Sistema domótico IoT basado en Raspberry Pi y control remoto por Telegram

Jesús Gómez Bellido

Abstract—The abstract goes here.

Index Terms—Domótica, IoT, Telegram, API, JavaScript, NodeJS, ARP.

1 Introduccion y objetivos

En este Trabajo fin de Máster (TFM) se tratará de mostrar el impacto que pueden generar las nuevas tecnologías que están comenzando a expandirse. Estas nuevas tecnologías tienen en mente una perspectiva, el llamado "Internet of Things" (IoT) o en español, el Internet de las Cosas. Este es un término se refiere a la interconexión de dispositivos físicos, vehículos, edificios y otros objectos – embebidos con electrónica, software, sensores, actuadores y conexión a internet que permiten la recolección de datos. Todo esto permite que los ordenadores interactúen con elementos de la vida real y ganen independencia de los seres humanos.

Bien es cierto, que el IoT va a suponer un gran impacto en cuanto a la industria y la investigación, pero no será menos para los ambientes domésticos ya que nos permite automatizar muchas funciones de nuestros hogares. En este entorno tiene gran parte de importancia el uso de los *smartphones*, pues son en muchos casos los encargados de comunicar a los seres humanos con nuestros dispositivos.

Otro de los dispositivos en auge y que han fomentado la automatización en los hogares son los micro-ordenadores, como son las Raspberry Pi, estos son dispositivos tremendamente versátiles y cada vez más potentes.

1.1 Objetivos

En el actual TFM, vamos buscar unos objetivos basándonos en IoT en un entorno doméstico.

Se realizará un sistema domótico básandonos en los principios del IoT.

Para llevar a cabo este primer objetivo, se va a usar una Raspberry Pi programandose con NodeJS, un entorno de ejecución para JavaScript, y viendo que este lenguaje puede ser una alternativa real a Python, el cuál es el lenguaje de programación más extendido para la Raspberry Pi.

El sistema domótico, realizará el control sobre persianas/toldos, basándose en las previsiones de servicios meteorológicos, prevaleciendo las acciones del usuario. El usuario también tendrá control sobre puertas, luces y la alarma.

Por otro lado también se tendrá un control de presencia dentro de la casa, registrando las entradas y salidas de los usuarios mediante la MAC de su smartphone y el protocolo ARP.

Por último, el control de nuestro sistema domótico se realizará mediante Telegram, las aplicaciones de mensajerías son algo indispensable hoy en día para las personas, así que viendo la API que este servicio de mensajería nos proporciona para realizar bots, parece interesante estudiar qué clase de posibilidades se nos abren con estos tres elementos.

2 Arquitectura del sistema

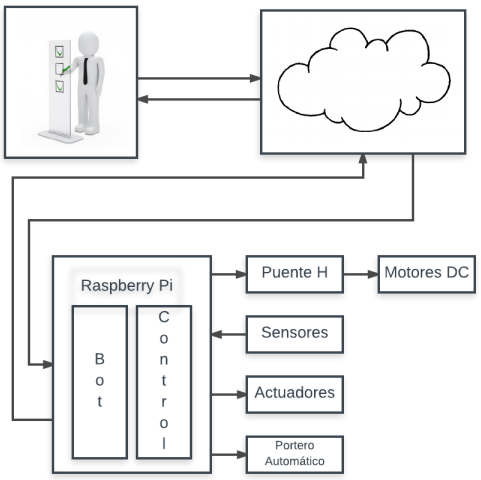

Fig. 1: Arquitectura del sistema

En la figura 1 se muestra la arquitectura del sistema que se va a desarrollar. Se puede ver como el usuario no tiene contacto directo con el sistema. Pues aunque se encuentre la Raspberry Pi y el usuario en la misma red, la comunicación se establece con el servidor de telegram de por medio.

Una vez que el usuario envía un comando y el bot lo recibe, éste provoca un evento y envía la orden hacia la lógica de control. La cual actuará en consecuencia de su alrededor.

De igual forma que el usuario envía comandos al sistema, el sistema enviará al usuario los diferentes datos que

recoge por los sensores y servicios que se estén manejando en ese momento.

El componente principal del sistema será una Raspberry Pi, en concreto usaremos la versión 3. La Raspberry Pi se puede definir como un ordenador de placa reducida. Esta en su interior cuenta con un procesador ARM de 64bits con 4 núcleos a 1,2GHz. Cuenta con 1GB de RAM, conectividad Wi-Fi y bluetooth.

Se puede decir que los demás componentes del sistema son genéricos, por lo que no dependemos de ningún componente concreto.

3 Desarrollo Software

Para el desarrollo del software se va a utilizar NodeJS. La comunicación con telegram se llevará a cabo mediante uno de los muchos módulos que se pueden encontrarar para este lenguaje. En concreto, este módulo se utilizará para interactuar con el Bot API oficial de Telegram. Este módulo nos proporciona los métodos y eventos principales a la hora de comunicarnos con el bot.

En base a los comandos que el usuario envía, se va a desarrollar una máquina de estados que nos permitirá interactuar con nuestro sistema.

Las características que se van a incluir en nuestro sistema domótico son las siguientes:

- **Control de usuarios:** el acceso a los comandos de nuestro sistema queda restringido a usuarios autorizados. Se gestionarán dos tipos de permisos, administradores y usuarios.
- **Control de presencia:** el sistema domótico mediante el protocolo ARP, puede saber los dispositivos que se encuentran en ese momento en la red. Teniendo en cuenta que según datos estadísticos el 51% de la población mundial tiene un smartphone. Podemos tener constancia de quien se encuentra en casa y guardar un registro de entrada y salida.
- **Control remoto del portero automático:** se dispondrá de una cámara en nuestra puerta la cual enviará una foto de quien se encuentra en ella cuando llamen al timbre. Pudiendo también el usuario enviar un mensaje de voz mediante Telegram y que este sea reproducido por el portero automático.
- **Control de temperatura:** mediante el servicio online *openweathermap*, obtendremos la temperatura en nuestra localización
- **Control de alarma:** se desarrollará un sistema de alarma por control remoto, asistido en gran parte por nuestro sistema de control de presencia.
- **Control de sensores y actuadores:** con los 18 GPIO de la Raspberry Pi se pueden combinar diferentes sensores y actuadores.

3.1 API de Telegram

Telegram ofrece dos tipos de API. La *Telegram API*, que permite crear tu propio cliente telegram y la *API Bot*, que permite crear programas dentro de la interfaz de un cliente de telegram. Para el caso que a nosotros nos corresponde nos vamos a centrar en la *API Bot*.

Estos bots son aplicaciones de terceros que corren dentro de un cliente de telegram. Los usuarios podrán interactuar con estas aplicaciones mediante mensajes, comandos, envío de ficheros, mensajes de voz, etc. Que serán interpretados por nuestras aplicación.

3.1.1 ¿Qué se puede hacer con los bots?

Los bots ofrecen diferentes posibilidades, por lo que vamos a enumerar algunas de ellas:

- **Notificación de noticias.** Un bot puede interactuar con el usuario enviando publicaciones de interés tan pronto como estas sean publicadas.
- **Integrar Telegram con otros servicios.** Estos bots pueden incluirse en un chat y hacer peticiones a servicios externos, como por ejemplo: Github, Wikipedia, Youtube, IMDB.
- **Juegos.** Se puede integrar contenido en HTML5, para de esta forma incluir juegos.
- **Crear herramientas.** Herramientas para usar tanto en chats como de forma individual, como por ejemplo: predicciones meteorológicas, traducciones, bots para votar, etc.

3.1.2 ¿Como funcionan los bots?

Los usuarios pueden interactuar con los bots de dos formas diferentes:

- Tratandolo como un *usuario*, podemos abrir un chat con él o agregarlo a un grupo de tal forma que estará siempre escuchando y respoderá sólo cuando se introduzca un comando registrado.
- Ejecutandolo como una acción sobre el teclado, tan sólo tendremos que introducir @botname seguido de la consulta, esto permite usar el bot en cualquier chat o grupo sin necesidad de añadirlo.

Todos los mensajes, comandos y solicitudes enviados por el usuario pasan directamente a la aplicación en ejecución por el usuario. El servidor de Telegram se encarga del cifrado y la comunicación. La aplicación se comunica con el servidor de Telegram con un sencilla interface HTTPS, que es el llamado *API Bot*.

En este punto se pueden distinguir dos procesos diferentes, el envío de mensajes del usuario al bot y el envío de mensajes desde el bot al usuario.

3.1.3 Mensajes del usuario al bot

Este tipo de mensajes que nosotros vamos a recibir en nuestra aplicación, son representados por un objeto JSON. Este objeto tiene 3 campos principales, cada uno representado por un nuevo objeto JSON:

- User. Este objeto contiene los datos del usuario que envía el mensaje. Podemos ver los campos más relevantes en la tabla 1

id	Identificador único de usuario
first_name	Nombre de usuario
last_name	Apellido del usuario
username	Alias único del usuario

Tab. 1: User

- Chat. Este objeto contiene los datos del chat desde el que se envía el mensaje. Podemos ver los campos más relevantes en la tabla 2

id	Identificador único de usuario
type	Tipo de chat, puede ser "private", "group", "supergroup" or "channel"
title	Título del chat

Tab. 2: Chat

- Message. En este objeto encontramos todos los campos referentes al mensaje. Podemos ver los campos más interesantes en la tabla 3

message_id	Identificador del mensaje
from	Identificador del emisor, vacío si viene de un chat
date	Fecha del mensaje
text	Texto recibido en el mensaje
audio	Información sobre el archivo.
document	Información sobre el archivo
voice	Información sobre el archivo
photo	Información sobre el archivo

Tab. 3: Message

Los campos que no contengan información relevante no serán recibidos.

3.1.4 Mensajes del bot al usuario

Al igual que el usuario puede enviar al bot mensajes de texto, audios, mensajes de voz, fotos, etc. el bot puede hacer lo propio hacia los usuarios. La única información necesaria en este caso es el id del usuario. Por ejemplo, para enviar un audio, usaremos el método *sendAudio()*, como argumentos a este método le pasaremos el ID del usuario y la ruta al fichero de audio que se va a enviar.

3.2 Máquina de estados

En la figura 2 se puede ver el diagrama de flujo de la primera toma de decisiones.

Cuando nosotros enviamos un mensaje a nuestro control, la primera premisa que tenemos es ser usuario del sistema, si esto no se cumple directamente seremos expulsados. Si somos usuarios del sistema, se evaluará si hemos enviado una acción o un mensaje, en el caso de ser una acción esta será evaluada como una acción de usuario o de administrador tal y como podemos ver en las figura 3, si no coincide con ninguna de ellas se enviará un mensaje de error.

Además de estas acciones, existe una acción especial */help*, que le muestra al usuario todas las acciones que tiene disponible.

En el caso de que no sea una acción, se evaluará si el controlador está esperando una respuesta de nosotros a una acción anterior. Este proceso forma una nueva máquina de estados que estará controlada por la variable *"estado actual"*, que es independiente para cada usuario. Este tratamiento lo veremos con más detalle en la subsección 3.3

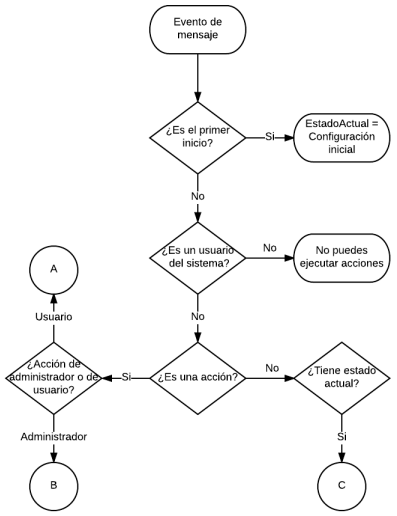

Fig. 2: Máquina de estados Principal

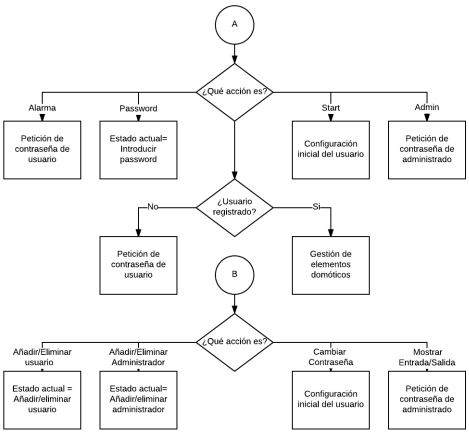

Fig. 3: Máquina de estados Acciones

3.3 Control de usuarios

A continuación, vamos a ver como se gestionan los diferentes usuarios del sistema y cuales son las posibilidades que tiene cada tipo de usuario.

Cada usuario en telegram se identifica por un ID único, que se asigna automáticamente en el momento del registro en la aplicación, y por un alias, también único, el cual se asigna el propio usuario. Nuestro API puede usar los dos identificadores para enviar los mensajes, sin embargo, hemos optado por realizar el envío de de los mensajes siempre a través del ID, pues de esta forma tenemos la certeza de que nunca va a fallar el envío del mensaje.

Los usuarios se almacenan en una variable *users*, identificando a cada usuario por su alias.

La estructura de usuario se puede ver en la fig. 4. A continuación se va a explicar el significado y el uso de cada propiedad:

- Alias: es el nombre con el que se identificarán a los usuarios. Cuando el bot recibe un mensaje de un usuario, recibimos su alias como identificación.

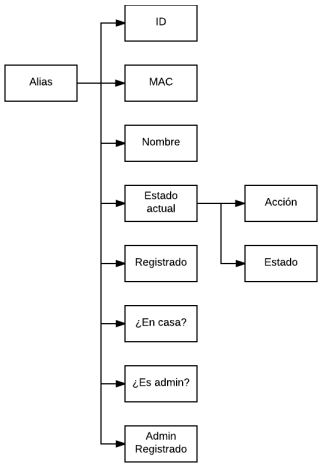

Fig. 4: Estructura de usuario

- ID: número de identificación, que se usará para enviar mensajes a este usuario cuando sea necesario, al igual que el alias lo recibimos en cada mensaje que envía el usuario.
- MAC: aquí almacenaremos la MAC del teléfono del usuario. La MAC la se utilizará en el sistema de control de presencia y la obtendremos habilitando un servidor desde nuestra Raspberry Pi y pidiendole al usuario cuando se registre que acceda a ese servidor.
- Registrado: cuando el usuario introduzca la contraseña de usuario del sistema, en este campo se introducirá la hora del registro, para controlar que cuando se cumpla el tiempo configurado por el administrador este usuario tenga que volver a introducir la contraseña.
- ¿En casa?: esta propiedad indica si el usuario está en casa.
- ¿Es admin?: esta propiedad indica si el usuario es administrador.
- Admin registrado: tiene la misma función que la propiedad *registrado*, pero esta vez como administrador
- Estado actual: esta propiedad nos permite interactuar con el sistema continuamente, cuando se utilice una acción que necesite más información, la propiedad *acción* toma el valor de la acción pedida y la propiedad *estado* se irá actualizando dependiendo de la cantidad de datos que se necesiten.

Un apartado importante sobre el control domótico es la seguridad, no se puede permitir que una persona externa se conecte a nuestro sistema. El problema que se encuentra al introducir la contraseña es la imposibilidad de que se edite el mensaje automáticamente por el sistema, ya que telegram sólo ofrece la posibilidad de editar mensajes propios.

Para subsanar este problema, el sistema generará una key aleatoria, la cual enviará con un botón de edición. El usuario deberá sumar esta key a la contraseña real término a término. De esta forma, al pulsar el botón de *Editar*, el sistema automáticamente eliminará el mensaje con la key, podemos ver el proceso en la figura 5

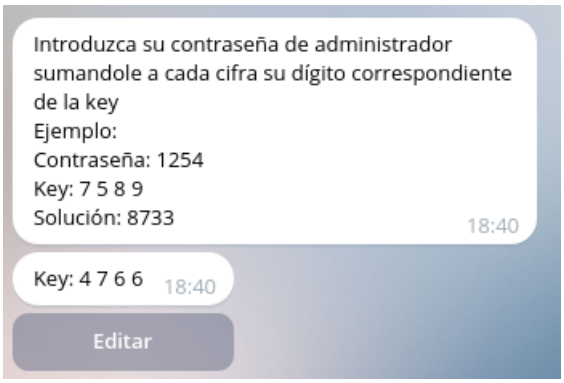

Fig. 5: Ejemplo de introducción de contraseña

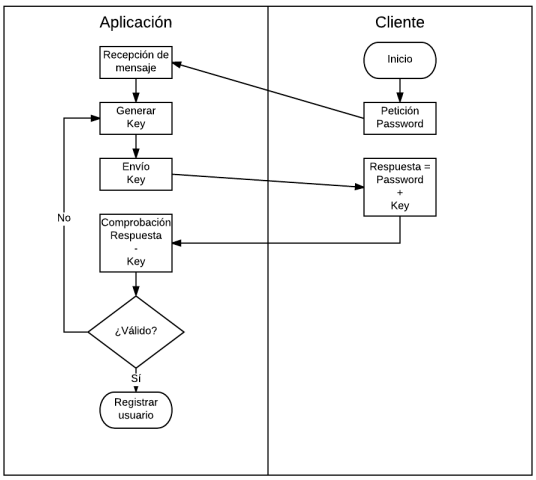

Fig. 6: Diagrama de Actividad introducción de contraseña

3.4 Control de presencia

Para la detección de presencia nos hemos basado en el protocolo ARP. Sabiendo la ínfima posibilidad que hay de coincidir dos MAC en un misma red, cada usuario será identificado por el sistema con la MAC de su smartphone. De esta forma, cada vez que tengamos una conexión o desconexión en nuestra red se detectará el dispositivo y se asociará con un usuario.

Estas entradas y salidas se van a ir almacenando en un fichero CSV diario, a los cuales sólo los administradores del sistema tendrán acceso.

La forma de detectar la MAC del usuario es mediante un servidor creado en nuestra red local, al cual el usuario deberá conectarse. Con este proceso el sistema obtendrá la IP del dispositivo del usuario. Cabe recordar que la comunicación con el sistema se realiza a través de la aplicación de mensajería, por lo tanto aunque nos encontremos en la misma red que el sistema no se podrá obtener la IP enviando un mensaje, pues la identificación que se utiliza en la aplicación de mensajería es diferente.

Una vez que se tenga la IP del usuario, en el siguiente reconocimeinto que se haga de la red se identificará la MAC del usuario y se le asignará en su correspondiente entrada, como se vio en la subsección 3.3. Se puede ver un diagrama

de este proceso en la figura 7

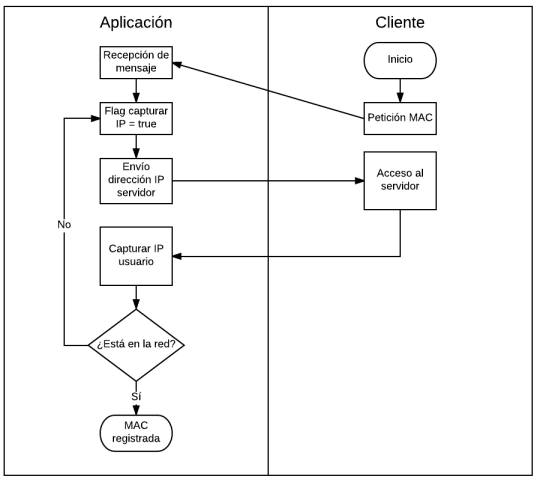

Fig. 7: Diagrama de Actividad Petición de MAC

3.5 Control remoto del portero automático

Mediante los mensajes de voz integrados en Telegram, un altavoz y una cámara, se ha desarrollado este sistema para el control remoto del portero automático.

Este control se activará cuando alguien llame al timbre. Uno de los GPIO de la Raspberry Pi emitirá el evento que iniciará una captura de la cámara. Esta captura se enviará a cada usuario del sistema. Para poder responderle a quien llama, el usuario del sistema puede enviar un mensaje de voz mediante telegram, este mensaje de voz se reproducirá por un altavoz colocado en el portero automático.

Cuando un usuario de Telegram envía un mensaje de voz mediante la aplicación, este mensaje se almacena en el servidor de Telegram, por lo que cuando recibamos el mensaje de voz en nuestra aplicación la información que obtendremos será el ID del fichero en el servidor. Por lo que una vez recibido este ID, tendremos que descargar el audio, mediante el método *downloadFile*. El fichero descargado se encuentra en formato *ogg* usando el codec *opus*, este fichero no se puede reproducir directamente desde la terminal de linux. Por lo tanto, se tiene que convertir a un formato común como es el *wav*, que sí se puede reproducir mediante el comando *play* del paquete *sox*.

3.6 Control de sensores y actuadores

Para el control de los sensores y los actuadores se van a usar los GPIOs de la Raspberry Pi, aunque tambien es posible encontrarl elementos controlados por USB.

Para el control de los GPIOs se usará el módulo de NodeJS llamado *rpi-gpio*, con este módulo tenemos preparado los métodos para configurar los GPIO, escribir, leer y escuchar los eventos que producen los cambios.

Los sensores y actuadores a incluir dependerá de las peticiones del usuario, ya que los GPIOs son bastante polivalentes en cuanto a funcionalidad, por lo que se dejarán preparadas las funciones para trabajar con los siguientes elementos, aunque se podrían añadir más:

- **Luces:** un funcionamiento sencillo como es el apagar y encender luces, a petición del usuario el pin configurado para esta función se pondrá al valor pedido.
- **Climatización:** al igual que con las luces, se puede preparar la climatización para ser encendida y apagada.
- **Toldos y persianas:** este control se puede simular con motores DC, estos motores regulan su velocidad mediante PWM y se controla la posición inicial y final mediante finales de carrera. Por consiguiente, midiendo el tiempo entre la posición final y la inicial se puede hacer aproximaciones de posiciones intermedias. La velocidad de subida y bajada será fija.

3.7 Control de temperatura

El control de la temperatura se va a realizar mediante el API del servicio online openweathermap, Cada hora se ejecutará un método en nuestro sistema que hará una petición al servicio de openweathermap, cuando se realice esta petición, dependiendo de la configuración de los sensores y actuadores se realizarán una serie de evaluaciones para ajustar nuestro hogar a las condiciones meteorológicas y a la temperatura del momento.

Nuestro sistema necesita tener las coordenadas de su ubicación, así que cuando se realice la configuración inicial del bot nos pedirá que enviemos la ubicación en la que se encuentra. Esto se realiza mediante las opciones de telegram, el sistema recibe la ubicación y almacenará las sus coordenadas.

3.8 Control de alarma

La alarma contará con sensores de apertura de puertas y ventanas para la detección y una señal acústica para indicar la presencia dentro del hogar.

El proceso de activar y desactivar la alarma será siempre manual, aunque el sistema de detección de presencia servirá de asistente, de forma que cuando no haya nadie en casa enviará a todos los administradores un mensaje avisando de la desprotección del hogar.

Todos los usuarios tienen permisos para activar y desactivar la alarma, pero aunque estén registrados tendrá que insertar la contraseña para que la acción tenga efecto. De esta forma, una vez que el sistema detecte presencia, en la entrada, dará al usuario un margen de 15 segundos para desactivar la alarma.

3.9 Almacenaniento de opciones

En este apartado, vamos a ver como se almacenan las opciones de forma permanente a la espera para prevenir la pérdida de datos con la caida de la aplicación. La cantidad de información que se maneja no tiene tamaño como para usar una base de datos como mySQL, MongoDB, etc. Se ha optado por almacenar los datos persistentes en archivos CSV, de esta forma tenemos implementado un módulo de lectura y escritura de este tipo de archivos para en un futuro poder usar el envío de ficheros CSV para configurar el sistema, añadir usuarios, etc.

4 Pruebas de Rendimiento

Se han realizado pruebas de rendimiento en los siguientes componentes del sistema:

- Respuesta al protocolo ARP
- Captura de foto
- Proceso de recepción y reproducción de audio
- Petición y tratamiento de datos meteorlógicos

Estas pruebas de rendimientos se van a realizar sobre diferentes plataformas como son:

- Raspberry Pi 3
- SSOO Debian virtualizado sobre MacBook Pro

Se ha intentado medir el tiempo de respuesta de la petición, pero el dato que recibimos en el mensaje de telegram se da en formato *Unix Time*, de forma que está dado en segundos por lo que esto nos da un tiempo que no se corresponde con la realidad.

4.1 Respuesta protocolo ARP

Los tiempos que se han medido podemos verlos en la tabla 4:

Dispositivo	Tiempo
Raspberry Pi	13380ms
MacBook	11778ms

Tab. 4: Tiempos ARP

Se pueden aprecidar que son tiempos altos, pero no es una detección que se necesite con gran precisión. Por lo que son tiempos asumibles.

4.2 Captura de foto

Estos tiempos son tomados desde que se realiza la foto hasta que ésta se almacena en nuestro disco duro. Pues es necesario este almacenamiento para poder enviarla mediante el módulo que estamos usando para comunicarnos con el API de Telegram.

Dispositivo	v4l2camera	fswebcam
Raspberry Pi	1246ms	1278ms
MacBook	1038ms	986ms

Tab. 5: Tiempos captura de foto

Se han medido la velocidad de dos módulos diferentes para ver cual elegir y podemos apreciar que nos hay diferencia entre ellos. Por lo que nos quedamos con el módulo v4l2camera que es más flexible con respecto a la programación.

Por otro lado, se ven que son tiempos relativamente altos, pero estos tiempos tienen también como factor determinante la velocidad de captura de la cámara, que en este caso sólo se ha hecho pruebas con un modelo.

4.3 Proceso de recepción y reproducción de audio

En esta medida pasamos por alto el tiempo de subida del audio, puesto que es un factor que depende de muchos factores. El fichero de audio que se utiliza es de aproximadamente dos segundos.

Dispositivo	Descarga	Decode	Total
Raspberry Pi	609ms	1278ms	
MacBook	524ms	986ms	

Tab. 6: Tiempos mensajes de voz

4.4 Petición y tratamiento de datos meteorológicos

La comparación entre los tiempos de petición de los datos meteorológicos podemos verlos en la tabla 7

Como conclusión podemos destacar que hay una diferencia sustancial entre ambas máquinas, pero es algo totalmente obvio viendo las grandes diferencias de cada máquina. Pero de igual forma, vemos que son tiempos totalmentes asumibles para nuestro desarrollo.

Dispositivo	Tiempo
Raspberry Pi	330ms
MacBook	280ms

Tab. 7: Tiempos datos meteorológicos

En relación a las pruebas en la que necesitamos conexión a internet, hay que mencionar que el MacBook se encuentra conectado a una red Inalámbrica, mientras que la Rasberry Pi se encuentra conectado al router mediante cable Ethernet.

5 Costes

El tiempo desarrollo de este sistema domótico ha sido de 300 horas, tomando un precio por hora de 30€, tenemos un coste de desarrollo de 9000€.

Luego cada sistema tendría un coste fijo de:

Raspberry Pi	.....	60€
Sensores Alarma	.....	10€
Cámara	.....	15€
Altavoz	.....	10€
Total	.....	95€

Tab. 8: Costes fijos

Para cubrir los costes con la venta de 100 terminales, cada terminal debería tener un precio de 185€(IVA no incl.) A estos costes habría que sumarle los sensores y actuadores que desea el cliente y el coste de su instalación.

6 Conclusiones

Durante el desarrollo del TFM, se ha podido ver como todos los objetivos planteados se han podido satisfacer positivamente.

En primer lugar, se ha podido ver como la Raspberry Pi, aunque con sus limitaciones cumple perfectamente con las exigencias que se pedian.

Con respecto al desarrollo del software mediante NodeJS, se ha podido ver como ha cumplido perfectamente con lo que esperabamos de él. La gran cantidad de módulos que se pueden encontrar desarrollados por la comunidad han facilitado el trabajo enormemente.

El control de presencia que se planteó se ha podido implementar en el sistema, aunque hemos visto en la sección 4 que lleva un tiempo bastante alto el reconocimiento de toda la red. Pero al no requerir que este sistema lleve un control exhaustivo podemos permitirnos estos tiempos.

El último objetivo que se planteó fue que el sistema se controlara mediante una aplicación de mensajería. Se ha elegido Telegram, la cual hemos visto el API que ofrece para crear aplicaciones dentro de su cliente. Este API en conjunto con uno de los módulos de NodeJS nos ha permitido desarrollar nuestra aplicación sin dificultad pudiendo ofrecer todas las opciones que se deseaban.

REFERENCES

[1] Download Raspbian for Raspberry Pi [online]
Available at: https://www.raspberrypi.org/downloads/raspbian/

[2] Telegram APIs. [online]
Available at: https://core.telegram.org/api

[3] Telegram Bot API [online]
Available at: https://core.telegram.org/bots/api

[4] RFC 826 - Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. [online]
Available at: https://tools.ietf.org/html/rfc826 [Accessed 6 Dec. 2016].

[5] Current weather and forecast. OpenWeatherMap
Available at: http://openweathermap.org

[6] Node Telegram Bot API. [online]
Available at: https://github.com/yagop/node-telegram-bot-api

[7] Control Raspberry Pi GPIO pins with node.js [online]
Available at: https://www.npmjs.com/package/rpi-gpio

[8] Capturing images from USB(UVC) webcam on linux machines [online]
Available at: https://www.npmjs.com/package/v4l2camera

[9] Get weather from OpenWeatherMap [online] Available at: https://www.npmjs.com/package/openweather-node