Desarrollo de un prototipo de sistema de detección del volumen de llenado de contenedores de recogida de ropa

Ismael Martin Martín, Gregorio López López, and Sergio Gonzélez Jiménez

Departamento de Electrónica, Automática y Comunicaciones, Universidad Pontificia Comillas ICAI-ICADE

Resumen La recogida actual de los contenedores de ropa de segunda mano que Cáritas tiene distribuidos a lo largo de la Comunidad de Madrid presenta algunas ineficiencias. Mediante la digitalización y monitorización de estos contenedores se pretende resolver estas ineficiencias. Para ello se realiza un estudio de las diferentes alternativas existentes en el mercado y se desarrolla un prototipo para comprobar la funcionalidad del sistema. El sistema se compone de tres partes diferenciadas: un *hardware* o dispositivo compuesto por un sensor de ultrasonidos que envía la capacidad libre del contenedor a través de la red de comunicaciones de *Sigfox* hasta su propio *backend*. Mediante la creación de un *callback* este mensaje es redirigido a la plataforma *cloud* de Azure donde es procesado, almacenado y representado. Mediante el desarrollo de una aplicación web o *frontend* el usuario es capaz de conocer la capacidad de los contenedores y crear rutas optimizadas en base a ello resolviendo las ineficiencias planteadas inicialmente.

Palabras Clave Cáritas, ONG, Digitalización, Monitorización, Contenedores, capacidad, Sigfox, Azure, Arduino MKRFox 1200, Azure IoT Hub, Azure Time Series Insights, Azure Maps, Azure Blob Storage, backend, frontend

1. Introducción

Este estudio se engloba dentro del proyecto Moda Re- de Cáritas, cuyo objetivo es el reciclado de ropa de segunda mano a través de una red de contenedores distribuidos a la red de la Comunidad de Madrid.

Ahora mismo el sistema de recogida de la ropa cuenta con bastantes ineficiencias causadas principalmente por el desconocimiento de cuál es el volumen real de los contenedores. Esto provoca que se recojan contenedores que aún no están llenos o se desborden acumulándose las bolsas de ropa en el exterior. Esto crea un problema de hurto de la ropa y de mala imagen para la organización que puede ser resuelto mediante la implementación de un sistema para conocer la capacidad de los contenedores en tiempo real.

Para ello se plantea un proyecto de colaboración entre Cáritas Madrid y la Fundación Ingenieros ICAI de monitorización de la capacidad libre de los

contenedores. Con él se pretende dotar a la organización de un sistema con el que poder mejorar la eficiencia mediante la optimización de las rutas recogida.

El sistema consistirá en sensores colocados en cada contenedor que realizarán una medición del espacio disponible. Este dato será enviado a través de una red especializada en entornos IoT hasta una plataforma *backend* donde será procesado y almacenado para poder ser representado en un *frontend*.

Este trabajo de fin de máster se engloba dentro de este proyecto. En él se pretende hacer un primer análisis del sistema y la implementación de un prototipo con el que se consigan validar resultados de cara a una posible implementación futura.

2. Análisis del estado del arte y desarrollo de estudio tecnológico para Cáritas

Actualmente existen un gran número de empresas en España y el mundo que se dedican a la implementación de sistemas de monitorización de contenedores. Aunque dependiendo del contenido y la aplicación en particular existen multitud de variaciones, todos los sistemas se caracterizan por estar sustentados sobre tres pilares clave mostrados en la figura 1: una conjunto de dispositivos colocados en cada contenedor que se encargan de evaluar la capacidad libre, una red de comunicaciones que se utiliza para enviar el dato desde cada dispositivo hasta el *backend* donde se procesa, almacena y analiza para poder representarlo en una aplicación web.

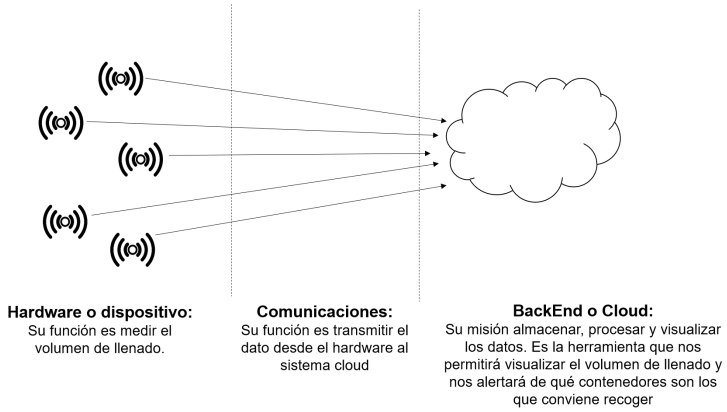

Figura 1: Esquema general de funcionamiento del sistema

Se ha realizado un estudio de cada una de las partes del sistema, analizándose cuáles de las diferentes opciones son óptimas para la aplicación del proyecto, considerándose parámetros tanto tecnológicos como económicos.

El primer elemento a decidir es la red de comunicaciones ya que va a condicionar tanto el desarrollo del dispositivo como el *backend*. Para el sistema de comunicaciones se opta por el uso de las redes LPWAN frente a otras como celular o WLAN debido a sus mejores prestaciones en esta aplicación. Se caracterizan

por tener un bajo ancho de banda pero un alto rango, lo que las hace idóneas para aplicaciones relacionadas con el Internet de las cosas. Emitir a baja frecuencia las permite cubrir grandes áreas, pudiendo llegar a todos los contenedores. Por el contrario, existe una penalización en el ancho de banda pudiendo enviar solo una cantidad de información limitada. De cualquier forma está previsto que se haga una o dos mediciones diarias con muy poco peso cada una, por lo que esto no sería una restricción para la aplicación de los contenedores.

Dentro de estas redes existen dos grupos dependiendo de si la banda de frecuencia utilizada tiene su uso restringido o no. Las redes MNO (*mobile-network Operator* utilizan una red de telefonía ya existente y destacan NBIoT y LTE-M. Aportan seguridad y fiabilidad a expensas de un mayor coste.[1] Por otro lado las principales redes No-MNO son Sigfox y LoRaWAN que presentan ventajas de coste pero se pierde fiabilidad.[2]

Pese a que las características de todas las redes son muy similares para el desarrollo del proyecto se ha optado por utilizar la red de Sigfox que presenta ventajas a la hora de implantación y desarrollo, ya que no es necesario un contrato y su funcionamiento no se hace en base a tarjetas SIMs como NBIoT.

Una vez elegida la red de comunicaciones se ha optado para el desarrollo del prototipo por elegir la plataforma Arduino por su simplicidad y coste. Entre otras opciones se encontraban la compra de un dispositivo que suponía un coste demasiado elevado para el prototipo o el desarrollo de una placa Ad-Hoc lo que conllevaría un alto tiempo y coste. Por ello se ha seleccionado el Arduino MKR 1200 MKRFox 1200 debido a que viene de serie con un chip de comunicaciones de Sigfox, lo que facilita la comunicación sin necesidad de añadir un *shield* extra y resulta económicamente accesible. Para cumplir con la funcionalidad de leer el valor de la capacidad se incorpora un sensor de medición. Además el consumo de batería es bastante bajo para este sistema.

Por último, para el desarrollo de la plataforma de visualización se puede optar por un desarrollo propio o la utilización de un servicio *cloud*. Se opta por esta última opción por sus ventajas en cuanto a coste, mantenimiento, tiempo de puesta en marcha, fiabilidad, seguridad y escalabilidad. La mayoría de las empresas que venden estos servicios cuentan con servicios especializados en IoT. Entre las tres principales están *Amazon Web Services*, Microsoft Azure y Google Cloud, pero se opta por el uso de la plataforma de Microsft Azure para el desarrollo del prototipo debido a su amplia gama de herramientas especializadas en IoT.[3]

3. Diseño y desarrollo de una prueba de concepto de sistema de monitorización remota de ocupación de contenedores de ropa

Tras haber analizado las diferentes posibilidades de implementación se determina que se va optar por el desarrollo de un prototipo del sistema basado en la red de comunicaciones de Sigfox. El objetivo es enviar el dato de capacidad

libre desde un dispositivo hasta un *backend* dónde es procesado, almacenado, analizado y representado en un aplicación web.

Para realizar una correcta medición del volumen es necesario analizar el tipo de sensor que mejor se ajusta a esta aplicación. Entre los tres valorados (sensor de peso, infrarrojos y de ultrasonidos) se opta por este último debido a sus buenas prestaciones y robustez de la medida. Su principal ventaja es la forma cónica de la onda con la que se consigue barrer todo el volumen del contenedor de forma precisa al contrario que el sensor de infrarrojos que tiene una onda lineal. Para el diseño y desarrollo del prototipo se ha optado por utilizar el modelo SRFO4 [4]. Este sensor se ha conectado a una placa Arduino MKRFox 1200 cuya función es interpretar la lectura del sensor y enviarla al *backend* a través de la red de comunicaciones de Sigfox (la propia placa incluye un chip para facilitar las comunicaciones con la red [5]).

Para comprobar el correcto funcionamiento del dispositivo se realizan dos ensayos. En el primero se calibra el sensor de ultrasonidos realizando diferentes mediciones a distancias conocidas. Se comprueba que la fórmula 1 teórica [6] para obtener la distancia en función del tiempo entre envío y recepción de onda es correcta , existiendo un pequeño error de medida que se puede considerar despreciable ya que no afecta a la aplicación.

$$D(m) = \frac{t(s)}{2x343m/s} \tag{1}$$

En el segundo ensayo se somete al dispositivo a condiciones reales dentro de un contenedor. El objetivo es determinar si existen interferencias con las paredes debido a la forma cónica de la onda. Se observa que existe un pequeño margen de error máximo de 6 cm cuando la distancia es menor de 70 cm. Pese a ello, se considera correcto el funcionamiento para la aplicación ya que no se requiere una gran precisión.

Para la parte de comunicaciones la red utilizada es la red de Sigfox. Esta red se caracteriza por utilizar una banda de frecuencia libre de 868 MHz y estar dentro de la tecnología UNB (*Ultra Narrow Band*) teniendo bajas velocidades de transferencia, del orden de entre 10 y 1000 bits por segundo, pero pudiendo cubrir grandes áreas (llegando a los 25 Km en campo abierto).[5]

Al estar en una banda libre uno de los principales problemas que aparece es la seguridad y fiabilidad. Para garantizarlas, cada mensaje se repite tres veces con diferentes números de serie, asegurando así su correcta recepción [7], y limitando el número máximo de envíos a 144 mensajes diarios [8]. Otra de las principales ventajas de la red es que permite la localización de los contenedores geográficamente mediante triangulación, obteniendo unas coordenadas aproximadas en el *backend* de Sigfox. Como se ha comprobado durante el desarrollo del prototipo, se obtiene la localización con demasiado margen de error (entorno a 1 kilometro) por lo que no se pueden localizar los contenedores basándose en la triangulación de Sigfox.

En la figura 2 se muestra el flujo que sigue la información. El Arduino MKRFox 1200 realiza el envío del dato hasta el *backend* de Sigfox. Las fun-

cionalidades de esta plataforma son muy limitadas, por lo que se hace necesario realizar un reenvío de la información hasta una plataforma *cloud* que permita opciones de procesamiento y visualización. Esto se va a hacer mediante un *callback* personalizado predefinido por el entorno, que conecta el *backend* de Sigfox con la herramienta de Microsoft *Azure IoT Hub*. Este *callback* es creado mediante un *HTTP Request* de tipo *POST* que envía un archivo JSON con las variables predifinidas a las IP definida mediante la *connection string* del dispositivo del *IoTHub* creado previamente.[9]

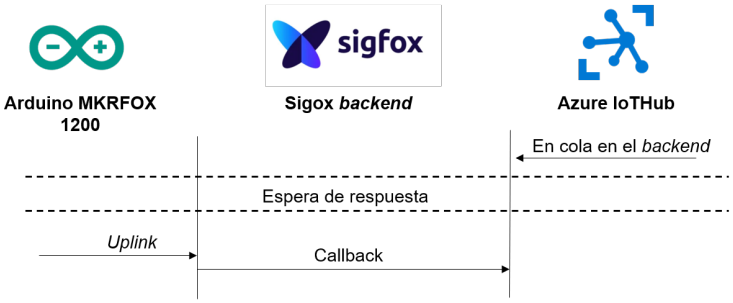

Figura 2: Flujo de las comunicaciones

Las variables enviadas en este *callback* son el valor de capacidad recibido desde el dispositivo, las coordenadas reales obtenidas mediante triangulación de la señal en la plataforma de Sigfox y las coordenadas supuestas en las que se tiene que encontrar el contenedor.

Finalmente esta información tiene que ser procesada, almacenada y analizada. Para ello se va a utilizar la plataforma de Microsoft Azure que cuenta con herramientas que permiten el tratamiento del dato de forma sencilla, segura y escalable a diferentes niveles. Además, posibilita la creación de una aplicación web en la que el usuario es capaz de ver el estado real de la capacidad libre de los contenedores así como su representación en un mapa.

La figura 3 muestra el flujo de información que sigue el prototipo, así como las herramientas utilizadas dentro de Azure para el desarrollo de la *app* de visualización. La información se ingesta en la plataforma de Azure a través de la herramienta especializada *Azure IoT Hub* desde dónde se reenvía el mensaje a la herramienta Azure Time Series Insight, que permite una visualización y tratamiento temporal de la información. Además, también también se utiliza la herramienta *Azure Blob Storage* para realizar un almacenamiento en frío (atemporal pero de acceso lento) y un almacenamiento en caliente (con una duración de 7 días pero acceso rápido) del que se alimenta *Azure Time Series Insights*.Finalmente, mediante una API, Azure Maps hace una petición de la información almacenada y procesada en *Azure Time Series Insights* que luego utilizará la app de visualización desarrollada.

La herramienta *Azure IoT Hub* actúa como centro de mensajes entre la aplicación y los dispositivos, permitiendo la ingesta de grandes volúmenes de datos de telemetría desde los dispositivos.[10] Su funcionamiento se basa en la creación

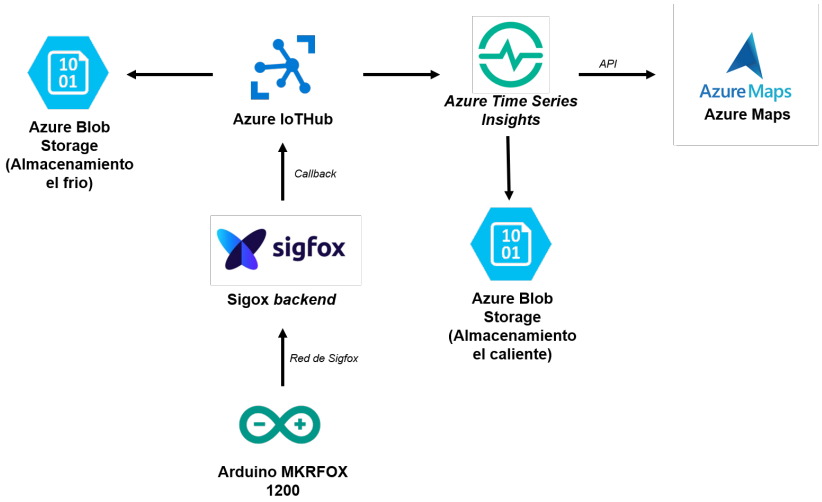

Figura 3: Flujo de la información en el sistema

de dispositivos virtuales con su dirección IP que se encontrarán conectados a los dispositivos reales.

Para el prototipo se crean dos dispositivos diferentes: un dispositivo real que corresponde al dispositivo creado previamente mediante la placa Arduino; y un dispositivo simulado que se utilizará para facilitar las labores de testeo y que será creado mediante una máquina virtual dentro de Azure.

Esta máquina virtual tiene alojado un fichero en Python que simula el funcionamiento del dispositivo del contenedor, enviando unos valores de capacidad aleatorios y unas coordenadas geográficas predefinidas. Utilizando la *connection string* del dispositivo creado en *Azure IoT Hub* es posible enviar los datos cada vez que el fichero sea ejecutado, consiguiendo un comportamiento idéntico al dispositivo real.

Una vez que la información ha sido recibida en Azure ha de ser almacenada. Para ello la plataforma cuenta con la herramienta *Azure Blob Storage* que almacena la información en forma de *blobs*, estructuras de datos que no se ciñen a ningún modelo de datos o definición concreto como texto o datos binarios.[11]. Dentro de esta herramienta se van a utilizar dos tipos de almacenamiento.Por un lado, se va a utilizar un almacenamiento en frío que permite un uso atemporal de la información, aunque tiene un acceso a ella más lenta. El enrutamiento de esta información se hará mediante un *message brokering*, que consiste una asignación de un *endopoint* el contenedor para almacenamiento de la información. [12]. Por otro lado, toda la información se va a guardad en un almacenamiento en caliente durante un periodo de 7 días, ya que la herramienta *Azure Time Series Insights* necesita un acceso rápido a los datos que solo se consigue con el almacenamiento en caliente.

Para realizar un procesamiento temporal y poder visualizar los datos es necesario el uso de otra herramienta de Azure: *Azure Time Series Insights*. Se trata de una plataforma como servicio (Paas) que se encarga de la recopilación,

procesamiento, análisis y consulta de datos obtenidos de los contenedores. Está diseñada principalmente para las necesidades del IoT industrial, con herramientas como almacenaje multicapa, modelado de series temporales o *queries* de coste reducido.[13]

Para la configuración de esta herramienta se ha tenido en cuenta que el entorno utilizado es PAYG (*Pay-as-you-go*)[14] en el cual se paga en función de la ingesta de datos, las series temporales son identificadas en base al *device id* de cada dispositivo y se va a crear un almacenamiento en caliente para almacenar la información de manera que pueda ser procesada de una manera eficiente.

La ultima herramienta utilizada es Azure Maps que proporciona funcionalidades geoespaciales mediante el uso de mapas con el objetivo de proporcionar un contexto geográfico para la aplicación del contenedor.[15] La suscripción es gratuita y solo tiene un coste cuando el número de peticiones que se realiza es muy elevado.

El objetivo final del sistema es poder conocer el estado del volumen de los contenedores en tiempo real para poder optimizar las rutas de recogida. Es por ello que se hace necesario el desarrollo de un *frontend* o aplicación web mediante la cual el usuario es capaz de visualizar la información y manipularla de forma sencilla. El desarrollo de ella se ha basado en el proyecto de Microsoft *Azure IoT Workshop: Real-Time Asset Tracking* [16]. La aplicación web se desarrolla en un entorno HTML al que se le han aplicado varias funcionalidades utilizando Javascript. Muchas de estas funcionalidades se han basado en ejemplos de Microsft SDK de Azure Maps.[17] Para obtener los datos, tanto espaciales como temporales, se hace mediante una llamada API a los rescursos de *Azure Time Series Insights* y *Azure Maps*.

Las principales funcionalidades de la aplicación son la obtención del dato de capacidad, la localización de los contenedores en un mapa, la representación visual del estado mediante un código de colores, la representación de gráficas temporales que muestren la evolución y la creación de rutas entre los puntos que necesiten ser recogidos.

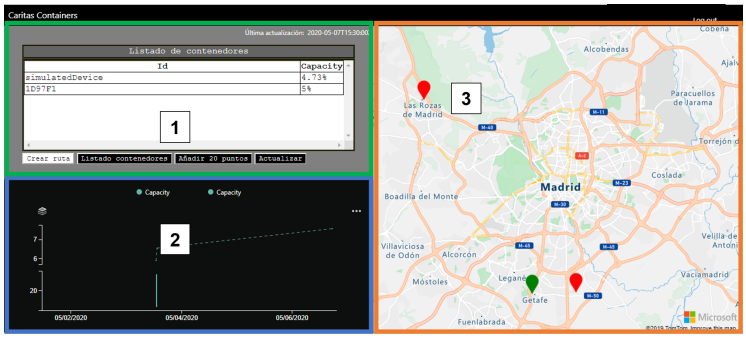

Figura 4: Estructura de la interfaz de usuario de la aplicación web

En la aplicación web mostrada en la figura 3 se pueden distinguir tres áreas diferenciadas: una interfaz de usuario en la que se visualizan los contenedores y su capacidad y unos botones que permiten acceder a las diferentes opciones de la aplicación (zona 1 - verde), una zona dónde se muestra le evolución temporal

de los diferentes contenedores (zona 2 - azul) y un mapa en el que se representan geográficamente los contenedores y su capacidad en base a un código de colores (zona 3 - amarilla).

El funcionamiento básico de la aplicación viene recogido en el esquema representado en la figura 5. Al ejecutarse la aplicación se obtienen los datos de capacidad y localización desde *Azure Time Series Insigts* y con ellos se crean las marcas en el mapa, se escriben los valores en la tabla de la interfaz de usuario y se dibujan las series temporales. En este momento el programa se queda en posición de espera hasta que se presiona algún botón de la interfaz de usuario que ejecuta alguna funcionalidad de al aplicación.

Si se presiona el botón *Crear ruta* se crea una ruta pasando por todos aquellos contenedores cuyo valor de capacidad libre sea inferior al 20 %, comenzando y acabando en la nave base de Getafe. Esta ruta se representa en un mapa y en la tabla de la interfaz de usuario aparecen los contenedores en orden de recogida. Si se presiona el botón *listado de contenedores* se volverá a la tabla inicial mostrando todos los contenedores independientemente de si han superado su capacidad límite o no.

Para realizar la comprobación del correcto funcionamiento de la aplicación se ha añadido la opción de *Añadir 20 puntos* que crea 20 contenedores con una capacidad aleatoria. Finalmente el botón de *Actualizar* repite el proceso de lectura y ejecuta la aplicación con los últimos valores obtenidos.

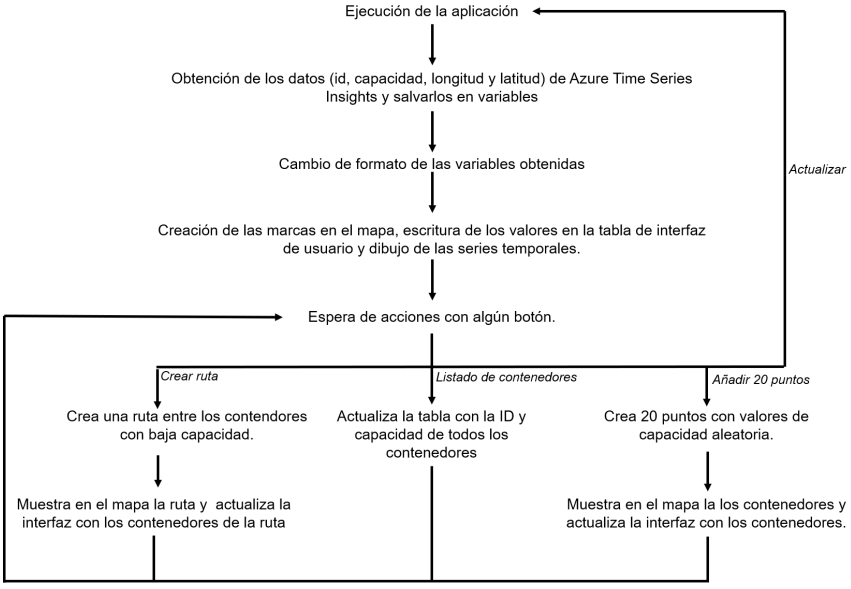

Figura 5: Esquema funcionamiento de la aplicación web

4. Validación del sistema desarrollado

El objetivo de este trabajo de Fin de Máster es realizar un primer análisis de la viabilidad del proyecto de Cáritas de monitorización de los contenedores de ropa que tienen distribuidos a lo largo de toda la Comunidad de Madrid. El objetivo final es, por tanto, validar la implantación futura de un sistema que permita a la organización mejorar la eficiencia en la recolección de ropa de los contenedores mediante la digitalización. En este apartado se va a realizar una validación funcional del prototipo analizando las posibles áreas de mejor para los diferentes componentes del sistema.

Para el diseño de dispositivo *hardware* se considera correcta la elección del sensor de ultrasonidos ya que realiza mediciones con una precisión suficiente y la geometría cónica de la onda la hace ideal para el contenedor. Además se ha comprobado que el rango de 3 m del sensor utilizado [4] es suficiente para la altura de 2 m del contenedor y en los ensayos realizados se ha demostrado que no existe una influencia negativa del contenedor.Existe un margen de error cuando la distancia es inferior al 70 % que puede llegar a un error absoluto de 6 cm. De cara a una posible mejora del dispositivo se puede optar por el uso de un sensor de ultrasonidos con une mejor precisión aunque esto supondría un aumento del coste del dispositivo.

Como red de comunicaciones se ha optado por el uso de las redes LPWAN que ofrecen un largo alcance a costa de disminuir el ancho de banda. Para la aplicación del proyecto donde se pretende hacer uno o dos envíos al día de poco peso esto no supone un impedimento. Se ha optado por el uso de la red de LPWAN Sigfox, que utiliza una banda de frecuencia libre. El principal problema encontrado a la hora del desarrollo ha sido la pérdida de mensajes. En las zonas donde existe una buena cobertura el mensaje se enviaba y recibía, pero en otros ensayos dónde la cobertura se ha visto reducida se ha producido la pérdida de mensajes. Esto es un error muy grave a tener en cuenta ya que puede ocasionar graves problemas por la organización al pasar por alto contenedores que están llenos en las rutas de recogida, haciendo el sistema inservible. Para resolver este problema hay varias soluciones planteables. Por un lado, se puede optar por la creación de una comunicación *downlink* cuando se espere un mensaje y no se reciba, asegurando así un reenvío del dato cuando no se hay enviado correctamente, asegurarando la comunicación. Otra opción es cambiar el diseño del sistema y utilizar otra red LPWAN como NBIoT. La última opción consiste en la implementación de repetidores de señal cerca de los contenedores donde la cobertura sea baja con el objetivo de asegurar una buena comunicación.

Por otro lado, también se ha comprado que las coordenadas del dispositivo obtenidas durante el desarrollo del prototipo por parte de la plataforma son muy poco precisas en los casos de buena cobertura (obteniéndose un error de más de 1Km) y nulos en las áreas donde la cobertura es baja. Esto hace imposible el utilizar la localización de la red para determinar la posición de los contenedores. Como solución a este problema, el prototipo se ha desarrollado de forma que asociado a cada ID del contenedor vengan una coordenadas geográficas. De esta

manera la localización obtenida por Sigfox solo se utiliza en caso de robo del contenedor y que la distancia haya sido mucho más alta.

De cualquier forma, la red cumple con los requisitos del sistema en cuanto a ancho de banda y cobertura en todas las zonas, presenta la ventaja de permite una comunicación bidireccional (aunque más limitada)[8] en caso de que sea necesario comunicarse con el dispositivo. Además presenta un menor coste asociado frente al resto de redes de comunicación. El propio *backend* de Sigfox recibe todos los mensajes y es capaz de redireccionarlos de forma automática al *backend* de Azure. Se ha comprobado la existencia de un cierto retardo de 1 minuto entre el envío desde el *backend* de Sigfox hasta la plataforma de Azure, inexistente cuando se hace el envío desde la máquina virtual con el dispositivo simulado. De cualquier forma este retardo no se tiene en consideración ya que durante todos los ensayos realizados no se ha producido ninguna pérdida del mensaje.

Para el almacenaje y procesado de los datos se ha optado por el uso de una plataforma *cloud* frente al desarrollo de un *backend* propio por sus ventajas de fiabilidad, mantenimiento y escalabilidad. Se han utilizado e integrado las herramientas *Azure IoT Hub*, *Azure Time Series Insights*, *Azure Blob Storage* y *Azure Maps* de Microsfot Azure para recibir, procesar y almacenar los datos de capacidad enviados tanto desde el dispositivo desarrollado como desde el dispositivo simulado con la máquina virtual. Los resultados obtenidos son muy buenos ya que son herramientas que permiten una fácil integración a un coste bajo y una alta escalabilidad.

El núcleo del proyecto es la aplicación web ya que es la herramienta que el usuario va a utilizar para conocer el volumen de los contenedores en tiempo real y poder planificar rutas acorde a ello. Para la comprobación del correcto funcionamiento se ha desarrollado un modo de testeo que crea 20 contenedores con una capacidad aleatoria comprendida entre 0 % y 100 %. En este modo el programa también es capaz de crear rutas seleccionando solo aquellos contenedores cuya capacidad libre sea inferior al 20 %.

Entre las futuras ampliaciones necesarias que requiere la aplicación es el desarrollo de rutas óptimas. Las rutas obtenidas en esta primera versión siguen un orden de creación en base a un listado que no es el óptimo, ya que este necesitaría el de algoritmos que se van fuera del alcance de una primera validación del proyecto.

Finalmente, se comprueba que el prototipo cumple con los requisitos del proyecto de ser capaz de dotar a la organización de un sistema con el que monitorizar la capacidad libre de los contenedores de ropa de segunda mano con el objetivo de crear rutas óptimas y mejorar la eficiencia del sistema de recogida. Este sistema tiene un coste económico bajo ya que los costes asociados del *hardware*, red de comunicaciones y *backend* han sido optimizados. Tiene una interfaz de usuario *user-friendly* por lo que la aplicación es accesible a personal de la organización con cualquier tipo de cualificación. Además, presenta una alta adaptabilidad y escalabilidad gracias al uso de plataformas *cloud* que permiten la realización de modificaciones de una forma sencilla sin necesidad de alterar el sistema.

5. Conclusiones y trabajos futuros

Cáritas Madrid actualmente tiene distribuidos a lo largo de la Comunidad de Madrid alrededor de 165 contenedores en los que se recoge ropa de segunda mano que se destina a población que se encuentra en riesgo de exclusión social. Con el objetivo de conseguir una mayor eficiencia en la recogida de estos, se plantea el desarrollo de un sistema que permita monitorizar la capacidad libre para ropa restante. El objetivo de este trabajo de fin de máster es la realización de un primer análisis de este sistema.

Para la elaboración de este análisis se han seguido dos fases: desde un análisis del estado tecnológico del sistema hasta el desarrollo de un prototipo a pequeña escala y su validación funcional para la aplicación.

En un primer momento se ha hecho un estudio exhaustivo de cual es el estado tecnológico de este tipo de tecnología. Para ello, se ha comparado los diferentes proveedores de este tipo de sistemas, determinando cuáles son las características que tienen en común (principalmente alimentación por batería y el uso de sensor de ultrasonidos para realizar la medición de la capacidad libre). También se ha realizado un análisis de cuales son las diferentes tecnologías, en caso de realizar un sistema desarrollado *ad-hoc*, que se ajustan al proyecto a nivel de *hardware*, sistema de comunicaciones y *backend*. Finalmente, se determina el uso del sensor de ultrasonidos y la red de comunicaciones de Sigfox.

Se desarrolla un prototipo que cuenta con las tres partes funcionales de este sistema: un dispositivo que realiza la medición, una red de comunicaciones por la cual enviar esta información y una plataforma en la que poder procesar, almacenar y visualizar la información a tiempo real.

El objetivo del desarrollo del prototipo es la creación de un entorno de prueba en el que comprobar la funcionalidad de este sistema de cara a una posible puesta en marcha futura. Se ha comprobado que la implantación de un sistema tecnológico que permita la recolección de los contenedores de ropa de una forma más eficiente puede traer consigo numerosas ventajas para una organización sin ánimo de lucro como Cáritas. Por un lado se conseguiría mejorar la imagen de la organización. Actualmente uno de los principales problemas que tienen es que la ropa no se recoge en los contenedores y acaba acumulándose en bolsas en el exterior de los mismos. Esto crea una mala imagen de la organización y el hurto de la mejor ropa. Mediante este sistema se evitaría la acumulación y mejoraría la imagen de la ONG.

Por otro lado, también se conseguiría hacer un uso más eficiente de los recursos de la organización. Otro de los problemas que tienen es que se recogen contenedores que aún están vacíos. Con la ayuda del sistema implantado se podrían optimizar las rutas para recoger solo aquellos contenedores cuya capacidad sea equivalente a un 80 %, consiguiendo una reducción del número de ellas. Esto implicaría un ahorro económico, ya que supondría un menor gasto en combustible y de personal que realizan las rutas, y una redistribución de los recursos humanos a tareas dónde puedan aportar un mayor valor a la organización consiguiendo crear un mayor impacto a la sociedad.

Con este proyecto se quiere demostrar que el impacto de la digitalización y el desarrollo tecnológico se puede aplicar a todos los sectores de la sociedad. Una organización sin ánimo de lucro también se puede ver altamente beneficiada mediante la aplicación de nuevas tecnologías en sus labores humanitarias.

Este proyecto se va a presentar a la conferencia iberóamericana sobre ciudades inteligentes que se realiza el próximo mes de Noviembre en Costa Rica[18]. Con el se quiere concienciar en la necesidad de implantar desarrollos digitales en las ciudades, independientemente de la organización o sus recursos.

Entre los principales trabajos a futuro queda la implementación y desarrollo del sistema completo. Para ello es imprescindible solucionar los problemas con la red de Sigfox encontrados durante el desarrollo de este trabajo de fin de máster y desarrollar un algoritmo con el que se consiga la optimización de las rutas.

A largo plazo es necesario estudiar la viabilidad del proyecto con la empresa y comenzar el desarrollo con una fase piloto en la que solucionar los posibles problemas que aparezcan. Será entonces cuando se hará una puesta en marcha a gran escala ocasionando un gran beneficio para Cáritas.

6. Agradecimientos

Este proyecto se ha llevado a cabo gracias a la colaboración entre la ONG Cáritas Madrid que plantea el problema, la Fundación Ingenieros ICAI que canaliza el problema y le intenta dar respuesta y la Universidad Pontificia Comillas con la cual se realiza este trabajo de Fin de Master para darle solución.

No podría haber sido posible realizar este trabajo sin contar con el apoyo del director del TFM, Gregorio López López, que ha asistido en todo momento dando apoyo tecnológico y creativo al trabajo. También agradecer a Sergio González Jiménez por su colaboración como codirector.

Agradecer también en especial a Marta Reina, gerente de la Fundación Ingenieros ICAI, por su actuación para conseguir el proyecto y su papel mediador entre el alumno y la ONG.

Referencias

[1] Raquel Ligero. “Diferencias entre NB-IoT y LTE-M”. En: *accent systems* (mayo de 2018). https://accent-systems.com/es/blog/diferencias-nb-iot-lte-m/.

[2] Brian Ray. “Sigfox vs. LoRa: A comparison between technologies = Business models”. En: *Link Labs* (mayo de 2018). https://www.link-labs.com/blog/sigfox-vs-lora.

[3] José Miguel Álvarez Vañó. “Modelo Comparativo de Plataformas Cloud y Evaluación de Microsoft Azure, Google App Engine y Amazon EC2”. Trabajo de fin de grado, Grado en Ingeniería Informática. Escola Tècnica Superior d’Enginyeria Informàtica, Universitat Politècnica de València, 2018.

[4] *Documentación del sensor de ultrasonidos SRF04*. SRF04 http://www.robot-electronics.co.uk/htm/srf04tech.htm.

[5] *Aprendiendo Arduino: Sigfox*. https://www.aprendiendoarduino.com/2018/03/05/arduino-y-sigfox/. Aprendiendo con arduino. Mar. de 2018.
[6] Luis Llamas. "Medir distancia con Arduino y sensor de ultrasonidos HC-SR04". En: (jun. de 2015). https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/.
[7] *Todo lo que necesitas saber sobre la seguridad de Sigfox*. https://www.wnd-group.io/2017/02/17/todo-lo-que-necesitas-saber-sobre-la-seguridad-de- sigfox/. WND Group.
[8] *Sigfox: Asses your project's needs*. https://build.sigfox.com/study.
[9] *Sigfox documentation: Callback Api*. https://backend.sigfox.com/apidocs/callback.
[10] *Microsft Azure Documentation. Azure IoT Hub: ¿Qué es Azure IoT Hub?* https://docs.microsoft.com/es-es/azure/iot-hub/about-iot-hub. Microsoft Azure. Ago. de 2019.
[11] *Microsft Azure Documentation. Introducción a Azure Blob Storage*. https://docs.microsoft.com/es-es/azure/storage/blobs/storage-blobs-introduction. Microsoft Azure. Ago. de 2019.
[12] Sergio González Jiménez. *Azure IoT Hands on Lab*. Url: https://github.com/SeryioGonzalez/azure-iot. IoT - Universidad Pontificia Comillas ICAI. Madrid, nov. de 2020.
[13] *Microsft Azure Documentation. Azure Time Series Insights: ¿Qué es la versión preliminar de Azure Time Series Insights?* https://docs.microsoft.com/es-es/azure/time-series-insights/time-series- insights-update-overview. Microsoft Azure. Ago. de 2019.
[14] *Documentación Microsoft Azure. Detalles de precios de Azure Time Series Insights*. https://azure.microsoft.com/es-es/pricing/details/time-series -insights/. Microsoft.
[15] *Microsft Azure Documentation. Azure Maps. ¿Qué es Azure Maps?* https://docs.microsoft.com/es-es/azure/azure-maps/about-azure-maps. Microsoft Azure. Ene. de 2020.
[16] *Microsft Azure Documentation. Azure IoT Workshop. Real-Time Asset Tracking*. https://github.com/Azure/iot-workshop-asset-tracking. Microsoft Azure.
[17] *Azure Maps Web SDK Samples*. https://azuremapscodesamples.azurewebsites.net/. Microsoft.
[18] *III Congreso Iberoamericano de Ciudades Inteligentes:* http://www.icsc-cities2020.com/ES_index.html.