MÁSTER UNIVERSITARIO DE INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

Creación de red LWM2M para redes de visión y lectores RFID

Trabajo Fin de Máster

Código 31105128 - Ingeniería Software

Alumno: **Víctor Arcos Barraquero**

Director: **Carlos Cerrada Somolinos**

Co-Director: **Ismael Abad Cardiel**

Curso 2014/2015

16/09/2015

16 de septiembre de 2015

MÁSTER UNIVERSITARIO DE INVESTIGACIÓN EN INGENIERÍA DE SOFTWARE Y SISTEMAS INFORMÁTICOS

Itinerario del Trabajo: Ingeniería Software

Código de la asignatura: 31105128

Título del Trabajo: Creación de red LWM2M para redes de visión y lectores RFID

Tipo de Trabajo: Tipo A

Nombre del estudiante: Víctor Arcos Barraquero

Nombre del Director: Carlos Cerrada Somolinos

Nombre del Co-Director: Ismael Abad Cardiel

PÁGINA INTENCIONALMENTE EN BLANCO

IMPRESO TFDM05_AUTOR
AUTORIZACIÓN DE PUBLICACIÓN
CON FINES ACADÉMICOS

UNED ETS de Ingeniería Informática

Impreso TFdM05_Autor. Autorización de publicación

y difusión del TFdM para fines académicos

Autorización

Autorizo/amos a la Universidad Nacional de Educación a Distancia a difundir y utilizar, con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firma del/los Autor/es

Fdo. Víctor Arcos Barraquero

Juan del Rosal, 16
28040, Madrid

Tel: 91 398 89 10
Fax: 91 398 89 09

www.issi.uned.es

Resumen

El presente trabajo recoge el desarrollo de una red cliente servidor basada en el estándar OMA LWM2M. Dicho sistema será empleado para habilitar una conexión remota con dispositivos de visión y lectores RFID, permitiendo el acceso a imágenes y la información proporcionada por etiquetas RFID de una manera remota.

La creación de dicho sistema tiene como objeto final el procesamiento de información de un espacio inhabitado para su posterior modelado 3D. El uso de redes de cámaras reduce la complejidad del procesamiento de las imágenes. Por otro lado, la información recibida de sensores RFID complementa la información recibida de las imágenes con información relacionada con los objetos de la escena (textura, dimensiones, etc.).

El trabajo aquí propuesto sirve como punto de arranque a dicha concepción estableciendo las bases de un sistema basado en URL que permite el intercambio de información remota e independiente del hardware empleado.

Abstract

This paper presents the development of a Client Server network based on the OMA LWM2M Standard. The system here presented will be employed for enabling a remote connection with vision devices and RFID readers, allowing remote access to images and information provided by RFID tags.

The creation of such a system has as final objective the information processing of uninhabited interiors for its later 3D modeling. Using of camera networks reduces the complexity of image processing. Furthermore, the information received from RFID sensors complements the information received from the images with information related to objects in the scene (texture, size, etc.).

The work proposed herein serves as a starting point to this conception laying the foundations of a URL-based system that allows the exchange of information remotely and independently of the hardware employed.

Palabras Clave

LWM2M, OMA, Cliente, Servidor, Visión, RFID.

Contenido

Índice de Figuras

1. Introducción

IoT, también conocido como Internet de las cosas, es un concepto que recoge la idea de que todos los objetos que nos rodean son accesibles o se encuentran conectados a Internet, de tal manera que en el futuro Internet no sólo fuera una red de intercambio de información entre personas, sino también entre objetos.

El crecimiento de este tipo de tecnología traerá consigo la creación de sistemas muy dispares entre sí, pero con medios de comunicación comunes (Internet), de ahí que la estandarización juegue un papel crucial en la coexistencia de todas estas nuevas tecnologías aún por llegar.

La estandarización en el mundo de IoT es importante no sólo por problemas de seguridad entre sistemas, sino que dicha estandarización posibilitará la creación de sistemas abiertos para que futuros sistemas puedan interactuar con los existentes, permitiendo una integración continua y escalable.

OMA (Open Mobile Alliance) [2] se postula como el punto focal en la industria para la estandarización de los servicios de los dispositivos móviles, proporcionando un conjunto de estándares que permiten unificar las comunicaciones a través de dispositivos móviles.

Hablar de IoT es hablar también de M2M (Machine-to-Machine), este concepto de máquina a máquina engloba a todas aquellas relaciones establecidas directamente entre dispositivos sin estar envuelta persona alguna. Actualmente este tipo de relaciones se dan aunque se espera que en los próximos años este tipo de mercado crezca significativamente, tal y como se puede contemplar en [1] .

El habilitador OMA Lightweight M2M (LWM2M) está dirigido hacia dispositivos con poca capacidad, ya sea RAM, Flash, de bajo consumo que requiere poco uso de ancho de banda. Por otro lado, es igual de adecuado para otros dispositivos que requieran de una comunicación eficiente. LWM2M proporciona una comunicación segura y compacta junto con un modelo de datos eficiente que permiten habilitar la gestión del dispositivo y del servicio para dispositivos M2M.

En el presente trabajo hace uso de IoT para la construcción de una red de cámaras y lectores RFID cuyo propósito final es la reconstrucción virtual de espacios inhabitados [4]. El uso de tecnologías RFID como soporte a técnicas de reconocimiento de imágenes acelera el proceso de reconstrucción de dichos espacios y permite una reconstrucción más precisa.

La dependencia de numerosos dispositivos impuesta por la aplicación final, junto con la complejidad de los cálculos realizados llevan la necesidad crear una infraestructura que permita el control remoto de dispositivos de visión y lectores RFID. El uso de tecnologías IoT permite el acceso remoto de las aplicaciones a los recursos, permitiendo que varias aplicaciones puedan trabajar de manera simultánea con los dispositivos pertenecientes a la red.

El desarrollo aquí propuesto presenta la creación de una red que permite la gestión de dispositivos de visión junto con lectores RFID. Dicho sistema seguirá la especificación LWM2M de OMA que permitirá aislar la tecnología subyacente del nivel de aplicación. LWM2M posibilita la gestión remota de dispositivos, lo cual permite que la aplicación acceda de manera remota a estos dispositivos.

2. Ámbito del proyecto y objetivos

Con el crecimiento de la tecnología en el mundo de IoT, son cada vez más numerosas y complejas las aplicaciones que van apareciendo. Las aplicaciones que tradicionalmente se han venido desarrollando en el ámbito de IoT se caracterizan por ser fuertemente dependientes de la tecnología o de la implementación escogida. Esto hace que sea cada vez más necesario establecer estándares que permitan el desarrollo de trabajo reutilizable y que por tanto permita extender el ámbito de aplicación en el futuro.

OMA LWM2M surge como respuesta a esta necesidad postulando un estándar para la gestión de dispositivos M2M, cuya aceptación se va haciendo cada vez más notable. Esto se puede ver en la cantidad de empresas fabricantes de pequeños dispositivos y circuitos integrados como ARM, Vodafone, HOP Ubiquitous, etc., que han abrazado este estándar al igual que otro tipo de proyectos que pretenden convertirse en herramientas de referencia tal y como Leshan [11], proyecto de IoT de Eclipse o Wakaama [15], también de Eclipse.

La aplicación final a la que se destina este trabajo es a la creación de una red de dispositivos que permitan la reconstrucción de espacios inhabitados [4]. Con tal objetivo, se propone a lo largo de este trabajo el uso de estándares de tal manera que la solución propuesta permita crear una infraestructura abierta a diversas tecnologías y por tanto adaptable a dichas necesidades.

La implementación de dicha red está basada en Leshan [11]. Dicha herramienta ha sido elegida por ofrecer un ejemplo de cliente y servidor LWM2M con la funcionalidad completa para realizar las operaciones básicas de una red LWM2M. El estándar LWM2M es recientemente nuevo por tanto, no son muchas las herramientas disponibles. Leshan es una de las pocas herramientas existentes que permiten testear y trabajar con el estándar LWM2M [22].

Los objetivos del sistema basado en LWM2M aquí propuesto son:

- Crear un sistema cliente-servidor basado en la especificación LWM2M de OMA. Dicho sistema registrará los dispositivos que han sido inicializados siguiendo dicha especificación. Dicho sistema estará basado en el proyecto Leshan, de Eclipse.

- Permitir la comunicación remota con dispositivos de visión. Esto implica especificar la comunicación con un dispositivo de visión y el acceso remoto a dicha información.

- Permitir la comunicación remota con lectores RFID. Al igual que los dispositivos de visión, es necesario establecer la comunicación con dichos dispositivos y el acceso remoto a sus recursos.

- Crear una aplicación que interactúe con el servidor LWM2M basada en MATLAB. Dicha aplicación mostrará el acceso a los recursos expuestos por los clientes registrados y el almacenamiento de dicha información.

El uso de MATLAB como entorno de una aplicación diseñada para LWM2M será el punto de partida para el posterior procesamiento de los datos obtenidos por los dispositivos de la red. Dicho entorno fue empleado en [4] para la reconstrucción de espacios inhabitados. Además, el uso de un entorno independiente al empleado para LWM2M puede servir como ejemplo del potencial en el uso de estándares como LWM2M que permiten la independencia en el uso de distintos tipos de plataformas.

3. Estado del arte

El desarrollo expuesto en el presente trabajo surge como continuación con la línea de investigación de reconstrucción virtual de espacios inhabitados mediante técnicas de visión artificial [4]. En [4] se propone la reconstrucción de un espacio interior por medio de un sensor de barrido laser y tecnología RFID. La tecnología RFID permite reducir los tiempos de computación y procesamiento aportando datos valiosos con información del entorno.

Siguiendo la línea de investigación en la reconstrucción virtual de espacios no habitados, se propone que el único sensor de barrido junto con el único sensor RFID empleado sea sustituido por un parque de cámaras (3D y/o 2D) y lectores RFID. El presente trabajo centra los esfuerzos en crear una infraestructura para permitir a las cámaras y lectores RFID transmitir la información capturada de manera que pueda ser gestionada fácilmente.

La comunicación con dispositivos de distinta naturaleza y fabricante puede suponer un problema, problema que no añade valor a la aplicación. De ahí surge la necesidad de crear software para abstraernos de la interacción con un dispositivo específico. En este sentido son numerosas las propuestas que han sido realizadas para la creación de middleware que permita dicha abstracción, se pueden ver ejemplos en [5] y [6].

El desarrollo aquí expuesto pretende mostrar una forma de gestionar estos dispositivos sin necesidad de conocer cómo interactuar con el hardware. Entre las soluciones proporcionadas en [5] y [6], ambas tienen en común que el middleware proporcionado corresponde a una solución a la que han llegado los autores. El problema de este tipo de soluciones es que no convergen a un punto en común, por tanto a pesar de que se elimina la dependencia con los dispositivos, se crea una dependencia con el middleware. Por tanto, es importante alcanzar soluciones comunes para el desarrollo de este tipo de software, basar el desarrollo en estándares.

Así pues, la solución escogida es la implementación de un sistema basado en OMA Lightweight M2M (LWM2M). La elección de dicho estándar se produce por ser un estándar que surge de la necesidad de implementar un método común para gestionar pequeños dispositivos (con baja capacidad de procesamiento) conectados entre sí que además se beneficia de un gran respaldo al ser impulsado por una organización como es OMA.

En contraste con OMA Device Management (OMA DM) que se centra principalmente en dispositivos móviles, el habilitador OMA LWM2M se centra no solamente en la gestión, sino en habilitar servicios para dispositivos LWM2M [8]. Los dispositivos que son objetivo para este habilitador son aquellos que tienen pocos recursos para realizar procesamientos pesados. Por tanto, este habilitador LWM2M de OMA ofrece un protocolo ligero y compacto con una estructura de datos plana.

3.1. OMA LWM2M

LWM2M de OMA define el protocolo de comunicación de la capa de aplicación entre un servidor y un cliente. El servidor se localiza típicamente en un centro de datos público o privado y puede ser albergado por el proveedor de servicios M2M, el proveedor de servicios de red o el proveedor de servicios de aplicación. Por otro lado, el cliente LWM2M reside en el propio dispositivo y se integra típicamente como una librería o una función embebida del módulo o dispositivo [3].

Se definen cuatro interfaces entre el servidor y el cliente [7]:

- Bootstrap: recoge la funcionalidad necesaria para el arranque y la configuración del dispositivo.
- Device Discovery and Registration: inicialización y log in del dispositivo.
- Device Management and Service Enablement: gestión del dispositivo y de la red.
- Information Reporting: reporte del estatus del dispositivo.

La Figura 1 muestra la arquitectura del estándar LWM2M. En ella se ven las interfaces definidas arriba. Los interfaces LWM2M se describirán en detalle en la sección 3.1.2.

El habilitador LWM2M emplea CoAP (*Constrained Application* Protocol) con enlaces UDP y SMS. La seguridad sobre UDP es proporcionada mediante DTLS (*Datagram Transport Layer Security*) [9]. La pila de protocolo del habilitador LWM2M se muestra en la Figura 2.

La estructura de los clientes LWM2M en objetos y recursos así como los objetos definidos en la especificación se detallan en la sección 3.1.1.

En el marco de especificación de OMA LWM2M se han creado otros objetos aparte de los definidos en [9] (ver sección 3.1.1), OMA define otros objetos LWM2M. Entre estos objetos, es necesario hablar del objeto Device Capability Management (ID = 15) [10] que detallaremos en detalle en la sección 3.1.3 por su importancia en este trabajo.

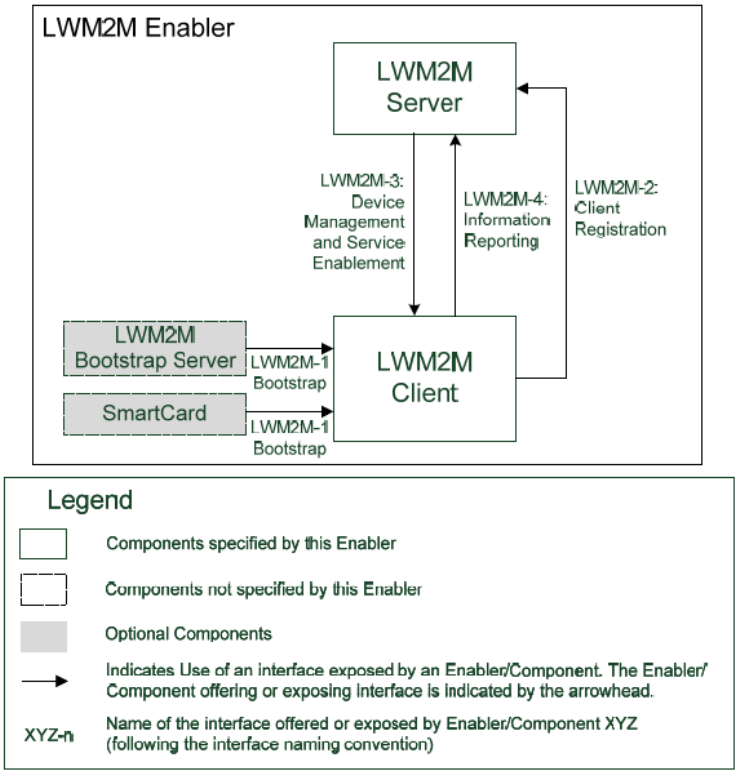

Figura 1. Arquitectura OMA LWM2M [7].

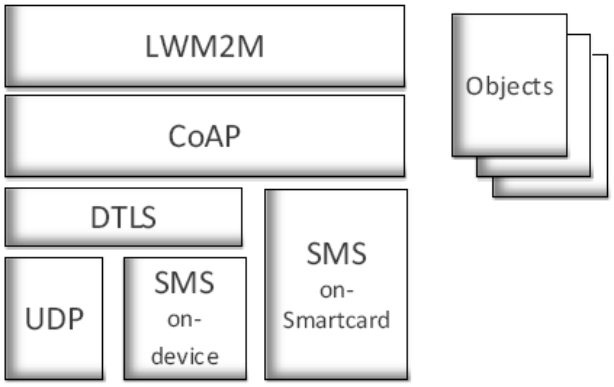

Figura 2. Pila de protocolo del habilitador LWM2M [9].

3.1.1. Cliente LWM2M

El cliente LWM2M se compone de uno o varios objetos y, a su vez, cada objeto se compone de uno o varios recursos, que son los elementos que contienen información correspondiente al cliente.

Los objetos y recursos pueden tener una o varias instancias (dependiendo de si la instanciación está contemplada como atributo del objeto/recurso). Cada objeto tiene un único identificador y cada recurso, igualmente, tiene un único identificador dentro de su objeto correspondiente.

La Figura 3 muestra la estructura de la que está compuesta un cliente LWM2M en objeto y recursos.

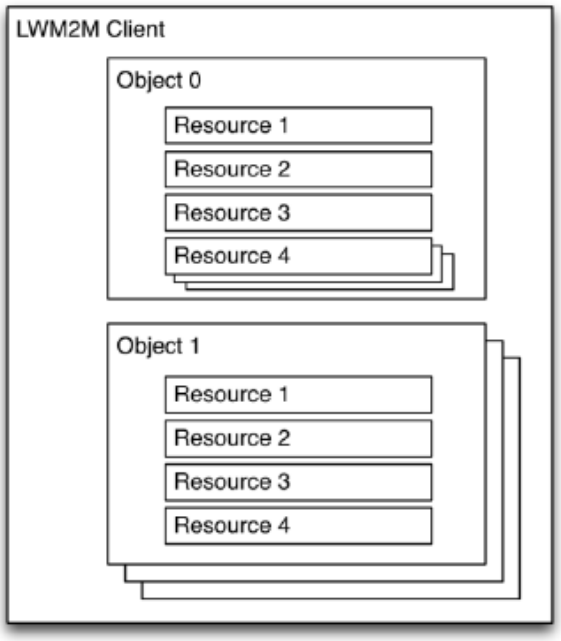

Figura 3. Relación entre cliente LWM2M, Objeto y Recursos [9].

Los objetos especificados en la primera revisión del estándar OMA LWM2M se definen en la especificación técnica [9] en detalle. Estos son:

- LWM2M Security (ID = 0): contiene los recursos para gestionar la seguridad en la comunicación entre servidor y cliente M2M.
- LWM2M Server (ID = 1): contiene los recursos para definir datos y funciones relacionadas con la gestión de servidores.
- Access Control (ID = 2): para definir los tipos de derechos de accesos que el servidor tiene sobre cada objeto del cliente.
- Device (ID = 3): para detallar información específica del dispositivo.
- Firmware (ID = 4): para actualizaciones de firmware.
- Location (ID = 5): para conocer la localización del dispositivo.
- Connectivity Monitoring (ID = 6): para monitorizar el estado de la conexión.
- Connection Statistics (ID = 7): para obtener información estadística de la conexión.

3.1.2. Interfaces LWM2M

Como vimos anteriormente, las cuatro interfaces definidas por LWM2M son:

- Bootstrap.
- Client Registration.
- Device Management and Service Enablement.
- Information Reporting.

El interfaz “Bootstrap” se compone de las siguientes operaciones:

- Uplink o Link Cliente – Servidor: petición de Bootstrap (*Request Bootstrap*).
- Downlink o Link Servidor – Cliente: peticiones Borrado (*Delete*) y Escritura (*Write*).

Esta operación se usa para inicializar los objetos necesarios del cliente LWM2M con uno o más de un servidor LWM2M. Esta operación es también definida para el uso de carga desde Flash o por Smartcard [9]. La Figura 4 muestra las operaciones realizadas con esta interfaz.

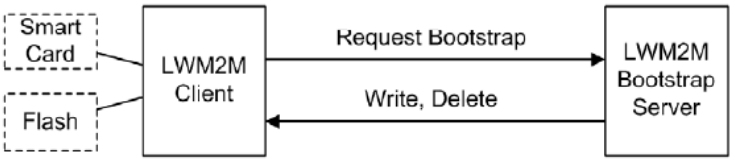

Figura 4. Interfaz Bootstrap [9].

La interfaz "Client Registration" se compone de las siguientes operaciones uplink: Registrar, Actualizar y De-registrar [9]. La Figura 5 muestra dichas operaciones.

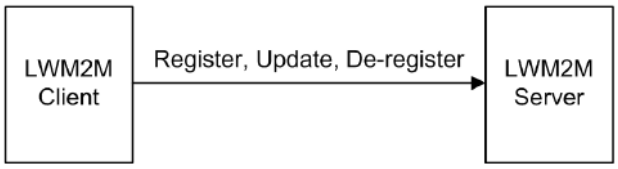

Figura 5. Interfaz Client Registration [9].

Las operaciones del interfaz "Device Management and Service Enablement" son downlink. Estas son: Leer, Crear, Borrar, Escribir, Ejecutar, Escribir Atributos y Descubrir. Estas operaciones sirven para interactuar con recursos, instancias de recursos, objetos e instancias de objetos [9]. La muestra las operaciones realizadas con esta interfaz.

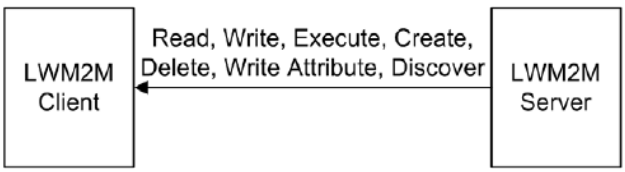

Figura 6. Interfaz Device Management y Service Enablement [9].

Las operaciones de la interfaz "Information Reporting". Las operaciones downlink son: Observar y Cancelar Observación. La operación uplink es Notificar [9].

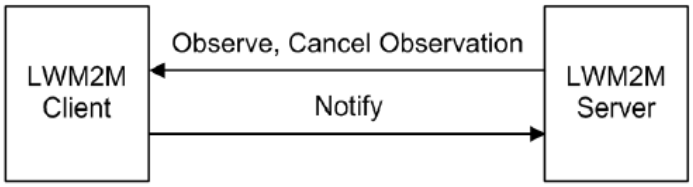

Figura 7. Interfaz Information Reporting [9].

3.1.3. Device Capability Management

El objeto Device Capability Managemet [10] está orientado para especificar los mecanismos requeridos para la gestión remota de las capacidades del dispositivo, no para sólo habilitar o deshabilitar remotamente las capacidades del dispositivo, sino también para exponer al

servidor LWM2M las capacidades del hardware retirable cuando están enlazados al dispositivo.

Las siguientes funcionalidades de capacidad de gestión básicas del dispositivo son consideradas en este objeto:

1. Habilitar/Deshabilitar una capacidad de un dispositivo LWM2M dado.
2. Exponer al servidor las capacidades del hardware del dispositivo LWM2M.

Cualquier capacidad de un dispositivo LWM2M debe referir a una de estas capacidades [10]:

- **0: SENSOR** dedicado a adquirir datos en eventos físicos.
- **1: CONTROL** dedicado a ajustar varios niveles.
- **2: CONNECTIVITY** dedicado a proporcionar capacidad de comunicación.
- **3: NAVIGATION** dedicado a proporcionar capacidad de geo localización.
- **4: STORAGE** dedicado a proporcionar capacidad de memoria.
- **5: VISION** dedicado a proporcionar capacidad de visión: capacidad de video, capacidad de foto.
- **6: SOUND** dedicado a proporcionar capacidad de sonido: buzzer, altavoz.
- **7: ANALOG INPUT** se refiere a una capacidad de entrada analógica genérica.
- **8: ANALOG OUTPUT** se refiere a una capacidad de salida analógica genérica.
- **9-15: reservados.**

En el presente trabajo, cabe destacar la capacidad de visión la cual servirá para identificar a las cámaras de entre todos los dispositivos de la red creada. Además, la capacidad de conectividad puede ser empleada para dispositivos lectores RFID. Por otro lado, al implementar dicho objeto como parte del cliente LWM2M, éstos podrán ser controlados desde la aplicación.

3.2. Redes de visión

Las redes de visión están emergiendo como herramientas valiosas para aplicaciones de seguridad en entornos tan diversos como residencias, estaciones de metro, autopistas, etc. Por redes de visión se entiende un conjunto de dispositivos de visión (cámaras, videocámaras, sensores de rango, etc.) que se encuentran integrados en una red.

Donde antes sólo se empleaba un único procesador para el procesamiento de las imágenes, las redes multicámaras modernas están compuestas de varias cámaras distribuidas espacialmente cada una de las cuales pueden tener su propio procesador, e incluso, su propia fuente de alimentación [20].

Los sistemas sensoriales convencionales basados en visión (cámaras, sensores de rango, sensores láser, sistemas basados en luz estructurada, etc.) proporcionan únicamente información de una vista de la escena, esto tiene como consecuencia la aparición de "defectos" en la información recibida. Se entiende por estos "defectos" que la información recibida no es suficiente. Debido a esto, el trabajo llevado a cabo para reconstruir con precisión el modelo 3D completo de un objeto es una ardua tarea en la que confluyen diversas

sub-líneas de investigación: digitalización con oclusiones, selección de puntos de vista, correspondencia y registro de superficies, tratamiento y acoplamiento de texturas, detección y rellenado de huecos, etc.

Una red de visión o un parque de dispositivos de visión integrado pueden proporcionar la información visual necesaria para ayudar en el procesado de la escena gracias a los distintos puntos de vista que dicho sistema puede ofrecer. De esta manera, un sistema compuesto por varias cámaras puede agilizar el procesamiento y reducir el tiempo de computación frente al procesamiento realizado por un sistema compuesto por una única cámara. La reducción de dicho tiempo de procesamiento es consecuencia de la reducción del número de oclusiones, el disponer de distintos puntos de vista, etc. que evita el tener que realizar un procesamiento innecesario.

El posicionamiento de los objetos de una escena en un sistema de coordenadas es posible conociendo los parámetros internos y externos de las cámaras. Esto permite cartografiar los objetos en el espacio desde distintos puntos de vista. Con este sistema se puede generar un modelo 3D del entorno integrando las imágenes de la red de cámaras, objetivo compartido en [17]. En [17] se expone el seguimiento de personas a través de un sistema de cámaras de vigilancia, como motivación para este desarrollo expone la creación de un modelo 3D consistente que abstraiga la gestión de dichas cámaras.

El uso de numerosas cámaras en una red puede plantear dificultades a la hora de procesar las imágenes de toda la red, problemas de potencia computacional. Por tanto, surge un problema de escalabilidad en estos sistemas.

Ante el problema de escalabilidad en redes de sistemas de visión, han surgido varias alternativas como es el uso de los nodos o cámaras como elementos inteligentes que ofrecen un pre procesamiento del entorno, tales como los ejemplos que pueden observarse en [23], [24] y [17]. Esto alivia la carga de procesamiento realizada por el computador principal de la red encargado de procesar la información recibida por los nodos.

Las *smart cameras* o cámaras inteligentes, son dispositivos de visión que son capaces de extraer información específica para una aplicación mediante el procesamiento de la imagen capturada. Se tratan de sistemas de visión auto contenido que están equipados con una infraestructura de alto rendimiento en comunicación y procesamiento combinando monitorización de video, procesamiento y comunicación en un único dispositivo [24].

El uso de cámaras inteligentes para el pre procesamiento de imágenes alivia el problema de escalabilidad permitiendo crear redes de visión mayores. Sin embargo, sigue existiendo el problema de escalabilidad al pretender monitorizar áreas cada vez mayores dentro de la misma red de visión. Dicho problema hace que comiencen a plantearse distintos paradigmas en el procesamiento de las imágenes adquiridas por los sistemas de visión, de tal manera que permitan reducir la cantidad de información a ser transmitida al igual que permita reconstruir la realidad con la calidad adecuada. Un ejemplo de esta aproximación es el reflejado en [21], en el cual se propone una reducción de la información transmitida sin afectar a la calidad de la imagen reconstruida tras procesar las imágenes del sistema.

El modelo propuesto en [21] permite una reducción de la frecuencia de muestreo, reduciendo el tráfico en la red y permite, en el caso de redes de visión, generar imágenes de gran calidad mediante el solapamiento de campos de visión. El uso de este tipo de técnicas, aplicadas a la reconstrucción de espacios inhabitados, permite generar imágenes de gran calidad que reducirán el procesamiento requerido debido a defectos en la imagen como huecos, cambio de texturas y oclusiones.

Con el mismo propósito, el de compensar y obtener imágenes con más detalle así como el de estimar la posición de los objetos y realizar un seguimiento de éstos en la escena, se mencionan otro tipo de algoritmos en [20]. Dichos algoritmos son los algoritmos de consenso (*consensus algorithms*) y de coordinación (*coordination algorithms*). Los algoritmos del tipo *consensus* son apropiados para solucionar problemas de estimación, como, por ejemplo, la estimación de la posición de los objetos en la escena [20]. Por otro lado, los algoritmos del tipo *coordination* son más apropiados para aplicaciones en las que se requiera seguir el movimiento de seres u objetos a lo largo de una escena [20].

El uso de redes de visión favorece a los sistemas de procesamiento de imágenes permitiendo la obtención de mejores resultados con menor esfuerzo computacional.

3.3. Redes RFID

La meta de IoT es crear una red que habilite a los objetos a estar conectados en cualquier momento, en cualquier lugar, con cualquiera que use cualquier red. Para cualquier servicio en este mundo, los objetos a nuestro alrededor conocen qué nos gusta, dónde estamos y qué necesitamos sin necesidad de instrucciones directas [18].

La visión inicial de IoT fue crear una red donde todas las “Cosas” son etiquetadas e identificadas por transpondedores RFID. RFID es una tecnología emergente que se está convirtiendo en una herramienta importante para IoT. Debido a que RFID permite rastreo de una gran cantidad de objetos unívocamente identificados, esta tecnología es aplicada como un habilitador crítico de IoT [18].

Radio Frequency Identification o RFID es un método que usa de la radio frecuencia para trasferir información, con el propósito de identificación y el seguimiento de *tags* o etiquetas unidas a cosas [18]. Esto, a diferencia de los sistemas tradicionales de identificación como, por ejemplo los lectores de códigos de barras, no requiere que el objeto a identificar se encuentre en el campo de visión de un objeto, ni que dicho objeto sea dispuesto de una manera concreta [25].

La tecnología RFID puede ser pasiva o activa, dependiendo de las etiquetas empleadas. La diferencia estriba en que las etiquetas activas necesitan de alimentación, como baterías, mientras que las etiquetas pasivas no. Entre estas dos topologías se encuentran también las etiquetas semi-activas que como su propio nombre indican, son alimentadas parcialmente para poder gestionar sensores u otros dispositivos, como memorias.

Las etiquetas RFID activas suelen operar a frecuencias de 455 MHz, 2.45 GHz, o 5.8 GHz teniendo un alcance de 20 a 100m [25]. Sin embargo, las etiquetas RFID pasivas operan a menores frecuencias y se distinguen entre baja frecuencia (124 kHz, 125 kHz o 135 kHz), alta frecuencia (13.56 MHz) y frecuencia ultra alta o UHF (de 860 MHz a 960 MHz). El alcance de las etiquetas pasivas es bastante menor al de las etiquetas activas estando comprendido entre unos pocos centímetros hasta unos 10m [25]. Por otro lado, el hecho de que la tecnología RFID pasiva sea la más empleada se basa en los casi nulos costes de mantenimiento y los costes de fabricación de dichas etiquetas, los cuales son significativamente menores a los de las activas.

La tecnología RFID ha evolucionado enormemente en las últimas décadas, permitiendo sistemas más rápidos, más pequeños, menos costosos, etc. Cada vez es más común encontrarse este tipo de tecnología en nuestro día a día. Las mejoras en esta tecnología han llevado al uso de esta tecnología hacia nuevas aplicaciones, cada vez más complejas. Estas nuevas aplicaciones junto con un uso cada vez más extendido de esta tecnología han llevado al desarrollo de una nueva generación de RFID, el RFID de segunda generación.

Los sistemas RFID de segunda generación no sólo contienen información estática como descripción e identificación de objetos, sino que están dotados de reglas dinámicas de codificación de etiquetas ofreciendo así los requisitos de servicio hasta la fecha y configuración dinámica de la red RFID [26].

En un sistema de información, siempre se espera que proporcione respuestas para las siguientes cuestiones: Quien, Qué, Cuándo y Dónde. La tecnología RFID proporciona el Quién (Identidad del objeto). Por dicha razón, las actividades de investigación hoy en día están orientadas a investigar en la creación de infraestructuras capaces de crear el Qué, Cuándo y Dónde. Una de estas infraestructuras es hacer que RFID funcione en una red con sensores y redes de sensores [19].

El uso de sistemas RFID es cada vez más frecuente y cada vez son mayores los sistemas que integran RFID gestionando cantidades ingentes de información. Esto hace que la necesidad de crear un middleware para gestionar redes RFID sea cada vez más necesaria en sistemas no triviales y heterogéneos los cuales pueden llegar a gestionar múltiples lectores, instancias de aplicaciones, procesado de datos, semántica de negocio, etc.

El uso de middleware para sistemas RFID se hace indispensable por las siguientes razones [26]:

- La necesidad de filtrar datos para evitar redundancia en éstos o información no necesaria para las aplicaciones de negocio, permitiendo a la vez una optimización de recursos.
- La necesidad de abstraerse de la tecnología empleada en el sistema, sirviendo de interfaz y permitiendo un procesado de datos estandarizado sin recurrir a la lógica de negocio.
- La flexibilidad de integrar sistemas RFID para soportar la auto-identificación en distintas aplicaciones y modelos de proceso.

En [19] se resalta la importancia de permitir que una red pueda abstraerse de los dispositivos que la componen, creando una solución independiente de éstos que permite la integración de

información compleja proporcionada por distintos dispositivos simples, como lectores RFID y sensores inalámbricos. En este trabajo se recogen varios tipos de topologías de red cuyo objetivo es la generación de una infraestructura que pueda gestionar simultáneamente datos de tecnología RFID y sensores inalámbricos. Los tipos de topología expuestos en [19] son:

- Arquitectura de agente de red: el agente de red es una arquitectura en capas en la que las redes RFID y WSN. El agente combina y gestiona de manera conjunta los datos provenientes de la red de sensores inalámbricos y de la red RFID actuando como link.
- Arquitectura de "lector RFID como sensor": agrupa los sensores inalámbricos y los lectores RFID como dispositivos del mismo tipo, interpretando que el lector RFID es un tipo más de sensor.
- Arquitectura "tag RFID como sensor": es similar al anterior. Sin embargo, se usa el concepto de sensor aplicado a las etiquetas.

Igualmente, en [18] se expone una serie de soluciones basadas en redes de sensores y RFID mostrando la importancia de crear igualmente estructuras de datos complejas basadas en la combinación de la información aportada por dispositivos de diversa naturaleza.

4. Arquitectura de la red LWM2M

En esta sección se presenta el desarrollo de la arquitectura del sistema LWM2M para la gestión de dispositivos de visión (cámaras, videocámaras, cámaras 3D, etc.) y de lectores RFID. La implementación de dicha red será realizada a través del proyecto Leshan [11], de Eclipse.

El objetivo del sistema LWM2M es proporcionar a las aplicaciones los recursos proporcionados por los dispositivos para que de esta manera puedan interactuar con ellos de manera remota e independientemente de las particularidades del dispositivo, ya que el protocolo LWM2M estandariza la forma de comunicación con los dispositivos.

La Figura 8 muestra de manera gráfica la arquitectura del sistema. Cómo los dispositivos encapsulados en clientes LWM2M intercambian información con el servidor. A la vez, dicho servidor pone dichos recursos a disposición de las aplicaciones, las cuales pueden conectarse de manera remota.

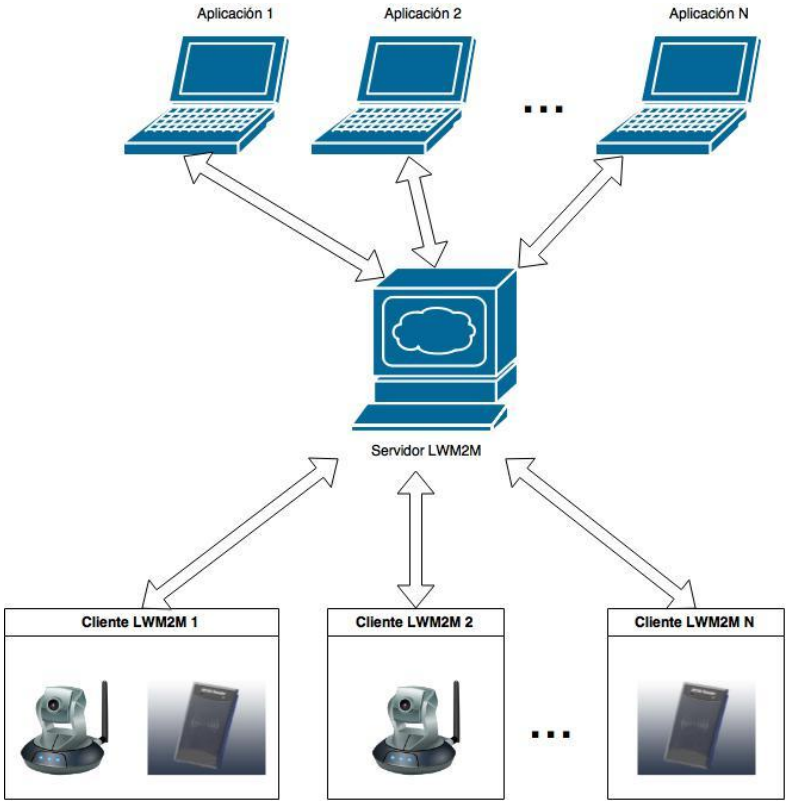

Figura 8. Arquitectura del sistema.

La arquitectura presentada permite una total flexibilidad en la incorporación de nuevos clientes independientemente del tipo de cliente que se trate (dispositivo de visión, lector RFID, una combinación de ambos, etc.), dicha flexibilidad aporta a este proyecto un gran potencial para ser empleado en numerosas aplicaciones.

Los dispositivos son automáticamente registrados por el servidor el cual pone a disposición de las aplicaciones los recursos de dichos clientes de manera que las aplicaciones no tienen exclusividad sobre los dispositivos, sino que son compartidos.

El servidor y los clientes de la red LWM2M están basados en Leshan [11] que es el proyecto sobre el que se ha basado el presente trabajo para proporcionar la infraestructura LWM2M. Dicho proyecto presenta un ejemplo de cliente LWM2M e igualmente, una implementación de servidor LWM2M que permite una rápida implementación del estándar OMA LWM2M sobre el sistema propuesto en el presente trabajo.

Las aplicaciones son basadas en el acceso a recursos compartidos a través de URL, lo cual permite un acceso remoto a los recursos proporcionando una gran flexibilidad en el despliegue de esta plataforma para su aplicación final, que es la reconstrucción de espacios no habitados [4]. Gracias a LWM2M dichas aplicaciones no dependen del hardware subyacente y la manera de comunicación con los dispositivos será invariante, siempre y cuando ofrezcan los mismos servicios.

Debido a la independencia de los dispositivos y de la plataforma empleada para la creación de aplicaciones sobre el servidor LWM2M, es posible el uso de herramientas como MATLAB para tal fin. MATLAB es una potente herramienta que permite no sólo gestionar los recursos expuestos en la red LWM2M, sino que además puede ser a la vez empleado en el tratamiento de dichos recursos para su objetivo final, la reconstrucción virtual de espacios inhabitados. Para tal fin fue utilizado en previos trabajos, como se puede ver en [4]. En la sección 5.3 se expone un ejemplo de implementación de aplicación para LWM2M empleando MATLAB.

4.1. Leshan

Leshan [11] es una implementación de cliente y servidor M2M. Proporciona una completa infraestructura para construir soluciones IoT basadas en OMA LWM2M y en el lenguaje Java. Está compuesto por:

- Una librería de servidor de administración de dispositivos.
- Una librería de cliente de administración de dispositivos.
- Un servidor de administración de dispositivos con una interfaz de usuario web.
- Un servidor de arranque o *bootstrapping server* (servidor a cargo de la configuración inicial de los dispositivos, ver sección 3.1.2).

Este proyecto implementa las cuatro interfaces de LWM2M (sección 3.1.2) que pueden verse a través de las implementaciones ejemplo de servidor (Leshan Client Example) y de servidor (Leshan standalone).

Leshan está implementado sobre el proyecto Eclipse Californium [13] para la implementación CoAP y DTLS.

Leshan está disponible en GitHub [12]. La Figura 9 muestra un ejemplo de la interfaz gráfica del servidor con un cliente LWM2M.

Este proyecto proporciona la infraestructura necesaria para desarrollar el sistema propuesto en este trabajo. Leshan proporciona el servidor LWM2M requerido y la estructura inicial de un

cliente LWM2M, la cual particularizaremos más adelante para servir a los propósitos del sistema propuesto.

Leshan está compuesto de los siguientes módulos:

- Leshan-core : elementos comunes.
- Leshan-server-core : lógica del servidor LWM2M.
- Leshan-server-cf : implementación de servidor basada en Californium [13].
- Leshan-client-core : lógica del cliente LWM2M.
- Leshan-client-cf : implementación de cliente basada en Californium [13].
- Leshan-all : todos los módulos previos en un jar.
- Leshan-client-example : un ejemplo de API cliente.
- Leshan-standalone : un servidor de demostración con una UI web. Ejemplo de la cual puede observarse en la Figura 9.
- Leshan-bs-server : un servidor bootstrap de demostración.
- Leshan-integration-tests : tests automáticos de integración.

En el desarrollo del sistema propuesto se trabajará sobre los ejemplos de cliente (Leshan-client-example) y de servidor (Leshan-standalone) ya que dichos módulos proporcionan las API necesarias para implementar una red LWM2M.

El servidor LWM2M no necesita ser modificado para el sistema propuesto, puesto que la funcionalidad de éste no se ve afectada para el propósito deseado. Si bien es cierto que al igual que el cliente LWM2M, la lista de los objetos LWM2M disponibles por el servidor debe ser actualizada como indica la sección 5.1.1. Sin embargo, el ejemplo de cliente LWM2M proporcionado por el proyecto Leshan sí debe ser modificado para adaptarse a las necesidades del sistema. Dichas modificaciones se recogen en la sección 0.

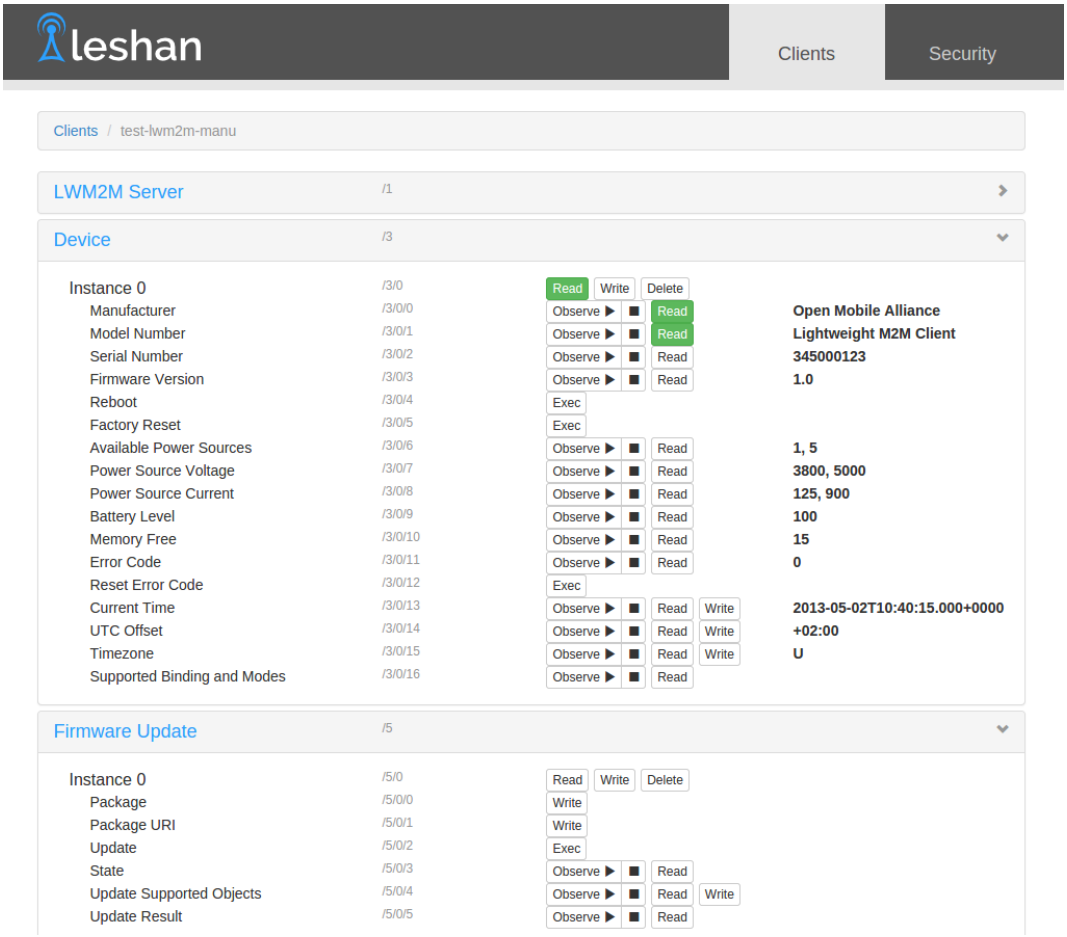

Figura 9. Ejemplo de Servidor Leshan con cliente registrado [12].

4.2. Definición de objetos LWM2M utilizados

En esta sección se definen los objetos OMA LWM2M necesarios para reflejar la funcionalidad de los dispositivos del sistema (dispositivos de visión como cámaras, video cámaras, sensores de rango láser, etc. y lectores RFID).

Los clientes OMA LWM2M, tal y como se explicó en la sección 3.1.1, están compuestos de objetos que a su vez están compuestos por recursos. A la hora de considerar qué objetos deben componer un cliente LWM2M se debe tener en cuenta que hay una serie de objetos que todo cliente LWM2M debe implementar. Estos son objetos de carácter obligatorio y contienen los recursos necesarios para identificar al dispositivo y el servidor y ser gestionados dentro de la red LWM2M. En la sección 4.2.1 se tratará este tipo de objetos en detalle.

Por otro lado, para particularizar el cliente LWM2M, es necesario incluir objetos adicionales que proporcionen los recursos necesarios para entregar información acerca de las características del cliente y la carga útil en sí. Dichos objetos son explicados en detalle en la sección 4.2.2.

4.2.1. Objetos LWM2M Obligatorios

Todos los clientes LWM2M, deberán implementar los objetos LWM2M obligatorios especificados en [9]. Dichos objetos son (ver sección 3.1.1 para una breve descripción):

- LWM2M Security (ID = 0).
- LWM2M Server (ID = 1).
- Device (ID = 3).

Dichos objetos contienen información que identifican al dispositivo que implementa dicho cliente, así como el servidor LWM2M al que se encuentra conectado y los protocolos de seguridad que implementa.

Los objetos LWM2M y LWM2M Server están presentes en el cliente. Sin embargo, debido a que no aportan funcionalidad a este sistema, no han sido particularizados para los clientes LWM2M creados como parte del sistema aquí presentado.

El objeto Device, sin embargo, ha sido particularizado para ser usado como punto de entrada a la identificación del dispositivo. De entre sus recursos, se ha considerado particularizar los siguientes:

- Manufacturer: este campo indica el fabricante, para este campo, hemos usado el identificador “REHABITA”.
- Model Number: es un campo que indica el modelo del dispositivo. Para este campo, hemos usado tres identificadores: Model 500, Model 600 y Model 700; todos ellos se encuentran detallados en la sección 4.3.

4.2.2. Objetos LWM2M Adicionales

Además de los objetos anteriores, considerados obligatorios para un cliente LWM2M, es necesario añadir otros que especifiquen la naturaleza del dispositivo, tanto para el control del dispositivo, como para conocer los recursos que ofrece dicho dispositivo.

El objeto especificado dentro del marco LWM2M que cubre dichos requisitos es el objeto Device Capability Management [10], explicado anteriormente en la sección 3.1.3. El objeto deberá considerar los siguientes atributos:

Device Capability Management (ID = 15)

- Property: es un atributo de tipo texto, indicará las capacidades dentro del siguiente recurso (Group). Este atributo deberá mostrar las capacidades del dispositivo, tanto para visión, como para comunicaciones (en el caso de lectores RFID).
- Group (capabilities):
    - 5: se corresponde con el grupo VISION, es la capacidad empleada para indicar que el dispositivo aporta información visual al servidor.
    - 2: se corresponde con el grupo CONNECTIVITY, capacidad empleada por los lectores RFID para mostrar al servidor el envío plano de datos.

- opEnable habilita al dispositivo para actuar.
- opDisable deshabilita el dispositivo.

El resto de atributos serán implementados tal y cómo se indica en el Anexo A. Inclusión de nuevos objetos en Leshan.

Sin embargo, el objeto Device Capability Management no permite el intercambio de la información registrada por el dispositivo. No existe objeto en LWM2M que permita el intercambio continuo de gran cantidad de datos, es por ello que es necesario crear un objeto LWM2M para el intercambio de la información registrada por los dispositivos del sistema. La información intercambiada será volcada directamente al cliente LWM2M en un recurso del tipo opaco ("OPAQUE").

El tipo "opaque" de LWM2M [9] es una secuencia de bytes cuya longitud puede ser indefinida.

El objeto que debe ser creado para registrar la información enviada por los dispositivos de visión deberá tener la siguiente forma:

Device Data (ID = 50[1])

El objeto Device Data deberá permitir múltiples instancias y será un componente opcional.

- Capability: es un campo de texto con un miembro del recurso Group member (objeto Device Capability Management) recurso de solo lectura y tipo "string". Este dato es obligatorio. Los datos serán mostrados para contemplar tanto la propiedad como el grupo al que pertenecen los datos (existe un vínculo con las propiedades definidas en el objeto Device Capability Management).
- Timing: es un recurso de lectura y escritura del tipo "integer" que indica el tiempo entre secuencias de datos enviados por el dispositivo. Este dato es opcional.
- Units: recurso de sólo lectura del tipo "string" que indica las unidades del recurso anterior. Este dato es opcional.
- Datatype: recurso de sólo lectura del tipo "string" que indica el formato de la secuencia de datos enviados por el dispositivo. En el caso de dispositivos de visión, dicho formato deberá indicar el tipo de datos y el número de píxeles. En el caso de lectores RFID, el tipo de codificación y longitud de la trama a enviar. Este dato es obligatorio.
- Raw Data: recurso de sólo lectura del tipo "opaque" donde el dispositivo vuelca los datos directamente enviados por el dispositivo. El atributo anterior indicará a la aplicación el formato en el que estos datos son enviados para que pueda procesarlos. Este dato es obligatorio.

[1] Objeto no registrado en OMNA (http://technical.openmobilealliance.org/Technical/technical-information/omna). OMA es la autoridad que registra los nuevos objetos proporcionando un identificador.

4.3. Definición de los modelos de clientes LWM2M

En la sección 0 reflejamos la funcionalidad esperada por el sistema propuesto. Como se ha mencionado en apartados anteriores, la finalidad del sistema es la de ofrecer la información recogida por dispositivos de visión y lectores RFID. Para dar soporte a estos dispositivos, se han creado los siguientes modelos de clientes LWM2M: Model 500, Model 600 y Model 700.

Los modelos propuestos en esta sección se han escogido como elementos básicos para construir una red compuesta por dispositivos de visión y lectores RFID. Dichos elementos son:

- Dispositivo de visión (cámara, videocámara, sensor de rango, etc.).
- Lector RFID.

El modelo Model 500 ha sido creado para exponer los recursos de un dispositivo de visión genérico a disposición de la red. Análogamente se ha creado el modelo Model 600 para un lector RFID genérico.

Adicionalmente, y dada la naturaleza de la aplicación final (reconstrucción virtual de espacios inhabitados, como se explicó anteriormente en la sección 0) se ha considerado útil la creación de un tercer modelo, Model 700 el cual pone a disposición de la red las propiedades de un dispositivo mixto de visión y lector RFID. Esto hace que sea más fácil la gestión de la red para tal fin, ya que al estar asociados al mismo cliente, puede asociarse la información de los tags RFID recibidos con las imágenes que el dispositivo de visión está emitiendo.

4.3.1. Model 500

El modelo “Model 500” ha sido creado fundamentalmente para dar soporte a un dispositivo de visión. Básicamente pone a disposición del servidor LWM2M los recursos necesarios para interactuar con un dispositivo de visión.

En el arranque dicho modelo de cliente LWM2M deberá mostrar los objetos declarados en la sección 4.2.

Para el caso concreto del dispositivo LWM2M que queremos crear en el presente trabajo, éste debe presentar las siguientes instancias:

- LWM2M Security: 1 instancia.
- LWM2M Server: 1 instancia.
- Device: 1 instancia.
- Device Capability Management: 1 instancia.
- Device Data: 1 instancia.

La instancia del objeto Device Capability Management debe ser configurada con los siguientes parámetros:

- Property:
  - “0”. Si se trata de una cámara.
  - “1”. Si se trata de una videocámara.

    - "0;1". Si puede emplearse como cámara y video cámara.
- Group (capabilities):
    - 5: se corresponde con el grupo VISION.

Por otro lado, la instancia del objeto Device Data contendrá los siguientes parámetros:

- Capability: este recurso deberá estar alineado con los recursos Property y Group del objeto Device Capability Management y podrá ser:
    - "5(0)". Si se trata de una cámara.
    - "5(1)". Si se trata de una video cámara.
    - "5(0;1)". Si puede emplearse como cámara y video cámara.
- Units: normalmente, esta unidad suele ser "fps" (frames per second) si el dispositivo se trata de una video cámara.
- Datatype: "XXX_<Columnas>x<Filas>". Donde XXX son las siglas de formato. Como ejemplos: "RGB_1280x960" o "YUV_640x480".
- Raw Data: es un vector con la imagen a transmitir.

4.3.2. Model 600

El modelo "Model 600" ha sido creado fundamentalmente para dar soporte a un dispositivo lector de RFID. Básicamente pone a disposición del servidor LWM2M los recursos necesarios para interactuar con un dispositivo lector de RFID.

En el arranque dicho modelo de cliente LWM2M deberá mostrar los objetos declarados en la sección 4.2.

Para el caso concreto del dispositivo LWM2M que queremos crear en el presente trabajo, éste debe presentar las siguientes instancias:

- LWM2M Security: 1 instancia.
- LWM2M Server: 1 instancia.
- Device: 1 instancia.
- Device Capability Management: 1 instancia.
- Device Data: 1 instancia.

La instancia del objeto Device Capability Management debe ser configurada con los siguientes parámetros:

- Property:
    - "3". Indica comunicación NFC, que es lo más parecido que puede obtenerse de la versión actual de [10].
- Group (capabilities):
    - 2: se corresponde con el grupo CONNECTIVITY.

Por otro lado, la instancia del objeto Device Data contendrá los siguientes parámetros:

- Capability: este recurso deberá estar alineado con los recursos Property y Group del objeto Device Capability Management y podrá ser:

    - "2(3)"para todo lector RFID.
- Datatype: "RAW_1x<Columnas>". RAW indica que el vector ofrecido es la información tal y como es leída, nuevos formatos RFID pueden ser incorporado usando estas tres letras.
- Raw Data: es un vector con la trama a transmitir.

4.3.3. Model 700

El modelo "Model 700" ha sido creado como un modelo especial que combina los dos anteriores, de tal manera que podemos concentrar en un cliente RFID los recursos de un dispositivo de visión y un dispositivo lector de RFID.

En el arranque dicho modelo de cliente LWM2M deberá mostrar los objetos declarados en la sección 4.2.

Para el caso concreto del dispositivo LWM2M que queremos crear en el presente trabajo, éste debe presentar las siguientes instancias:

- LWM2M Security: 1 instancia.
- LWM2M Server: 1 instancia.
- Device: 1 instancia.
- Device Capability Management: 2 instancias.
- Device Data: 2 instancias.

Para mostrar las capacidades de dispositivo de visión además de lector de RFID es necesario instanciar por cada dispositivo los objetos Device Capability Management y Device Data, como se ha mostrado anteriormente. Por convención el criterio para asignar dichas instancias es el siguiente:

- Device Capability Management:
    - Instancia 0: Dispositivo de visión (parámetros dados en 4.3.1).
    - Instancia 1: Dispositivo RFID (parámetros dados en 4.3.2).
- Device Data:
    - Instancia 0: Dispositivo de visión (parámetros dados en 4.3.1).
    - Instancia 1: Dispositivo RFID (parámetros dados en 4.3.2).

5. Implementación de la red LWM2M

El objetivo final del presente trabajo era la creación de una infraestructura LWM2M para gestionar la información recibida por un dispositivos de visión (cámara, video cámara, etc.) y lectores RFID para posteriormente ser gestionados por una aplicación que reflejara el potencial de dicha red. Dicho objetivo se plasma en el resultado mostrado a lo largo de esta sección con la creación de tres modelos de cliente LWM2M (ver sección 4.3) basados en el proyecto Leshan de Eclipse que posea las capacidades de ser gestionado y poner a disposición de las aplicaciones la información recibida por los dispositivos. Sobre esta red ha sido creada una aplicación basada en MATLAB capaz de gestionar los recursos ofrecidos por esta red.

La implementación del sistema recogida a lo largo de esta sección es un ejemplo de implementación de la arquitectura propuesta en la sección 4 y la Figura 8.

La solución alcanzada en el presente trabajo ofrece una especificación de clientes LWM2M para numerosos dispositivos, ya que se ha tratado de mantener la generalidad en su creación, ofreciendo una solución para aquellas aplicaciones que requieran dispositivos que transmitan gran cantidad de información.

A lo largo del presente trabajo se ha ido mostrando cómo ha sido adaptado el estándar LWM2M a las necesidades del proyecto. Como parte de la creación del cliente LWM2M para dispositivos de visión y lectores RFID se ha creado el objeto LWM2M Device Data. A través de este objeto, se ha intentado crear un medio genérico para intercambiar grandes cantidades de información, notificando a través de uno de sus recursos el formato en el que dicha información es transmitida. La creación de dicho objeto ha sido necesaria pues no existe un objeto similar en los registros de OMNA (Open Mobile Naming Authority).

La implementación de dicha infraestructura LWM2M (servidor y cliente) en Leshan se detalla en la sección 0. Donde se refleja cómo se ha trabajado sobre el proyecto existente, disponible en [12] para ser aplicado al propósito de crear un parque de cámaras y lectores RFID para la reconstrucción virtual de espacios no habitados.

Para alimentar la infraestructura LWM2M creada, se ha implementado, a modo de ejemplo de dispositivo de visión, un modelo de cámara que recoge las imágenes captadas por la webcam integrada en un ordenador portátil. Por otro lado, para estimular el sistema desde el punto de vista de un lector RFID se ha generado un modelo que construye tramas de manera aleatoria simulando la recepción de datos de etiquetas RFID. Dichos ejemplos de dispositivos controlados por los clientes LWM2M mostrado pueden encontrarse en la sección 5.2.

Por último, la sección 5.3 muestra cómo los recursos puestos a disposición de la red LWM2M son gestionados a través de una aplicación. El acceso a dichos recursos está basado en URL, por tanto esto permite independencia con el entorno en el que la aplicación es desarrollada. En el ejemplo aquí creado se ha desarrollado dicha aplicación en MATLAB por ser una herramienta con grandes ventajas y posibilidades en el procesamiento de imágenes y datos.

La aplicación que podemos ver en la sección 5.3, llamada LWM2MApp agrupa los clientes según su tipo (dispositivos de visión o lectores RFID) y recoge la información transmitida por éstos para posteriormente ser almacenada. Durante su ejecución, muestra la información recibida y almacenada junto con su etiqueta de tiempo, la cual es útil para un posterior procesamiento de los datos almacenados.

5.1. Implementación en Leshan

En el arranque el cliente LWM2M deberá mostrar los objetos declarados previamente en la sección 4.2.2. Dichos objetos son Device, Device Capability Management y Device Data, junto con los objetos LWM2M que son obligatorios para todo cliente LWM2M (ya declarados en la sección 4.2.1).

Leshan dispone de una "librería" donde alberga los objetos LWM2M disponibles para ambos, cliente y servidor. Cualquier elemento fuera de dicha "librería" no es reconocido ni por el cliente, ni por el servidor. Dicha librería no es más que una especificación en JSON [14] (JavaScript Object Notation) que mapea los identificadores de objetos y de recursos a una estructura de nombres y valores admisibles y modos de acceso a dicho recurso. La creación de la especificación JSON para los objetos LWM2M se recoge en la sección 5.1.1, mientras que la implementación de dichos objetos se muestra en la sección 5.1.3.

La Figura 10 muestra la UI web del servidor Leshan con un cliente que implementan los objetos LWM2M aquí especificados. En la figura sólo se muestran los objetos que han sido implementados, como indicamos anteriormente.

Las instancias de cada uno de los objetos LWM2M anteriores y los dispositivos asociados a ellos dependen del modelo a implementar, dichos modelos son los previamente definidos en la sección 4.3. La implementación de dichos modelos será recogida en la sección 5.2. En esta sección se reflejan las modificaciones realizadas sobre el proyecto Leshan [12] para la implementación del sistema propuesto.

Implementar el sistema propuesto en Leshan requiere manipular las siguientes clases:

- *LeshanClientExample* es un ejemplo de cliente implementado en Leshan, en esta clase se recoge el nivel de aplicación del cliente.
- *ObjectsInitalizer* inicializa los objetos del cliente LWM2M. Debido a su funcionalidad restringida, no permite crear las instancias requeridas para el dispositivo en el arranque.

Además, es necesario que la estructura creada en Leshan para la implementación de esta red permita alimentar el sistema con distintos tipos de dispositivos. A lo largo de esta sección se detallará cómo se ha generado una arquitectura que posibilita dicha generalización en el acceso a dispositivos, esto se detalla en la sección 5.1.3 para ser posteriormente ampliado en la sección 5.2.

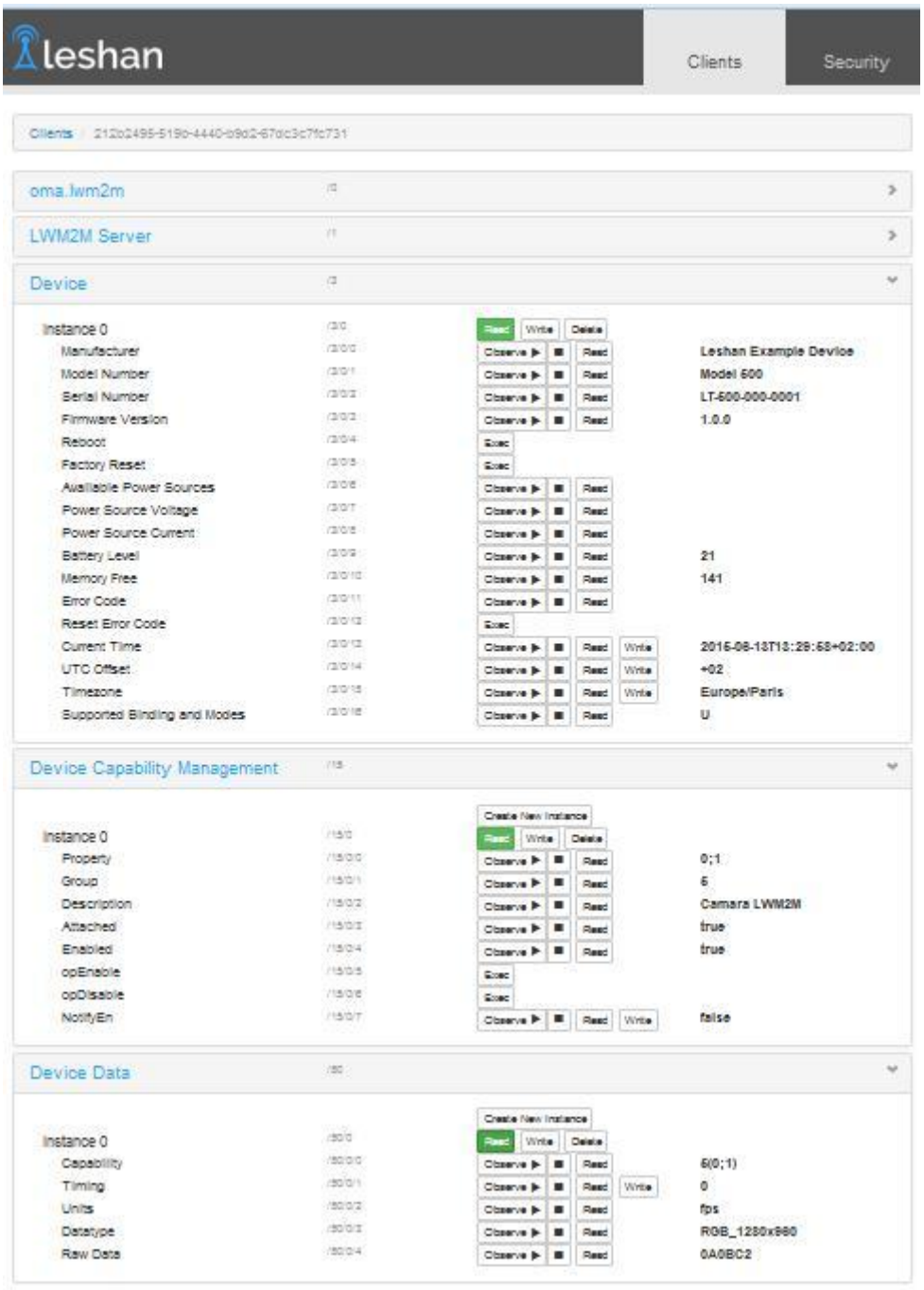

Figura 10. Ejemplo de cliente para dispositivos de visión y lectores RFID.

5.1.1. Implementación de los objetos en JSON

La definición de los objetos declarados anteriormente debe ser expresada en formato JSON [14]. JSON es un estándar de es un sistema de intercambio de datos empleado por Leshan para la transmitir objetos OMA LWM2M al servidor y para la creación de clientes LWM2M utilizado en el mapeo de identificadores de objetos y recursos. Ésta es la estructura gestionada por Leshan tanto por el cliente como por el servidor (ambos deben compartir los mismos objetos):

```
{
"name": "<Nombre Objeto>",
"id": <Número de ID>,
"instancetype": "<multiple (varias instancias) o single (una única instancia)>",
"mandatory": <false (implementación no requerida) o true (implementación requerida)>,
"description": "<descripción opcional>",
"resourcedefs": [
        {
                "id": <Número de ID del recurso dentro del objeto>,
                "name": "<Nombre del recurso>",
                "operations": <"R" (solo lectura), "RW" (solo escritura) o "E" (ejecutable)>,
                "instancetype": "<multiple (varias instancias) o single (una única instancia)>",
                "mandatory": <false (implementación no requerida) o true (implementación requerida)>,
                "type": <El tipo de dato que gestiona el recurso, puede ser: "string", "integer", "float", "boolean", "opaque", "time">,
                "range": "<rango que abarcan los datos>",
                "units": "<unidades del recurso (opcional)>",
                "description": "<descripción del recurso (opcional)>"
        },
        {
        …
        }
   ]
},
```

La descripción de la estructura que poseen los objetos está recogida en OMA LWM2M [9]. En ella se da una breve descripción de los campos que componen cada objeto y el significado que ello tiene.

Leshan [12] posee por defecto todos los objetos de OMA LWM2M identificados en [9]. Por tanto, sólo los objetos definidos en detalle en la sección 4.2 (Device Capability Management y Device Data) han tenido que ser añadidos al fichero **oma-objects-spec.json** que es archivo que posee dichas especificaciones.

La implementación de dichos objetos puede verse en el Anexo A. Inclusión de nuevos objetos en Leshan.

5.1.2. Inicialización del cliente

El punto de entrada al cliente en Leshan es el ejemplo de cliente que se proporciona en el proyecto. Se ha partido de esta implementación para construir nuestro cliente.

Este cliente como punto de partida, implementa el método *main* que básicamente consiste en la inicialización del objeto *LeshanClientExample*, proporcionando como argumentos la IP del servidor y el puerto de dicho servidor al que el cliente va a conectarse. Dichos argumentos deben ser introducidos al arrancar el programa.

La inicialización de los objetos e instancias se realizan en la clase *ObjectsInitializer*. Leshan no permite la creación de objetos e instancias "a la carta". Por tanto, ha sido necesaria la manipulación de dicha clase para tal fin creando el método *CreateClient,* que veremos con más detalle a continuación.

Para inicializar el cliente desde *LeshanClientExample*, es necesario realizar dos acciones:

- Asignar la clase que gestionará dicho objeto. Dicha clase será empleada para posteriormente gestionar las operaciones de lectura, escritura y ejecución de los recursos de cada instancia, como se verá en el apartado 5.1.3. La asignación se realiza con el método *setClassForObject(int objectID, Clazz Clase.class).*
- Llamar al método *CreateClient* de la clase *ObjectsInitializer.* Este método creará las instancias requeridas de los objetos indicados en la inicialización.

Las instancias deseadas para cada cliente LWM2M se han indicado en la sección 4.3 y dependen del modelo asignado. Así pues, para generar dicha configuración de cliente, debemos tener en cuenta dicho modelo. En función de éste, las configuraciones son:

- Model 500: *initializer.CreateClient(0,1,3,15,50);*
- Model 600: *initializer.CreateClient(0,1,3,15,50);*
- Model 700: *initializer.CreateClient(0,1,3,15,15,50,50);*

Donde *initializer* es el objeto de la clase *ObjectsInitializer*.

El modelo es proporcionado por la clase *DevManager* que veremos en detalle en la sección 5.2.

Esta inicialización proporciona los *enablers* que son necesarios para iniciar el cliente Leshan (clase *LeshanClient*). El objeto de la clase *LeshanClient* es el que se encarga de gestionar el cliente LWM2M una vez inicializado con el método *start()* tras haberle proporcionado los *enablers.*

Todo esto lo podemos ver resumido en el diagrama de clases de la Figura 11.

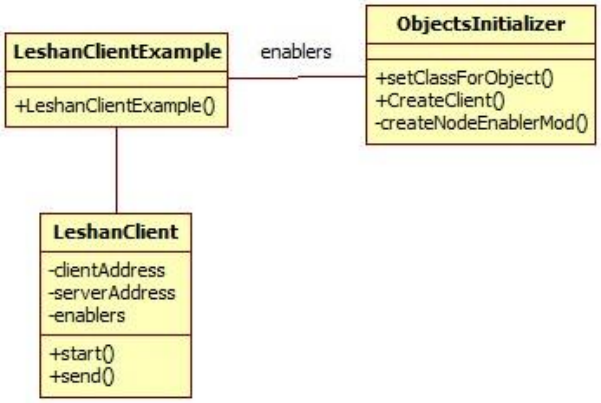

Figura 11. Diagrama de clases del cliente implementado.

5.1.3. Gestión de los recursos del cliente

Cada objeto del cliente LWM2M es gestionado por una clase, la cual es asociada a través del método *setClassForObject()* de la que hablamos anteriormente en la sección 5.1.2.

En este caso, sólo se ha considerado gestionar los siguientes objetos:

- Device (ID = 3).
- Device Capability Managemen (ID = 15).
- Device Data (ID = 50).

Dichos objetos serán gestionados por las siguientes clases: *Device, DevCapMgmt* y *DataMgmt,* respectivamente. Las cuales recogen métodos para gestionar cada uno de los recursos del objeto enlazado.

Una vez que cada clase es asociada a cada objeto, se inicializan las instancias como se ha visto en el apartado 5.1.2 y los *enablers* pasan al objeto de la clase *LeshanClient* es éste el momento en el que por cada instancia de cada objeto se crea un objeto de la clase a la que ha sido enlazado previamente.

A partir del en lace establecido entre clase y objeto, cada recurso de la instancia del objeto LWM2M pasa a ser gestionado por la instancia de la clase a la que ha sido enlazado, realizando sobre éstos las operaciones permitidas por cada recurso (lectura, escritura y ejecución). A través de estas clases se da respuestas a dichas peticiones, por tanto, son estas clases las que deberán gestionar la información del dispositivo y las que actuarán sobre éste.

La función de estas clases es meramente la gestión de los objetos e instancias del cliente LWM2M, la interacción con los dispositivos y el tratamiento de su información es realizada por otras clases, como veremos en la sección 5.2.

El diagrama de la Figura 12 muestra cómo las clases que gestionan los objetos LWM2M del cliente se relacionan con la clase principal que inicializa el cliente (*LeshanClientExample*).

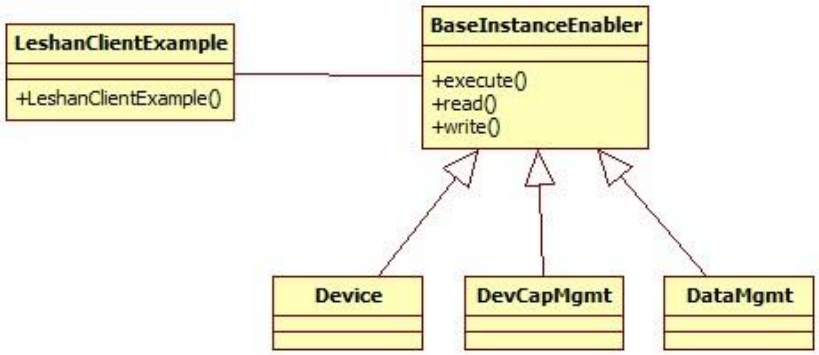

Figura 12. Gestión de los recursos del cliente.

5.2. Implementación de los modelos de cliente LWM2M

La implementación de los modelos de cliente en la sección 4.3 ha sido realizada sobre Leshan. La gestión del modelo del cliente LWM2M es realizada a través de la clase *DevManager*. Dicha clase recoge el modelo del que se trata (Model 500, Model 600 ó Model 700) y sirve como punto de unión de todas las peticiones lanzadas por las instancias de los objetos LWM2M.

Para implementar dicha clase, se ha empleado el patrón de diseño Singleton (Figura 13), el cual permite que todos los objetos de diferentes clases apunten al mismo objeto. De esta manera, los objetos creados por las clases mostradas en la sección 5.1.3 siempre interactuarán con el mismo objeto de la clase *DevManager*. Esto proporciona la ventaja de asociar cada instancia de los objetos LWM2M al dispositivo correspondiente. La Figura 14 muestra dicha implementación.

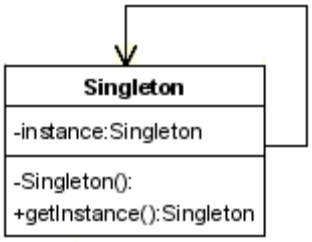

Figura 13. Patrón Singleton.

Como se puede ver en la Figura 14, el objeto único de la clase *DevManager* recoge la información de las clases encargadas de gestionar los objetos LWM2M para procesarla en base al modelo al que pertenece. Es por ello que es necesario conocer la instancia del objeto que llama a la clase *DevManager*, ya que en base a esto la información puede llamar a la clase encargada de gestionar un dispositivo u otro, como puede ser *Camera* para los dispositivos de visión o *RFIDReader* para los lectores RFID.

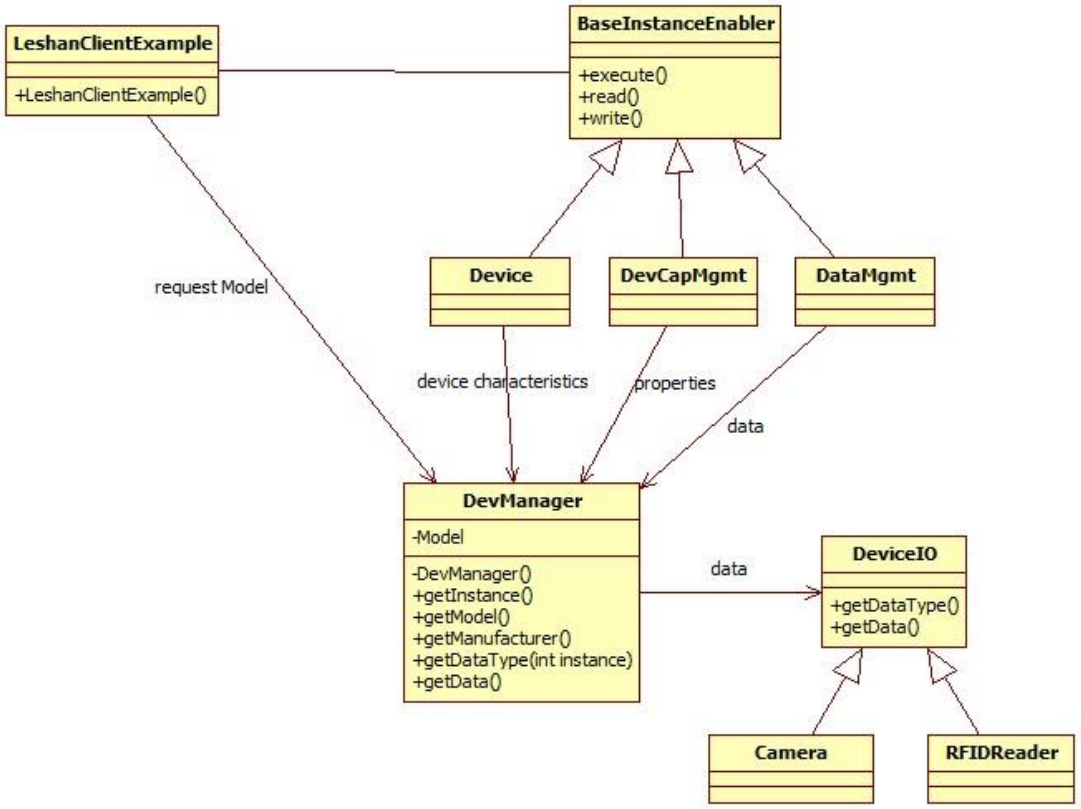

Figura 14. Implementación de los modelos de cliente.

El modelo recogido por esta clase sirve además para crear la estructura del cliente LWM2M. Como vimos en la sección 5.1.2, la clase *LeshanClientExample* es la encargada de crear el cliente LWM2M con los objetos e instancias necesarias según el modelo. El modelo de cliente empleado es consultado a la clase *DevManager* y en base a éste, se crea dicho cliente.

El objeto por tanto de la clase *DevManager* no es otro que el de concentrar las peticiones a los distintos recursos del cliente LWM2M y gestionarlos en base al modelo de cliente. La ventaja proporcionada por la arquitectura mostrada en la Figura 14 es principalmente la flexibilidad que proporciona a la hora de gestionar distintos dispositivos bajo un mismo cliente LWM2M.

5.2.1. Clase *Camera*

La clase *Camera* (Figura 14) es la encargada de gestionar dispositivos de visión y la recogida y tratamiento de sus imágenes para ser transmitidas al cliente correspondiente. Es esta clase la encargada de interactuar con este tipo de dispositivos poniendo a disposición del sistema las imágenes recogidas.

En la implementación realizada en el presente trabajo, el dispositivo de visión empleado es la propia webcam integrada del ordenador. El objeto de la clase *Camera* interactúa con ella a través de la API JMF (*Java Media Framework*) [16].

La API JMF habilita la captura de imágenes desde las cámaras registradas. El objeto de la clase *Camera* toma una captura del dispositivo registrado por JMF bajo demanda y devuelve una

imagen. Tras crear esta imagen, ésta es convertida en un buffer de datos con la imagen, devolviendo el vector con bytes que es requerido por el cliente LWM2M.

5.2.2. Clase *RFIDReader*

La clase *RFIDReader* (Figura 14) pretende simular el comportamiento de un lector RFID. El propósito de esta clase en el sistema es el de conectarse al dispositivo y transmitir la información recibida por éste.

En un lector RFID la información transmitida es el contenido recibido de la etiqueta al cual puede añadir una etiqueta de tiempo sobre cada trama.

El caso considerado para estimular el sistema desarrollado es la transmisión únicamente de las tramas recibidas. El objetivo de este ejercicio era el de estimular el sistema, por tanto, no es necesario el uso de un dispositivo real. La técnica empleada para estimular dicho sistema ha sido el de generar las tramas con un generador de números aleatorios que proporcione al sistema la estimulación necesaria para enviar información de manera dinámica.

En la aplicación que procesa esta información (sección 5.3), la información es solamente leída y posteriormente almacenada.

5.3. LWM2MApp

LWM2MApp es la aplicación creada para mostrar las capacidades del sistema presentado en el presente trabajo. Es una aplicación creada en MATLAB por el potencial que presenta esta herramienta para el post procesamiento de imágenes e información recibida de las etiquetas RFID o incluso de dicho procesamiento en tiempo real.

LWM2MApp está formada por un conjunto de funciones MATLAB que permiten que la aplicación LWM2MApp.m se mantenga simple en su estructura. La descripción de dicho sistema de ficheros puede encontrarse en el Anexo B. LWM2MApp: Estructura de ficheros.

El acceso a la red LWM2M y a los recursos disponibles en ésta se realiza mediante URL, el servidor LWM2M pone a disposición de las aplicaciones los recursos logados a éste, los clientes. Una vez recogidos los clientes logados en la red LWM2M, LWM2MApp clasifica y posteriormente almacena la información transmitida por los dispositivos de la red.

5.3.1. Intercambio de datos con Leshan

El acceso de la aplicación LWM2MApp a los recursos de Leshan es realizado mediante una API basada en URL proporcionada por este proyecto, el cual es la base para construir aplicaciones sobre la estructura LWM2M.

LWM2MApp sólo requiere la lectura de los recursos disponibles en la red LWM2M, por tanto, sólo realiza operaciones de lectura sobre éstos. Son estas operaciones de lectura las que se tratan en este apartado.

El acceso a los recursos de una URL en MATLAB se realiza a través de la función *urlread()*. El argumento de entrada a esta función es la URL en formato string y como salida la función devuelve un string con los recursos recogidos.

Leshan devolverá siempre la respuesta a la URL consultada en formato JSON [14]. La API de Leshan es llamada cada vez que se emplea la siguiente estructura:

http://<serverIp>:<serverPort>/api/clients

Donde:

<serverIp> es la IP del servidor Leshan. Por defecto, Leshan funciona sobre la dirección "*localhost*" para poder ser utilizado localmente.

<serverPort> es el puerto del servidor Leshan, que por defecto es 8080.

Para permitir que la aplicación funcione independientemente de la IP y del puerto en el que reside el servidor, la interfaz de LWM2MApp.m considera los anteriores parámetros como argumentos de entrada:

```
function LWM2MApp(serverIp, serverPort)
```

Si LWM2MApp es llamada sin argumentos de entrada empleará por defecto *localhost:8080.*

La URL anteriormente mostrada no sólo es el punto de entrada a los recursos de Leshan, sino que como respuesta devuelve los clientes logados a la red. Como en el siguiente ejemplo:

```
>> urlread('http://localhost:8080/api/clients')
ans =
[{"endpoint":"473bac3c-b385-4468-947e-fca335a6be9b",
"registrationId":"fviQt9Djm5",
"registrationDate":"2015-09-12T20:10:53+01:00",
"address":"/127.0.0.1:55394",
"lwM2MmVersion":"1.0",
"lifetime":86400,
"bindingMode":"U",
"rootPath":"/",
"objectLinks":[{"url":"/","attributes":{"rt":"oma.lwm2m"}},{"url":"/0/0","attributes":{},"objectId":0,"objectInstanceId":0},{"url":"/1/0","attributes":{},"objectId":1,"objectInstanceId":0},{"url":"/3/0","attributes":{},"objectId":3,"objectInstanceId":0},{"url":"/15/0","attributes":{},"objectId":15,"objectInstanceId":0},{"url":"/50/0","attributes":{},"objectId":50,"objectInstanceId":0}],
"secure":false}]
```

Esta estructura se repite tantas veces como clientes estén logados en la red LWM2M los caracteres [] indicarán el comienzo y final de la respuesta de la URL y la separación entre clientes vendrá representada por el nivel superior de corchetes {}. Del anterior ejemplo, cabe destacar el atributo "endpoint" que será la puerta de acceso a los recursos de dicho cliente.

Por otro lado, el atributo “objectLinks” nos muestra la estructura de objetos LWM2M e instancias que componen dicho cliente.

El acceso a un recurso determinado de un cliente se realiza añadiendo el <endpoint> de dicho cliente y posteriormente el <objeto> y la <instancia> y finalmente el <recurso> como se indica a continuación:

http://<serverIp>:<serverPort>/api/clients/<endpoint>/<objeto>/<instancia>/<recurso>

Por ejemplo, el acceso al recurso “Manufacturer” (fabricante) que se encuentra en el objeto “Device” (ID=3) sería:

```
>> urlread('http://localhost:8080/api/clients/473bac3c-b385-4468-947e-fca335a6be9b/3/0/0')
ans =
{"status":"CONTENT","content":{"id":0,"value":"REHABITA"}}
```

Si se accede a una instancia, en su lugar, Leshan devuelve como respuesta el conjunto de recursos asociados con su valor.

Una vez visto cómo la aplicación interactúa con la plataforma LWM2M, en las siguientes secciones se mostrará la funcionalidad de LWM2MApp.

5.3.2. Clasificación de los clientes de la red

LWM2MApp detecta los clientes logados en el servidor y comienza a filtrarlos. El primer filtro que establece es a través de la consulta del fabricante. La aplicación sólo procesa aquellos clientes cuyo recurso Manufacturer (recurso ID=0) del objeto Device (ID =3) es REHABITA. Sólo en el caso de que el cliente haya sido especificado así, se considerará para ser procesado.

Una vez que el campo fabricante (Manufacturer) del cliente ha sido chequeado, éste es filtrado según su modelo distinguiendo entre:

- Model 500: dispone de un dispositivo de visión.
- Model 600: dispone de un dispositivo lector de RFID.
- Model 700: dispone de un dispositivo de visión junto con un dispositivo lector RFID.

Tras clasificar los clientes en base a su modelo, la aplicación agrupa los recursos de dichos clientes en cámaras y lectores RFID. Dicha información es posteriormente procesada y almacenada. La Figura 15 muestra gráficamente el proceso de clasificación de recursos.

A partir de este momento, la aplicación no trata clientes, sino el conjunto de recursos que han sido proporcionados por los distintos clientes logados en el servidor.

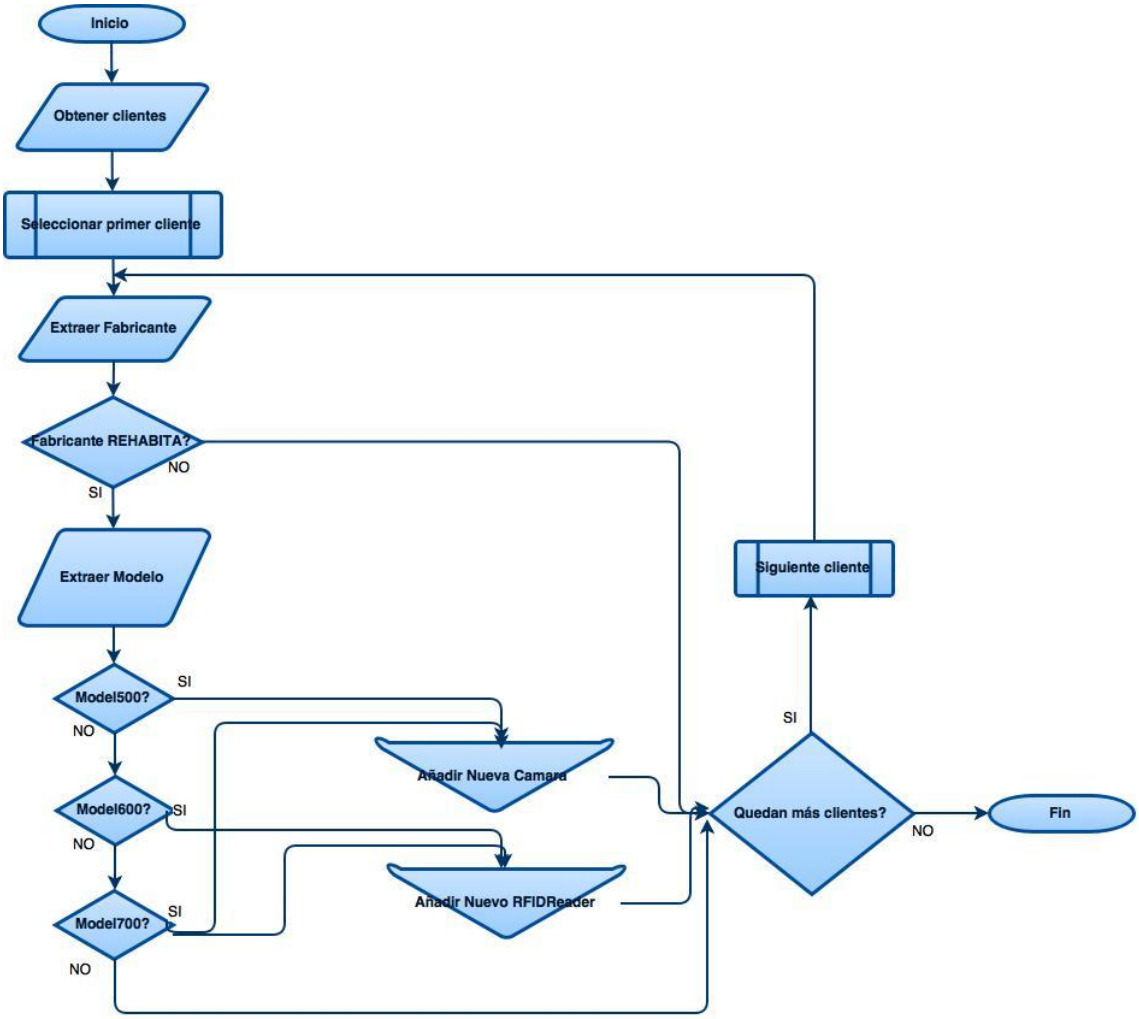

Figura 15. LWM2MApp. Clasificación de los recursos.

5.3.3. Lectura y almacenamiento de recursos

Una vez clasificados los recursos de los clientes en Cameras (cámaras) y RFIDReaders (lectores RFID), la aplicación LWM2MApp recoge de manera ininterrumpida los recursos de dichos dispositivos almacenándolos para su posible posterior procesado.

Las imágenes de los dispositivos de visión son almacenadas en formato JPEG separadas por cámara en distintos directorios. Para etiquetar el momento en el que la imagen fue adquirida, se emplea el campo "*Comment*" de la imagen, que posteriormente puede ser leído por MATLAB con el comando *imfinfo('Nombre.jpeg')* sobre cualquier imagen almacenada por la aplicación.

Por otro lado, las capturas de lectores RFID son almacenadas en ficheros XML, donde cada fichero corresponde a un lector RFID, la especificación de dicho fichero puede verse en el Anexo C. LWM2MApp: Formato de fichero XML para lecturas RFID.

Todos los ítems almacenados (tanto imágenes en visión como tramas RFID) son almacenadas junto con la etiqueta de tiempo registrada en la propiedad *Current Time* en el objeto *Device* del cliente LWM2M.

La ejecución de este bloque se interrumpe si la lista de clientes logados cambia, puesto que implica volver a clasificarlos, o si se decide terminar voluntariamente, mediante teclado.

La Figura 16 muestra el diagrama correspondiente a la gestión de los recursos recogidos en la fase de clasificación, donde se plasma lo anteriormente reflejado.

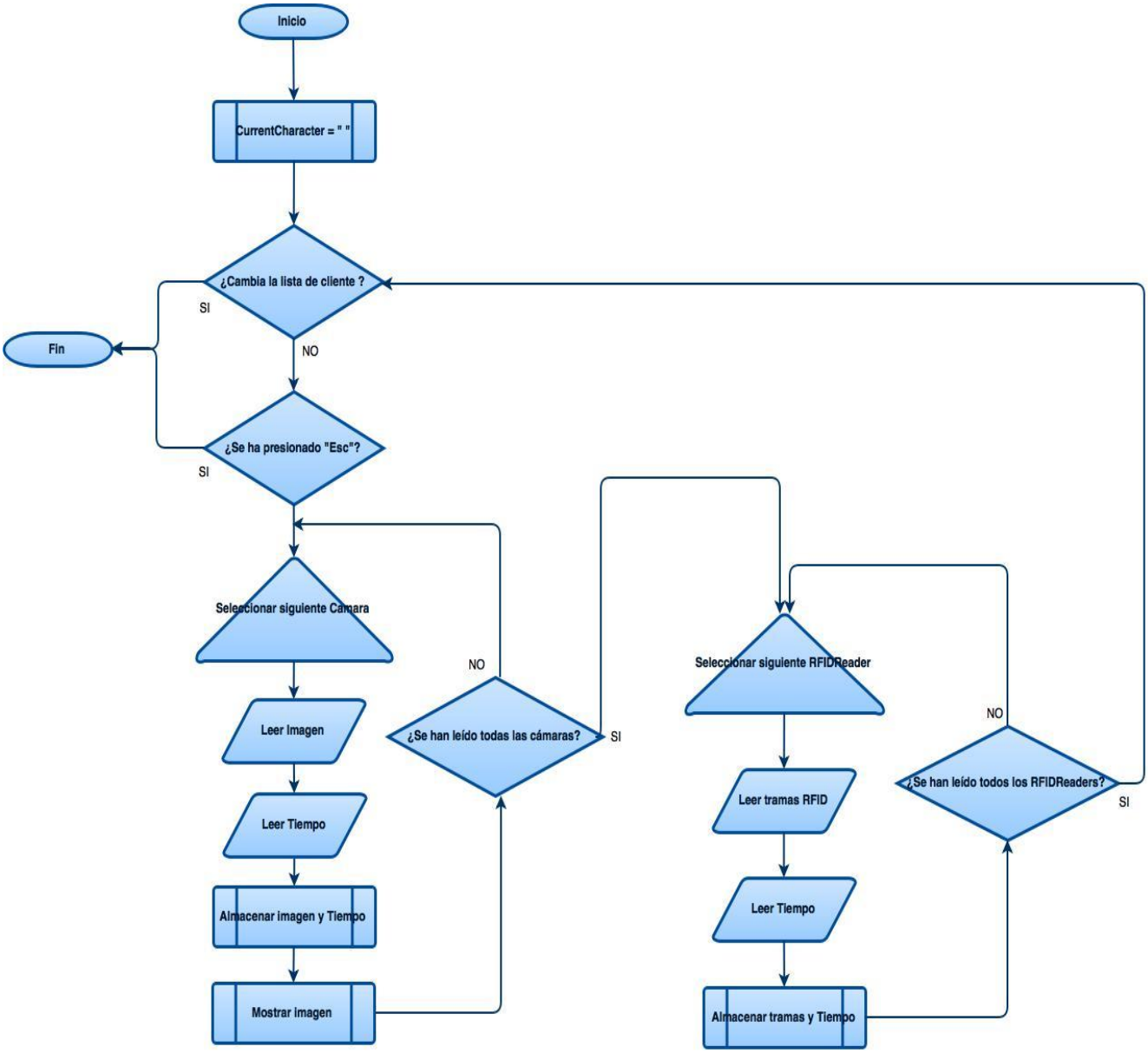

Figura 16. LWM2MApp. Gestión de Cámaras y lectores RFID.

Durante el tiempo de ejecución, la aplicación muestra la información de los dispositivos de visión en tiempo real etiquetándolos con la etiqueta de tiempo proporcionada por el cliente LWM2M. Así, la aplicación genera una ventana para mostrar las imágenes en tiempo real recibida por cada una de las cámaras, mientras que la información de los lectores RFID es mostrada en la consola tal y como muestra la Figura 17.

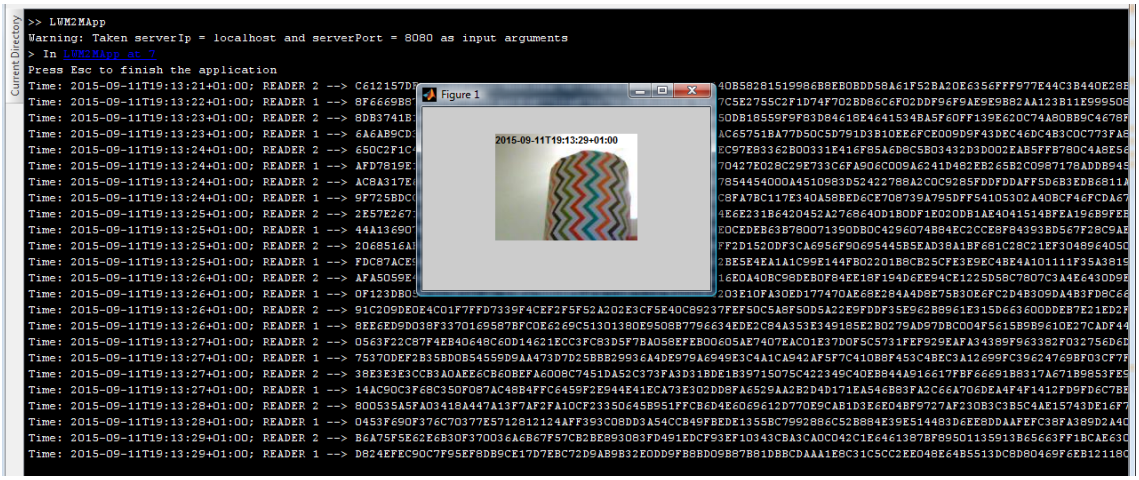

Figura 17. LWM2MApp en ejecución.

6. Conclusiones y trabajo futuro

La gestión de dispositivos a través de un servidor LWM2M permite compartir los recursos que ofrecen a distintas aplicaciones a través de un acceso remoto. El servidor realiza tareas de monitorizado sobre los clientes, de tal manera que se hace un chequeo regular sobre los dispositivos.

El uso de clientes LWM2M resulta de gran interés para la aplicación final del presente trabajo, la gestión de un parque de cámaras y lectores RFID para la reconstrucción virtual de espacios inhabitados. Gracias a la implementación LWM2M la aplicación final podrá independizarse de los dispositivos empleados para tal fin, por tanto, no dependerá del tipo de cámaras empleadas en dicha aplicación centrándose en lo importante, el tratamiento de imágenes.

Por otro lado, la aplicación de LWM2M al propósito de reconstrucción virtual de espacios no habitados permitirá añadir el número de dispositivos que se consideren oportunos de una manera sencilla. Incluso combinar dispositivos de distinta naturaleza en un cliente, de tal manera que nos pueda proporcionar distinta información recogida del entorno, como es el caso del uso de lectores RFID que se propone en [4].

Es importante recalcar la facilidad que tiene el cliente propuesto en el presente trabajo a ser adaptado a dispositivos de distinta naturaleza (lector RFID, sensores de rango, etc.), ya que se ha creado pensando en la generalidad. La especificación LWM2M permite una gran flexibilidad en la creación de clientes, mediante la adición y de objetos e instancias, permitiendo su adaptación a distintos dispositivos y necesidades.

El uso de información proveniente de lectores RFID junto con imágenes recibidas de cámaras o la información proveniente de sensores de rango es otro aspecto que se recalca en [4] para reducir los tiempos de computación. La solución propuesta combina la información de un lector RFID y de sistemas de visión como sensores de rango o cámaras en un único cliente, Model 700, recogiendo, de manera simultánea, la información de varias fuentes.

La solución aquí aportada sirve como punto de partida para el uso de este tipo de tecnología, muestra un ejemplo de integración de los dispositivos en una red LWM2M y su posterior uso a través de la aplicación creada en MATLAB. Como trabajo futuro para desplegar esta tecnología para el propósito reflejado anteriormente se proponen las siguientes líneas:

En primer lugar, crear clientes LWM2M adaptados a los dispositivos que serán empleados finalmente para el propósito de reconstruir espacios inhabitados. Dicha implementación puede seguir la arquitectura propuesta, la cual no presentará ningún cambio, ya que la comunicación con los dispositivos se realiza a través de las clases *Camera* (para los dispositivos de visión, ver sección 5.2.1) y *RFIDReader* (para los lectores RFID, ver sección 5.2.2).

Una segunda línea propuesta es la de crear la aplicación requerida para el procesado de imágenes e integración con etiquetado RFID en la reconstrucción de espacios inhabitados. La aplicación LWM2MApp, recogida en la sección 5.3, puede servir como punto de partida, ya que muestra en tiempo real y almacena los recursos de los clientes LWM2M. Una opción más

eficiente en este caso podría ser la de crear otra aplicación que acceda a los recursos almacenados por LWM2MApp para su procesado.

La solución desarrollada en el presente trabajo proporciona además la base para infinidad de aplicaciones relacionadas con el acceso remoto a dispositivos e intercambio de información. El uso de LWM2M permite la abstracción del dispositivo que adquiere dicha información, por tanto la red LWM2M puede ser fácilmente modificada para otro tipo de propósitos tales como: un sistema de trazabilidad RFID en un entorno industrial, sistemas de vigilancia, sistemas de percepción remota, etc.

7. Bibliografía

[1] “oneM2M White Paper”. oneM2M, January 2015, URL: http://www.onem2m.org/images/files/oneM2M-whitepaper-January-2015.pdf

[2] OMA (Open Mobile Alliance). URL: http://openmobilealliance.org/about-oma/

[3] “OMA Lightweight M2M White Paper”. Open Mobile Alliance™, March 2014, URL: http://openmobilealliance.hs-sites.com/free-m2m-whitepaper-from-oma

[4] Valero, E.; Adan, A.; Cerrada, C. “Automatic Construction of 3D Basic-Semantic Models of Inhabited Interiors Using Laser Scanners and RFID Sensors”. *Sensors* 2012, *12*, 5705-5724.

[5] Esposito, F.; Chiti, F.; Fantacci, R.; Hosio, S.; Junzhao S. "Agent Based Adaptive Management of Non-Homogeneous Connectivity Resources". *Communications, 2006. ICC '06. IEEE International Conference on*, vol.4, no., pp.1754,1759, June 2006.

[6] Louie F. Cervantes, Young-Seok Lee, Hyunho Yang, Jaewan Lee, "A Hybrid Middleware for RFID-based Parking Management System Using Group Communication in Overlay Networks". *IPC*, 2007, *Intelligent Pervasive Computing, International Conference on, Intelligent Pervasive Computing, International Conference on 2007*, pp. 521-526.

[7] “Lightweight Machine to Machine Architecture”, Open Mobile Alliance™, OMA-AD-LightweightM2M-V1_0-20131210-C, URL: http://openmobilealliance.hs-sites.com/lightweight-m2m-specification-from-oma

[8] “Lightweight Machine to Machine Requirements”, Open Mobile Alliance™, OMA-RDLightweightM2M-V1_0-20131210-C, URL: http://openmobilealliance.hs-sites.com/lightweight-m2m-specification-from-oma

[9] “Lightweight Machine to Machine Technical Specification”, Open Mobile Alliance™, OMA-TSLightweightM2M-V1_0-20141126-C, URL: http://openmobilealliance.hs-sites.com/lightweight-m2m-specification-from-oma

[10] “Lightweight M2M – Device Capability management Object (LwM2M Object – DevCapMgmt)”, Open Mobile Alliance™, OMA-TS-LWM2M_DevCapMgmt-V1_0-20150120-C, URL: http://openmobilealliance.hs-sites.com/lightweight-m2m-specification-from-oma

[11] Eclipse Leshan Proposal - https://projects.eclipse.org/projects/iot.leshan

[12] Leshan GitHub - https://github.com/eclipse/leshan

[13] Eclipse Californium – https://www.eclipse.org/californium/

[14] JSON Specification - http://www.json.org/json-es.html

[15] Eclipse Wakama - https://github.com/eclipse/wakaama

[16] Oracle JMF - http://www.oracle.com/technetwork/java/javase/tech/jmf-140515.html

[17] Sven Fleck, Florian Busch, Peter Biber, Wolfgang Straßer, "3D Surveillance – A Distributed Network of Smart Cameras for Real-Time Tracking and its Visualization in 3D". *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop*, IEEE, 2006.

[18] Zihan Wang. "A Network-Centric discussion of the Internet of Things". *University of Birmingham*, 2014.

[19] YANG, H. and YANG, S.H. "RFID sensor network architectures to integrate RFID, sensor and WSN". *Measurement and Control*, 40 (2), pp.56-59, 2007.

[20] Aghajan, H. and Cavallaro, A. "Multi-Camera Networks. Principles and Applications". *Elsevier*, 2009. ISBN: 978-0-12-374633-7.

[21] Mitra, K.; Veeraraghavan, A.; Baraniuk, R. G.; Sankaranarayanan, A. C. "Toward Compressive Camera Networks". IEEE Computer Society, 2004.

[22] OMA LWM2M Public Review - https://github.com/OpenMobileAlliance/OMA-LwM2M-Public-Review/wiki

[23] Qureshi, F.; Terzopoulos, D. Smart Camera Networks in Virtual Reality. Proceedings of the IEEE Vol. 96, No. 10, October 2008.

[24] Quaritsch, M.; Kreuzthaler, M; Rinner, B.; Bischof, H.; Strobl, B. "Autonomous Multicamera Tracking on Embedded Smart Cameras". *EURASIP Journal on Embedded Systems* Volume 2007, Article ID 92827.

[25] Poslad, S. "Ubiquitous Computing: Smart Devices, Environments and Interactions". *Wisley*, 2009. ISBN: 978-0-470-03560-3.

[26] Abad, I.; Cerrada, C.; Cerrada, A.; Heradio, R.; Valero E. "Managing RFID Sensors Networks with a General Purpose RFID Middleware". *Sensors* 2012, 12, 7719-7737.

8. Siglas, abreviaturas y acrónimos

API – Application Programming Interface (Interfaz de Programación de Aplicaciones)

CoAP – Constrained Application Protocol

DTLS – Datagram Transport Layer Security

IoT – Internet of Things

Jar – **J**ava **Ar**chive

JMF – Java Media Framework

JSON – JavaScript Object Notation

LWM2M – Lightweight Machine to Machine

M2M – Machine to Machine

OMA – Open Mobile Alliance

OMNA – Open Mobile Naming Authority

RAM – Random Access Memory

RFID – Radio Frequency Identification

RGB – Red Green Blue

SMS – Short Message Service

UDP – User Datagram Protocol

UI – User Interface (Interfaz de Usuario)

URL – Uniform Resources Locator (localizador uniforme de recursos)

WSN – Wireless Sensor Network (Redes de Sensores Inalámbricos)

YUV – es un sistema de procesamiento de imágenes en color, donde Y es la componente de brillo y UV son las dos componentes de color.

Anexo A. Inclusión de nuevos objetos en Leshan

Tal y como se describe en la sección 5.1.1, los siguientes objetos han sido añadidos a la especificación json del proyecto Leshan:

Device Capability Management (ID = 15)


```
{
"name": "Device Capability Management",
"id": 15,
"instancetype": "multiple",
"mandatory": false,
"description": "",
"resourcedefs": [
        {
                "id": 0,
                "name": "Property",
                "operations": "R",
                "instancetype": "single",
                "mandatory": true,
                "type": "string",
                "range": "",
                "units": "",
                "description": ""
        },
        {
                "id": 1,
                "name": "Group",
                "operations": "R",
                "instancetype": "single",
                "mandatory": true,
                "type": "integer",
                "range": "",
                "units": "",
                "description": ""
        },
        {
            "id": 2,
            "name": "Description",
            "operations": "R",
            "instancetype": "single",
            "mandatory": false,
            "type": "string",
            "range": "",
```

```
        "units": "",
        "description": ""
    },
    {
        "id": 3,
        "name": "Attached",
        "operations": "R",
        "instancetype": "single",
        "mandatory": false,
        "type": "boolean",
        "range": "",
        "units": "",
        "description": ""
    },
    {
        "id": 4,
        "name": "Enabled",
        "operations": "R",
        "instancetype": "single",
        "mandatory": true,
        "type": "boolean",
        "range": "",
        "units": "",
        "description": ""
    },
    {
        "id": 5,
        "name": "opEnable",
        "operations": "E",
        "instancetype": "single",
        "mandatory": true,
        "type": "boolean",
        "range": "",
        "units": "",
        "description": ""
    },
    {
        "id": 6,
        "name": "opDisable",
        "operations": "E",
        "instancetype": "single",
        "mandatory": true,
        "type": "boolean",
        "range": "",
```

```
            "units": "",
            "description": ""
        },
        {
            "id": 7,
            "name": "NotifyEn",
            "operations": "RW",
            "instancetype": "single",
            "mandatory": true,
            "type": "boolean",
            "range": "",
            "units": "",
            "description": ""
        }
    ]
}
```


Device Capability Management (ID = 50)


```
{
"name": "Device Data",
"id": 50,
"instancetype": "multiple",
"mandatory": false,
"description": "",
"resourcedefs": [
        {
                "id": 0,
                "name": "Capability",
                "operations": "R",
                "instancetype": "single",
                "mandatory": true,
                "type": "string",
                "range": "",
                "units": "",
                "description": ""
        },
        {
                "id": 1,
                "name": "Timing",
                "operations": "RW",
                "instancetype": "single",
                "mandatory": false,
                "type": "integer",
                "range": "",
```

```
            "units": "",
            "description": ""
        },
        {
            "id": 2,
            "name": "Units",
            "operations": "R",
            "instancetype": "single",
            "mandatory": false,
            "type": "string",
            "range": "",
            "units": "",
            "description": ""
        },
        {
            "id": 3,
            "name": "Datatype",
            "operations": "R",
            "instancetype": "single",
            "mandatory": true,
            "type": "string",
            "range": "",
            "units": "",
            "description": ""
        },
        {
            "id": 4,
            "name": "Raw Data",
            "operations": "R",
            "instancetype": "single",
            "mandatory": true,
            "type": "opaque",
            "range": "",
            "units": "",
            "description": ""
        }
    ]
}
```

Anexo B. LWM2MApp: Estructura de ficheros

La aplicación LWM2MApp contiene varios ficheros .m o funciones MATLAB para abstraer a la aplicación principal (LWM2MApp.m) de operaciones secundarias. La relación entre las distintas funciones se muestra en la Figura 18. Además, a lo largo de este anexo se describe la funcionalidad de cada uno de estos ficheros.

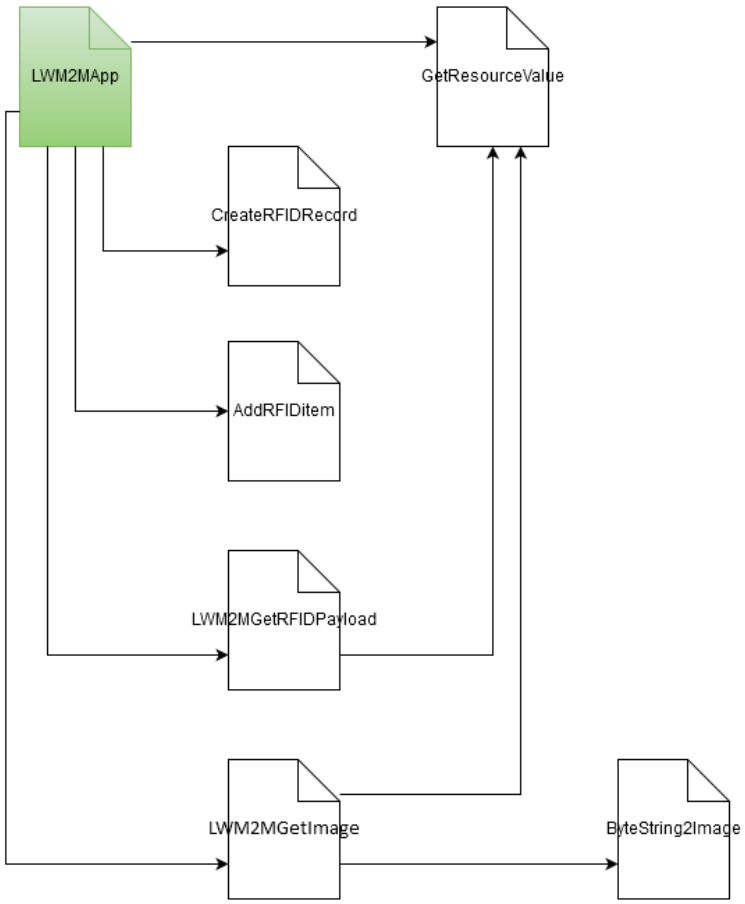

Figura 18. LWM2MApp: Relación entre ficheros.

LWM2MApp.m

LWM2MApp.m es el fichero principal de la aplicación. Es la función principal de la aplicación el resto de ficheros proporcionan funciones secundarias a éste y son llamadas desde esta función. La funcionalidad de LWM2MApp es explicada a lo largo de la sección 5.3.

GetResourceValue.m

Esta es una función que tiene como propósito devolver el valor obtenido de un recurso LWM2M perteneciente a una instancia de un objeto de un cliente determinado.

Como entrada a esta función debe proporcionársele la URL del recurso cuyo valor se busca. La función emplea la función *urlread()* de MATLAB y parsea el resultado para obtener el valor buscado.

CreateRFIDrecord.m

CreateRFIDrecord crea el fichero XML en el que se almacenarán posteriormente los datos del correspondiente lector RFID, registrando el identificador del cliente en la red y la cabecera del fichero XML (fichero que es detallado en Anexo C. LWM2MApp: Formato de fichero XML para lecturas RFID).

Como parámetros de entrada, esta función requiere el directorio donde almacenar el fichero, el índice empleado para clasificar el lector RFID (Reader1, Reader2, etc.) y el identificador del cliente en la red o endpoint.

Una vez creado dicho fichero, la función **AddRFIDitem.m** será llamada para añadir entradas correspondientes a las lecturas RFID.

LWM2MGetImage.m

La función LWM2MGetImage.m es llamada por LWM2MApp.m para obtener la imagen captada por una cámara y su estampa de tiempo.

Su argumento de entrada es la URL que debe apuntar al recurso LWM2M que contiene la imagen (objeto id=50 y recurso 4, es decir 50/<instancia>/4).

Esta función no obtiene la imagen por sí sola, sino que llama a la función **ByteString2Image.m** la cual convierte la cadena de bytes recibida en una imagen apta para ser representada por MATLAB, en función del formato seleccionado, proporcionado por esta función.

ByteString2Image.m

Esta función transforma la imagen recibida del cliente LWM2M en un formato matricial que MATLAB puede representar.

El formato de la imagen (dimensiones de la matriz) es recibido como parámetro de entrada, junto con la cadena de bytes que es la imagen.

LWM2MGetRFIDPayload.m

Esta función análogamente a **LWM2MGetImage.m** ofrece la cadena de bytes resultado de la lectura de una etiqueta RFID por parte de un lector. Añade además, la estampa de tiempo en el que dicha lectura fue recibida.

Su argumento de entrada es la URL que debe apuntar al recurso LWM2M que contiene la lectura del lector RFID (objeto id=50 y recurso 4, es decir 50/<instancia>/4).

AddRFIDitem.m

AddRFIDitem es la función empleada para añadir entradas en el fichero XML correspondientes a lecturas realizadas por lectores RFID, estas entradas son añadidas al fichero XML del lector correspondiente generado a través de **CreateRFIDrecord.m**.

Anexo C. LWM2MApp: Formato de fichero XML para lecturas RFID

En esta sección se recoge el formato creado para almacenar la información recogida por los lectores RFID posteriormente procesada por la aplicación LWM2MApp.

La información recibida por lectores RFID se almacena en ficheros XML versión 1.0 y la codificación es UTF-8. El fichero cuenta con un atributo versión (*Record Version*) para permitir el control de configuración de los ficheros generados, de tal manera que cambios en esta estructura se vean reflejados en un cambio de la versión, haciendo que sea trazable y consistente. Tras esto, en el fichero se refleja propiamente la carga útil del documento. El formato es el siguiente:

recoge el formato creado para almacenar la información recogida por los lectores RFID posteriormente procesada por la aplicación LWM2MApp.

La información recibida por lectores RFID se almacena en ficheros XML versión 1.0 y la codificación es UTF-8. El fichero cuenta con un atributo versión (*Record Version*) para permitir el control de configuración de los ficheros generados, de tal manera que cambios en esta estructura se vean reflejados en un cambio de la versión, haciendo que sea trazable y consistente. Tras esto, en el fichero se refleja propiamente la carga útil del documento. El formato es el siguiente:

<Device> indica el dispositivo que ha transmitido la información. Por tanto, recoge la URL desde la que se ha recibido la trama RFID, el cliente LWM2M.

<!-- Captures --> es un comentario que indica el comienzo de las capturas tomadas para el dispositivo previamente reflejado.

Es tras el anterior comentario donde las tramas RFID se insertan. El formato de una trama RFID es el siguiente:

<Frame Timestamp="*Tiempo en el que se ha recibido la trama*"> VALOR DE LA TRAMA
</Frame>

A continuación se muestra un ejemplo de fichero XML generado:

```
<?xml version="1.0" encoding="UTF-8"?>
-<Record version="2.0">
-<Device>http://localhost:8080/api/clients/35310b01-d992-4316-8236-365125a06df1/50/0/4
<!-- Captures -->
 <Frame Timestamp="2015-09-06T16:51:40+01:00">35AE952985F5202E436B458E23</Frame>
 <Frame Timestamp="2015-09-06T16:51:40+01:00">B92DEB166B458CADE3B785E23</Frame>
<Frame Timestamp="2015-09-06T16:51:41+01:00">EBA560FDCE279F6897FA472323</Frame>
</Device>
</Record>
```