



ugr | Universidad
de **Granada**

TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA

Domotics Agents

Gestión Domótica Contralada Por Agentes

Autor

Marcos Avilés Luque (alumno)

Director

Luis Castillo Vidal (tutor)



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍAS INFORMÁTICA Y DE
TELECOMUNICACIÓN

Granada, Septiembre de 2018



Domotics Agents

Gestión Domótica Controlada Por Agentes.

Autor

Marcos Avilés Luque (alumno)

Director

Luis Castillo Vidal (tutor)

Domotics Agents: Gestión Domótica Controlada Por Agentes

Marcos Avilés Luque (alumno)

Palabras clave: Magentix, Agente, Pi4j-gpio, Picamera, RPi.GPIO, TelegramBots, Sensores, Raspberry Pi, JAVA, PYTHON,

Resumen

Este proyecto está dirigido a realizar un sistema multiagente para construir un sistema domótico inteligente basado en un sistema multiagente distribuido en una red de dispositivos Raspberry Pi conectados a dispositivos de control domótico. En este caso se busca un sistema no sólo para controlar la gestión domótica de uso doméstico, como por ejemplo, contar con el sistema de seguridad, apertura y cierre de puertas y/o cochera, control de la temperatura, encendido de luces..., sino también aplicar la inteligencia que nos permite la versatilidad de los agentes, así como el cambio de comportamiento entre ellos en un determinado instante, y la actuación automática según las propiedades de cada agente.

Una vez tenemos el sistema domótico de la vivienda, el siguiente paso es poder actuar sobre él, para poder sacar el máximo beneficio y rendimiento de este sistema, aprovechando la potencialidad de estos agentes se diseña un aplicación que permite establecer una comunicación tanto interna como externamente, es decir, desde dentro o fuera de la vivienda domótica, para poder así actuar desde cualquier parte o incluso ser notificado en un momento determinado ante una situación específica.

Domotics Agents: Domotic Management Controlled by Agents

Marcos Avilés Luque (student)

Keywords: Magentix, Agente, Pi4j-gpio, Picamera, RPi.GPIO, Telegram-Bots, Sensors, Raspberry Pi, JAVA, PYTHON,

Abstract

This project is aimed at making a multi-agent system to build an intelligent home automation system based on a multi-agent system distributed in a network of Raspberry Pi devices connected to home automation control devices. In this case, a system is sought not only to control home automation management for domestic use, such as having the security system, opening and closing doors and/or garages, controlling the temperature, turning on lights ... , but also apply the intelligence that allows us the versatility of the agents, as well as the change of behavior between them at a certain moment, and the automatic performance according to the properties of each agent.

Once we have the home automation system, the next step is to act on it, in order to get the maximum benefit and performance from this system, taking advantage of the potential of these agents an application is designed to establish a communication both internally and externally , that is, from inside or outside the home automation, to be able to act from anywhere or even be notified at a certain time in a specific situation.

Yo, **Marcos Avilés Luque**, alumno de la titulación Grado en Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 25342389T, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Marcos Avilés Luque

Granada a 14 de JULIO de 2018.

D. Luis Castillo Vidal (tutor), Catedrático en la ETSIIT del Departamento Ciencias de la computación e Inteligencia Artificial (DECSAI) de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado *Domotics Agents, Gestión Domótica Controlada Por Agentes*, ha sido realizado bajo su supervisión por **Marcos Avilés Luque (alumno)**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 04 de AGOSTO de 2018.

Los directores:

Luis Castillo Vidal (tutor)

Agradecimientos

En primer lugar dar las gracias al profesor catedrático Luis Castillo Vidal (tutor de este proyecto), ya que sin él este proyecto quizás no fuera sido llevado a cabo, por otra parte haber contado con la suerte de haber sido alumno suyo en varias de las asignaturas a lo largo de la carrera, ya que muchos de los conceptos obtenidos y aprendidos en dichas asignaturas han sido clave para el desarrollo. Así como su constante interés para alcanzar el objetivo de este proyecto, proponiendo diferentes alternativas y ofreciendo todo el apoyo posible.

En un segundo lugar dedicar estos agradecimientos aquellas personas externas a este proyecto, como aquellos autores de las diferentes librerías usadas para este proyecto, que han facilitado el desarrollo del mismo, ya que se sabe, que hay un gran trabajo por su parte bajo cada una de estas herramientas, así como las distintas fuentes de información, referenciadas a lo largo de este proyecto, que también gracias a ellas, ha sido posible investigar las posibles soluciones más óptimas y cumplir unos de los objetivos de este sistema.

Y no dejar atrás aquellas personas que, tras día a día me han acompañado durante todo este tiempo dando esos ánimos y fuerzas que ciertamente son esenciales y necesarias para poder seguir adelante. No especialmente por los grandes conocimientos ni ayuda ofrecida, sino por el continuo apoyo moralmente que a veces nos hace conseguir el éxito sin darnos cuenta. Especialmente, me gustaría dedicar unas simples gracias a Sandra Pareja Sanchez, por acompañarme no solo en este viaje sino a lo largo de todos estos años.

Índice general

	Página
1. Introducción	1
1.1. Descripción del problema	1
1.2. Trabajos relacionados	2
1.2.1. Gestión domótica de una casa unifamiliar basada en Arduino.	2
1.2.2. La casa más 'inteligente' de 2014.	3
1.2.3. Edificio la Vela. Ciudad BBVA Madrid.	4
1.2.4. Costes y precios.	5
1.3. Primeras conclusiones	6
1.4. Objetivo	7
2. Herramientas y Plataformas	8
2.1. Raspberry Pi	8
2.1.1. ¿Por qué Raspberry Pi y no otro componente?	10
2.1.2. Pines GPIO Raspberry Pi 3 Model B.	12
2.2. Sensores y dispositivos	14
2.2.1. DHT11 Sensor de temperatura y humedad.	14
2.2.2. Módulo KY-004 Botón o pulsador.	16
2.2.3. Módulo KY-011 Módulo LED 2 colores.	17
2.2.4. YL-83, LM393 Sensor de LLuvia.	18
2.2.5. Módulo de conversión DC/DC de múltiple salida.	20
2.2.6. Módulo L298N - Puente H.	22
2.2.7. Módulo KY-033 - Tracking Sensor Module.	25
2.2.8. Módulo cámara.	26
2.3. Plataforma Magentix	28
2.4. Agentes	29
2.4.1. ACL (Agent Communications Language).	30
2.5. Raspbian	32
3. Propuesta de solución	35
3.1. Introducción a la solución.	35
3.2. Diagramas	38

3.2.1. Diagramas de secuencia	38
3.2.2. Diagramas de actividad	49
4. Implementación.	54
4.1. Agentes	54
4.1.1. Cámara	57
4.1.2. Cochera	60
4.1.3. Led	62
4.1.4. Lluvia	62
4.1.5. Offline	64
4.1.6. Persiana	66
4.1.7. Movimiento	66
4.1.8. Temperatura	66
4.1.9. Controlador	67
5. Versión Ampliada.	69
5.1. Telegram.	70
5.1.1. Bot Telegram.	70
5.2. Diseño.	71
5.3. Instalación.	73
5.4. Implementación.	76
5.5. Novedades.	80
6. VersionFinal.	83
7. Conclusiones.	89
7.1. Bajo Coste	89
7.2. Sistema de Agentes	90
8. Trabajo futuro	91
8.1. Vivienda unifamiliar real.	91
8.2. Aplicación Android	93
8.3. Aprendizaje	94
Bibliografía	99

Índice de figuras

1.1.	Casa más inteligente 2014 (Madrid). Domotizada por la empresa ”+Spacio”	3
1.2.	Sistemas incluidos en la casa más inteligente 2014.	4
1.3.	Edificio la Vela. Complejo BBVA 2017.	5
2.1.	Raspberry Pi 3 Model B.	9
2.2.	Placa Arduino UNO.	11
2.3.	Numeración Pines GPIO Raspberry Pi 3 Modelo B.	13
2.4.	DHT11 Sensor de temperatura y humedad.	15
2.5.	KY-004 Módulo pulsador.	17
2.6.	KY-011 Módulo LED 2 colores.	18
2.7.	Módulo YL-83 y placa LM393 (Sensor de lluvia).	19
2.8.	Conexiones Módulo YL-83 Sensor de Lluvia.	19
2.9.	Módulo de conversión DC/DC de múltiple salida.	21
2.10.	Módulo L298N - Puente H.	22
2.11.	Jumper Módulo L298N - Puente H.	23
2.12.	Configuración electrónica Módulo L298N - Puente H.	24
2.13.	KY-033 Módulo Sensor de Seguimiento.	26
2.14.	Módulo HC-SR501, sensor de movimiento.	26
2.15.	Módulo cámara para Raspberry Pi.	27
2.16.	Conector CSI para cámara Raspberry Pi.	27
2.17.	Directorio generado posteriormente a la instalación de Magentix2.	28
2.18.	Proceso de instalación de Raspbian con Win32 Disk Imager. .	33
2.19.	MicroSD Card Slot Raspberry Pi 3.	34
2.20.	Sistema Raspbian.	34
3.1.	Diagrama de Clases.	37
3.2.	Diagrama de secuencia Módulo Cámara.	39
3.3.	Diagrama de secuencia Módulo L298N Cochera.	40
3.4.	Diagrama de secuencia Módulo Lluvia.	41
3.5.	Diagrama de secuencia Módulo Movimiento.	42
3.6.	Diagrama de secuencia Módulo Led.	43

3.7. Diagrama de secuencia Módulo Offline.	44
3.8. Diagrama de secuencia Módulo L298N Persianas.	45
3.9. Diagrama de secuencia Módulo Temperatura.	46
3.10. Diagrama de secuencia activando el sistema de Seguridad.	47
3.11. Diagrama de secuencia detectando Lluvia.	48
3.12. Diagrama de secuencia detectando Movimiento.	49
3.13. Diagrama de actividad del Agente Cámara.	50
3.14. Diagrama de actividad del Agente Cochera.	50
3.15. Diagrama de actividad del Agente Led.	51
3.16. Diagrama de actividad del Agente Lluvia.	51
3.17. Diagrama de actividad del Agente Movimiento.	52
3.18. Diagrama de actividad del Agente Offline.	52
3.19. Diagrama de actividad del Agente Persianas.	53
3.20. Diagrama de actividad del Agente Temperatura.	53
 4.1. Ejemplo de creación y lanzamiento de un agente.	55
4.2. Ejemplo de una clase Agente.	56
4.3. Interacción con mensajes, objeto de la clase ACLMessage. . . .	57
4.4. Script en Python para generar una imagen con la cámara Pi. . .	58
4.5. Código para ejecución de un Script en Python desde Java. . .	59
4.6. Script en Python para generar una vídeo con la cámara Pi. .	60
4.7. Código para la preparación de un GPIO.	61
4.8. Método pulse() para accionar un GPIO.	62
4.9. Agente no bloqueante para recibir mensajes.	63
4.10. Comprobación de Lluvia.	64
4.11. Comportamiento y capacidad del agente Lluvia.	64
4.12. Manejador de eventos digitales en un pin GPIO.	65
4.13. Arquitectura de la sociedad de agentes.	68
 5.1. Diagrama de secuencia agentes Telegram y el sistema Raspberry Pi.	72
5.2. BotFather. API Bot de Telegram.	74
5.3. BotFather. API Bot de Telegram. \newbot.	75
5.4. Creación de un nuevo Bot de Telegram.	76
5.5. Estructura Java de DomoticAgentsBot.	77
5.6. Estructura Java de DomoticAgentsBot.	78
5.7. Administrando nuestro Bot de Telegram.	78
5.8. Implementando la recepción de mensajes de nuestro Bot de Telegram.	79
5.9. Recibiendo un comando desde el Bot de Telegram.	79
5.10. Envío de mensajes a través del Bot de Telegram.	80
 6.1. Maqueta finalizada.	84
6.2. Principio de estado del sistema domótico.	85

6.3. Diseño de la maqueta.	86
6.4. Construcción de la maqueta.	86
6.5. Primeras pruebas de la maqueta.	87
6.6. Añadidos todos los sensores.	87
8.1. Módulo MAX485, utilizando el protocolo RS485.	93
8.2. Ejecución Aplicación Android.	94

Capítulo 1

Introducción

1.1. Descripción del problema

Según la definición de la Real Academia Española [35], domótica se define como conjunto de sistemas que automatizan las diferentes instalaciones de una vivienda. Así como la definición otorgada en [44] ”*Se llama domótica a los sistemas capaces de automatizar una vivienda o edificación de cualquier tipo, aportando servicios de gestión energética, seguridad, bienestar y comunicación, y que pueden estar integrados por medio de redes interiores y exteriores de comunicación, cableadas o inalámbricas, y cuyo control goza de cierta ubicuidad, desde dentro y fuera del hogar. Se podría definir como la integración de la tecnología en el diseño inteligente de un recinto cerrado*”.

En primer lugar, nos encontramos con uno de los problemas actuales, como es buscar la comodidad o la necesidad en ciertos casos, para el acceso, gestión y uso de una casa doméstica. El avance de la tecnología es inevitablemente mayor cada día, y desde mi punto de vista, las personas buscamos cada vez busca mayor comodidad y facilidad de usar cualquier cosa que se pueda automatizar dentro de lo posible.

Investigamos un poco o navegamos por Internet, y encontramos bastantes soluciones actualmente, como por ejemplo, sistemas para controlar la calefacción, en el que se realiza una instalación propia para ese sistema y controlando su funcionalidad basándose en la temperatura de la casa, apagando o encendiendo dicha calefacción. ¿Si ahora quiero ampliar y utilizar el mismo sistema para el aire acondicionado, necesito realizar una nueva instalación distinta?. ¿Necesito contratar una segunda empresa para el sistema de seguridad? ¿No puede ser una instalación que sirva para todo? Incluso si no nos encontramos en domicilio y queremos accionar sobre todos estos elementos, ¿que solución distribuida podemos encontrar?

Cuando hablamos de interconectar todos los elementos, o poder tener

un sistema, como por ejemplo, con todos los requisitos mencionados anteriormente, nace uno de los temas también mas discutidos actualmente y con un gran interés, conocido como IOT (Internet of things), o también llamado como el Internet de las cosas. No solo consiste en buscar la opción de conectar todo de la manera que sea posible, que tenga conexión a la red, o pueda ser accedido desde cualquier parte del mundo, lo cual también es unos de los pilares de este tema, sino que también sea autosuficiente, es decir, que el sistema tenga esa capacidad de decisión y acción ante una determinada situación de manera autónoma.

Se produce la necesidad de tener un sistema lo mas completo posible, que sea accesible por nosotros desde cualquier parte del mundo, o de fácil uso para aquellas personas que realmente lo necesitan, y contenga esa inteligencia que pueda actuar ante una serie de acontecimientos o sucesos que ocurran en un determinando instante de tiempo.

1.2. Trabajos relacionados

Antes de comenzar a diseñar este problema y entrar en detalles, se realiza una pequeña investigación para encontrar que proyectos y trabajos hay realizados sobre este ámbito actualmente, y como veremos a continuación, existen bastantes soluciones y propuestas que ayudan a coger algunas ideas, plantearlas de distinta manera y encontrar otras vías o soluciones, que hacen este proyecto diferente a los demás.

1.2.1. Gestión domótica de una casa unifamiliar basada en Arduino.

En este proyecto [16], como TFG de la universidad de Valladolid de este año anterior (2017), es un proyecto bastante similar, orientado a la automatización domótica de una casa, hay que destacar que este proyecto solo se basa en la gestión y diseño del problema, es decir, desde un punto teórico intenta resolver el problema, pero no hay nada práctico ni prueba real del funcionamiento de esta posible solución, aún así es interesante por las posibles soluciones. Veamos los siguientes puntos:

- Utiliza sensores de movimiento concretamente el modelo "HC-SR501" que es capaz de detectar movimiento desde 3 a 7 metros, para simular el sistema de seguridad de la vivienda activando los buzzers, que son como pequeños zumbadores para emitir pitidos.
- El sensor "TMP36", es un sensor de temperatura, para en este caso controlar la calefacción de la vivienda a través de unas electroválvulas y un suelo radiante.

- También menciona una fotorresistencia, llamadas LDR, que es capaz de medir la intensidad de luz ambiental de un determinado lugar. Estos elementos son normalmente utilizados para el ahorro energético de la vivienda, ya que podría permitir calcular la energía necesaria para la luz artificial de la vivienda.
- Incluye en su propuesta una pantalla táctil LCD, conectada al controlador Arduino, que permite la visualización del estado de la calefacción, y poder accionar sobre ella.

Todo esto anterior mencionado antes, esta controlado por la placa Arduino, que controla los sensores anteriores mencionados, y acciona sobre el sistema de seguridad, el garaje y la calefacción. En este proyecto se adjunta como podría ser la implementación de dicho sistema, aunque el propio autor menciona que solo es un diseño, y el control de todos estos sensores mediante la placa Arduino se hace mas complicado a medida que se incluyan mas sensores y controladores.

1.2.2. La casa más 'inteligente' de 2014.

Un pequeño artículo del periódico el Mundo [29] nombra el premio de innovación otorgado por la comunidad de Madrid en 2014, a la empresa "*+Spacio*", en la que consta de una vivienda unifamiliar real con control de toda la iluminación, la climatización, estores y persianas, la difusión sonora, el sistema de seguridad, la depuradora de la piscina y el sistema de riego de la vivienda.



Figura 1.1: Casa más inteligente 2014 (Madrid). Domotizada por la empresa "*+Spacio*".

En definitiva este proyecto busca suprimir la acciones usuales mediante botones y mecanismos manuales a lo que estamos acostumbrados, dar esa

comodidad y centralización unificada desde dispositivos móviles o tablets. Y en mayor medida, desde un punto de vista se le da más prioridad o importancia al consumo energético, tanto electricidad como agua, por lo que se incorporan los sistemas de iluminación LEDs, suelo radiante y refrescante, riego automático...algunos de ellos los podemos ver la siguiente Figura 1.2

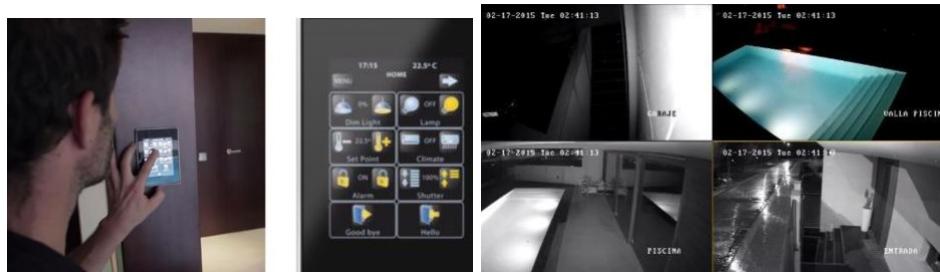


Figura 1.2: Sistemas incluidos en la casa más inteligente 2014.

Sin poder conocer más, ya que nos encontramos ante un pequeño artículo, no se puede conocer como se ha realizado las instalación o la implementaciones de dichos sistemas, así como un presupuesto o una ligera aproximación sobre los costes, dispositivos necesarios, controladores y demás detalles técnicos. Aunque si bastante interesante para coger las propuestas de este proyecto como ideas necesarias y más relevantes para un proyecto domótico.

1.2.3. Edificio la Vela. Ciudad BBVA Madrid.

En esta noticia [4] llama bastante la atención una cosa en particular, no que sea un proyecto sin importancia ya que se trata de un complejo de más de 114.000 metros cuadrados y un edificio con la mayor innovación, sostenibilidad y ahorro energético actual, cuyo proyecto obtiene el premio del certamen de la comunidad de madrid en 2017.

El objetivo es centrarse en este punto, gracias a la información [25] que proporciona la empresa "*Id.Domotica*" que participó en el proyecto del edificio "La Vela", y consta de que instalaron **4.075 multisensores** asociados a 23.577 luces interiores del edificio como podemos apreciar en la Figura 1.3. A lo que me hace reflexionar sobre la capacidad de comunicación entre tal cantidad de sensores o controladores.



Figura 1.3: Edificio la Vela. Complejo BBVA 2017.

Junto con el trabajo relacionado anterior 1.2.1, se menciona la difícil comunicación o control de mayor cantidad de sensores, aunque mi proyecto esta orientado para una vivienda, posiblemente no contenga o sea necesario tal cantidad de sensores, pero es bueno tener en cuenta la posible escalabilidad para futuras implementaciones.

1.2.4. Costes y precios.

Llevar a cabo un proyecto o no, se basa en estudiar los precios y los costes que tienen estas instalaciones domóticas es una casa unifamiliar, por lo que buscando rápidamente en varios lugares se puede encontrar algunos precios o presupuestos orientativos.

Sin realizar una búsqueda profunda, ya que sólo interesa saber que precios nos encontramos sobre este tema actualmente. Por ejemplo, las dos primeras empresas referenciadas y dedicadas a estas instalaciones, ofrecen unos presupuestos orientativos como son [15] Domintell y [33] Fermax, la primera impresión es que tanto una empresa como otra, afirman que el coste aproximado de una instalación domótica convencional ronda entre **1,5 % y 6,8 % del total** del presupuesto necesario para la construcción de la vivienda.

Por ejemplo, para ser un poco más específico, en Domintell [15], se encuentra un pequeño presupuesto que incluye, un monitor táctil de visualización y control, varios sensores, detectores de movimiento, incendio, central de aviso telefónico y un par de actuadores por el precio de **1.364 €**, el coste de la instalación no está incluido en el precio.

Hablar de precios y definir si este sistema es caro o barato, es una opinión distinta de cada persona en función de sus posibilidades y sus capacidades, pero particularmente desde un punto de vista considerable, no son unos precios tan competitivos, y llamativos para ciertas personas que no vean necesario estos sistemas.

Se consigue un objetivo en este apartado, que para un proyecto domótico tenemos que tener en cuenta si es para un futuro proyecto, para una vivienda ya existente, o para una construcción unifamiliar nueva, ya que esto permitirá o mejor dicho facilitará el trabajo de adaptación e incorporación de los sistemas domóticos, y como es lógico el importante incremento del presupuesto, como por ejemplo:

- Si la vivienda ya se encuentra construida y no cuenta con una calefacción o refrigeración central, no se podrá controlar dicha temperatura de confort para cada habitación individualmente, aunque incorporemos sensores de temperatura en cada habitación.
- Igualmente el precio se verá afectado si se desea incorporar el riego automático pero no existe la pre-instalación del riego.
- O el simple hecho de la apertura de puertas y persianas de la vivienda, es necesario una motorización para el uso de cada una de ellas, por lo que si estos motores están incluidos o no el presupuesto podrá verse bastante afectado.

1.3. Primeras conclusiones

Después de esta investigación anterior con los trabajos relacionados 1.2, ya se obtiene cierta información necesaria sobre la tendencia de éste ámbito actualmente, que es lo más importante, mejor valorado y los principales requisitos a tener en cuenta.

Para una posible solución hay que encontrar y tener en cuenta varios aspectos importantes, desde cierto punto de vista y algunas conclusiones obtenidas:

- Los sensores y componentes estándares utilizados tienen actualmente unos precios bastante elevados desde mi punto de vista. Hay que buscar unos sensores que sean económicos, por el trabajo relacionado 1.2.1, es interesante la utilización de los sensores de Arduino que tiene un coste bastante económico.

- Aunque hablamos de una casa unifamiliar, puede que lleguemos a conseguir reunir una cantidad de sensores y controladores considerable, por lo que hay que buscar un sistema de comunicación y coordinación entre todos los sensores para que no sea un gran problema a la hora de ponerlo en marcha, y nos facilite el trabajo en la implementación.
- En relación al punto anterior, es interesante aplicar una comunicación distribuida, es decir, que pueda ser accesible no sólo desde la propia vivienda físicamente, sino desde fuera de ella mediante un dispositivo móvil, como un smartphone o tablet.
- Otro objetivo que se propone es aplicar una cierta "*inteligencia*" a la vivienda, es decir, dotar al sistema de la capacidad de actuar por sí solo en un determinado caso, por ejemplo, en caso de que haya una intrusión en la vivienda (en el sistema de seguridad), cierre del garaje o puertas en caso de olvido y no se encuentre nadie en la vivienda...

1.4. Objetivo

Relacionado con las distintas conclusiones anteriores, crear un sistema domótico en el que:

- Se pueden agregar tantos componentes o sensores sean necesarios, o incluso deseados por algún motivo concreto, por lo que estos sensores deben tener el mínimo coste posible para mantener un presupuesto competitivo, y poder agregar tantos como se deseen.
- Un sistema bien diseñado y lo más sencillo posible, ya que he encontrado un sistema con solo varios sensores controlados con el mismo software, llegando a ser bastante tedioso y posiblemente imposible de escalar.
- Con este justo punto anterior, intentar controlar cada componente o sensor individualmente, es decir, poder generar como un puzzle, que cada sensor sea una pieza distinta del puzzle, y a través de una organización se forme completamente dicho puzzle.
- Una vez generado el sistema poder aplicarle esa inteligencia que sea capaz de dotar a la vivienda unifamiliar la capacidad de actuar ante una o varias situaciones determinadas de forma autónoma.

Capítulo 2

Herramientas y Plataformas

En este capítulo vamos a ver las herramientas, plataformas y componentes que son necesarias o más óptimas para satisfacer la solución al problema, en este caso se destacará en cada apartado las características más importantes y porqué esa elección y no otra.

2.1. Raspberry Pi

Raspberry Pi es una organización que se fundó sobre el año 2009 en Reino Unido [21], en la que se dedicaba en sus principios al diseño de circuitos impresos hechos a medida. Unos de sus proyectos era desarrollar un pequeño ordenador de bajo coste y un alto rendimiento para que aumentar el alcance a más personas, también una primera prioridad ser capaces de animar a aumentar la educación a los niños en su formación escolar para aprender informática y promover la enseñanza de ciencias de la computación, así como también atraer a otras personas para el desarrollo y solución de distintos problemas, como trabajos de futuro gracias a este proyecto.

Aunque hasta el año 2012 no se comercializó la primera placa de tamaño reducido, que ya consta de una computadora bastante potente para su tamaño, conocida como **Raspberry Pi 1 Model A**. No voy a detallar las características de este modelo ya que actualmente cuenta con varias versiones mejoradas, hasta llegar a la última versión que fue lanzada en Marzo de este mismo año (2018) denominada **Raspberry Pi 3 Model B+**. Más adelante si veremos las características del modelo adquirido y utilizado para este proyecto.

Si mencionar que esta fundación sobre el año 2015 lanzó una segunda línea de fabricación, de placas de aún menor tamaño, con unas prestaciones

mínimas pero bastante aceptables, pero lo más interesante es el coste inferior a **5\$** [22].

Ahora sí podemos ver el modelo utilizado para este proyecto [23], denominado **Raspberry Pi 3 Model B** y lanzado en 2016.



Figura 2.1: Raspberry Pi 3 Model B.

Las características son las siguientes:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- Base Ethernet
- 40-pin extended GPIO
- 4 USB 2 ports
- 4 Pole stereo output and composite video port
- Full size HDMI
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- Micro SD port for loading your operating system and storing data
- Upgraded switched Micro USB power source up to 2.5A

Gracias a este modelo que ya cuenta con una arquitectura ARM, con un microprocesador bastante potente para el tamaño de la placa y con una suficiente capacidad de memoria RAM, existe la posibilidad de trabajar sobre un sistema operativo derivado de GNU/Linux, denominado **Raspbian** que se detallará más adelante, en la que ya podemos incorporar distintos compiladores o intérpretes de lenguajes de programación. Es importante mencionar esto, ya que se debe de utilizar por ejemplo, el lenguaje de programacion **JAVA** y **PYTHON**, para el desarrollo de esta solución a este problema, y este elemento de la Figura 2.1 es uno de los mejores que se adaptaban a este proyecto.

Otras de las cosas extraordinarias de este componente es el consumo energético, es decir, esta placa con se puede ver en las características, trabaja mediante 5V y una intensidad máxima hasta de 2.5A, por lo que con este mínimo consumo de 5V estará alimentado todo el sistema.

Y no olvidar el coste de este elemento, aunque yo tuve la oportunidad de encontrar una buena oferta en el mercado como importación desde China, su precio ronda aproximadamente entre unos **24€**.

2.1.1. ¿Por qué Raspberry Pi y no otro componente?

En el trabajo relacionado 1.2.1 encontramos los sensores y dispositivos que mejor se adaptaban a este tipo de problemas, por el mínimo consumo que necesitaban (en algunos casos 3,3V o 5V como máximo), posiblemente el pequeño tamaño de estos dispositivos y la condición más tomada en cuenta, como es el precio, que más adelante se detallará cada uno de ellos, así como su precio y sus características más importantes.

Para el control de estos dispositivos principalmente es necesario tres conexiones, un voltaje de +5V, una toma a tierra, y una última conexión de entrada y salida de datos, dedicada a recibir la información del sensor o la posibilidad de comunicarnos con dicho sensor. Todo esto también los podemos ver mas detalladamente en los siguientes puntos.

Estos sensores fueron diseñados por el motivo de la comercialización de una placa anterior a la Raspberry Pi, denominada Arduino [2], comercializada en 2005 y nombraba por el mismo nombre de la fundación que la diseñó. Esta placa tuvo bastante éxito en el ámbito de la robótica, en la que se utilizó para muchos proyectos tecnológicos y dieron lugar a la comercialización de una infinidad de sensores.



Figura 2.2: Placa Arduino UNO.

Resumidamente, esta placa entre sus características más importantes en comparación con la Raspberry Pi son:

- Tiene un mayor rango de voltaje de alimentación, hablamos **entre 6V y 12V** que necesita Arduino, respecto a las 5V que utiliza Raspberry Pi.
- Arduino no dispone de un microprocesador como Raspberry Pi, sino que cuenta con un **microcontrolador**, el cual no permite cargar un sistema operativo como la Raspberry Pi, o incluso instalar algunos paquetes necesarios como lenguajes de programación o alguna plataforma indispensable para mí como es este caso. Por ejemplo la máquina virtual de java (**JVM**).
- El lenguaje por excelencia de Arduino es **C++**, mientras que Raspberry Pi puedes elegir el lenguaje o intérprete que deseas utilizar.
- El coste de una placa y otra apenas es relevante, ya que los dos elementos se aproximan en su mismo precio.

La decisión final fue saber que estos sensores son totalmente compatibles por ambas placas, la necesidad de tener un sistema operativo independiente y poder tener disponibilidad de incluir tanto plataformas diferentes como otros paquetes en Raspberry Pi, por igualdad de precios, es muy interesante y competitiva la decisión de adquirir Raspberry Pi.

2.1.2. Pines GPIO Raspberry Pi 3 Model B.

Antes de entrar en detalles con los sensores y los componentes específicos, es necesario conocer sobre que son los GPIO y en que consisten, ya que forma parte de la Raspberry Pi y es fundamental para la conexión de todos los sensores y controladores para la vivienda domótica. Si nos fijamos en la Figura 2.1, podemos encontrarlos en la parte superior de la misma foto, es fácil de localizar ya que este modelo son los únicos pines que se encuentran en la placa y suman un total de 40 pines.

GPIO (General Purpose Input/Output) [43], es un pin genérico de entrada y salida, cuyo comportamiento puede ser controlado en tiempo de ejecución mediante un lenguaje de programación. Como se verá en la implementación de estos pines más adelante, para cada pin que vayamos a utilizar debe de definirse su comportamiento en tiempo de ejecución. Un pin GPIO podemos activarlo o desactivarlo en cualquier momento, pero para activarlo debemos definir que comportamiento va a tomar dicho pin. Estos estados pueden ser dos tipos, de salida o de entrada. Como su propio nombre indica, cuando definimos un GPIO de modo entrada, quiere decir que activamos dicho GPIO para consultar su estado del GPIO en un momento determinado o para obtener información de lectura sobre ese pin. En caso contrario, si definimos el comportamiento de un GPIO en modo salida, habilitamos en este caso el pin para enviar información a través de él o para poder modificar su estado en un momento determinado.

La **interacción** mediante estos GPIOs ya sea para entrada o salida, se utilizan señales digitales o analógicas, por simplicidad se habla sobre las señales digitales que son las más utilizadas usualmente, y las que se han utilizado a lo largo del desarrollo de este proyecto, pero sí mencionar que se pueden utilizar ambas. A groso modo, la diferencia está en que una señal digital como sabemos puede ser binaria, 0 o 1, en aspectos un poco más específicos se utiliza una señal alta como por ejemplo 5V para identificar un 1, y en caso contrario una señal baja de 0V para identificar un 0. En cambio, las señales analógicas nos permiten valores intermedios entre este rango de 0 o 1, por ejemplo, 0.2, 0.3,... Cuando analizamos los sensores utilizados se comprenderá mejor este apartado, pero por poner un ejemplo más claro en este momento, imaginemos que tenemos un sensor de luminosidad, que toma medidas sobre la cantidad o intensidad de luz que hay en un determinado momento, en el caso digital, podemos obtener 0 si no hay luz ó 1 si hay el máximo de luz que pueda medir dicho sensor. Para este caso si sería interesante utilizar señales analógicas para obtener mejor precisión del valor de luminosidad, por ejemplo, si obtenemos 0.2 podemos saber que la intensidad de luz es un poco baja pero existe luz, o 0.6 que podríamos deducir que hay una luz relativamente adecuada y no una intensidad demasiado alta.

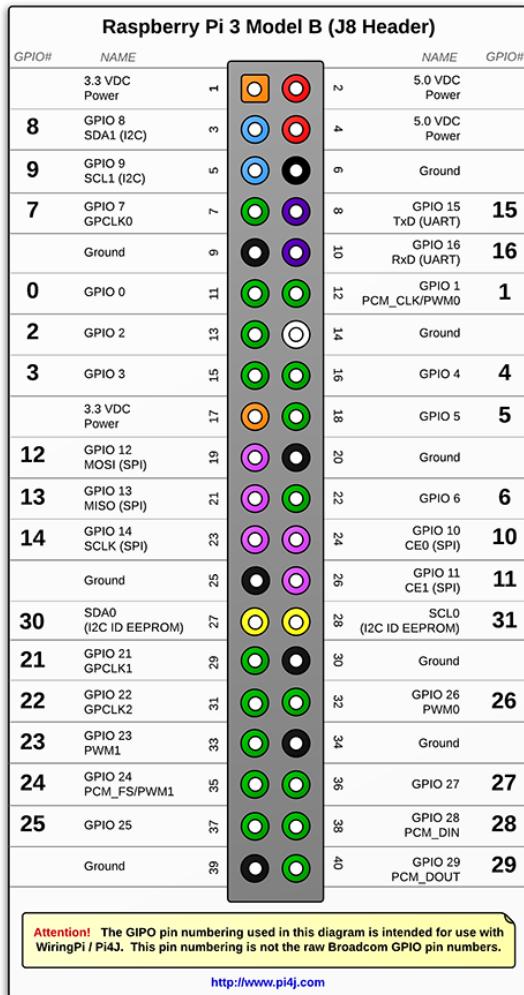


Figura 2.3: Numeración Pines GPIO Raspberry Pi 3 Modelo B.

En la figura anterior 2.3 podemos ver el orden y enumeración de los GPIO para el modelo concreto de Raspberry Pi según la librería **Pi4J** que hablaremos de ella también más adelante. Hay que tener especial cuidado, porque dependiendo del modelo de placa que estemos manejando, y también dependiendo de la librería de programación que estemos utilizando, pueden verse afectados la numeración y el orden de los GPIO [31]. Incluso también podemos ver en esta figura que existen distintos tipos de pines, aunque todos los pines de esta placa pueden ser utilizados como digitales, hay que tener especial cuidado con los pines que no son GPIO, como son los de voltaje, identificados por el color negro para una conexión de tierra, el rojo para una conexión de +5V, y el color naranja para +3V.

El fabricante insiste mantener cuidado con los voltajes, ya que Raspberry Pi trabaja a un voltaje de 5V, que es el máximo voltaje que puede admitir cualquier GPIO, si por error hacemos llegar mayor voltaje a cualquiera de estos pines podemos tener **serios problemas** con la placa.

2.2. Sensores y dispositivos

La cantidad de diferentes sensores, controladores, actuadores, placas de expansión... disponibles para Arduino y Raspberry Pi es incontable, es cierto que hay una enorme abundancia de elementos que ni podríamos imaginar, detector de movimiento, detección de obstáculos, humo, fuego, vibraciones, infrarrojos, luminosidad, lluvia, relés, peso, calidad del aire, humedad, temperatura... se puede ver rápidamente en un catálogo ofrecido por la empresa "*Cetronic*" de todos los componentes electrónicos disponibles [5], no es el objetivo realizar publicidad o comparaciones de precios, sino que es una de las páginas Webs que cuenta con mayor variedad, para darnos cuenta de la infinidad de componentes y posibilidades que existen a día de hoy. Al encontrar con una situación indecisa por no saber exactamente que sensores adquirir o poder utilizar en este proyecto, ya que todos podrían ser incorporados, existe un pequeño "Kit" que contiene unos 37 sensores y componentes más usuales, que podrían ser de los más destacados, y ayudarían a satisfacer las necesidades principales de una vivienda domótica.

En los siguientes puntos cuando se detalle sobre el diseño del proyecto, se podrá mostrar que la cantidad de sensores no es tan relevante, al dotar al sistema de esa escalabilidad necesaria, nos permite incorporar nuevas funcionalidades o ampliación de componentes en cualquier momento. Por simplicidad se ha escogido varios de ellos y ahora se detallará cada uno de ellos, así como su funcionalidad y sus características más importantes.

Para el **correcto funcionamiento** de la mayoría de estos sensores, es necesario realizar un circuito electrónico específico para cada uno de ellos, es decir, a cada sensor debe aplicarse en la mayoría de los casos unas resistencias necesarias para el correcto funcionamiento y no se provoque ninguna anomalía en el sistema o en el propio sensor. En este caso ya todos los sensores vienen incluidos en un pequeño módulo, el cuál, ya contiene los componentes electrónicos necesarios para el correcto funcionamiento de cada sensor.

2.2.1. DHT11 Sensor de temperatura y humedad.

Este sensor [27] va a ser el encargado de tomar la temperatura y humedad en un determinado momento, la temperatura se obtiene en grados Celsius,

y la humedad en tanto por ciento. Este sensor es capaz de tomar medidas de temperatura entre el intervalo 0 y 50 grados con un margen de error de 2°C, al igual que la humedad entre 20 % y 80 % con un margen de error del 5 %.

Es cierto que existe un sensor denominado "DHT22" posterior a éste modelo ("DHT11"), con mejores prestaciones y una precisión mayor, como por ejemplo, un margen de error de 0.5°C para la temperatura. El funcionamiento es exactamente el mismo para ambos, y el precio ronda aproximadamente **0.70€**.

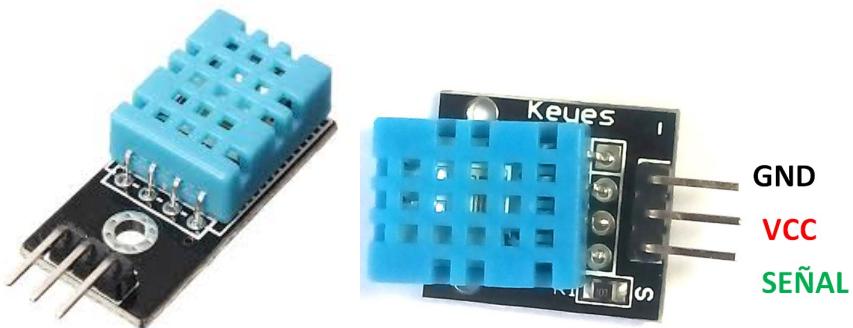


Figura 2.4: DHT11 Sensor de temperatura y humedad.

Como podemos ver en la Figura 2.4, este módulo cuenta con 3 pines, pero hay que tener especial cuidado en las referencias del fabricante, para la correcta conexión (GND o tierra, Vcc o fuente de alimentación positiva +5V, y el pin para entrada y salida de datos), ya que pueden variar los módulos dependiendo del fabricante o el proveedor. En esta situación como es mi caso, podemos fijarnos en la Figura 2.3 como se realizaría la correcta conexión, pero se insiste que cada módulo puede ser distinto.

El funcionamiento si puede ser un poco más complicado que el resto de sensores, en este caso el proceso consiste en:

1. Hay que activar el GPIO de la Raspberry Pi en el que se conectado, para emitir un "1" por dicho pin, es decir, activar el GPIO como salida y enviar el valor binario "1". Esto hace que el dispositivo accione y comience a realizar la medición, ya que mientras no reciba esta señal, el sensor queda inhabilitado".
2. Una vez enviada la señal digital y sea recibida por el sensor, comenzará a realizar la medición y nos enviará 5 secuencias de 8 bits cada una. Por lo que inmediatamente después de enviar la primera señal digital, hay que cambiar el comportamiento del GPIO, es decir, habilitar en

modo entrada, para recibir esta información. Cada secuencia una vez recibida hay que transformarla a formato decimal, y cada una de ellas pertenece:

[–] La primera secuencia pertenece a la parte entera de la temperatura.

[–] La segunda secuencia es la parte decimal de la temperatura, en este caso, este sensor siempre va a enviar 0, ya que no está diseñado para obtener tal precisión, para ello tendríamos que utilizar el sensor "DHT22" que si permite esta precisión.

[–] La tercera secuencia pertenece a la parte entera de la humedad.

[–] La cuarta secuencia de 8 bits coincide con la parte decimal de la humedad, que en este caso ocurre lo mismo que con la temperatura, el sensor enviará 0 porque no está preparado para tal precisión.

[–] La última secuencia es utilizada como un "*check*" o secuencia de paridad, es decir, debe de coincidir esta última secuencia de 8 bits obtenida del sensor con la suma de las 4 secuencias anteriores. Esto nos confirmará que la recepción de las secuencias o la información recibida ha sido correcta, y no ha habido ninguna perdida de información o ha ocurrido algún error. En caso de fallo debemos realizar de nuevo el proceso.

En el capítulo 4, se volverá a nombrar este proceso más concretamente mediante código para ver como sería una posible implementación real, y poder exponer esta solución propuesta.

2.2.2. Módulo KY-004 Botón o pulsador.

Este dispositivo denominado botón o pulsador, asumirá a mi proyecto la posibilidad de activar o desactivar manualmente el sistema de seguridad. Este componente tiene un sencillo funcionamiento, como podemos ver en la Figura 2.5 se indica cuáles son los correspondientes pines de dicho módulo, insistiendo que cada módulo puede ser distinto dependiendo del fabricante o proveedor, en este caso el pin 1 corresponde a tierra, el pin 2 a un voltaje +5V, y el tercer pin restante que debemos conectar a un GPIO de la Raspberry Pi, para más información y características se puede consultar [3].

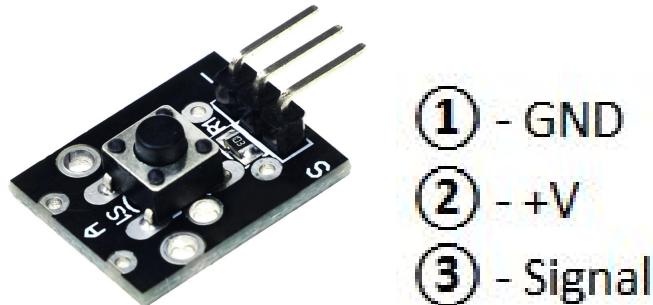


Figura 2.5: KY-004 Módulo pulsador.

Éste módulo está diseñado para tener un único funcionamiento, es decir, por el pin 3 dedicado a la conexión GPIO, podemos obtener una señal binaria "0" en el estado natural, es decir, no se está actuando sobre él, y "1" si ha sido presionado. Es bueno mencionar una particularidad tenida en cuenta, al ser un pulsador, cuando se presiona el botón se recibe como se ha mencionado anteriormente una señal digital "1", pero al soltar el botón también se envía una señal digital "0". Por ejemplo, algo parecido a la función en JAVA mouse_up y mouse_down, que son dos eventos distintos cuando se pulsa el ratón y cuando se suelta.

Respecto al coste de este componente fluctúa entre **0.42€** aprox. Se verá la implementación y más ejemplos más adelante.

2.2.3. Módulo KY-011 Módulo LED 2 colores.

Este sencillo componente está diseñado con un LED, que puede obtener el color verde o color rojo. En este proyecto se ha utilizado para indicar si el sistema de seguridad está activado o no, es decir, por defecto el sistema de seguridad se encuentra desactivado, es decir, luz roja, en cambio cuando se pulsa el sensor 2.2.2 anterior, se activará el sistema de seguridad y el LED cambiará a color verde.



Figura 2.6: KY-011 Módulo LED 2 colores.

Para este componente como vemos en la Figura 2.6, necesita la conexión de dos pines GPIO en nuestra Raspberry Pi, configurados en modo entrada, el pin 1 que esta dedicado a la toma a tierra (GND) y los dos pines restantes para los GPIO. Nos podemos plantear porqué este componente no necesita alimentación como los otros sensores, y se debe que, para que un LED funcione necesita un voltaje entorno a 1.6V o 3.3V, dependiendo de los materiales y la composición del diodo, estos valores son suministrados a través de los pines GPIO, que recordemos como se ha mencionado en los puntos anteriores, una señal digital "1"por uno de estos GPIO, puede equivaler a +5V. También incluyendo que este módulo no necesita emitir ninguna señal digital (sólo recibe) no necesita de una alimentación externa para generar estas señales.

Respecto al coste de este componente se encuentra entre **0.24€** aprox.

Entorno al funcionamiento es muy sencillo, en este caso, si se habilita el GPIO utilizado en la Raspberry Pi que corresponde al pin 2 del módulo (Color Verde) como señal digital "1" y el GPIO de la Raspberry Pi correspondiente al pin 3 del módulo (Color Rojo) como señal "0", el LED encenderá luz Verde, y al contrario se conseguirá luz roja.

Destacar que se ha de tener especial cuidado con los valores digitales, y asegurarse en cada momento en el que se vaya a cambiar el estado del LED, que unos de los GPIO obtengan el estado "0", para evitar posibles problemas que afecten al módulo.

2.2.4. YL-83, LM393 Sensor de LLuvia.

Este sensor como su propio nombre indica, identifica si está lloviendo en un determinado momento, a groso modo, analizando cuando hay existencia de agua en el sensor del módulo. Este componente consta de dos elementos como podemos ver en la Figura 2.7, el módulo LM393, que es la placa con pistas electrónicas encargada de identificar cuando hay existencia de agua

entre dichas pistas. Y el coste de estos dos componentes ronda los **0.65€**.



Figura 2.7: Módulo YL-83 y placa LM393 (Sensor de lluvia).

El objetivo de este elemento viene dado por la existencia de dos pistas independientes a lo largo de la placa, y cuando hay existencia de agua, como se puede saber el agua es conductora de la corriente cuando contiene sales minerales (no es conductora en estado puro), por lo tanto se produce un contacto entre ambas pistas y el sensor identifica que hay existencia de agua. El otro componente que vemos en la misma imagen, es el módulo encargado de suministrar la energía a la placa LM393, y dispone de dos pines más, uno para salida digital y otro para salida analógica como podemos ver en la Figura 2.8.

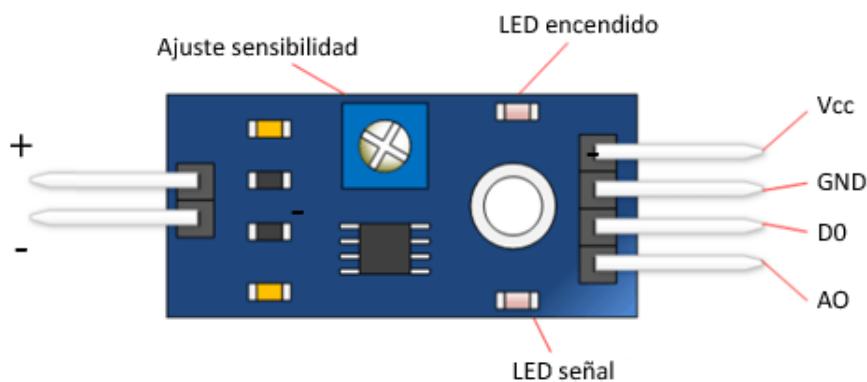


Figura 2.8: Conexiones Módulo YL-83 Sensor de Lluvia.

Como ha sido mencionado en el apartado 2.1.2, sobre la diferencia entre

señales digitales y analógicas, se encuentra que este sensor está preparado para emitir la información mediante ambas señales. Para mayor simplicidad y particularmente el caso que se ha utilizado es obtener la lectura por el pin digital. En el sensor, como podemos ver en la Figura 2.8, podemos encontrar un regulador de sensibilidad, que a través de él podemos calcular la cantidad de agua que debe, podemos decir "*acumular*", en la placa para que el sensor nos envíe una señal.

Por ejemplo, este módulo va a enviar por el pin digital (DO), "1" si no está lloviendo, pose por defecto, y "0" en el caso que detecte agua en la placa, este cambio entre uno y cero puede ser ajustado por el regulador de sensibilidad que mencioné anteriormente, a grosso modo, para que el sensor sea capaz de detectar lluvia con más o menos cantidad de agua en la placa. En [26] se puede encontrar más características específicas y las imágenes de esta subsección.

Por último introducir que este sensor se puede utilizar, como es lógico para identificar si está lloviendo o no, pero el objetivo importante no es sólo eso, lo interesante es el punto de introducir en cierta medida la "*inteligencia*" de la vivienda, en el que nos encontramos el siguiente caso:

- Lo más importante es que esté el sistema de seguridad activado, ya que si no, significa que no nos encontramos en casa o no queremos que esté activado el sistema autónomo de la vivienda. En tal caso, asumiendo que el sistema está activado:

[*] Si en el momento de detectar lluvia las persianas se encuentran abiertas (subidas), automáticamente los agentes se encargarán de cerrar dichas persianas.

[*] En caso contrario, si ya se encuentran cerradas, no tendrá tal efecto, tan sólo se enviará la notificación de la detección de lluvia.

- Posteriormente cuando la lluvia llega a su fin o parar de llover, el sensor nos indicará el momento que ya no se detecta agua, por lo que el sistema de agentes **inteligente** aprende cuando comenzó a llover en nuestra vivienda y el estado en que se encontraban las persianas (en este caso abiertas), por lo que vuelve a establecerlas en su punto de origen en el que se encontraban.

2.2.5. Módulo de conversión DC/DC de múltiple salida.

Éste módulo no tiene especial funcionalidad domótica, solo ha sido incluido al proyecto para poder suministrar los diferentes voltajes de energía necesaria para cada módulo del proyecto. Como se puede ver en la Figura 2.3 podemos ver que la Raspberry Pi cuenta con cuatro pines GPIO de voltaje, dos de ellos para 3.3V y otros dos para 5V, por lo que una vez que

superemos 4 componentes externos como son cualquiera de estos módulos de sensores detallados, nos quedamos sin salidas de voltaje de la Raspberry Pi.

Es cierto que cualquier pin GPIO de la Raspberry Pi, (de los 40 pines disponibles), excepto los pines GND (toma tierra) o los propios pines de voltaje, podemos habilitarlos como pines de salida y con una señal digital alta, consiguiendo así por ese pin una alimentación de +5V, pero **recordemos** que la propia Raspberry Pi trabaja a +5V y un máximo de intensidad de 2.5A, por lo que la alimentación de la propia Raspberry Pi para su mínimo funcionamiento puede verse afectado. Según en [23] la información proporcionada por el fabricante, Raspberry Pi prioriza la energía mínima para su funcionamiento interno (microprocesador, memorias, vídeo, Wi-Fi...) y posteriormente suministra el voltaje restante repartido a los demás componentes, pero mi intención no es quedarme sin energía en un momento determinado y algún sensor deje de funcionar. Directamente aprovechando toda la energía externa para los distintos componentes y dejando libre de carga a la propia Raspberry Pi.

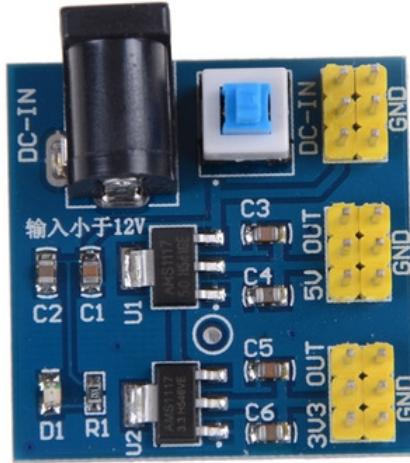


Figura 2.9: Módulo de conversión DC/DC de múltiple salida.

Por ello se incluye este componente como podemos ver en la Figura 2.9, su alimentación externa de 6V o 12V, y como vemos en la misma figura anterior, proporciona:

- 3 salidas de 3.3V.
- 3 salidas de 5V.

- Y las tres salidas restantes que podemos identificar como DC-IN en la placa, que estas salidas pueden variar en función de la alimentación externa que hayamos aplicado, es decir, si esta placa la alimentamos con 6V, estas salidas tendrán 6V cada una, si por el contrario aplicamos 12V como alimentación externa, estas salidas que estoy mencionando darán salida de 12V.

Cada salida se compone de dos pines, la salida positiva del voltaje correspondiente y su conexión a tierra (GND). Particularmente en este caso, se ha utilizado una alimentación externa de 12V, ya que se necesita dos salidas a 12V para alimentar los motores encargados de accionar sobre la puerta del garaje y las persianas como se explicará más adelante. El coste de este componente es **1.04€** aprox.

2.2.6. Módulo L298N - Puente H.

Este módulo aunque tiene un sencillo funcionamiento y para este proyecto un simple objetivo, tiene capacidad de realizar mayor funciones y más específicas, por lo que puede ser un poco complicado de introducir, o detallar cada una de ellas.

Para simplificar un poco este punto y sea mejor comprendido, esta placa se puede utilizar básicamente para controlar dos motores DC, es decir, los pequeños motores de corriente continua a una sola fase, o podemos controlar un motor paso a paso de dos fases, los cuales se utilizan para giros o rotación exactas y con mucha precisión, por ejemplo, los motores de rotación de un disco duro, lector de CD...

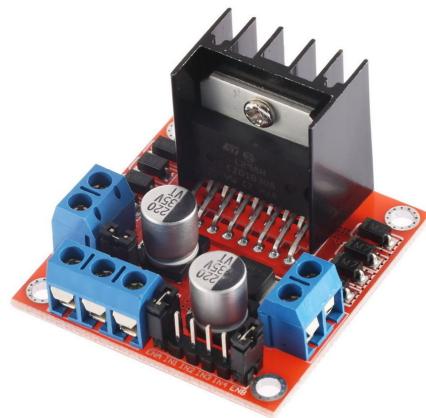


Figura 2.10: Módulo L298N - Puente H.

El nombre de "*Puente H*" viene dado por un simple motivo, sabemos que un motor DC de corriente continua simple, gira en ambos sentidos, es decir, derecha e izquierda, dependiendo del sentido de la corriente. Esta placa se encarga de facilitar esta funcionalidad, y no sólo de un motor sino de dos motores independientes, por ello utiliza sus conexiones internas formando un circuito en forma de H, que hace posible esta funcionalidad que estamos mencionando. También se tiene que tener especial cuidado en las conexiones de este componente ya que admite varias configuraciones, y dependiendo del voltaje que se vaya a trabajar hay que realizar una configuración u otra. En [1] se puede encontrar un pequeño tutorial, que aunque está diseñado para Arduino, la configuración electrónica es la misma y sirve de ayuda para encontrar la configuración que se busca.



Figura 2.11: Jumper Módulo L298N - Puente H.

Se intentará hacerlo lo más simple y resumidamente posible, a muy groso modo, el jumper que vemos identificado en la Figura 2.11 podemos decir que es lo más importante a tener en cuenta, ya que si se encuentra ese jumper como en la Figura 2.11, entonces tenemos habilitado el regulador de voltaje integrado en la placa, por lo que a este módulo le podemos suministrar un voltaje máximo de hasta +12V, y tendremos dicho voltaje disponible para alimentar a los motores.

En tal caso de quitar este jumper anterior, se está deshabilitando el regulador de la placa, y ahora permitiría alimentación de hasta máximo 35V.

Ahora veremos como se realizan las conexiones, pero era importante mencionar esta pequeña introducción anterior, en este caso personalmente se ha utilizado una alimentación a la placa de 12V, ya que se necesita un

poco de potencia en los motores DC para la elevación y cierre de la puerta del garaje, ya que con conexiones entre 3.3V y 6V no son suficientes.

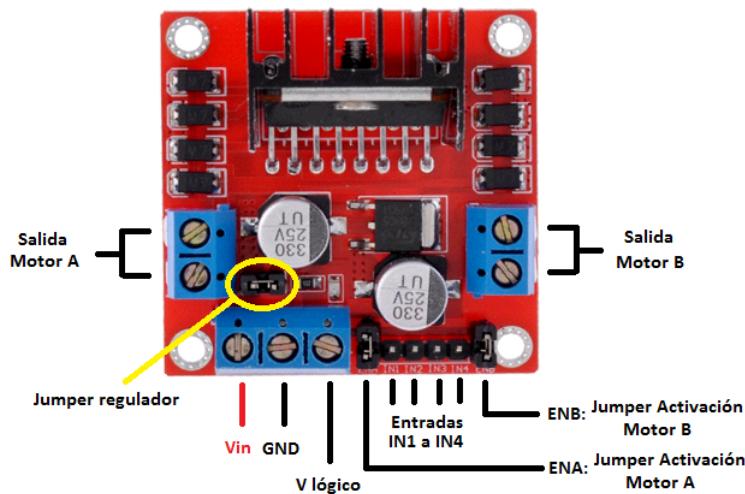


Figura 2.12: Configuración electrónica Módulo L298N - Puente H.

Ahora nos fijamos en la Figura 2.12, y podemos encontrar la identificación de la mayoría de los elementos de esta placa, por simplicidad se va a nombrar los más importantes y necesarios.

- El jumper regulador como se ha mencionado anteriormente.
- Vin, que es la entrada de voltaje positivo para la alimentación de la placa, si tenemos activado el jumper regulador, y como fue en mi propio caso, Vin es la entrada de +12V.
- GND, aquí si es muy importante una consideración, GND como su propio nombre indica es la conexión a tierra, pero es muy importante que conectemos un cable a la tierra (GND) perteneciente a la fuente de alimentación que hayamos utilizado, y otro cable GND a unos de los GPIO de las Raspberry Pi que corresponda a tierra (GND), para tener disponibles ambas conexiones a tierra, con esa seguridad de evitar problemas ante cualquier situación de corto-circuito.
- Y por último las entradas IN1 - IN4, que son los pines reservados para el control de los motores, en función de la señal que indiquemos. Por eso en este caso, estos pines son utilizados por 4 GPIO conectados a la Raspberry Pi.

Una vez que ya hemos realizado esta configuración anterior, tendremos la posibilidad de controlar dos motores DC, uno de ellos por la salida Motor A, y el otro por la salida Motor B, como vemos en la Figura 2.12. La secuencia de señales digitales que se debe enviar por los GPIO podemos encontrarla en [45], así como también podemos encontrar la secuencias necesarias para un motor paso a paso, por ejemplo, se utilizará este módulo cuando sea necesario la acción sobre la puerta del garaje o la persiana. Más adelante se incluirá las sentencias que se ha utilizado en este proyecto.

El coste de este módulo ronda los **1.46€**.

2.2.7. Módulo KY-033 - Tracking Sensor Module.

Este sensor también llamado "seguimiento de líneas" o "detección de obstáculos", funciona a través de sus dos sensores infrarrojos, el emisor que emite una señal infrarroja a una frecuencia determinada, y el receptor que es capaz de detectar si recibe una reflexión de dicha señal infrarroja. La distancia o la intensidad de esta señal infrarroja puede ser configurada por el regulador de sensibilidad de este módulo, como podemos ver en las características proporcionadas por uno de los diferentes proveedores [24] que comercializa este componente, y nos indica que la percepción podría ser óptima entre 2 y 40cm.

Possiblemente este sensor no sea el más recomendado para el uso común de detección de movimiento como es este caso, sabiendo que el más adecuado para estos casos es el módulo de detección "HC-SR501", Figura 2.14, que está dedicado para este fin y es capaz de detectar dicho movimiento desde 3 a 7 metros. Se menciona esto ya que en el "Kit" disponible no se encuentra este componente y por ello se ha utilizado el módulo ky-033, Figura 2.13, para simular esta funcionalidad.

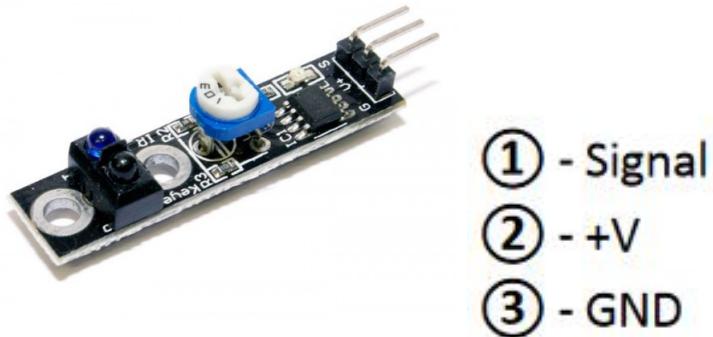


Figura 2.13: KY-033 Módulo Sensor de Seguimiento.

Como se ha mencionado anteriormente, el módulo sensor de la Figura 2.13 emite una señal infrarroja a una distancia determinada y es captada por el sensor receptor de dicha señal si existe alguna reflexión, por lo que se utilizará este componente en este proyecto para simular un detector de movimiento, ya que, cuando un objeto opaco se coloque una distancia entre 2 y 3cm frente a este sensor, detectará una intrusión, y emitirá "1" (señal digital alta) por el pin de salida digital, que ayudará a identificar si se produce un movimiento de un objeto en esa distancia.

Este componente suele tener un precio desde **0.40€**.



Figura 2.14: Módulo HC-SR501, sensor de movimiento.

2.2.8. Módulo cámara.

Por último, este componente utilizado consta de una simple cámara compatible con Raspberry Pi [18], denominada "Camera V2", entre sus carac-

terísticas principales la calidad de imagen, en este caso, 5 megapixel, generar vídeos a 1080p (Full HD), y una capacidad máxima de grabación de 30 fps(frame rate). Aunque en el mercado se encuentran ya diferentes modelos con mejores prestaciones, cámaras nocturnas...

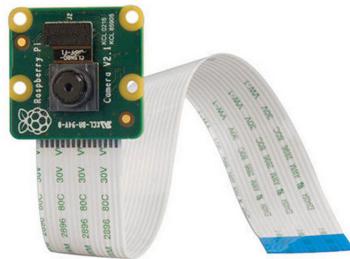


Figura 2.15: Módulo cámara para Raspberry Pi.

Este módulo se conecta a la Raspberry mediante el conector CSI (Camera Serial Interface), que mediante la interfaz MIPI (Mobile Industry Processor Interface) el procesador anfitrión se comunica con la cámara digital, este conector lo podemos identificar en la Figura 2.16.

Mediante una simple configuración, y la utilización de la librería adecuada se puede utilizar este componente para realizar una foto, un video... Estos aspectos de programación se verá más en detalle en el capítulo de Implementación.



Figura 2.16: Conector CSI para cámara Raspberry Pi.

El precio de esta cámara ronda los **4€** dependiendo del modelo de la cámara y el proveedor.

2.3. Plataforma Magentix

Magentix es una plataforma de programación de agentes y sistemas multi-agente abierta, desarrollada en la Universidad Politécnica de Valencia [18], esta plataforma proporciona todos los servicios de bajo nivel necesarios para la creación, comunicación e interacción entre los agentes. Aunque en el siguiente punto se definirá que es un agente, que puede hacer y en qué consiste, ahora se explicará esta plataforma, la cuál es el medio por el que los agentes se pueden crear y llevar a cabo sus comunicaciones.

Los requisitos indispensables para poder lanzar esta plataforma son mínimos, el más importante que esté disponible la máquina virtual de JAVA (JVM), Oracle Java Development Kit (JDK) 6 o posterior, que podemos encontrar en [30] si no se encuentra instalada en el sistema que estamos utilizando. Posteriormente dependiendo del sistema operativo que se use y su arquitectura, debemos de descargar en [11] la plataforma Magentix correspondiente en formato ".jar" e lanzarlo. En este caso como se ha mencionado en el apartado 2.1, Raspbian es un derivado de linux, con una arquitectura de 64 bits, porque lo que se va a exponer este caso particular. En caso de otras alternativas toda la información necesaria la podemos encontrar en el manual de usuario de Magentix2 realizado por el propio grupo de investigación [42].

Una vez se tiene descargada la plataforma Magentix debemos de lanzar la instalación con la siguiente orden:

```
java -jar Magentix2Desktop-x64.jar
```

Recordar dar permisos "Root" para la correcta instalación, una vez termina este proceso, veremos que se crea un nuevo directorio como podemos ver en la Figura 2.17, donde se encuentra los script Start-Magentix.sh y Stop-Magentix.sh, necesarios para levantar la plataforma y cancelarla.

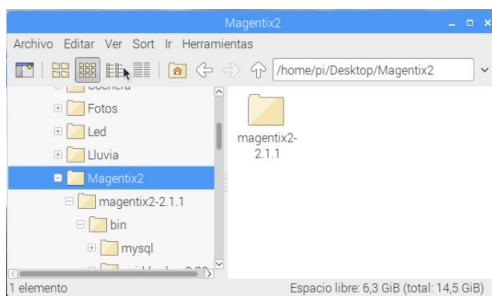


Figura 2.17: Directorio generado posteriormente a la instalación de Magentix2.

Es necesario disponer de la plataforma Magentix2 ejecutada antes de

crear o lanzar algún agente como podemos ver más adelante, la orden correspondiente es la siguiente:

```
sh ~/Start-Magentix.sh
```

Antes de entrar en detalles sobre los agentes, queda por mencionar que esta plataforma sigue el estándar de comunicación ACL (Agent Communications Language) reconocido internacionalmente y establecido por **FIPA** (Foundation for Intelligent Physical Agents). Este estándar implica a diseñar y modelar el estilo de comunicación, es decir, los tipos de mensajes que se pueden enviar y recibir, así como la estructura de cada mensaje.

Así como también la mejora sobre la anterior plataforma Magentix, proporcionando nuevos servicios, herramientas para mejorar la seguridad y optimización de este sistema multiagente que no entrará en más detalles por viabilidad de este proyecto, pero todo esto disponible en [42].

2.4. Agentes

Un agente se conoce como un sistema software situado en un entorno y capaz de actuar de forma independiente o autónoma para conseguir sus propios objetivos, en función de sus capacidades y sus propiedades diseñadas. Cuando nos encontramos con más de un agente en un mismo entorno hablamos de un sistema **multiagente**, los cuales interactúan los unos con los otros, para la consecución de sus propios objetivos o de objetivos comunes, para lo cual necesitan comunicarse, cooperar, coordinarse y negociar entre ellos.

Dentro de estos agentes y diseñado por Magentix, nos podemos encontrar diferentes tipos de agentes, en función de sus capacidades comunicativas, entre ellos los más importantes:

- BaseAgent, es el tipo de agente más básico capacitado para enviar y recibir mensajes, por lo que nuestro agente debe heredar de esta clase.
- SingleAgent, es el mismo tipo de agente que BaseAgent, pero con una pequeña diferencia, en la que se define una nueva recepción de mensajes, es decir, ahora la recepción de un nuevo mensaje es bloqueante, el agente se quedará a la espera de recibir dicho mensaje sin poder realizar otra operación. Aunque esto veremos más adelante que puede ser relativo y veremos como se puede conseguir que el agente no se quede en estado bloqueado. En este caso, nuestro agente debe de extender de la clase SimpleAgent.

Todas estas clases están disponibles en la librería magentix2-2.01-jar-with-dependencies.zip disponible en [41], como son "es.upv.dsic.gti_ia.core.

BaseAgent” y ”es.upv.dsic.gti_ia.core.SingleAgent” respectivamente.

Esto es una de las mejores posibles soluciones a este problema, ya que ahora se puede contar con un agente que se encargue de cada sensor, se comuniquen entre ellos, y consigan un objetivo común, por ejemplo, activar el sistema de seguridad, cerrar las persianas en caso de lluvia... Proporcionado con un sistema de **seguridad** y una gran **optimización**. Así como lo más importante, cada agente se encarga de un módulo sensor, facilitando tanto el **diseño** como la **implementación**, evitamos que se produzca colisiones y el incremento exponencial de dificultad a medida que se van añadiendo sensores y funcionalidades al sistema, es decir, nos olvidamos de tener un programa principal que controle todo y tenga en cuenta todas las posibles situaciones, que tanto se estén dando como puedan ocurrir.

Por ello, en el Capítulo 4 se detalla el tipo de cada agente que se ha diseñado y lo necesario para su diseño.

2.4.1. ACL (Agent Communications Language).

Mencionado justo en el punto anterior, los agentes de esta plataforma siguen el estándar de comunicación ACL (Agent Communications Language), en el que podemos encontrar los diferentes tipos de mensajes, denominado performativas [40], entre las más destacadas (existen más tipos):

- Accept-proposal. Aceptar una negociación.
- Agree. Añadir una propuesta para una petición.
- Cancel. Cancelar una petición o una operación ya comenzada.
- CFP (Call For Proposal). Iniciar una negociación entre agentes.
- Confirm. Confirmación sobre una petición realizada.
- Failure. Mensaje enviado cuando se produce un fallo o error realizando una tarea.
- Inform. Generalmente, se usa para informar sobre cualquier contexto.
- Not-Understood. De su traducción en inglés, ”No te comprendo”, se utiliza para cuando un agente no está preparado para responder, o la formulación del mensaje no ha sido correcta.
- Query-if. Realizar una consulta a un agente.
- Refuse. Rechazar una tarea o petición.
- Request. Pedir una acción o tarea a un agente.

- Subscribe. Realizar un proceso de suscripción en una comunidad de agentes.

Cada uno de estos elementos tiene su propio fin, aunque pueden parecer lo mismo algunas performativas para realizar la misma funcionalidad, pero cada performativa tiene su particularidad, como por ejemplo, **REQUEST** y **CFP** tienen el mismo objetivo, pedir a un agente una tarea o un proceso determinado, pero en cambio, la primera performativa se utiliza para encargar dicha tarea al agente, independientemente si está disponible y puede realizarla o no, y la segunda, inicia un proceso de negociación entre varios agentes para consultar quién está disponible y se ofrece para realizar dicha tarea. En el apartado 3.2 podemos ver cada performativa usada, en qué situación y para qué fin.

Un mensaje, independientemente a la performativa que pertenezca, este lenguaje estándar aplica a dichos mensajes una estructura específica, dividida en diferentes campos:

- Performative. Tipo de performativa de las mencionadas anteriores a la que pertenece el mensaje.
- Sender. Identificador o nombre del agente emisor
- Receiver. Identificador o nombre del agente receptor, en este caso se pueden incluir varios, ya que el mensaje puede ser recibido por varios agentes.
- Reply-to. En caso de que el mensaje pase a través de un agente intermedio, este campo será necesario para saber quién es el receptor final.
- Content. Contenido del mensaje.
- Language. Idioma
- Encoding. Normalmente utilizado para indicar el formato del contenido del mensaje. Por ejemplo, JSON.
- Ontology. Significado de los objetos del contenido.
- Conversation-id. Cada vez que se abre una conversación se puede generar una identificación única, la que se utiliza para identificar los mensajes de una conversación determinada.
- Reply-with. Identificación que se debe usar para contestar a este mensaje.
- In-reply-to. Identificación del mensaje al que se quiere responder.

Aunque existen más diferentes parámetros, dependiendo de la complejidad de las conversaciones entre agentes, se suelen usar mas o menos parámetros, es cierto que a más información se aporte a los mensajes mas riqueza contendrá la conversación, y será más fácil de identificar la traza de una conversación en busca de fallos o errores por parte del programador. En situaciones más simples algunos de estos parámetros se pueden omitir o se obvian para conseguir una rápida funcionalidad sin que llegue a ser muy compleja.

2.5. Raspbian

Como se puede ver en el apartado 2.1, la fundación Raspberry Pi [21] ofrece como organización benéfica, el sistema recomendado y optimizado para Raspberry Pi, basado en una distribución de GNU/Linux concretamente Debian, de donde procede el nombre del sistema.

Este sistema ya parte con algunas herramientas y plataformas disponibles nativas, como por ejemplo, Python, Scratch, Sonic Pi, Java...

En [19] podemos encontrar el sistema Raspbian para su descarga, mencionar que nos encontramos ante dos posibilidades:

- Raspbian Stretch with desktop. Esta imagen contiene el sistema Raspbian con entorno gráfico, es decir, contiene toda la funcionalidad gráfica como, menús, ventanas, iconos, fondo de pantalla...
- Raspbian Stretch Lite. En este caso contamos con el mismo sistema completo, pero sin entorno gráfico, adecuado para aquellos usuarios más avanzados.

En este caso se utilizó la primera opción, con entorno gráfico por simplicidad y comodidad para utilizar ciertas herramientas, como por ejemplo, Netbeans para desarrollar directamente este proyecto.

Antes de nada, para seguir con la preparación de la Raspberry Pi, necesitamos una tarjeta de memoria "microsd", en la que la propia fundación recomienda un almacenamiento mínimo de 4GB. Hay que decir que dependiendo del sistema que elijamos, con entorno gráfico o no, puede ser recomendable un tamaño u otro, ya que el sistema con entorno gráfico puede llegar a ocupar unos 2GB aprox. y la versión lite tan sólo unos 40MB. Para este proyecto, personalmente se ha utilizado una tarjeta de memoria de 16GB, ya que se ha elegido el sistema completo con entorno gráfico y contaba con la mayor probabilidad de instalar distintas herramientas, a igual que con la cámara se va a generar imágenes e incluso vídeos, que se podría necesitar más almacenamiento.

Una vez esta disponible la tarjeta de memoria **MicroSD**, se debe de asegurar que este completamente vacía y lo más importante que se encuentre con el formato de archivos **Fat32**, para mayor seguridad se puede realizar un formateo rápido con este sistema de archivos.

El siguiente paso consiste en pasar la imagen Raspbian que se ha descargado anteriormente a la tarjeta MicroSD, para ello se puede utilizar cualquier programa que nos permita volcar una imagen a dicha tarjeta de memoria, por ejemplo, Win32 Disk Imager, Rufus, UNetbootin... En este punto se utilizó Win32 Disk Imager, de código abierto y lo podemos encontrar en [28].

Recordar descomprimir el archivo ".zip" que descarguemos del sistema Raspbian, para conseguir la imagen que necesitamos. Por último, abrimos el programa Win32 Disk Imager, y seguimos los siguientes pasos como podemos ver en la Figura 2.18:

1. Aquí seleccionamos el directorio donde se encuentra la imagen que hemos descomprimido anteriormente.
2. Seleccionamos el dispositivo que se corresponde a la tarjeta de memoria MicroSD.
3. Pulsamos sobre Write.

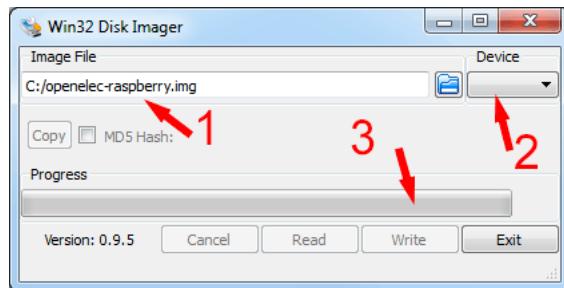


Figura 2.18: Proceso de instalación de Raspbian con Win32 Disk Imager.

Una vez finalizado el proceso, podemos extraer la tarjeta de memoria del ordenador y la introducimos en el Slot correspondiente en la Raspberry Pi, que la podemos encontrar en la parte inferior de la placa como podemos ver en la Figura 2.19.

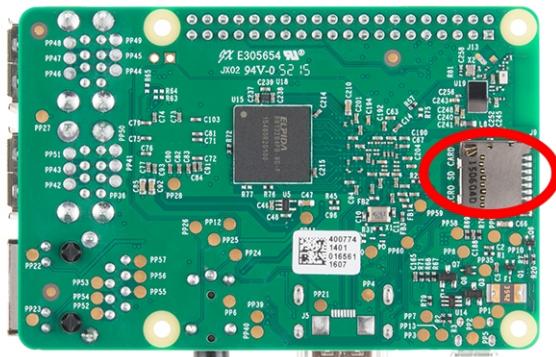


Figura 2.19: MicroSD Card Slot Raspberry Pi 3.

Ya por último podemos encender la Raspberry Pi conectándole una alimentación de 5V, y si todo ha ido bien y no ha ocurrido ningún problemas podemos ver como arranca el sistema Raspbian correctamente 2.20.



Figura 2.20: Sistema Raspbian.

Capítulo 3

Propuesta de solución

3.1. Introducción a la solución.

Ya pasada la parte teórica y de investigación sobre con que nos encontramos actualmente 1, cuáles son los principales problemas y objetivos a buscar, que podemos utilizar para abarcar este problema propuesto o con qué tecnologías disponemos actualmente que mejor se adaptan 2, ahora se da paso al análisis de cómo se puede desarrollar esta solución para llevarla acabo.

En primer lugar, se cuenta con una serie de componentes, módulos de sensores, que ya se ha identificado cuál es su funcionalidad de cada uno de ellos, y que se va a utilizar para este proyecto. Estos sensores pueden ser conectados a la Raspberry Pi, y para que tanto el diseño como la implementación sean más simples, obteniendo una estructura de todos ellos, cada sensor debe ser controlado por un agente.

También este diseño cuenta con un agente controlador, que se encarga de controlar a todos los demás agentes del sistema, es decir, en vez de contener un sistema de agentes, en el que cada uno trabaja a su voluntad y sin ninguna supervisión, se va a desarrollar un agente controlador que supervise estos agentes, y sea el encargado de que distribuir las acciones pertinentes ("es el jefe"), y lo más importante va disponer de la capacidad de decidir por él mismo en las distintas situaciones que se puedan producir.

Este sistema obtiene el nombre de una sociedad de agentes jerárquica, es decir, los agentes cooperan debido a que reciben órdenes de una autoridad para hacerlo, en este caso es el controlador, en la que aparece una comunicación para controlar que las órdenes se han ejecutado.

El lenguaje de programación que se debe de utilizar para programar estos agentes de Magentix2, como se vio en su apartado correspondiente, es

JAVA, a su vez cada agente va a tener un comportamiento distinto, pero no influye en sus estructura interna, por lo que en la siguiente sección se va a detallar los diagramas de secuencia y actividad para que se refleje dicha estructura y su posible comunicación de cada agente.

El diagrama de clases del sistema es el siguiente como podemos ver en la Figura 3.1:

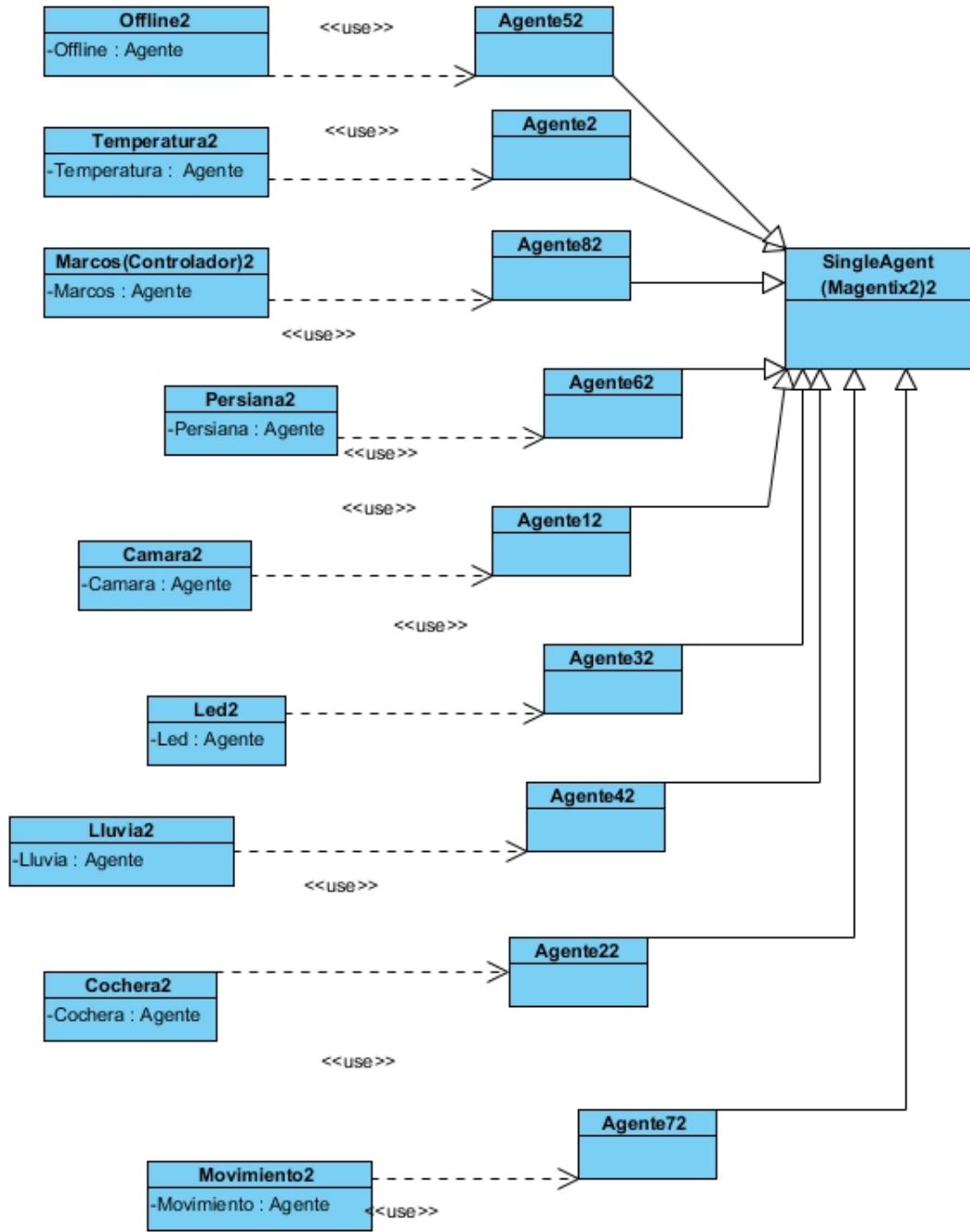


Figura 3.1: Diagrama de Clases.

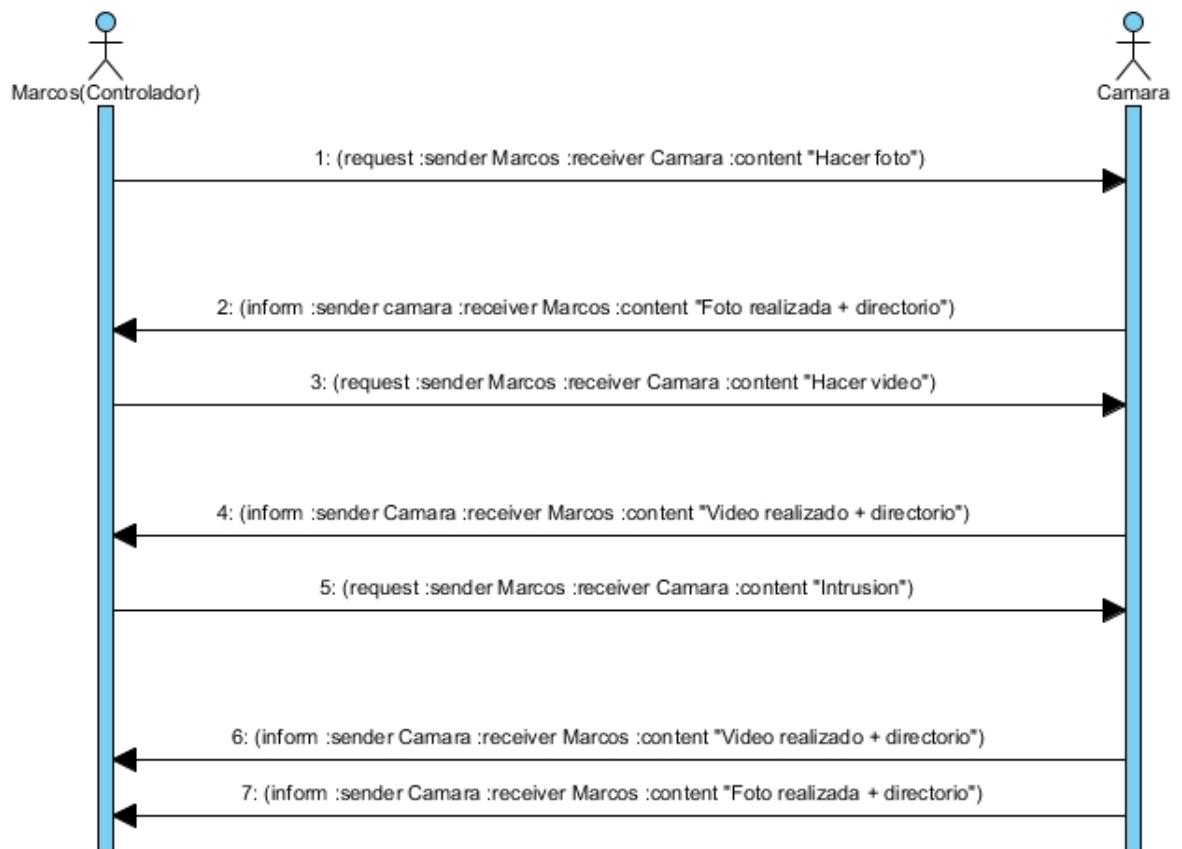
Este diagrama representa la estructura de clases del sistema, como podemos ver cada componente sensor del sistema, utiliza un agente para el

control y funcionalidad de dicho sensor, que a sus vez cada agente hereda de la clase "SingleAgent" de Magentix2, para que cada agente tenga los métodos necesarios para su creación, inicio, y herede las características esenciales de un agente. Desde aquí podemos diferenciar distintos comportamientos de agentes, en el Capítulo 4 se citará específicamente el comportamiento de cada uno de ellos.

3.2. Diagramas

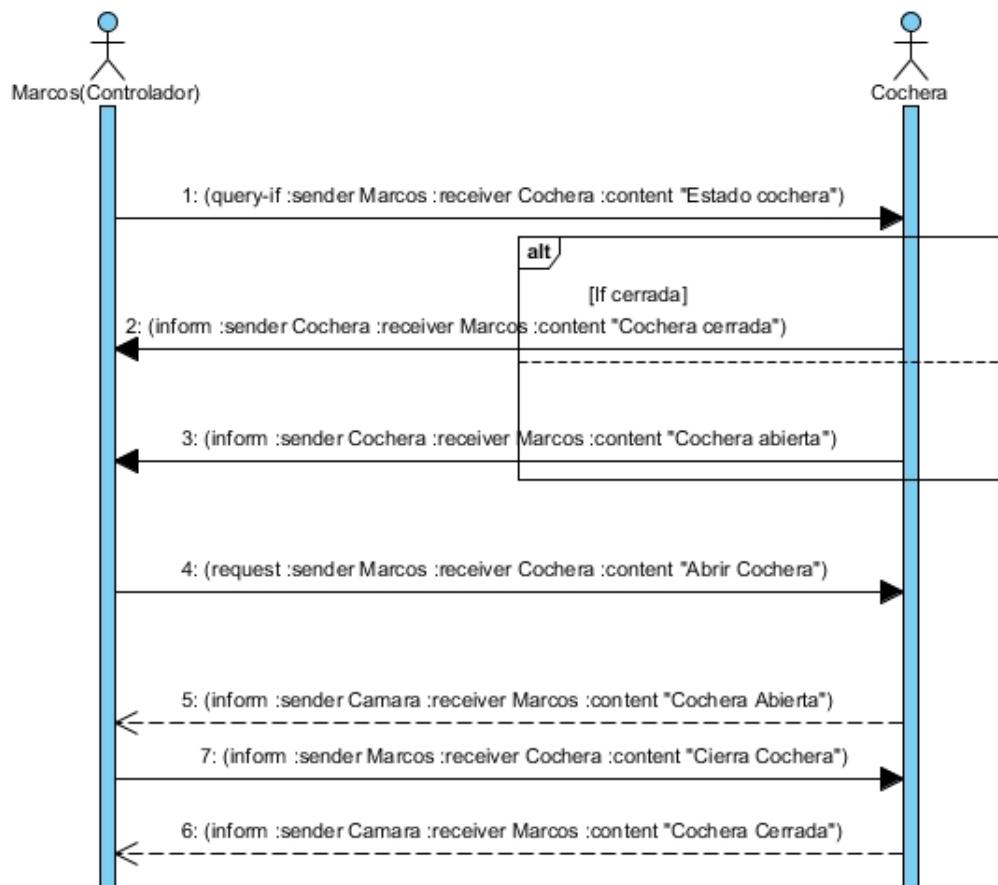
3.2.1. Diagramas de secuencia

Se va a mostrar cual sería el diseño y el comportamiento de cada componente (sensor) individualmente.



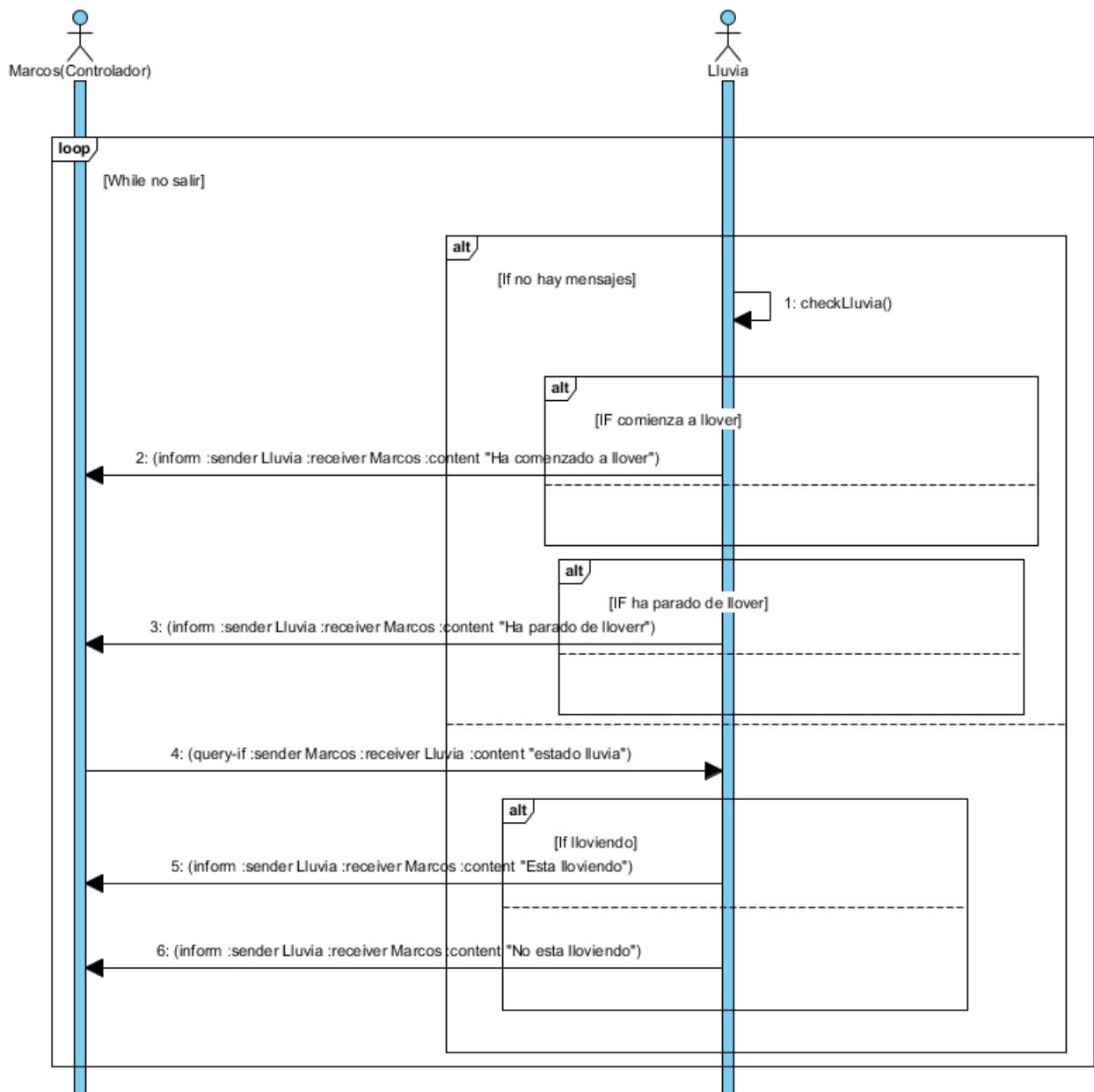
Si el agente recibe un mensaje con una performativa incorrecta o el mensaje no está formulado correctamente, siempre responderá con el mensaje correspondiente.

Figura 3.2: Diagrama de secuencia Módulo Cámara.



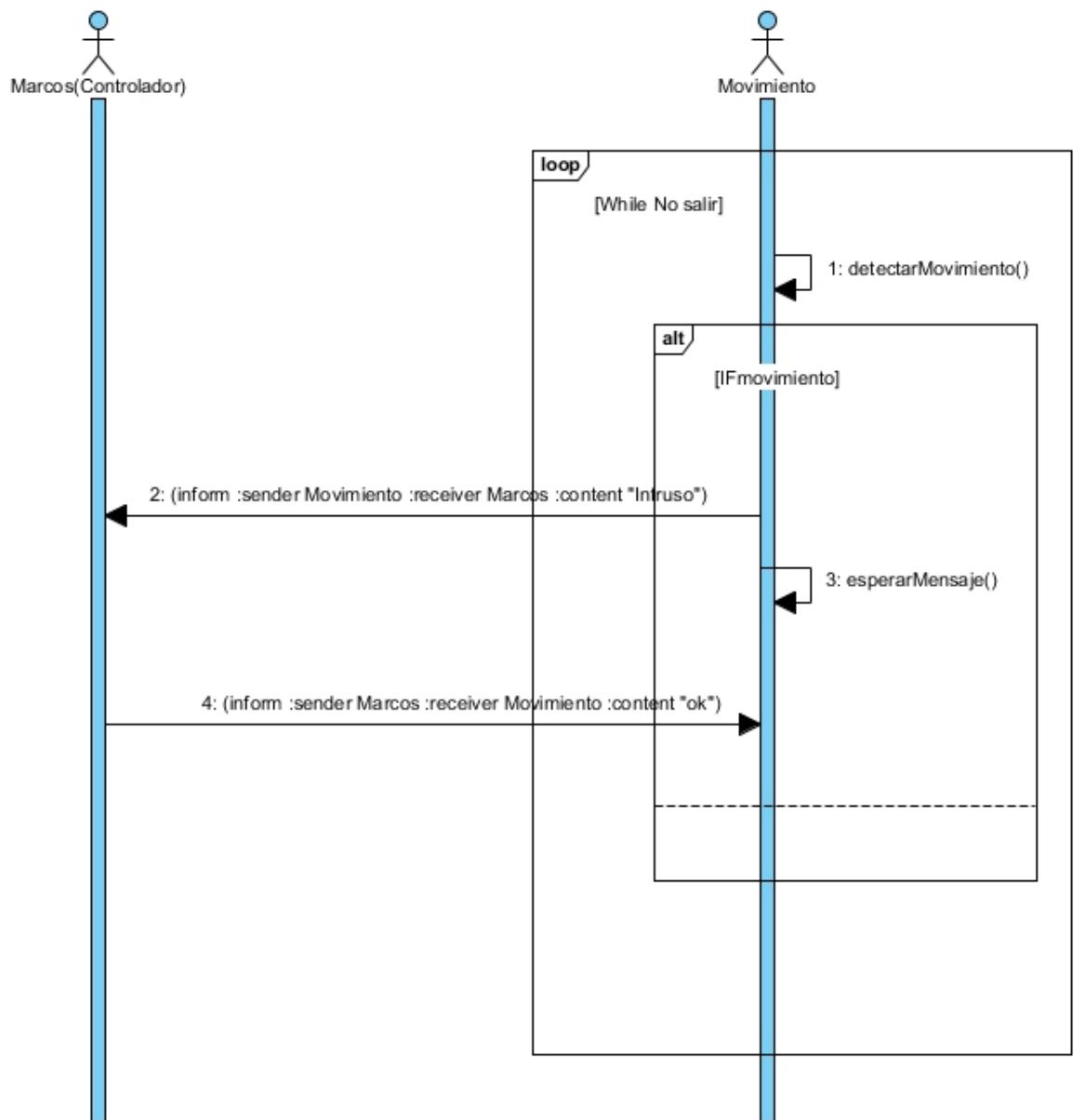
Si el agente recibe un mensaje con una performativa incorrecta o el mensaje no está formulado correctamente, siempre responderá con el mensaje correspondiente.

Figura 3.3: Diagrama de secuencia Módulo L298N Cochera.



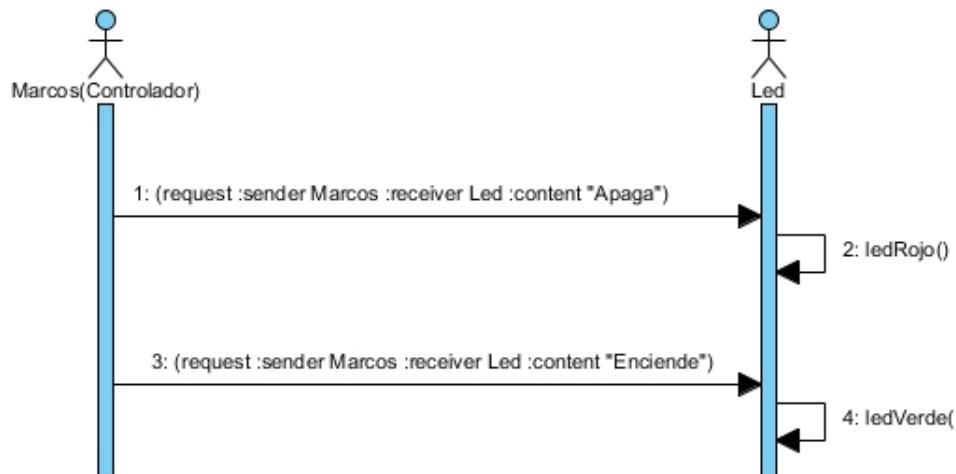
Si el agente recibe un mensaje con una performativa incorrecta o el mensaje no está formulado correctamente, siempre responderá con el mensaje correspondiente.

Figura 3.4: Diagrama de secuencia Módulo Lluvia.



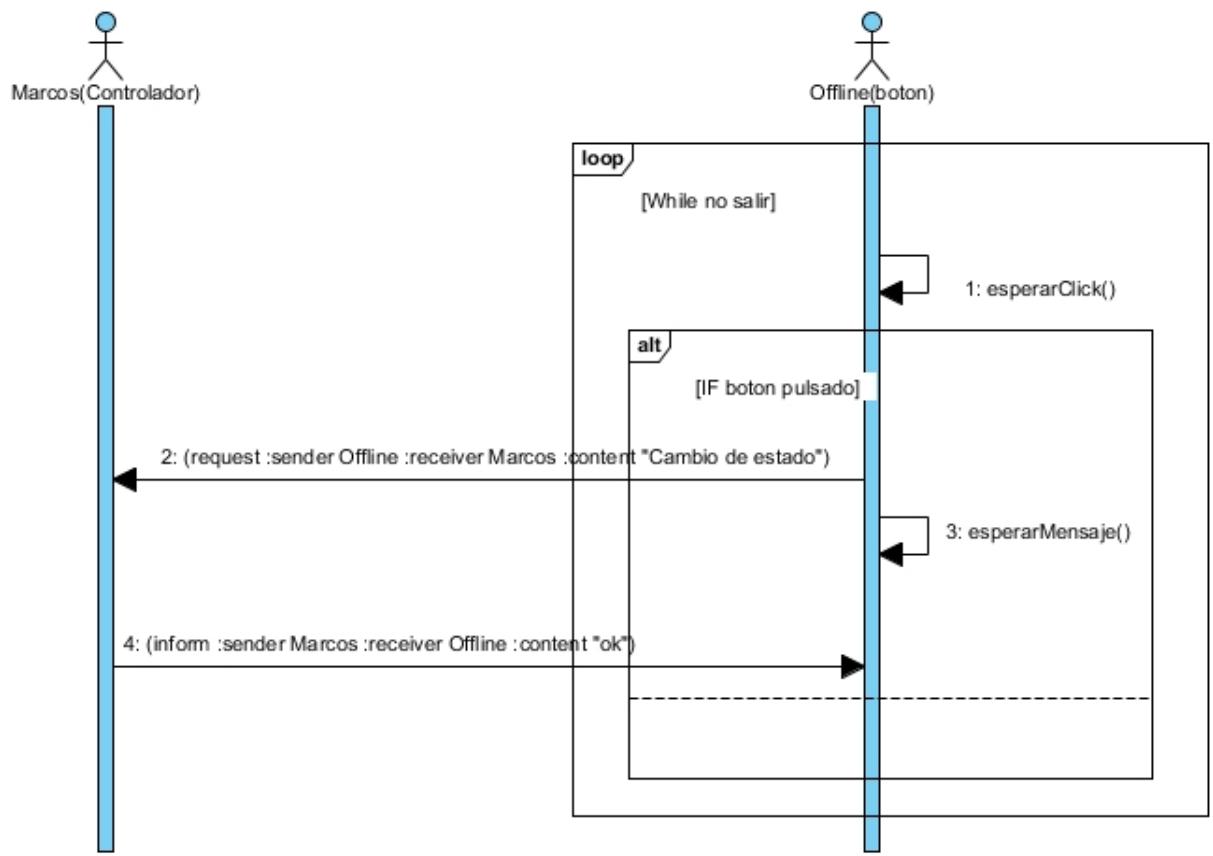
Si el agente recibe un mensaje con una performativa incorrecta o el mensaje no está formulado correctamente, siempre responderá con el mensaje correspondiente.

Figura 3.5: Diagrama de secuencia Módulo Movimiento.



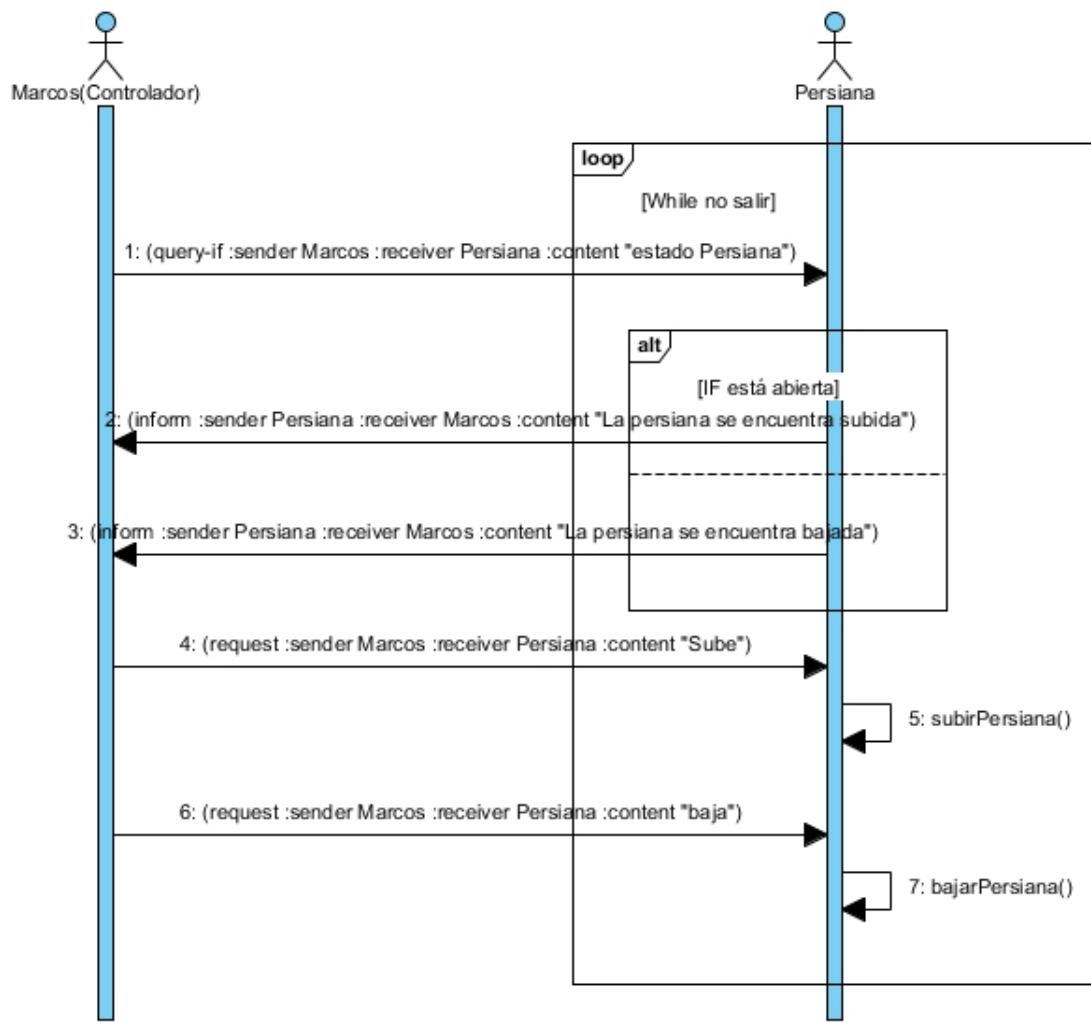
Si el agente recibe un mensaje con una performativa incorrecta o el mensaje no está formulado correctamente, siempre responderá con el mensaje correspondiente.

Figura 3.6: Diagrama de secuencia Módulo Led.



Si el agente recibe un mensaje con una performativa incorrecta o el mensaje no está formulado correctamente, siempre responderá con el mensaje correspondiente.

Figura 3.7: Diagrama de secuencia Módulo Offline.



Si el agente recibe un mensaje con una performativa incorrecta o el mensaje no está formulado correctamente, siempre responderá con el mensaje correspondiente.

Figura 3.8: Diagrama de secuencia Módulo L298N Persiana.

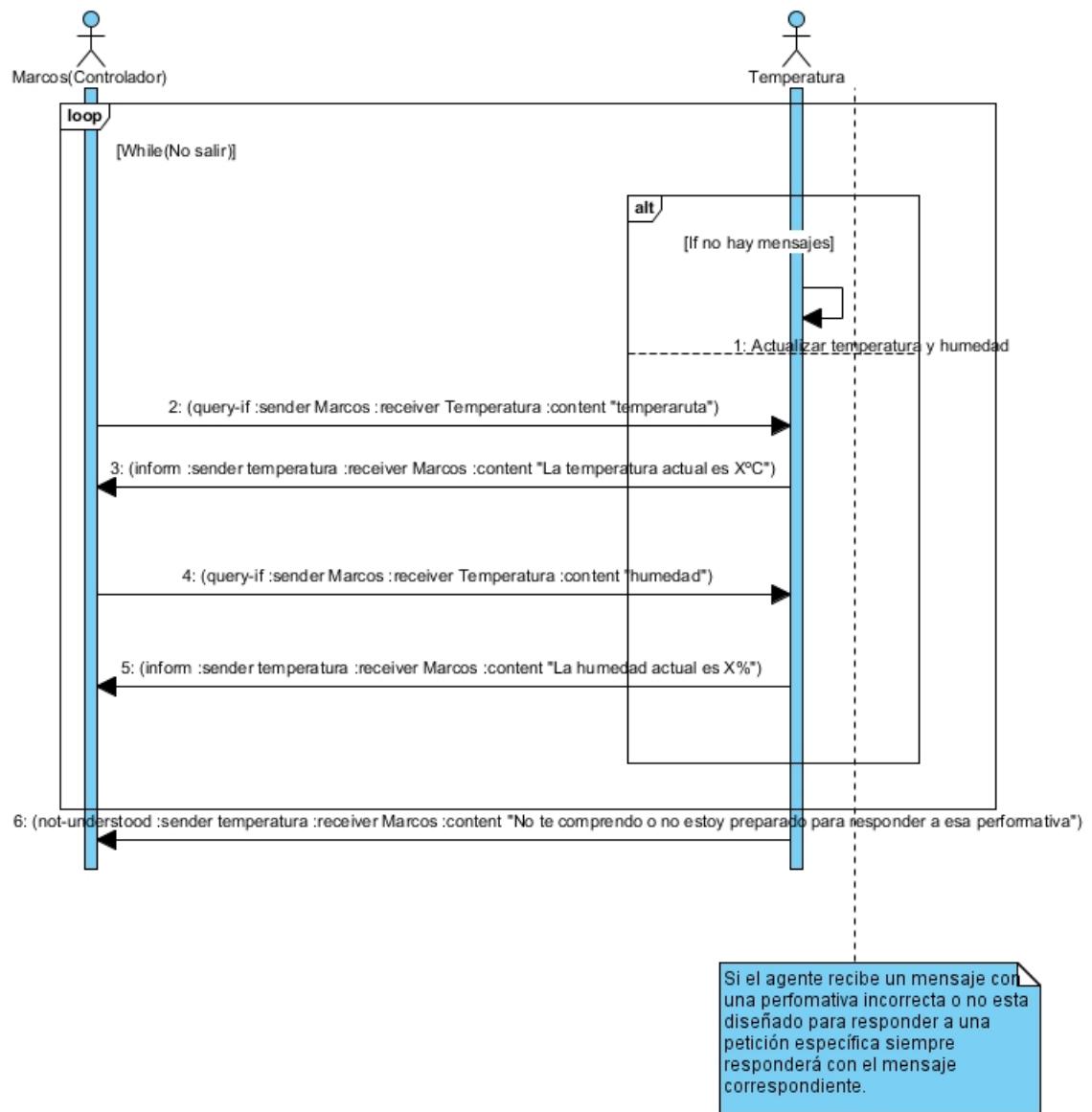


Figura 3.9: Diagrama de secuencia Módulo Temperatura.

Todos estos diagramas anteriores es el comportamiento de cada agente para controlar su sensor correspondiente de forma individual, ahora a continuación se pasa a mostrar los diagramas de secuencia cuando interactúan varios agentes, con el comportamiento **inteligente** ante una situación determinada:

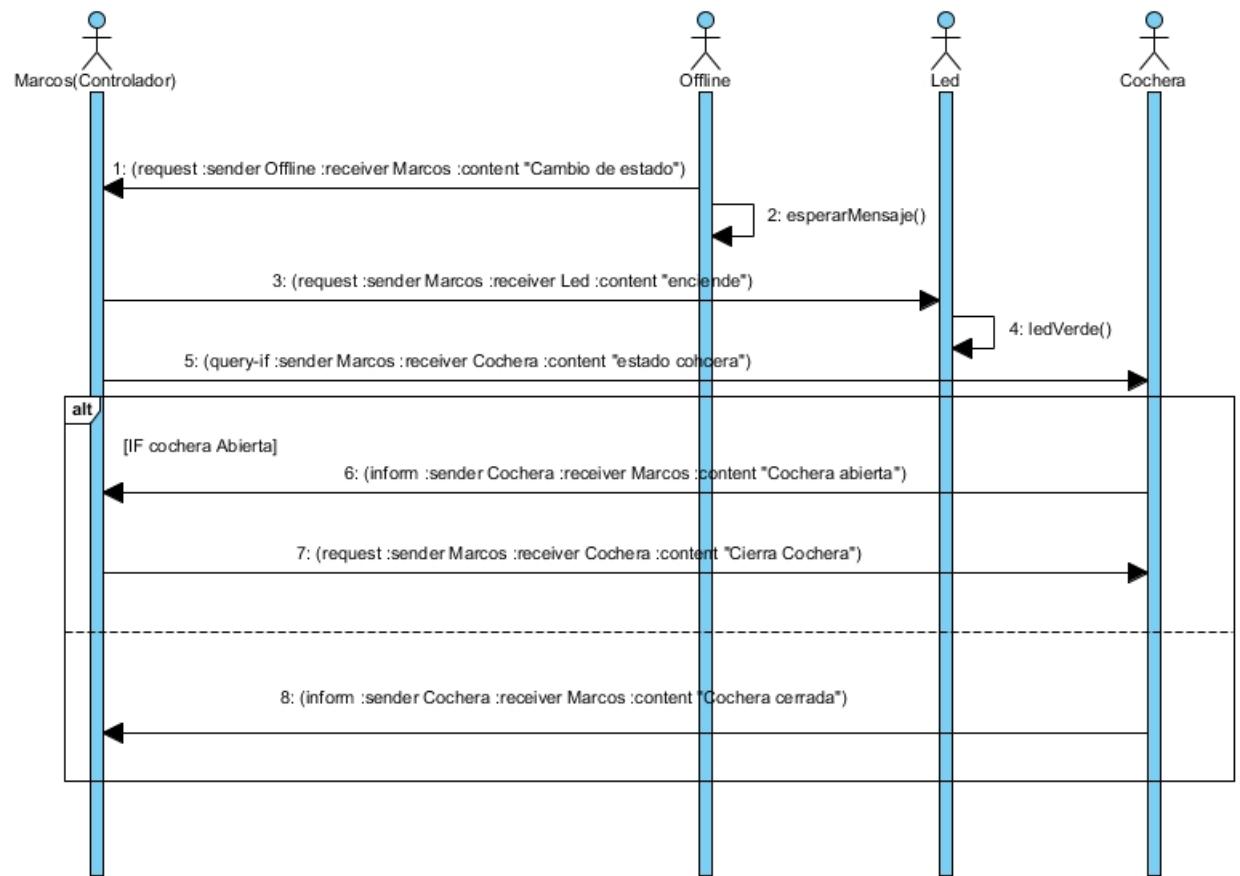


Figura 3.10: Diagrama de secuencia activando el sistema de Seguridad.

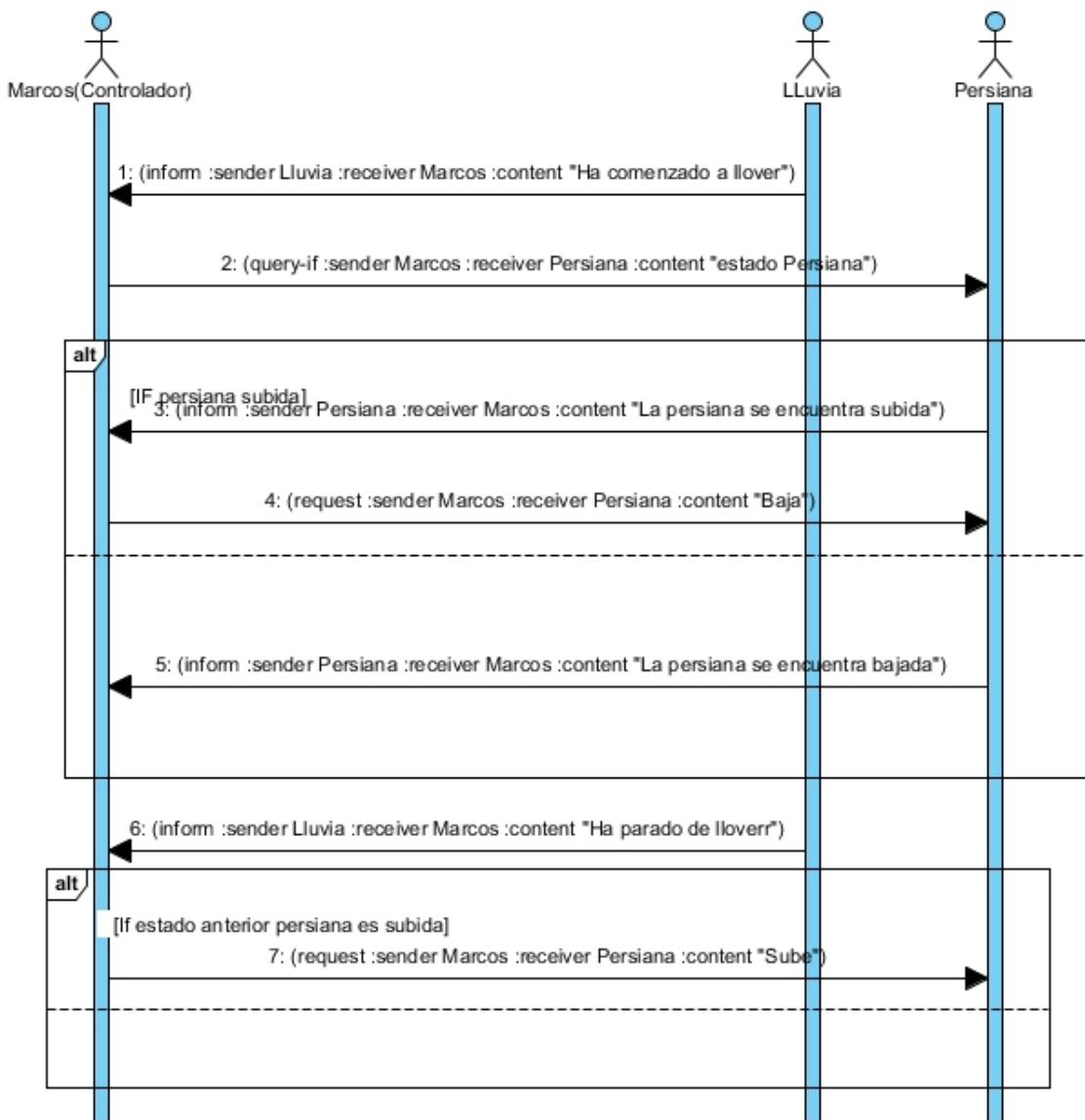


Figura 3.11: Diagrama de secuencia detectando Lluvia.

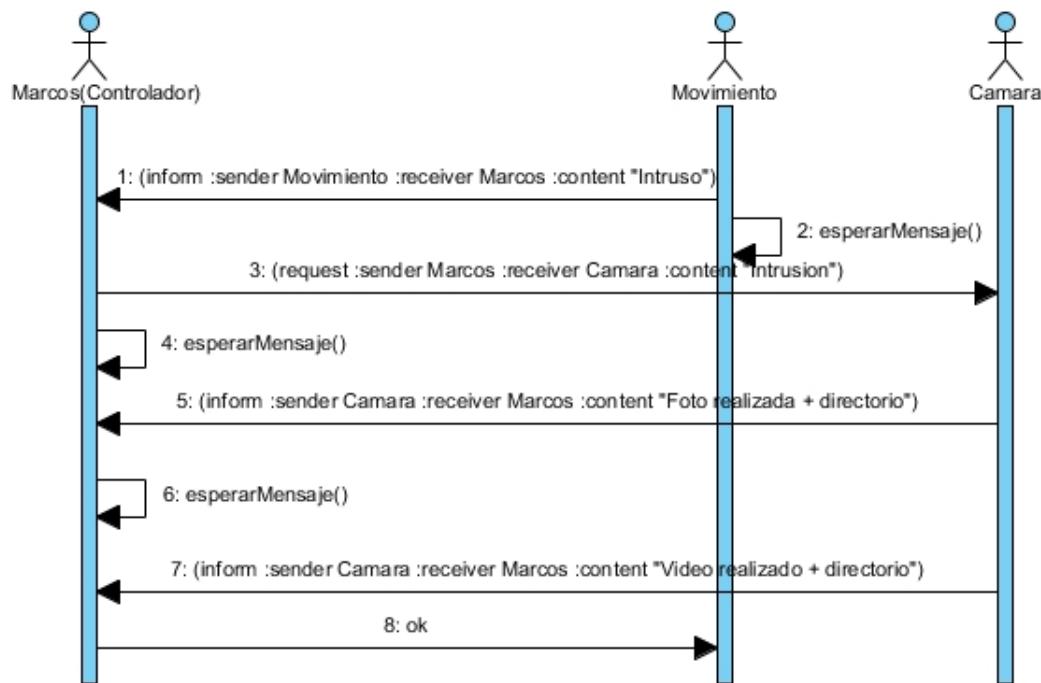


Figura 3.12: Diagrama de secuencia detectando Movimiento.

3.2.2. Diagramas de actividad

En este apartado se muestran los diferentes diagramas de actividad, en los que se muestra los distintos estados que sigue cada agente para llevar a cabo su funcionalidad:

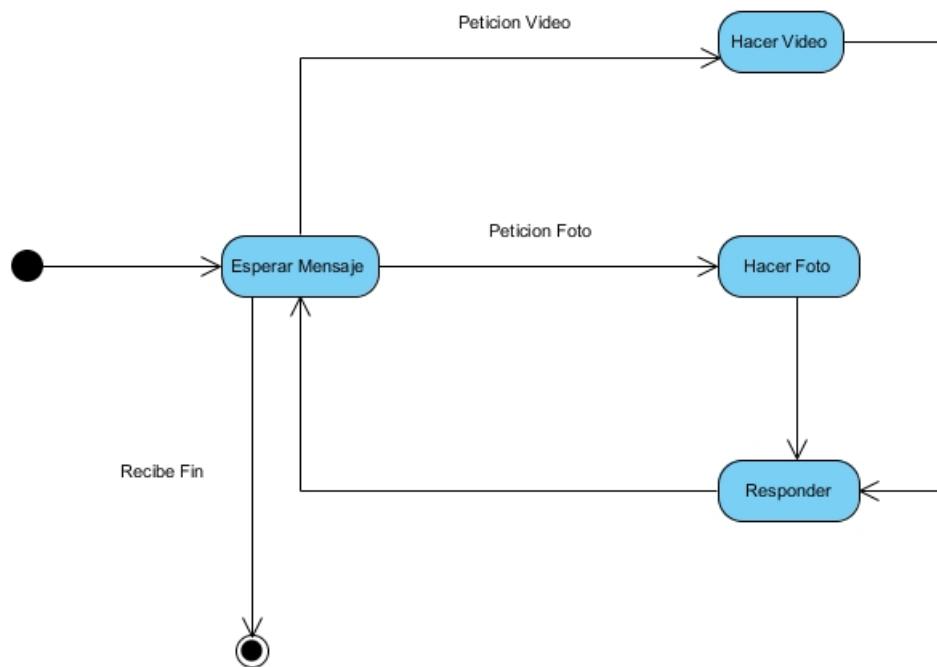


Figura 3.13: Diagrama de actividad del Agente Cámara.

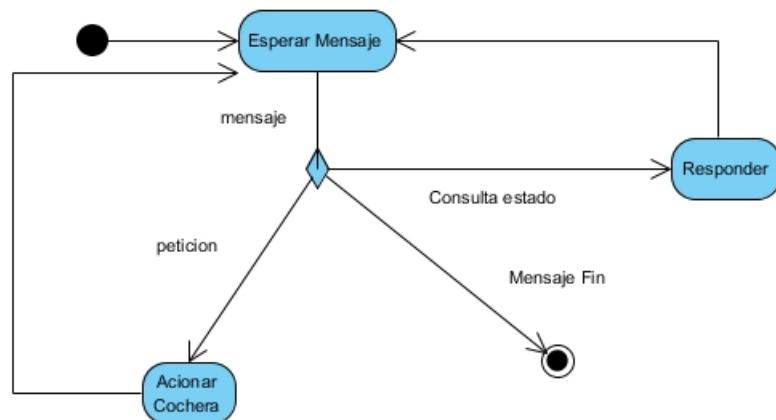


Figura 3.14: Diagrama de actividad del Agente Cochera.

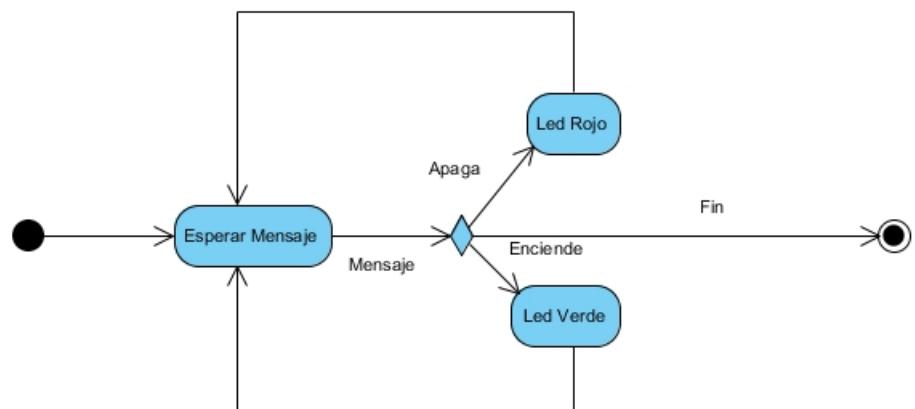


Figura 3.15: Diagrama de actividad del Agente Led.

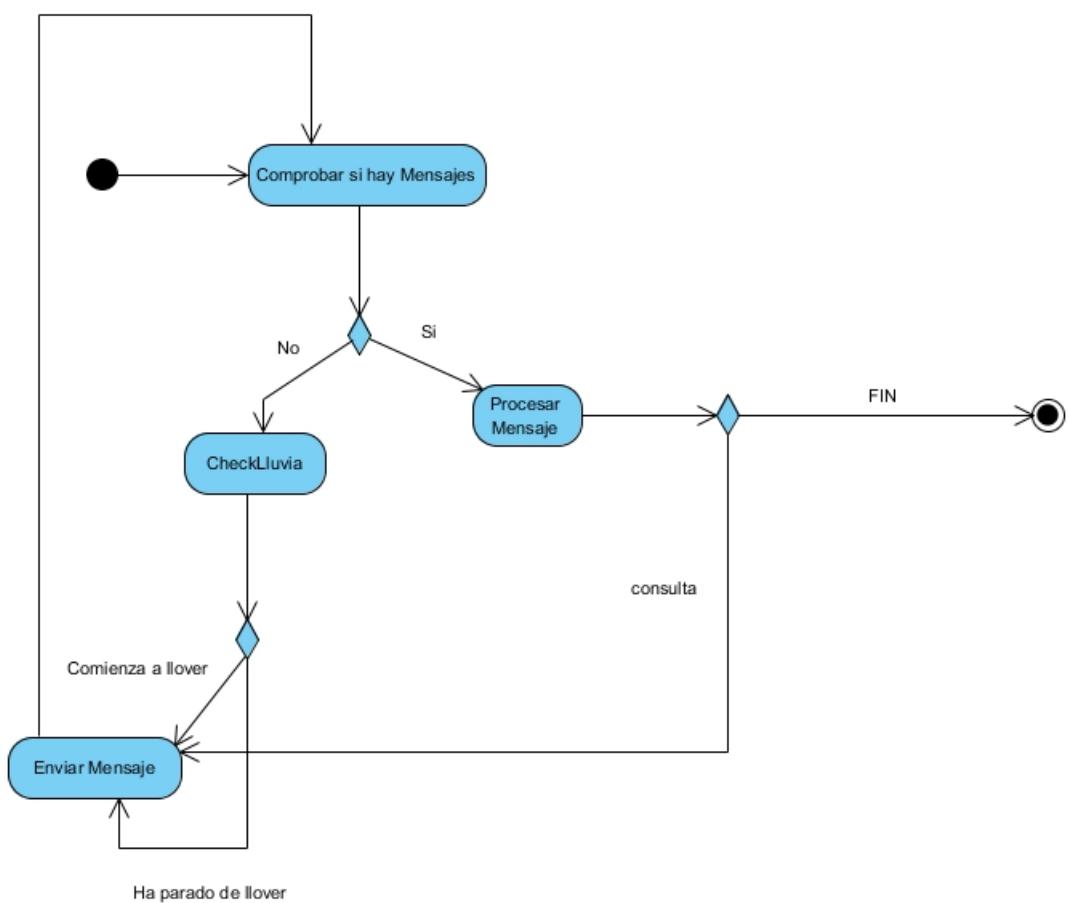


Figura 3.16: Diagrama de actividad del Agente Lluvia.

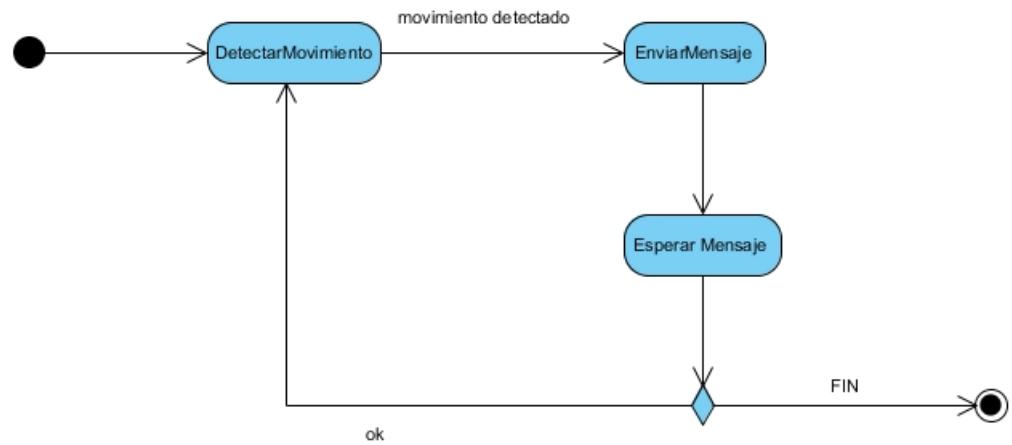


Figura 3.17: Diagrama de actividad del Agente Movimiento.

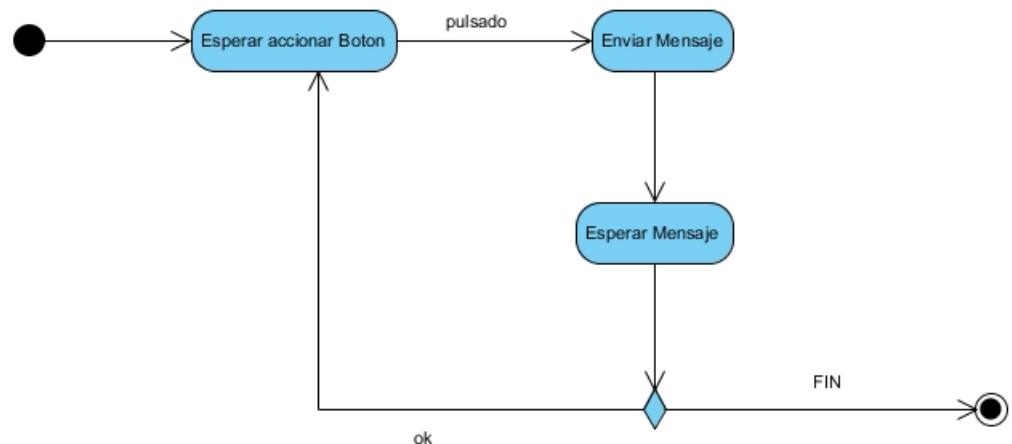


Figura 3.18: Diagrama de actividad del Agente Offline.

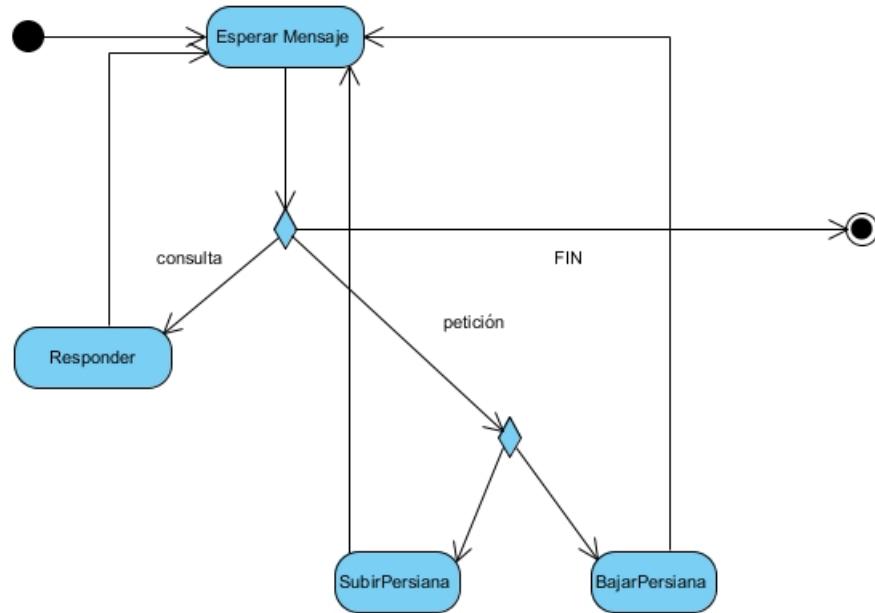


Figura 3.19: Diagrama de actividad del Agente Persiana.

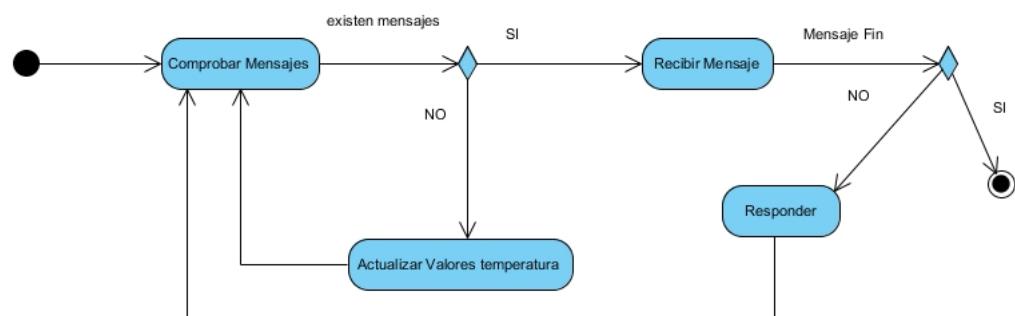


Figura 3.20: Diagrama de actividad del Agente Temperatura.

Capítulo 4

Implementación.

Aunque no es un dato relevante, pero es pertinente mencionar, que para cada agente o mejor dicho para cada sensor, se ha diseñado como un proyecto nuevo (por ejemplo un nuevo proyecto de Netbeans para cada componente), por lo que cada uno contendrá su propio "main" y trabajará independientemente a los demás. Este objetivo esta pensado para no tener que parar la ejecución de todos los agentes para un posible modificación, así como también dotar al sistema de la posible escalabilidad de añadir en cualquier momento un nuevo componente con su correspondiente agente, sin que la jerarquía de agentes se vea afectada.

Comenzando por los agentes, todos los agentes para su programación y funcionamiento es la misma, aunque cada agente tenga un comportamiento distinto, veamos los puntos y funciones necesarias y más importantes en el siguiente apartado.

La documentación oficial de Magentix2, así como toda la información sobre sus clases y métodos concretos, funcionalidades e incluso ejemplos de agentes para llevar a cabo la implementación, la podemos encontrar en [10]. Todo el código que se ha utilizado y se ha desarrollado para este proyecto está disponible actualmente en [6],

4.1. Agentes

Unos de los primeros pasos es poder añadir la biblioteca [42] a este proyecto como ya se mencionó en el apartado 2.4, la biblioteca que corresponde a la plataforma magentix2.

Una vez se tiene disponible la librería se debe de importar la clases necesarias, como podemos ver en la Figura 4.1, si utilizamos una herramienta de programación como por ejemplo en este caso, "Netbeans", obtendremos

también una pequeña ayuda a la hora de importar las clases necesarias que se encuentran dentro de esta librería.

Unos de los primeros puntos que se debe tener en cuenta antes de lanzar uno de estos agentes, es que se tiene que realizar una conexión con la plataforma Magentix2, es decir, como se vio en el apartado 2.3, se tiene que tener levantada la plataforma para poder crear ahí cualquier agente. Para realizar esta conexión es necesario utilizar la sentencia de la línea 23 de la Figura 4.1, que es la encargada de realizar dicha conexión, tenemos que tener en cuenta dónde hemos levantado la plataforma Magentix2, si es en la misma máquina "localhost", o si en cambio la tenemos en otro ordenador, debemos de especificar su dirección IP así como su el puerto utilizado.

```

8 import es.upv.dsic.gti_ia.core.AgentID;
9 import es.upv.dsic.gti_ia.core.AgentsConnection;
10
11 /**
12 *
13 * @author pi
14 */
15 public class Led {
16
17     /**
18      * @param args the command line arguments
19      */
20     public static void main(String[] args) {
21         // TODO code application logic here
22         Agente a;
23         AgentsConnection.connect("localhost", 5672, "test", "guest", "guest", false);
24         try {
25             a = new Agente(new AgentID("Led"));
26             a.start();
27         } catch (Exception ex) {
28             System.out.println("Error al crear el agente");
29         }
30     }
31 }
32
33 }
```

Figura 4.1: Ejemplo de creación y lanzamiento de un agente.

Si todo ha ido bien, ya podemos instanciar nuestro agente, identificándolo por un nombre que será único, ya que si intentamos crear o lanzar varios agentes con el mismo nombre tendremos un error y no será posible ejecutar nuestro agente. Como vemos en las sentencias de las líneas 25 y 26 de la Figura 4.1.

Para que un agente sea correctamente ejecutado, necesita dotarse de algunos métodos, pero el más importante el método sobrescrito "execute()", que podemos decir que es el método motor del agente, una vez es creado

e iniciado, su comportamiento será definido en dicho método. Voy a seguir con un ejemplo sencillo como podemos ver en la Figura 4.2.

```

24  /**
25  *
26  * @author marcos
27  */
28 public class Agente extends SingleAgent {
29
30     private final GpioController gpio;
31     private GpioPinDigitalOutput Pin1;
32     private GpioPinDigitalOutput Pin2;
33     private boolean salir;
34     private boolean led;
35
36     public Agente(AgentID aid) throws Exception{
37         super(aid);
38         gpio = GpioFactory.getInstance();
39         Pin1 = gpio.provisionDigitalOutputPin (RaspiPin.GPIO_22);
40         Pin1.setShutdownOptions(false, PinState.LOW);
41         Pin2 = gpio.provisionDigitalOutputPin (RaspiPin.GPIO_23);
42         Pin2.setShutdownOptions(false, PinState.LOW);
43         salir=false;
44         led=false;
45     }
46
47     @Override
48     public void execute(){
49         Pin2.low();
50         Pin1.high();
51         while(!salir){
52             try {
53                 this.recibirMensaje();
54             } catch (InterruptedException ex) {
55                 Logger.getLogger(Agente.class.getName()).log(Level.SEVERE, null, ex);
56             }
57         }
58     }
59 }
60

```

Figura 4.2: Ejemplo de una clase Agente.

En esta figura anterior 4.2 lo más destacado y recordando la necesidad de heredar de la clase **SingleAgent**, y aquí se puede ver como se ha implementado un simple método ”execute()” para que el agente cuando se inicie, ejecute ese comportamiento, una vez acaba su comportamiento de este método, el agente finaliza y termina su ejecución (desaparece de la plataforma Magentix2).

La esencia de estos agentes es la comunicación entre otros agentes, aunque ya con esta base, a un agente lo podemos caracterizar con el comportamiento que deseemos. Para llevar a cabo una comunicación, lo más importante aparte del emisor y el receptor, es el mensaje. Para ello hablamos de la

clase **ACLMensaje**. Si se instancia un objeto de esta clase se está creando un mensaje compatible con nuestro sistema de agentes, para mayor claridad nos fijamos en la Figura 4.3.

```

61 ▼   private void enviarMensaje(String receptor, String contenido, int performativa){
62     ACLMessage outbox= new ACLMessage();
63     outbox.setSender(this.getAid());
64     outbox.setReceiver(new AgentID(receptor));
65     outbox.setContent(contenido);
66     outbox.setPerformativa(performativa);
67     this.send(outbox);
68     System.out.println("\nEnvio mensaje: "+contenido+ " a: "+receptor);
69   }
70
71 /*Recibir cualquier mensaje y se filtra según el tipo de mensaje recibido*/
72 ▼   private void recibirMensaje() throws InterruptedException{
73     ACLMessage mensaje = new ACLMessage();
74     mensaje=this.receiveACLMessage();
75     System.out.println("Recibo un mensaje de "+mensaje.getSender()+" : "+mensaje.getContent());
76     String emisor=this.procesarSender(mensaje.getSender());
77     String performativa=mensaje.getPerformativa();

```

Figura 4.3: Interacción con mensajes, objeto de la clase **ACLMensaje**.

Aquí en esta Figura 4.3, se ve un ejemplo sencillo para enviar y recibir mensajes, lo primero que necesitamos es crear un objeto Mensaje como se puede ver en la línea 62 o 73 de esta figura. Vemos como podemos se puede enriquecer este objeto, como añadiendo el emisor, receptor, contenido del mensaje, performativa que vamos a utilizar... y una vez preparado el mensaje ya sea para ser enviado o recibir un mensaje utilizamos los siguientes métodos:

- **receiveACLMensaje()**, para recibir un mensaje, recordemos que este sentencia es bloqueante, es decir, hasta que el agente no reciba un mensaje no puede continuar con su ejecución.
- **send(Mensaje)**, para enviar un mensaje, una vez tenemos preparado el mensaje, es necesario indicar obligatoriamente al menos el receptor del mensaje. Esta operación no es bloqueante.

Ya visto una base bastante simple pero necesaria para la ejecución de un agente de Magentix2, se procede a detallar a grandes rasgos el comportamiento más destacado de cada agente.

4.1.1. Cámara

Este agente está diseñado para estar pendiente de recibir un mensaje por parte del controlador (Marcos), para realizar una captura con la cámara o generar un vídeo en un instante determinado.

Para el uso de la cámara se ha tenido que recurrir al lenguaje de programación **Python**, ya que en un primer momento, se realizó el control de la cámara mediante la librería ”JRPiCam” que podemos descargarla en [14] y podemos ver toda su documentación en [13]. Esta librería es usada para el lenguaje de programación JAVA, que como sabemos, es un lenguaje de programación interpretado y trabaja mediante la máquina virtual de JAVA, debido a que las Raspberry Pi cuenta con un avanzado microprocesador pero posiblemente no lo suficiente potente para realizar capturas de imágenes en un tiempo determinado. También puede ser posible por la acumulación de los diferentes procesos del sistema y otras herramientas de la Raspberry Pi, lo que producía un retardo en la generación de una fotografía.

Es decir, el objetivo de este componente en este proyecto es asegurar el sistema de seguridad o vigilancia de la vivienda, por lo que es un requisito no fundamental pero primordial que tenga un tiempo de respuesta bastante corto. Esta situación producía un problema, ya que al detectar una intrusión mediante el sensor de movimiento transcurría un retardo de unos 6 segundos aproximadamente desde la detección hasta la generación de la fotografía (el posible intruso no podría ser captado con este tiempo de retardo).

El problema, provenía de la generación de la imagen, ya que la comunicación entre los agentes y la detección de movimiento era totalmente instantáneo. Realizando varias pruebas y buscando la solución más óptima, un script en Python satisfizo este requisito.

```

1 import picamera
2 import time
3
4 fecha = time.strftime("%c")
5
6 camara = picamera.PiCamera()
7
8 ruta = "/home/pi/Desktop/Fotos/" +fecha+".jpeg"
9 camara.capture(ruta)
10
11 camara.close()
12 print(ruta)
```

Figura 4.4: Script en Python para generar una imagen con la cámara Pi.

Utilizando la librería ”picamera” en Python [20], y con unas simples sentencias como podemos ver en la Figura 4.4, puedo obtener una fotografía en el directorio deseado, con la fecha y hora actual en la que se ha generado, y lo más importante en un tiempo prácticamente instantáneo.

Observando estos resultados, se dota al agente correspondiente, del control de la cámara, que desde JAVA, mediante el método que podemos ver en la Figura 4.5, ejecuta este script para la generación de fotografías lanzando

un nuevo proceso, concretamente en la línea 115.

El resto de sentencias existentes en este método corresponden a lo siguiente, cuando se ejecuta el script se genera un nuevo archivo de tipo imagen ("JPG"), con una ruta y un nombre de fichero, por lo que el agente necesita conocer esta información para saber dónde se localiza dicha imagen y que nombre se le ha asignado. Para solventar esto, solo se necesita imprimir por pantalla dicho directorio cuando se está ejecutando el Script de Python y así es posible capturar este directorio desde Java mediante la lectura de un buffer (BufferedReader).

```
113     private String hacerFoto() throws IOException{
114         String line;
115         Process p= Runtime.getRuntime().exec("/usr/bin/python2.7 /home/pi/Desktop/foto.py");
116         BufferedReader bri = new BufferedReader(new InputStreamReader(p.getInputStream()));
117         if((line = bri.readLine()) != null){
118             if(!line.contains("ERR_CRC") || line.contains("ERR_RNG")){
119                 }
120             else{
121                 System.out.println("Data Error");
122             }
123         }
124     }
125     bri.close();
126     return line;
127 }
```

Figura 4.5: Código para ejecución de un Script en Python desde Java.

Por último, para realizar un vídeo con la cámara se sigue el mismo proceso, tan sólo cambiando unas sentencias en el Script de Python como podemos ver en la Figura 4.6.

```

1  import picamera
2  import time
3  import subprocess
4
5  fecha = time.strftime("%c")
6
7  camara = picamera.PiCamera()
8
9  aux = "/home/pi/Desktop/Videos/" + fecha + ".h264"
10 aux = aux.replace(' ', '_')
11 aux = aux.replace(':', '.')
12 ruta = aux.replace('h264', 'mp4')
13 camara.start_recording(aux)
14 camara.wait_recording(8)
15 camara.stop_recording()
16
17 camara.close()
18
19 convertir = 'MP4Box -add '+aux+' '+ruta
20 error = subprocess.Popen(convertir, shell = True)
21 error.wait()
22 import os
23 os.remove(aux)
24 print(ruta)

```

Figura 4.6: Script en Python para generar una vídeo con la cámara Pi.

4.1.2. Cochera

Todos los agentes van a seguir la misma estructura, se encuentran en la espera de recibir un mensaje para realizar una acción determinada, que estas acciones si son distintas para cada uno de ellos dependiendo del sensor que estén controlando.

Para este caso, se menciona la cochera, el módulo L298N del apartado 2.2, por lo que este agente tiene la capacidad de actuar sobre pines GPIO que se mencionó en la sección 2.1.2.

Aprovechando este agente se detalla la librería y las operaciones más importantes para poder usar estos pines GPIO, ya que varios agentes de este proyecto operan sobre GPIO distintos, pero el objetivo es el mismo, y evitar así duplicar información en cada uno de ellos.

En primer lugar necesitamos la librería "pi4j-core.jar" de código abierto y para una plataforma JAVA, que podemos descargar en [34], y encontrar toda la documentación oficial en [32].

Siempre que necesitamos actuar o consultar un GPIO concreto, lo primero que tenemos que hacer es instanciar un controlador como podemos ver en la Figura 4.7, concretamente la línea 30 y 38. Una vez ya tenemos nues-

tro controlador de GPIO ahora tenemos que indicar qué GPIO exactamente vamos a utilizar y qué modo va adquirir dicho pin, ya sea entrada o salida:

```

24  /**
25  *
26  * @author marcos
27  */
28 public class Agente extends SingleAgent {
29
30     private final GpioController gpio;
31     private GpioPinDigitalOutput Pin1;
32     private GpioPinDigitalOutput Pin2;
33     private boolean salir;
34     private boolean led;
35
36     public Agente(AgentID aid) throws Exception{
37         super(aid);
38         gpio = GpioFactory.getInstance();
39         Pin1 = gpio.provisionDigitalOutputPin (RaspiPin.GPIO_22);
40         Pin1.setShutdownOptions(false, PinState.LOW);
41         Pin2 = gpio.provisionDigitalOutputPin (RaspiPin.GPIO_23);
42         Pin2.setShutdownOptions(false, PinState.LOW);
43         salir=false;
44         led=false;
45     }
46

```

Figura 4.7: Código para la preparación de un GPIO.

- si necesitamos un GPIO de salida para alterar el estado de dicho pin en un determinado momento, necesitamos fijarnos por ejemplo, en la línea 31 y 39, donde declaramos exactamente el GPIO número 22 de modo salida. Recordemos la numeración GPIO de la Raspberry Pi en la Figura 2.3.
- En cambio, si deseamos que dicho pin anterior obtenga la capacidad de entrada, para poder obtener información por el pin GPIO, solo necesitamos cambiar la característica "Output" por "Input".

Después de este proceso ya tenemos disponible un pin GPIO para recibir o enviar una señal, que recordar que una señal alta (high) es un "1" en señal digital, y baja(low) un "0". También podemos cambiar el modo de este pin en un momento determinado del programa, pero sólo es posible obtener un solo modo simultáneamente, es decir, podemos ir cambiando la propiedad de un GPIO entre entrada o salida en cualquier momento, pero no podemos dotar a un GPIO con ambos modos a la vez.

Ahora las siguientes operaciones más importantes:

- Para cambiar el estado:

Por ejemplo, Pin.low(), emite una señal digital "0".

Pin.high(), emite una señal digital "1".

- Para consultar el estado:

Por ejemplo, Pin.isLow(), se comprueba que su estado es "0", devuelve true si es cierto, falso en caso contrario.

También podemos utilizar Pin.isHigh(), que comprueba si estado es "1", devuelve true si es cierto, falso en caso contrario.

También existen los métodos getState() o isState() que una funcionalidad similar.

Volviendo a la implementación de este agente, tanto sólo tiene la especial característica de actuar sobre la cochera, que recordamos que es un motor DC, por lo que puede ser accionado en una dirección (derecha o izquierda) mediante el módulo L298N, es decir, mantener subiendo o bajando la cochera del garaje por un determinado tiempo.

Para ello existe un método llamado "pulse", que tenemos que especificar por parámetros, el tiempo en milisegundos que deseamos actuar, y el estado que desamos (true o false, "1" o "0"), como podemos ver en la Figura 4.8.

```
Pin1.pulse(10,true);
```

Figura 4.8: Método pulse() para accionar un GPIO.

4.1.3. Led

Este agente tiene la misma funcionalidad que el agente anterior, esperar recibir un mensaje para cambiar el color del Led, para ello utiliza dos pines GPIO, en el que alterna señales bajas y altas en los GPIO correspondientes configurados de modo salida, y conseguir el color ROJO o VERDE del Led.

4.1.4. Lluvia

Se sigue con la misma estructura, funcionalidad y objetivo de los agentes, aunque este agente si tiene una capacidad distinta, este agente no es bloqueante, es decir, es capaz de recibir mensajes a la misma vez que esta realizando otra tarea. Podemos decir que se genera una nueva "hebra" del agente que solo está pendiente de recibir los mensajes y almacenarlos en una cola, sin afectar al rendimiento y la eficiencia del agente principal.

Para ello necesitamos declarar una cola de mensajes, como podemos ver en la Figura 4.9, y a través de la sobrecarga del método ”onMessage()” aplicamos esta capacidad del agente en recibir un mensaje sin que sea bloqueado.

```

24 ▼ /**
25 *
26 * @author marcos
27 */
28 ▼ public class Agente extends SingleAgent {
29
30     private boolean salir;
31     private boolean lluvia;
32     private boolean cambio;
33     private Queue<ACLMensaje> cola;
34     private final GpioController gpio;
35     private GpioPinDigitalInput Pin;
36
37
38 ▼     public Agente(AgentID aid) throws Exception{
39         super(aid);
40         salir=false;
41         lluvia=false;
42         cola=new LinkedList<ACLMensaje>();
43         gpio = GpioFactory.getInstance();
44         Pin = gpio.provisionDigitalInputPin (RaspiPin.GPIO_00);
45         Pin.setShutdownOptions(true);
46
47     }
48
49     @Override
50     public void onMessage(ACLMensaje msg){
51         cola.add(msg);
52     }

```

Figura 4.9: Agente no bloqueante para recibir mensajes.

Esto es debido a una situación, este sensor o mejor dicho agente, está capacitado para comprobar en un determinado momento si se produce un diluvio, pero si su esfuerzo está en ir comprobando el estado del sensor continuamente para validar el caso de lluvia, no puede recibir en ningún momento un mensaje de otro agente.

Gracias a esto anterior, se ha podido capacitar a este agente que mientras está comprobando el estado del sensor cada 3 segundos Figura 4.10, en cualquier momento puede recibir un mensaje de cualquier otro agente del sistema.

```

131 ▼    private void checkLluvia(){
132        if(gpio.isLow(Pin)) lluvia=true;
133        else lluvia=false;
134    }

```

Figura 4.10: Comprobación de Lluvia.

Para llevar a cabo este objetivo es sencillo, incorporando al comportamiento del agente una sección que se ejecuta si no hay mensajes pendientes, en este caso cada 3 segundos, comprueba el estado del sensor como podemos ver en la Figura 4.11, aprovechando el procesamiento en "una segunda hebra" en la que, si en un determinado momento recibe el agente un mensaje, es capaz de encollarlo en su espacio de memoria reservado para almacenar estos mensajes, y posteriormente poder analizarlos y atender las peticiones.

```

54     @Override
55     public void execute(){
56         while(!salir){
57             while(cola.isEmpty()){
58                 this.checkLluvia();
59                 if(lluvia && !cambio){
60                     System.out.println("Llueve");
61                     //Informar al controlador que esta lloviendo
62                     this.enviarMensaje("Marcos", "Comienza a llover", ACLMessage.INFORM);
63                     cambio=true;
64                 }
65                 if(!lluvia && cambio){
66                     this.enviarMensaje("Marcos", "Ya no llueve", ACLMessage.INFORM);
67                     cambio=false;
68                 }
69             }
70             try {
71                 Thread.sleep(3000);
72             } catch (InterruptedException ex) {
73                 Logger.getLogger(Agente.class.getName()).log(Level.SEVERE, null, ex);
74             }
75         }
76         try {
77             recibirMensaje(cola.poll());
78         } catch (InterruptedException ex) {
79             Logger.getLogger(Agente.class.getName()).log(Level.SEVERE, null, ex);
80         } catch (IOException ex) {
81             Logger.getLogger(Agente.class.getName()).log(Level.SEVERE, null, ex);
82         }
83     }
84 }

```

Figura 4.11: Comportamiento y capacidad del agente Lluvia.

4.1.5. Offline

Este agente es el encargado de controlar el pulsador KY-004, cuyo fin es activar o desactivar el sistema de seguridad manualmente, como se ha visto anteriormente tanto en el diseño como en la especificación de este módulo. Por lo que el objetivo es el mismo, ya presentado, definir un pin GPIO de

entrada para consultar el estado del botón, en este caso, detectar cuando es pulsado.

Para este objetivo se aplica un nuevo sistema, también disponible en esta librería Pi4J en la que se está trabajando, que se trata de un escuchador (listener), es decir, un manejador de eventos nativo de JAVA transladado a los eventos que pueden ocurrir en un pin GPIO, estos eventos son traducidos en posibles cambios de estado en dicho pin digital. La implementación sería como en la Figura 4.12.

```

98  private void ActivarSensor() throws InterruptedException{
99      // create and register gpio pin listener
100     Pin.addListener(new GpioPinListenerDigital() {
101         @Override
102         public void handleGpioPinDigitalStateChangeEvent(GpioPinDigitalStateChangeEvent event) {
103             // display pin state on console
104             System.out.println("Se ha presionado el botón");
105             if(gpio.isLow(Pin) && disponible){
106                 disponible=false;
107                 enviarMensaje("Marcos", "Cambio Offline", ACLMessage.REQUEST);
108                 try {
109                     recibirMensaje();
110                 } catch (InterruptedException ex) {
111                     Logger.getLogger(Agente.class.getName()).log(Level.SEVERE, null, ex);
112                 }
113             }
114         }
115     });
116 }
117 while(true){
118     Thread.sleep(500);
119 }
120 }
```

Figura 4.12: Manejador de eventos digitales en un pin GPIO.

Este proceso se mantendrá esperando un acción sobre el pulsador, es decir, cuando sea pulsado dicho botón, detectará el evento correspondiente y se ejecutará esta función.

Hay algo importante que mencionar aquí, porque este "listener" es capaz de detectar cualquier acción sobre el pulsador, es decir, por ejemplo al pulsar el botón y al soltarlo son dos eventos distintos. También a la hora de realizar las correspondientes baterías de pruebas, se detecta una posible saturación, que se producía cuando se pulsaba varias veces el botón demasiado rápido (varias veces en menos de medio segundo).

El objetivo de este componente es activar o desactivar el sistema de seguridad, como podemos ver en el correspondiente diagrama de secuencia, se detecta la pulsación del botón, y el controlador de los agentes (Jefe) realiza la comunicación pertinente para llevar a cabo este proceso, y aquí es

donde ocurrían los problemas mencionados anteriormente, es decir, se podía detectar dos posibles eventos distintos mientras que aún no se ha llevado a cabo el primer evento detectado, por ello en la línea 109 de la Figura 4.12, soluciona este problema, el agente no detecta el siguiente evento hasta que se resuelve la primera acción, esperando un mensaje por parte del controlador (Jefe) que confirma que ya se ha completado todo el proceso correspondiente y queda disponible para una nueva acción.

4.1.6. Persiana

Este agente si contiene las mismas capacidades y características que el agente Cochera 4.1.2.

4.1.7. Movimiento

Ídem del agente Offline 4.1.5, utiliza el mismo "listener" y el mismo objetivo de no detectar otro posible evento hasta que se lleve a cabo el proceso completo de una detección de movimiento. Este caso el problema es muy similar, el sensor podía detectar en milésimas de segundos varios eventos distintos, lo que generaba varias imágenes idénticas y una posible sobrecarga de procesamiento por parte de la cámara innecesaria.

4.1.8. Temperatura

Aquí si se puede encontrar una posible dificultad en este agente, como se vio en el apartado 2.2.1, la funcionalidad de este componente implica la necesidad de consultar el estado del pin GPIO, exactamente 40 veces, 5 secuencias de 8 bits cada una, y una lectura aproximadamente entre 0,27 y 0,35 milésimas de segundo entre señal y señal.

En el directorio de este proyecto, concretamente en la implementación de este agente, podemos ver el método en JAVA correspondiente a esta lectura del sensor, pero por el mismo motivo que la invocación de la cámara en JAVA, producía un retardo bastante considerable, debido a la obtención de información de este sensor mediante la máquina virtual de Java, por lo que la mayoría de las ejecuciones, la validación (la última secuencia de bits que se obtiene para comprobar que la lectura ha sido correcta) no era válida. Por tanto no se cumple un requisito mínimo para este apartado, en la que es necesario varias ejecuciones para determinar la temperatura y humedad.

Por este mismo motivo, se ha intentado realizar esta obtención de información del sensor mediante un Script en Python. Realizando un rápida investigación sobre el control de este sensor mediante este lenguaje de programación, se encontró en el blog [17] un Script en Python, desarrollado por

el autor citado en la bibliografía, que obtenía el 100 % de efectividad en las lecturas, en este enlace anterior se encuentra dicho Script bastante detallado y explicado para comprender su funcionamiento.

Para el desarrollo de este agente se ha utilizado la misma estrategia que utiliza el agente "Lluvia" en el apartado 4.1.4, es decir, es un agente asíncrono (no se bloquea esperando mensajes), mientras cada 3 segundos actualiza los valores de temperatura y humedad, ejecutando el Script Python anterior mencionado. Si recibe un mensaje, paralelamente lo almacena en su cola de mensajes. Cuando comprueba que tiene mensajes pendientes antes de actualizar los valores atiende las peticiones recibidas.

4.1.9. Controlador

Este agente específicamente no controla ningún pin GPIO, tan sólo como ya se ha mencionado en varios puntos anteriores y podemos ver en el diseño de este agente, sólo tiene la capacidad de llevar a cabo todas operaciones necesarias para completar una comunicación concreta, obteniendo el objetivo de la operación y llevando a cabo la misión más importante, que es distribuir las peticiones necesarias así como controlar que todo el sistema funcione perfectamente.

De aquí el nombre de una sociedad de agentes con una estructura jerárquica, en la que debe existir un agente como es este caso, que hace el papel de "jefe" que conoce a todos los agentes de la plataforma, como sus capacidades y sus identificadores correspondientes, y es el responsable de poder involucrar a los agentes necesarios para conseguir un objetivo común, o generar las comunicaciones pertinentes a los agentes necesarios por una petición recibida.

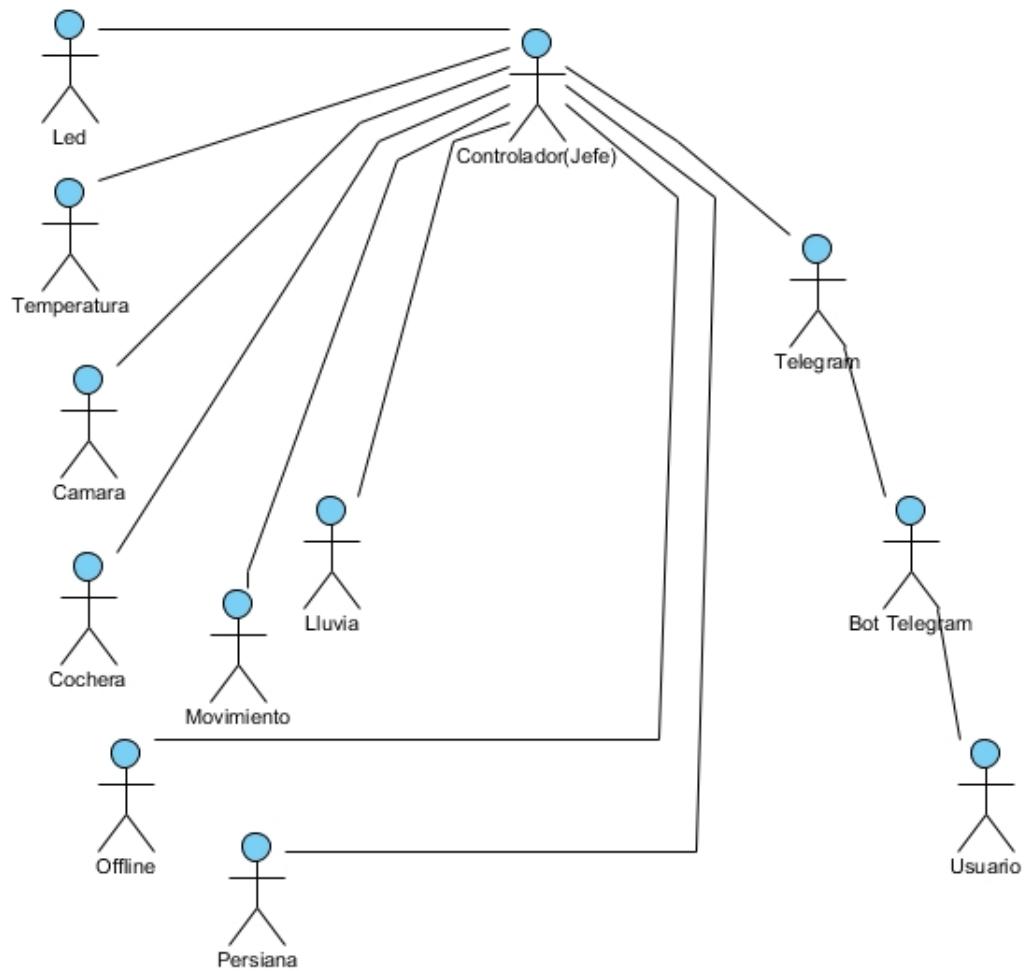


Figura 4.13: Arquitectura de la sociedad de agentes.

Capítulo 5

Versión Ampliada.

En resumen, el mayor trabajo de proyecto se centra en esta primera iteración, así como en primer lugar, investigar sobre que problema nos podemos encontrar, y que diferentes soluciones existen actualmente para solventar el problema, buscando su objetivo, sus ventajas y desventajas, e intentar mejorar la mayores factores posibles. Por lo tanto, ha consistido tanto del diseño como toda la implementación del sistema domótico descritos en los puntos anteriores, así como aplicar esa inteligencia para que la vivienda sea autónoma en todo lo posible.

Después de cubrir este objetivo, el siguiente objetivo que se plantea para este proyecto, consiste en poder establecer una comunicación externa con la vivienda, es decir, se pueda desde cualquier lugar saber el estado de nuestra vivienda, o algunos de sus componentes, o actuar sobre ella en un momento determinado por algún interés concreto.

Para ello la primera idea que se viene a la mente es una simple **aplicación web**, por su impacto a día de hoy, multiplataforma, volátil, accesible desde cualquier lugar... Pero desde la experiencia como alumno de este Grado en la UGR, se sabe lo que conlleva este proceso, es decir, facilitar un servicio web (http) para responder peticiones, diseñar una base de datos, montar tanto el servicio PHP como MySQL, y todos estos demás procesos que pueden ser tediosos y personalmente toman bastante tiempo, por lo que ¿No se podría encontrar una posible solución mas sencilla, fácil, y que no conlleve un proceso tan largo? Al sumergirse un poco en investigar que plataformas existen o que herramientas son más óptimas y podría utilizar es de ayuda.

Uno de los principales requisitos que se encuentra, es obtener un tiempo de respuesta lo suficiente real posible, por lo que esta idea de una aplicación Web puede que no sea la más recomendada.

5.1. Telegram.

Telegram es una aplicación de mensajería móvil, una plataforma abierta desarrollada por los fundadores de VKontakte, que consiste en la más importante red social de Rusia. Fue lanzada en 2013.

A pesar de la multitud de ventajas que podemos encontrar en esta aplicación [39], se procede a introducir las más destacadas y elegidas para este proyecto:

- Multiplataforma. Esta aplicación se encuentra hoy en día en todas las plataformas disponibles, como puede ser, Android, Iphone, Tablet, para escritorio Windows, Linux... Por lo que totalmente compatible para cualquier dispositivo que utilicemos o deseemos utilizar.
- Distribuido. Gracias a sus replicas en distintos lugares del mundo, y la disponibilidad de la gran cantidad de servidores que dispone la compañía a día de hoy, aseguran la fiabilidad y operatividad del servicio.
- Es una plataforma abierta, y totalmente libre, es decir, es gratis y no tiene ningún coste.
- Seguridad, privacidad, encriptación... .

5.1.1. Bot Telegram.

Un bot de Telegram podemos decir que es una aplicación o un cliente de terceros que se ejecuta dentro de Telegram, es decir, en otras palabras, es un usuario virtual que es dotado para interactuar con el a través de mensajes en el propio chat o comandos predefinidos [36].

Al contar con la Api de telegram que es totalmente abierta y disponible para cualquier usuario, tiene la posibilidad de crear un bot en cualquier momento y adaptarle un comportamiento específico. Ya hay miles de bots creados, con distintos fin, por ejemplos, juegos, plataformas de pagos, incorporando otros servicios externos, ciertamente hay una infinidad de posibilidades incluso la que tú deseas innovar.

Para llevar a cabo este proceso, debemos de utilizar dos interfaces o **API**, una primera muy fácil para poder crear nuestro bot, y la segunda un poco más completa que se encarga de poder dotar a nuestro bot de la funcionalidad deseada.

Aquí la genial idea y la gran pregunta, si se dispone de un sistema de agentes en la Raspberry Pi, que ya está desarrollada y funcionando, controla la vivienda unifamiliar domóticamente, es capaz de actuar antes ciertos acontecimiento que pueden ocurrir, pero se necesita ese extra para poder

comunicarme con este sistema y en tal caso el pueda también notificar cualquier problema o situación fuera de lo normal que haya ocurrido. Y si ahora se tiene una aplicación de mensajería, distribuida, totalmente compatible con cualquier dispositivo, segura, libre... **La unión** de estos dos sistemas sería perfecto para este objetivo, poder contar con un **Bot** que sea capaz de comunicarse con mi sistema de agentes, desde cualquier lugar y desde cualquier entorno. Ciento es que lo he conseguido, ahora se pasa a detallar el desarrollo de esta innovación.

5.2. Diseño.

En primer lugar es necesario generar dos nuevos agentes, el agente que se encarga de controlar el Bot de Telegram, que más adelante veremos como funciona y es capaz de administrar las peticiones, y un nuevo agente en mi plataforma de agentes de Magentix2 que su objetivo es sencillo, consta de redirigir las peticiones del usuario a través de la aplicación de Telegram al agente controlador (Jefe). Veamos los diagramas de secuencia y actividad para comprender mejor este proceso.

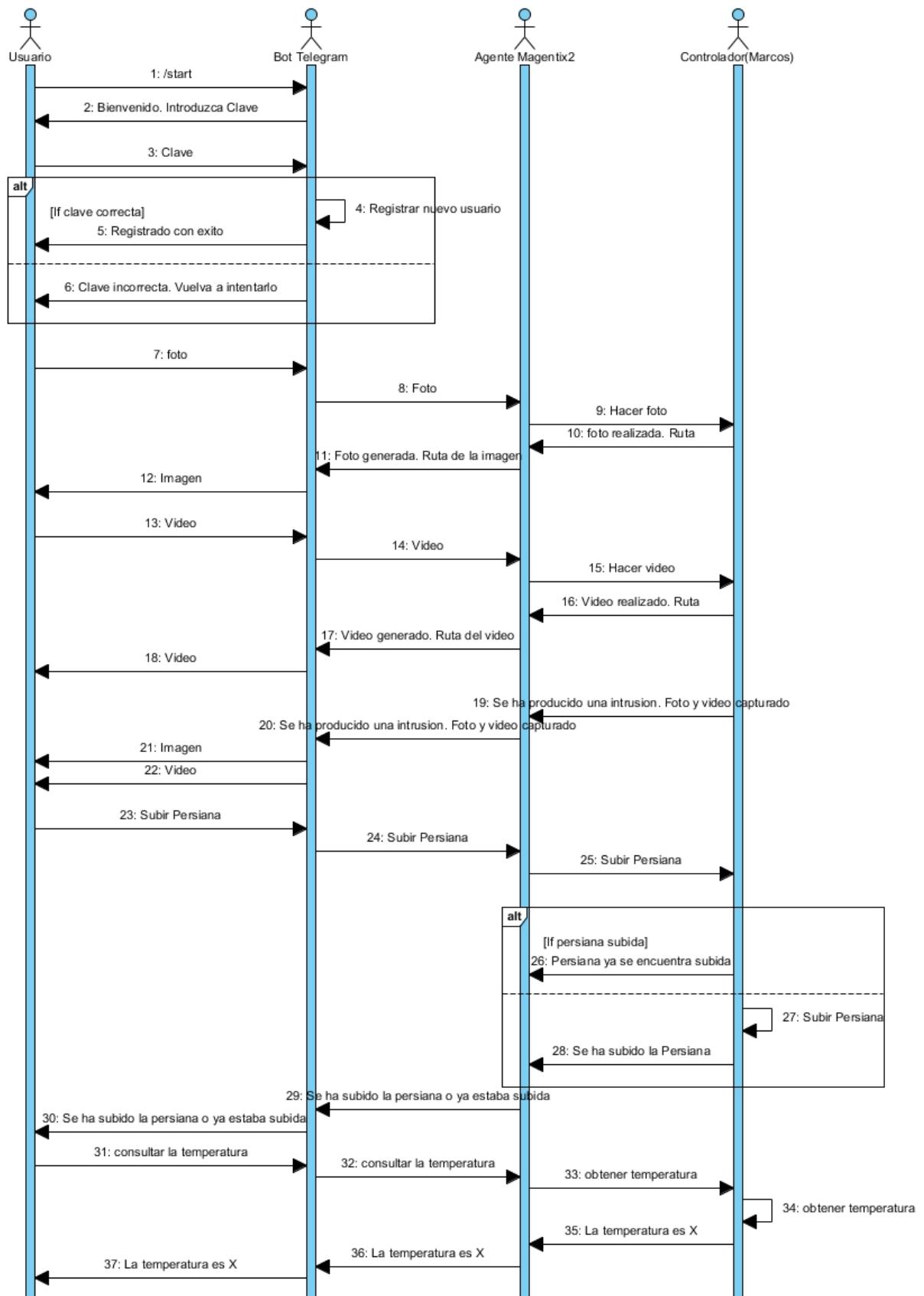


Figura 5.1: Diagrama de secuencia agentes Telegram y el sistema Raspberry Pi.

Por simplicidad la Figura 5.1 anterior, se ha obviado algunas operaciones, ya que son exactamente el mismo procesamiento, por ejemplo, accionar sobre una persiana que si está reflejado en el diagrama, es exactamente lo mismo que accionar la cochera, activar el sistema de seguridad..., otro ejemplo, la intrusión de un usuario en la vivienda con el sistema de seguridad, es el mismo proceso que la detección de lluvia. Debido también a la longitud del diagrama.

5.3. Instalación.

Antes de pasar al desarrollo de estos agentes con su implementación, se tiene que detallar unos detalles previos, necesarios para crear el Bot de Telegram.

En primer lugar se necesita hacer uso del Bot API [38], que aunque es un proceso bastante sencillo como veremos a continuación, es necesario para crear nuestro Bot. Esta interfaz es capaz de resolver las peticiones HTTP para registrar dicho Bot con el sistema propio de Telegram.

Es fundamental tener disponible la aplicación Telegram en nuestro dispositivo, ya sea móvil, tablet, navegador, en este caso se ha utilizado la plataforma de escritorio de Windows 10, y dicha aplicación para esta plataforma la podemos descargar aquí [37].

El siguiente paso, una vez tenemos instalada y disponible la aplicación Telegram, es utilizar la API que se ha mencionado anteriormente, en este caso consiste en buscar el Bot "padre", así lo han denominado los propios fundadores BotFather, dentro de la aplicación Telegram, como podemos ver en la Figura 5.2.

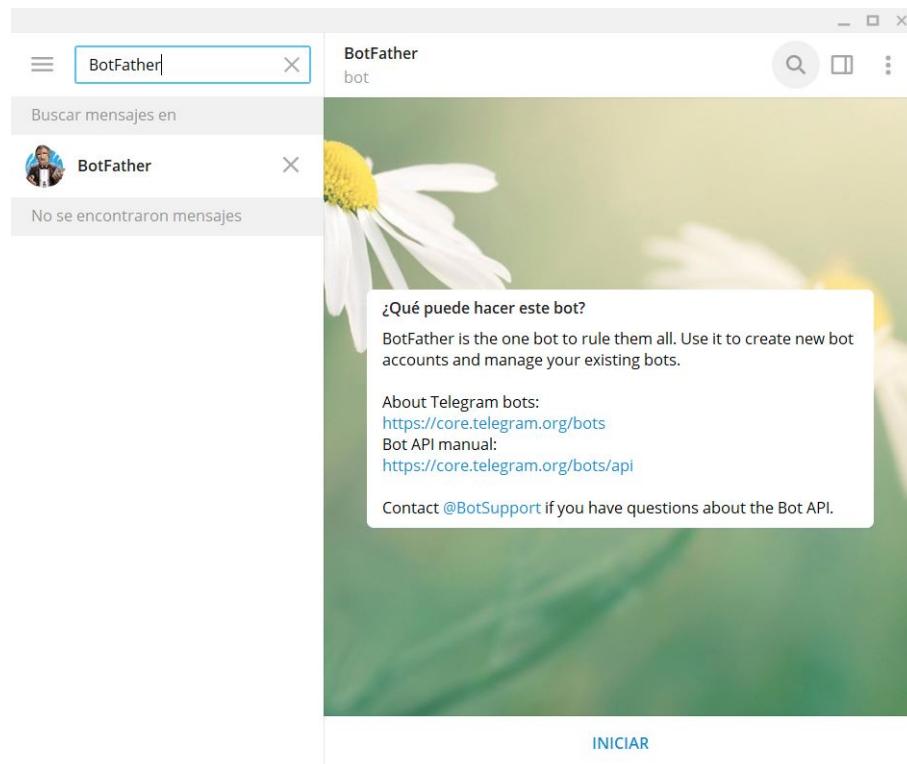


Figura 5.2: BotFather. API Bot de Telegram.

Una vez pulsamos en Iniciar, nos aparecerá una lista de comandos disponibles, así como una pequeña descripción de su objetivo, ahora mismo nos centramos en el comando ”\newbot ”, que podemos hacer click sobre él, o directamente escribir el comando como podemos ver en la Figura 5.3.

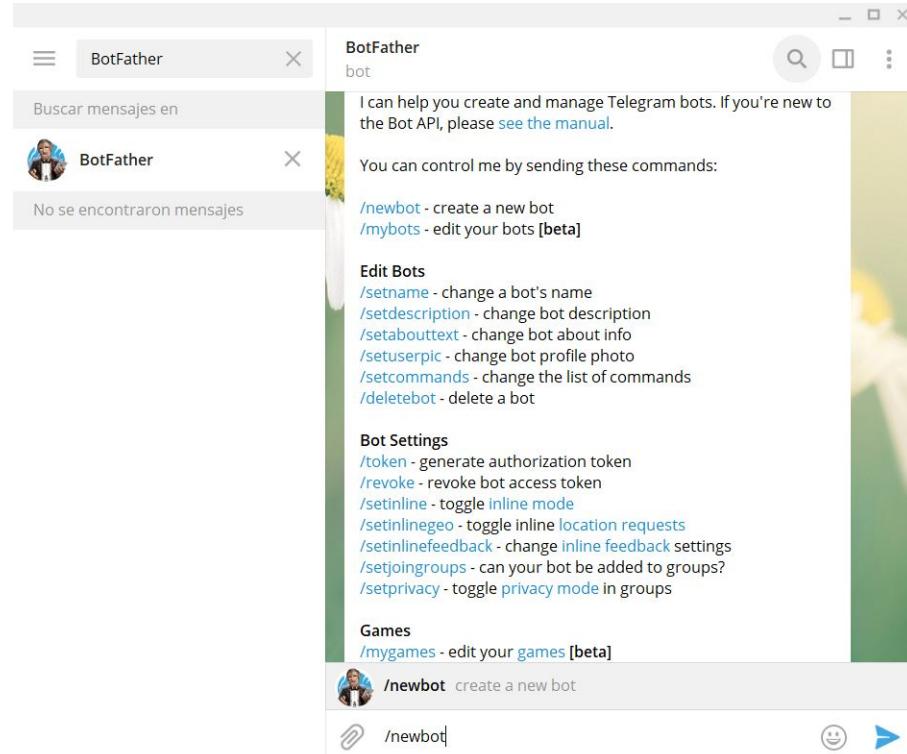


Figura 5.3: BotFather. API Bot de Telegram. \newbot.

A continuación debemos de introducir el nombre de nuestro Bot, y el nombre de Usuario de nuestro Bot, es decir, este último es el nombre que va a obtener nuestro Bot, por ejemplo para realizar su búsqueda, y debe terminar en la cadena de caracteres en Bot. Podemos seguir viendo el proceso con un ejemplo en la Figura 5.4.



Figura 5.4: Creación de un nuevo Bot de Telegram.

Ya tenemos nuestro Bot creado, podemos seguir haciendo operaciones como aplicarle una foto de perfil, una descripción, y mas operaciones [36]. Lo más importante de este punto y si es posible, guardar la cadena del token obtenido en el proceso anterior, como hemos visto en la Figura 5.4 anterior. Este token es único, y lo necesitaremos un poco más adelante para la implemetación de nuestro Bot, que nos permitirá identificarlo con este token.

Siguiendo el proceso anterior se generó el propio Bot de Telegram encargado de establecer la conexión con el sistema de agentes de mi vivienda domótica, cuyo nombre es "DomoticAgentsBot".

5.4. Implementación.

Antes de sumergirse en el código de este diseño, se procede a citar la librería que se ha utilizado y la documentación [9] disponible para más información.

Es cierto que la mayoría de ejemplos y aplicaciones de Bot actuales se

encuentran diseñadas y programadas en el lenguaje de programación Python, pero en este caso ya que está diseñado el sistema de agentes en lenguaje JAVA, se debe de continuar con este lenguaje, por eso se utiliza esta librería [8], de código abierto, libre, que nos permite solventar las peticiones HTTP que se realicen a través de la aplicación Telegram.

Para llevar a cabo este proceso, y desarrollar la funcionalidad de nuestro Bot de Telegram, en primer lugar se utiliza la herencia de la clase "TelegramLongPollingBot" disponible en esta librería, que hereda los métodos mínimos y necesarios para llevar a cabo una comunicación o interacción entre el usuario y este Bot, como podemos ver en la Figura 5.5.

```

17 import org.telegram.telegrambots.api.methods.send.SendMessage;
18 import org.telegram.telegrambots.api.methods.send.SendPhoto;
19 import org.telegram.telegrambots.api.methods.send.SendVideo;
20 import org.telegram.telegrambots.api.objects.Update;
21 import org.telegram.telegrambots.bots.TelegramLongPollingBot;
22 import org.telegram.telegrambots.exceptions.TelegramApiException;
23 /**
24 *
25 * @author marcos
26 */
27 public class Bot extends TelegramLongPollingBot{
28
29
30     private LinkedList<Long> usuarios;
31     private Agente a;
32     private String clave;
33
34     public Bot() throws Exception {
35         super();
36         usuarios = new LinkedList<Long>();
37         a = new Agente(new AgentID("Telegram"));
38         a.start();
39         clave= "1234";
40         a.asignarBot(this);
41     }
42 }
```

Figura 5.5: Estructura Java de DomoticAgentsBot.

Aquí se encontró un gran problema, ya que, necesitaba la implementación de esta clase de la figura anterior para la administración del Bot, pero JAVA no permite la multiherencia, por lo que se dificulta la posibilidad de heredar de la clase "TelegramLongPollingBot" y de la clase "SingleAgent" para poder definir en la misma clase tanto el agente que se encarga de administrar el Bot de Telegram, y el agente de Magentix2 que se encargaría de establecer la comunicación con el sistema de agentes de la Raspberry Pi,

los cuáles pueden interaccionar con la vivienda.

Para solventar esta situación, se definió una nueva clase independiente para el Agente de Magentix2, heredando de "SingleAgent" y dotándole del método de la Figura 5.6, con este objetivo, la clase Bot tiene el atributo del agente, es decir, cuando "DomoticAgentsBot" es iniciado se encarga de definir y lanzar el agente "Telegram" en la plataforma Magentix2 que se encuentra en la Raspberry Pi, seguidamente en su constructor (línea 40 de la Figura 5.5) la invocación de este método, el agente "Telegram" recibe una referencia el objeto propio de "Bot". Desde este momento ambos agentes se conocen y pueden acceder o invocar los métodos deseados ya sea de una clase u otro, el cuál era mi objetivo, y podemos decir que tenemos dos agentes distintos de diferentes plataformas (Magentix2 y Telegram) pero con la misma metodología de un agente, comunicarse entre sí para conseguir un objetivo común.

```
179     public void asignarBot(Bot bot){
180         this.bot=bot;
181     }
```

Figura 5.6: Estructura Java de DomoticAgentsBot.

El siguiente paso es importante para identificar nuestro Bot en la plataforma de Telegram, para cuando se realice la interacción de un usuario con nuestro Bot, es decir, un usuario envíe un mensaje a nuestro Bot por la aplicación Telegram, sea recibido por nuestro agente (Bot). Para ello es necesario la sobrecarga de los métodos como podemos ver en la Figura 5.7, con el nombre de nuestro Bot y el token obtenido en su creación.

```
167     @Override
168     public String getBotUsername() {
169         // Se devuelve el nombre que dimos al bot al crearlo con el BotFather
170         return "DomoticsAgentsBot";
171     }
172
173     @Override
174     public String getBotToken() {
175         // Se devuelve el token que nos generó el BotFather de nuestro bot
176         return "584124852:AAEcg8nSewrq7vXoAgrUl0mqx2m57mqTovg";
177     }
---
```

Figura 5.7: Administrando nuestro Bot de Telegram.

Ya sólo falta un último método, nos fijamos en la Figura 5.8, cualquier comunicación o mensaje que se genere en el chat de nuestro Bot de Telegram, podremos identificarlo en este método, así como analizar el contenido del mensaje recibido, identificar el emisor...

```

48     @Override
49     public void onUpdateReceived(final Update update) {
50         // Esta función se invocará cuando nuestro bot reciba un mensaje
51
52         // Se obtiene el mensaje escrito por el usuario
53         String mensajeRecibido = update.getMessage().getText();
54
55         // Se obtiene el id de chat del usuario
56         Long chatId = update.getMessage().getChatId();
57         SendMessage mensaje = new SendMessage().setChatId(chatId);

```

Figura 5.8: Implementando la recepción de mensajes de nuestro Bot de Telegram.

En el directorio del proyecto se puede ver todo el código completo y la funcionalidad completa de ambos agentes, por simplicidad se detalla un simple ejemplo, para observar como se puede analizar un mensaje recibido, y actuar sobre él, por ejemplo enviando una respuesta.

Por ejemplo, se va a seguir el proceso de obtener una foto actual de la vivienda a través de la cámara de seguridad. Para ello se envía a través del chat del Bot de Telegram ”DomoticAgentsBot” el mensaje del comando ”\hacer_foto”, y en el método de la Figura 5.8 se añade una cláusula como podemos ver en la Figura 5.9, en la que cuando se recibe este tipo de mensaje se ejecutarán las ordenes correspondientes. En este caso, se invoca el método del agente ”Telegram” encargado de comenzar la comunicación con el sistema de agentes de la Raspberry Pi, para obtener una captura con la cámara.

```

63 ▼ case "/hacer_foto":
64     try {
65         a.solicitarMultimedia("foto");
66     } catch (InterruptedException ex) {
67         Logger.getLogger(Bot.class.getName()).log(Level.SEVERE, null, ex);
68     }
69     break;

```

Figura 5.9: Recibiendo un comando desde el Bot de Telegram.

Una vez que el proceso se ha realizado, podemos notificar al usuario de Telegram que nos ha enviado la petición, enviando un mensaje o en este caso enviando la imagen obtenida. Para enviar un mensaje a través de nuestro Bot de Telegram podemos hacer uso de los métodos como podemos ver en la Figura 5.10.

```

194     public void enviaVideo(String ruta, String descripcion) throws TelegramApiException{
195         for(int i=0; i<usuarios.size(); i++){
196             SendVideo msg = new SendVideo()
197                 .setChatId(usuarios.get(i)).setNewVideo(new File(ruta))
198                 .setCaption(descripcion);
199             sendVideo(msg);
200         }
201     }
202 }
203
204     public void enviarMensaje(Long usuario, String descripcion){
205         SendMessage mensaje = new SendMessage().setChatId(usuario);
206         mensaje.setText(descripcion);
207         try {
208             // Se envía el mensaje
209             execute(mensaje);
210         }
211         catch (TelegramApiException e) {
212             e.printStackTrace();
213         }
214     }
215

```

Figura 5.10: Envío de mensajes a través del Bot de Telegram.

En el caso del método ”enviaMensaje” podemos enviar un mensaje a un usuario concreto, con una descripción determinada, o, un vídeo a través del método ”enviaVideo” de la figura anterior, utilizando los métodos correspondientes incluidos en la API.

Por último destacar el Script de Python de la generación de un vídeo a través de la cámara. Ya que es una condición heredada de Telegram, porque esta plataforma está diseñada para poder reproducir vídeos en un formato ”.mp4”, y la cámara genera contenido en formato ”.h264”, por lo que el contenido sí era enviado a través del Bot, pero no podría ser reproducido. Sólo mencionar por ello este dato sencillo, que se incluye en el Script de Python la conversión de un formato a otro para la reproducción óptima.

5.5. Novedades.

Con esta versión avanzada ya se puede disponer de una total comunicación con la vivienda domótica, es decir, a través de esta aplicación Telegram y mediante el Bot ”DomoticAgentsBot”, podemos llevar a cabo las acciones pertinentes, como accionar la persiana, la puerta del garaje, saber la temperatura actual de la vivienda, la humedad, obtener una foto o un vídeo en un determinado momento.

Además de todo lo anterior, las características más novedosas son las siguientes:

- Detección de lluvia. Como ya se ha mencionado en los apartados ante-

riores, la vivienda queda capacitada para detección de lluvia, y en tal caso, si el sistema de seguridad está activado y la persiana se encuentra subida, automáticamente se cierra, pero todo esto podía ocurrir en segundo plano, sin que nosotros como usuarios apreciemos tal objetivo si nos encontramos fuera de casa. En este caso, gracias al Bot de Telegram, seremos notificados del comienzo de la lluvia, si nos hemos dejado subida la persiana, y del momento que la lluvia amaine.

- Detección de intrusos. Igualmente a lo anterior, en caso del sistema de seguridad este activado, y se de la situación de un intruso en la vivienda, obtendremos una fotografía del intruso o de la situación dada en prácticamente tiempo real, y posteriormente, por seguridad se generará un vídeo como prueba de testimonio de lo ocurrido. Dicho vídeo también será enviado por el Bot de Telegram y lo obtendremos en el dispositivo justo después de ser generado el contenido.
- Activación de Seguridad. Ahora ya es posible también activar o desactivar el sistema de seguridad desde el dispositivo móvil o la plataforma que estemos utilizando, sin tener que activar el sistema de seguridad accionando manualmente el botón antes de salir de casa. En caso de olvido, podremos activarlo en cualquier momento y desde cualquier parte.
- Seguridad extra. Se sabe que cualquier usuario que disponga de una cuenta de Telegram, puede realizar la búsqueda de nuestro Bot y poder interaccionar con él sin ningún problema, debido a que todos los Bots que se generen en Telegram son públicos. Por ello aquí si encontramos un pequeño problema de seguridad, ya que se está dando libre y total uso de nuestra vivienda a cualquier usuario. El Bot ha sido incluido con una clave, podemos fijarnos en la línea 32 de la Figura 5.5, en la que para cada nuevo usuario que inicie nuestro Bot, le pedirá dicha **clave**, por la que sin ella no podrá tener acceso a la funcionalidad de nuestra vivienda domótica.
- Mutiusuario. Aprovechando esta característica que cualquier usuario puede usar nuestro Bot (conociendo la clave), o incluso también podemos compartir la clave con un usuario, es decir, como si prestáramos la llave de nuestra casa a un vecino o familiar para regar las plantas, pero en este caso electrónicamente facilitándole dicha clave. Y no solo eso, también podría tener acceso todos los componentes de la vivienda independientemente.
- Un poco relacionado con lo anterior, se ha añadido la capacidad de notificación individualmente, es decir, si deseamos consultar por ejemplo la temperatura, sólo sea notificado al usuario que ha realizado la

petición, o por ejemplo, en caso de intrusión si serán notificado a todos los usuarios que estén registrados en ese momento. Esto permite la difusión de cualquier problema importante, pero evita la saturación de mensajes por parte de otros usuarios registrados que no es de importancia para los restantes usuarios.

- Indirectamente podemos aprovechar otras herramientas potentes que facilita dicha aplicación de Telegram, como por ejemplo, el reconocimiento por voz, es decir, podemos transmitir los distintos comandos usando la escritura por voz de esta aplicación, aumentando así la capacidad de **accesibilidad** de nuestro sistema a aquellas personas con cierta discapacidad.

Capítulo 6

VersionFinal.

Una vez todo implementado, se realizaron las diferentes baterías de pruebas para comprobar efectivamente el rendimiento y funcionamiento de este proyecto. Es cierto que teóricamente es una parte y otra parte es la práctica, es decir, que verdaderamente funcione en la realidad.

Las pruebas obtuvieron resultados exitosos, comprobando la carga de trabajo del sistema de agentes, los tiempos de respuesta... Ante la carga de trabajo, dentro del máximo que puede ofrecer la Raspberry Pi, desde el primer inicio, arranque del sistema operativo Raspbian, arranque de la plataforma Magentix2, la generación y lanzamiento de todos los agentes de dicha plataforma, el arranque de todos los sensores, y el inicio del Bot de Telegram..., la carga de trabajo si podía llegar a unos valores entre el 80 % y 92 %, en los 38 segundos que tardaba establecer todo este proceso.

Una vez ya todo establecido y quedando a la espera de alguna petición o situación específica, la carga se volvía totalmente al 0 %, excepto cuando ya se aplicaban peticiones y se generaban las distintas comunicaciones, en tal caso la carga oscilaba entre el 5 % y el 25 %. Unos datos bastante interesantes para el componente de computación incluido en la Raspberry Pi.

Después de todo esto, se decidió realizar una pequeña maqueta de una vivienda unifamiliar, para trasladar todo este trabajo de éste proyecto a una situación real en miniatura. Los resultados obtenidos fueron los que se pueden observar en la Figura 6.1.



Figura 6.1: Maqueta finalizada.

Y a grandes pasos, muestro un poco el desarrollo del diseño y construcción de la maqueta:

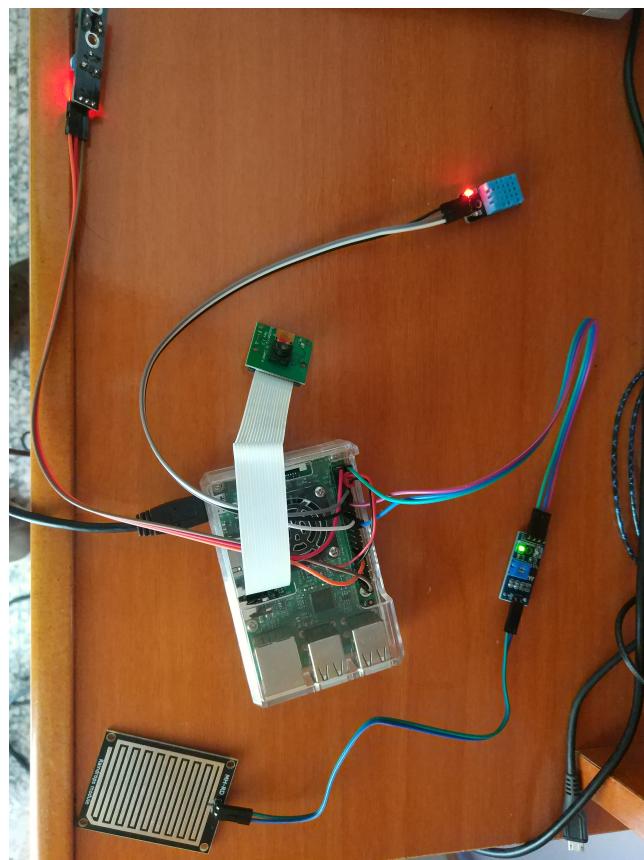


Figura 6.2: Principio de estado del sistema domótico.

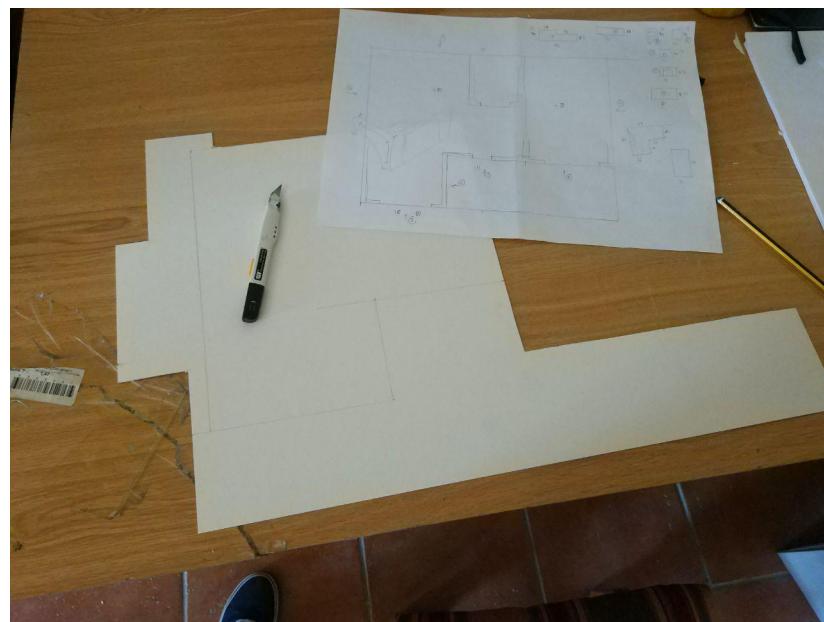


Figura 6.3: Diseño de la maqueta.

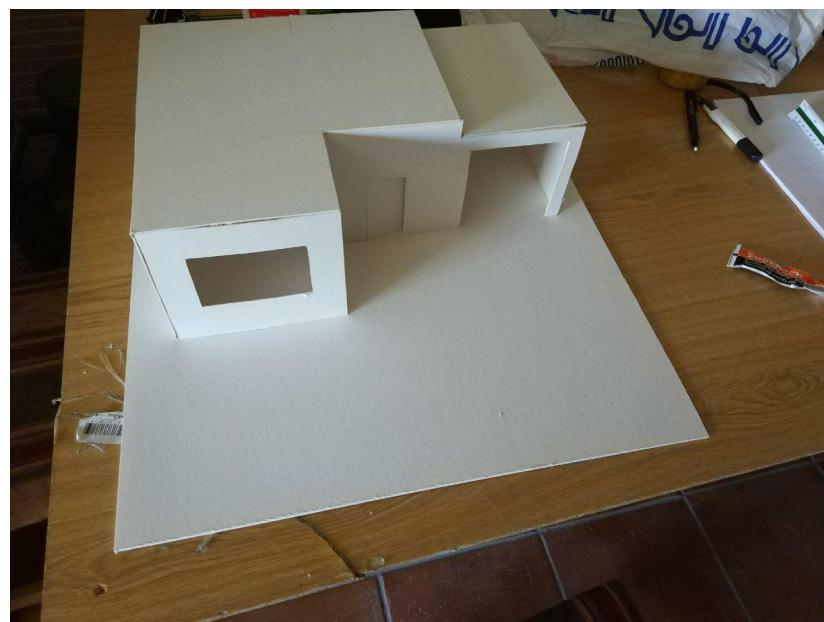


Figura 6.4: Construcción de la maqueta.

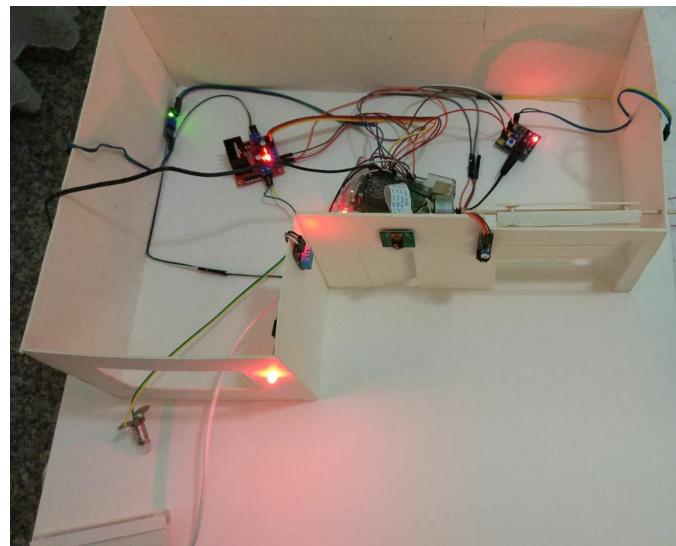


Figura 6.5: Primeras pruebas de la maqueta.

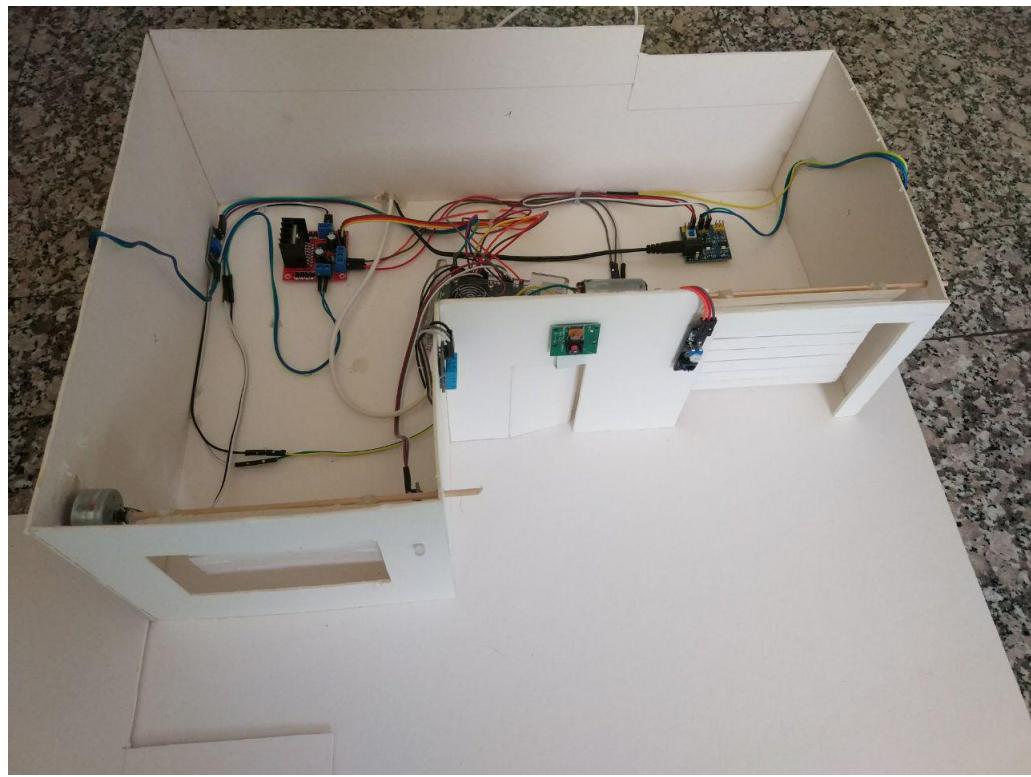


Figura 6.6: Añadidos todos los sensores.

Podemos verificar en estos últimos pasos, las pruebas se ejecutaron en tiempo real, manteniendo una respuesta bastante óptimo, por lo que podemos concluir con una vivienda domótica real y con grandes características, así como sus novedosas funcionalidades.

Capítulo 7

Conclusiones.

Llegado a este punto, se ha conseguido la implantación de un sistema domótico completo para una vivienda unifamiliar, marcando la diferencia en una serie de características que han sido demostradas y con una consistencia a considerar. Destacando algunas de ellas.

7.1. Bajo Coste

Llegando a la conclusión después de la investigación de trabajos relacionados e instalaciones reales realizadas, consiguen un coste entre 1,5 % y el 6,8 % del valor total del presupuesto de la vivienda, o incluso una instalación sencilla con sólo varias funcionalidades como, de seguridad, detección de fuego y minuciosas características básicas con un total de 1.364€.

A lo largo de este documento y en cada apartado se ha ido mencionando el coste total de cada componente. Que su suma total es de **32.91 €**, cierto es que no queda incluido el coste de instalación y el tiempo empleado para diseñar e implementar este sistema, pero realmente ¿Un sistema con mayor funcionalidad, autónomo, capaz de actuar y decidir por el mismo, escalable, y totalmente controlado y accesible desde cualquier parte, por este precio?. Y nos referimos a una vivienda unifamiliar, ¿Que ocurriría sobre esta misma situación pero potencialmente en un gran edificio? No sólo un edificio residencial, con 2 o 3 pisos, y 2 o 3 viviendas por piso, si por ejemplo, el actual edificio "La vela" de BBVA en Madrid 1.2.3.

Para esta situación futura, si será necesario ampliar este sistema de agentes, ya que será necesario incluir varias Raspberry Pi para abarcar todos los pisos (una por piso o incluso varias Raspberry Pi por piso), por lo que se deberá diseñar varios agentes controladores(Jefes) para interconectar todas estas Raspberry Pi, ampliando la jerarquía de agentes, gracias a esto será posible abarcar lo propuesto.

Aprovechando este punto del mínimo coste podemos ver una clara posibilidad de escalabilidad, en la que podemos incluir tantos sensores como existan, funcionalidades como alcancen nuestros conocimientos, incluso situaciones que pueden ser un reto.

7.2. Sistema de Agentes

Gracias a esta arquitectura social de agentes Magentix2, en la que cada agente es independiente, con una estructura interna distinta cada uno, capaces de percibir alguna característica de su entorno real, y mediante una comunicación con otros agentes u organizándose de una cierta manera, conseguir aquellos objetivos establecidos incluso su propio objetivo común, de manera totalmente autónoma.

Esto ya es un gran avance percibir en tiempo real una característica del mundo real, y actuar en función de dichos datos obtenidos, y no sólo eso, sino capaz de decidir entre sus capacidades qué acción tomar o es la más adecuada para un caso concreto, dando esa "inteligencia". Un software estándar sólo se obtendría esta característica a base de comprobaciones, teniendo en cuenta todas y cada una de las posibles situaciones que se pueden dar, aumentando el grado de dificultad tanto de diseño como de implementación, y probablemente inabordable para futuras actualizaciones o posibles ampliaciones.

Debido esto, se consigue una gran organización, simplicidad y ahorro de trabajo descomunal, tanto para el diseño como para el desarrollo, ya que podría darse un software más estándar con una dificultad exponencial a medida que crece el problema, llegando a un sistema sencillo, optimizado, y de un orden de dificultad lineal a medida que crece el problema.

Otra conclusión es la capacidad de comunicarse entre agentes, tienen distintas características, distintas plataformas, distintos lenguajes, pero el objetivo es el mismo, comunicarse a través de mensajes. Por eso poder conseguir dos agentes totalmente de distintas plataformas, como por ejemplo, Telegram y Magentix2, y con esta funcionalidad se ha permitido, a través de dichas plataformas, poder conectar los agentes necesarios, y tener total control con la vivienda unifamiliar desde cualquier dispositivo, y ser notificado ante cualquier situación específica.

Capítulo 8

Trabajo futuro

A lo largo de este proyecto, se ha conseguido un sistema completo, funcional y llevado a una situación real en miniatura, totalmente escalable, es decir, podemos añadir tantos sensores y nuevas funcionalidades como deseemos, así como sus respectivos agentes que los controlen.

Pero para potenciar aún más este proyecto, en un trabajo futuro, se podría aplicar varias posibilidades como veremos en los apartados siguientes.

8.1. Vivienda unifamiliar real.

Ya se ha demostrado realmente como puede ser un diseño, desarrollo y funcionamiento correcto de este sistema para una vivienda unifamiliar domótica, pero en este caso a pequeña escala.

Recordando un poco, las señales de comunicación son digitales, comprendidas entre escasos voltios cercanos a 0V para señales bajas ("0") y 5V para señales digitales altas ("1"), a medida que la distancia del cable aumenta, el propio cable actúa como resistencia, aplicando una caída de voltaje o pérdida de intensidad, interferencias que se pueden producir en el cable, posibles lecturas erróneas.... Como he mencionado, a unos escasos 40cm no hay ningún tipo de problema.

Una solución muy simple y sencilla es incorporar a los componentes de los sensores una comunicación vía Wifi o Bluetooth, que nos permite comunicación a largas distancias, pero personalmente esta idea podría ser la más tediosa, a la vez que también puede encarecer económicamente el proyecto, identificar y administrar todas las señales de cada uno de los sensores puede ser muy laborioso y confuso.

Realizando una investigación sobre este ámbito, se llega a la conclusión que a grandes rasgos las largas distancias de cable produce una resistencia

interna, es decir, el propio cable produce una caída de tensión. Esta resistencia viene dada por la longitud del cable, la sección o grosor del mismo, y el material por el que esta compuesto.

La alimentación de cada sensor no es problema, ya que puede ser administrada por una fuente de alimentación externa, como por ejemplo hemos utilizado en este proyecto con el módulo 2.2.5, ya que podemos alimentar los sensores organizados por grupos en función de la cercanía que se encuentren, o incluso como hemos mencionado anteriormente, el cable que he utilizado para este proyecto (2.54mm de cobre), a unos 100 metros, puede producir una pequeña resistencia aplicando una caída de tensión de aproximadamente 0.5V, que se puede subsanar aplicando esa diferencia de voltaje con el módulo anterior. Hay otras opciones, como ampliar el grosor del cable, en el cuál disminuye la resistencia. Es recomendable también añadir unos **optoacopladores** para proteger el sistema, en este caso, nuestra Raspberry Pi, para posibles potenciales eléctricos que podría dañar nuestro elemento.

Sin duda, la conclusión de utilizar es el estándar de comunicación **RS485** [7], o también conocido como EIA-485, este estándar trabaja mediante la capa física de comunicación, y consiste en trasmitir los datos o las señales digitales a través de un cable par trenzado y con un apantallamiento, por un cable se transmite la señal original y por el restante se emite la señal inversa. Esto permite una comunicación totalmente segura, sin pérdidas, evitando las interferencias, y llegar hasta una longitud de **1200 metros**.

Para llevar a cabo este proceso se necesita un módulo como vemos en la Figura 8.1, llamado módulo MAX485, y permite la utilización de este estándar de comunicación mencionado anteriormente. Con un coste de **0.35€** aproximadamente.

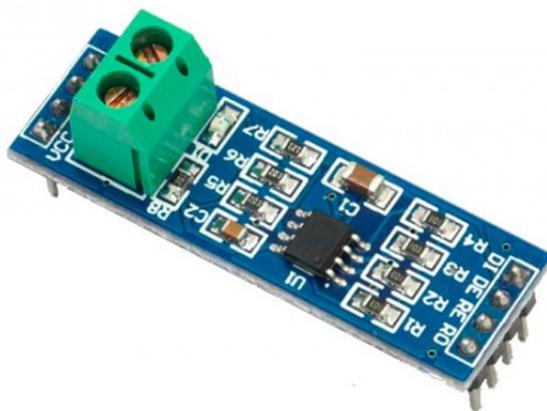


Figura 8.1: Módulo MAX485, utilizando el protocolo RS485.

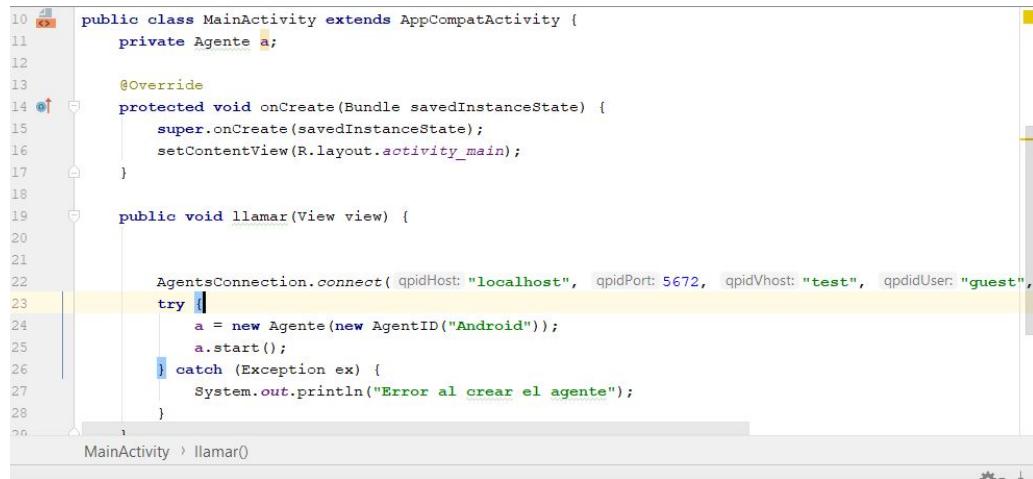
Como datos de interés este estándar ha sido utilizado en algunas centralitas de los coches actuales, para realizar comunicaciones como posibles averías, o incluso recoger datos como, el kilometraje, verificaciones de estados... Por lo que con esta solución nos aseguramos la correcta funcionalidad y comunicación con nuestro sistema de agentes, y nuestra vivienda.

8.2. Aplicación Android

Unas de las posibilidades consiste en incorporar una aplicación Android, para aprovechar su potencialidad, tanto visual como operatividad, posiblemente para no depender de una plataforma externa como Telegram, por motivo de suspensión del servicio, un error, avería...

Ciertamente se ha intentado diseñar una aplicación en Android, pero realizando unas primeras pruebas de contacto, para asegurar el posible funcionamiento con estas herramientas, como son la conexión con la plataforma Magentix2, o generación de agentes en esta plataforma Android. Después de incorporar la librería adecuada que hemos utilizado para la jerarquía de agentes, "Magentix", el sistema lo reconoce pero existen algunas incompatibilidades en la ejecución.

En la siguiente Figura 8.2, podemos ver como la librería si esta disponible, reconociendo los métodos heredados del agente, pero en la ejecución ocurre una serie de errores, y la aplicación es cerrada automáticamente.



```
10 public class MainActivity extends AppCompatActivity {
11     private Agente a;
12
13     @Override
14     protected void onCreate(Bundle savedInstanceState) {
15         super.onCreate(savedInstanceState);
16         setContentView(R.layout.activity_main);
17     }
18
19     public void llamar(View view) {
20
21
22         AgentsConnection.connect( qpidHost: "localhost", qpidPort: 5672, qpidVhost: "test", qpidUser: "guest",
23         try {
24             a = new Agente(new AgentID("Android"));
25             a.start();
26         } catch (Exception ex) {
27             System.out.println("Error al crear el agente");
28         }
29     }
30 }
```

Figura 8.2: Ejecución Aplicación Android.

Possiblemente se conseguir implementar esta posibilidad, y conseguiremos tener una aplicación propia, sin dependencia de otras plataformas.

8.3. Aprendizaje

Es muy común escuchar hoy en día, técnicas como ”machine learning, big data...” en la que consiste analizar grandes masas de datos, poder obtener algunos resultados interesante, o conseguir a base de ir almacenando posibilidades y situaciones ocurridas que un sistema vaya ”aprendiendo”, así que aplicar esto no es nada tedioso.

El sistema de agentes puede ir tomando esa inteligencia, por ejemplo, de saber las costumbres y el comportamiento de cada individuo de la vivienda, y actuar en función de estas características. Un ejemplo sencillo, por ejemplo, en un domicilio cualquiera, la temperatura ideal es una temperatura baja, y sin mucha luminosidad en la vivienda, este sistema de agentes sabrá reconocer a la familia de la vivienda, y saber quién de ellos se encuentra en casa, por lo tanto, automáticamente la vivienda la adapta a sus necesidades o preferencias.

Bibliografía

- [1] Ingeniería y Diseño Electrónico. Andrés Cano. ELECTRO-NILAB. Uso de driver l298n para motores dc y paso a paso con arduino. <https://electronilab.co/tutoriales/tutorial-de-uso-driver-dual-l298n-para-motores-dc-y-paso-a-paso-con-arduino/>. Actualizado el 14 de Mayo del 2014.
- [2] © 2018 Arduino. Learn arduino. <https://www.arduino.cc/en/Guide/HomePage>. Consultado el 8 de Agosto del 2018.
- [3] ArduinoModules. Ky-004 key switch module. <https://www.arduinomodules.info/ky-004-key-switch-module/>. Actualizado el 10 de Marzo del 2016.
- [4] Blanca Sanjuanbenito. BBVA. Ciudad bbva: La vela. <https://www.bbva.com/es/ciudad-bbva-unmacion-digital/1>. Actualizado Diciembre de 2015.
- [5] CETRONIC. Catálogo de sensores para arduino y raspberry pi. <https://www.cetronic.es/sqlcommerce/disenos/plantilla1/seccion/Catalogo.jsp?idIdioma=&idTienda=93&cPath=1343&página=2&expand=0>. Consultado el 8 de Agosto del 2018.
- [6] Marcos Avilés. TFG Universidad de Granada. Código completo. <https://drive.google.com/open?id=1OE-LAmFCiYqcBnhhZ353sg4907yoBRKd>. Consultado el 19 de Agosto del 2018.
- [7] Antonio Javier Barragán Piña. Profesor Contratado Universidad de Huelva. Tecnología de transmisión rs485. <http://uhu.es/antonio.barragan/content/311-tecnologia-transmision-rs485>. Consultado el 25 de Agosto del 2018.
- [8] Ruben Bermudez. Ingeniero de Software. Download librería telegrambots. <https://github.com/rubenlagus/TelegramBots/releases>. Actualizado el 8 de Agosto del 2018.

- [9] Ruben Bermudez. Ingeniero de Software. Documentation telegramlong-pollingbot. <https://github.com/rubenlagus/TelegramBots/blob/27e0751b7e249bd8b2484d001e2245f1d68c0bb0/TelegramBots.wiki/Getting-Started.md>. Consultado el 22 de Agosto del 2018.
- [10] GTI-IA. Grupo de Tecnología Informática Inteligencia Artificial. Documentation magentix2. <http://gti-ia.upv.es/sma/tools/magentix2/documents.php>. Consultado el 19 de Agosto del 2018.
- [11] Politécnica de Valencia. Download magentix 2. <http://users.dsic.upv.es/grupos/ia/sma/tools/magentix2/downloads.php>. Consultado el 21 de Julio del 2018.
- [12] Universidad de Valencia. Magentix 2. <http://www.gti-ia.upv.es/sma/tools/magentix2/>. Consultado el 21 de Julio del 2018.
- [13] Andrew Dillon Hopding. Software Developer. Class rpicamera. <http://hopding.com/docs/jrpicam/com/hopding/jrpicam/RPiCamera.html>. Consultado el 20 de Agosto del 2018.
- [14] Andrew Dillon Hopding. Software Developer. Dowload jrpicam. <https://github.com/Hopding/JRPiCam/releases/download/v1.1.1/jrpicam-1.1.1-release.zip>. Consultado el 20 de Agosto del 2018.
- [15] Domintell. Presupuesto componentes para una vivienda unifamiliar. <http://www.domintell.es/presupuesto>. Consultado en Julio de 2018.
- [16] Rodríguez Diego Miguel. TFG Gestión domótica de una casa unifamiliar. <https://uvadoc.uva.es/bitstream/10324/26328/1/TFG-P-709.pdf>. 2017.
- [17] Sergio Correa Lopez. Ingeniero Civil Electrónico. Script python sensor dht11. <http://fpaez.com/sensor-dht11-de-temperatura-y-humedad/>. Actualizado el 15 de Agosto del 2016.
- [18] Raspberry Pi Foundation. Camera module v2. <https://www.raspberrypi.org/products/camera-module-v2/>. Consultado el 12 de Agosto del 2018.
- [19] Raspberry Pi Foundation. Downloads. <https://www.raspberrypi.org/downloads/raspbian/>. Consultado el 15 de Agosto del 2018.
- [20] Raspberry Pi Foundation. Using python and picamera. <https://projects.raspberrypi.org/en/projects/getting-started-with-picamera>. Consultado el 20 de Agosto del 2018.

- [21] Raspberry Pi Foundation. About us. <https://www.raspberrypi.org/about/>. Consultado el 7 de Agosto del 2018.
- [22] Raspberry Pi Foundation. Buy raspberry pi zero. <https://www.raspberrypi.org/products/raspberry-pi-zero/#buy-now-modal>. Consultado el 7 de Agosto del 2018.
- [23] Raspberry Pi Foundation. Raspberry pi 3 model b. <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>. Consultado el 7 de Agosto del 2018.
- [24] © ICStation.com. Ky-033 tracking sensor module infrared obstacle avoidance for arduino. <http://www.icstation.com/tracking-sensor-module-infrared-obstacle-avoidance-arduino-project-p-2787.html>. Consultado el 12 de Agosto del 2018.
- [25] Id.domotica. Premio a la mejor instalación en 2017. <https://www.iddomotica.com/noticias/id-domotica-premios-madrid-2017/>. 2017.
- [26] Luis Llamas. Ingeniero Industrial. Detector de lluvia con arduino y sensor fc-37 o yl-83. <https://www.luisllamas.es/arduino-lluvia/>. Actualizado el 13 de Febrero del 2016.
- [27] Luis Llamas. Ingeniero Industrial. Medir temperatura y humedad con arduino y sensor dht11-dht22. <https://www.luisllamas.es/arduino-dht11-dht22/>. Actualizado el 29 de Marzo del 2016.
- [28] © 2018 Slashdot Media. Win32 disk imager. <https://sourceforge.net/projects/win32diskimager/>. Consultado el 15 de Agosto del 2018.
- [29] MARTA JURADO. Periódico El Mundo. La casa más 'inteligente' de 2014 <http://www.elmundo.es/economia/2015/04/22/55375859ca4741d1448b4575.html>. Actualizado Abril de 2015.
- [30] © Oracle. Oracle java archive. <http://www.oracle.com/technetwork/java/archive-139210.html>. Consultado el 13 de Agosto del 2018.
- [31] © 2012-2018 Pi4J. Esquema de numeración de gpio para raspberry pi, según pi4j. <http://pi4j.com/pin-numbering-scheme.html>. Consultado el 8 de Agosto del 2018.
- [32] © 2012-2018 Pi4J. Package com.pi4j.io.gpio. <http://pi4j.com/apidocs/index.html>. Consultado el 20 de Agosto del 2018.

- [33] FerMax Professional. Domótica: ¿qué precio tiene hacer un hogar inteligente?. <http://blog.fermax.com/esp/dom%C3%B3tica-qu%C3%A9-precio-tiene-hacer-un-hogar-inteligente>. Actualizado el 2 de Marzo del 2017.
- [34] The project team Pi4J. Download. <http://pi4j.com/download.html>. Actualizado el 25 de Abril del 2018.
- [35] Real Academia Española (RAE). Domotico. <http://dle.rae.es/?id=E7W0v9b>. Consultado el 26 de Agosto del 2018.
- [36] Telegram. Bots: An introduction for developers. <https://core.telegram.org/bots>. Consultado el 22 de Agosto del 2018.
- [37] Telegram. Download telegram desktop. <https://desktop.telegram.org/>. Consultado el 22 de Agosto del 2018.
- [38] Telegram. Telegram apis. <https://core.telegram.org/api>. Consultado el 22 de Agosto del 2018.
- [39] Telegram. Why switch to telegram?. <https://telegram.org/>. Consultado el 22 de Agosto del 2018.
- [40] Luis Castillo Vidal. Profesor UGR. Comunicación agentes acl. <https://drive.google.com/open?id=152g00qSHH5XhPI63YzZm5Yao0jCnzEFF>. Consultado el 13 de Agosto del 2018.
- [41] Grupo Investigación Inteligencia Artificial Intelligence Research Group UPV Valencia. Librería magentix2-2.01-jar. <https://drive.google.com/file/d/0Bz9bcIKIxhnPRDFpTXduSjZhbm8/view>. Consultado el 13 de Agosto del 2018.
- [42] Grupo Investigación Inteligencia Artificial Intelligence Research Group UPV Valencia. Manual de usuario magentix2. <https://drive.google.com/file/d/0Bz9bcIKIxhnPZUZ1ampJNS1ISVU/view>. Consultado el 13 de Agosto del 2018.
- [43] Wikipedia. General-purpose input/output. https://en.wikipedia.org/wiki/General-purpose_input/output. Actualizado el 19 de Junio del 2018.
- [44] Wikipedia®. Domótica. <https://es.wikipedia.org/wiki/Dom%C3%B3tica>. Consultado el 26 de Agosto del 2018.
- [45] ©Williams and Wang LLC. How to use the l298n dual h-bridge motor driver. <https://www.bananarobotics.com/shop/How-to-use-the-L298N-Dual-H-Bridge-Motor-Driver>. Consultado el 12 de Agosto del 2018.

- [46] Proveedor SensorKit X40. Manual de conexiones. https://www.elektor.com/downloads/dl/file/id/1518/datasheet_sensor_kit_x40.pdf. Consultado el 9 de Agosto del 2018.

