UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
TECNOLOGÍAS DE LA INFORMACIÓN

Desarrollo de Servicio REST API para el uso seguro de dispositivos IoT

Development of an API REST for IoT devices secure access

Realizado por
Javier Cuevas Braun

Tutorizado por
María Mercedes Amor Pinilla

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2019

Fecha defensa: Septiembre, 2019

Fdo. El/la Secretario/a del Tribunal

Resumen

Hoy en día, vivimos en una sociedad conectada, en la que la mayoría de los datos, acciones y decisiones diarias dependen de diferentes dispositivos interconectados entre sí a través de Internet, ya sea en el ámbito personal o industrial, dando lugar a la Internet de las Cosas, o Internet of Things en inglés. Este concepto viene de la mano del concepto de la Industria 4.0 (Industry 4.0) y las "Smart cities", que dan un paso más allá introduciendo los sistemas ciberfísicos en la industria y las ciudades, basándose en este concepto de interconexión a través de Internet.
La idea principal es que Internet llegue a todo tipo de objetos del día a día, no sólo dispositivos móviles o electrónicos, con el fin de controlar y monitorizar todos los recursos que una empresa tiene a través de Internet, con las ventajas que ello conlleva. Pero también puede poner en compromiso la integridad de la empresa si no se implementa la comunicación con los dispositivos de forma segura. Al ser un conjunto de dispositivos independientes, para que desarrollen su funcionalidad correctamente, se presenta un ecosistema muy complejo repleto de vulnerabilidades, por lo que proteger el software se convierte en un deber.
El objetivo principal del TFG es el desarrollo de un sistema que proporcione un servicio de video vigilancia de la IoT, que sea capaz de controlar y monitorizar una cámara sensor conectada, y por supuesto de forma segura, teniendo en cuenta la importancia de la seguridad en sistemas del Internet de las Cosas.
El desarrollo del sistema aborda tanto el montaje hardware de los diferentes componentes en dispositivos propios de la IoT, como el Desarrollo del software de comunicaciones entre la cámara y el servicio que proporciona acceso securizado Web (a través de una API Rest) a las imágenes/video de vigilancia.

Palabras clave:
IoT, Smart City, REST API, Sensor Cámara, Arduino, Raspberry Pi

Abstract

Nowadays, we are living in a connected society in which majority of the data, actions and daily decisions directly depend on different devices interconnected through the Internet, at industrial or personal scope, giving room to the Internet of Things. This idea comes from the Industry 4.0 and Smart Cities concepts that go further introducing the cyber-physical systems within the Industry and the cities, which are at the same time are based on the Internet of Things.

The main idea is to reach through the Internet any kind of day-to-day objects, not just mobile or electronic devices, to control and monitor all company´s assets through Internet, acquiring several advantages. But, if the communication between devices is not implemented in a secure way company's integrity could be at risk. As it is a set of independent devices, being communication crucial to develop their functionality correctly. The system becomes a very complex ecosystem full of vulnerabilities, so protecting the software becomes a must.
The TFG's main goal is to develop an IoT video security service capable of controlling and monitoring a connected camera/sensor, obviously in a secure way, keeping in mind that security has relevance when talking about IoT systems.
The system development addresses assembling all the different IoT hardware devices, as the development of communication software between the camera and the service that provides secure Web access (through an API Rest) to the images / video surveillance. In addition to sending and accessing the data, the remote configuration of the camera and associated sensors (developed in C) will be facilitated.

Keywords:
IoT, Smart City, REST API, Camera Sensor, Arduino, Raspberry Pi

Índice

Resumen **2**

Abstract **4**

Índice **1**

Introducción **1**
1.1 Motivación **1**
1.2 Objetivos **2**

Tecnologías usadas **5**
2.1 Hardware **5**
2.2 Software **10**

Diseño **19**
3.1 Diseño Hardware **21**
3.1.1 Videocámara sensor 21
3.1.1 Servidor IoT 25
3.2 Diseño Software **25**
3.2.1 Videocámara sensor 25
3.2.2 Diseño Servicio IoT 30

Implementación **39**
4.1 Primera iteración **40**
4.2 Segunda iteración **44**
4.3 Tercera iteración **45**

Conclusiones **47**

Bibliografía **51**

Manual de Instalación **53**
Requerimientos: **53**
Instalación Videocámara: **53**
Instalación del Servidor: **58**

Manual de Usuario **67**
Manual de Usuario: Videocámara **67**
Manual de Usuario: Servidor IoT **70**

1

Introducción

1.1 Motivación

Internet de las Cosas, o Internet of Things en inglés, es un concepto que está revolucionando la Industria hoy en día. Se refiere a la interconexión de objetos y dispositivos a través de Internet. Como dispositivos/activos, en la industria hablamos desde un pequeño sensor en la planta de producción hasta un empleado realizando un reparto cotidiano. Toda esta conexión de dispositivos se realiza con el fin de ser capaz de controlar y monitorizar todos los recursos que una empresa tiene a través de Internet, con las ventajas que tiene: acceso remoto, monitorización en tiempo real de toda la información del sistema, capacidad de automatizar tareas, etc... Pero también puede poner en compromiso la integridad de la empresa si no se implementa la comunicación con los dispositivos de forma segura.

Este nuevo concepto viene de la mano del concepto de la Industria 4.0 (Industry 4.0), que da un paso más allá introduciendo los sistemas ciberfísicos, ya que se basa en el concepto IoT.

La Internet de las Cosas no solo está relacionada con la Industria, si no que ya se empieza a implementar para uso doméstico sobre todo en domótica, aunque sigue siendo en la Industria en donde se hace la gran inversión y de dónde se le permite seguir evolucionando y creciendo, ya que es en este sector en el que se saca más partido a la cantidad de información que es capaz de recoger sobre los recursos de la empresa permitiendo así aprender de errores y rápidamente corregir fallos que antes se tardaban días en identificar.

Otro de los factores que hacen la Internet de las Cosas tan interesante es su versatilidad, ya que se basa en comunicación web, pudiendo utilizar cualquiera de los protocolos existentes o inventar uno nuevo. Grandes empresas, como Sigfox, se dedican a innovar nuevos protocolos de comunicación web destinados a sistemas IoT, en este caso creando la red 0G, una red del Internet de las Cosas de baja frecuencia, ya adoptada a gran escala, que promete convertirse en el nuevo estándar IoT.

Por tanto, al ser un tema muy innovador y latente en la Industria de hoy en día, se ha decidido realizar como proyecto un sistema IoT que incluye un servicio web REST API para interactuar de forma segura con dispositivos IoT, que en este caso será una sola cámara sensor. Para simular un caso real de IoT, el dispositivo (cámara) es remoto pero accesible desde un Servidor IoT, o IoT Gateway, que se encarga de la autenticación de los usuarios, para asegurar los dispositivos de la empresa, y de monitorizar y controlar los dispositivos directamente.

1.2 Objetivos

El objetivo principal del proyecto es la realización de un Servidor IoT para ser capaz de controlar y monitorizar una cámara sensor, y por supuesto de forma

segura. Para ello, podríamos distinguir los siguientes objetivos secundarios, que son necesarios de alcanzar para poder alcanzar también el principal:

- Montaje sobre una placa Arduino/Waspmote de una cámara con un sensor de presencia. Esta placa o mota, no es más que un dispositivo basado en Arduino, manufacturado por Libelium, que le añade pines y funcionalidades extra. Consta de la placa base o Waspmote, que sería el equivalente a una placa Arduino normal y que tiene una tarjeta microSD como almacenamiento, sobre la que después va montado un módulo WiFi para las comunicación con el Servidor, un módulo 3G+GPRS que controla el módulo de la cámara sensor y guarda las fotos/vídeos en una tarjeta microSD propia, y finalmente el módulo cámara sensor que consta de la cámara en sí, un sensor PIR de presencia, un sensor de luminosidad, un sensor de luminosidad IR y 2 bloques de LEDs para iluminar las fotos o vídeos cuando la luminosidad no es buena.
- Desarrollo del software para la configuración del comportamiento de la placa con sensores. Incluido en el código que controla el normal funcionamiento de la placa en sí, ya que en sí es un dispositivo Arduino.
- Instalación y despliegue de un servidor Web sobre Raspbian en una Raspberry Pi. Este servidor ofrecerá una API para acceder a los dispositivos IoT a través de una página web para operaciones básicas que realizarán usuarios no expertos, y para usuarios más expertos que realizarán operaciones más complejas a través de programación o herramientas como POSTMAN.

- Desarrollo del software de comunicaciones entre la mota y el servidor. La mota enviará datos constantemente al Servidor, pero además podrá recibir datos de configuración del mismo Servidor.
- Desarrollo de una API REST en el Servidor Linux de la Raspberry para acceder a la configuración y datos proporcionados por la cámara. Esta API facilitará el acceso seguro, con autenticación de usuario y sistema de tokens, a los datos de la cámara y/o cambiar su configuración.
- Desarrollo en C del software para la placa Arduino/waspmote con sensores (cámara). Es aquí donde se incluye la configuración de la misma mota/placa y parte de la comunicación con el Servidor. Toda la configuración se encuentra en la función setup() del código, mientras que el funcionamiento de la cámara sensores se encuentra dentro de loop() al igual que parte de configuración intermedia.
- Desarrollar un cliente Web que acceda a los datos/configuración de sensores a través del servidor Web localizado en la Raspberry. Para autenticarse y obtener el token para acceder el resto de APIs. La interfaz de usuario se ha desarrollado mayormente con HTML, CSS y JavaScript (jQuery).

De esta forma, tendremos una interacción Web Dispositivo/Usuario que parecerá directa, pero habrá un Servidor en medio controlando que se acceda de forma segura a los datos y configuración del dispositivo.

2

Tecnologías usadas

2.1 Hardware

A continuación se describen las tecnologías hardware utilizadas para llevar a cabo el proyecto:

Libelium Waspmote v12.

Libelium es una empresa española que diseña y fabrica hardware, con el SDK correspondiente para programar sobre ese hardware, para redes de sensores inalámbricos de Internet de las Cosas, o IoT. Todo el hardware que producen está directamente relacionado con hardware tipo Arduino.

La waspmote es una placa con microcontrolador basada en Arduino, pero que incluye más conexiones destinadas a la interconexión de dispositivos IoT. Como partes importantes incluye un microcontrolador ATmega1281 de 14.7456 Mhz, ranura para tarjeta microSD de hasta 2GB, conector para batería de 3.3-4.4 V, conexión micro USB para cargar programas en la microSD y para cargar la

batería, 7 pines analógicos, 8 pines digitales, 2 sockets UART, 1 socket I2C, sensor de temperatura y acelerómetro. Véase Figura 2.1 y Figura 2.2.

Todas las imágenes a continuación utilizadas para mostrar el Hardware utilizado han sido extraídas de la documentación de Libelium, que se puede encontrar en el siguiente enlace[1].

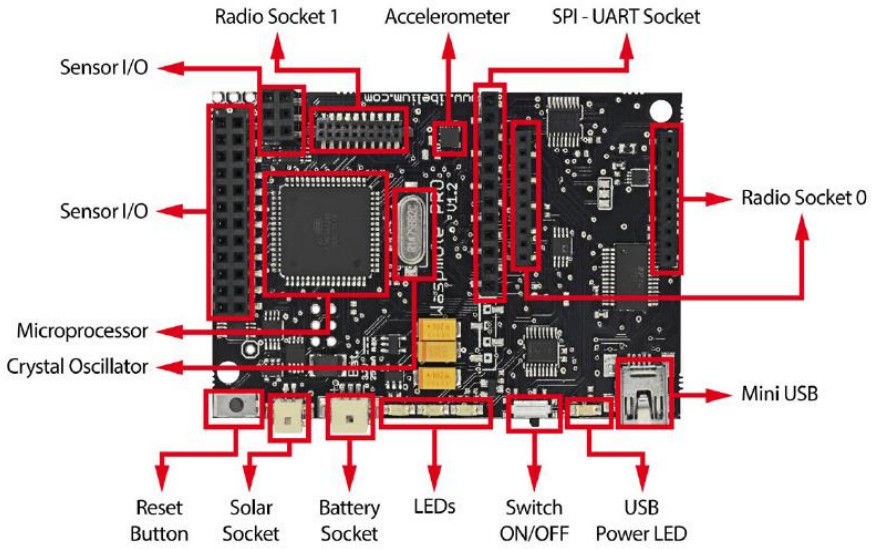

Figura 2.1. Libelium Waspmote versión 12, vista desde arriba.

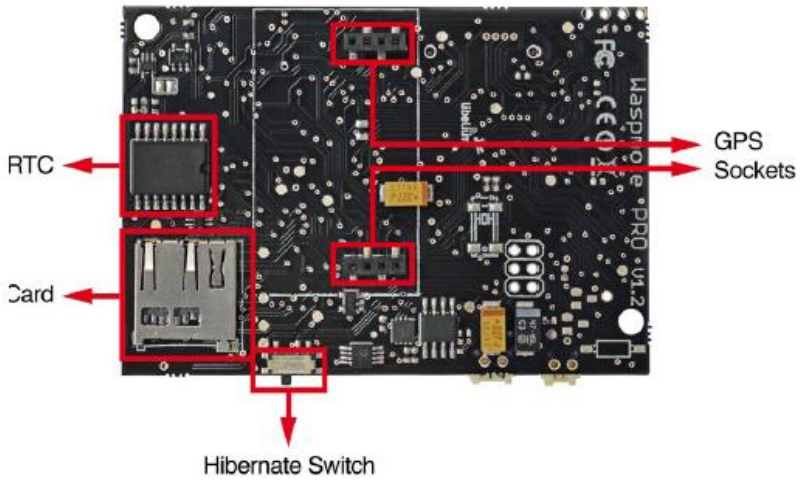

Figura 2.2. Libelium Waspmote versión 12, vista desde abajo.

[1] http://www.libelium.com/development/waspmote/documentation/waspmote-datasheet-v12/

- Batería recargable de 6600 mAh. La batería se recarga cuando estando conectada a la mota, ésta está conectada a la corriente eléctrica a través del cable micro USB.

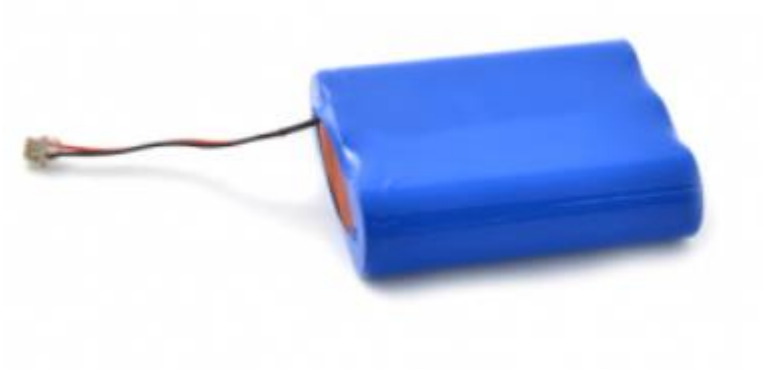

Figura 2.3. Batería recargable de 6600mAh, producida por Cooking Hacks.

- Módulo WiFi rovin networks, para añadir comunicación WiFi a la Waspmote. Incorpora el microcontrolador para la comunicación WiFi y antena WiFi.

Figura 2.4. Módulo WiFi rovin networks RN-VX rev3.

- Módulo 3G+GPRS con ranura microSD, añade comunicación 3G a la Waspmote, pero la función por la que se requiere en este sistema es para controlar la cámara sensor y para servir de almacenamiento interno de la misma. Primero se toma la foto o vídeo para almacenarse en la tarjeta SD del módulo 3G+GPRS, como la foto/vídeo será transmitida por WiFi el archivo es pasado a la tarjeta SD de la Waspmote y de ahí es transmitida a través del módulo WiFi.

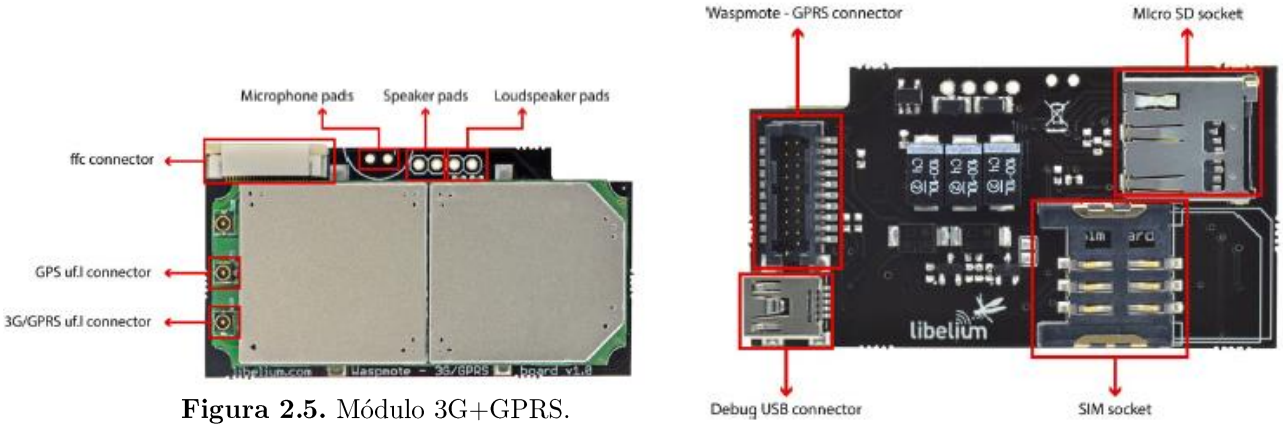

Figura 2.5. Módulo 3G+GPRS.

- Módulo cámara sensor, incluye la cámara, leds para iluminación y pines para conexión con la Waspmote. También tiene 3 sockets para conectar sensores para el correcto funcionamiento de la cámara.

Figura 2.6. Videocámara, con todos los sensores necesarios conectados.

- Sensor PIR, permite a la videocámara generar una interrupción a nivel hardware cuando presencia es detectada.

Figura 2.7. Sensor PIR.

- IR sensor, permite detectar la cantidad de luz IR para determinar si habilitar el filtro IR de la videocámara o no.

Figura 2.8. Sensor IR.

- Sensor de luminosidad, permite medir la cantidad de luminosidad para así determinar si encender los leds de la videocámara y que conjunto de ellos.

Figura 2.9. Sensor de luminosidad

- Raspberry Pi 3 B+, que actúa como servidor del Internet de las Cosas, ofreciendo servicio para controlar la videocámara, y con la posibilidad de incorporar y controlar más dispositivos remotos.

  Características técnicas del micro-ordenador Raspberry Pi 3 B+:

  - CPU + GPU: Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz.
  - RAM: 1GB LPDDR2 SDRAM.
  - Wi-Fi + Bluetooth: 2.4GHz y 5GHz IEEE 802.11.b/g/n/ac, Bluetooth 4.2, BLE.
  - Ethernet: Gigabit Ethernet sobre USB 2.0 (300 Mbps).
  - GPIO de 40 pines.
  - HDMI.
  - Puertos USB 2.0
  - Puerto CSI para conectar una cámara.
  - Puerto DSI para conectar una pantalla táctil.
  - Salida de audio estéreo y vídeo compuesto.
  - Micro-SD.
  - Power-over-Ethernet (PoE).

De entre ellas cabe destacar el módulo Wi-Fi integrado y el almacenamiento de tarjeta microSD, ya que reducen drásticamente el costo de montar un simple servidor para manejar varios dispositivos remotos, pero ofreciendo un buen rendimiento si se usa el software adecuado.

Junto a todos esas placas y sensores, evidentemente ha sido necesario el uso del cableado respectivo para alimentar los dispositivos aparte de tres tarjetas micro SD utilizadas como almacenamiento de la Waspmote, módulo 3G+GPRS y Raspberry pi respectivamente.

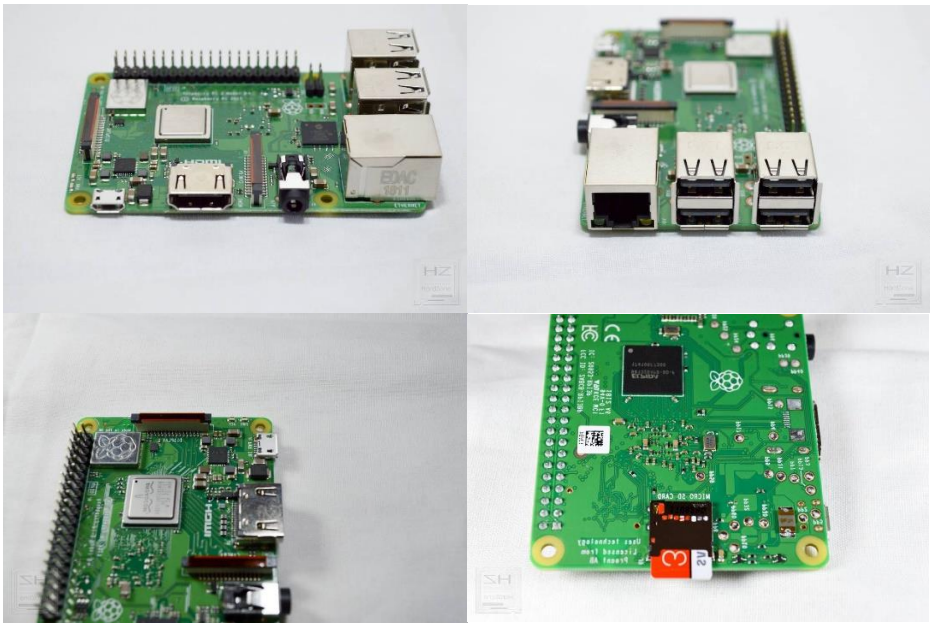

Figura 2.10. Raspberry Pi 3 modelo B+, Imágenes de las diferentes conexiones, con tarjeta microSD insertada. Fotos extraídas de la página oficial de Raspberry Pi Foundation.

En la sección 1 del Capítulo 3 se muestra detalladamente como ensamblar todo el hardware que ha sido utilizado durante el desarrollo del proyecto.

2.2 Software

Para llevar a cabo el proyecto se han utilizado las siguientes tecnologías y herramientas software y lenguajes de programación utilizados para desarrollar el código del sistema:

Waspmote

El programa que controla la Waspmote con la videocámara está desarrollado en C, haciendo uso de la extensa librería de Waspmote para controlar los diferentes sensores y módulos que entran en juego para este caso de uso. Ficheros de la librería proporcionados por Waspmote han sido modificados para hacer funcionar la comunicación FTP con el servidor ya que, en cierto modo, la Waspmote utilizada se ha quedado un poco obsoleta al ser una versión antigua.

Waspmote pro IDE v4, es la herramienta utilizada para cargar el software en la mota y administrar correctamente las librerías. También ha sido usado como herramienta para depurar el código de la Videocámara sensor, ya que proporciona una consola donde se muestran los mensajes enviados desde la mota.

Servidor/Gateway IoT

El servidor del Internet de las Cosas ha sido montado en una Raspberry Pi 3 B+, que simplemente es un ordenador de bajo coste del tamaño de una tarjeta de crédito, pero que es capaz de hacer cualquier tarea que un ordenador es capaz de.

El sistema operativo elegido para el servidor es Raspbian, la versión Debian desarrollada por Raspberry Pi Foundation para optimizar la arquitectura hardware de la Raspberry Pi y reducir el espacio de disco que normalmente ocuparía Debian en un ordenador normal. La versión escogida es Raspbian Stretch Lite, versión sin Interfaz de Usuario y con los mínimos paquetes instalados para poder funcionar como servidor e interactuar con él, a través de SSH, con la línea de comando (CLI).

Servicios instalados en el servidor:

- **pureFTP** ha sido instalado como servidor FTP en la Raspberry Pi para recibir los datos de los dispositivos IoT en el lado del servidor. Es el servidor FTP recomendado por libelium.

En el Capítulo 4.2, explicación detallada de porqué se ha elegido FTP como protocolo de comunicación para mandar ficheros desde la Videocámara al servidor.

- **Mongo DB**, es la base de datos que se ha escogido para almacenar toda la información relevante del sistema, como puede ser los Usuarios de la aplicación para controlar los dispositivos del Internet de las Cosas, guardar las rutas de los ficheros generados por la videocámara sensor con un *timestamp*, controlar los ficheros de configuración con versiones, manejar los tokens de los usuarios para el acceso a la API, y para controlar también las rutas de los ficheros de configuración generados por los usuarios.

  MongoDB es un sistema de base de datos NoSQL orientado a documentos de código abierto, qué, en lugar de guardar los datos en tablas, tal y como se hace en las bases de datos relacionales, MongoDB guarda estructuras de datos BSON (una especificación similar a JSON) con un esquema dinámico, haciendo que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

  Con la introducción del concepto IoT, algunos prevén que para 2020 unos 50.000 millones de dispositivos estarán interconectados a través de Internet. Evidentemente, las Bases de Datos relacionales no fueron diseñadas para el volumen y variedad de datos que estos dispositivos manejan, además a una gran velocidad. Por tanto, las bases de datos no relacionales juegan un papel muy importante en los sistemas IoT, ya que proporcionan de la velocidad y eficiencia necesaria que estos sistemas requieren. De ahí, que MongoDB sea la Base de Datos elegida, ya que es la Base de Datos no relacional más adoptada hoy en día.
- **Node.js.** Para la creación y despliegue de la REST API se ha escogido Node.js, ya que es JavaScript para código en el lado del servidor, ofreciendo

la posibilidad de crear aplicaciones web con la capacidad de respuesta en tiempo real eliminando de la ecuación los web sockets. Con Node.js, ambas partes pueden iniciar la comunicación, Cliente o Servidor. La idea general de Node.js: utilizar E/S sin bloqueo y dirigida por eventos para mantener así ligero y eficiente frente a aplicaciones de respuesta en tiempo real que hacen un uso intensivo de gran cantidad de datos provenientes de dispositivos distribuidos remotamente. Esto quiere decir, que Node.js cuando realmente es útil es para construir rápidamente aplicaciones en red escalables y que tengan la habilidad de manejar un gran número de conexiones simultáneas. Pero, por ejemplo, si nuestra aplicación requiriese de la realización de operaciones intensivas de CPU, Node.js no sería la elección correcta ya que este caso de uso anularía todas las ventajas que Node.js nos ofrece.

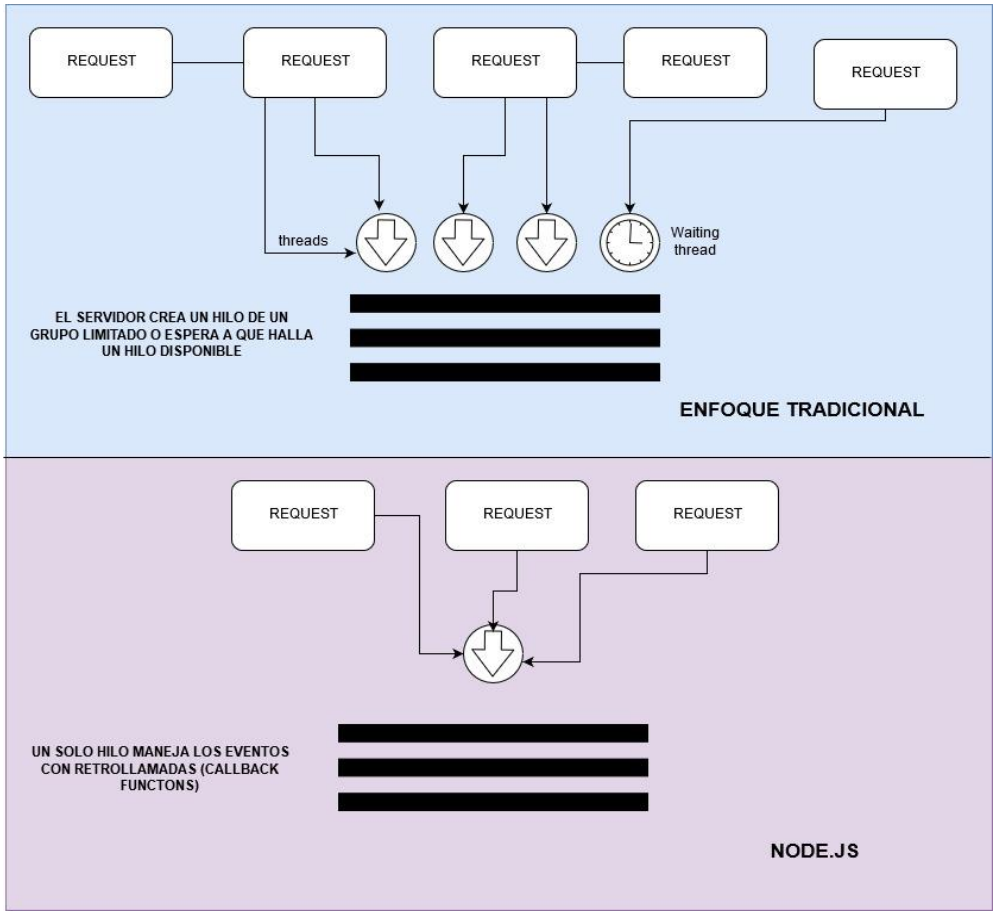

Figura 2.11 Node.js manejo de peticiones comparado con el manejo de peticiones tradicional (elaborada propia).

Otra de las grandes ventajas por las que se has escogido Node.js es el rápido crecimiento que está experimentando, ya que al ser un *framework open source* o de código libre, cada día aparecen nuevos paquetes y funcionalidades que la comunidad ha ido añadiendo a Node.js, pudiendo obtener así soporte rápido y fácilmente. Obsérvese Tabla 1.1 para obtener información más detallada de los paquetes instalados en Node.js.

Paquete	Nombre	Versión	Funcionalidad
npm	Node Package Manager	1.4.21	Este paquete de Node.js nos permite controlar todos los demás paquetes instalados y sus versiones.
nodemon	Nodemon	1.18.6	Es una herramienta que ayuda a desarrollar aplicaciones Node.js reiniciando la aplicación cada vez que un fichero cambia en el directorio raíz de la app.
bcrypt-nodejs	Bcrypt	0.0.3	Implementación de bcrypt para Node.js.
body-parser	Body parser	1.18.3	Node.js middleware para decodificación del cuerpo de las peticiones.
chokidar	Chokidar	3.0.1	Utilidad de línea de comandos rápida para el control de cambios en los ficheros.
cors	Cors	2.8.5	Paquete que provee del middleware necesario al paquete Express para habilitar diferentes funcionalidades multiplataforma.
express	Express	4.16.4	Framework web rápido, minimalista y sin opciones por defecto para Node.js.
express-session	Express-session	1.15.6	Middleware necesario para crear una sesión web cuando express es utilizado para el manejo web.

fs	File System	0.0.2	Para interactuar con el sistema de ficheros utilizando funciones POSIX estándar.
http	Hyper Text Transfer Protocol	0.0.0	Es un módulo que viene ya integrado con la instalación de Node.js, que simplemente permite transmitir datos a través del protocolo HTTP.
jQuery	jQuery	3.4.1	Es una librería JS muy rápida, pequeña y con múltiples funcionalidades para manipular documentos HTML, por lo tanto, es ejecutado en el lado del Cliente. Esta es la librería que incluye Ajax para hacer las llamadas a la REST API desde la interfaz de usuario del navegador.
jwt-simple	JSON Web Token	0.5.5	Módulo utilizado para codificar y decodificar, normalmente destinado al manejo de tokens.
moment	Moment	2.22.2	Funciona para ambos, Cliente y Servidor. Librería ligera para validar, traducir y manipular fechas.
mongoose	Mongoose	4.7.3	Mongoose proporciona una solución sencilla basada en esquemas para modelar los datos de su aplicación. Incluye casting tipográfico integrado, validación, construcción de consultas, y cualquier operación de interacción con base de datos.
path	path	0.12.7	Provee las funciones necesarias para trabajar con rutas de fichero y directorio.

Tabla 2.1 Paquetes Node.js instalados en el servidor (elaborada por mí).

Transferencia de datos: API REST

Una API, es la definición de un conjunto de reglas para que las aplicaciones se comuniquen entre ellas, es decir, son el mecanismo más útil para conectar distintos softwares entre sí.

Una API REST, es un protocolo Cliente/Servidor sin estado, ya que se basa en peticiones HTTP. Los objetos REST son manipulados con URIs y siguiendo la especificación HTTP, lo que forma una interfaz uniforme: POST para crear, GET para leer y consultar, PUT para editar y DELETE para borrar.

Al estar construyendo una API REST con JavaScript, el lenguaje utilizado para la comunicación entre el Servidor y el Cliente es JSON. JSON, o JavaScript Object Notation (Notación de Objectos JavaScript), establece un formato simplificado para estructurar datos de forma legible, y se utiliza principalmente para transmitir datos entre un servidor y una aplicación web, surgiendo como una alternativa a XML, eXtensible Markup Language en inglés.
Con JSON, fácilmente podemos recuperar los datos de las peticiones en el lado del Servidor para luego mandar una respuesta JSON al lado de Cliente, donde también muy fácilmente se puede obtener los datos de la respuesta para luego estructurarlos en un documento HTML, por ejemplo, para poder mostrarlos en una interfaz de usuario web.

Otras tecnologías utilizadas:

Github, ha sido utilizado para mantener una copia segura y consistente del código. También, para poder programar localmente en mi ordenador y con una simple combinación de comandos obtener el código en el Servidor o Raspberry Pi, ya que programar directamente desde la línea de comandos en Linux, ya sea con vi o nano, se hace bastante tedioso.

Cabe destacar que cuando sobrecargando la Raspberry Pi teniendo muchos terminales abiertos con distintos programas corriendo a la vez, ésta resetea automáticamente la conexión SSH corrompiendo muchos ficheros y programas, por lo que el uso de git como repositorio remoto ha sido de gran ayuda para siempre mantener una versión estable del código de la aplicación.

Notepad ++, a la hora de programar en local, Notepad ++ ha sido el editor elegido al estar bastante familiarizado con él. Al ser todo el código del lado del Servidor en JavaScript, un lenguaje que pretende ser ligero y rápido, tampoco hubiera sido de gran ayuda utilizar algún IDE más completo como Eclipse o Netbeans que ofrecen soporte extra para los *frameworks* y librerías de lenguajes más complejos como C++ o Java.

POSTMAN es la herramienta software recomendada para hacer peticiones a la API REST, a parte de las páginas web desarrolladas para las principales funcionalidades del sistema. POSTMAN ofrece una interfaz de usuario elegante con la que hacer peticiones HTTP sin tener que molestarse en escribir un montón de código para hacer peticiones HTTP simples que no requieren de una interfaz de usuario más parecida a una página web.

PuTTY ha sido la herramienta escogida para hacer la conexión SSH con el Servidor con el fin del desarrollo del software y la cambios en la configuración del servidor.

3

Diseño

En este capítulo se describe el diseño del sistema desarrollado. En la Figura 3.1 (a continuación) se muestra un diagrama explicativo que relaciona los diferentes componentes hardware y software que integran el sistema. Los componentes hardware se encuentran integrados en los dispositivos IoT, mientras que los componentes software se incluyen asociados a los dispositivos en los que se ejecutan (cómo el servidor IoT).

En el Apéndice B se incluye el Manual de Usuario, tanto del dispositivo o Videocámara sensor, y la interfaz del servicio web IoT.

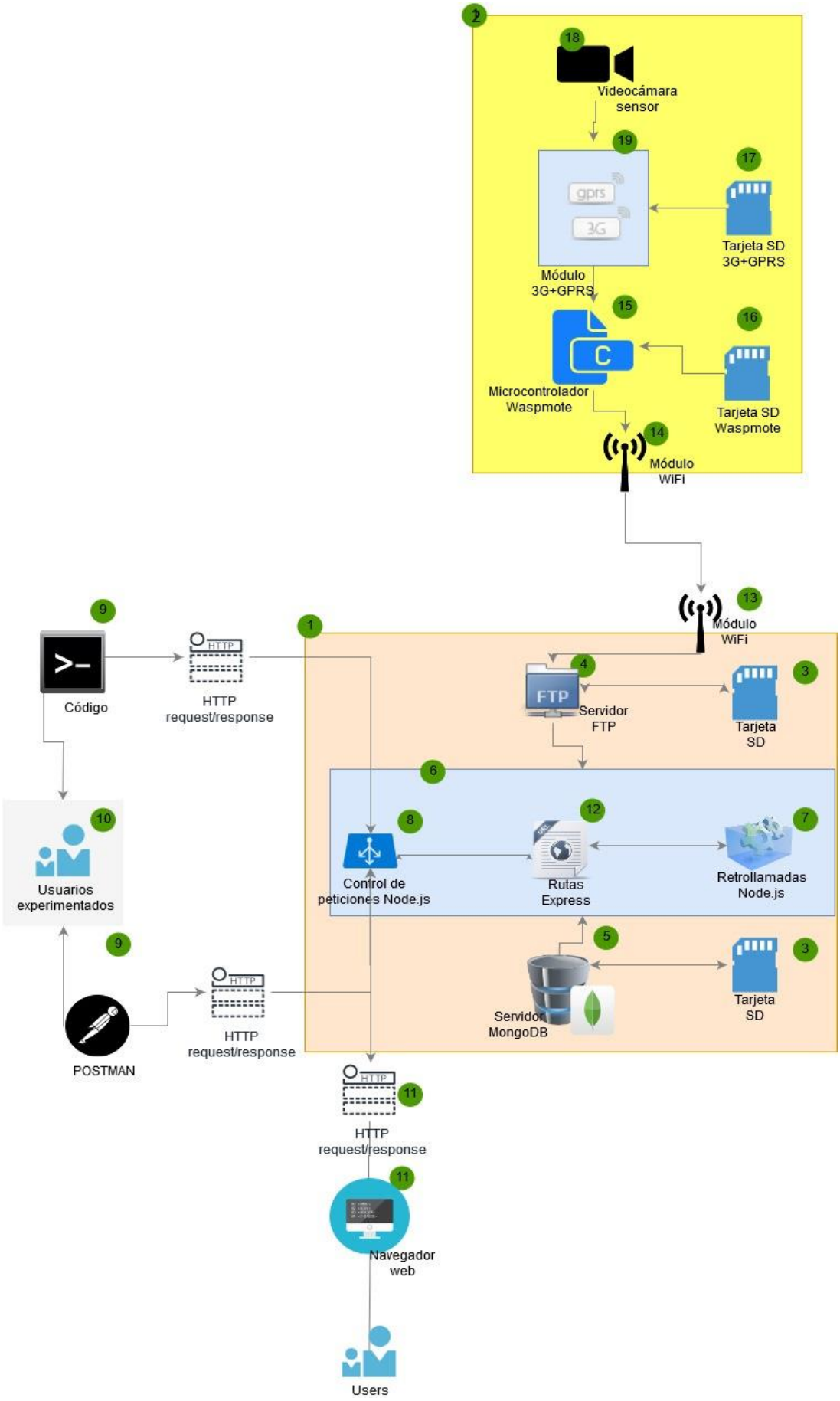

Figura 3.1. Diagrama explicativo del diseño hardware y software del sistema completo, dispositivo IoT y servidor (elaborada por mí).

3.1 Diseño Hardware

Entre el Hardware utilizado para llevar a cabo el proyecto podemos diferenciar dos partes, la que se identificaría como dispositivo del Internet de las Cosas y el servidor o portal del Internet de las Cosas, *Internet of Things gateway* en inglés. A continuación se describe el diseño y ensamblaje hardware.

3.1.1 Videocámara sensor

La videocámara sensor, o dispositivo IoT, consta de múltiples de los sensores y módulos, mencionados con anterioridad en el Capítulo 2, que interconectándolos de la forma que ahora se va a mostrar, forman una videocámara sensor que puede ser utilizada como dispositivo de seguridad.

A continuación, una breve explicación paso a paso de como interconectar esos dispositivos. En el diagrama de la Figura 3.1, todos estos componentes conforman el dispositivo marcado con el número 2. Todas las Figuras del Capítulo 3.1.1 han sido sacadas del manual de usuario de Libelium Waspmote, que se puede encontrar en el siguiente enlace[2].

1. Es recomendable empezar conectando la batería a la Waspmote como se indica en la siguiente Figura 3.2.

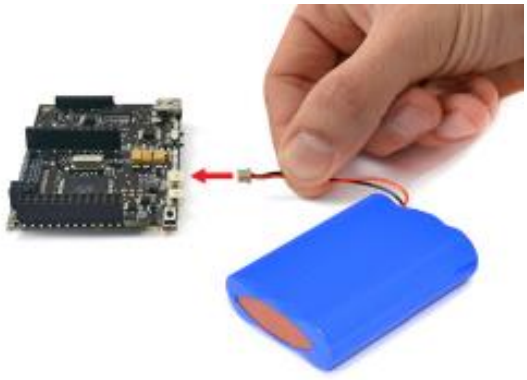

Figura 3.2. Cómo conectar batería a Waspmote.

A continuación, insertar tarjetas SD a módulo 3G+GPRS y Waspmote respectivamente como se indica a continuación:

[2] https://www.libelium.com/downloads/documentation/video_camera_guide.pdf

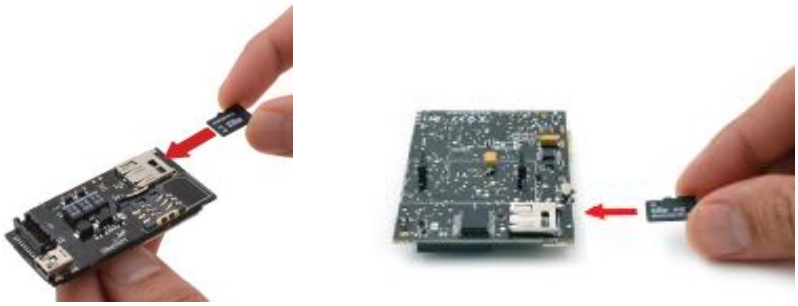

Figura 3.3. Dónde insertar tarjetas SD en módulos 3G+GPRS y Waspmote, de izquierda a derecha respectivamente.

2. Conectar el cable FPC con extrema precaución. Para ello primero hay que empujar los laterales del conector hasta sacarlos hacia afuera del módulo, luego se introduce el conector FPC con la parte metálica hacia fuera, y finalmente se vuelven a empujar los conectores laterales hacia adentro para que el conector quede cerrado y fijado. Habría que repetir el mismo proceso pero para conectar el otro lado del cable FPC al módulo 3G+GPRS. La Figura 3.3 muestra el proceso detallado. El cable FPC, en este caso, ha sido utilizado para transmitir imagen y vídeo de la videocámara al módulo 3G+GPRS.

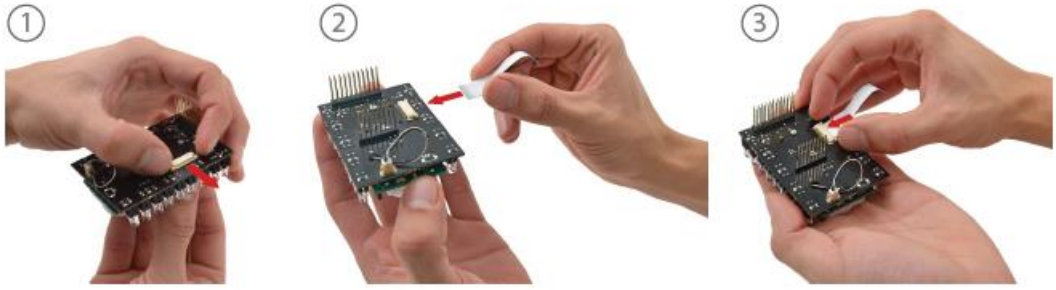

Figura 3.4. Cómo conectar cable FPC, paso a paso.

3. El siguiente paso es conectar el módulo 3G+GPRS al socket 1 de la Waspmote. Véase figura 2.1 para identificar el socket 1 en la Waspmote correctamente.

Figura 3.5. Conectar módulo 3G+GPRS a la Waspmote.

4. Con cuidado de no dañar la conexión realizada anteriormente con el cable FPC, la siguiente conexión a realizar es entre la videocámara sensor y la Waspmote, para ello se tienen que insertar los pines macho de Videocámara sensor en los pines hembra de la Waspmote. Hay que hacer corresponder los pines de la Videocámara con los pines de sensores I/O y SPI/UART de la figura 2.1, y dejando libres los dos pines de la parte izquierda y arriba de los que se correspondería con los pines de sensores I/O. Los pines de SPI/UART en la Waspmote deben de quedar todos ocupados, como se puede ver en la siguiente figura.

Figura 3.6. Conectar Videocámara sensor a la Waspmote

5. El último paso sería conectar los sensores necesarios a la Videocámara sensor, que serían el sensor de presencia PIR, sensor IR y sensor de luminosidad, figuras 2.7, 2.8 y 2.9 respectivamente. Sockets o pines donde conectar los sensores en la placa Videocámara sensor en Figuras 3.6, 3.7 y 3.8.

Figura 3.7. Conexión de sensor de presencia PIR a la placa de la Videocámara.

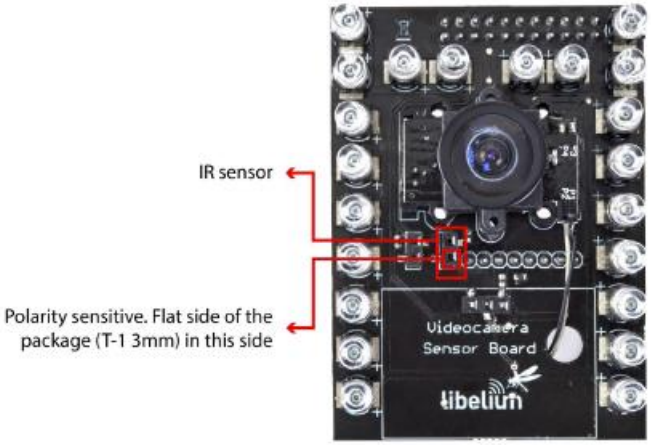

Figura 3.8. Conexión de sensor IR a la placa de la Videocámara. Polaridad del sensor explicada.

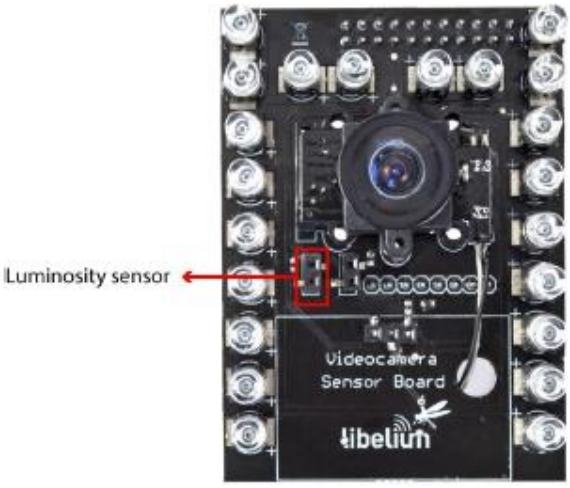

Figura 3.9. Conexión de sensor de luminosidad en la placa de la Videocámara.

3.1.1 Servidor IoT

La parte del servidor se instala y despliega en una tarjeta microSD que sirve como almacenamiento de disco para la placa Raspberry Pi 3 B+ descrita con anterioridad en el capítulo 2 donde se describen las tecnologías hardware usadas. Véase **Figura 2.10**, podrá ver fotos de todo el hardware que ha sido necesario para crear el servidor del Internet de las Cosas.

3.2 Diseño Software

Para describir el diseño del Software, también se pueden distinguir entre dos partes bien diferenciadas, el dispositivo IoT y el servidor. Por eso, primero se hará una descripción detallada de cómo funciona el Software interno del dispositivo, en este caso la Videocámara, explicando cómo se hace la comunicación interna entre los distintos módulos que forman dicho dispositivo. Pero, para la descripción del servidor, no tiene sentido hablar de él solo en sí, ya que depende de los dispositivos con los que se comunica, aunque esto no quiere decir que tiene un diseño interno completamente independiente al de los dispositivos.

3.2.1 Videocámara sensor

El Software en la Videocámara está en escrito en C, ya que es un dispositivo muy similar a un Arduino, y con la ayuda de las librerías del desarrollador de los componentes, Libelium. Gracias a la librería que Libelium proporciona, el Software en el dispositivo se simplifica bastante ya que ha permitido abstraerse de programar la comunicación entre los distintos módulos a nivel de byte. De entre todas las librerías de Libelium, las que se han utilizado para desarrollar el proyecto son:

- **WaspSD.h** y **WaspSD.cpp**, fichero de cabecera y fichero de implementación respectivamente. En estos ficheros se encuentran todas

las funciones necesarias para interactuar con la tarjeta SD, almacenamiento interno, de la Waspmote. No hace falta importarla como el resto de librerías, ya que es un módulo ya importado por defecto por Libelium. Ejemplo de uso:

```
SD.ON();
```

para activar el uso de la tarjeta SD y habilitar todas las funciones de la librería.

Hay más librerías usadas internamente por el paquete WaspSD y demás módulos que son importados por defecto, ya que se consideran del núcleo Libelium, pero que no son usadas explícitamente en el código.

- Otro módulo/paquete importado por defecto es el paquete USB, **WaspUSB.h** y **WaspUSB.cpp**, incluyendo toda la funcionalidad necesaria para la comunicación de la Waspmote con el IDE instalado en el ordenador. Paquete necesario para poder hacer uso del terminal, principalmente para depurar el código y mostrar mensajes por pantalla.

```
USB.ON();
USB.println(F("3G module ready..."));
```

Para activar el paquete USB y para mostrar mensaje en el terminal respectivamente.

- **WaspWIFI.h** y **WaspWIFI.cpp** son los ficheros que incluyen todas las funciones necesarias para realizar operaciones WiFi, para este caso de uso simplemente conectarse a una red WiFi para transmitir paquetes FTP al servidor. Las funciones nos son llamadas explícitamente en el código pero si es necesario importar el paquete o librería ya que es usado por el paquete 3G para transmitir las fotos/vídeos por WiFi. Las llamadas a funciones WiFi son realizadas desde las funciones del paquete 3G porque es el módulo 3G el que almacena las fotos/vídeos realizados y también es el módulo que controla completamente la Videocámara sensor.

```
#include <WaspWIFI.h>
```

Para importar el paquete con las funciones WiFi. Esta línea de código debe de aparecer antes que la línea que importa el paquete con las funciones para controlar el módulo 3G, explicado en el siguiente punto.

- Finalmente, el último paquete utilizado es la librería 3G, necesaria para controlar el módulo 3G+GPRS y la placa Videocámara sensor. Los ficheros involucrados son **Wasp3G.h** y **Wasp3G.cpp**, e incluyen todas las funciones necesarias para realizar operaciones 3G, GPRS y configurar y manejar la Videocámara sensor. Por ejemplo, para comprobar si el módulo 3G+GPRS tiene tarjeta SD insertada y para tomar una foto, respectivamente:

```
_3G.isSD();
_3G.takePicture();
```

Por tanto, el diseño software interno de la Videocámara se reduce a la programación de la comunicación entre los distintos módulos, con simples llamadas a funciones ya implementadas por la librería Libelium, para transferir los datos de un módulo a otro.

En la Figura 3.10, se puede observar detalladamente cada paso necesario en las comunicaciones para el principal caso de uso, tomar una foto con la Videocámara sensor y mandarla por WiFi al servidor a través del protocolo FTP.

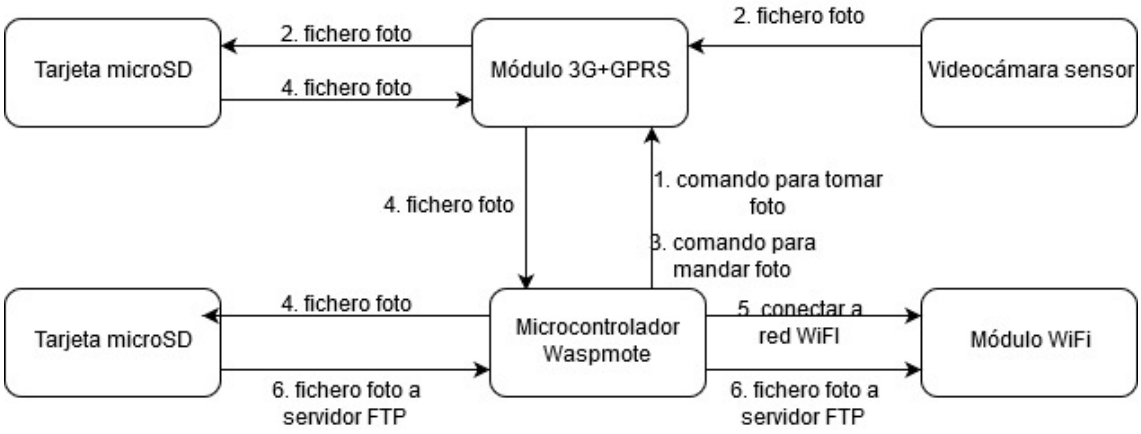

Figura 3.10. Videocámara sensor, diseño software interno (elaborado por mí).

Cada paso en el esquema se correspondería en el código con una llamada a una función de la librería antes mencionada, sin contar con la configuración intermedia entre cada paso y la configuración inicial, que serán explicadas en el capítulo 4. Antes de explicar detalladamente cada paso, cabe destacar que todo el código está almacenado en la tarjeta SD de la Waspmote, ya que es en la Waspmote donde se encuentra el microcontrolador. Explicación más detallada paso a paso, numeración de cada paso a continuación se corresponde a la numeración de la Figura 3.10:

1. El primer paso, después de toda la configuración inicial necesaria y después de que se haya detectado presencia, es realizar una foto o vídeo. Previamente, una interrupción a nivel de Hardware ha sido generada desde la Videocámara sensor cuando el sensor PIR ha detectado presencia, y toda la preconfiguración necesaria ha sido llevada a cabo. Con el siguiente código se habilita la interrupción a nivel PIR y se toma la foto:

```
_3G.enablePIRInterrupt();
_3G.takePicture();
```

2. Una vez la foto es tomada por la Videocámara, automáticamente la foto se almacena en la tarjeta SD del módulo 3G+GPRS. Si no hubiera tarjeta SD en dicho módulo, la foto se almacenaría en la RAM o memoria interna del módulo.

3. La siguiente operación es indicarle al módulo 3G que la foto va a ser mandada por WiFi. Es una sola función del paquete 3G que llama a su vez varias funciones USB y WiFi, que son los siguientes pasos. Es decir, a partir de este paso, todos los siguientes pasos o funciones están contenidos en la función que se llama en este punto:

```
/* sendFiletoWiFiFTP(char*, char*, char*, uint16_t, uint8_t, char*, char*, uint8_t, char*,
char*) - Uploads a file to a FTP using the WiFi connection
 *
 * This function uploads a file to a FTP using the WiFi connection. It does all steps.

 * Return '1' if success, '-2' if error setting the connection options, '-3' if error setting the
DHCP options
 * '-4' if error setting FTP parameters, '-5' if error setting authentification key, '-6' if error
setting the join mode
 * '-7' if error joining to the Wi-Fi network, '-8' if error opening the FTP session and '-9' if
error uploading the file
*/
int8_t Wasp3G::sendFiletoWiFiFTP(char* origin_path, char* destiny_name, char*
FTP_server, uint16_t FTP_port, uint8_t FTP_mode, char* FTP_username, char*
FTP_password, uint8_t auth_type, char* auth_key, char* SSID);
```

4. Para mandar la foto a través de WiFi y no a través de 3G, la foto ha de ser almacenada en la tarjeta SD de la Waspmote. Para ello

   ```
   int8_t Wasp3G::getXModemFile(const char* origin, const char* destiny)
   ```

   es la función que es implícitamente llamada para transferir el fichero de la tarjeta SD del módulo 3G+GPRS a la tarjeta SD de la Waspmote. Primera operación realizada cuando ejecutada función del paso 3.

5. Como el fichero va a ser transferido a través de una conexión WiFi, el siguiente paso es conectar la Waspmote a la red WiFi, evidentemente haciendo uso del paquete WiFi mencionado con anterioridad que ofrece múltiples funciones para la configuración de la conexión WiFi.

6. Una vez la conexión WiFi ha sido establecida con éxito, el fichero es mandado al servidor FTP, configurando primero los parámetros FTP: usuario, contraseña, puerto y modo FTP, activo o pasivo.

Si toda la configuración es correcta y la conexión WiFi ha sido establecida con éxito, la foto o vídeo será transferida por completo, con una latencia de entre 1

minuto y medio, al servidor, específicamente a la carpeta pureFTP en el directorio /home/pi.

Seguridad:

Cabe destacar, que, por razones de seguridad, la conexión a Internet realizada para transmitir los ficheros solo se realiza para la transmisión en sí, una vez se ha transmitido el fichero la conexión se cierra, quedando así el dispositivo aislado de Internet cuando no es necesario que este esté conectado a Internet. Así se evita comprometer la seguridad del sistema del Internet de las Cosas, puesto que los principales puntos de ataque a estos sistemas es a través de los dispositivos que interconecta ya que normalmente suelen ser elementos más vulnerables al tener que estar constantemente en comunicación con el servidor y otros dispositivos a través de Internet.

3.2.2 Diseño Servicio IoT

La figura 3.1 muestra todos los flujos de datos entre las distintas partes software involucradas en el servidor, que en el diagrama es la marcada con el número 1.
Cabe destacar, que todo el software del servidor esta almacenado y corre en la tarjeta SD de la Raspberry Pi, en la Figura 3.1 representada con el elemento número 3.
Para explicar el diseño del Software del Servidor, la mejor manera es hacerlo definiendo el flujo del sistema empezando por cuando el fichero es recibido desde el dispositivo, definiendo cada elemento software que entra en juego en cada momento del proceso.

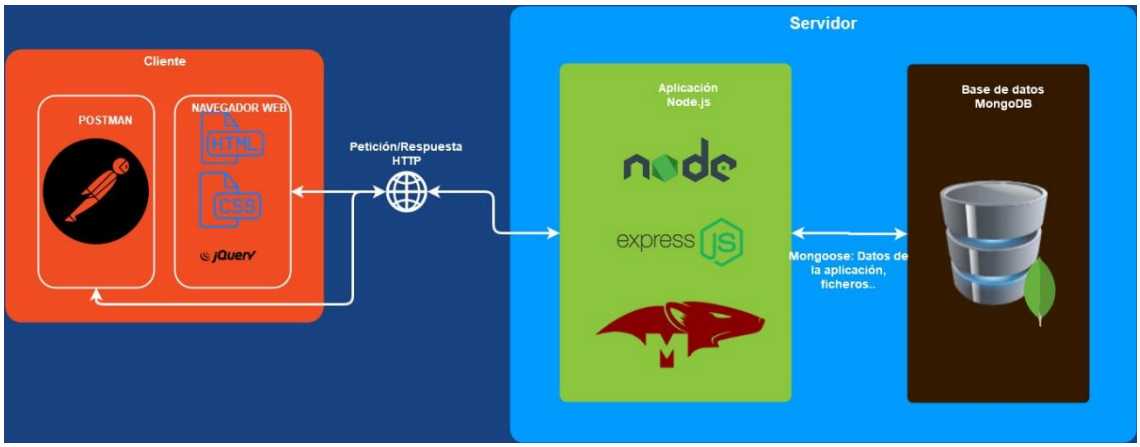

Figura 3.11. Diagrama interno del Servicio web IoT (elaborado por mí).

Si observamos en la Figura 3.11, la parte del Servidor. Esta es la estructura interna a nivel de Software de la caja numerada con un 6 del diagrama de la Figura 3.1.

Recepción de datos desde el dispositivo

El servidor FTP, **pureFTP** en este caso, recibe el fichero en el directorio /home/pi/pureFTP. El papel que juega el servidor FTP junto con la conexión WiFi es crucial para poder establecer la comunicación con la Videocámara. En el diagrama de la Figura 3.1, la conexión WiFi de la Raspberry Pi se representa con el número 13. El servidor pureFTP es indicado en el diagrama con el número 4.

En el Capítulo 4.2, explicación detallada de porqué se ha elegido FTP como protocolo de comunicación para mandar ficheros desde la Videocámara al servidor.

En el diagrama, representado con el número 6, el servidor que expone las rutas de la API REST, el código Node.js. Es la parte más importante del sistema ya que desarrolla el resto de funcionalidades del sistema.

Registro de datos del dispositivo IoT

Registrar en la base de datos MongoDB cada fichero recibido desde el dispositivo. Gracias al paquete instalado chokidar, cada vez que un fichero es guardado o modificado en el directorio del servidor FTP antes mencionado, ese evento disparará una función que guarda en la base de datos el nombre del fichero, el dispositivo del que proviene y la fecha y hora del evento.

Publicación de los datos

Con el paquete Node.js Express, la API es publicada en Internet para poder controlar e interactuar con los dispositivos IoT a través de una serie de rutas URI, por ejemplo /updateDev/:id/:name/:funct, para modificar la definición de uno de los dispositivos en el sistema. Representado en el diagrama de la Figura 3.1 con el número 12.

En el diagrama, número 8, el control de peticiones de Node.js permite recibir peticiones o paquetes HTTP (número 11 en el diagrama), que son tratadas luego por las funciones callback o retro llamadas (número 7 en el diagrama), a la API REST publicada con Express. Esas peticiones HTTP pueden ser realizadas, como se observa en el diagrama, de distintas formas dependiendo de la experiencia del usuario que las realice. Si el usuario es experimentado (número 19 en el diagrama) y posee las nociones básicas de servicios web, conociendo a fondo como HTTP funciona, podrá hacer uso de todas las funciones publicadas por la API codificando dichas peticiones, ya sea con POSTMAN o otra herramienta similar, directamente con código en cualquier lenguaje (número 9 en el diagrama). Si el usuario no es experimentado con este tipo de sistemas del todo, se ha desarrollado una interfaz de usuario básica, a la que se puede acceder con cualquier navegador, con la que se puede interactuar con el Servidor para hacer uso de

algunas de las funciones publicadas por la API REST, véase figura 3.12 a continuación.

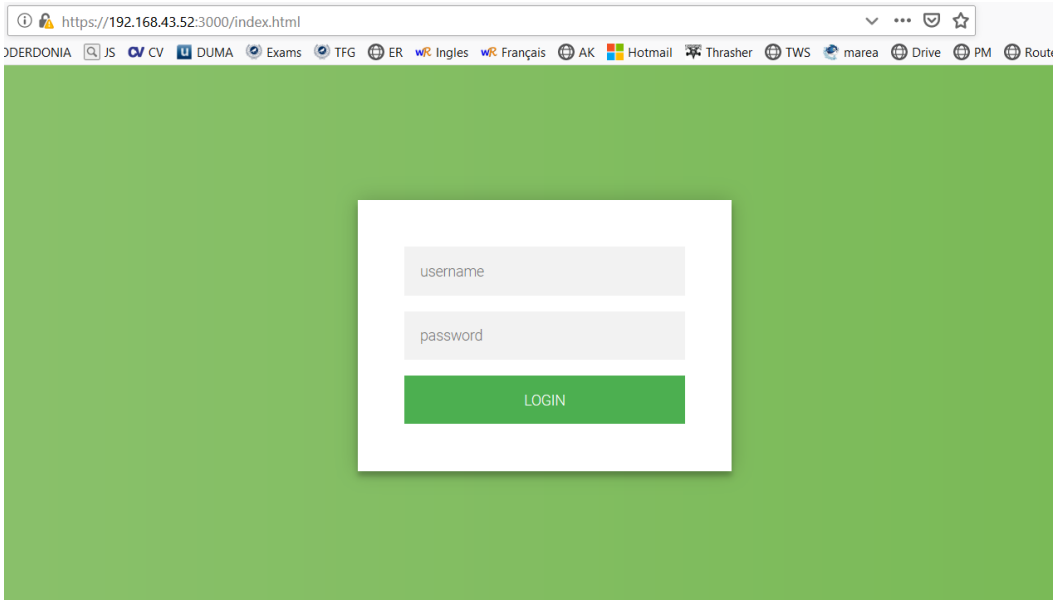

Figura 3.12. Interfaz de usuario para iniciar sesión en la aplicación y obtener el token con el que se accede a la API.

Base de Datos

La base de datos MongoDB, número 5 de la Figura 3.1, guarda toda la información relevante del sistema: usuarios, dispositivos IoT registrados, ficheros de configuración de dispositivos, tokens de cada usuario y los ficheros recibidos de la videocámara sensor.

En el Capítulo 2.2, explicación detallada de MongoDB y el porqué de su elección.

Estructura del Código

El código de la aplicación se ha estructurado de la forma más modular posible, permitiendo así la escalabilidad del código en el futuro. En el diagrama de la Figura 3.11, todo el código se encuentra en la parte que se indica como aplicación Node.js.

Estructura del código:

- En el directorio **credentials**, se guardan el certificado y la clave necesarias para poder utilizar HTTPS, HTTP con SSL.

- Carpeta middlewares contiene todo el middleware del sistema, en este caso el único middleware es el de autenticación. Para la implementación del control de autenticación de usuarios se ha utilizado un Secreto, una clave RSA de 256 bits almacenada como variable de entorno en el servidor para más seguridad, con el que se encripta la tupla usuario:contraseña, generando así el token con el que se accede al resto de URIs de la API.
    - La carpeta **models** ha sido utilizada para organizar todos los ficheros en los que se hace el mapeo de las tablas de la Base de Datos en la aplicación. Obsérvese el ejemplo siguiente, fichero que mapea con mongoose el esquema de la tabla User, donde se almacenan los usuarios de la aplicación:

```
/ /Load mongoose
var mongoose = require('mongoose').set('debug', true);

// Mongoose schemes
var Schema = mongoose.Schema;

// Creating Scheme object and attributes
var UserSchema = Schema({
        _id: Schema.Types.ObjectId,
        name: String,
        psswd: String,
},
{
        collection: 'User'
});

// Exporting the model to the other JS files
module.exports = mongoose.model('User', UserSchema);
```

    - En el directorio **controller** se han guardado los ficheros en los que se implementa la lógica principal de la aplicación. Por ejemplo, en

el fichero user.js dentro de esta carpeta se encuentran las funciones callback o retrollamadas para manejar los usuarios de la aplicación.

- Todo el contenido estático de la aplicación, es decir, todo el código relacionado con la interfaz de usuario, se ha organizado dentro del directorio llamado **public**. El código JavaScript que hace las llamadas a la API desde la interfaz de usuario se encuentra en la carpeta /public/static.

- El fichero **index.js** es donde se hace la conexión a la base de datos y se lanza la aplicación.

- Las URIs publicadas por la API se encuentran definidas en el fichero **app.js**. Las funciones definidas en la carpeta controller son utilizadas en este fichero.

Seguridad

Como ya se ha expuesto antes, una de las partes en las que hay que poner más hincapié a la hora de desarrollar Servicios o Sistemas IoT, es el tema de la seguridad, ya que este tipo de sistemas son especialmente vulnerables por estar constantemente transmitiendo y recibiendo información del Internet.

Por tanto, cuando desarrollando el Servicio Web IoT, se ha decidido implementar un sistema de tokens, para así dar acceso a las URIs de la API REST solamente a los usuarios que tenga un token correcto en posesión.

Esta es una técnica comúnmente utilizada en Servicios Webs, llamada **URL firmada.**

Este token se indica en la cabecera o cómo parámetro en la URI de la petición HTTP. Véase el Manual de Usuario del Apéndice B, para ver como hacer correcto uso del token cuando usando Interfaz web o POSTMAN.

A continuación, véase en la Figura 3.13 cómo funciona el sistema de tokens internamente mediante un diagrama de secuencia.

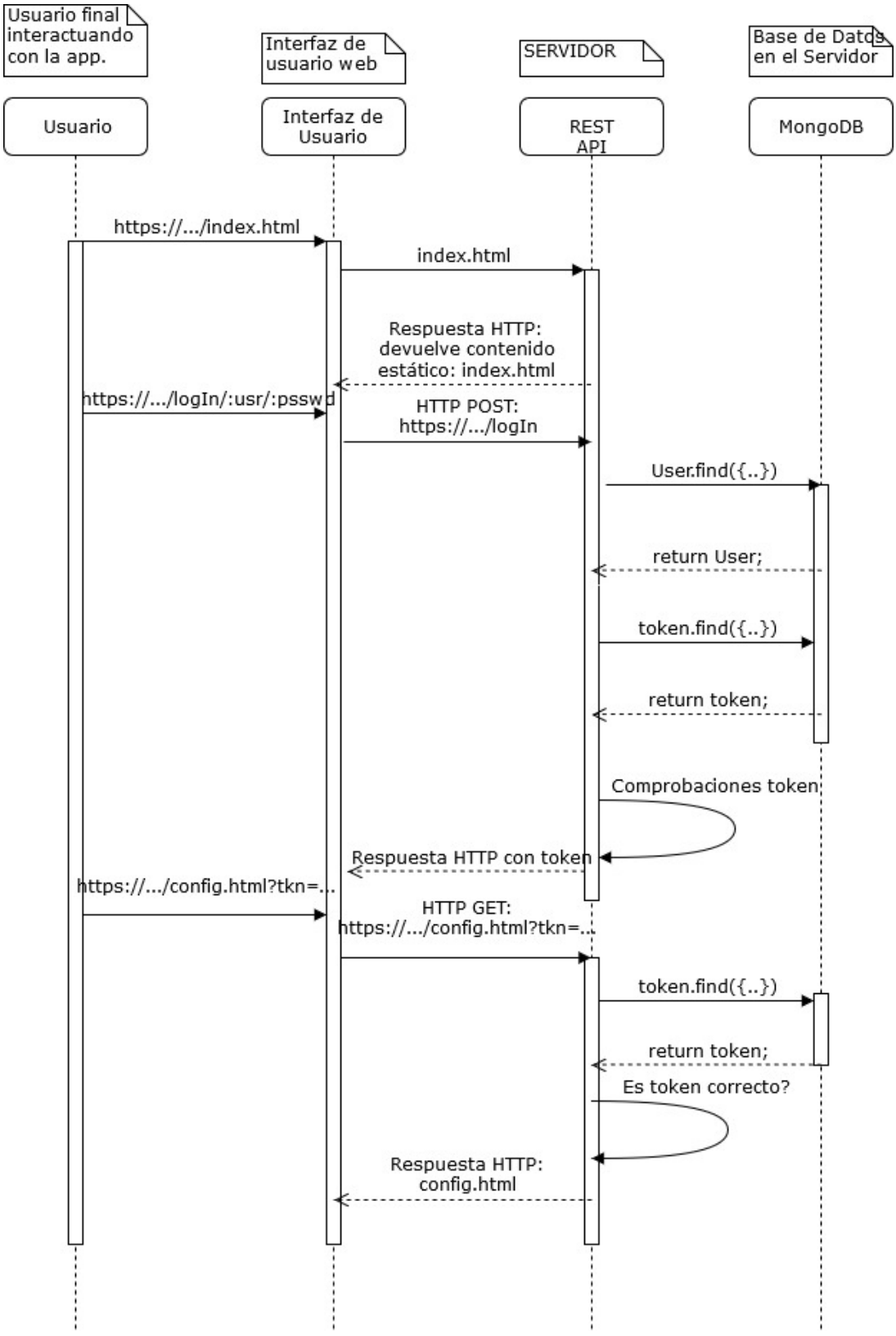

Figura 3.13. Diagrama de secuencia explicando funcionamiento interno del sistema de tokens.

El diagrama muestra lo siguiente, paso a paso:

1. El usuario accede a la página index.html de la aplicación, para así introducir su usuario y contraseña. Véase Figura 3.12
2. La información introducida por el usuario llega al Servidor como una petición HTTP a la URI: …/logIn/:user/:psswd
3. En el servidor, se comprueba que en la Base de Datos ese Usuario y contraseña existan. Si es así, se pasa a comprobar si el token también existiera en la Base de Datos. Si éste ya existe, por haberse generado con anterioridad, se le devuelve al Usuario en la Interfaz de Usuario web. Si no existiera, se genera simplemente encriptando la tupla [user:password] con una clave RSA 256, y se guarda en base de datos.
4. El usuario recibe el token en la interfaz de usuario, pudiendo directamente ser redirigido a la siguiente interfaz web en la que sí que es necesario el token para acceder a ella.

4

Implementación

En este capítulo se explica cómo se ha implementado el sistema, iteración por iteración, detallando también todos los problemas encontrados durante el proceso y como se han abordado.

El proyecto, inicialmente presenta una importante parte de montaje de distintos elementos HW. Por un lado, hay que montar sobre la placa Arduino,/Waspmote de la empresa Libelium, los diferentes sensores y componentes necesarios para su correcto funcionamiento: Waspmote o placa principal donde se encuentra la memoria (microSD) y el Microcontrolador, tarjeta Camara Sensor (tiene sensor de presencia, sensor de luminosidad, leds para visión nocturna y la cámara), tarjeta 3G/GPRS (controla las comunicaciones entre la tarjeta de la cámara y el microcontrolador) y tarjeta WiFi (encargada de establecer la comunicación con el Servidor IoT). Montaje de dichos dispositivos explicada Capítulo 3.
Para el desarrollo del software se aplicará una metodología iterativa incremental. Y se irán aplicando los distintos objetivos para establecer que se ha finalizado

cada iteración. De esta forma se obtendrán productos mínimamente viables en cada iteración que permitirán visualizar la evolución y avance del TFG.

El desarrollo de SW Iterativo, como su propio nombre indica, itera sobre las 4 fases típicas que se llevan a cabo, cuando hablamos del desarrollo de software, hasta que el proyecto se considera satisfactorio, es decir, cuando se cumplen todos los requisitos y funcionalidades. Esas fases sobre las que se itera son: Planificación, Diseño, Desarrollo y Testeo. En cada nueva iteración se pasa por cada una de ellas otra vez, modificando lo necesario gracias el feedback obtenido en la iteración anterior. Conforme vamos aumentando la iteración, los tiempos para planificación y diseño van disminuyendo, quedando posiblemente una última iteración en la que el 80% de su tiempo estará dedicado al testeo.

Antes de la implementación del sistema se ha tenido que realizar un estudio en profundidad de las herramientas y lenguajes a utilizar, sobre todo con todo lo relacionado con el dispositivo del Internet de las Cosas utilizado en este caso, la Videocámara sensor, ya que incluye una extensiva librería que se ha tenido que entender y modificar.

4.1 Primera iteración

Las tareas realizadas en la primera iteración de la implementación del sistema son las siguientes:

- Montaje del dispositivo IoT, la videocámara sensor, proceso descrito detalladamente en el capítulo 3.1.1. Se ha llevado a cabo según el manual de Libelium.

  El principal problema que se ha encontrado en este punto es con la batería de la Waspmote, que tiene unos cables muy frágiles que se rompieron varias veces, frenando el proceso bastante.

- Una vez montado el dispositivo, el desarrollo del código que lo controla se comenzó a desarrollar.

  Libelium ofrece múltiples ejemplos de uso de todas las funciones de sus librerías, por lo tanto, se ha comenzado el desarrollo de dicho programa a partir de uno de estos programas de ejemplo, primero haciendo un primer programa muy simple para testear el correcto funcionamiento de los módulos o sensores del dispositivo.

  El problema que se ha encontrado cuándo desarrollando el código del dispositivo fue que las librerías que Libelium proporciona no funcionan de por sí, si no que contienen varios errores que tuvieron que identificarse y cambiarse. Como los ficheros de dichas librerías son muy extensos y contienen múltiples funciones y distintas configuraciones posibles, ha sido un poco complicado solucionar dicho problema, pero finalmente con ayuda del soporte técnico de Libelium a través del foro de desarrolladores pudieron ser identificadas las líneas de código a modificar en los siguientes ficheros:

  - Fichero **Wasp3G.h**, línea 39,

    ```
    #include <WaspWIFI.h>
    ```

    estaba comentada, se descomenta, incluyendo así las funciones WiFi en el paquete 3G.

  - En el mismo fichero, **Wasp3G.h**, en la línea de código 43, se cambia la variable de configuración CAMERA_FUSE de 0 a 1, simplemente activando las funciones relacionadas con la Videocámara sensor:

    ```
    #define CAMERA_FUSE          1
    ```

  - Fichero **Wasp3G.cpp**, en la línea 9469, en la función llamada para mandar una foto a un servidor FTP a través de WiFi,

Wasp3G::sendFiletoWiFiFTP, el tercer parámetro de la llamada a la función WIFI.uploadFile, se cambia de la variable destiny_name a '.', indicando así que el fichero se guarde en el directorio raíz del servidor FTP con el mismo nombre que en el dispositivo.

```
answer = WIFI.uploadFile(origin_path, ".", ".");
```

- Otra vez en el fichero **Wasp3G.cpp**, en la línea 377, la variable constante FTP_TIMEOUT se ha tenido que cambiar de nombre por entrar en conflicto con el fichero WaspWIFI.h que también define esta variable en él, por tanto estando duplicada y generando un error. Se cambia el nombre a FTP_TIMEOUT2 en este fichero y no en el paquete WiFi, porque para este caso de uso específico, esta variable no se utiliza en el fichero Wasp3G.cpp pero si en el fichero WaspWIFI.cpp, siendo trivial para el correcto funcionamiento.

Una vez solucionados los errores de la librería, se ha comprobado el correcto funcionamiento de todo el hardware del dispositivo y se ha programado la primera versión del programa controlando la videocámara. Esta primera versión, que simplemente realiza una foto y la manda al servidor, a través de conexión WiFi, al servidor FTP.

- Instalación de todos el software necesario en el Servidor o Raspberry Pi. En el Capítulo 2.2, se lista todo el software instalado.

  El mayor problema afrontado en este punto fue la instalación de la base de datos MongoDB, ya que al utilizar una Raspberry Pi como hardware para el servidor, ésta tiene un microcontrolador con una arquitectura especial que las últimas versiones de MongoDB no soportan. El problema se ha resuelto instalando la versión de MongoDB que Raspberry Pi foundation proporciona con el comando apt-get. Esta versión de MongoDB

no es original pero está basada en la versión estable original de MongoDB, la versión 2.4.14.

- Una vez todo el software en el lado del servidor ha sido instalado con éxito, se crea la Base de Datos a utilizar en mongoDB, VideoCamera, con la tabla User, para guardar los usuarios de la API REST. Se crea el usuario a nivel de Base de Datos para añadir aún más seguridad al acceso a la Base de Datos.

```
pi@raspberrypi:~/VideoCameraAPI $ mongo
MongoDB shell version: 2.4.14
connecting to: test
> use VideoCamera
switched to db VideoCamera
> db.auth('mongo','Elkosaone18')
1
> db.User.find()
{ "_id" : ObjectId("5d04d79593a961c22d2d1a09"), "name" : "Javier", "psswd" : "Elkosaone18!" }
{ "_id" : ObjectId("5d062e344ff9c7788f1ba031"), "name" : "Mercedes", "psswd" : "TFGuma2019" }
>
```

Figura 4.1. Usuarios de la API REST en MongoDB.

- La primera versión de la aplicación ha sido desarrollada en esta primera iteración, cuando todo el software ha sido instalado y configurado con éxito.

  Esta primera versión no contiene nada más que la conexión a la Base de Datos y varias URIs a modo de prueba de la aplicación. Incluye también el middleware necesario para la autenticación de los usuarios y el manejo de tokens con los que los usuarios luego accederán a la API.

  Las primeras funciones para obtener los ficheros generados por la videocámara son creadas en esta iteración. Ejemplo:

```
app.get("/pictureDownload/:id", md_auth.ensureAuth, function(request, response, next) {
  var file = dirname + request.params.id;
  response.download(file);
});
```

Con este código se publica la URI pictureDonwload/, donde :id sería sustituido por el identificador de la foto en la base de datos.

md_auth.ensureAuth es la función middleware que comprueba que haya un token en la cabecera de la petición HTTP y que este sea correcto.

4.2 Segunda iteración

En esta segunda iteración de desarrollo software, se ha estudiado cual sería el mejor protocolo de comunicación compatible con la Waspmote utilizada para realizar la comunicación desde el dispositivo o videocámara hacia el servidor de la forma segura más segura posible. Los protocolos que se han evaluado como opción son:

- MQTT, ya que es uno de los principales protocolos adoptados por las aplicaciones del Internet de las Cosas. Es un protocolo de publicación/subscripción que permite la implementación en hardware de dispositivos altamente restringidos y en redes de ancho de banda limitada y de alta latencia.

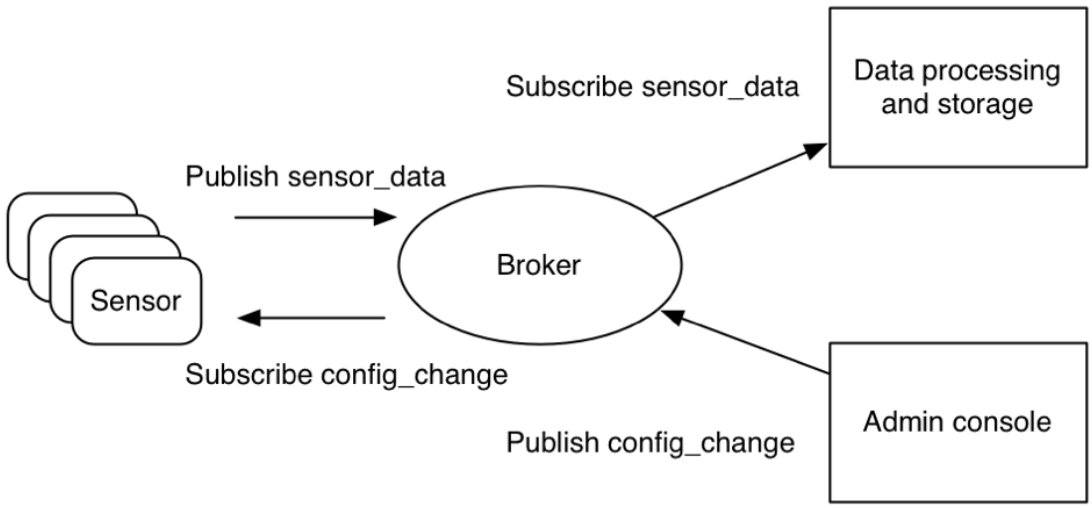

Figura 4.2. Diagrama explicativo de funcionamiento de protocolos publicación/subscripción.

El problema por el cual no se ha hecho uso de este protocolo para llevar a cabo la comunicación con el servidor es que la versión de Libelium Waspmote utilizada para este proyecto fue producida en 2012, mientras que el protocolo MQTT

comenzó a convertirse en un estándar en 2014, la librería de la Waspmote no soporta dicho protocolo.

- SFTP, SSH File Transfer Protocol. Permite transferir datos cifrados entre un cliente y un servidor. Envía y recibe los datos en binario, no en formato texto como FTP.
- SCP, Secure Copy Protocol. Similar a SFTP, también utiliza SSH para transferir datos cifrados.
- FTPS o FTP/SSL. Es una extensión de FTP que utiliza SSL para el cifrado de los datos. Utiliza los mismos comandos que FTP.

No se ha podido utilizar ninguno de estos protocolos por que la versión de Waspmote utilizada en el proyecto, versión 12, no soporta la encriptación necesaria que utilizan estos protocolos. En cambio, la siguiente versión de Waspmote, la versión 15, si soporta el protocolo FTPS.
Por lo tanto, el protocolo utilizado para transferir los ficheros de la Waspmote al servidor es FTP, ya que es un protocolo que funciona con TCP. Libelium proporciona en su librería las funciones necesarias ya implementadas para transferir datos con FTP.

4.3 Tercera iteración

Esta es la última iteración de desarrollo software. En la previa iteración todo el software necesario ha sido instalado. Por lo tanto, el programa final controlando la videocámara ha sido desarrollado. También se han añadido todas las funcionalidades necesarias en el lado del Servidor para que el usuario pueda añadir, modificar y eliminar: dispositivos, ficheros de configuración de dichos dispositivos y los ficheros generados por dichos dispositivos.

Cuando desarrollando el código de la videocámara sensor, el problema que se ha afrontado es el de recibir los datos de configuración enviados desde el servidor. Como se ha decido hacer la conexión a Internet solamente para transferir datos al servidor, recibir datos de configuración en el dispositivo se convierte en un problema. La configuración remota del dispositivo no se ha implementado, teniendo que configurarse en el código del dispositivo localmente. Se ha programado el código de manera que los LEDs se auto configuran dependiendo de la luminosidad detectada, y los otros aspectos configurables como la resolución y el brillo se tienen que cambiar manualmente en el código. Véase Apéndice 1 para ver cómo se ha de configurar la Videocámara.

Cuando desarrollando el código del servidor, se han realizado las siguientes tareas:

- Crear las tablas necesarias en la base de datos para guardar la información necesaria para dar soporte a las nuevas funcionalidades añadidas a la API.
- Añadir las siguientes funcionalidades en el código de la aplicación:
    - Registrar y modificar nuevos dispositivos en el sistema IoT.
    - Obtener los usuarios autorizados para utilizar la API.
    - Control de eventos de ficheros recibidos por la videocámara.
    - Descargar u obtener fotos y vídeos recibidos con anterioridad por la videocámara.
    - Crear y modificar ficheros de configuración para los dispositivos registrados en la aplicación.
    - Desarrollo de la interfaz de usuario.

5

Conclusiones

Finalmente, después de la última iteración explicada en el Capítulo 4.3, se ha logrado cumplir con todos los objetivos del TFG expuestos en el Capítulo 1.2, llegando a comprender la importancia que tiene la seguridad en sistemas de la Internet de las Cosas, ya que ofrece muchas ventajas pero múltiples puntos de vulnerabilidad a la vez. Y tan sólo con que ser capaz de vulnerar uno de ellos, desde un sistema informático completo de una gran empresa a una casa doméstica, se pueden ver en compromiso.

La creación con éxito de un sistema IoT completo, entraña ciertos retos. Se ha de realizar esfuerzos extra para cumplir con todos los requisitos de la empresa o el negocio, empezando con los componentes a utilizar, que deben tener larga vida de baterías con bajo consumo de energía, enlaces de red seguros y hardware robusto, y continuando por un diseño del software, con múltiples protocolos de comunicación, que ha de ser eficiente y super seguro a la vez.

Durante el desarrollo del proyecto se ha tenido la oportunidad de:

- Trabajar con dispositivos IoT reales, mejorando mis habilidades con dispositivos hardware Arduino.
- Comprender mejor la importancia de la selección de los protocolos de comunicación en un sistema IoT, aprendiendo nuevos protocolos como MQTT, SFTP, FTPS y SCP.
- Mejorar mis habilidades con JavaScript en general, entendiendo mejor porque JSON se está convirtiendo en el nuevo estándar de intercambio de datos entre servicios web.
- Aprender a fondo el funcionamiento de Node.js y la cantidad de soporte de la comunidad que tiene, por lo que tiene una gran importancia hoy en día para la programación de servicios Web ligeros y eficientes.
- Desarrollar desde cero mi primera API REST con Node.js y Express, teniendo que hacer múltiples tutoriales y estudiar desde 0 las Promesas (Promises) de Node.js, para poder hacer buen uso de las funciones de retro llamada o callback. Estas funciones son necesarias para el correcto funcionamiento de Node.js y Express, cómo se puede observar en la Figura 2.11.

Como mayor problema encontrado cuando desarrollando el proyecto, cabe destacar la dificultad de integración de múltiples dispositivos *legacy* con otros múltiples protocolos de comunicación, teniendo que encajar el protocolo de comunicación con el hardware que lo soporte. Específicamente, por tener una versión un poco antigua de la Libelium Waspmote, no se ha podido lograr implementar la comunicación del Servidor a la Waspmote para enviar datos de configuración. Sería muy interesante, para un trabajo futuro, añadir al sistema esta comunicación.

También, otra tarea que se podría considerar como el siguiente paso, es la implementación de la comunicación de la Videocámara sensor al servidor con más seguridad. Para ello, haría falta utilizar la versión 15 de la Waspmote, que incorpora el hardware necesario para poder realizar la encriptación que requiere el protocolo FTPS, por ejemplo.
Ya que el sistema ha sido diseñado para incorporar más dispositivos IoT, gracias a su modularidad, sería muy fácil integrar la lógica necesaria en la aplicación para poder interactuar con más dispositivos, o con POSTMAN, o con la creación de más interfaces de usuario web.

Haber realizado este proyecto me ha motivado a continuar especializándome en las nuevas tendencias directamente relacionadas con la Internet de las Cosas y la Industria 4.0, como la computación en la nube, los microservicios y los sistemas automatizados y auto-escalables.

Bibliografía

Título: Guía Técnica de Libelium Waspmote
Autor: Libelium.
URL: http://www.libelium.com/v12/development/waspmote

Título: REST API Security Essentials.
Autor: RESTfulAPI.net.
URL: http://restfulapi.net

Autor: Raspberry.org.
URL: https://www.raspberrypi.org/

Título: Estructura de una REST API con Node.js, Express y MongoDB.
Autor: William Bastidas.
URL: https://medium.com/williambastidasblog/estructura-de-una-api-rest-con-nodejs-express-y-mongodb-cdd97637b18b

Título: MQTT
Autor: MQTT org.
URL: http://mqtt.org/

Título: Mongoose, elegant mongodb object modelling for node.js
Autor: Mongoose
URL: https://mongoosejs.com/

Título: ExpressJS
Autor: Node.js Foundation
URL: https://expressjs.com/

Título: Node.js for Beginners
Autor: Maciej Sopyło
URL: https://code.tutsplus.com/tutorials/nodejs-for-beginners--net-26314

Título: Stack Overflow
Autor: Stack Overflow
URL: https://stackoverflow.com/

Apéndice A

Manual de Instalación

Requerimientos:

Para poder instalar y utilizar el sistema que se ha implementado durante el desarrollo del TFG es necesario primero tener todo el hardware listado en el Capítulo 2.1, y ensamblarlo o montarlo tal y como se indica en dicho Capítulo.

La instalación consta de dos partes bien diferenciadas, la instalación del dispositivo, Videocámara sensor en este caso, y la instalación del servidor.

Instalación Videocámara:

El primer paso de la instalación del dispositivo es la descarga e instalación del IDE que nos permite trabajar con el dispositivo, cargando el código y depurándolo. Libelium ofrece múltiples versiones de dicho IDE, por lo tanto es importante identificar la versión que es compatible con la versión de Waspmote utilizada, en nuestro caso como se ha escogido la Libelium Waspmote versión 12, la última versión del IDE que soporte a nuestra versión de Waspmote es la versión 4.

1. Para descargar la versión correcta del IDE, acceder al siguiente enlace[3].

[3] http://www.libelium.com/v12/development/waspmote/sdk_applications/.

Al descargar el IDE, se descargan junto con él la API de Libelium, o librería, con todas las funciones necesarias para controlar todos los módulos integrables a la Waspmote.

Cuando se descomprima el fichero descargado cuando descargando el IDE, es muy importante preservar la estructura de los directorios y ficheros descargados, si no el IDE no funcionará correctamente.

2. Una vez el IDE correcto ha sido descargado y descomprimido, el siguiente paso es la instalación de los drivers necesarios en el ordenador para que este pueda reconocer a la Waspmote cuando conectada a través del cable micro USB.

   La instalación de dichos drivers depende del sistema operativo utilizado.

   Windows:

   En el Sistema Operativo Windows, la instalación de estos drivers es automática. El proceso de instalación se inicia automáticamente una vez se conecta la Waspmote al ordenador con el cable micro USB, si el usuario no ha conectado nunca a su ordenador la Waspmote.

   Si la instalación no ocurriera automáticamente y el ordenador no reconociera a la Waspmote, los drivers pueden ser instalados explícitamente descargándolos desde el siguiente enlace[4].

   Mac OS X:

   En Mac, el fichero ejecutable de la aplicación debe ser copiado en el directorio de Aplicaciones en primer lugar.

[4] https://www.ftdichip.com/Drivers/VCP.htm

La instalación de los drivers no se realiza de forma automática en el caso de usar este Sistema Operativo, por lo tanto, han de descargarse explícitamente a través del enlace[5].

Una vez descargados los drivers correspondientes para Mac OS, con hacer doble click en dicho fichero la instalación se realiza automáticamente sin necesidad de interacción del usuario.

Linux:

Para la instalación en Linux, hay que atender a la distribución de Linux utilizada, ya que los paquetes a instalar dependen de esta. Los programas a instalar son:

- Entorno de Java, Java Runtime Environment en inglés: de entre openjdk-7-jre, openjdk-6-jre, Sun's Java 6 runtime o Oracle JRE 7, escoger el programa compatible con la distribución de Linux utilizada.
- El IDE de Waspmote incluye un compilador gcc interno pre-construido, para poder compilar el código C escrito en la videocámara, pero si se ha instalado un compilador avr-gcc propio la versión a descargar del IDE debe ser la 4.7.2 o superior.
- El paquete libxtx-java. Librería Java para dispositivos con micro-controladores de arquitectura similar a las placas Arduino.

Una vez dichos programas o paquetes han sido instalados, sea cual sea el sistema operativo, ya se puede conectar la Waspmote al ordenador y esta ha de ser reconocida por el ordenador sin problema. Si esto no ocurriera, revisar que el cable micro USB no está dañado y que esté bien conectado a ambos extremos.

[5] https://www.ftdichip.com/Drivers/VCP.htm

3. El último paso para completar la instalación o despliegue de la Videocámara sensor es cargar el código o programa desarrollado para controlar dicho dispositivo.

    a. Abrimos el IDE de Waspmote con la placa o dispositivo Waspmote conectado al ordenador con el cable USB, versión 4. Obsérvese Figura A.1.

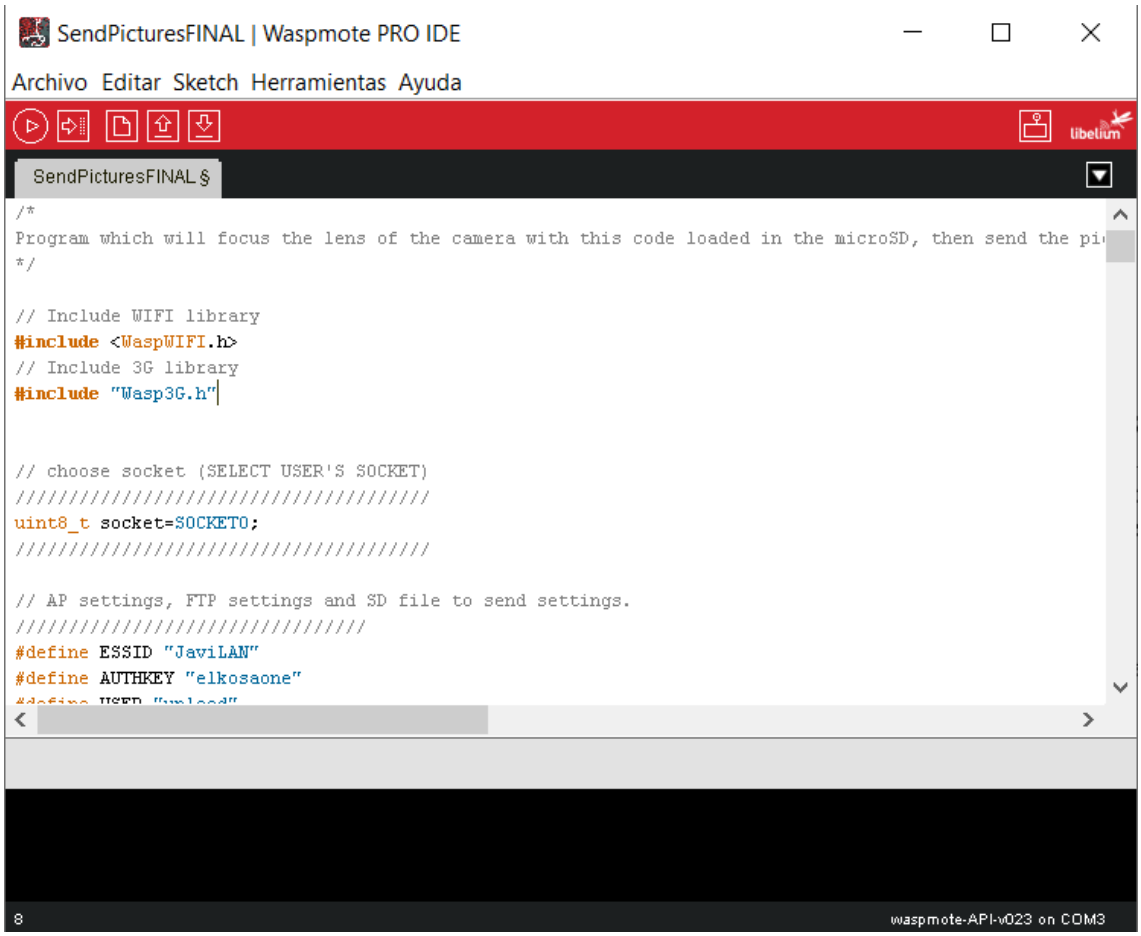

Figura A.1. Waspmote IDE versión 4.

    b. Ahora es cuando debemos de asegurarnos que la Waspmote es reconocida por el ordenador, para poder cargar más adelante el código que controla el dispositivo.

    Cuando estando en la Figura A.1, haga click en Herramientas y en puerto serial debe aparecer el puerto en el que está conectado la Waspmote, pero este valor va a variar dependiendo del Sistema Operativo que se esté utilizando. Antes de revisar el valor puerto

serial, asegurarse que el único dispositivo conectado a través de USB al ordenador es la Waspmote.

En **Windows**, la Waspmote será vinculada a un puerto COMx, donde la x será sustituida por un número.

En **Mac OS X**, el valor de puerto serial viene con el formato: */dev/tty.usbserial-XXXXXX*. Obsérvese Figura A.2 para ver ejemplo de dicho formato.

En **Linux,** el formato es parecido al de Mac OS, pero un poco distinto: /dev/ttyUSBX.

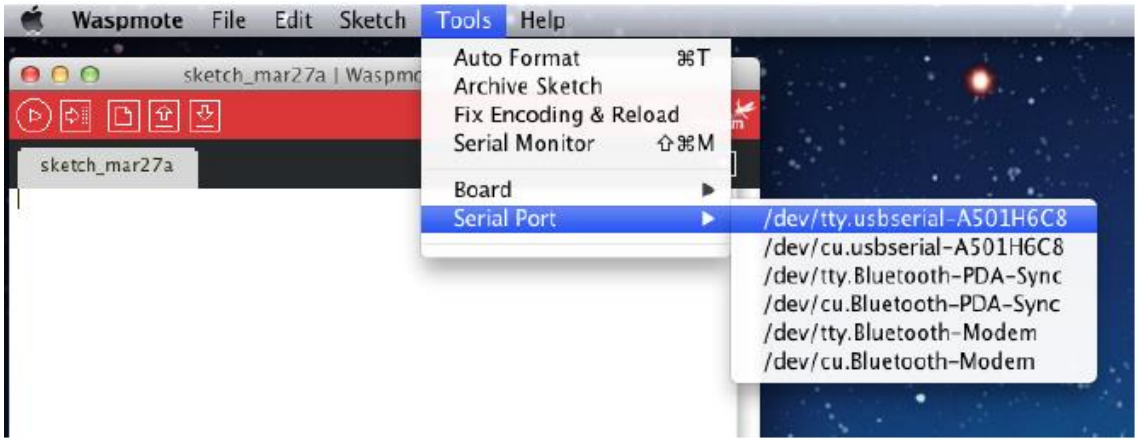

Figura A.2. Ejemplo de puerto serial para Mac OS X.

Escoger el de entre los valores puerto serial el correspondiente a la Waspmote.

c. A continuación, el fichero hay que buscar el fichero del programa de control de la Videocámara entre los ficheros proporcionados. Hacer click en el botón Cargar, véase Figura A.3, para subir el código que se encuentre en el IDE a la Waspmote. Buscar entre los archivos proporcionados la carpeta con nombre VideocamaraController, y el fichero a cargar se llama VideocamaraController.pde. Para abrir el explorador de archivos, click en Archivo y luego click en Abrir....

Figura A.3. Waspmote IDE versión 4, botón para cargar código.

d. Al hacer click en Cargar con el código correcto escogido y abierto en el IDE, en la ventana negra de debajo de la Figura A.1 debe aparecer el mensaje mostrado en la Figura A.4, y si abrimos el Monitor Serial, haciendo click en Herramientas también, podremos observar los mensajes que el código muestra, para observar si funciona correctamente. Monitor Serial explicado en el Manual de Usuario.

   Si algún problema ocurriera durante la compilación o carga del código en la Waspmote, el mensaje de la figura A.4 se mostraría en rojo, especificando el error exacto.

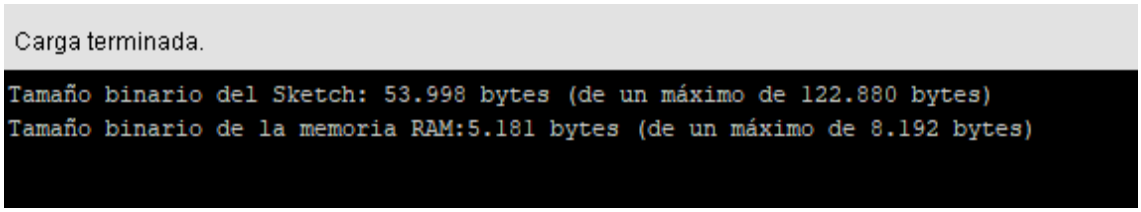

Figura A.4. Mensaje de Correcta compilación y carga del código en la Waspmote.

Instalación del Servidor:

Para la instalación de todo el Software del Servidor desde 0, seguir los siguientes pasos:

1. Lo primero es descargar e instalar el Sistema Operativo del Servidor, Raspbian Stretch Lite. Sigue el siguiente enlace para descargar el Sistema Operativo, página oficial de Raspberry Pi Foundation[6]:
2. Para escribir la imagen del Sitema Operativo correctamente en la tarjeta SD de la Raspberry Pi, se necesita de una herramienta de escritura que

[6] https://www.raspberrypi.org/downloads/

permita instalar el sistema operativo en la tarjeta SD desde la imagen descargada.

La herramienta recomendada por Raspberry Pi Foundation para llevar a cabo la instalación del Sistema Operativo es Balena Etcher:

a. Descarga e instala la herramienta a través del siguiente enlace[7].
b. Conecta la tarjeta SD al ordenador , ya sea con un lector de tarjetas SD interno o externo.
c. Abre balenaEtcher, selecciona del disco duro el fichero descargado con la imagen Raspberry Pi .img o .zip.
d. Selecciona la tarjeta SD en la que instalar la imagen de Sistema Operativo.
e. Revisa que la imagen elegida sea la correcta al igual que la tarjeta SD, haz click en Flash! para empezar la escritura y espera a su finalización.

3. Una vez escrita la imagen del Sistema Operativo en la tarjeta SD, insertar ésta en la Raspberry Pi.
4. Conectar la Raspberry Pi a una pantalla a través del conector HDMI, y a ratón y teclado a través de las conectores USB.
5. Conectar la Raspberry Pi a la corriente con un cable USB cagador tipo B, obsérvese Figura A.1. Este cable es el utilizado normalmente para la carga de la batería de los Smartphones.
6. Cuando la Raspberry es conectada a la corriente, el diálogo para la primera configuración del Sistema Operativo comenzará, en donde ha de indicarse País, lenguaje y demás preferencias de usuario. Hay que prestar especial atención al nombre de usuario y contraseña elegidos, así como de la configuración WiFi, en la que se debe indicar la red a la que se desea que

[7] https://www.balena.io/etcher/

la Raspberry Pi se conecte automáticamente al arrancar. Si se deseara modificar la red WiFi a la que la Raspberry se debe conectar automáticamente, esto se puede configurar modificando el fichero /etc/wpa_supplicant/wpa_supplicant.conf del Sistema Operativo, definiendo cada red a la que la Raspberry se conecta automaticamente de la siguiente manera:

```
network={
    ssid="JaviLAN"
    psk="elkosaone"
    key_mgmt=WPA-PSK
}
```

Se pueden definir tantas redes como se quiera.

Figura A.5. Cargador USB tipo B.

7. Antes de apagar o reiniciar la Raspberry Pi, es muy importante habilitar la conexión SSH para poder así continuar trabajando con el Servidor sin necesidad de conectar la Raspberry Pi a una pantalla, teclado y ratón. Para ello, abrir el terminal de Raspbian e escribir el siguiente comando:

   ```
   sudo raspi-config
   ```

   para abrir la herramienta de configuración software que Raspbian proporciona. Elegir la opción, Interfacing Options, para activar la conexión por SSH, Obsérvese Figura A.6.

En la siguiente ventana (Figura A.7), elegir SSH y seleccionar en la siguiente venta Sí, para activar el acceso al Servidor a través de SSH.

Si se desea añadir más seguridad al sistema, cuando en la Figura A.3, elegir la opción VNC para luego indicar No, deshabilitando así el acceso remoto gráfico a través de VNC.

Para terminar, haga click en Finish o Terminar, dependiendo del lenguaje elegido, para así terminar la configuración.

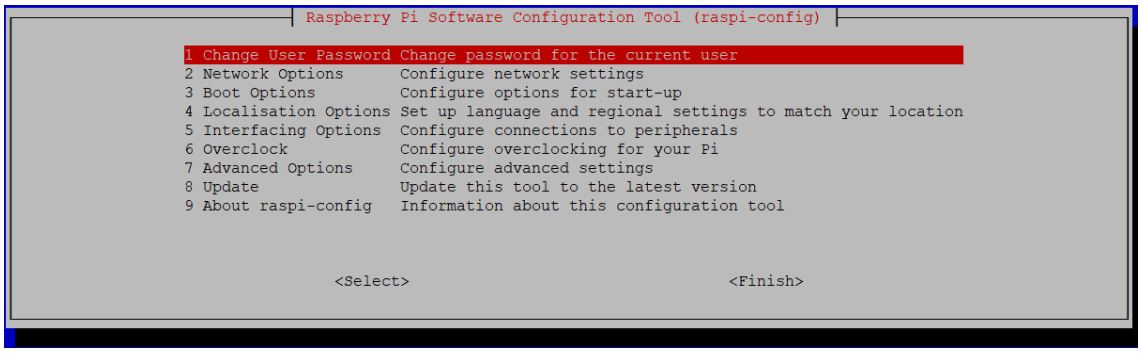

Figura A.6. Herramienta de configuración de Software de Raspberry Pi (raspi-config).

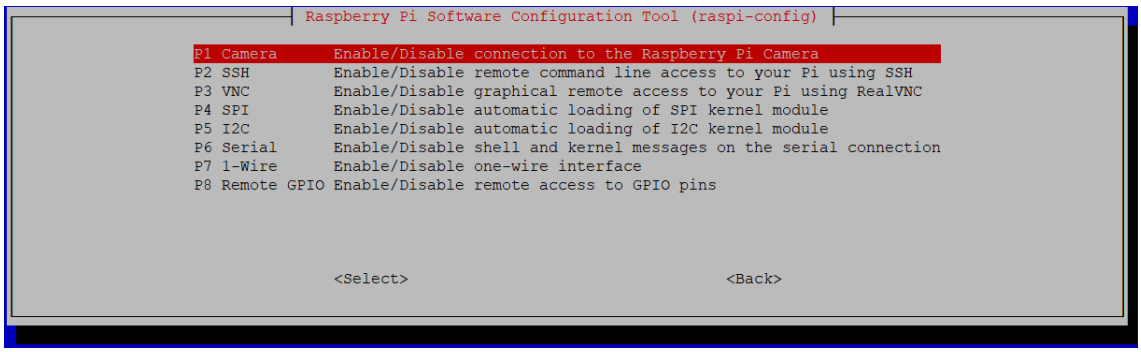

Figura A.7. Opciones de interconexión de raspi-config (Interfacing Options).

8. Para conectarse a través de SSH al servidor, es necesario, antes de reiniciar o apagar la Raspberry Pi, configurar Raspberry Pi para que obtenga la misma IP estática cada vez que se reinicie. El fichero ha modificar para realizar dicha configuración es /etc/dhcpcd.conf, el fichero de configuración del servicio DHCP preinstalado en Raspbian.

Añadir el siguiente código al final de dicho fichero cambiando la IP estática deseada para el Servidor y la dirección del enrutador, que van a depender específicamente de la red:

```
interface wlan0
static ip_address=192.168.43.52/24
static routers=192.168.43.1
```

Si la conexión fuera por cable Ethernet, cambiar wlan0 por eth0.

9. Reiniciar Raspberry Pi para aplicar los cambios de configuración recién realizados. Para reiniciar, ejecutar:

```
sudo reboot now
```

Mientras el dispositivo se reinicia, desconectar de la Raspberry todas las conexiones excepto la de alimentación.

10. A partir de ahora, para interactuar con el Servidor se utilizará conexión SSH, recomendable utilizar PuTTY. Obsérvese Figura A.4, en donde se debe indicar la dirección IP del servidor y el puerto escogido para la conexión SSH, que por defecto es el 22 si no se ha cambiado.

    Cuando PuTTY pide la configuración, dejarla como aparece en la Figura A.3 y simplemente indicar la dirección IP del servidor, luego hacer click en Open.

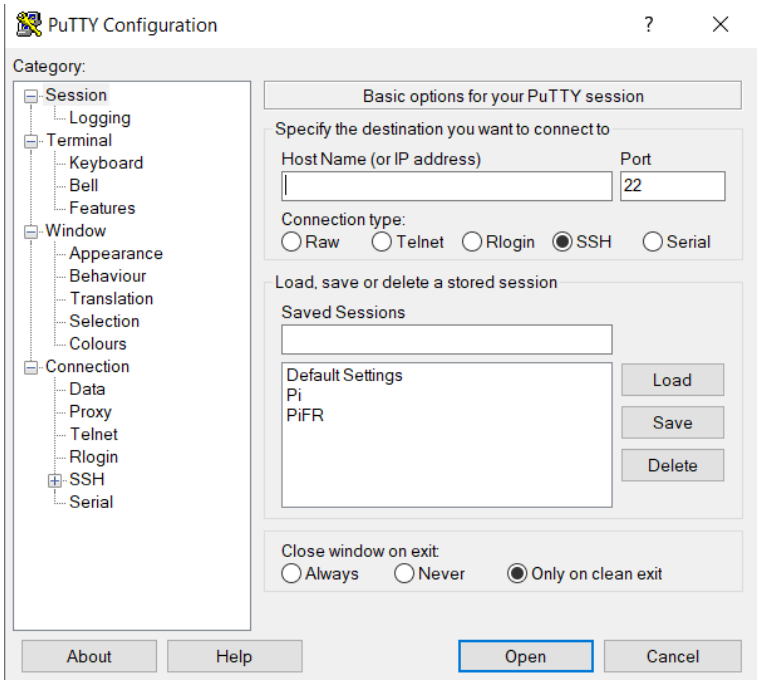

Figura A.8. PuTTY para conexión SSH con el Servidor.

11. Introducir usuario y contraseña que se indicaron en el paso 6 para acceder a la línea de comandos del servidor.
12. Siguiente paso, es la instalación del software necesario para que la aplicación funcione correctamente.
    a. Instalación del servidor FTP, pureFTP:

    Instala el programa:

    ```
    sudo apt-get install pure-ftpd
    ```

    Crea un grupo del sistema para pureFTP, crea el directorio raíz del Servidor FTP y crea un usuario del sistema que no pueda acceder a la línea de comandos pero si al directorio raíz de pureFTP:

    ```
    groupadd ftpgroup
    mkdir /home/pi/pureFTP
    useradd -g ftpgroup -d /dev/null -s /home/pi/pureFTP ftpuser
    sudo chown -R ftpgroup:frpuser /home/pi/ftp
    ```

    Crear usuario de pureFTP:

    ```
    pure-pw useradd upload -u www-data -g www-data -d
    /home/pi/pureFTP
    ```

    b. Instalación y configuración de MongoDB:

    Para instalar la versión correcta de MongoDB, que es la desarrollada por Raspberry Pi Foundation, ejecutar el siguiente comando:

    ```
    sudo apt-get install mongodb
    ```

    Cambiar el fichero de configuración de mongoDB por el fichero proporcionado, mongodb.conf. Debe de sustituirse en el directorio /etc/.

    Ejecutar mongoDB y crear el usuario de la Base de Datos, ejecutando el siguiente comando en mongo DB:

    ```
    db.createUser({
            user: "mongo",
             pwd: "ElKosaone18",
            roles: [ { role: "readWrite", db: "VideoCamara" } ]
             }
    )
    ```

Crear las colecciones necesarias para el funcionamiento de la aplicación, ejecutando los siguientes comandos, en orden, una vez dentro de mongoDB:

Elegir el esquema de base de datos y autenticarse en la base de datos:

```
use VideoCamera;
db.auth('mongo','Elkosaone18');
```

Para crear la colección o tabla que va contener la información necesaria de los dispositivos registrados en la aplicación:

```
db.createCollection('device', { autoIndexId: true }, { _id: new
ObjectId(), name: String, function: String });
```

Colección files, que guarda toda la información necesaria relativa a los ficheros generados por la videocámara:

```
db.createCollection('files',{autoIndexId: true},{_id: new ObjectId(),
filename: String, time : { type : Date, default: Date.now }, dev: String
});
```

Colección configuration, almecena la ruta de los ficheros de configuración generados para cada dispositivo:

```
db.createCollection('configuration', {autoIndexId: false}, {device:
String, filePath: String});
```

Colección User, que ha sido creada para almacenar los usuarios que autorizados para interactuar con la API REST:

```
db.createCollection('User', {autoIndexId: true}, {_id: new
ObjectId(), name: String, psswd: String});
```

Colección token, que guarda las tuplas usuario:contraseña junto con el token generado para dicho usuario:

```
db.createCollection(token, {autoIndexId: true}, {_id: new
ObjectId(), user: String, token: String});
```

Los dos siguientes comandos de MongoDB insertan los usuarios iniciales de la aplicación:

```
db.User.insert({ "_id" : ObjectId("5d04d79593a961c22d2d1a09"), "name" : "Javier", "psswd" : "Elkosaone18!" });

db.User.insert({ "_id" : ObjectId("5d062e344ff9c7788f1ba031"), "name" : "Mercedes", "psswd" : "TFGuma2019" });
```

c. Instalación y configuración de Node.js y sus paquetes (Tabla 2.1). Esta parte de la instalación es más simple.

Primero debemos instalar Node Package Manager, programa para administrar los paquetes de Node.js:

```
sudo apt-get install node npm
```

El siguiente paso, es copiar los ficheros proporcionados de la aplicación del Servidor en el directorio /home/pi. Los ficheros de la aplicación se encuentran todos en la carpeta VideoCameraAPI, y se debe copiar el directorio entero quedando en el servidor en la ruta /home/pi/VideoCameraAPI.

Navegar en el interior del directorio de la aplicación Node.js e instalar los paquetes necesarios, ejecutando los siguientes comandos:

```
cd /home/pi/VideoCameraAPI

npm install
```

No hace falta ejecutar más comandos, ni instalar paquetes uno a uno, ya que el programa npm organiza los paquetes instalados y sus versiones en el fichero package.json. Por tanto, al ejecutar el comando anterior, npm buscará, en el directorio donde se acaba de ejecutar, el fichero package.json para instalar los paquetes indicados en él.

Apéndice B

Manual de Usuario

En este Capítulo se explica como el usuario final, ya sea experimentado o no, puede utilizar todas las funcionalidades que el sistema ofrece. Para ello, primero se explicará como configurar el dispositivo o Videocámara sensor, utilizando el IDE que Libelium ofrece. Más tarde, una guía de las URIs que la API creada ofrece y de cómo utilizarlas.

Manual de Usuario: Videocámara

Véase Apéndice A, dónde se encuentra la guía para instalar todo el software necesario para interactuar con la Videocámara, que representa un dispositivo del Internet de las Cosas.

Una vez que todo el Software ha sido instalado sin problemas y el código cargado exitosamente, la configuración interna de la Videocámara sensor puede ser modificada. Como realmente la Videocámara tampoco ofrece un extensa lista de distintas configuraciones, con observar la tabla B.1 se puede entender fácilmente los aspectos configurables de este dispositivo.

La columna variable, es el nombre de la variable en el código de la cámara. Es decir, la variable #define a buscar para cambiar dentro del mismo código a cargar en la Videocámara. Obsérvese Figura B.1, para ver el nombre de las variables en

el código. Esta parte del código se encuentra en las primeras líneas, justo después de la importación de las librerías necesarias y justo antes de la función setup(). La función setup() del código se ejecuta una vez al encender el dispositivo, mientras que la función loop() del código, se ejecuta repetidamente hasta que se apaga el dispositivo. Funciona exactamente igual que cualquier código para cualquier dispositivo Arduino.

Variable	Definición	Valores código	Valor real
RESOLUTION	Resolución de fotos y vídeos.	0	80x48
		1	160x120
		2	176x144
		3	320x240
		4	352x288
		5	640x480
ROTATION	Ángulo de rotación de la cámara.	0	0º
		90	90º
		180	180º
		270	270º
VIDPIC	Escoger si hacer foto o vídeo.	picture	Para hacer fotos
		video	Para hacer videos
FPS	Si se ha escogido realizar videos, para establecer los fps.	0	7.5 fps
		1	10 fps
		2	15 fps
BRIGHTNESS	Configuración de brillo.	[0-6]	De mínimo a máximo brillo posible.
PICNAME	Como nombrar a los ficheros generados.	0	pic_XXXX.jpg
		[String]	[String]_XXXX.jpg

Tabla B.1. Posibles valores de configuración de la Videocámara sensor.

```
// Variables needed for the VideoCamera specific settings.
////////////////////////////////////
#define ANGLE 0        // Angle for camera rotation
#define RESOLUTION 5
#define BRIGHTNESS 0
#define PICVID "picture"
#define FPS 0
#define PICNAME 0
////////////////////////////////
```

Figura B.1. Variables de configuración en el código de la Videocámara.

Explicación de cómo operar con el IDE de Waspmote. Los números a los que se hace referencia para la explicación se corresponden con los números de los elementos de la Figura B.2.

1. Con el botón número 1 se puede compilar el código. El resultado de la compilación se muestra en la ventana negra de debajo de la Figura B.2.
2. El botón número 2, aparte de compilar el código, sirve para cargar el código en la Waspmote.
3. Si el botón con el número 3 es pulsado, una nueva ventana del IDE exactamente igual será abierta, pudiendo tener varias hojas de código abiertas a la vez.
4. El botón 4 abre el explorador de archivos para abrir otra ventana con otra hoja de código, pero ésta vez una hoja de código ya existente.
5. El botón 5 sirve para guardar la hoja de código actual.
6. Si se selecciona la opción 6, se puede elegir entre las distintas versiones de la API o librería Libelium, si es que se tienen distintas versiones en el directorio dónde se encuentra el IDE.
7. Al seleccionar Puerto Serial, se despliega la lista de puertos en los que hay dispositivos reconocidos conectados.
8. Lista de puertos. En esta imagen, Figura B.2, aparece el puerto COM3 porque la máquina en la que se tomó la captura tiene un Sistema Operativo Windows, y sólo tiene conectada la Waspmote.

9. Por último, el Monitor Serial, por el cuál si la Waspmote está encendida y con el código de la aplicación cargado, se mostrará información de depuración de errores del código.

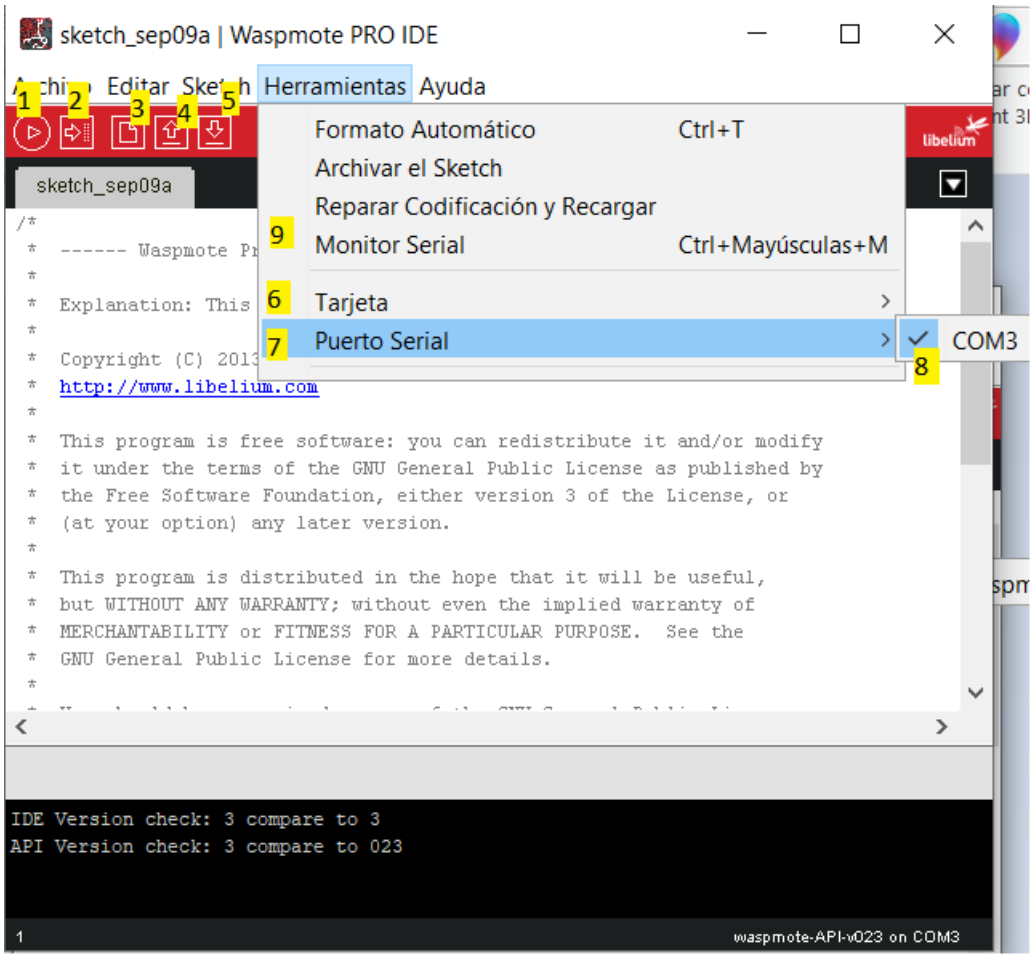

Figura B.2. Explicación del IDE de Waspmote.

Manual de Usuario: Servidor IoT

Una vez finalizada la configuración e instalación del Servidor del Internet de las Cosas, Apéndice A, el usuario podrá interactuar con la Videocámara sensor y los ficheros que esta genera a través de una API REST, que publica una serie de URIs para cada funcionalidad que se ha implementado. Si se hace una petición HTTP a esas URIs con los parámetros correctos, el servidor responderá devolviendo los datos que se han pedido, si la petición HTTP fuera del tipo GET. En cambio, si la petición es del tipo DELETE, el servidor simplemente devolverá

un mensaje de control indicando si se ha borrado algo del sistema o no, indicando el estado de la respuesta de la petición con los códigos de respuesta HTTP.

Explicación de los códigos de estado de peticiones HTTP:

- **1XX:** Normalmente indican **respuestas informativas**, que normalmente indican al Cliente cual es el siguiente paso a seguir para continuar la interacción con el Servidor.
- **2XX:** Indican **respuesta satisfactoria**, por ejemplo un código de respuesta 200 significa OK, 202 significa petición aceptada y 201 significa Creado, normalmente devuelto después de una petición HTTP del tipo PUT que se ha resuelto satisfactoriamente.
- **3XX:** Enviado como código de respuesta cuando la petición que lo ha generado ha provocado que el navegador muestre una página diferente. Si el Cliente recibe un código de respuesta de **Redirección**, este necesitará tomar una serie de decisiones para manejar la redirección.
- **4XX:** Los código de respuesta que empiezan por el número 4, indican **Error del Cliente**. Normalmente son lanzadas cuando el Cliente comete un error grave o este no está autorizado a realizar dicha petición.
- **5XX:** Representa **error interno del Servidor**. Normalmente, este tipo de respuesta son lanzadas por el Servidor cuándo este ha encontrado alguna situación que no sabe como manejar, o si éste está temporalmente caído.

Véase Tabla B.2 para observar todas las URIs publicadas por la API REST que se ha desarrollado. En dicha tabla se pueden observar las URIs en sí, el tipo de petición HTTP (GET, POST, PUT, DELETE...), y breve explicación de la funcionalidad que dicha petición ha implementado.
Cabe destacar, que de entre todas esas URIs, todas requieren de la cabecera de autenticación, o Authentication Header en inglés. Las únicas operaciones que no

requieren de esta cabecera son las URIs que tienen como función principal, devolver el token que conformaría dicha cabecera para poder acceder al resto de URIs.

Para acceder a cualquiera de las URIs, hay que especificar el siguiente inicio de ruta: https://[IP_SERVER]/[Operación]

Dónde en IP_SERVER se debe indicar la dirección IP que se ha configurado cuando la instalación, y en Operación se indica el resto de la ruta indicada en la tabla B.2 como URI.

En Tabla B.2, todos los elementos de la URI que vienen precedidos por :, indican un parámetro necesario para que la petición no resulte en error 4XX, o error del Cliente.

URI	Tipo	Descripción
/picture/:id	GET	Para mostrar en el navegador la foto con nombre [:id].
/pictureDownload/:id	GET	Indicando el nombre de fichero correcto, inicia el diálogo para la descarga de dicho fichero.
/getFile/:dev/:id	GET	Indicando como parámetros el nombre del dispositivo y el nombre de fichero, este es devuelto. Para filtrar por dispositivo si se desea.
/newDev/:name/:funct	POST	Si el token existe y es correcto, se registrará un nuevo dispositivo (:dev) en la aplicación, con funcionalidad :funct
/updateDev/:id/:name/:funct	PUT	Modifica la información de un dispositivo por su :id.

/getAllDevs	GET	Devuelve todos los dispositivos registrados en la base de datos. Obviamente si el token existe y es correcto.
/logIn/:user/:psswd	GET	Si el usuario (:user) y contraseña (:psswd) son correctos, devuelve el token necesario para acceder al resto de URIs. No requiere de token como cabecera de autenticación
/getAllUsers	GET	Devuelve un documento JSON con la lista de todos los usuarios registrados en la aplicación.
/getAllConfigFiles	GET	Devuelve un documento JSON con la lista conteniendo todos los ficheros de configuración registrados en la aplicación.
/uploadDevConfiguration	POST	Esta petición no tiene parámetros en la URI, ya que van en el cuerpo de la petición (body). Esto es así, porque tiene múltiples parámetros que son enviados por la entrega de un formulario HTML (submit) desde la interfaz de usuario.
/logIn	POST	URI utilizada cuando el usuario introduce su nombre y contraseña desde la interfaz de usuario. Estos parámetros son indicados en el cuerpo de la petición, y no en la URI en sí.

/logIn/:usr/:psswd	GET	Realiza la misma función que la anterior, pero ésta es llamada con GET, cuando se realiza la petición desde código o POSTMAN.

Tabla B.2. Descripción de las URIs publicadas por la API REST.

A continuación, una serie de capturas de pantalla de como interactuar con el Servicio IoT, ya sea con POSTMAN o a través de la interfaz de usuario proporcionada.

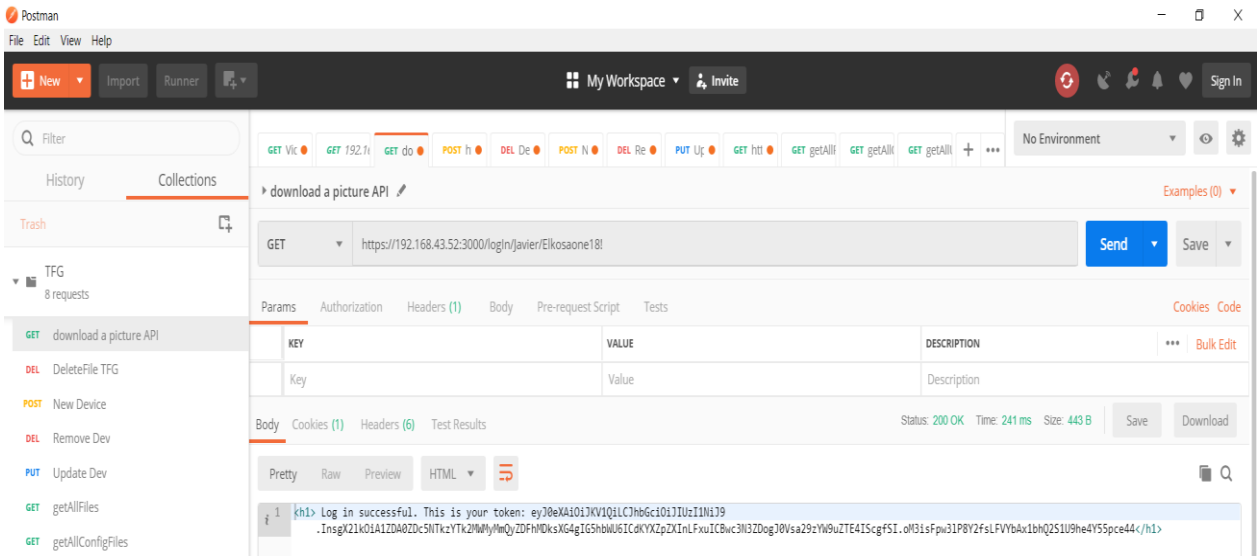

Figura B.3. Ejemplo de como obtener el token de seguridad utilizando POSTMAN. URI logIn con método GET. Los parámetros necesarios se indican en la URI.

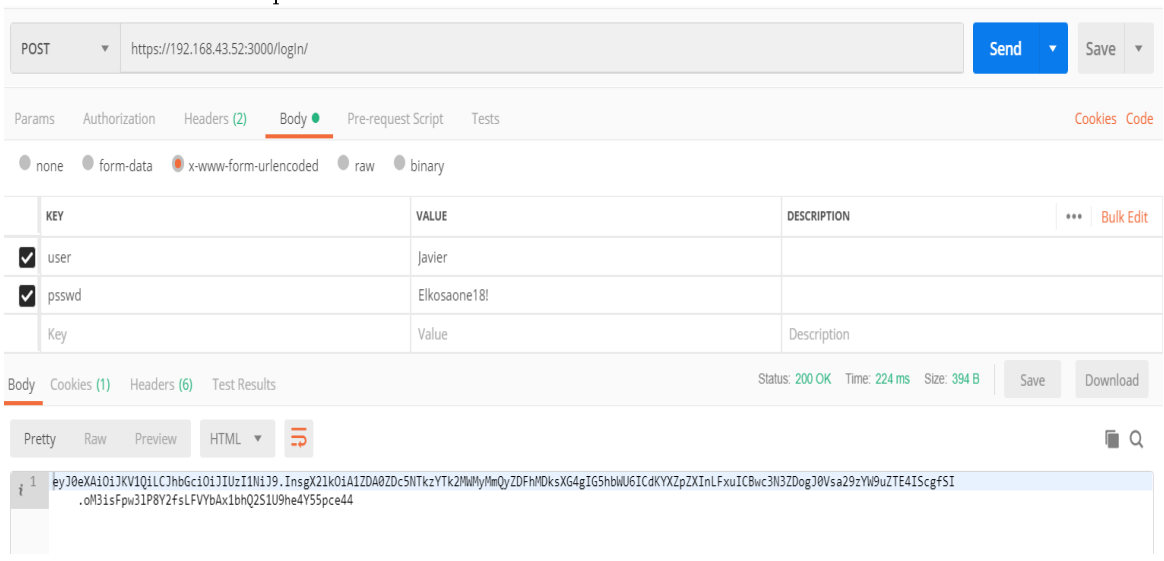

Figura B.4. Obtener token de seguridad a través de la URI /logIn, pero con método POST. Los parámetros se indican ahora en el cuerpo de la petición.

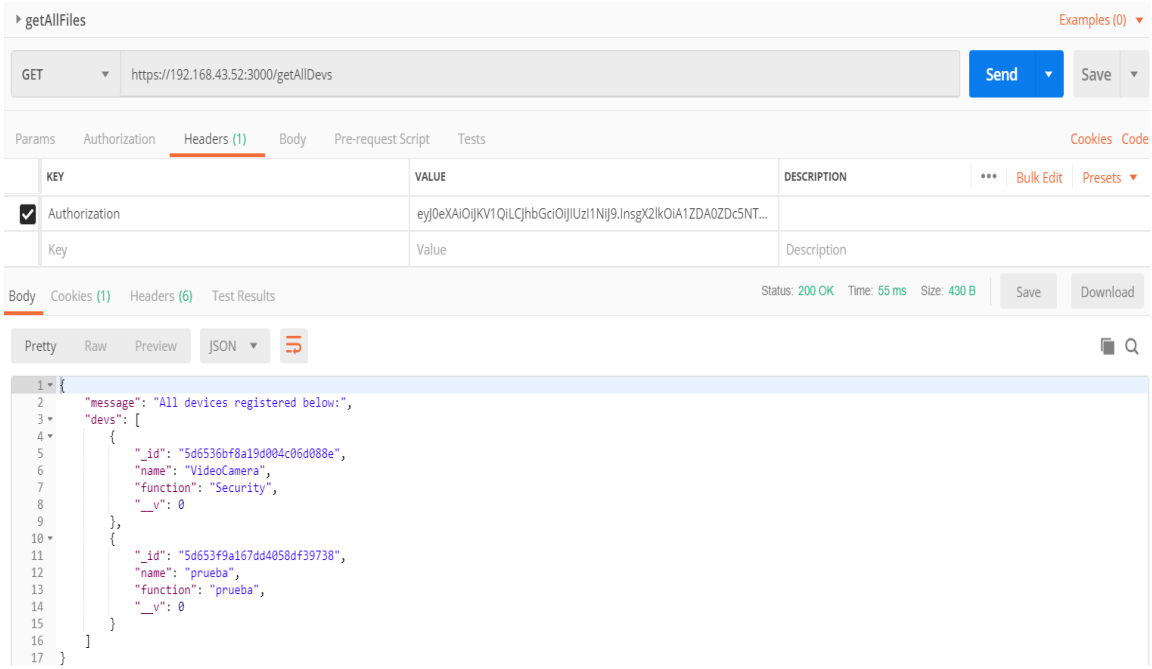

Figura B.5. Petición GET a la URI /getAllDevs, que devuelve todos los dispositivos registrados en una lista JSON. Obsérvese cómo se indica el token de seguridad.

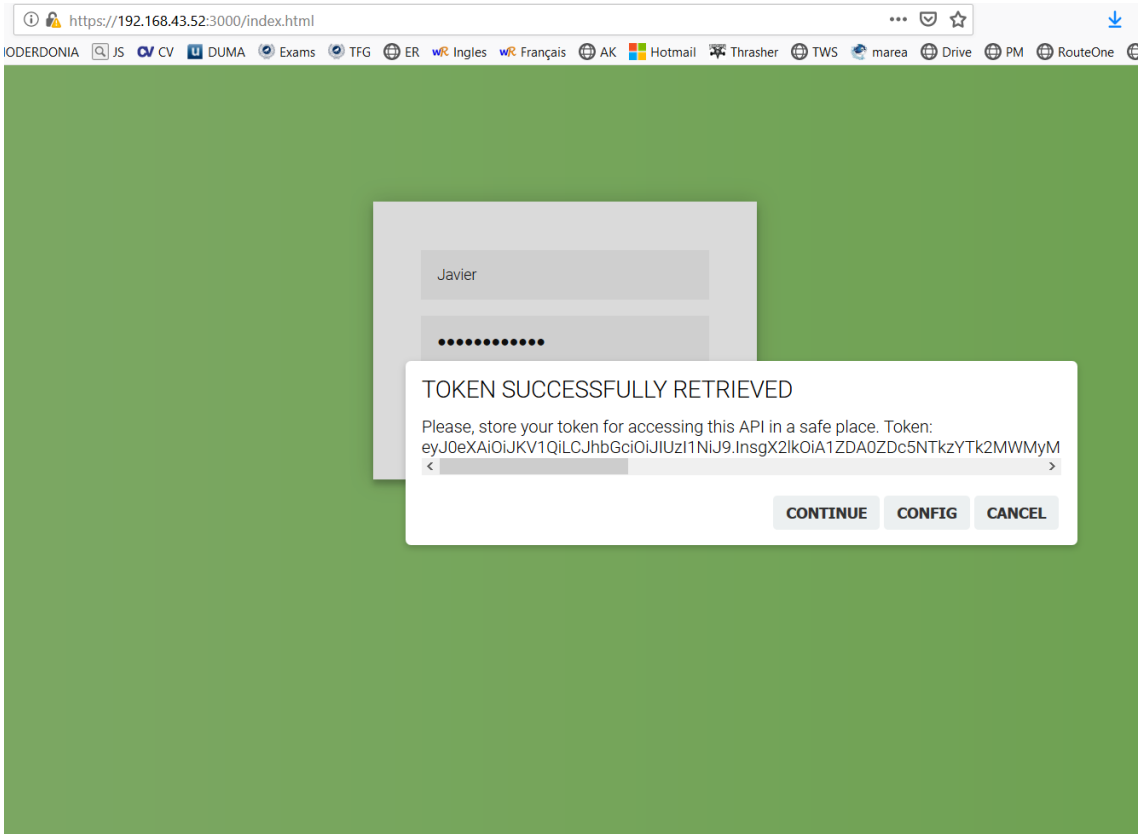

Figura B.6. Obtención del token de seguridad a través de la Interfaz de usuario.

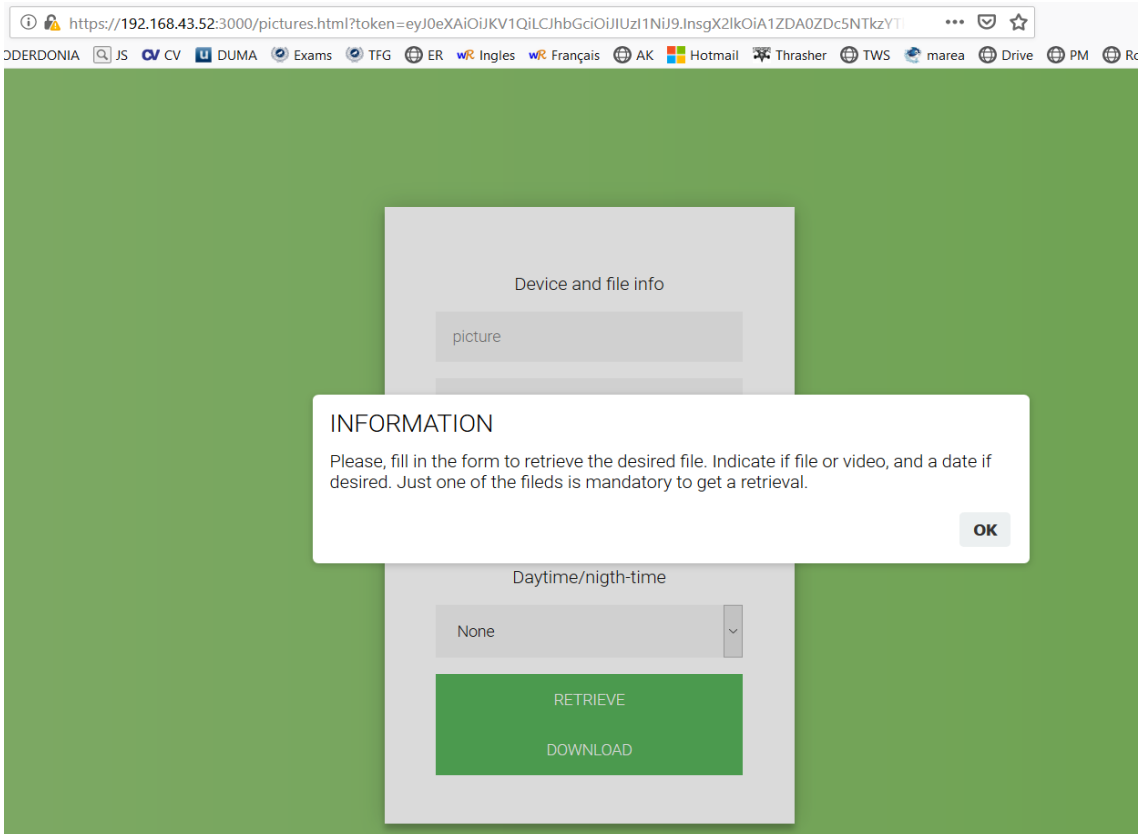

Figura B.7. Interfaz de usuario web que requiere de token de seguridad. Obsérvese como se indica el token en la URL.

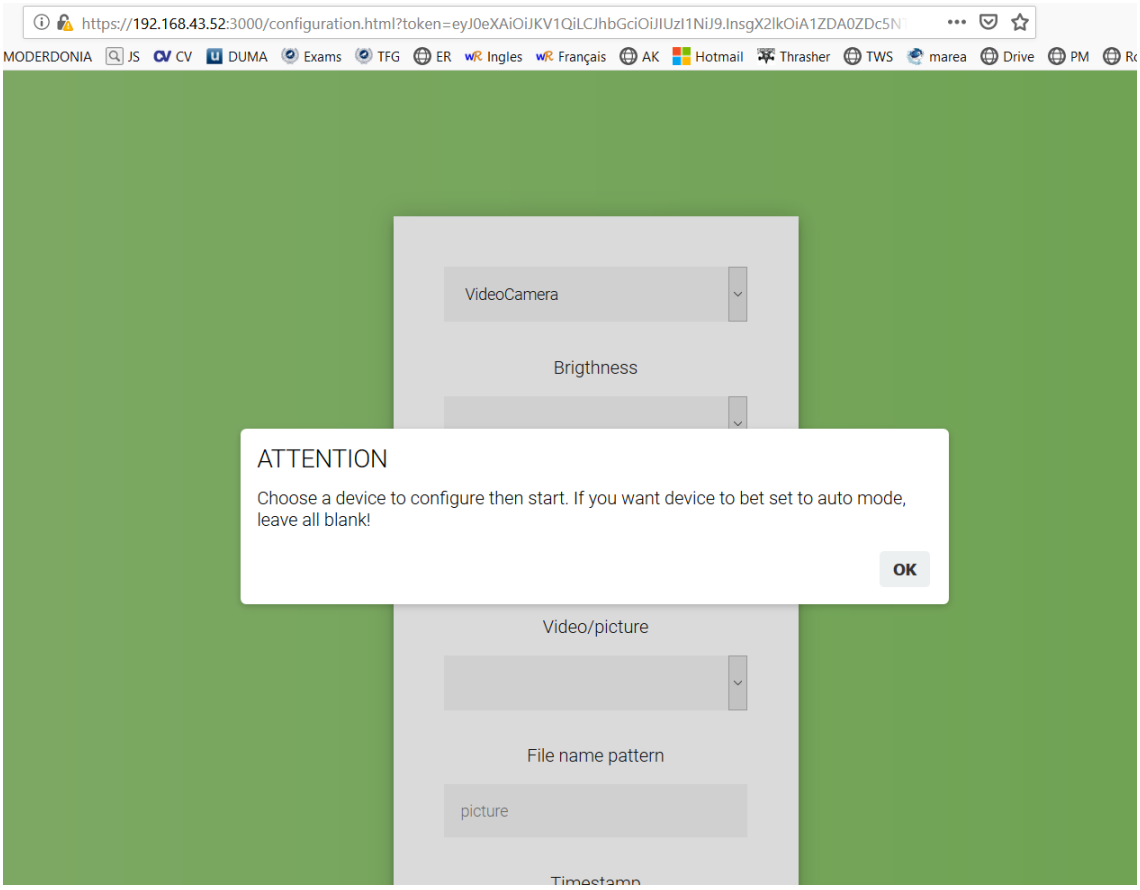

Figura B.8. Página en la que el usuario puede genera un fichero de configuración en el Servidor.

Finalmente, destacar que para interactuar con las URIs que publica el Servicio IoT, simplemente hay que indicar los parámetros correctos y sobre todo prestar vital atención al uso del token, ya que sin éste no se puede acceder ninguna URI que no sean las destinadas a la obtención de dicho token.