



16 de mayo de 2021

# Instrumentación científica

Wasmote: comunicación con XBee

Alberto Ramos Sánchez

Universidad de Las Palmas de Gran Canaria

Máster Universitario de Sistemas Inteligentes y Aplicaciones Numéricas



## Introducción

En este trabajo se ha implementado una aplicación cliente-servidor para comunicar un dispositivo Wasmote con un PC mediante comunicación radio utilizando dispositivos XBee.

## Implementación

### Paquetes

#### Paquete de inicio de comunicación

Este paquete es enviado al inicio de la comunicación por el servidor para indicar al cliente la hora actual y

Su formato es:

- Identificador de paquete: 0
- Fecha y hora en formato unix.
- Segundos en los que el waspmote se mantendrá en modo *deepSleep*.

Un ejemplo de mensaje podría ser: **0|1620930725|20**

#### Paquete de estado

Este paquete es enviado por el cliente al servidor cuando se produce una interrupción. Su formato es:

- Identificador de paquete: 1
- Fecha y hora en formato unix.
- Ángulo en el eje x del acelerómetro.
- Ángulo en el eje y del acelerómetro.
- Ángulo en el eje z del acelerómetro.
- Porcentaje de batería.
- Causa de la interrupción: 0 es desconocida, 1 es por el acelerómetro y 2 es por el *timer*.

Un ejemplo de mensaje podría ser: **1|1620930902|192|12|22|92.6|1**

### Comunicación

La parte del servidor se encuentra implementada en Python utilizando el módulo *DigiXbee* de Python [1]. El servidor se encarga de descubrir los *XBee* disponibles —cada cierto tiempo indicado por el usuario—, y enviarles un mensaje de inicio de comunicación. Este proceso se lleva a cabo por el hilo inicializado en la función *start\_scanner* de la clase *WaspServer*. El escaneo se realiza mediante la función *start\_discovery\_process* de *DigiXbee* [2].

En la parte del cliente, al arrancar el *Wasmote*, éste espera a recibir un paquete de inicio de comunicación. Cuando lo recibe, establece la hora y el tiempo de *deepSleep*. Además, almacena la dirección de quién envió el mensaje como dirección del servidor. Tras ello, entra en modo *deepSleep*. En este estado, al producirse una interrupción (bien por el acelerómetro o por el *timer*), se envía un mensaje de estado al servidor donde se indica la hora, la orientación, el

porcentaje de batería y la causa de la interrupción. Estos paquetes son recibidos por el servidor a través de otro hilo, inicializado en la función *recv\_message* de la clase *WaspServer*.

## Instalación y uso

Se ha incluido junto al programa del servidor dos ficheros para instalar las dependencias del servidor.

Los módulos de python pueden ser instalados creando un entorno de *conda* con el comando:

```
$ conda env create -f xbee_env.yml
```

También pueden instalarse mediante *pip* con el comando:

```
$ pip install -r requirements.txt
```

Para arrancar el servidor, el programa debe llamarse de la siguiente forma:

```
$ python main.py
```

Opcionalmente podemos indicar otros parámetros:

<code>--help</code>	Mensaje de ayuda.
<code>--scantime [5-60]</code>	Frecuencia en la que el servidor busca dispositivos cercanos. (Por defecto 10 segundos).
<code>--notify_time [5-60]</code>	Frecuencia en la que el cliente indica su estado. (Por defecto 10 segundos).
<code>--usb [USB]</code>	Dispositivo donde se encuentra el XBee. (Por defecto <code>/dev/ttyUSB0</code> ).

El cliente debe arrancarse con el IDE de Waspmote.[3]

## Bibliografía

- [1] «XBee Python Library — Digi XBee Python library 1.4.0 documentation». <https://xbplib.readthedocs.io/en/latest/index.html> (accedido may 13, 2021).
- [2] «Discover the XBee network — Digi XBee Python library 1.4.0 documentation». [https://xbplib.readthedocs.io/en/latest/user\\_doc/discovering\\_the\\_xbee\\_network.html#discover-the-network](https://xbplib.readthedocs.io/en/latest/user_doc/discovering_the_xbee_network.html#discover-the-network) (accedido may 13, 2021).
- [3] «Waspmote IDE». <https://development.libelium.com/waspmote-ide-v06/> (accedido may 14, 2021).