, Base specification. http://tools.ietf.org/html/rfc4944. [Fecha de última consulta: 08-03-2013].

[2] RFC 4919, Problem statement. http://tools.ietf.org/html/rfc4919. [Fecha de última consulta: 08-03-2013].

[3] RFC 3315, DHCPv6 Definition. http://wiki.tools.ietf.org/html/rfc3315. [Fecha

de última consulta: 08-03-2013].

[4] RPL Draft. http://tools.ietf.org/html/draft-ietf-roll-rpl-19. [Fecha de última consulta: 08-03-2013].

[5] Django framework, https://www.djangoproject.com. [Fecha de última consulta: 08-03-2013].

[6] Contiki OS, http://www.contiki-os.org/. [Fecha de última consulta: 08-03-2013].

[7] Debian OS, http://www.debian.org/. [Fecha de ultima consulta: 08-03-2013].