Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación Android para la gestión de recursos domóticos usando el framework IoTivity

Autor: Carlos Fernández Rojas
Tutor: María Teresa Ariza Gómez

Dep. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2019

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Aplicación Android para la gestión de recursos domóticos usando el framework IoTivity

Autor:
Carlos Fernández Rojas

Tutor:
María Teresa Ariza Gómez
Profesor titular

Dep. Ingeniería Telemática
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2019

Proyecto Fin de Carrera: Aplicación Android para la gestión de recursos domóticos usando el framework IoTivity

Autor: Carlos Fernández Rojas

Tutor: María Teresa Ariza Gómez

El tribunal nombrado para juzgar el Proyecto arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2019

El Secretario del Tribunal

A mi familia

A mis profesores

A mis amigos

Agradecimientos

Durante el tiempo que he estado estudiando el Grado en Ingeniería de las Tecnologías de Telecomunicación, he aprendido multitud de cosas tanto a nivel personal, como académico. Esto va desde trabajar y convivir con otros muchos compañeros, aprender a sobrellevar tanto los aciertos como los errores cometidos, hasta aprender multitud de herramientas y conocimientos que me han abierto la mente. Ya que quizás no todo finalmente sea útil o lo uses en tu día a día. Pero ha ido moldeando una forma de razonar y actuar que estoy seguro, me será útil en un futuro.

No puedo tampoco dejar de pensar en la labor que ha desempeñado mi familia como sustento económico y sobre todo emocional. Estando ahí en los momentos malos para dar ánimos, orientarme en el buen camino cuando dudaba y los momentos buenos, que también han sido muchos, al alegrarse por mí cuando veían que estaba haciendo una carrera universitaria con mucha ilusión y que me gustaba (y me sigue gustando).

Tampoco puedo dejar de pensar en todos esos compañeros que, durante toda mi vida académica y no solo universitaria, me han acompañado. Ya que el compartir experiencias con gente que está viviendo prácticamente lo mismo que tú, ayuda mucho a hacer piña y tirar hacia adelante en los momentos difíciles. Con respecto a esto último, tengo que darle las gracias a un compañero en concreto que ha caminado a mi lado desde hace muchos años ya. Concretamente desde que decidimos recorrer "teleco" y del que espero crucemos caminos también en el ámbito profesional.

Ya para ir terminando, me gustaría agradecer a todos los profesores que me han dado clase durante estos años y especialmente a toda la rama de Telemática. Por enseñarme tanto su proyecto docente, como sus conocimientos nacidos de años de enseñanza y/o trabajo en el sector profesional. Lo cual me ha ayudado a aprender muchísimo más de lo que vienen en los libros o podría encontrar en internet.

Por último y no menos importante, me gustaría agradecer a mi tutora de este proyecto fin de grado, María Teresa Ariza Gómez, por todo el trabajo y apoyo prestado. Gracias por guiarme a la hora de buscar un tema para él y guiarme durante su desarrollo.

Carlos Fernández Rojas

Sevilla, 2019

Resumen

La tecnología siempre ha buscado hacer nuestro día a día más cómodo, incluso nos brinda nuevas formas de ocio para nuestro tiempo libre. Así durante esta última década, con el auge de las nuevas tecnologías como el teléfono inteligente y una mejora exponencial en las telecomunicaciones, hemos sufrido un nuevo avance tecnológico. Tanto es así que, por primera vez desde el auge de la informática en el siglo XX, ahora es cuando de verdad dicha revolución ha conseguido permear en la sociedad cambiando nuestro día a día y la forma que interaccionamos, entre nosotros y el mundo.

De esta forma, hemos pasado a estar permanentemente "conectados al mundo" gracias al desarrollo de las nuevas redes de operadoras como el 4G y 5G, que nos permiten tener internet en cualquier lugar. Y a que cualquier dispositivo pueda tener su propio sistema informático para su gestión y automatización. Recordemos que gracias a IPv6 cada objeto del mundo cotidiano podría tener su propia dirección de red hoy en día.

Aquí es donde entra la necesidad de establecer unas reglas de comunicación entre todos estos nuevos tipos de "actores" que se conectarán a la red, tanto local como internet. Ya que la idea es que podamos comunicarnos tanto con otras personas, como con nuestras herramientas cotidianas: ámbito de la domótica, salud o industrial. Para ello usaremos las especificaciones establecidas por la **Open Connectivity Foundation (OCF)** y más concretamente **The IoTivity project**, que busca establecer un punto de referencia para el rápido y cómodo desarrollo de las aplicaciones que utilizarán los dispositivos mencionados anteriormente.

Abstract

Technology has always sought to make our day to day more comfortable, it even gives us new forms of leisure for our free time. Thus during this last decade, with the rise of new technologies such as the smartphone and an exponential improvement in telecommunications, we have undergone a new technological advance. So much so that, for the first time since the rise of computer science in the twentieth century, it is now when this revolution has truly permeated society by changing our day to day and the way we interact, between ourselves and the world.

In this way, we have become permanently "connected to the world" thanks to the rise of new operator networks such as 4G and 5G, which allow us to have internet anywhere. And that any device can have its own computer system for its management and automation. Recall that thanks to IPv6 every object in the everyday world could have its own network address today.

This is where the need to establish communication rules between all these new types of "actors" that will connect to the network, both local and internet, comes in. Since the idea is that we can communicate with other people, as well as with our daily tools: home automation, health or industrial. For this we will use the specifications established by the Open Connectivity Foundation (OCF) and more specifically The IoTivity project, which seeks to establish a benchmark for the fast and convenient applications that in turn will use the devices mentioned above.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE ILUSTRACIONES

1 INTRODUCCIÓN

Las ciudades industriales que se hicieron en el siglo XX hay que reconvertirlas en ciudades de servicios, donde la gente no tenga que transportarse una o dos horas para llegar a otro lugar.

- Carlos Slim Helú -

Para empezar, en este primer capítulo de introducción, se expondrán las motivaciones y objetivos del proyecto. Así como una primera aproximación a qué es "el internet de las cosas", las soluciones que podemos encontrar actualmente en el mercado y como se estructurará el resto de la memoria.

1.1 MOTIVACIÓN

La temática del proyecto ha estado influenciada en gran medida por el auge de las aplicaciones para *smartphones* y la tendencia de que muchos dispositivos vienen con su propio *software* de gestión.

Otro motivo de los motivos es el hecho de haber cursado asignaturas relacionadas tanto con el desarrollo software (diseño, java, aplicaciones distribuidas, android) como electrónicas. Aportando cierto interés en seguir profundizados en dichas competencias, a la vez que se le da un fin práctico.

Así el trabajo busca crear una solución al ámbito de la domótica, haciendo uso de los conocimientos adquiridos y nuevos por adquirir, haciendo hincapié en las soluciones IoT (*internet of things*). Presentes en el mercado y aún en desarrollo.

1.2 OBJETIVOS

El proyecto tiene como meta desarrollar una aplicación Android para la gestión de dispositivos inteligentes, en nuestro caso del ámbito del hogar. Utilizando el *framework* de IoTivity y las referencias de la OCF (*Open Connectivity Foundation)*. Para ello utilizaremos una Raspberry y un Arduino como dispositivos IoT, aprovechando para profundizar en la parte servidor (del recurso IoT), además de la aplicación cliente mencionada.

De esta forma, tendremos una visión global de todo el sistema IoT para cada recurso prestado: cliente, comunicación y servidor. Además de profundizar en el uso de las especificaciones de la OCF mientras desarrollamos dos aplicaciones basadas en el código y librerías proporcionadas por IoTivity.

Por último, desde un punto de vista más práctico, buscaremos desarrollar una aplicación accesible para buscar, gestionar y controlar los diversos dispositivos de nuestro día a día.

1.3 ¿QUÉ ES "INTERNET DE LAS COSAS"?

Antes de nada, cabe aclarar que, aunque es un tema de actualidad, el concepto se remonta a inicios de los 80s cuando empresas y universidades americanas empezaron a trabajar con dispositivos embebidos en máquinas cotidianas [1]. Por ejemplo, hay documentado un proyecto de una máquina expendedora de Coca-Cola, situada en la universidad Carnegie Mellon (Pittsburgh, Pensilvania), donde los programadores locales monitorizaban las existencias y temperatura de los refrescos [2] [3].

Pero no fue hasta 1999 que Kevin Ashton, el director ejecutivo de "*Auto-ID Labs*", usó el termino de "*the Internet of Things*" durante una conferencia para "*Procter & Gamble*", multinacional estadounidense de bienes de consumo con sede en Cincinnati (Ohio) [3]. Fue allí donde el Sr. Ashton declaró:

"Los ordenadores actuales —y, por tanto, internet— son prácticamente dependientes de los seres humanos para recabar información (...) El problema es que las personas tienen un tiempo, una atención y una precisión limitados, y no se les da muy bien conseguir información sobre cosas en el mundo real. Y eso es un gran obstáculo (...) Si tuviéramos ordenadores que supieran todo lo que tuvieran que saber sobre las "cosas", mediante el uso de datos que ellos mismos pudieran recoger sin nuestra ayuda, nosotros podríamos monitorizar, contar y localizar todo a nuestro alrededor, de esta manera se reducirían increíblemente gastos, pérdidas y costes. Sabríamos cuándo reemplazar, reparar o recuperar lo que fuera, así como conocer si su funcionamiento estuviera siendo correcto. El internet de las cosas tiene el potencial para cambiar el mundo tal y como hizo la revolución digital hace unas décadas. Tal vez incluso hasta más." [3]

Kevin Ashton, 1999

Así llegamos a 2013, cuando las aplicaciones basadas en dicho concepto aumentaron y el termino de IoT ganó popularidad [1]. Actualmente se buscan nuevas formas de interconectar diferentes objetos del día a día, con el fin de mejorar la eficiencia:

Cualquier dispositivo puede ser interconectado con otro u otros dispositivos [3].

Cosa que permite desde programar una serie de rutinas domésticas de antemano, dotar a las ciudades de "inteligencia" para gestionar el tráfico y consumo energético o controlar ámbitos tan importantes como la salud de las personas [3].

1.4 SOLUCIONES PRESENTES EN EL MERCADO

Como se ha comentado anteriormente, el mercado de IoT está de actualidad y podemos encontrar diversas soluciones en el mercado. Así distinguimos dos tipos de propuestas según su enfoque comercial: de código abierto-libre y licenciadas. Y dos según cómo afrontan el problema: conexiones dispositivo-dispositivo o utilizando la "*cloud*" como núcleo de la plataforma.

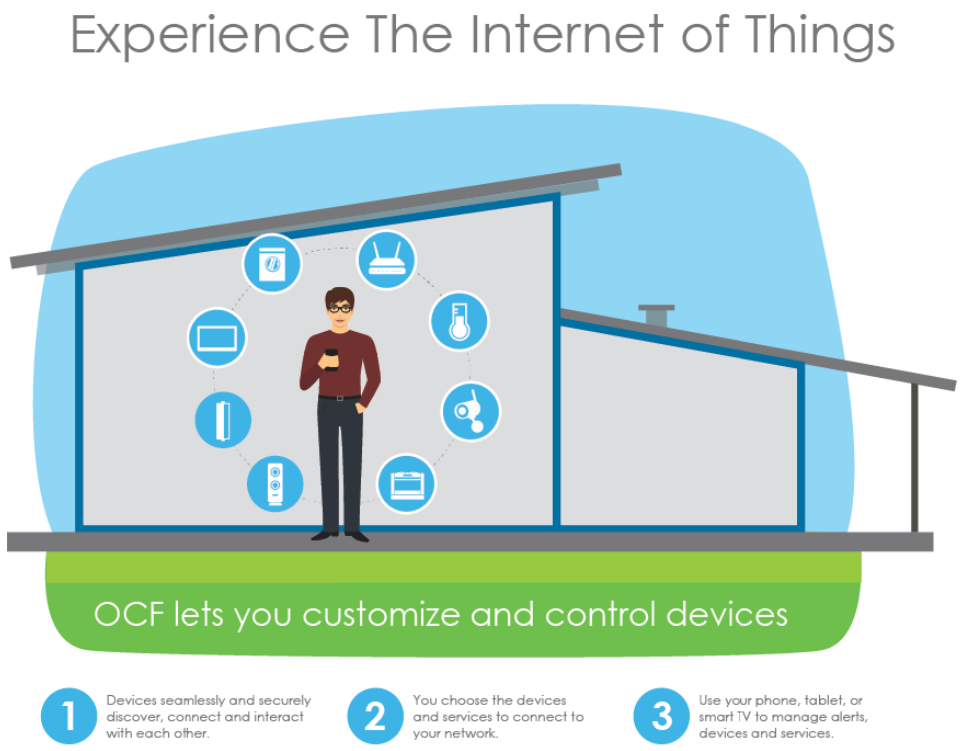


Ilustración 1. Solución Dispositivo-Dispositivo [35]

Ilustración 2. Solución con servicio en la nube [34]

De este modo, dejando de lado soluciones tanto para el ámbito empresarial: Cisco IoT y

Oracle IoT. Como diferentes empresas que venden dispositivos IoT de consumo doméstico, pero con su software propio. Y dejando también Google IoT que ofrece su propia solución al usuario/empresario, pero en la nube. Encontramos tres proyectos colaborativos de código abierto y libre:

Ilustración 3. Logo IoTivity

Ilustración 4. Logo Fiware

Ilustración 5. Logo OpenIoT

1.5 SOLUCIÓN PROPUESTA: IOTIVITY

En nuestro caso vamos a usar el framework de IoTivity, que a su vez está basado en la especificación OCF 1.3 [4]. Para montar el siguiente escenario, ilustración 6:

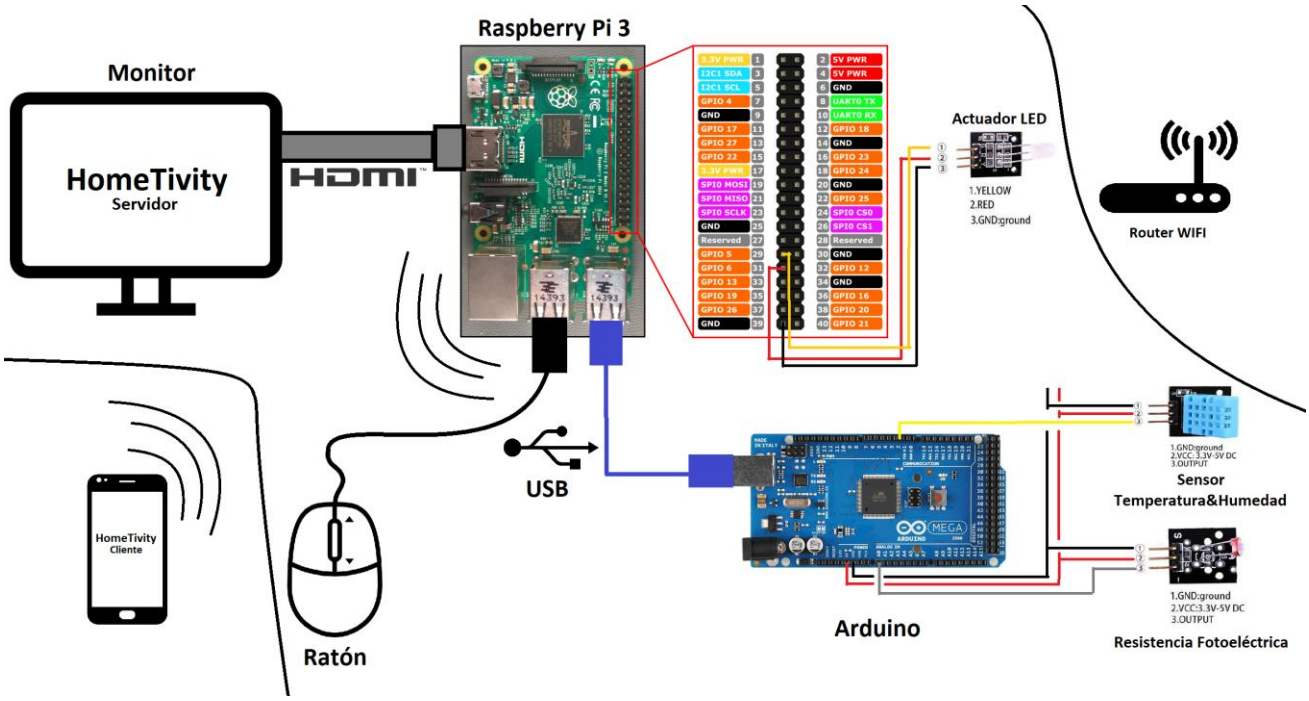


Ilustración 6. Escenario completo

En él estamos recreando un sistema servidor formado por los dispositivos de la zona central, que proporcionan los servicios de los recursos sensores, resistencias y actuadores. Así como un cliente formado por el terminal móvil de la parte inferior izquierda, que gestionará y monitorizará el escenario. A su vez todo estará dentro de la misma red local, representado por el router en la zona superior derecha.

Para esto último se desarrollarán dos aplicaciones: HomeTivityServidor y HomeTivityCliente. La primera (Ilustración 7) será la encargada de proporcionar el servicio actuando como servidor y gestionando el alta, baja y acceso a los recursos. La segunda será la aplicación cliente que gestionará de forma remota los recursos dados de alta en el área local donde se encuentre el servidor (Ilustración 8).

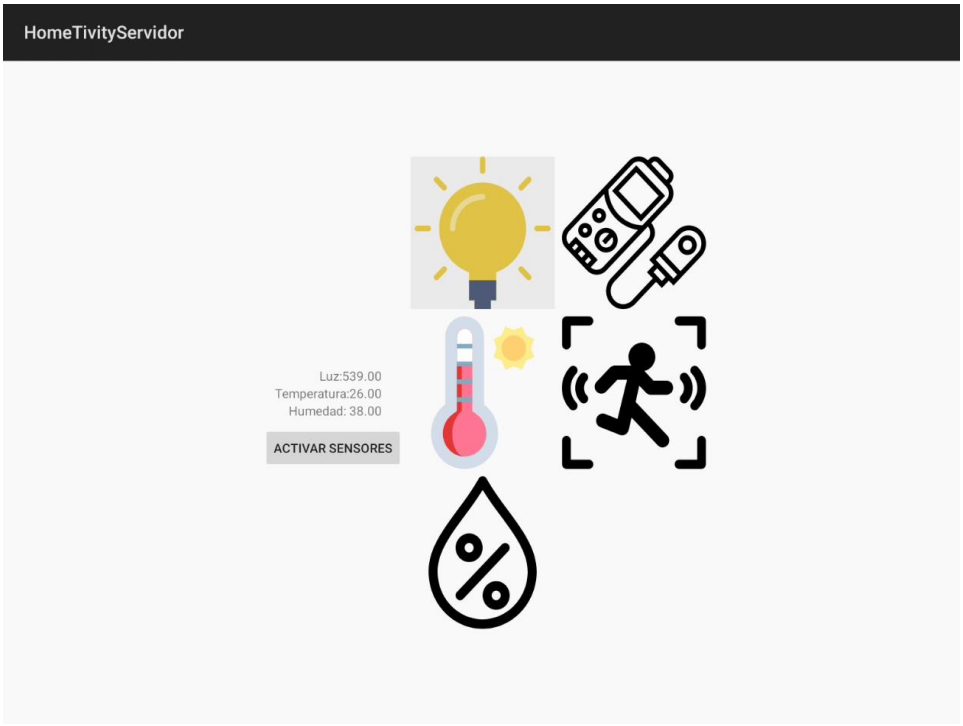


Ilustración 7. HomeTivityServidor Pantalla

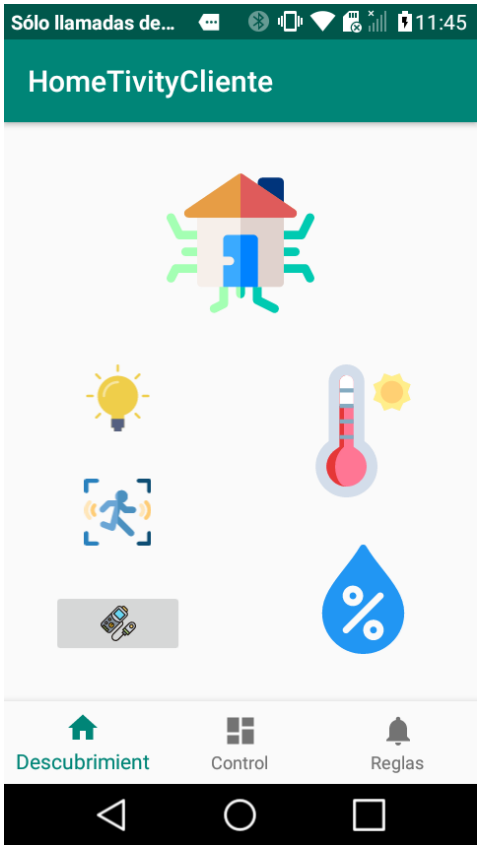


Ilustración 8. HomeTivityCliente Pantalla

1.6 ESTRUCTURA DE CAPÍTULOS Y ANEXOS

En los siguientes capítulos de la memoria se expondrá:

- **Capítulo 2: Tecnologías empleadas**

  Resumen de las tecnologías y herramientas software utilizadas, así como los dispositivos físicos y electrónicos usados para recrear el escenario del proyecto, o bien para el desarrollo del trabajo en términos generales.

- **Capítulo 3: IoTivity y OCF**

  Primera aproximación detallada al framework IoTivity y a las especificaciones de la OCF (Open Connectivity Foundation). Esto incluye: protocolos, requisitos de sistema, implementación de recursos/dispositivos y la comunicación entre dispositivos. Por último, también se detalla la puesta en marcha de un proyecto Android utilizando IoTivity.

- **Capítulo 4: Planificación temporal**

  Incluye el historial de versiones que se han ido desarrollando, así como los diferentes añadidos de cada versión. También se aporta una lista de tareas y subtareas con los tiempos de desarrollo como referencia del tiempo dedicado a cada aspecto o apartado del trabajo.

- **Capítulo 5: Especificación de requisitos**

  Dedicado a la ingeniería de requisitos, en este capítulo se realizará un análisis técnico del sistema para desarrollar las dos aplicaciones Android. Lo cual incluye: Identificación de los actores y procesos, desarrollo de los requisitos del sistema y los diagramas que detallan el funcionamiento y modelo de las aplicaciones.

- **Capítulo 6: Aplicación android – servidor**

  En él explicamos como realizar el montaje del servidor a nivel físico, se detalla el funcionamiento de la aplicación y funcionalidades implementadas, y por último se explica las partes más importantes del diseño y desarrollo del código de la aplicación.

- **Capítulo 7: Aplicación android – cliente**

  Se explica la ejecución de la aplicación y funcionalidades implementadas, además de detallar algunas partes del diseño y código que se estiman más importantes.

- **Capítulo 8: Pruebas y validación**

  Lista de pruebas a ejecutar para verificar que lo definido en la especificación de requisitos, se cumple correctamente.

- **Capítulo 9: Conclusiones**

  Último capítulo centrado en las conclusiones, competencias desarrolladas y planes de mejora del proyecto.

Terminamos con los anexos, tres de ellos son para explicar de manera detallada el montaje del escenario y un cuarto con los errores más problemáticos/interesantes encontrados y su solución.

- **ANEXO I: Raspberry pi 3 y AndroidThing**
- **ANEXO II: Arduino y Sensores**
- **ANEXO III: Montaje escenario**
- **ANEXO IV: Problemas/Soluciones**

2 TECNOLOGÍAS EMPLEADAS

El logro más impresionante de la industria del software
es su continua anulación de los constantes y
asombrosos logros de la industria del hardware.

- Henry Petroski -

En este capítulo vamos a enumeras las diferentes herramientas, dispositivos y tecnologías que se han utilizado. Ya sea para el diseño, desarrollo o creación de la memoria.

2.1 HARDWARE

Dispositivos físicos utilizados durante el desarrollo del trabajo.

2.1.1 Ordenador de sobremesa

-SO: Windows 10 Pro

-RAM: 24,0 GB

-Procesador: Intel® Core™ i7 a 3.60 GHz

-Periféricos: Pantalla LCD, ratón y teclado.

2.1.2 Arduino

-Marca: ELEGOO

-Modelo: Mega 2560

Ilustración 9. Arduino

2.1.3 37 Sensor Kit (ELEGOO)

Kit de sensores analógicos y digitales. Viene con librerías y tutoriales como extras para iniciarse en la programación sobre Arduino.

Ilustración 10. Kit sensores

2.1.4 Raspberry Pi 3

-Marca: Raspberry Pi Spain

-Modelo: Pi 3 Modelo B

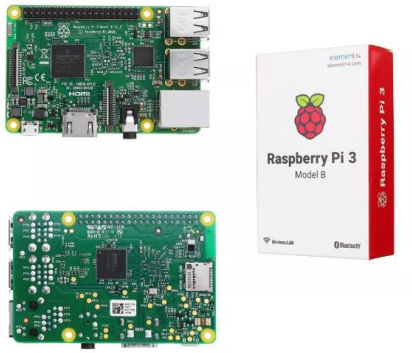


Ilustración 11- Raspberry pi 3

2.1.5 Terminal móvil: LG K4 2017

Procesador: MSM8909 1.1GHz Quad-core

RAM: 1GB

Memoria interna: 8 GB

Ilustración 12. Terminal móvil LG

2.2 SOFTWARE

Herramientas de desarrollo, tecnologías informáticas y utilidades varias.

2.2.1 Android Studio

Es el IDE (entorno de desarrollo integrado) oficial de Android. Está diseñado para facilitar el desarrollo de aplicaciones Android y derivados. Ofrece herramientas completas de edición, depuración, pruebas y perfilamientos de códigos.

Dicho *software* está basado en **IntelliJ IDEA** de JetBrains y publicado de forma gratuita a través de la **Licencia Apache 2.0**. Actualmente tiene versiones para Windows (32 y 64 bits), MAC (64-bits), Linux (64-bits) y Chrome OS.

Ilustración 13. Android Studio Logo

2.2.1.1 Android

Sistema operativo desarrollado por Google, basado en el Kernel de Linux y otros softwares de código abierto. Es el SO para móviles con más cuota de mercado del mundo, terminó rozando el 90% en países como España a finales de 2018. También se utiliza para tabletas, relojes inteligentes, automóviles y televisores.

Ilustración 14. Android Logo

2.2.1.2 Android Thing

Es la alternativa de Google para el mercado de IoT. Dicho sistema operativo está basado en Android y está orientado a la creación de aplicaciones para electrodomésticos y pequeños dispositivos que quieran estar conectados a internet.

A principios de 2019 Google ha decidido cambiar el enfoque, IoT, y centrarse en altavoces y pantallas inteligentes concretamente.

Ilustración 15. Android Thing Logo

2.2.2 Arduido /Genuino

Software de código abierto (IDE) para escribir código y subirlo a la placa Arduino. El entorno está escrito en Java y basado en **Processing** y otros softwares libres.

Tiene soporte para Windows, MAC y diferentes arquitecturas Linux.

Ilustración 16. Arduino IDE Logo

2.2.3 VMware Workstation 15 Player

Versión gratuita de VMWare, empresa filial de EMC Corporation y propiedad de Dell Inc, orientado a la virtualización de máquinas.

Dicho programa simula un sistema físico, con unas características determinadas, permitiendo la ejecución de software. Normalmente se utiliza para la ejecución simultánea de diferentes y/o varios SO, en un mismo hardware, permitiendo un mejor aprovechamiento de los recursos.

Ilustración 17. Vmware Logo

2.2.3.1 Ubuntu 64bits

Sistema operativo de código abierto, distribución de GNU/Linux, y basada en la arquitectura Debian. Dentro de las distribuciones Linux tiene una cuota aproximada del 50%.

Ilustración 18. Ubuntu Logo

2.2.3.2 OpenJDK

Versión libre de la plataforma de desarrollo Java. Que es constantemente mejorado por la comunidad.

Dicha comunidad tiene dos secciones: una pública y una privada si se ha aceptado la licencia JRL. De esta forma destacan dos roles. Como desarrollador utilizando la API o herramientas, y como contribuidor del código fuente.

Ilustración 19. OpenJDK Logo

2.2.3.3 SCons

Herramienta de código abierto para el compilado y/o instalación de software a través de scripts, hechos con Python.

Tiene como objetivo hacer más accesible la compilación de aplicaciones con múltiples dependencias, estando orientado a múltiples sistemas operativos.

Ilustración 20. Scons Logo

2.2.4 Wireshark

Es una herramienta para analizar el tráfico de red, se utiliza mucho para realizar análisis y solucionar problemas de software y protocolos.

Destaca frente su competencia por tener una interfaz gráfica y muchas opciones de organización y filtrado, las cuales hacen muy accesible su uso para el usuario.

Ilustración 21. Wireshark Logo

2.2.5 Magic Draw UML

Herramienta CASE, compatible con el estándar UML, diseñada para trabajar en equipo y con múltiples IDEs.

Una herramienta CASE (*Computer Aided Software Engineering*) es un conjunto de aplicaciones o programas informáticos destinadas al desarrollo software, a la vez que mejora la eficiencia en términos de costes (tiempo y dinero).

Ilustración 22. Magicdraw Logo

3 IoTivity y OCF

Para aportar ideas realmente interesantes y tecnologías en ciernes a una empresa para que pueda seguir innovando por años, se requiere una gran cantidad de disciplina.

- Steve jobs-

Capítulo en el cuál profundizaremos en IoTivity y la OCF (Open Connectivity Foundation) como solución de código abierto y libre, a la interconexión entre dispositivos. Para ello primero haremos una introducción a dicha tecnología, para después explicar como trabajador en ella: tanto la puesta en marcha como las funciones/características principales.

3.1 Descripción de la tecnología

IoTivity es un proyecto patrocinado por Open Connectivity Foundation (OCF), la cual se dedica a garantizar la interoperabilidad segura para consumidores, empresas e industrias ofreciendo: una plataforma de comunicación, especificaciones puente, un código abierto y un programa de certificación. Esto permite que se comuniquen diversidad de dispositivos sin importar las características hardware, software y de red de las que disponen. [5] [6]

Así IoTivity asume todas las funcionalidades obligatorias y opcionales con la OCF 1.3 para crear una implementación de referencia (código abierto). Sirviendo como punto de entrada para el desarrollo y certificado de producto OCF. [6]

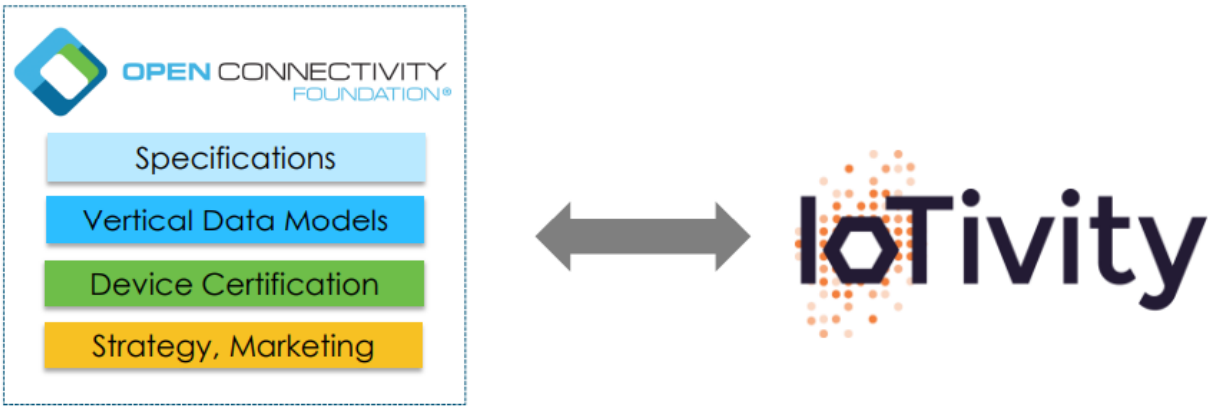


Ilustración 23. Colaboración OCF e IoTivity [5]

3.2 ESPECIFICACIONES Y PROTOCOLOS

Los objeticos de las especificaciones que propone la OCF, es la de permitir el desarrollo de aplicaciones (Vertical Profiles) a la vez que se mantiene interoperabilidad del resto. De esta forma, el núcleo del framework queda definido con elementos basados en casos de uso aprobados y normalizados. Entre estos casos de uso encontramos [5]:

1. Discovery. Método común para el descubrimiento de dispositivos.
2. Messaging. Soporte de dispositivo restringido como preterminado, así como la traducción del protocolo a través de puentes.
3. Common Resource Model. Entidades del mundo real definido como modelos de datos (recursos).
4. CRUDN. Peticiones simples de solicitud/respuesta: *Create*, *Retrieve*, *Update*, *Delete* y *Notify*.
5. ID & Addresing. OCF IDs y direccionamiento para entidades OCF: *Devices*, *Clients*, *Servers*, *Resources*.
6. Protocol Bridge/GW. Control de las especificaciones de puente con implicaciones en el núcleo.

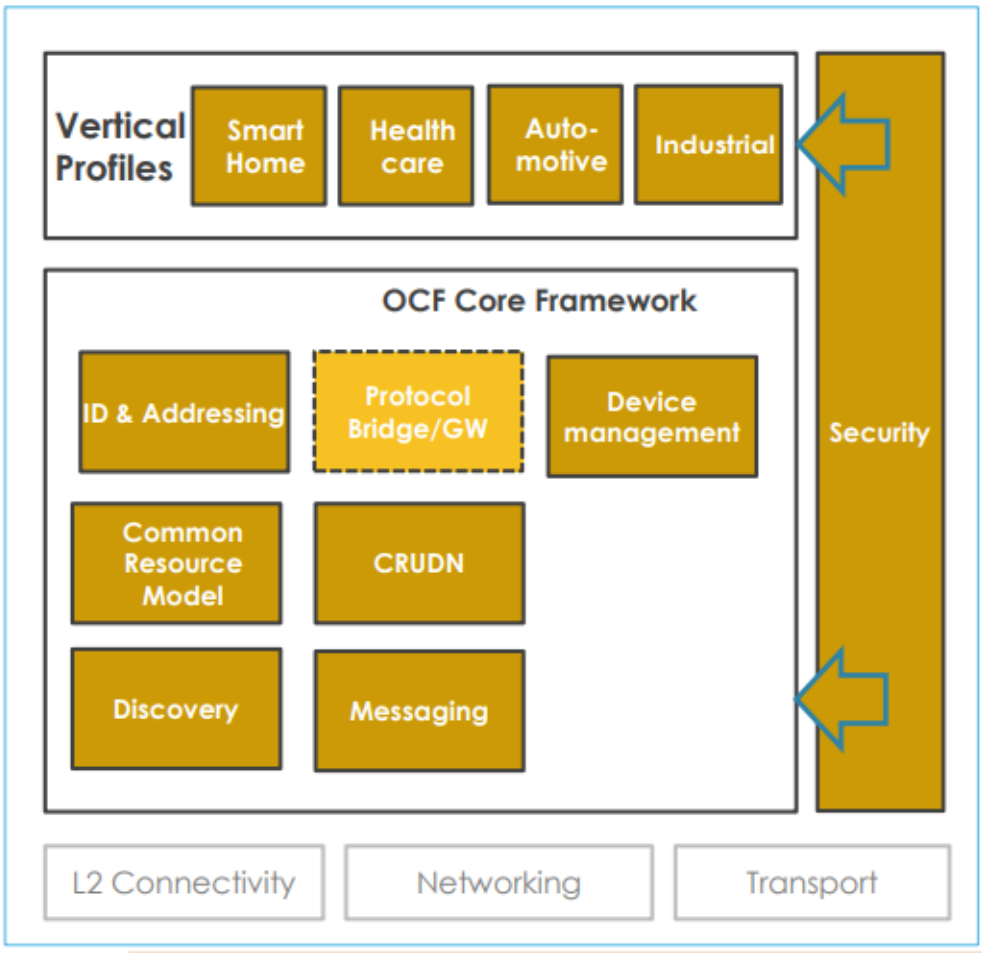


Ilustración 24. Torre de especificaciones OCF [5]

Esta arquitectura está diseñada para ser escalable, tanto para dispositivos limitados en recyrsosm, como ricos de ellos. Y para ser reutilizable con otros estándares abiertos, por ejemplo en nuestro caso IoTivity. La torre de protocolos OCF sería la mostrada en la ilustración 25

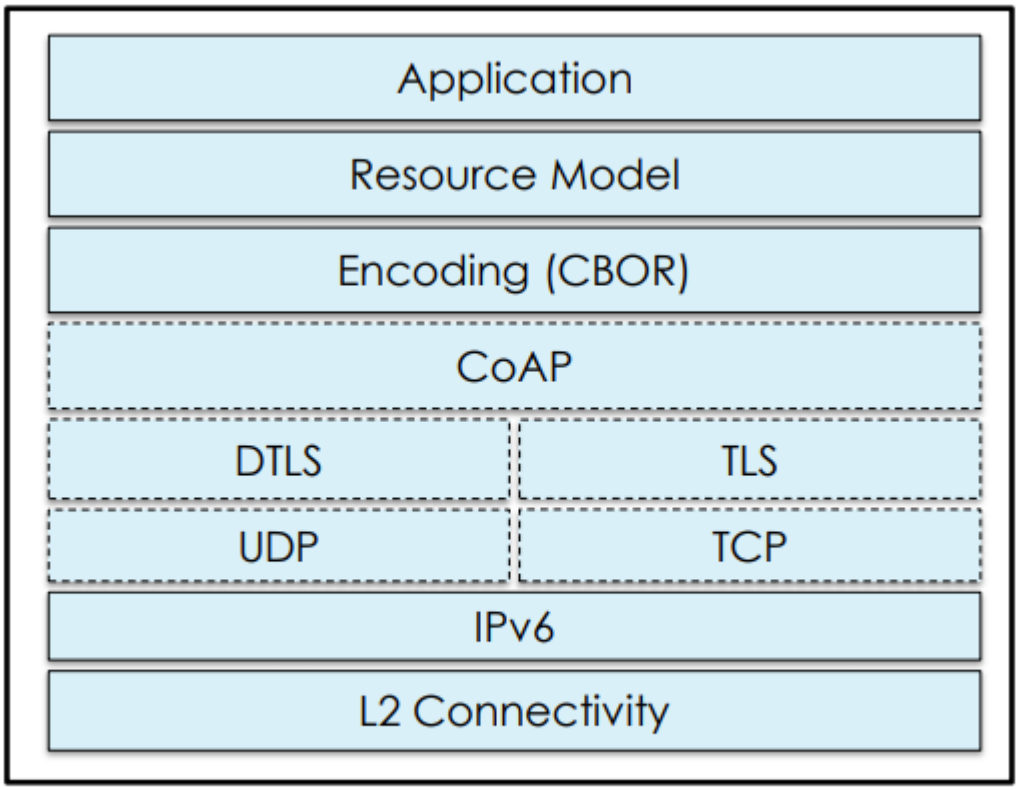


Ilustración 25. Torre de protocolos OCF [5]

3.3 RECURSOS Y DISPOSITIVOS

Los recursos sirven para describir entidades del mundo real. Así cada dispositivo que podemos encontrar, en los diferentes ámbitos ya descritos, estará formado por uno o más recursos. Estos recursos contienen interfaces que proporcionan un vistazo al resto de dispositivos y define como se comunican con ellos: lista de peticiones y respuestas soportadas [5].

A continuación, se proporciona un ejemplo de dispositivo "luz" que está formado por los recursos "conmutador binario" y "brillo" los cuales definen el objeto bombilla en el mundo real. En este caso vemos también que el primero es de carácter obligatorio, tiene que estar implementado en el dispositivo (mandatory), y el segundo es opcional (optional). El resto de recursos definen al dispositivo en el servidor que anuncia el recurso [5].

Device Title	Device Type	Associated Resource Type	M/O
Light	oic.d.light	oic/res (oic.wk.res)	M
		oic/p (oic.wk.p)	M
		oic/d (oic.d.light)	M
		Binary switch (oic.r.switch.binary)	M
		Brightness (oic.r.light.brightness)	O

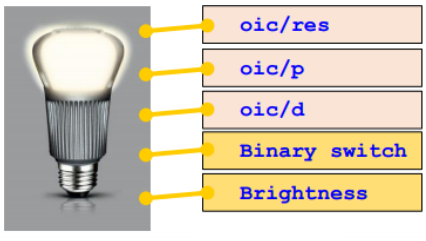


Ilustración 26. Ejemplo de dispositivo según OCF [5]

En las referencias proporcionadas por la OFC [7] [8] podemos encontrar una lista detallada tanto de recursos, como dispositivos y sus implementaciones, se detallan que tipo de variables tienen que tener dichos recursos.

3.4 ARQUITECTURA Y SECUENCIA DE COMUNICACIÓN

Los dispositivos OCF utilizan ***CoAP Discovery*** para el descubrimiento de otros dispositivos desconocidos, los cuales responden con una lista de enlaces; donde cada uno representa un recurso alojado en un dispositivo servidor. Entre los enlaces que proporciona podemos encontrar [5]:

- Enlace de referencia.
- Relación
- Puntos finales de enlaces.
- Interfaces compatibles
- Monitorización del recurso

Además, los dispositivos OCF (cliente-servidor) utilizan una arquitectura ***RESTful (Representational State Transfer)***. Esto significa que las operaciones se realizan mediante la manipulación de los propios recursos, es decir, tenemos una representación de la entidad real la cuál solicitamos y manipulamos con operaciones petición/respuesta **CRUDN** [5].

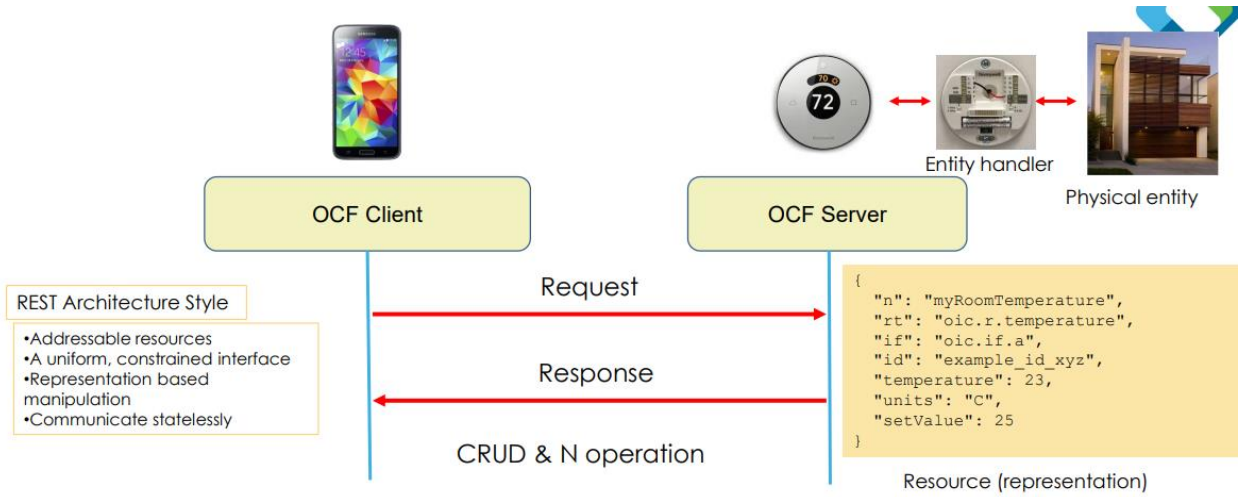


Ilustración 27. Ejemplo de comunicación según la OCF [5]

3.5 PUESTA EN MARCHA DE IOTIVITY

Antes que nada, se debe descargar "IoTivity Project" del repositorio Gerrit. Para ello debemos seguir los siguientes pasos desde un equipo Linux, en nuestro caso Ubuntu [10]:

1. Primero tenemos que configurar el acceso SSH al repositorio en cuestión. Para ello tendremos que crear una "llave SSH" con el comando:

```
$ ssh-keygen -t rsa -C "Your name <your_email_address>"
```

   Configurar la conexión editando `~/.ssh/config` con las líneas:

```
Host gerrit.iotivity.org

    Hostname "gerrit.iotivity.org"

    IdentityFile ~/.ssh/id_rsa

    User <Linux Foundation ID>

    Port 29418
```

   Ahora toca subirla a "IoTivity Gerrit", para ello nos registraremos en primer lugar. Seguiremos los siguientes pasos:

```
Settings (top right) --> SSH Public Keys --> Add Key...
```

   Para pegar la clave pública obtenida anteriormente id_rsa.pub en el directorio `~/.ssh` y pulsar 'Add'. Por último, podemos verificar que todo esté correctamente con el siguiente comando:

```
$ ssh gerrit.iotivity.org
```

   De forma que aparecerá el siguiente mensaje en caso de estar todo correcto:

```
****    Welcome to Gerrit Code Review    ****
```

2. El siguiente paso es configurar Git para el acceso de Gerrit. Para ello introduciremos los siguientes comandos con la información anterior:

```
$ git config --global user.name "John Doe"

$ git config --global user.email "john.doe@example.com"
```

3. Ahora ya podemos clonar el proyecto a nuestro git local introduciendo el siguiente comando:

```
$ git clone ssh://gerrit.iotivity.org/<Gerrit_Project>
```

Podemos ver la lista de proyectos con el siguiente comando:

```
$ ssh gerrit.iotivity.org gerrit ls-projects
```

En nuestro caso será:

```
$ git clone ssh://gerrit.iotivity.org/iotivity
```

Nota: para seguir trabajando con git en tu rama local, será necesario realizar el siguiente comando (en nuestro caso no trabajaremos sobre el proyecto):

```
$ cd iotivity

$ scp gerrit.iotivity.org:hooks/commit-msg .git/hooks/
```

Una vez que hemos hecho los preparativos previos y tenemos el proyecto en nuestro ordenador, es el momento de realizar la compilación correspondiente [9]. En nuestro caso para obtener las librerías que posteriormente usaremos en Android, previa importación en Android Studio.

1. Instalamos OpenJDK 1.7 o superior (si no lo tenemos). Para ello:

```
$ sudo apt-get update

$ sudo apt-get install openjdk-7-jdk

$ java -version
```

2. Instalamos SCons 2.4.0 o superior siguiendo las instrucciones, si no tenemos Python también tenemos que instalarlo:

```
$ sudo apt-get install python3

$ python -m pip install scons
```

Para más información sobre la instalación de scons consultar la página oficial [11]

3. Ahora ya podemos realizar la compilación para Android. Navegamos hasta el directorio raíz donde hemos descargado el proyecto IoTivity. El comando para ejecutar es:

```
$ scons TARGET_OS=android TARGET_ARCH=<target arch> TARGET_TRANSPORT=<target
transport> RELEASE=<release mode> SECURED=<secure> ANDROID_HOME=<path to android
SDK> ANDROID_NDK=<path to android NDK> ANDROID_GRADLE=<path to gradle/bin>
```

Donde tiene las siguientes opciones:

TARGET_ARCH	**Supported architectures:**
	x86 (default)
	x86_64
	armeabi
	armeabi-v7a
	arm64-v8a
TARGET_TRANSPORT	**Supported transport types:**
	ALL (default)
	IP
	BT
	BLE
	Note: You may also build permutations of the above options by passing a comma-separated list.
RELEASE	**Supported release modes:**
	1 (release mode) (default)
	0 (debug mode)
SECURED	**Supported security modes:** DTLS (Datagram Transport Layer Security)
	1 (DTLS enabled) (default)
	0 (DTLS disabled)
ANDROID_HOME	This takes the path to the Android SDK (if installed). If not, the SCons script will try to install Android SDK.
ANDROID_NDK	This takes the path to the Android NDK (if installed). If not, the SCons script will try to install Android NDK.
ANDROID_GRADLE	This takes the path to the Gradle bin (if installed). If not, the SCons script will install Gradle.

Tabla 1. Opciones de compilado para IOTIVITY

En nuestro caso:

```
$ scons TARGET_OS=Android TARGET_ARCH=armeabi SECURED=0
```

En caso de errores, visitar la bibliografía [9] con algunas soluciones de los problemas más comunes.

4. Ya tenemos tanto las librerías en formato '. arr' como las apk de ejemplo en los directorios:

```
<iotivity>/java/iotivity-android/build/outputs/aar/iotivity-base-<release
mode>.aar

<iotivity>/java/android-examples/<example name>/build/outputs/apk/<example
name>-<release mode>.apk
```

   Donde <iotivity> es el directorio raíz donde descargamos el proyecto.

Ahora sólo nos queda preparar el proyecto Android donde vamos a desarrollar la aplicación, en el sistema que elijamos (Windows 10 en nuestro caso).

1. Creamos un nuevo proyecto (API 21 como mínimo).

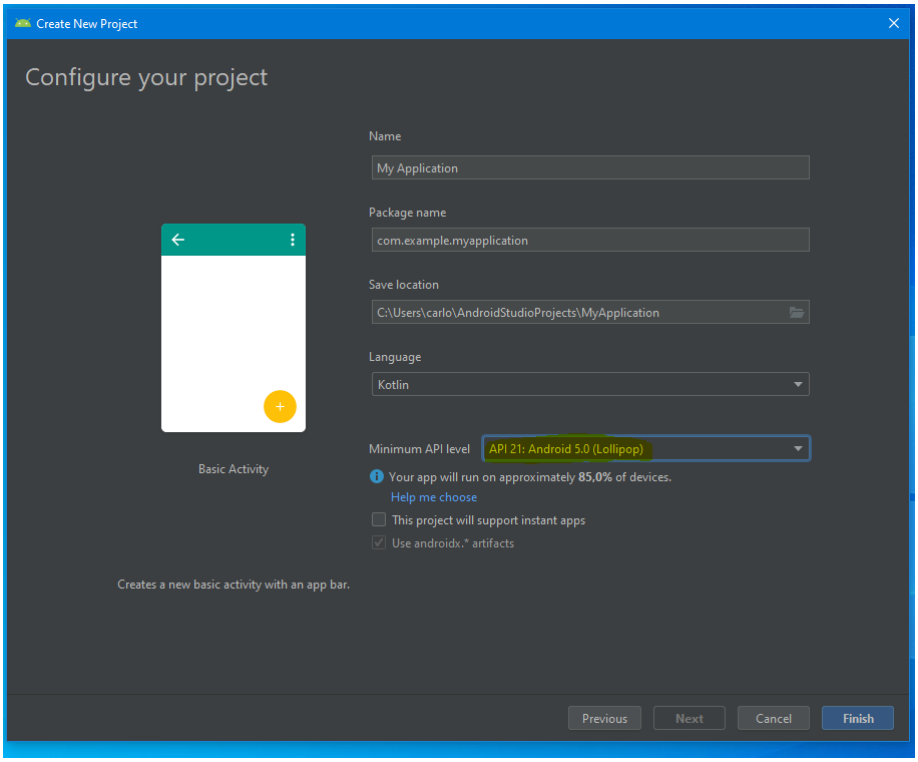


Ilustración 28. Crear nuevo proyecto Android Studio

2. Importamos la API de IoTivity

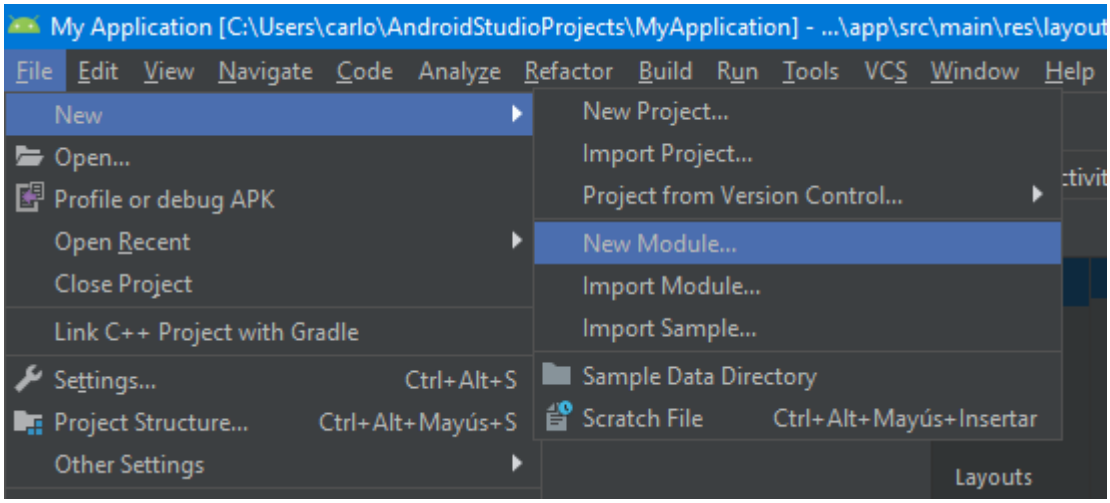


Ilustración 29. Nuevo módulo 01

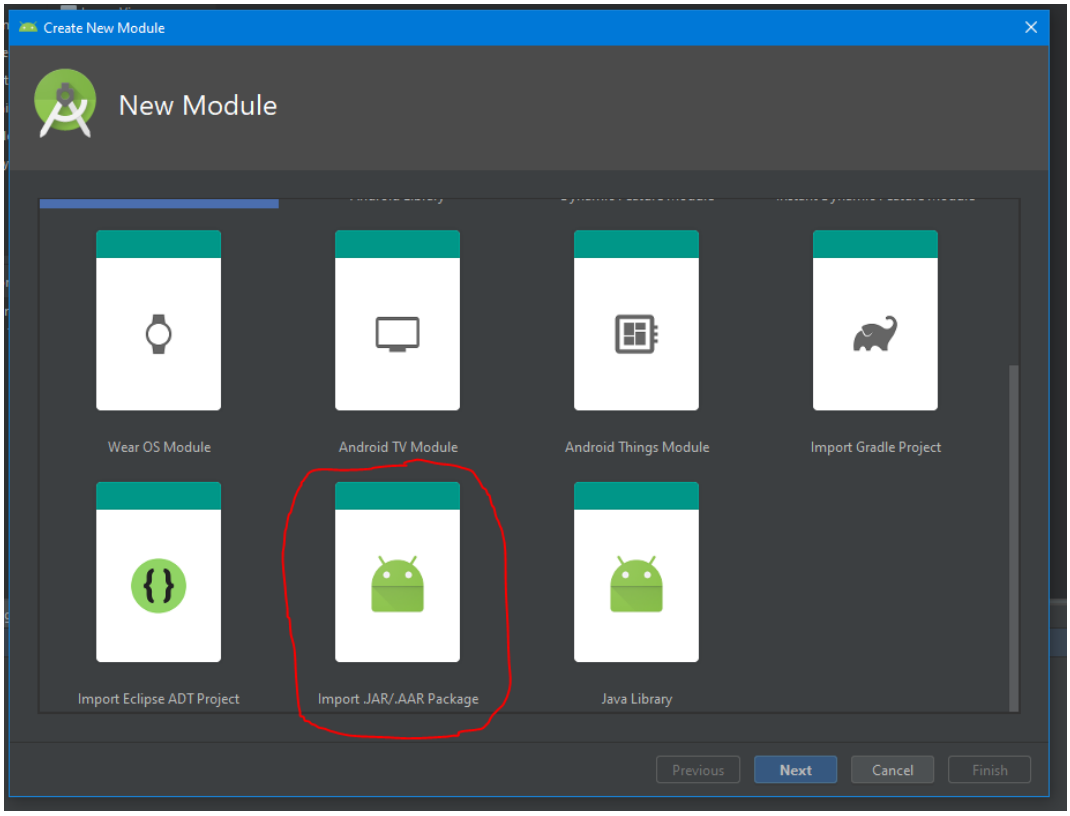


Ilustración 30. Nuevo Módulo 02

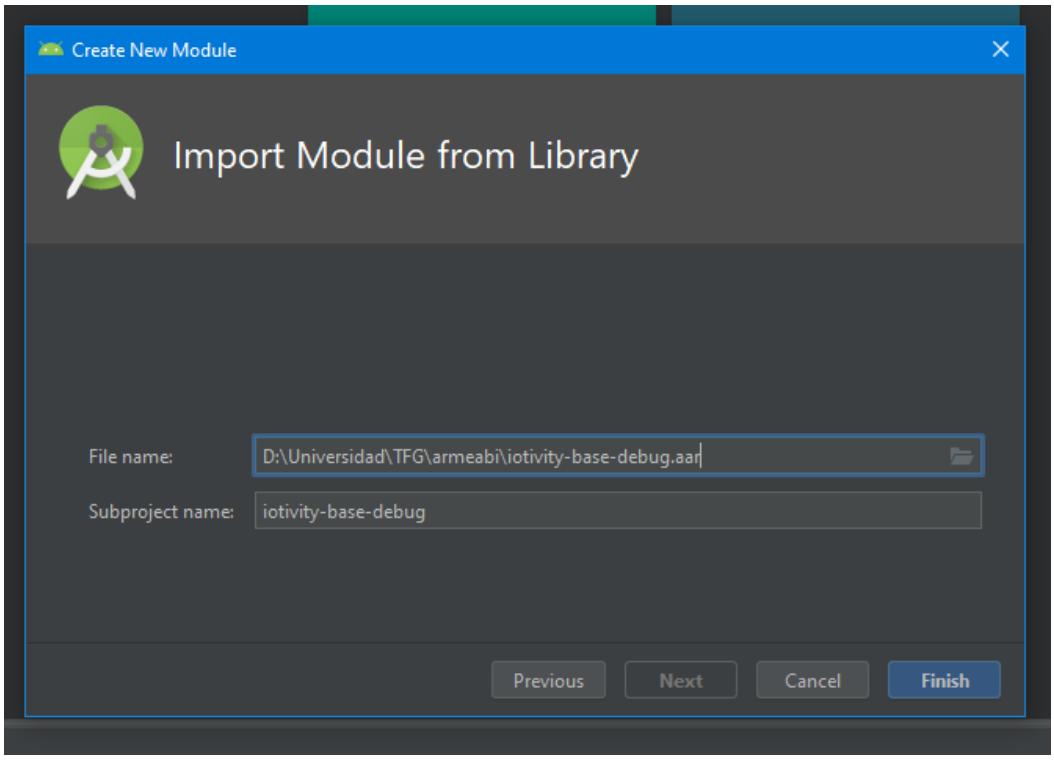


Ilustración 31. Nuevo Módulo 03

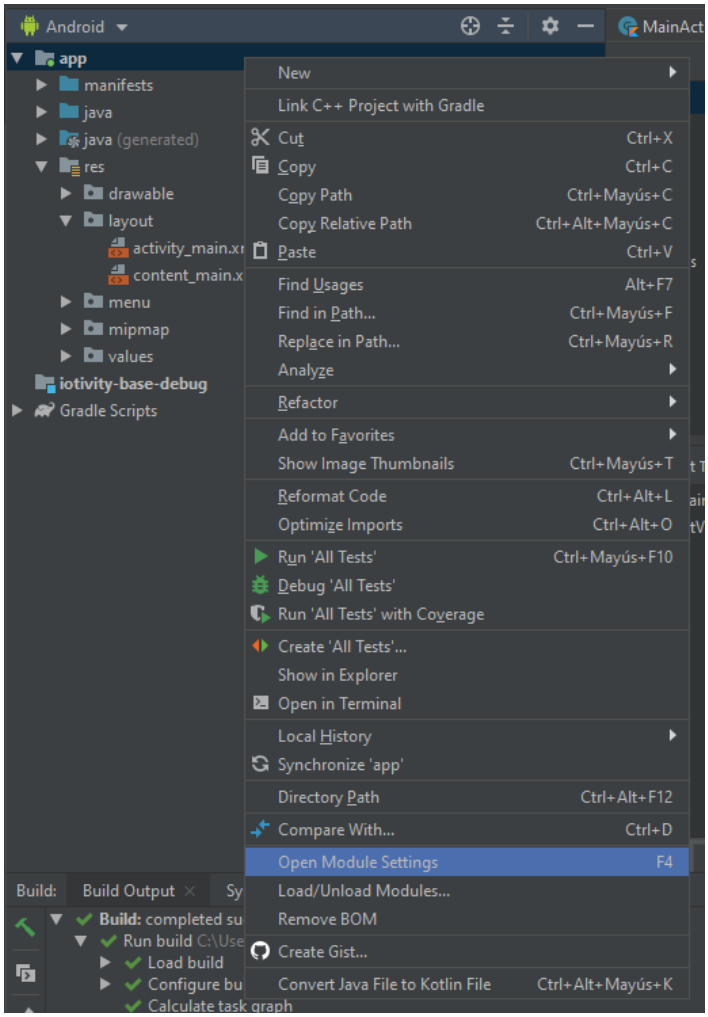


Ilustración 32. Añadir módulo a proyecto actual 01

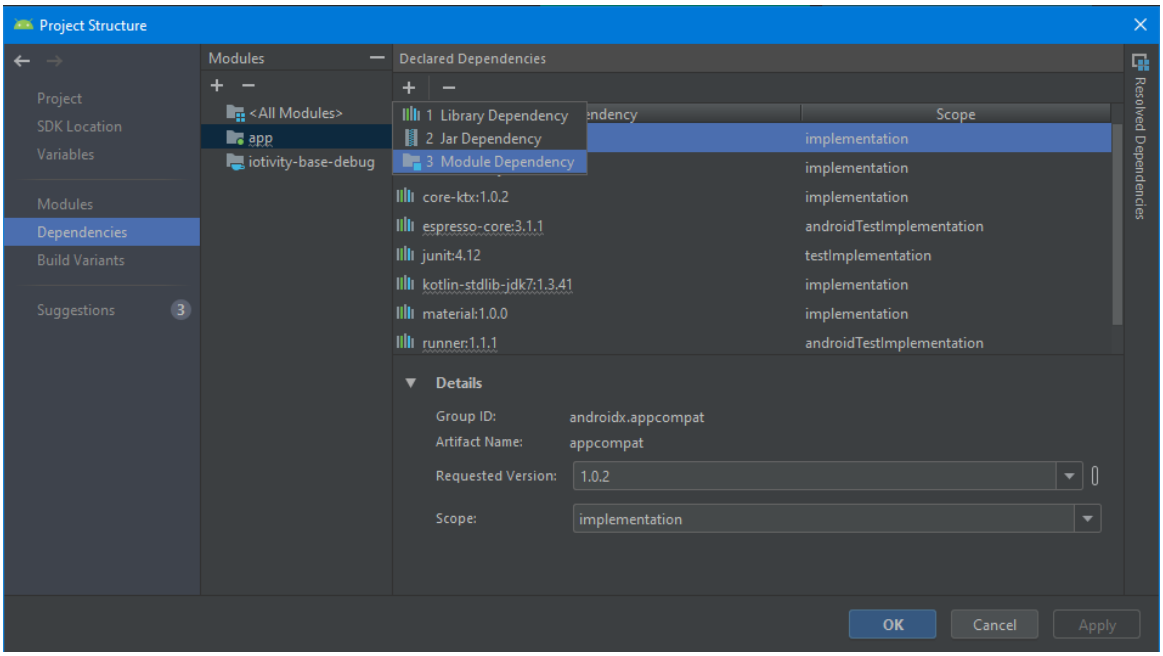


Ilustración 33. Añadir módulo a proyecto actual 02

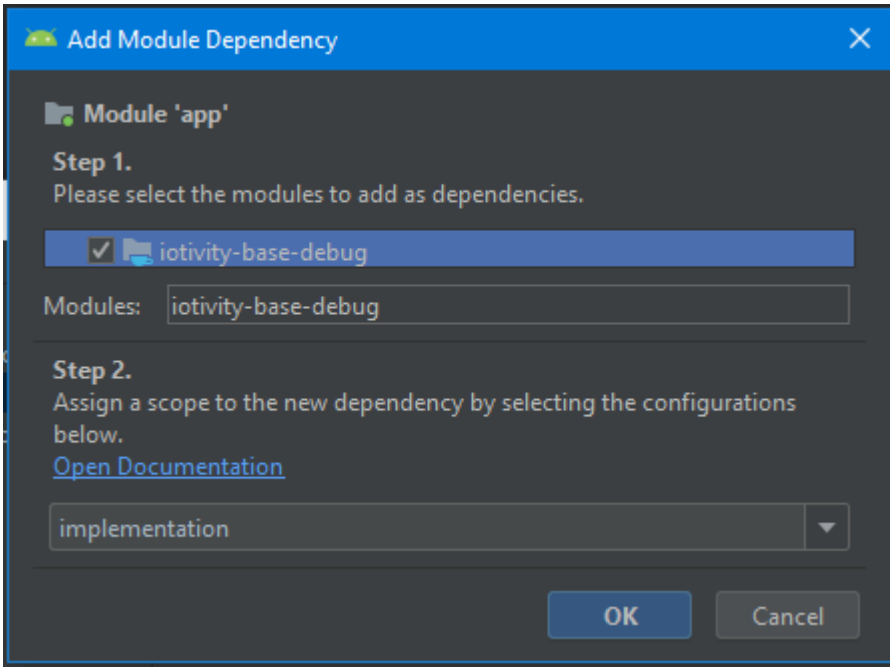


Ilustración 34. Añadir módulo a proyecto actual 03

4 PLANIFICACIÓN TEMPORAL

La persistencia es muy importante. No debes renunciar
al menos que te veas obligado a renunciar.

- Elon Musk -

A modo resumen, del proceso de desarrollo, en el siguiente capítulo se expone un historial de versiones y la documentación real de horas dedicadas a cada tarea. De esta forma queda reflejado cual ha sido el proceso y etapas por las que ha pasado el proyecto. También quiero aclarar que no se ha realizado una estimación temporal previo ya que se carecen de los conocimientos y experiencias necesarios para llevarla a cabo, sobre todo cuando está implicado una tecnología totalmente desconocida (a priori) para el ponente.

4.1 HISTORIAL DE VERSIONES

Tras realizar las primeras pruebas con las aplicaciones de ejemplo, obtenidas durante "la puesta en marcha", se optó por realizar una primera versión denomianda HomeTivity. Dicha versión incluía las mismas funciones que los ejemplos, pero con una interfaz y funcionalidades propias.

Durante esta versión el plano servidor, que proporciona el recurso iot, y el plano cliente, para gestionarlo, compartían aplicación. Por lo que el desarrollo estaba centrado en dos dispositivos móviles y la información que intercambiaban entre ellas.

Nombre	Fecha
HomeTivity1.0	26/02/2019
HomeTivity2.0	05/04/2019
HomeTivity2.5	05/04/2019
HomeTivity3.0	09/04/2019
HomeTivity3.5	25/04/2019

Tabla 2. Historial de versiones 01

Para la segunda versión de HomeTivity se separó en dos aplicaciones: el plano cliente (control, gestión y monitorización) y el plano servidor (recursos iot). El cliente seguiría ejecutándose en el dispositivo móvil y el servidos pasaría a la raspberry pi 3, con Android Thing instalado.

Nombre	Fecha	Añadidos
HomeTivityCliente01	25/04/2019	Nueva interfaz
HomeTivityCliente02	26/04/2019	Adaptación de la parte cliente, **HomeTivity3.5**, a la nueva interfaz.
HomeTivityCliente03	26/05/2019	Nuevos recursos IoT Notificaciones (sin configuración)
HomeTivityCliente04	15/07/2019	Implementación de notificaciones para todos los objetos y formulario dinámico para configurarlas.

Tabla 3. Historial de versiones 02

Nombre	Fecha	Añadidos
HomeTivityServidor01	08/05/2019	Interfaz para pruebas Cuatro recursos implementados, solo dos funcionales: lectura de temperatura y humedad
HomeTivityServidor02	10/05/2019	Lectura de cantidad de luz Comunicación con HomeTivity Cliente.
HomeTivityServidor03	26/05/2019	Nuevos recursos IoT Notificaciones para iluminación funcional
HomeTivityServidor04	15/07/2019	Notificaciones en casi (implementados en cliente) todos los objetos

Tabla 4. Historial de versiones 03

4.2 PLANIFICACIÓN TEMPORAL REAL

Como se ha comentado al principio del capítulo, a continuación, se recoge una lista de tareas realizadas durante el desarrollo del proyecto y las horas dedicadas a cada una. En la segunda tabla, se ve un conteo de las horas agrupado por versión.

Descripción de tarea	Descripción de subtarea	Horas dedicadas	Horas acumuladas/tarea
Investigación sobre IoT y Alljoyn		No documentadas	
Puesta en marcha IoTivity	Montar maquina virtual con ubuntu	1	
	Registro web Linux Foundation y Gerrit	1	2
Ejemplos de android usando Iotivity	Android Build Instructions	10	
	Hacer funcionar ejemplos (x86)	8	
	Hacer funcionar ejemplos (armeabi)	4	22
Rasberry Pi 3 + Android Thing	Puesta en marcha de Rasberrypi3 + Android Thing	3	
	Construcción de app para Android Thing usando IoTivity	5	8
Diseño de primera versión de HomeTivity	Interfaz y estructura	3	
	Programación primera versión	4	
	Funcionalidades	9	
	Ejemplo de RecyclerView	6	
	Filtrado de objetos	3	
	Actualización de compilado de librerias	3	
	Estudio de código y limpieza de funciones no utilizadas	3	
	Nuevas funciones servidor	4	
	Nuevas funciones cliente	3	
	Depuración y errores	8	46
Mejoras primera versión de HomeTivity	Descubrimiento	3	
	Filtrado de recursos	3	
	Mejora interfaz	3	
	Plano gestor implementación	8	
	Menu desplegable + formulario	3	
	Depuración errores	10	30
Nuevos recursos implementación	Documentación nuevos recursos	3	
	Creación+Implementación recurso temperatura	3	
	Actualizar cliente con nuevo recurso	7	13
Documentación de app		3	3
HomeTivity V2	Documentación y diseño	3	2
Raspberry+sensores	Documentación	4	
	servidor+sensor	9	13
Nueva versión cliente	Interfaz cliente	6	
	gestor interfaz	4	
	formularios	3	
	interfaz	4	
	funciones iotivity	4	
	errores y depuración	5	26
Nuevos recursos	Implementación app	9	
	Nuevos sensores	7	
	Solución de problemas y depuración	5	21
Notificaciones	Implementación app	8	7
Rediseño plano control	Diseño	3	
	Implementar reglas	7	
	Mejora interfaz	3	
	Notificaciones implementación	4	
	Arreglar errores	8	
	Lista seleccionable de favoritos	5	
	Depuración y mejoras finales	4	34
Memoria	Proyecto UML	30	
	Investigación, recopilación y redactado	100	130

Ilustración 35. Lista de tareas y horas dedicadas

Tiempo prototipos	
32	
Tiempo 1º versión	
92	
Tiempo 2º versión	
103	
Memoria y UML	Total
130	357

Ilustración 36. Resumen horas dedicadas

5 ESPECIFICACIÓN DE REQUISITOS

El diseño no es solo lo que se ve o lo que se siente.
Diseño es cómo funciona.

- Steve jobs-

Como estamos ante un proyecto software, es necesario realizar la especificación de requisitos para evaluar el sistema a desarrollar. De esta forma, en el capítulo actual, detallaremos los actores involucrados, comportamiento del sistema, objetivos de este y las limitaciones existentes.

5.1 IDENTIFICACIÓN DE LOS ACTORES

En las siguientes tablas se recogen los actores involucrados en los sistemas. He de señalar que como tenemos dos sistemas, para cada sistema, el otro sistema es un actor.

ACT001	Usuario de aplicación móvil
Versión	1.0
Descripción	Persona que utilizará la aplicación móvil y dará de alta los recursos IOT simulados.
Comentarios	interactúa con ambas aplicaciones

Tabla 5. ACT001: Usuario de aplicación móvil

ACT002	Aplicación móvil (Android)
Versión	1.0
Descripción	Dispositivo de gestión y control de los recursos IOT
Comentarios	Cuando miramos desde la perspectiva de la aplicación para Android Thing

Tabla 6. ACT002: Aplicación móvil

ACT003	Rasberry Pi 3 (Android Thing)
Versión	1.0
Descripción	Dispositivo que simulará varios recursos IOT
Comentarios	Cuando miramos desde la perspectiva de la aplicación para Android (móvil)

Tabla 7. ACT003: Rasberry Pi 3 (Android Thing)

AC004	Elegoo MEGA2560
Versión	1.0
Descripción	Dispositivo encargado de leer los sensores analógicos.
Comentarios	Cuando miramos desde la perspectiva de la aplicación para Android Thing

Tabla 8. AC004: Elegoo MEGA2560

5.2 PROCESOS

Recopila las diferentes funciones y formas de interacción que tiene el sistema con los actores.

PROC001	Aceptar permisos
Versión	1.0
Dependencias	Ninguna
Descripción	Confirmación por parte del usuario de los permisos necesarios para el funcionamiento de la aplicación.
Actores	ACT001

Tabla 9 . PROC001: Aceptar permisos

PROC002	Dar de alta recurso
Versión	1.0
Dependencias	Ninguna
Descripción	Crea un recurso iot y lo da de alta en la red local mediante el framework IoTivity.
Actores	ACT001

Tabla 10. PROC002: Dar de alta recurso

PROC003	Dar de baja recurso
Versión	1.0
Dependencias	Ninguna
Descripción	Da de baja un recurso iot del framework IoTivity.
Actores	ACT001: Usuario Aplicación móvil

Tabla 11. PROC003: Dar de baja recurso

PROC004	Leer sensores
Versión	1.0
Dependencias	Ninguna
Descripción	Proceso en bucle que lee por el puerto serie, USB, los datos que lee el Arduino de los sensores analógicos.
Actores	ACT004: Elegoo MEGA2560

Tabla 12. PROC004: Leer sensores

PROC005	Actualizar valores sensores
Versión	1.0
Dependencias	PROC004: Leer sensores
Descripción	Método para actualizar los valores que tienen los recursos dados de alta según los valores actuales leídos.
Actores	Proceso interno

Tabla 13. PROC005: Actualizar valores sensores

PROC006	Monitorización
Versión	1.0
Dependencias	PROC005: Actualizar valores sensores PROC008: SEND RESPONSE
Descripción	Proceso paralelo que controla los recursos y crea eventos si se cumplen ciertas condiciones en los recursos.
Actores	Proceso interno

Tabla 14. PROC006: Monitorización

PROC007	Manejador REQUEST
Versión	1.0
Dependencias	PROC005: Actualizar valores sensores PROC008: SEND RESPONSE PROC009: DISCOVERY REQUEST PROC010: PUT REQUEST PROC011: GET REQUEST PROC012: REGISTER OBSERVER PROC013: UNREGISTER OBSERVER
Descripción	Conjunto de métodos para tratar las peticiones del cliente y ejecutar las acciones adecuadas.
Actores	Proceso interno

Tabla 15. PROC007: Manejador REQUEST

PROC008	SEND RESPONSE
Versión	1.0
Dependencias	Ninguna
Descripción	Método encargado de “montar” la respuesta y enviarla de vuelta al cliente.
Actores	ACT002: Aplicación móvil (Android)

Tabla 16. PROC008: SEND RESPONSE

PROC009	DISCOVERY REQUEST
Versión	1.0
Dependencias	Ninguna
Descripción	Conjunto de métodos que permite a los usuarios de la red local listar los recursos iot que se encuentran en ella dados de alta.
Actores	ACT002: Aplicación móvil (Android) ACT003: Rasberry Pi 3 (Android Thing)

Tabla 17. PROC009: DISCOVERY REQUEST

PROC010	PUT REQUEST
Versión	1.0
Dependencias	Ninguna
Descripción	Conjunto de métodos que permiten modificar valores de los recursos iot.
Actores	ACT002: Aplicación móvil (Android) ACT003: Rasberry Pi 3 (Android Thing)

Tabla 18. PROC010: PUT REQUEST

PROC011	GET REQUEST
Versión	1.0
Dependencias	Ninguna
Descripción	Conjunto de métodos que permiten obtener valores de los recursos iot.
Actores	ACT002: Aplicación móvil (Android) ACT003: Rasberry Pi 3 (Android Thing)

Tabla 19. PROC011: GET REQUEST

PROC012	REGISTER OBSERVER
Versión	1.0
Dependencias	Ninguna
Descripción	Conjunto de métodos que permite al usuario suscribirse a una lista de observadores, la cual se notificará cuando se cumplen ciertos eventos en los recursos iot.
Actores	ACT002: Aplicación móvil (Android) ACT003: Rasberry Pi 3 (Android Thing)

Tabla 20. PROC012: REGISTER OBSERVER

PROC013	UNREGISTER OBSERVER
Versión	1.0
Dependencias	Ninguna
Descripción	Conjunto de métodos que permite al usuario de suscribirse a una lista de observadores, la cual se notificará cuando se cumplen ciertos eventos en los recursos iot.
Actores	ACT002: Aplicación móvil (Android) ACT003: Rasberry Pi 3 (Android Thing)

Tabla 21. PROC013: UNREGISTER OBSERVER

PROC014	REQUEST COMPLETED
Versión	1.0
Dependencias	Ninguna
Descripción	Conjunto de métodos para tratar las diferentes respuestas generadas en el recurso iot.
Actores	ACT003: Rasberry Pi 3 (Android Thing)

Tabla 22. PROC014: REQUEST COMPLETED

PROC015	Descubrir recursos
Versión	1.0
Dependencias	PROC009: DISCOVERY REQUEST
Descripción	Método por el cual el usuario puede buscar diferentes tipos de recursos iot en la red local.
Actores	ACT003: Rasberry Pi 3 (Android Thing)

Tabla 23. PROC015: Descubrir recursos

PROC016	Actualizar valores recurso
Versión	1.0
Dependencias	PROC010: GET REQUEST
Descripción	Método por el cual el usuario puede actualizar los valores locales de los recursos almacenados. Según los valores más actuales del dispositivo iot.
Actores	ACT003: Rasberry Pi 3 (Android Thing)

Tabla 24. PROC016: Actualizar valores recurso

PROC017	Modificar valores recurso
Versión	1.0
Dependencias	Método por el cual el usuario puede modificar o interactuar con los recursos iot.
Descripción	PROC011: PUT REQUEST
Actores	ACT003: Rasberry Pi 3 (Android Thing)

Tabla 25. PROC017: Modificar valores recurso

PROC018	Añadir favorito
Versión	1.0
Dependencias	Ninguna
Descripción	Acción que nos permite añadir un recurso, tras realizar una búsqueda, a la lista de favoritos.
Actores	ACT003: Rasberry Pi 3 (Android Thing)

Tabla 26. PROC018: Añadir favorito

PROC019	Quitar Favorito
Versión	1.0
Dependencias	Ninguna
Descripción	Acción que nos permite quitar un recurso de la lista de favoritos.
Actores	ACT003: Rasberry Pi 3 (Android Thing)

Tabla 27. PROC019: Quitar Favorito

PROC020	Añadir regla
Versión	1.0
Dependencias	PROC009: DISCOVERY REQUEST PROC010: GET REQUEST PROC011: PUT REQUEST PROC012: REGISTER OBSERVER
Descripción	Conjunto de métodos que nos permite: crear una regla, configurarla y unirnos a la lista de observadores de ducho recurso iot.
Actores	ACT003: Rasberry Pi 3 (Android Thing)

Tabla 28. PROC020: Añadir regla

PROC021	Quitar regla
Versión	1.0
Dependencias	PROC013: UNREGISTER OBSERVER
Descripción	Acción que nos permite eliminar una regla creada anteriormente y de suscribirnos de la lista de observadores.
Actores	ACT003: Rasberry Pi 3 (Android Thing)

Tabla 29. PROC021: Quitar regla

5.3 DIAGRAMA CASOS DE USO

Combinación gráfica de los dos puntos anteriores: Actores y procesos. Un diagrama por sistema.

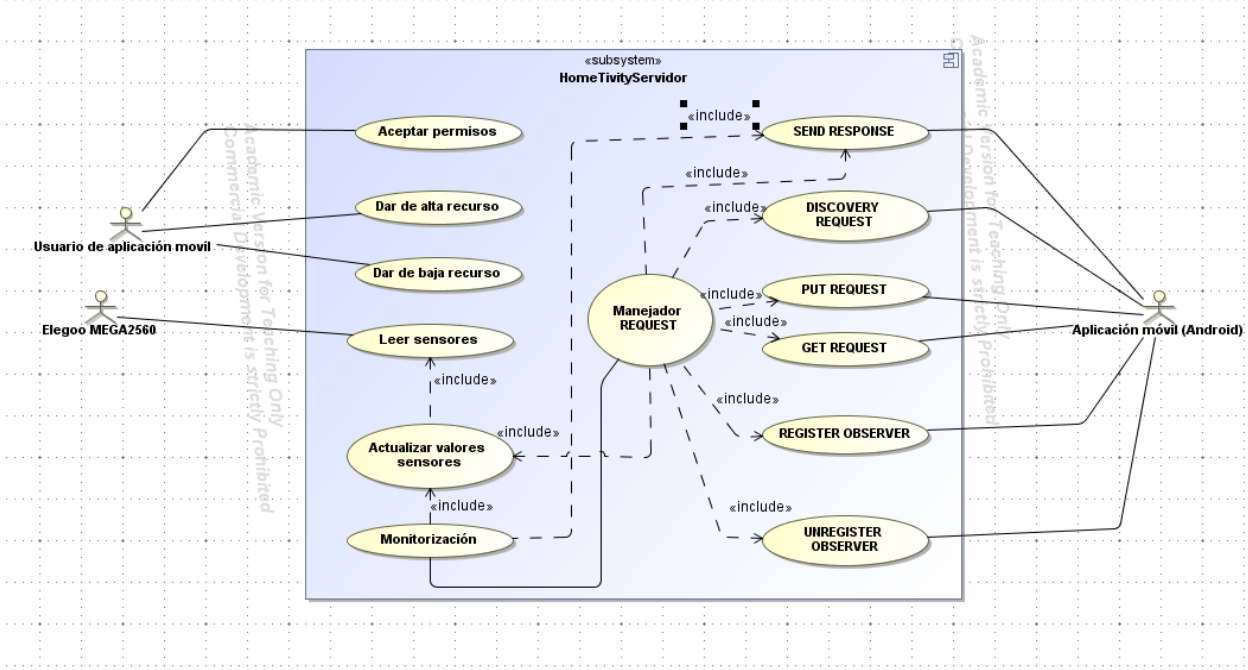


Ilustración 37. Diagrama caso de uso: HomeTivityServidor (subsistema)

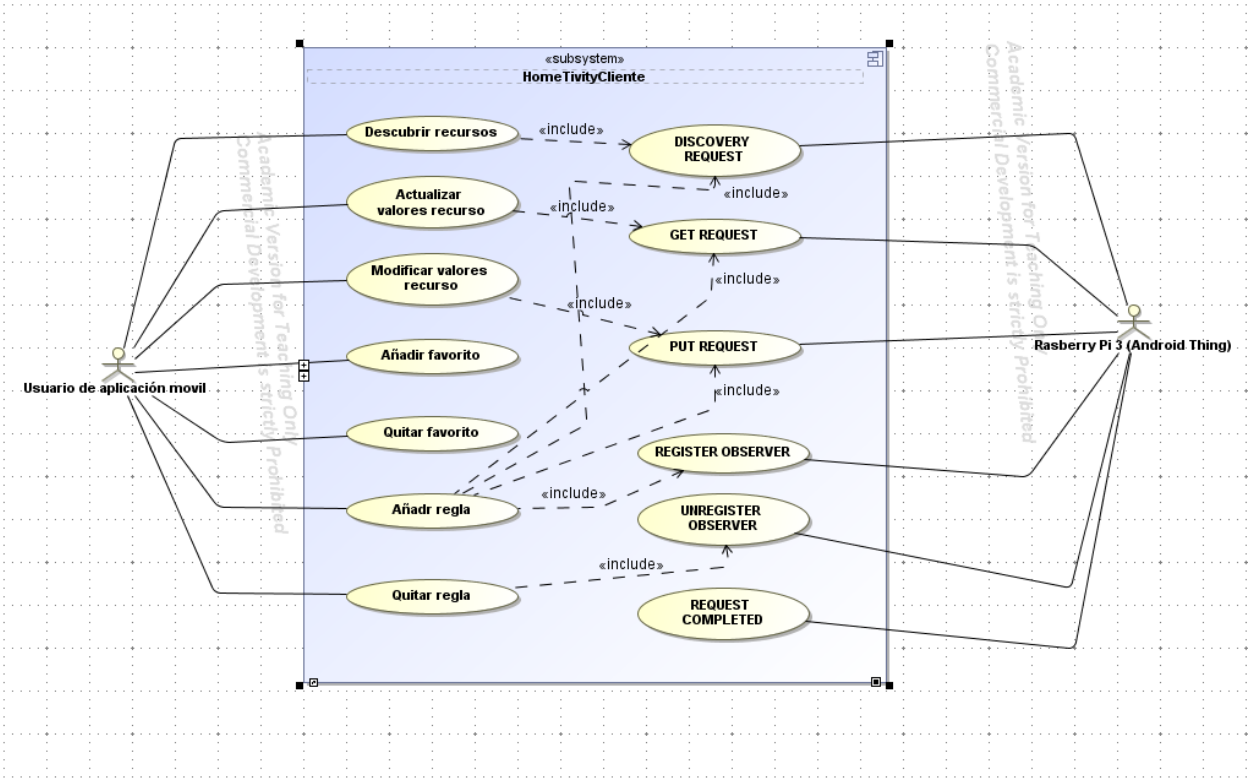


Ilustración 38. Diagrama caso de uso: HomeTivityCliente (subsistema)

5.4 REQUISITOS GENERALES

Características y objetivos que deben de cumplir los sistemas.

REQ001	Anunciar dispositivos
Versión	1.0
Descripción	El dispositivo y los recursos que contiene deben de poder ser visibles y accesibles en la red.
Prioridad	Esencial

Tabla 30. REQ001- Anunciar dispositivos

REQ002	Detectar dispositivos
Versión	1.0
Descripción	Un cliente debe de poder encontrar, mediante descubrimiento, los recursos buscando su identificador.
Prioridad	Esencial

Tabla 31. REQ002- Detectar dispositivos

REQ003	Lectura de sensores y actuadores
Versión	1.0
Descripción	El servidor debe de poder leer y modificar los sensores y actuadores a los que está conectado.
Prioridad	Esencial

Tabla 32. REQ003- Lectura de sensores y actuadores

REQ004	Controlar y gestionar recursos a distancia
Versión	1.0
Descripción	Un cliente debe poder acceder a un recurso de forma remota, previamente estableciendo una conexión con el servidor que los recoge.
Prioridad	Esencial

Tabla 33. REQ004- Controlar y gestionar recursos a distancia

REQ005	Monitorizar recursos
Versión	1.0

Descripción	Se debe poder establecer una serie de alertas para monitorizar los recursos, de forma que los clientes sean avisados ante determinados eventos (previamente configurados).
Prioridad	Esencial

Tabla 34. REQ005- Monitorizar recursos

REQ006	Interfaz gráfica
Versión	1.0
Descripción	Las interfaces de usuario deben de ser accesibles y garantizar el funcionamiento de todas funciones,
Prioridad	Esencial

Tabla 35. REQ006- Interfaz gráfica

REQ007	Almacenamiento de la configuración y estados de los recursos
Versión	1.0
Descripción	Las aplicaciones deben de tener almacenamiento de la información que contiene para que, en futuras ejecuciones, la configuración y estado anteriores se mantenga.
Prioridad	Esencial

Tabla 36. REQ007- Almacenamiento de la configuración y estados de los recursos

5.4.1 Funcionales

Los requisitos funcionales describen como debe de comportarse el sistema respecto al conjunto de entradas, comportamiento interno y salidas.

REQF001	Ficheros locales servidor
Versión	1.0
Tipo	Información
Descripción	La aplicación almacena toda la configuración y estados de los recursos. En un JavaScript de manera local para su posterior recuperación.
Prioridad	Esencial

Tabla 37. REQF001- Ficheros locales servidor

REQF002	Ficheros locales cliente
Versión	1.0
Tipo	Información
Descripción	La aplicación almacena toda la configuración, estados de los recursos e información de usuario. En un JavaScript de manera local para su posterior recuperación.
Prioridad	Esencial

Tabla 38. REQF002- Ficheros locales cliente

REQF003	Información recursos – subsistema servidor
Versión	1.0
Tipo	Información
Descripción	Los recursos IOT tendrán una serie de valores generales y otros propios. Resumen: • Una cadena de texto por variable • Una variable por elemento del recurso • URI recurso • Tipo recurso • Tipo interfaz • Manejador del recurso
Prioridad	Esencial

Tabla 39. REQF003- Información recursos – subsistema servidor

REQF004	Información recursos – subsistema cliente
Versión	1.0
Tipo	Información
Descripción	Los recursos IOT tendrán una serie de valores generales y otros propios. Resumen: • Una cadena de texto por variable • Una variable por elemento del recurso
Prioridad	Esencial

Tabla 40. REQF004- Información recursos – subsistema cliente

REQF005	La aplicación móvil podrá gestionar varios recursos y de diverso tipo.
Versión	1.0
Tipo	Funciones
Descripción	No importa el tipo de recursos o cantidad de ellos, siempre que esté dentro de los recursos implementados en la aplicación móvil.
Prioridad	Esencial

Tabla 41. REQF005- La aplicación móvil podrá gestionar varios recursos y de diverso tipo.

REQF006	La aplicación móvil podrá gestionar varios dispositivos y de diverso tipo.
Versión	1.0
Tipo	Funciones
Descripción	No importa el tipo de dispositivo donde están los recursos. Aunque es necesario que implementen el mismo framework: IoTivity.
Prioridad	Esencial

Tabla 42. REQF006- La aplicación móvil podrá gestionar varios dispositivos y de diverso tipo.

REQF007	Los recursos serán accesibles por varios usuarios a la vez.
Versión	1.0
Tipo	Comportamiento
Descripción	Un mismo dispositivo puede ser gestionado/accedido desde varios dispositivos cliente. Sólo se debe garantizar la integridad de la información.
Prioridad	Esencial

Tabla 43. REQF007- Los recursos serán accesibles por varios usuarios a la vez.

REQF008	Cada recurso tiene una selección de acciones disponible.
Versión	1.0
Tipo	Comportamiento
Descripción	Todos los recursos Iot no tienen los mismos parámetros ni funciones. Por lo cual no responden a cualquier tipo de petición.
Prioridad	Esencial

Tabla 44. REQF008- Cada recurso tiene una selección de acciones disponible.

5.4.2 No funcionales

Los requisitos no funcionales especifican criterios de calidad y características de funcionamiento que deben cumplir.

REQNF001	El sistema debe de recuperarse ante fallos de forma autónoma
Versión	1.0
Tipo	Fiabilidad
Descripción	Tanto la aplicación que simula los recursos iot como la parte cliente, deben de tener tolerancia a fallos. De forma que sean capaces de seguir su ejecución normal, tras tratar los diversos errores que se pudieran dar.
Prioridad	Esencial

Tabla 45. REQNF001- El sistema debe de recuperarse ante fallos de forma autónoma

REQNF002	Los mensajes intercambiados en la red deben ser mínimos.
Versión	1.0
Tipo	Eficiencia
Descripción	Cómo se utilizará la red local, normalmente con la tecnología Wifi. Es necesario que la red no se congestione por las labores de gestión y monitorización de los recursos iot.
Prioridad	Esencial

Tabla 46- REQNF002- Los mensajes intercambiados en la red deben ser mínimos.

REQNF003	La aplicación móvil debe ser accesible
Versión	1.0
Tipo	Usabilidad
Descripción	Estará en completo español, usará un lenguaje claro y conciso y además se podrá manipular con el mínimo de acciones posibles todas las funcionalidades.
Prioridad	Esencial

Tabla 47. REQNF003- La aplicación móvil debe ser accesible

REQNF004	Debe seguir la norma de la Open Connectivity Foundation (OCF)
Versión	1.0
Tipo	Regulatorios
Descripción	Para su posterior escalado y desarrollo, es indispensable que todos los recursos implementados y tecnologías estén desarrollados según las especificaciones.
Prioridad	Esencial

Tabla 48. REQNF004- Debe seguir la norma de la Open Connectivity Foundation (OCF)

5.5 DIAGRAMAS DE ACTIVIDAD

En las siguientes ilustraciones se muestran diagramas de flujo para moldear el comportamiento del sistema.

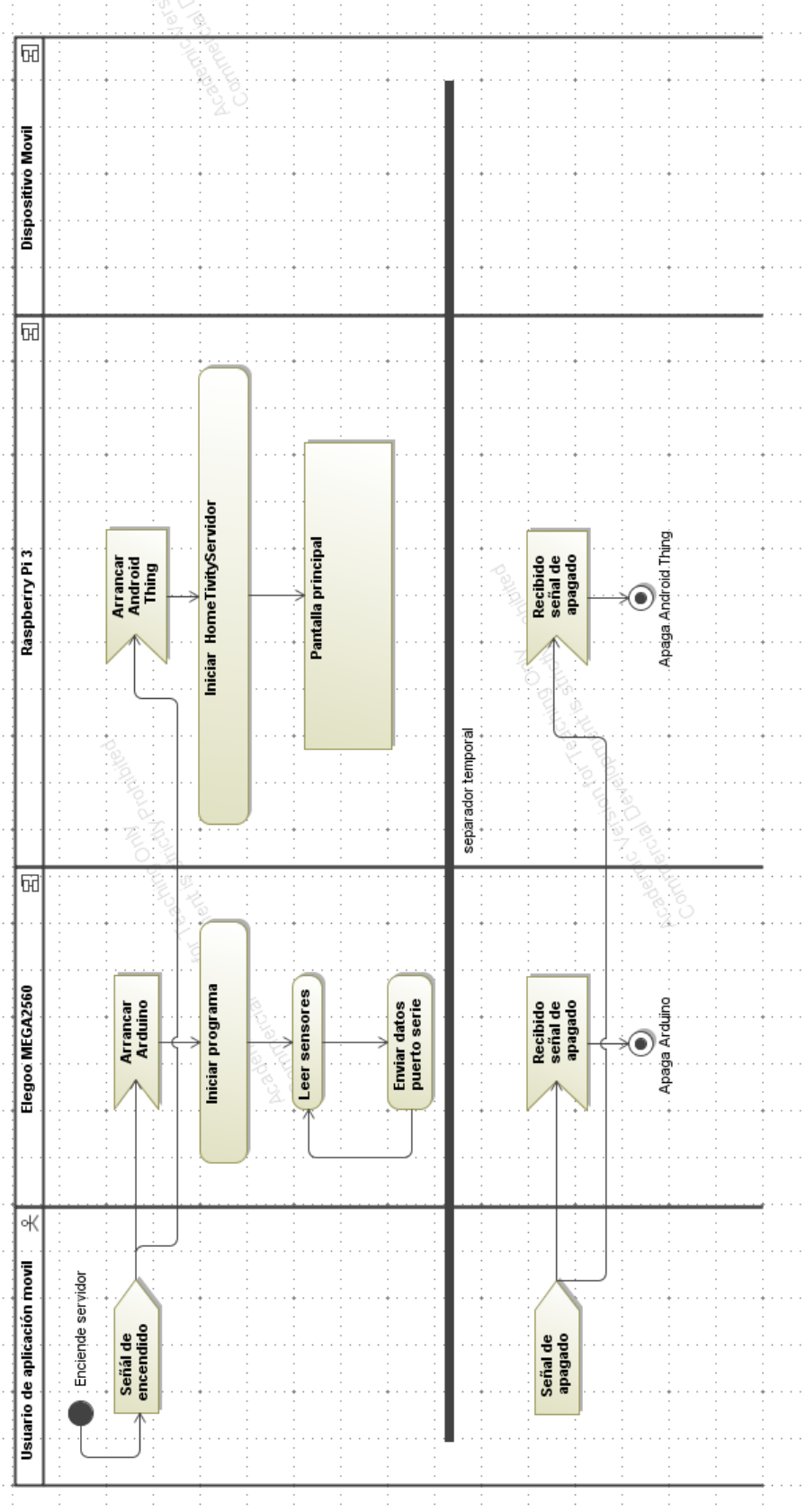


Ilustración 39. Diagrama Actividad: HomeTivityServidor (subsistema) – Inicio/Cierre

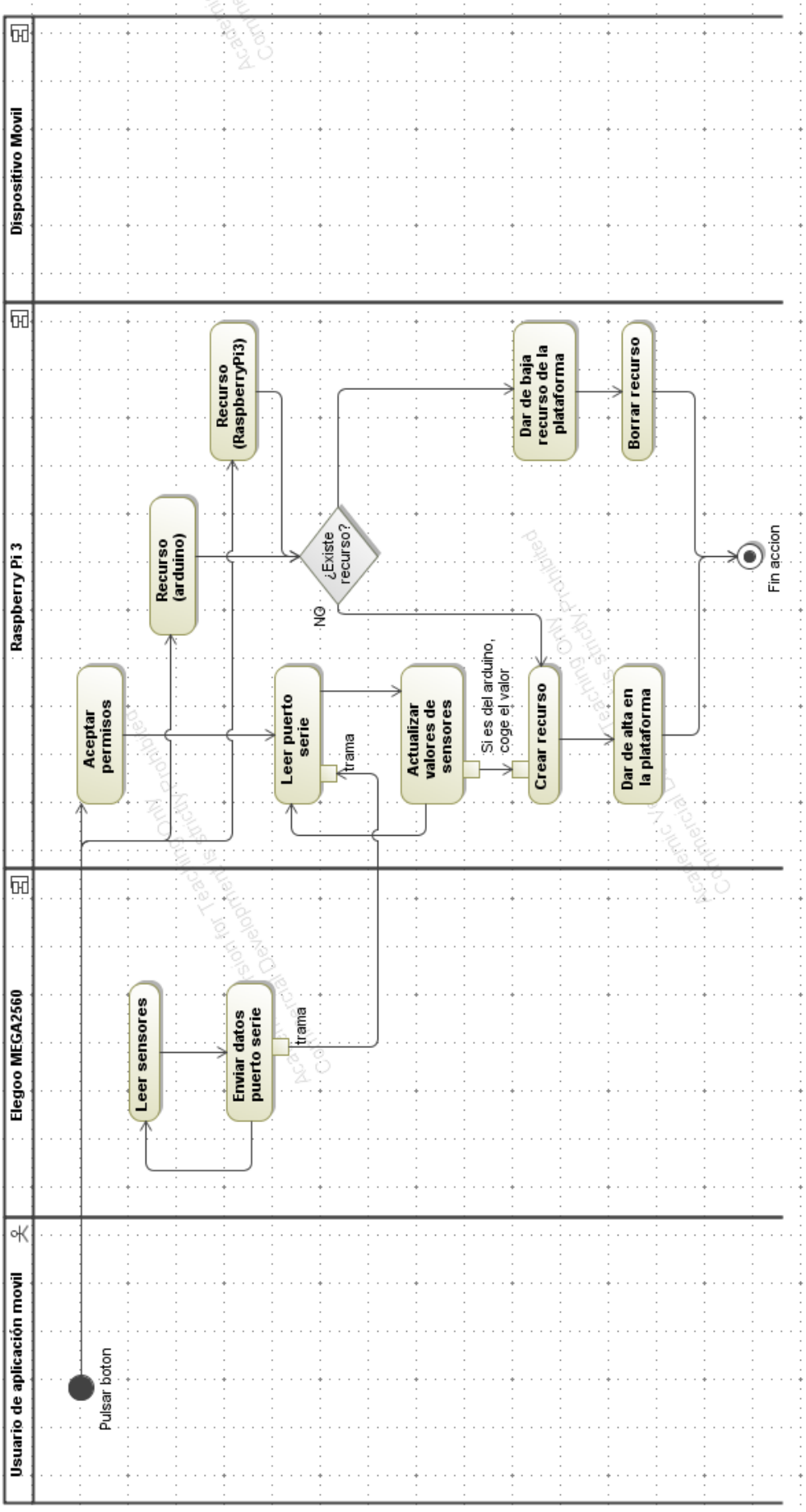


Ilustración 40. Diagrama Actividad: HomeTivityServidor (subsistema) – Alta/baja recurso

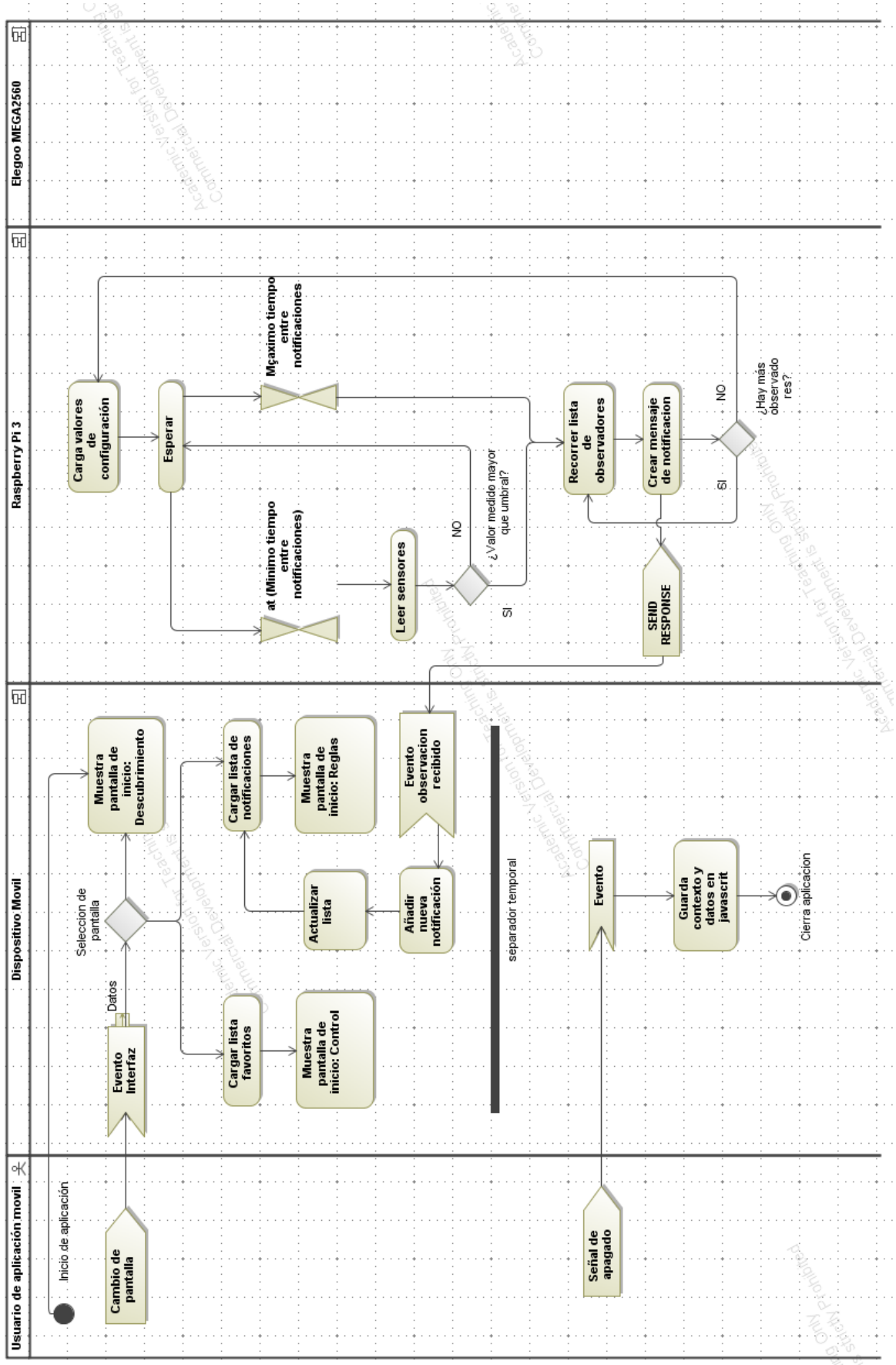


Ilustración 41. Diagrama Actividad: HomeTivityCliente (subsistema) – Inicio/pantallas/cierre + Notificaciones

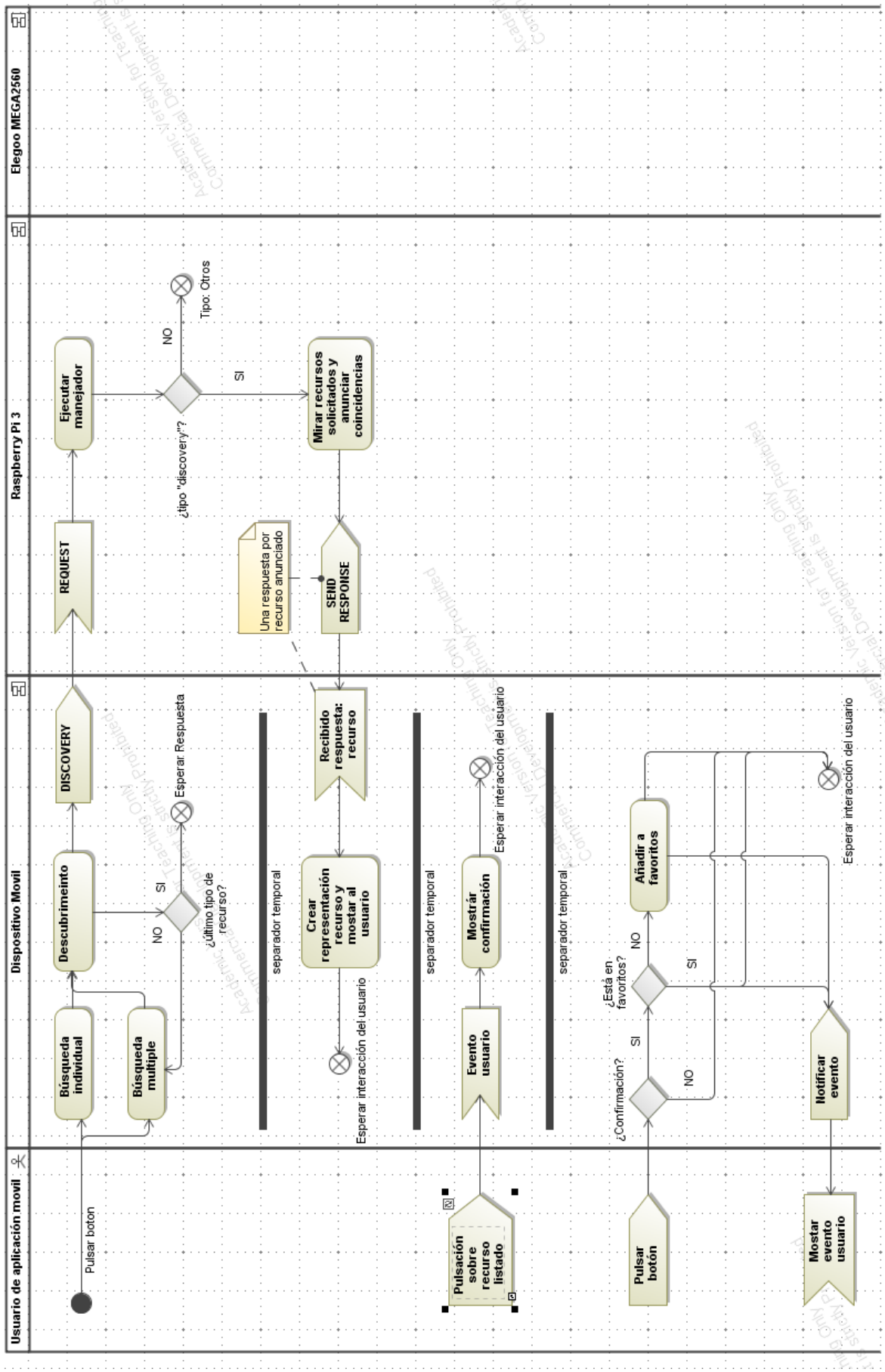


Ilustración 42. Diagrama Actividad: HomeTivityCliente (subsistema) – Descubrimiento/Favoritos

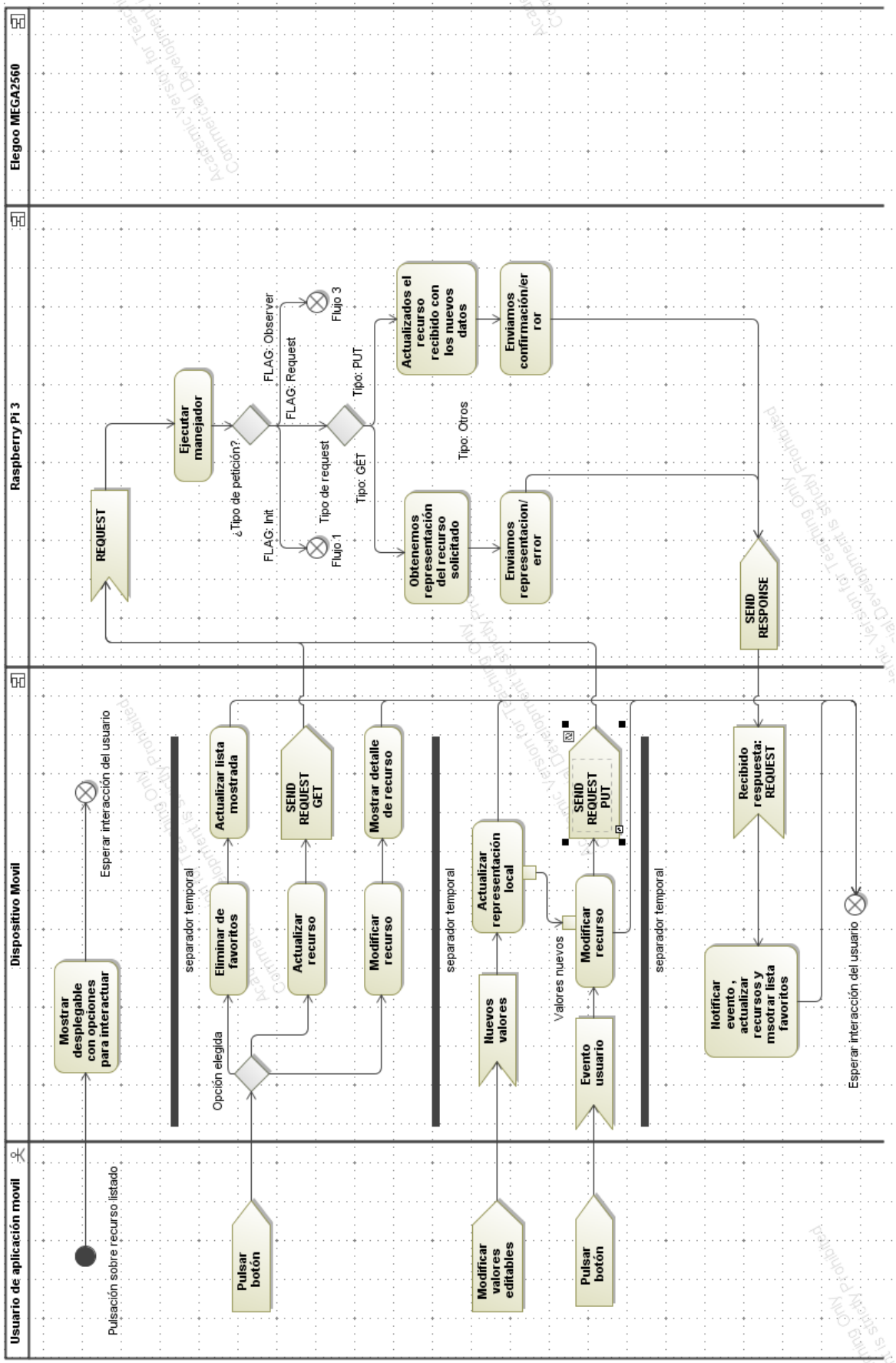


Ilustración 43. Diagrama Actividad: HomeTivityCliente (subsistema) – Pantalla control

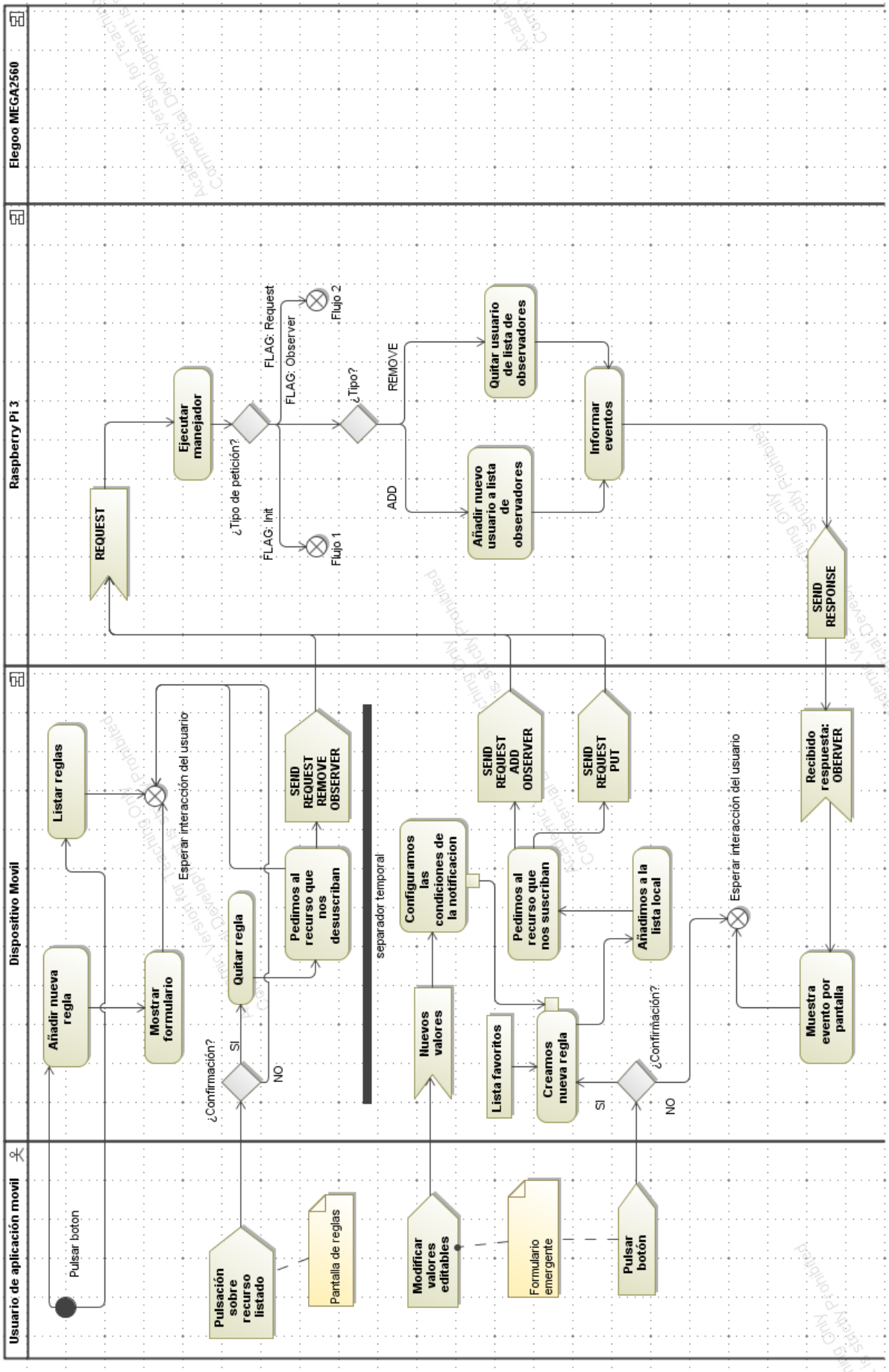


Ilustración 44. Diagrama Actividad: HomeTivityCliente (subsistema) – Alta/baja nueva regla

5.6 DIAGRAMAS DE CLASES

Las siguientes ilustraciones describen la estructura estática de los sistemas. Es decir, nos muestran como están organizadas las clases, métodos y atributos. Además de cómo se relacionan entre sí.

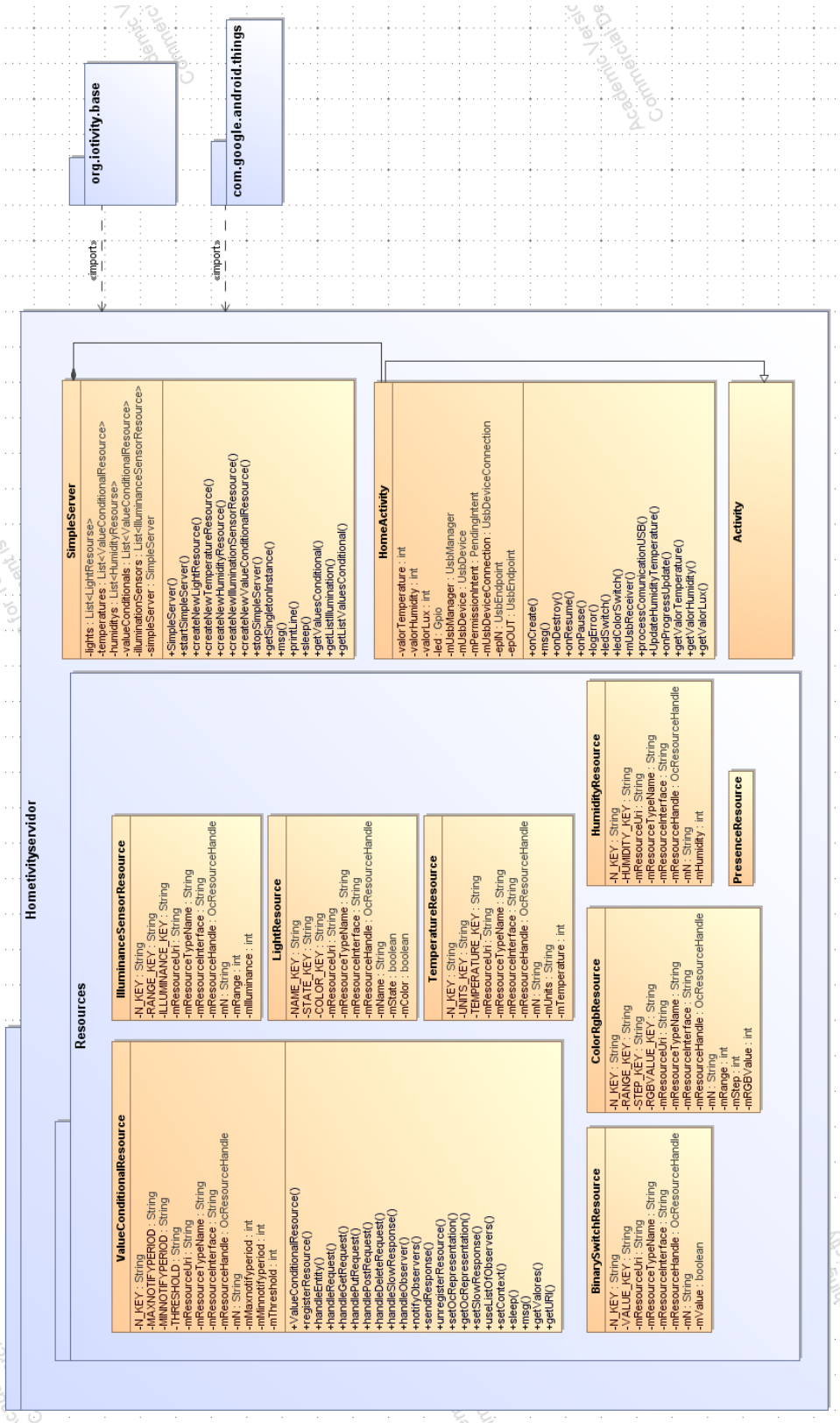


Ilustración 45. Diagrama de clases: HomeTivityServidor

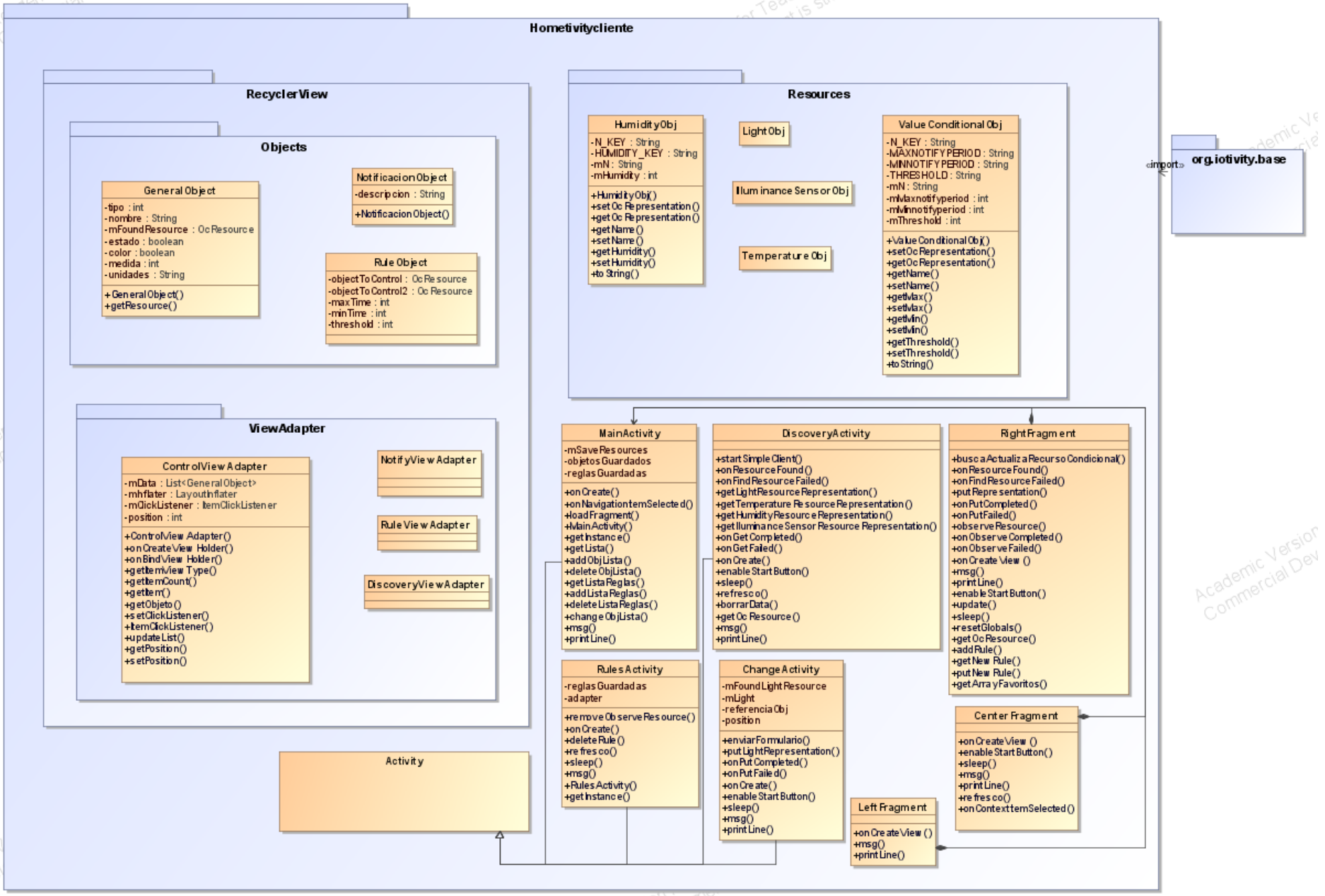


Ilustración 46. Diagrama de clases: HomeTivityCliente

6 APLICACIÓN ANDROID – SERVIDOR

La simplicidad es la máxima sofisticación.

- Leonardo da Vinci -

Capitulo centrado en la aplicación servidor. En él se explicará cómo realizar el montaje del servidor y puesta en marcha. Como es el funcionamiento y ejecución. Y se concretará el diseño y código empleado para su funcionamiento.

6.1 COMPONENTES Y MONTAJE DEL ESCENARIO

Para recrear el servidor que dará soporte a los recursos IoT en la red local, será necesario:

-Raspberry Pi 3

-Arduino

-Sensor de Humedad&Temperatura

-Sensor de luz

-LED

-2xCable USB

-Cable HDMI

-Ratón USB

Así tendremos que conectar ambos dispositivos como se ve en las ilustraciones 43 y 44. Además de conectarlas entre ellas por un cable USB.

Con el otro cable USB se alimentará la Raspberry Pi 3 (este a su vez alimenta al Arduino). Además de conectarlo a una pantalla con el HDMI y ratón para poder interaccionar con la aplicación una vez arrancada.

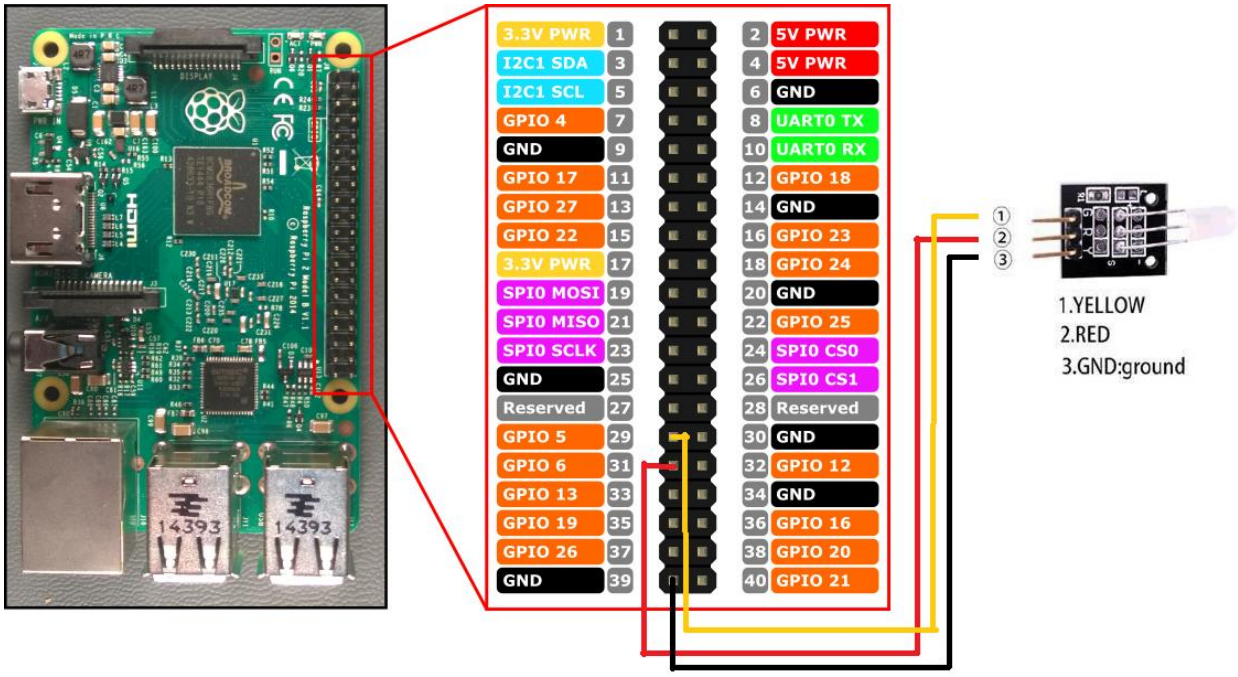


Ilustración 47. Montaje Raspberry Pi 3 con LED.

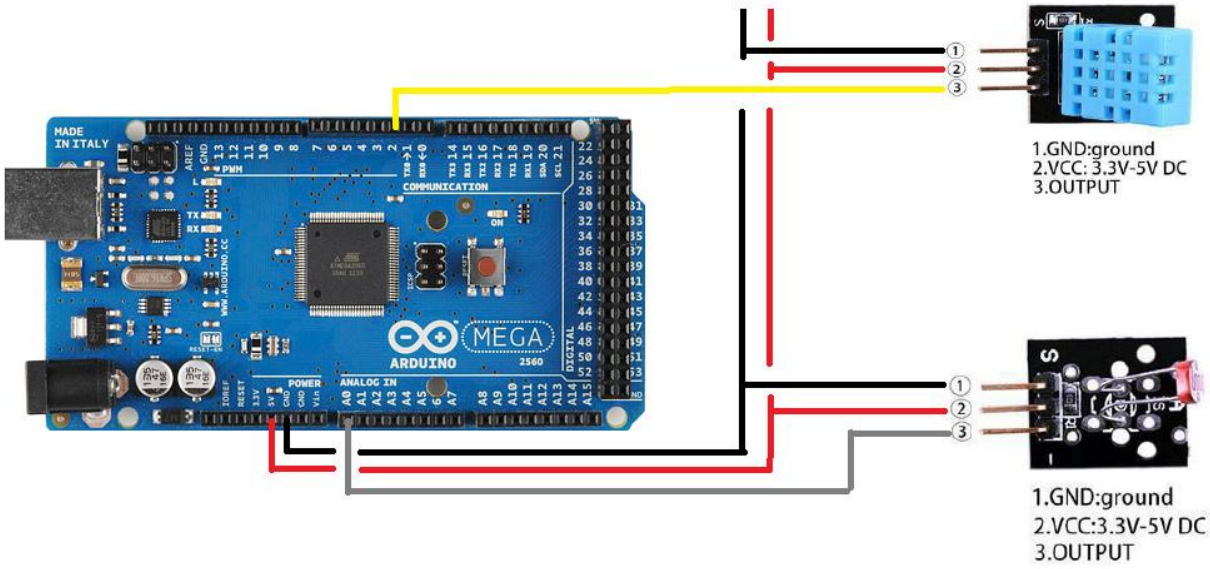


Ilustración 48. Montaje Arduino con sensores de Humedad&Temperatura y Luz.

NOTA: En los ANEXOS I y II está descrito como realizar la configuración previa de ambos dispositivos.

6.2 FUNCIONAMIENTO E INTERFAZ GRAFICA

El dispositivo “Raspberry Pi 3” tiene instalado el sistema Android Thing que, previamente configurado y conectado a la red local, ejecutará la aplicación HomeTivityServidor.

Dicha aplicación tiene como objetivo simular un conjunto de recursos IoT y darles servicio mediante el framework de Iotivity. Para ello hemos creado una interfaz básica, en un solo *activity*, para gestionarlos.

Primero encontramos el botón “Aceptar permisos sensores” que nos permite obtener permisos sobre los pines del dispositivo. Nota: Android Thing, al igual que pasa en su versión móvil, requiere de la confirmación del usuario de forma previa para ciertos permisos.

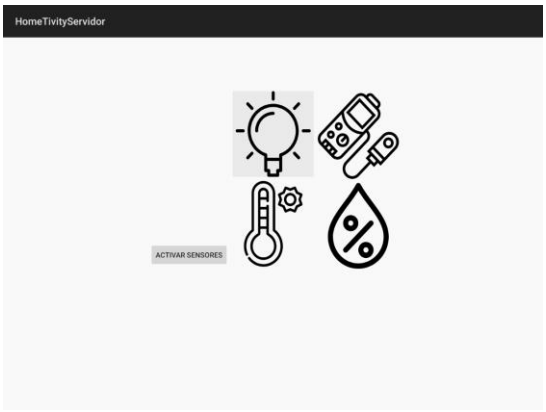


Ilustración 50. Activar sensores

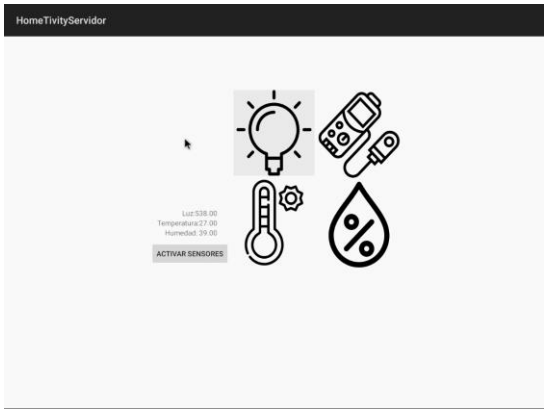


Ilustración 49. Lectura sensores

Una vez que la aplicación puede comunicarse con el Arduino y tiene acceso a todos los pines. Ya podemos interactuar con el resto de los elementos de la pantalla principal, que está formada por varios *toggle buttom*.

Así podremos encontrar dar de alta o baja cada recurso IOT en la red local, cada botón realiza ambas funciones dependiendo de su estado previo. De esta forma, según se aprecia en las figuras 48 y 49, cada recurso dado de alta estará “en color”.

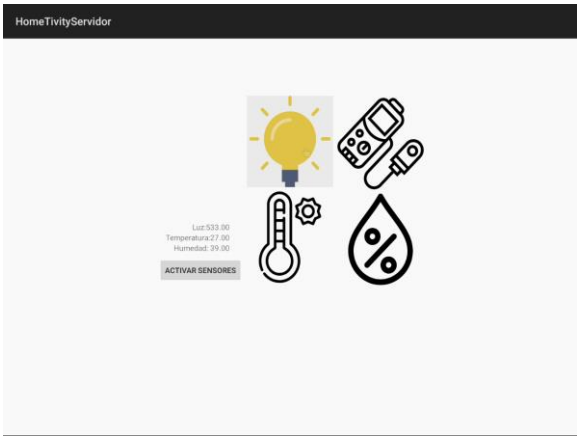


Ilustración 52. Alta de recursos 01

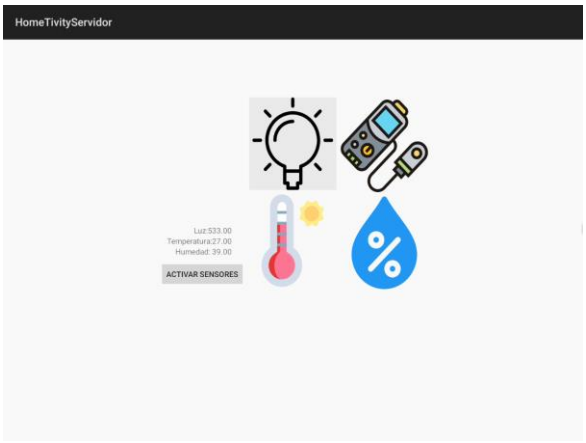


Ilustración 51. Alta de recursos 02

6.3 DISEÑO Y DESARROLLO

6.3.1 Arrancar y configurar plataforma

La aplicación servidor, para poder actuar como tal, lo primero que tiene que hacer es dar de alta la plataforma y configurarla. Para ello llamamos al método siguiente donde crea una nueva instancia y la configura con valores deseados. En nuestro caso: modo servidor, calidad de servicio bajo y permitiendo todas las interfaces disponibles.

```
public void startSimpleServer() {
    Context context =
com.example.hometivityservidor.HomeActivity.getInstance();
    PlatformConfig platformConfig = new PlatformConfig(
            com.example.hometivityservidor.HomeActivity.getInstance(),
            context,
            ServiceType.IN_PROC,
            ModeType.SERVER,
            "0.0.0.0", // By setting to "0.0.0.0", it binds to all available
                       // interfaces
            0,         // Uses randomly available port
            QualityOfService.LOW
    );

    msg("Configuring platform.");
    OcPlatform.Configure(platformConfig);
    msg("Waiting for the requests...");
    printLine();
}
```

6.3.2 Alta/Baja de recursos

Para dar de alta un recurso primero tenemos que crear una nueva instancia **OCResourse**, del tipo que queramos. Le pasamos al método contructor la uri (dirección del recurso) y los valores deseados para sus variables. En nuestro caso: nombre (string), unidades (string), temperatura (int). Y posteriormente se da de alta en pla plataforma con el método **registerResourse**.

```
public void createNewTemperatureResource(String resourceUri, String
resourceName, String resoruceUnits, int resourceTemperature){
    msg("Creating a temperature");
    TemperatureResource temperature = new TemperatureResource(
            resourceUri,
            resourceName,
            resoruceUnits,
            resourceTemperature
    );
    msg(temperature.toString());

temperature.setContext(com.example.hometivityservidor.HomeActivity.getInstanc
e());

    msg("Registering temperature as a resource");
    try {
        temperature.registerResource();
    } catch (OcException e) {
```

```
            msg("Failed to register a temperature resource");
        }

        temperatures.add(temperature);
        createNewValueConditionalResource(resourceUri +
"/ValueConditionalResURI", "valores", 30, 10, 35 );
    }
```

Detalle del método para registrar el recurso creado anteriormente en la plataforma. Implementado dentro del **recurso.java**.

```
public synchronized void registerResource() throws OcException {
    if (null == mResourceHandle) {
        mResourceHandle = OcPlatform.registerResource(
                mResourceUri,
                mResourceTypeName,
                mResourceInterface,
                this,
                EnumSet.of(ResourceProperty.DISCOVERABLE,
ResourceProperty.OBSERVABLE)
        );
    }
}
```

Al contrario que con el anterior, con el método **unregisterResource** podemos dar de baja un recurso en la plataforma. Se muestra a continuación un fragmento de código donde damos de baja a los recursos tipo temperatura que están en la lista **temperatures**.

```
case TEMPERATURE:
    for (TemperatureResource temperature : temperatures) {
        try {
            temperature.unregisterResource();
        } catch (OcException e) {
            msg("Failed to unregister a temperature resource");
        }
    }
    temperatures.clear();

    break;
```

Detalle del método para eliminar el recurso de la plataforma. Implementado dentro del **recurso.java**.

```
public synchronized void unregisterResource() throws OcException {
    if (null != mResourceHandle) {
        OcPlatform.unregisterResource(mResourceHandle);
    }
}
```

6.3.3 Implementar un nuevo recurso - Servidor

Para implementar un nuevo recurso lo primero que tenemos que hacer es darle nombre, una etiqueta, a las diferentes variables que tendrá el recurso. De esta forma podrán comunicarse diferentes SO, arquitecturas y entornos. NOTA: las etiquetas tienen que coincidir en el cliente.

Ejemplo de etiquetas para el recurso "temperatura"

```
private static final String N_KEY = "n";
private static final String UNITS_KEY = "units";
private static final String TEMPERATURE_KEY = "temperature";
```

Ejemplo de variables para el recurso "temperatura". Las primeras 4 son a nivel interno de la plataforma.

```
private String mResourceUri;                //resource URI
private String mResourceTypeName;           //resource type name.
private String mResourceInterface;          //resource interface.
private OcResourceHandle mResourceHandle;   //resource handle

private String mN;                        //temperature name
private String mUnits;                      //temperature unids
private int mTemperature;                           //temperature number
```

Ejemplo de método constructor de un recurso.

```
public TemperatureResource(String resourceUri, String name, String units, int
temp) {
    mResourceUri = resourceUri;
    mResourceTypeName = "oic.r.temperature";
    mResourceInterface = OcPlatform.DEFAULT_INTERFACE;
    mResourceHandle = null; //this is set when resource is registered

    mN = name;
    mUnits = units;
    mTemperature = temp;
}
```

6.3.4 Métodos del servidor para la gestion de peticiones.

Los recursos pueden tener implementados o no una serie de peticiones para que sean accesibles desde el cliente. A continuación, se exponen los diferentes métodos que tratan las diferentes peticiones.

Método "set" para modificar la representación del recurso. Si coinciden las etiquetas de la petición con las del recurso, se hace el traspaso de la información.

```
public void setOcRepresentation(OcRepresentation rep) {
    try {
        if (rep.hasAttribute(N_KEY)) mN = rep.getValue(N_KEY);
        if (rep.hasAttribute(UNITS_KEY)) mUnits = rep.getValue(UNITS_KEY);
        if (rep.hasAttribute(TEMPERATURE_KEY)) mTemperature =
rep.getValue(TEMPERATURE_KEY);
    } catch (OcException e) {
        Log.e(TAG, e.toString());
        msg("Failed to get representation values");
    }
}
```

Método “get” para obtener la representación del recurso y crear una respuesta que enviar al servidor.

```
public OcRepresentation getOcRepresentation() {
    OcRepresentation rep = new OcRepresentation();
    readSensor();
    try {
        rep.setValue(N_KEY, mN);
        rep.setValue(UNITS_KEY, mUnits);
        rep.setValue(TEMPERATURE_KEY, mTemperature);
    } catch (OcException e) {
        Log.e(TAG, e.toString());
        msg("Failed to set representation values");
    }
    return rep;
}
```

Método para notificar a una lista de observadores cuando se poduce uno o varios eventos.

```
private void notifyObservers(OcResourceRequest request) {
    int numWait=0;
    while (true) {
        SimpleServer servidor = SimpleServer.getSingletonInstance();
        //sacar de la lista tus recursos valor, 1 por notificacion
        ValueConditionalResource auxValue =
servidor.getValuesConditional(TEMPERATURE);

        int aux[] = auxValue.getValores();
        int mMinnotifyperiod = aux[0];
        int mMaxnotifyperiod = aux[1];
        int mThreshold = aux[2];

        msg("Esperando minimo delay...");
        sleep(mMinnotifyperiod);
        numWait+=1;
        readSensor();
        int valor = mTemperature;

        if (mThreshold < valor){
            numWait=0;
            msg("Notifying observers...");
            msg(this.toString());
            try {
                if (mIsListOfObservers) {
                    OcResourceResponse response = new OcResourceResponse();

response.setResourceRepresentation(getOcRepresentation());
                    OcPlatform.notifyListOfObservers(
                            mResourceHandle,
                            mObservationIds,
                            response);
                } else {
                    OcPlatform.notifyAllObservers(mResourceHandle);
                }


            } catch (OcException e) {
                ErrorCode errorCode = e.getErrorCode();
                if (ErrorCode.NO_OBSERVERS == errorCode) {
                    msg("No more observers, stopping notifications");
                    mObserverNotifier = null;
                    return;
                }
            }
```

```
        }
    }
}
```

Método para enviar la respuesta creada anteriormente al cliente.

```
private EntityHandlerResult sendResponse(OcResourceResponse response) {
    try {
        OcPlatform.sendResponse(response);
        return EntityHandlerResult.OK;
    } catch (OcException e) {
        Log.e(TAG, e.toString());
        msg("Failed to send response");
        return EntityHandlerResult.ERROR;
    }
}
```

6.3.5 Lectura/modificación de pines de raspberry pi 3

Para poder gestionar los pines de la placa será necesario importar las librerías:

```
import com.google.android.things.pio.Gpio;
import com.google.android.things.pio.PeripheralManager;
```

y declarar las variables:

```
private static final String LED_GREEN_PIN = "BCM5";
private static final String LED_RED_PIN = "BCM6";

private Gpio led;
```

Así utilizando los siguientes comandos podremos activar un pin concreto, usando la etiqueta adecuada (mirar ANEXO I), y darle un valor:

```
led = service.openGpio(LED_RED_PIN);
led.setDirection(Gpio.DIRECTION_OUT_INITIALLY_LOW);
led.setValue(false);
```

y con el siguiente, desactivarlo:

```
led.close();
```

Información y Código de ejemplo sacados de la referencia [12]

6.3.6 Lectura de puerto serie por USB

Como se explica en el ANEXO II: el Arduino enviará por el puerto USB una secuencia de datos con los valores medidos por los sensores. Así que solo nos queda leer dichos valores, tratar la información y almacenarla. Estos fragmentos de código están obtenidos de la referencia [13] y adaptados a nuestra aplicación. A continuación, se exponen los métodos utilizados (secuencia de ejecución).

NOTA: nos harán falta las siguientes librerias:

```
import android.hardware.usb.UsbConstants;
import android.hardware.usb.UsbDevice;
import android.hardware.usb.UsbDeviceConnection;
import android.hardware.usb.UsbEndpoint;
import android.hardware.usb.UsbInterface;
import android.hardware.usb.UsbManager;
import android.hardware.usb.UsbRequest;
```

```
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.HashMap;
import java.util.Iterator;
```

Y declarar las siguientes variables:

```
// TODO: Variables USB
UsbManager mUsbManager;
UsbDevice mUsbDevice;
PendingIntent mPermissionIntent;
UsbDeviceConnection mUsbDeviceConnection;
UsbEndpoint epIN = null;
UsbEndpoint epOUT = null;
```

Así el primer paso es habilitar un botón para aceptar los permisos se lectura por el USB:

```
// TODO: Boton Conectar.
btn = (Button) findViewById(R.id.buttonejemplo);
btn.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {

        //TODO: Obtemos el Manager USB del sistema Android
        mUsbManager = (UsbManager) getSystemService(Context.USB_SERVICE);

        // TODO: Recuperamos todos los dispositvos USB detectados
        HashMap<String, UsbDevice> deviceList = mUsbManager.getDeviceList();

        //TODO: en nuestor ejemplo solo conectamos un disposito asi que sera
        // el unico que encontraremos.
        Iterator<UsbDevice> deviceIterator = deviceList.values().iterator();
        if(deviceIterator.hasNext()){
                mUsbDevice = deviceIterator.next();
                Log.d(TAG, "Name: " + mUsbDevice.getDeviceName());
                Log.d(TAG, "Protocol: " + mUsbDevice.getDeviceProtocol());
                //TODO: Solicitamos el permiso al usuario.
                if (mUsbDevice.getDeviceName().contains("005")) {
                    mUsbManager.requestPermission(mUsbDevice,
mPermissionIntent);
                }
        }else{
            Log.e(TAG, "Dispositvo USB no detectado.");
        }
```

```
    }
});
```

Cuando se pulse dicho botón, saltará el siguiente método:

```
// TODO: Al conectar a un dispositvo USB se solicita un permiso al usuario
// este broadcast se encarga de recoger la respuesta del usuario.
private static final String ACTION_USB_PERMISSION =
"com.android.example.USB_PERMISSION";

private final BroadcastReceiver mUsbReceiver = new BroadcastReceiver(){
    public void onReceive(Context context, android.content.Intent intent) {
        String action = intent.getAction();

        // TODO: Al aceptar el permiso del usuario.
        if (ACTION_USB_PERMISSION.equals(action)){
            synchronized (this) {
                //UsbDevice device = (UsbDevice)
intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
                if
(intent.getBooleanExtra(UsbManager.EXTRA_PERMISSION_GRANTED, false)){
                    Log.d(TAG, "Permiso aceptado");
                    processComunicationUSB();
                }else{
                    Log.e(TAG, "Permiso denegado");
                }
            }
        }

        // TODO: Al desconectar el dispositivo USB cerramos las conexiones y
liberamos la variables.
        if (UsbManager.ACTION_USB_DEVICE_DETACHED.equals(action)) {
            UsbDevice device =
(UsbDevice)intent.getParcelableExtra(UsbManager.EXTRA_DEVICE);
            if (device != null) {
                // call your method that cleans up and closes communication
with the device
            }
        }
    }
};
```

De forma que, al terminar su ejecución de forma afirmativa, se llama a **processComunicationUSB.** Método que será el encargado de realizar la comunicación con el Arduino en bucle.

```
protected void processComunicationUSB() {

    boolean forceClaim = true;

    mUsbDeviceConnection = mUsbManager.openDevice(mUsbDevice);
    if(mUsbDeviceConnection == null){
        Log.e(TAG, "No se ha podido conectar con el dispositivo USB.");
        finish();
    }

    // TODO: getInterfase(1) Obtiene el tipo de comunicacion CDC
(USB_CLASS_CDC_DATA)
    UsbInterface mUsbInterface = mUsbDevice.getInterface(1);

    // TODO: Obtenemos los Endpoints de entrada y salida para el interface
que hemos elegido.
    for (int i = 0; i < mUsbInterface.getEndpointCount(); i++) {
        if (mUsbInterface.getEndpoint(i).getType() ==
UsbConstants.USB_ENDPOINT_XFER_BULK) {
            if (mUsbInterface.getEndpoint(i).getDirection() ==
UsbConstants.USB_DIR_IN)
                epIN = mUsbInterface.getEndpoint(i);
            else
                epOUT = mUsbInterface.getEndpoint(i);
        }
    }

    mUsbDeviceConnection.claimInterface(mUsbInterface, forceClaim);

    // TODO: Mensaje de configuración para el Device.
    int baudRate = 9600;
    byte stopBitsByte = 1;
    byte parityBitesByte = 0;
    byte dataBits = 8;
    byte[] msg = {
            (byte) (baudRate & 0xff),
            (byte) ((baudRate >> 8) & 0xff),
            (byte) ((baudRate >> 16) & 0xff),
            (byte) ((baudRate >> 24) & 0xff),
            stopBitsByte,
            parityBitesByte,
            (byte) dataBits
    };

    mUsbDeviceConnection.controlTransfer(UsbConstants.USB_TYPE_CLASS | 0x01,
0x20, 0, 0, msg, msg.length, 5000);
    // (UsbConstants.USB_TYPE_CLASS | 0x01) 0x21 -> Indica que se envia un
parametro/mensaje del Host al Device (movil a la placa leonardo)
    // 0x20 -> paramtro/mensaje SetLineCoding

    mUsbDeviceConnection.controlTransfer(UsbConstants.USB_TYPE_CLASS | 0x01,
0x22, 0x1, 0, null, 0, 0);
    // (UsbConstants.USB_TYPE_CLASS | 0x01) 0x21 -> Indica que se envia un
parametro/mensaje del Host al Device (movil a la placa leonardo)
    // 0x22 -> paramtro/mensaje SET_CONTROL_LINE_STATE (DTR)
    // 0x1  -> Activado.
    // Mas info: http://www.usb.org/developers/devclass_docs/usbcdc11.pdf

    // TODO: Ejecutar en un hilo
    new UpdateHumidityTemperature().execute();
}
```

Como podemos ver en la última línea, llamará a **UpdateHumidityTemperature** de forma asíncrona para ir actualizando los valores sin comprometer su integridad y el funcionamiento del resto de la aplicación.

```
private class UpdateHumidityTemperature extends AsyncTask<String, String,
String> {

    @Override
    protected String doInBackground(String... params) {

        String line=new String();

        int bufferMaxLength=epIN.getMaxPacketSize();
        ByteBuffer mBuffer = ByteBuffer.allocate(bufferMaxLength);
        UsbRequest inRequest = new UsbRequest();
        inRequest.initialize(mUsbDeviceConnection, epIN);

        while(inRequest.queue(mBuffer, bufferMaxLength) == true){

            mUsbDeviceConnection.requestWait();

            try {

                // Recogemos los datos que recibimos en un
                line = line + new String(mBuffer.array(), "UTF-8").trim();

                if (line.length()>0){

                    char endLine = line.charAt(line.length()-1);
                    if (endLine == ';'){

                        Log.d(TAG, "Encontrada final de linea: " + line);

                        // TODO: Procesar Linea
                        String[] parts = line.split(",");
                        String humidity = parts[0].split(":")[1];
                        //String temperature =
parts[1].split(":")[1].replace(";", "");
                        String temperature = parts[1].split(":")[1];
                        String lux = parts[2].split(":")[1].replace(";", "");
                        // TODO: Actualizamos el GUI
                        publishProgress(humidity, temperature, lux);

                        line = "";
                    }
                }

            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        return null;
    }
```

7 APLICACIÓN ANDROID – CLIENTE

Presta atención al feedback negativo y solicítalo, particularmente de los amigos. Difícilmente alguien hace eso y es de mucha ayuda.

- Elon Musk -

Capítulo centrado en la aplicación cliente. En él se explicará cómo es el funcionamiento y ejecución de este. Y se concretará el diseño y código empleado para su funcionamiento.

7.1 FUNCIONAMIENTO E INTERFAZ GRÁFICA

La aplicación HomeTivityCliente está formado por 3 pantallas principales: descubrimiento, control y reglas. Desde las cuales interactuaremos para realizar todos los procesos de gestión y control de los recursos IoT que compartan red local con dicha aplicación. Estás 3 pantallas principales están formadas por un Activity principal y tres fragment, uno por pantalla, con la idea de estructurar el código mejor.

De esta forma, la pantalla inicial será "descubrimiento" y está formada por varios image button. Podemos distinguir 2 tipos, un primero más grande que nos permite explorar todos los recursos IoT de la red local y un segundo tipo, que a su vez son varios, para realizar una búsqueda exhaustiva de un tipo concreto.

Nota: en cualquier momento podemos alternar entre los 3 fragment utilizando los botones inferioes de la pantalla.

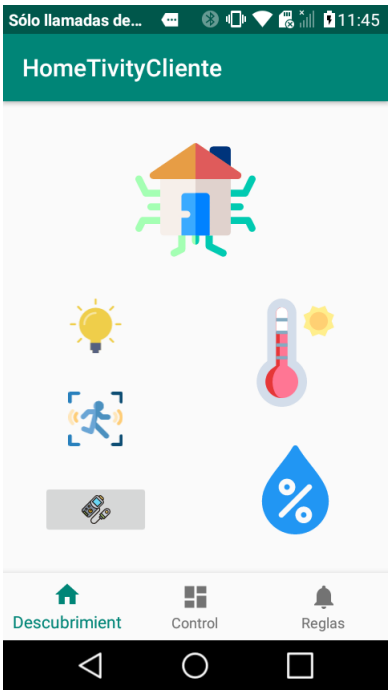


Ilustración 53. Pantalla principal: Descubrimiento

Una vez seleccionemos la opción correspondiente, se lanzará un nuevo activity formado por un myrecyclerview que listará todos los recursos encontrados. Dichas coincidencias dependerán de la opción elegida y la situación de la red local: recursos dados de alta.

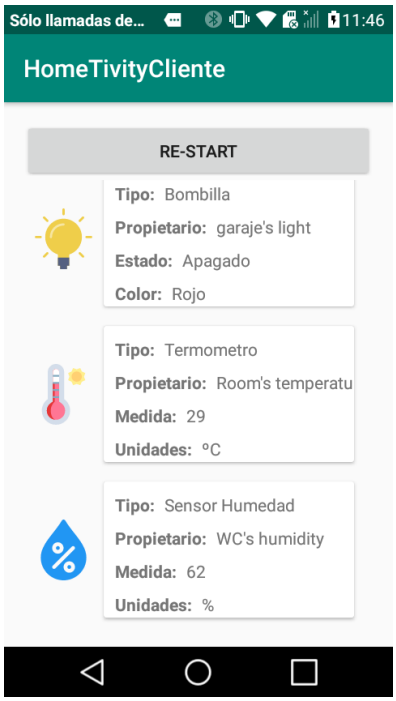


Ilustración 54. Resultado búsqueda

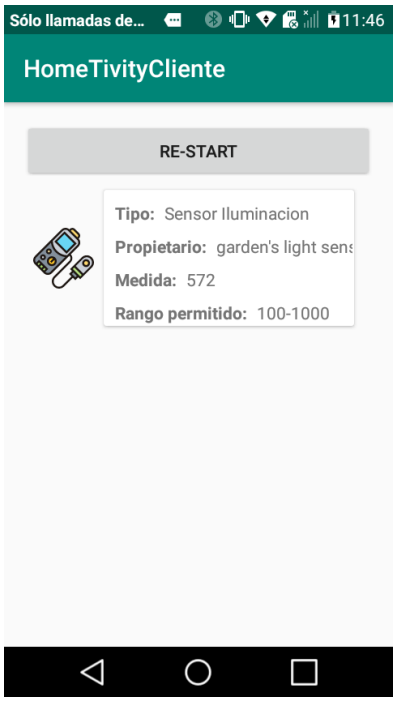


Ilustración 55. Resultado búsqueda

Interactuando con los diferentes recursos listados, podremos añadirlos a favoritos para posteriormente gestionarlos. Si dicho recurso ya se encuentra en favoritos, saltará el mensaje de error de la figura 58.

El mensaje de confirmación está formado por un *alert dialog* con dos botones. Si confirmamos, se añadirá dicho recurso a una lista interna. Nota: al contrario que los recursos listados en el descubrimiento, la lista de favoritos y su contenido son datos persistentes de la aplicación que se almacenan. Es decir, en posteriores usos de la aplicación seguirán estando.

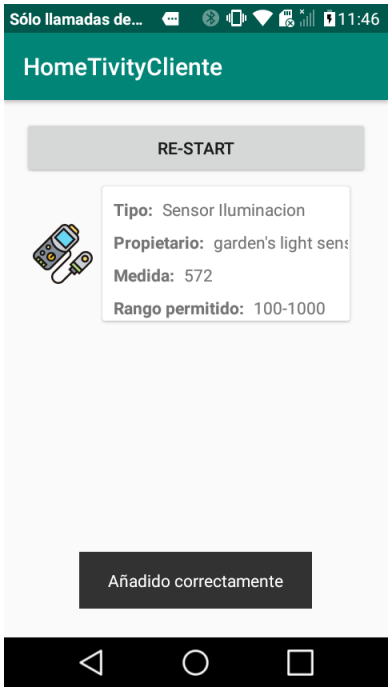


Ilustración 56. Añadir a favoritos

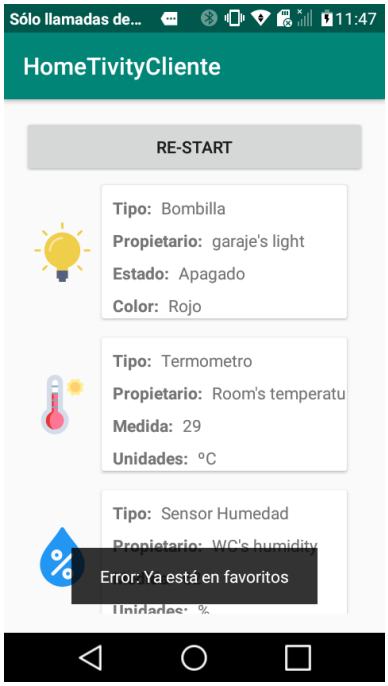


Ilustración 57. Favoritos éxito

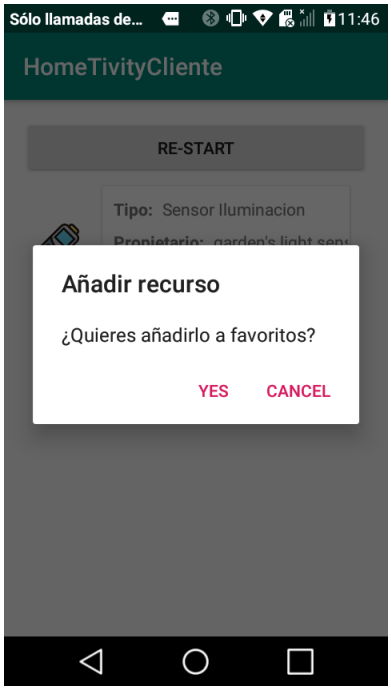


Ilustración 58. Favoritos error

Respecto a la pantalla de control, encontramos otro *myrecyclerview* pero esta vez listando los recursos almacenados en favoritos figura 55. Así mantenienndo pulsado un *item*, de desplegará un menú emergente, formado por un *alert dialog*, donde podrás realizar tres acciones. Actualizar los valores del recurso, en ese momento la representación puede no ser la misma con respecto al recurso real, modificar algún parámetro de este o eliminarlo de favoritos.

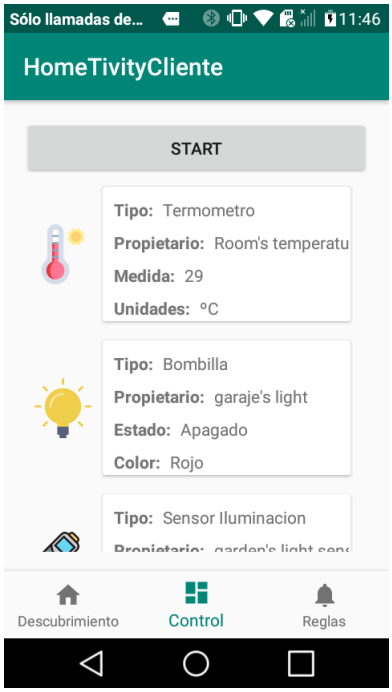


Ilustración 59. Pantalla principal: control

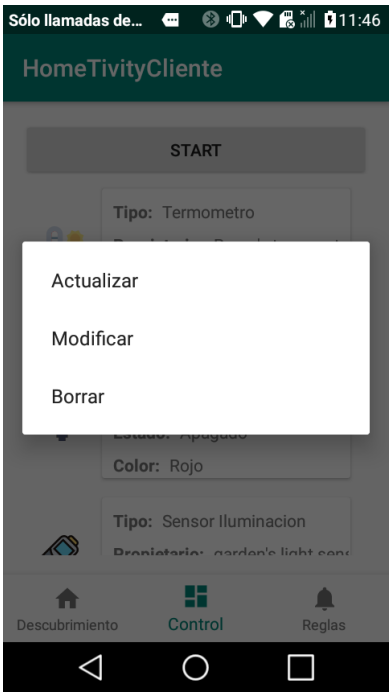


Ilustración 60. Menu acciones, favoritos

En el caso de que seleccionemos “modificar recurso”, se abrirá un nuevo activity que estará formado por un formurario con descripciones y valores editables, ya sea mediante texto editable, desplegable o botones. Figuras 57 y 58

Ilustración 61. Formulario ejemplo

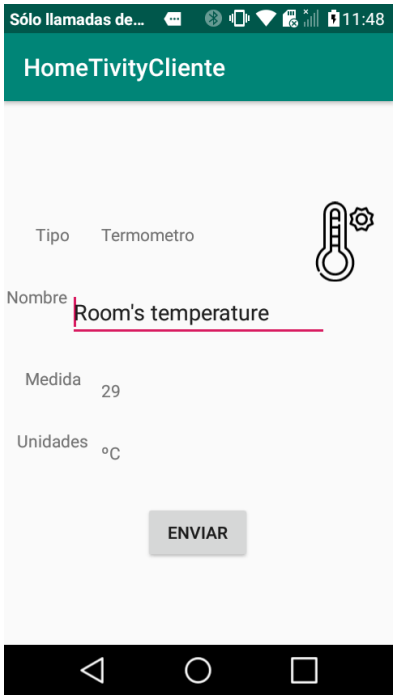


Ilustración 62. Formulario ejemplo

Por último, encontramos la pantalla reglas, formada por dos botones y un *myrecyclerview*. El primer botón nos permite añadir nuevas reglas, el segundo gestionarlas (quitarlas) y listará las notificaciones recibidas: figuras 59 e 60

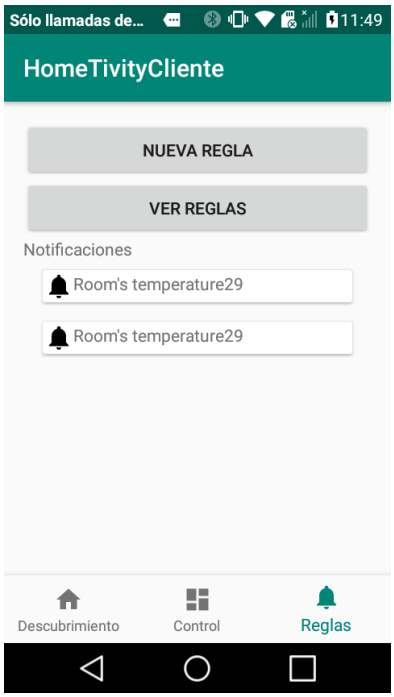


Ilustración 63. Pantalla principal: reglas

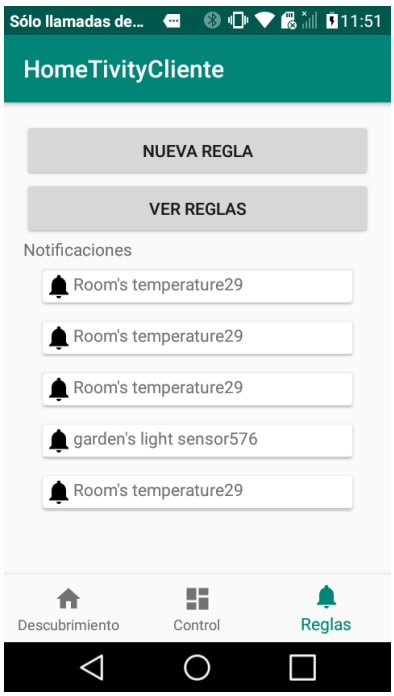


Ilustración 64. Pantalla principal: reglas

Como ya hemos comentado, también podemos añadir nuevas reglas mediante un formulario emergente dentro de un *alert dialog* personalizado, figura 61. Además de listar las reglas creadas para darlas de baja llegado el caso, figura 66 y figura 67 respectivamente.

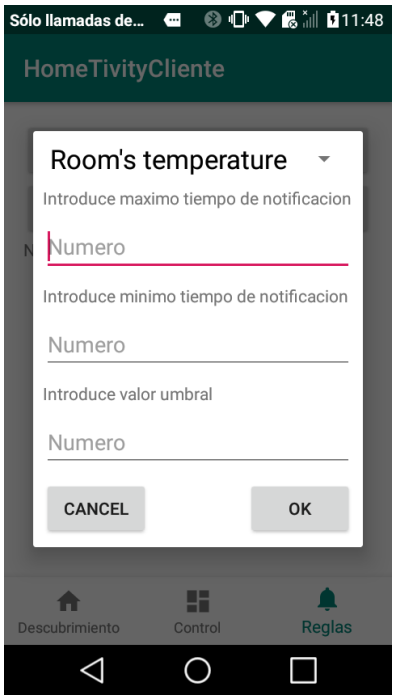


Ilustración 65. Nueva regla

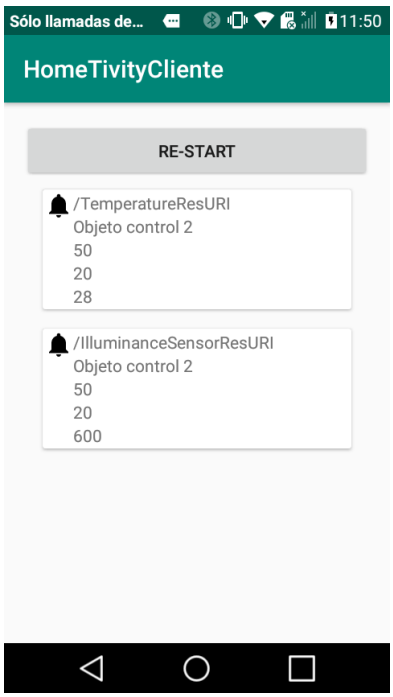


Ilustración 66. Lista reglas

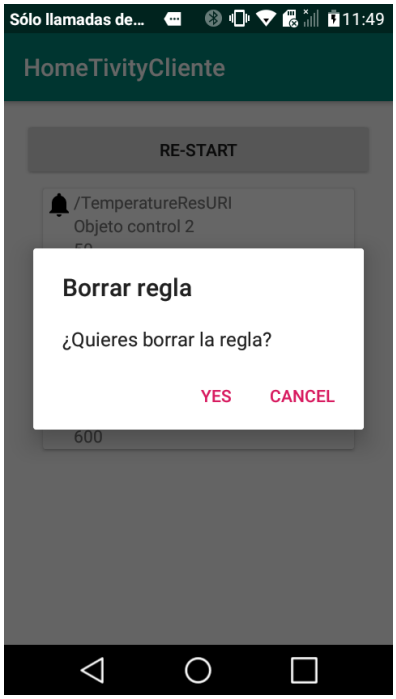


Ilustración 67. Borrar regla

Hay que añadir que el formulario en cuestión tiene un formato fijo y el primer valor hace referencia al objeto a controlar. Además, dicho objeto tiene que estar en la lista de favoritos.

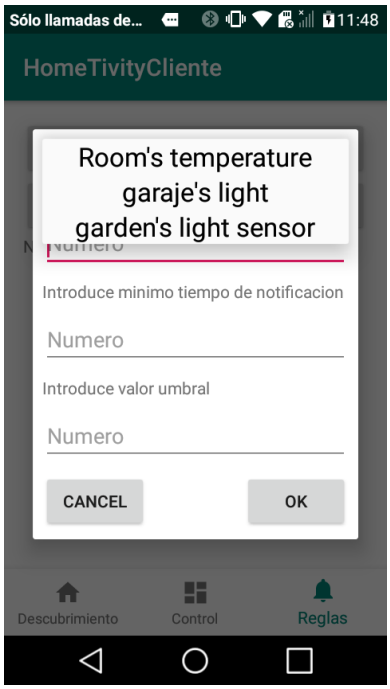


Ilustración 68. Seleccionar recurso de favoritos

Por ultimo a modo de resumen, se presenta la ilustración 68. Donde se muestra de forma secuencial como podemos movernos entre pantallas.

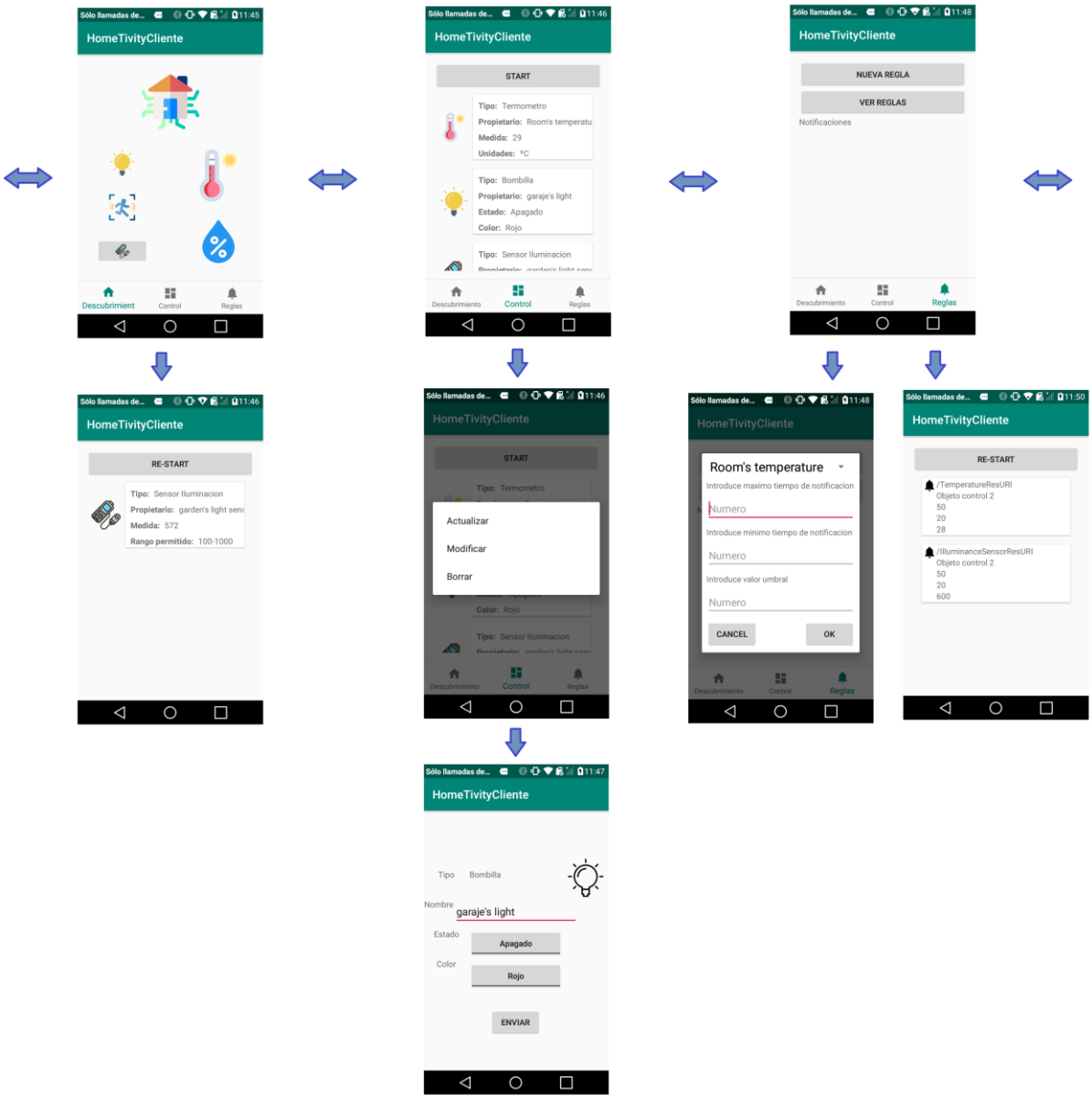


Ilustración 69. Resumen pantallas - secuencial

7.2 DISEÑO Y DESARROLLO

7.2.1 Implementar un nuevo recurso - Cliente

Para implementar un nuevo recurso en el cliente tenemos que crear una representación local del recurso. Dotarle de métodos para leer y modificar esas variables. Además de establecer las mismas etiquetas que en el servidor para que se puedan comunicar diferentes SO, arquitecturas y entornos.

```
public class TemperatureObj {

    public static final String N_KEY =  "n";
    public static final String UNITS_KEY = "units";
    public static final String TEMPERATURE_KEY = "temperature";

    private String mN;
    private String mUnits;
    private int mTemperature;

    public TemperatureObj() {
        mN = "";
        mUnits = "C/F/K";
        mTemperature = 0;
    }

    public void setOcRepresentation(OcRepresentation rep) throws OcException
{
        mN = rep.getValue(N_KEY);
        mUnits = rep.getValue(TemperatureObj.UNITS_KEY);
        mTemperature = rep.getValue(TemperatureObj.TEMPERATURE_KEY);
    }

    public OcRepresentation getOcRepresentation() throws OcException {
        OcRepresentation rep = new OcRepresentation();
        rep.setValue(N_KEY, mN);
        rep.setValue(UNITS_KEY, mUnits);
        rep.setValue(TEMPERATURE_KEY, mTemperature);
        return rep;
    }
```

7.2.2 Métodos del cliente para realizar peticiones.

Fragmento del método para realizar el descubrimiento de recursos. En este caso correspondiente al recurso temperatura.

```
case TEMP_KEY:
    try {
        msg("Finding all resources of type \"oic.r.temperature\".");
        String requestUri = OcPlatform.WELL_KNOWN_QUERY +
"?rt=oic.r.temperature";
        OcPlatform.findResource("",
                requestUri,
                EnumSet.of(OcConnectivityType.CT_DEFAULT),
                this
        );

    } catch (OcException e) {
        Log.e(TAG, e.toString());
        msg("Failed to invoke find resource API");
    }

    printLine();
    break;
```

Método para preguntar por estado actual de un recurso al servidor.

```
/**
 * Local method to get representation of a found temp resource
 */
private void getTemperatureResourceRepresentation() {
    msg("Getting temperature Representation...");

    Map<String, String> queryParams = new HashMap<>();
    try {
        // Invoke resource's "get" API with a OcResource.OnGetListener event
        // listener implementation
        sleep(1);
        mFoundTemperatureResource.get(queryParams, this);
    } catch (OcException e) {
        Log.e(TAG, e.toString());
        msg("Error occurred while invoking \"get\" API");
    }
}
```

Método para actualizar el estado de un recurso en el servidor.

```
private void putRepresentation() {
    RuleObject newRule = getNewRule();

    //set new values
    mValueConditional.setName("Condicion");
    mValueConditional.setMax(newRule.maxTime);
    mValueConditional.setMin(newRule.minTime);
    mValueConditional.setThreshold(newRule.threshold);


    msg("Putting light representation...");
    OcRepresentation representation = null;
    try {
        representation = mValueConditional.getOcRepresentation();
    } catch (OcException e) {
        Log.e(TAG, e.toString());
        msg("Failed to get OcRepresentation from a light");
    }

    representation.toString();
    Map<String, String> queryParams = new HashMap<>();

    try {
        sleep(1);
        // Invoke resource's "put" API with a new representation, query
parameters and
        // OcResource.OnPutListener event listener implementation
        mFoundConditionalResource.put(representation, queryParams, this);
    } catch (OcException e) {
        Log.e(TAG, e.toString());
        msg("Error occurred while invoking \"put\" API");
    }
}
```

Método para suscribirse a la lista de observadores de un recurso.

```
private void observeResource(OcResource newRule) {
    try {
        sleep(1);
        // Invoke resource's "observe" API with a observe type, query
parameters and
        // OcResource.OnObserveListener event listener implementation
        newRule.observe(ObserveType.OBSERVE, new HashMap<String, String>(),
this);
    } catch (OcException e) {
        Log.e(TAG, e.toString());
        msg("Error occurred while invoking \"observe\" API");
    }
}
```

Hay que comentar que esto son solo fragmentos de la aplicación. Por ejemplo, toda la parte de los *handle* (manejadores) para tratar las peticiones y elegir que hacer en cada momento, no se ha representado en la memoria por extensidad. Por lo que se anima a utilizar el CD proporcionado para tener una visión más completa.

8 PRUEBAS Y VALIDACIÓN

Si la depuración es el proceso de eliminar errores, entonces la programación debe ser el proceso de introducirlos.

- Edsger W. Dijkstra -

Capítulo dedicado al desarrollo de una serie de pruebas con el fin de detectar errores que impidan alcanzar los objetivos anteriormente detallados en el Capítulo 5: Especificación de requisitos.

TEST01	**Lectura de sensores**
Descripción	Al pulsar el botón "Activar sensores" la aplicación debe de leer de forma continuada el valor de los sensores a través del puerto USB. Tratar la información correctamente y no quitar el control al usuario para el alta/baja de recursos.
Importancia	Indispensable
Éxito	SI

Tabla 49. TEST01- Lectura de sensores

TEST02	**Alta/Baja de recursos**
Descripción	Cuando pulsamos un botón vinculado a un recurso, la primera vez deberá darlo de alta en la plataforma y la segunda de baja. Para verificarlo usaremos un cliente y/o log del servidor.
Importancia	Indispensable
Éxito	SI

Tabla 50. TEST02- Alta/Baja de recursos

TEST03	**Búsqueda de recursos (global)**
Descripción	Desde la pantalla "Descubrimiento" en el cliente y pulsando el botón superior (con forma de casa). Se debe realizar una búsqueda de todos los recursos implementados en las redes conectadas. Iremos dando de alta y bajas varios recursos para comprobar que todo funciona correctamente.
Importancia	Indispensable
Éxito	SI

Tabla 51. TEST03- Búsqueda de recursos (global)

TEST04	**Búsqueda de recursos (Específica)**
Descripción	Desde la pantalla "Descubrimiento" en el cliente y pulsando unos de los botones inferiores al botón superior (con forma de casa). Se debe realizar una búsqueda del recurso deseado en las redes conectadas. Iremos dando de alta y bajas varios recursos para comprobar que todo funciona correctamente.
Importancia	Importante
Éxito	SI

Tabla 52. TEST04- Búsqueda de recursos (Específica)

TEST05	**Añadir y listar a favoritos**
Descripción	Después de realizar una búsqueda, se podrá añadir a una lista de favoritos. También se comprobará que no se puede añadir el mismo recurso dos veces y que la información se almacena. Para lo último se minimizará y reiniciará la aplicación.
Importancia	Principal
Éxito	SI

Tabla 53. TEST05- Añadir y listar a favoritos

TEST06	**Actualizar favorito**
Descripción	En el servidor damos de alta el sensor luminosidad, lo buscamos y añadimos a favoritos. Dentro de la pantalla “control” seleccionamos el mismo recurso y le damos a actualizar. Ahora tapamos el sensor con la mano y volvemos a actualizar. Debería haber cambiado el valor y coincidir servidor-cliente.
Importancia	Principal
Éxito	SI

Tabla 54. TEST06- Actualizar favorito

TEST07	**Borrar favorito**
Descripción	Añadimos varios recursos a favoritos y borramos posteriormente. Deberían de irse borrando uno a uno y en el orden seleccionado.
Importancia	Principal
Éxito	SI

Tabla 55. TEST07- Borrar favorito

TEST08	**Modificar favorito**
Descripción	Damos de alta el recurso LED en el servidor, lo añadimos a favoritos y le damos a modificar. Entraremos en una nueva pantalla donde podemos actuar sobre el LED en tiempo real. Probaremos: encender, cambiar color, apagar, cambiar color y encender. Deberá corresponderse la secuencia realizada en el cliente con el comportamiento del LED.
Importancia	Principal
Éxito	SI

Tabla 56. TEST08- Modificar favorito

TEST09	**Crear una regla**
Descripción	Damos de alta y añadimos el recurso temperatura a favoritos. Ahora en la pantalla reglas: pulsamos el botón nueva regla, seleccionamos el recurso de la lista de favoritos y rellenamos el formulario.
Importancia	Secundario
Éxito	SI

Tabla 57. TEST09- Crear una regla

TEST10	**Listar las reglas**
Descripción	Después de crear varias reglas, con diferentes recursos, le damos al botón ver reglas. Deberían de estar todas las reglas previamente creadas.
Importancia	Secundario
Éxito	SI

Tabla 58. TEST10- Listar las reglas

TEST11	**Borrar una regla**
Descripción	Tras listar las reglas creadas, manteniendo pulsada una, debería salir la opción de borrar la regla. De esta forma deberíamos dejar de recibir notificaciones y de estar listada.
Importancia	Secundario
Éxito	SI

Tabla 59. TEST11- Borrar una regla

TEST12	**Verificar que las notificaciones se envían correctamente según la configuración.**
Descripción	Damos de alta y añadimos el recurso temperatura a favoritos. Ahora en la pantalla reglas: pulsamos el botón nueva regla, seleccionamos el recurso de la lista de favoritos y rellenamos el formulario. En dicho formulario configuramos el umbral en 6000 y 20 s y 50s, de forma que cuando tapemos el sensor de luz, recibamos cada 20 y cuando no cada 50s una notificación.
Importancia	Secundario
Éxito	SI

Tabla 60. TEST12- Verificar que las notificaciones se envían correctamente según la configuración.

9 CONCLUSIONES

Grandes descubrimientos y mejoras implican
invariablemente la cooperación de muchas mentes.

- Alexander Graham Bell -

Con este último capítulo expondremos las conclusiones y experiencias obtenidas con el desarrollo del proyecto. Además de marcar posibles pautas/caminos de mejora, de cara a seguir desarrollando el proyecto o relacionados que lo usen como punto de partida.

9.1 CONCLUSIONES

Este trabajo de final de grado me ha ayudado a entender que nada es "anecdótico"; cualquier problema que quieras enfrentar, puede acarrear una serie de imprevistos derivados. Así algo que en un principio estimabas en un par de horas, pueden terminar siendo días. Y eso suponiendo que no tengas que retroceder sobre tus pasos y encontrar un nuevo camino de actuación.

Además, cuando aplicamos esto a una tecnología en desarrollo como es IoTivity, donde la información y ayuda de la que dispones es muy limitada, la exigencia a nivel personal y autosuficiencia aumenta exponencialmente.

Aún con estos inconvenientes, he de admitir que la experiencia ha sido muy satisfactoria. Ya que conforme conseguía unir las piezas que suponía este "puzzle", mis habilidades y conocimientos también iban mejorando. Cosas que creo serán muy útiles para el día de mañana en el ámbito laboral.

Haciendo hincapié desde un punto de vista más técnico sobre el tema a desarrollar, Internet de las cosas. Me gustaría destacar el potencial que tiene como concepto para hacer nuestro día a día más cómodo, pero a la vez el gran problema al que debe afrontar ya desde sus primeros días.

Ya que al ser un mercado en auge y con mucha popularidad, cada compañía y/u organización pretenderá establecer su propias normas, software e infraestructuras. Pero debido al propio concepto de IoT, que busca la intercomunicación de dispositivos con dispositivos, sin importar características físicas, de software o de red. Hace necesario, como ya plantea la OCF y pone en práctica IoTiviy, establecer normas y códigos abiertos coolaborativos. Con el fín de que poco a poco y entre todos, establezcamos un nuevo entorno de desarrollo común que nos permita seguir avanzando hacia ese "mundo intercomnectado de dispositivos". Donde jugaremos un papel de observador.

9.2 POSIBLES MEJORAS

Como se ha comentado, IoTivity es una tecnología en desarrollo: constantemente va aumentando sus funcionalidades y actualiza periódicamente sus especificaciones según la OCF.

Si a eso le sumamos que este trabajo es una primera aproximación a dicha tecnología. se ha tenido que aprender a utilizar herramientas y dispositivos sobre la marcha, a la vez que se profundizaba en el tema central. Quedan una serie de mejoras a futuro. Entre las que considero más importantes:

- Añadir más variedad de recursos y dispositivos. Actualmente ya hay una gran gama de dispositivos y recursos en las especificaciones de la OCF [7] [8]. Y es una lista que no para de aumentar, ya que es una tecnología que se está usando cada vez en más ámbitos.
- Añadir nuevas funcionalidades (IoT). El framework tiene mucho más potencial del que se ha podido trabajar en este proyecto [9]. Por ejemplo, tiene funciones para agrupar recursos y dispositivos. Detectar la presencia de dispositivos IoT sin realizar una búsqueda. Establecer puentes entre diferentes framework o realizar operaciones directamente sobre los grupos mencionados.

- Mejorar la seguridad de la comunicación. Uno de los retos que plantea el IoT y las comunicaciones en general, es la necesidad de mejorar en términos de seguridad. Así unas de las principales mejoras que puede implementarce, sería un sistema de autentificación y control de acceso de recursos [14]. Así como mejorar la encriptación de la comunicación entre dispositivos con certificados y llaves.

- Realizar un rediseño de la interfaz de usuario. Entrando en un aspecto más de usuario, la aplicación tiene un diseño muy simple y poco vistoso de cara al usuario final. Por lo que otro de las posibles mejoras pasa por realizar un rediseño de toda la interfaz y apariencia, para proporcionar una aplicación más “comercial” .

- Experimentar con dispositivos IoT comerciales. Por último, sería interesante investigar la compatibilidad entre tecnologías y dispositivos presentes en el mercado, de diferentes tipos como se planteó en la sección 1.4, con IoTivity. Para ver hasta que punto funciona la propuesta que ofrece su framework y las especificaciones de la OCF al ambito de la IoT.

BIBLIOGRAFÍA

[1] K. L. Lueth, «IOT ANALYTICS,» [En línea]. Available: https://iot-analytics.com/internet-of-things-definition/.

[2] Anónimo, «Wikipedia,» [En línea]. Available: https://en.wikipedia.org/wiki/Internet_of_things#History.

[3] K. D. Foote, «DATAVERSITY,» [En línea]. Available: https://www.dataversity.net/brief-history-internet-things/#.

[4] IoTivity, «IoTivity,» [En línea]. Available: https://iotivity.org/about.

[5] OCF, «OPEN CONNECTIVITY FOUNDATION,» [En línea]. Available: https://openconnectivity.org/wp-content/uploads/2019/03/2.-OCF-Architecture-Introduction.pdf.

[6] OCF, «OPEN CONNECTIVITY FOUNDATION,» [En línea]. Available: https://openconnectivity.org/wp-content/uploads/2019/03/4.-IoTivity-Overview.pdf.

[7] OCF, «OPEN CONNECTIVITY FOUNDATION,» [En línea]. Available: https://openconnectivity.org/specs/OCF_Resource_Type_Specification_v2.0.4.pdf.

[8] OCF, «OPEN CONNECTIVITY FOUNDATION,» [En línea]. Available: https://openconnectivity.org/specs/OCF_Device_Specification_v2.0.4.pdf.

[9] IoTivity, «IoTivity Wiki,» [En línea]. Available: https://wiki.iotivity.org/how_to_use_gerrit.

[10] IoTivity, «IoTivity Wiki,» [En línea]. Available: https://wiki.iotivity.org/android_build_instructions.

[11] Scons, «Scons,» [En línea]. Available: https://www.scons.org/doc/production/HTML/scons-user.html#chap-build-install.

[12] QUENTIN, «Tengio,» [En línea]. Available: https://www.tengio.com/blog/android-things-for-beginners/.

[13] ameliacv, «GitHub,» [En línea]. Available: https://gist.github.com/ameliacv/1ee1c33bebaac0084fea1aef627981f9.

[14] OCF, «OPEN CONNECTIVITY FOUNDATION,» [En línea]. Available: https://openconnectivity.org/wp-content/uploads/2019/03/3.-Security-Introduction.pdf.

[15] M. Şimşek, «medium,» [En línea]. Available: https://medium.com/mobiwise-blog/unsatisfiedlinkerror-problem-on-some-android-devices-b77f2f83837d.

[16] D. Panchal, «stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/33666071/android-marshmallow-request-permission.

[17] K. Rajput, «stackoverflow,» [En línea]. Available:

https://stackoverflow.com/questions/42687607/application-installation-failed-in-android-studio.

[18] OCF, «OPEN CONNECTIVITY FOUNDATION,» [En línea]. Available: https://openconnectivity.org/developer/specifications.

[19] IoTivity, «IoTivity,» [En línea]. Available: https://iotivity.org/documentation/linux/programmers-guide/finding-resource.

[20] IoTivity, «IoTivity,» [En línea]. Available: https://iotivity.org/documentation/linux/programmers-guide/registering-resource.

[21] IoTivity, «IoTivity,» [En línea]. Available: https://iotivity.org/documentation/linux/programmers-guide/querying-resource-state-get.

[22] IoTivity, «IoTivity,» [En línea]. Available: https://iotivity.org/documentation/linux/programmers-guide/setting-resource-state-put.

[23] IoTivity, «IoTivity,» [En línea]. Available: https://iotivity.org/documentation/linux/programmers-guide/observing-resource-state-observe.

[24] Google, «Developers,» [En línea]. Available: https://developer.android.com/things/hardware/raspberrypi.

[25] Google, «Developers,» [En línea]. Available: https://developer.android.com/guide/topics/ui/dialogs?hl=es-419.

[26] Google, «Developers,» [En línea]. Available: https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=es-419.

[27] Google, «Developers,» [En línea]. Available: https://developer.android.com/guide/topics/ui/menus?hl=es-419#context-menu.

[28] Google, «Developers,» [En línea]. Available: https://developer.android.com/guide/topics/ui/controls/spinner.html.

[29] Hackeandoelgenoma, «Hackeandoelgenoma,» [En línea]. Available: https://hackeandoelgenoma.com/2014/08/conectar-android-con-arduino-por-usb/#Aplicacin-Android.

[30] J. Revelo, «Hermosa Programación,» [En línea]. Available: http://www.hermosaprogramacion.com/2015/08/tutorial-layouts-en-android/.

[31] advantej, «stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/6495898/findviewbyid-in-fragment.

[32] CommonsWare, «stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/5766609/save-internal-file-in-my-own-internal-folder-in-android.

[33] T. Dao, «stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/16425146/runonuithread-in-fragment.

[34] enrique7mc, «stackoverflow,» [En línea]. Available: https://es.stackoverflow.com/questions/2464/c%C3%B3mo-iterar-a-trav%C3%A9s-de-un-hashmap.

[35] E. Friedman-Hill, «stackoverflow,» [En línea]. Available:

https://stackoverflow.com/questions/10930624/creating-json-objects-directly-from-model-classes-in-java/10930794#10930794.

[36] nicopasso, «stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/31367599/how-to-update-recyclerview-adapter-data.

[37] A. Savin, «stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/26245139/how-to-create-recyclerview-with-multiple-view-type.

[38] H. Shah, «stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/26466877/how-to-create-context-menu-for-recyclerview.

[39] Suragch, «stackoverflow,» [En línea]. Available: https://stackoverflow.com/questions/40584424/simple-android-recyclerview-example.

[40] B. Khan, «SIMPLIFIED CODING,» [En línea]. Available: https://www.simplifiedcoding.net/bottom-navigation-android-example/.

[41] Elegoo, «Elegoo,» [En línea]. Available: http://www.elegoo.com.

[42] Google, «Google Cloud,» [En línea]. Available: https://cloud.google.com/solutions/iot/?hl=es.

[43] OCF, «OPEN CONNECTIVITY FOUNDATION,» [En línea]. Available: https://openconnectivity.org/consumer.

[44] J. M. Aguilar, «variablenotfound,» [En línea]. Available: https://www.variablenotfound.com/2008/02/101-citas-clebres-del-mundo-de-la.html.

ANEXOS I – RASPBERRY PI 3 Y ANDROIDTHING

A continuación, se detallará el proceso de instalar Android Thing en una raspberry pi 3 y su posterior configuración. Además de como conectarla a Android Studio.

Antes de nada, es necesario:

-Un cable USB-micro USB

-Cable Ethernet

-Lector de SD o MicroSD

-Tarjeta microSD de 8gb o más

-Cable HDMI

Y descargar "Android Things Setup Utility" desde:

https://partner.android.com/things/console/#/tools

NOTA: hay que iniciar sesión con una cuenta de Google y aceptar los términos de servicio.

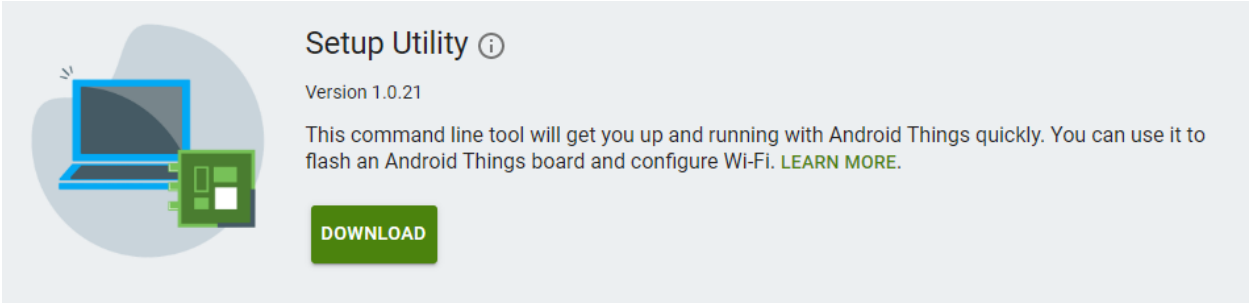


Ilustración 70. Android Thing montaje 01

A continuación, ejecutamos la aplicación como administrador, clic derecho en Windows o el siguiente comando en Mac/Linux:

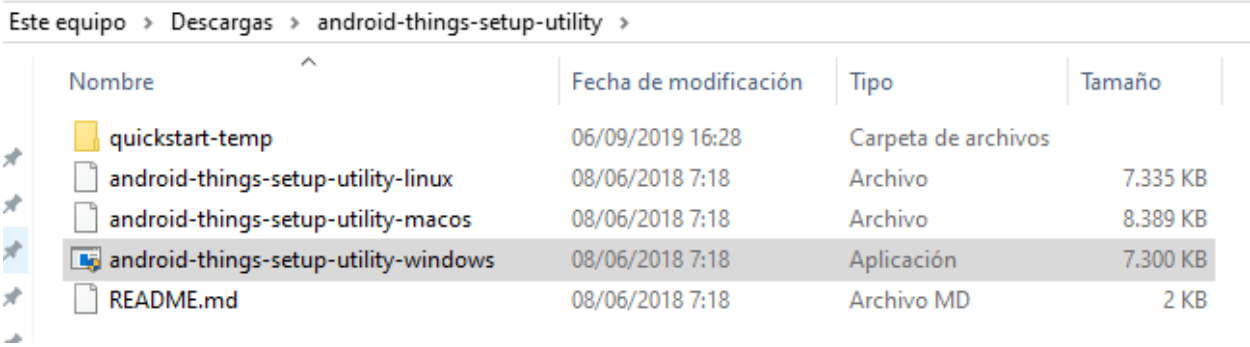


Ilustración 71. Android Thing montaje 02

```
$ sudo ~/Downloads/android-things-setup-utility/android-things-setup-
utility-linux
```

Ahora seguiremos los siguientes pasos:

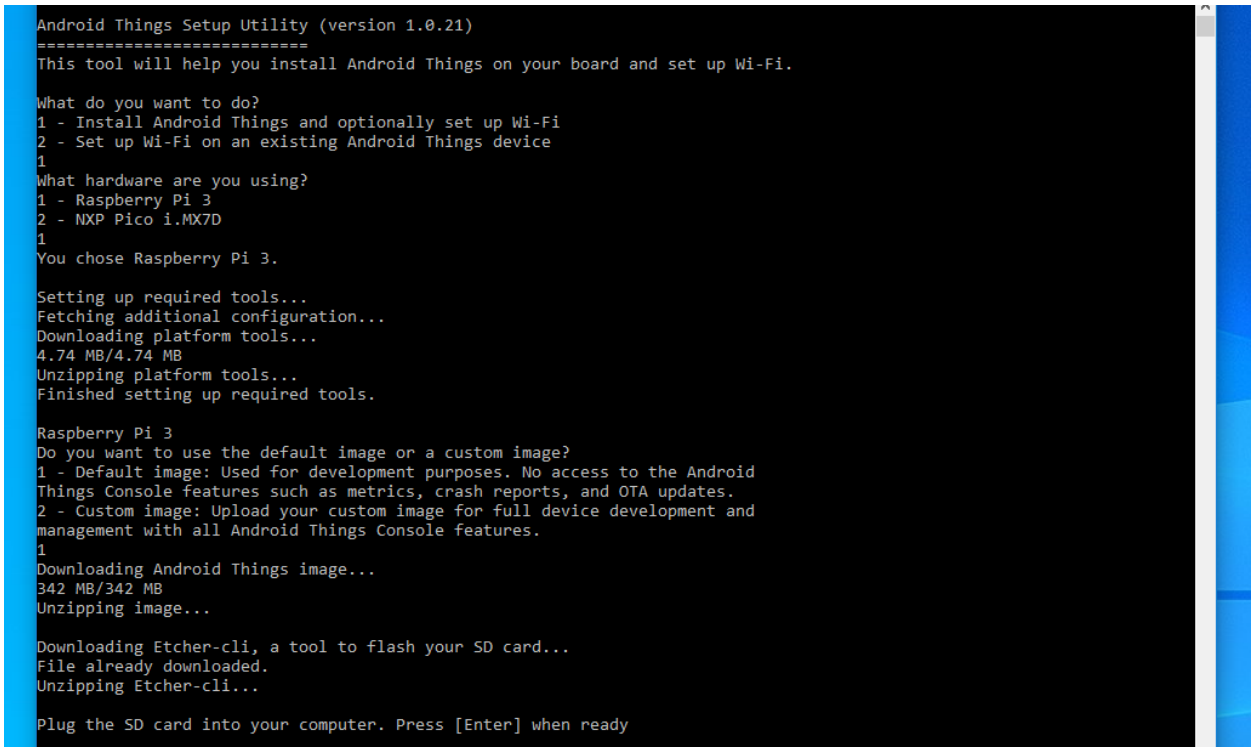


Ilustración 72. Android Thing montaje 03

Introducimos la microSD y pulsamos “intro”:

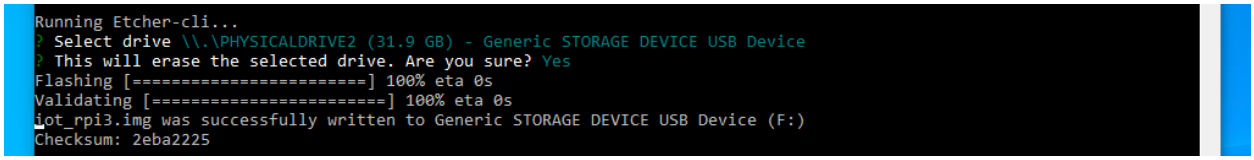


Ilustración 73. Android Thing montaje 04

Una vez terminemos podemos pasar directamente a la configuración vía wifi o salir del programa:

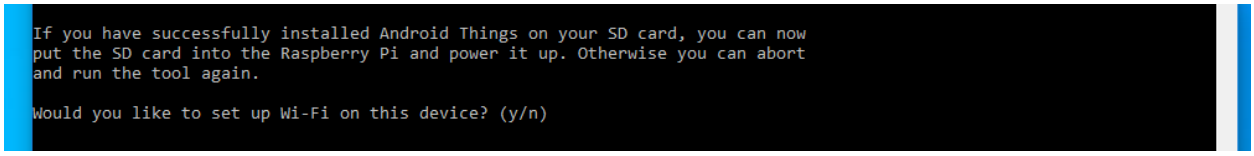


Ilustración 74. Android Thing montaje 05

NOTA: En esta ejecución usamos una ISO estándar que se descarga sobre la marcha, en caso contrario, pulsar la opción dos en la tercera pregunta.

Para configurar el Wifi de la raspberry pi 3 podemos hacerlo vía interfaz gráfica conectándola a una pantalla o mediante la herramienta anterior. Solo tenemos que seleccionar la opción dos como se muestra en la zona marcada en amarillo en la figura X, conectarlo al router por ethernet y seguir los pasos indicados, zona marcada en rojo de la figura X.

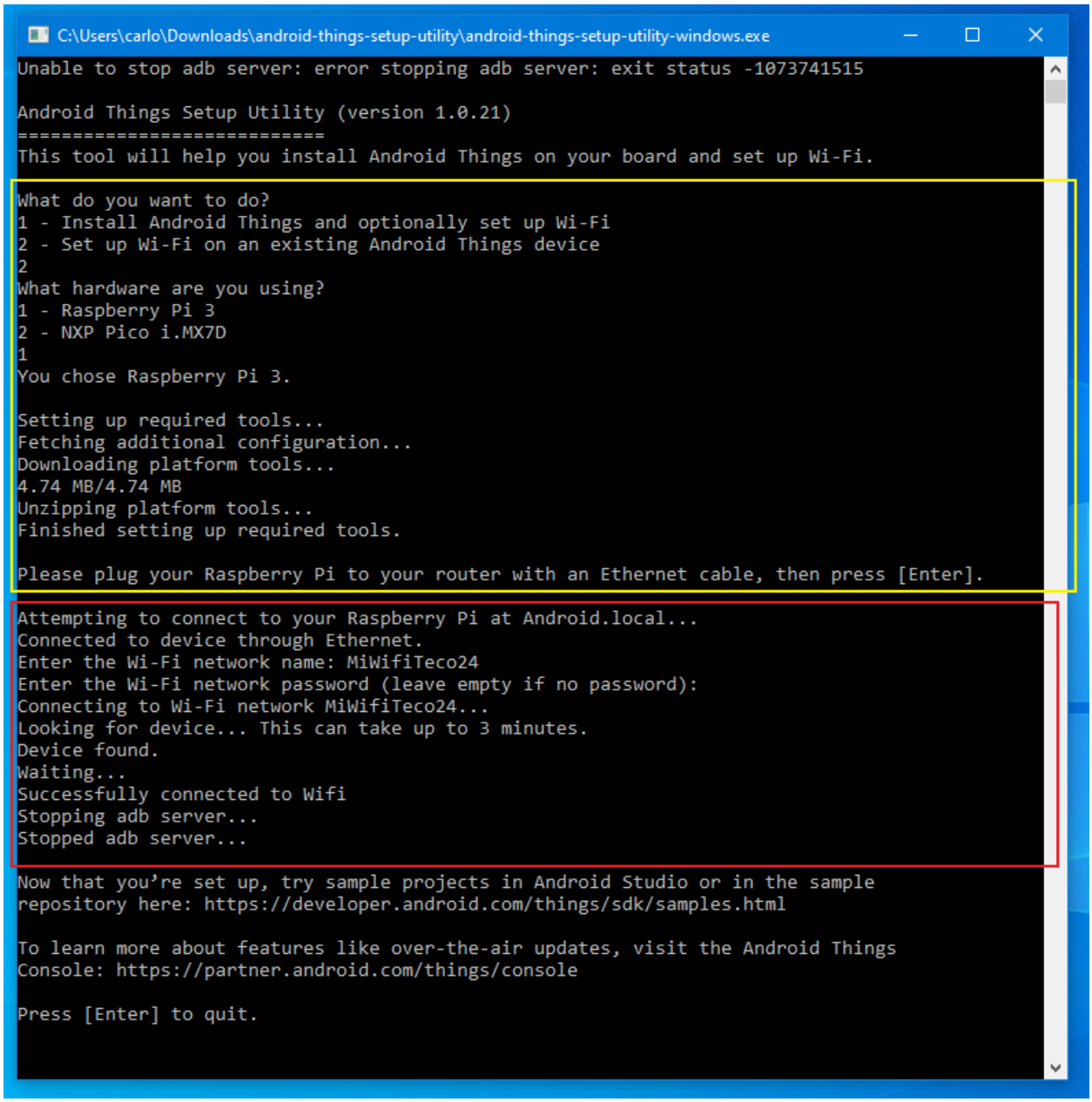


Ilustración 75. Android Thing montaje 06

Con esto el dispositivo tendrá una ip local con la que podremos vincularla a Android Studio. Primero miramos las ip asociadas desde la interfaz gráfica o router:

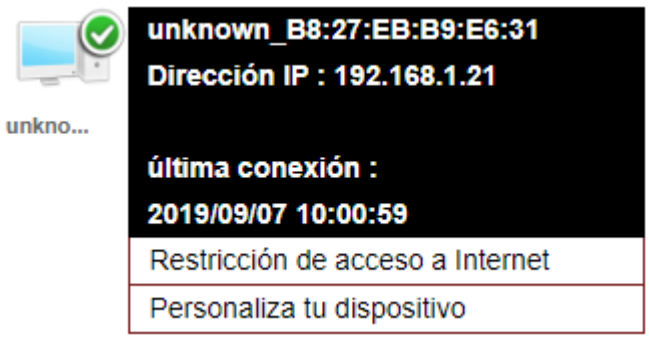


Ilustración 76. Ip Raspberry Pi 3

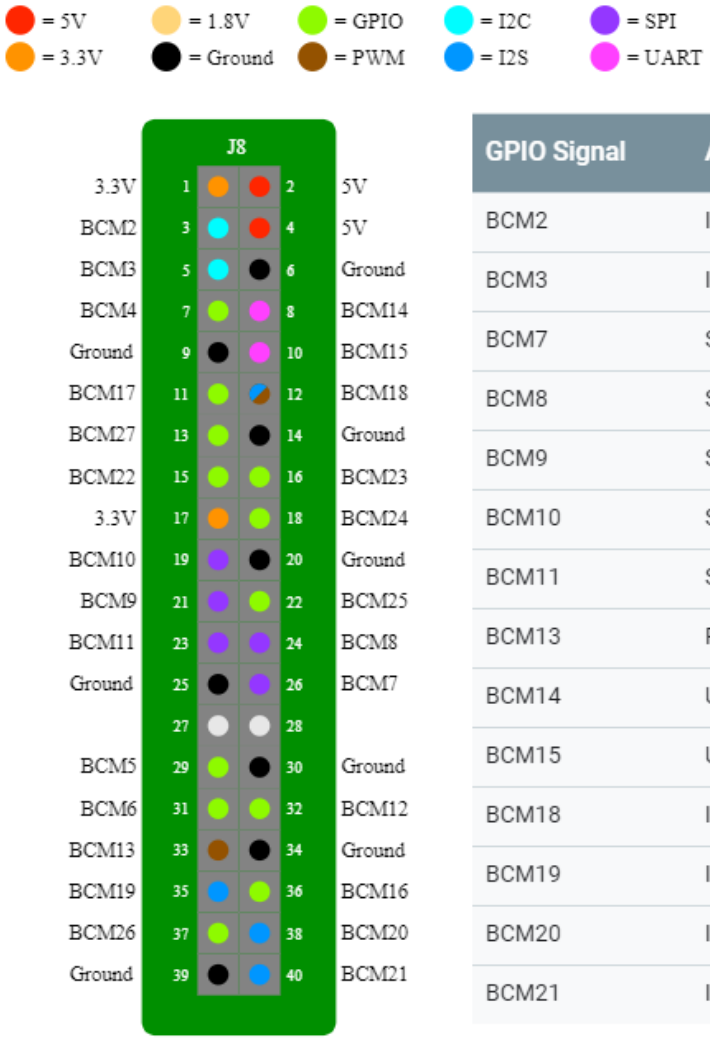
Y desde el terminal ejecutamos:

```
$ adb connect <ip-address>:<port> o $ adb connect Android.local
C:\platform-tools\adb connect 192.168.1.21:5555 %ejemplo real
```

Si queremos descargar las herramientas del SDK por separado:

https://developer.android.com/studio/releases/platform-tools.html

Para terminar, se adjunta un resumen de los pines de entrada y salida de raspberry pi 3 y su nombre en Android Thing:



GPIO Signal	Alternate Functions	
BCM2	I2C1 (SDA)	
BCM3	I2C1 (SCL)	
BCM7	SPI0 (SS1)	
BCM8	SPI0 (SS0)	
BCM9	SPI0 (MISO)	
BCM10	SPI0 (MOSI)	
BCM11	SPI0 (SCLK)	
BCM13	PWM1	
BCM14	UART0 (TXD)	MINIUART (TXD)
BCM15	UART0 (RXD)	MINIUART (RXD)
BCM18	I2S1 (BCLK)	PWM0
BCM19	I2S1 (LRCLK)	
BCM20	I2S1 (SDIN)	
BCM21	I2S1 (SDOUT)	

Ilustración 77. Esquema pines Raspberry Pi 3

Referencias:

ANEXOS II – ARDUINO Y SENSORES

En este anexo vamos a explicar cómo obtener y configurar el IDE de Arduino. Además de aportar el código que se ejecutará en ella para recrear nuestro escenario.

Primero descargaremos el IDE del siguiente enlace y lo instalaremos:

https://www.arduino.cc/en/Main/Software

Nota: el instalador trae un tutorial detallado para diferentes SO y posibles problemas.

Una vez instalado lo ejecutaremos y añadiremos las librerías necesarias. En nuestro caso tenemos un .zip proporcionado por ELEEGO con librerías y ejemplos de programas para todos los sensores que trae el kit de sensores. Pero tiene su propio gestor de librerías incorporado, ampliable y actualizable por internet.

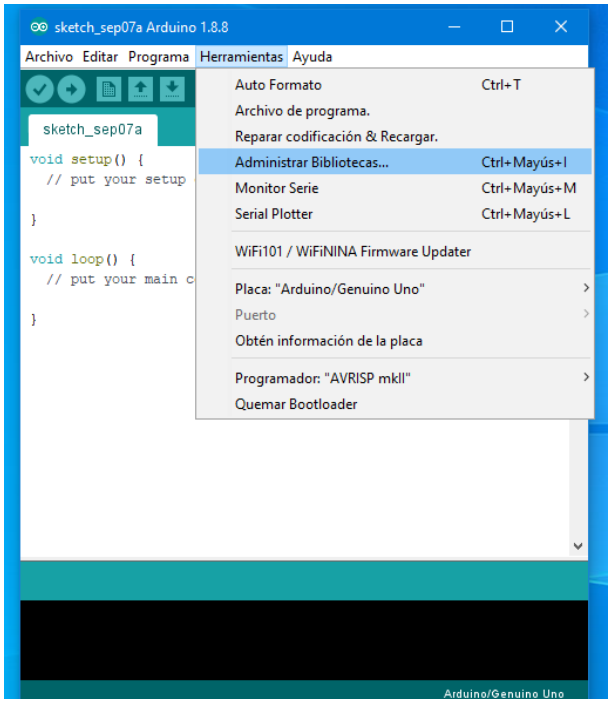


Ilustración 78. Añadir librería a Arduino 01

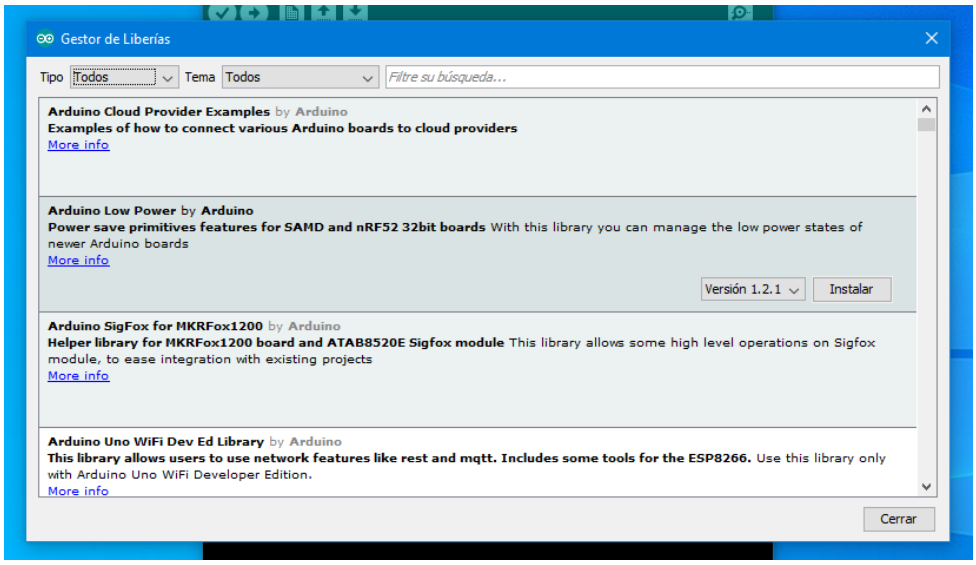


Ilustración 79. Añadir librería a Arduino 02

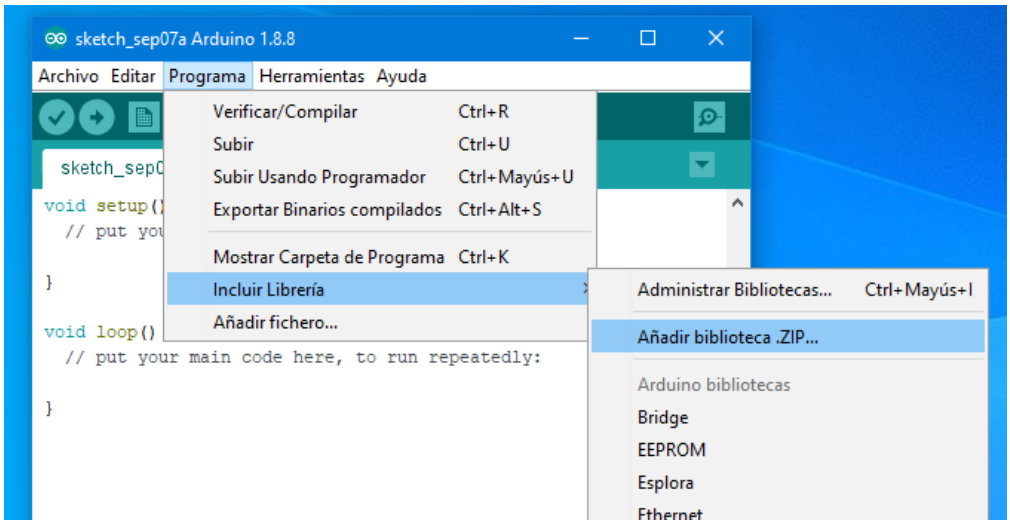


Ilustración 80. Añadir librería a Arduino 03

Como alternativa, ya que es más rápido, también puedes copiar directamente las librerías en el directorio adecuado. El siguiente ejemplo muestra la ruta por defecto en Windows 10 64-bits.

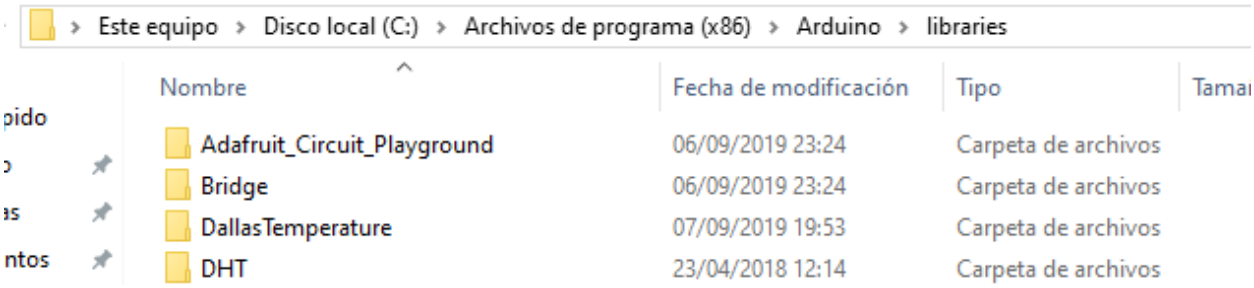


Ilustración 81. Añadir librería a Arduino 04

Para terminar, se adjunta el programa diseñado para la lectura de los sensores y su transferencia por el puerto serie (USB). También se adjunta una ilustración con el botón “subir” que nos permite grabar el programa en el Arduino.

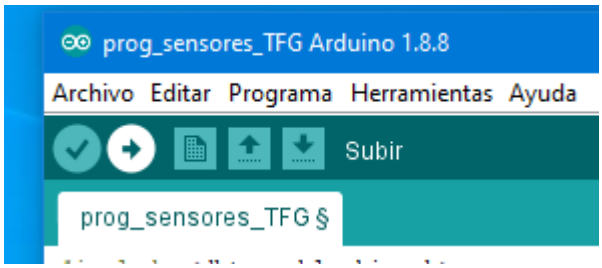


Ilustración 82. Grabar programa en Arduino

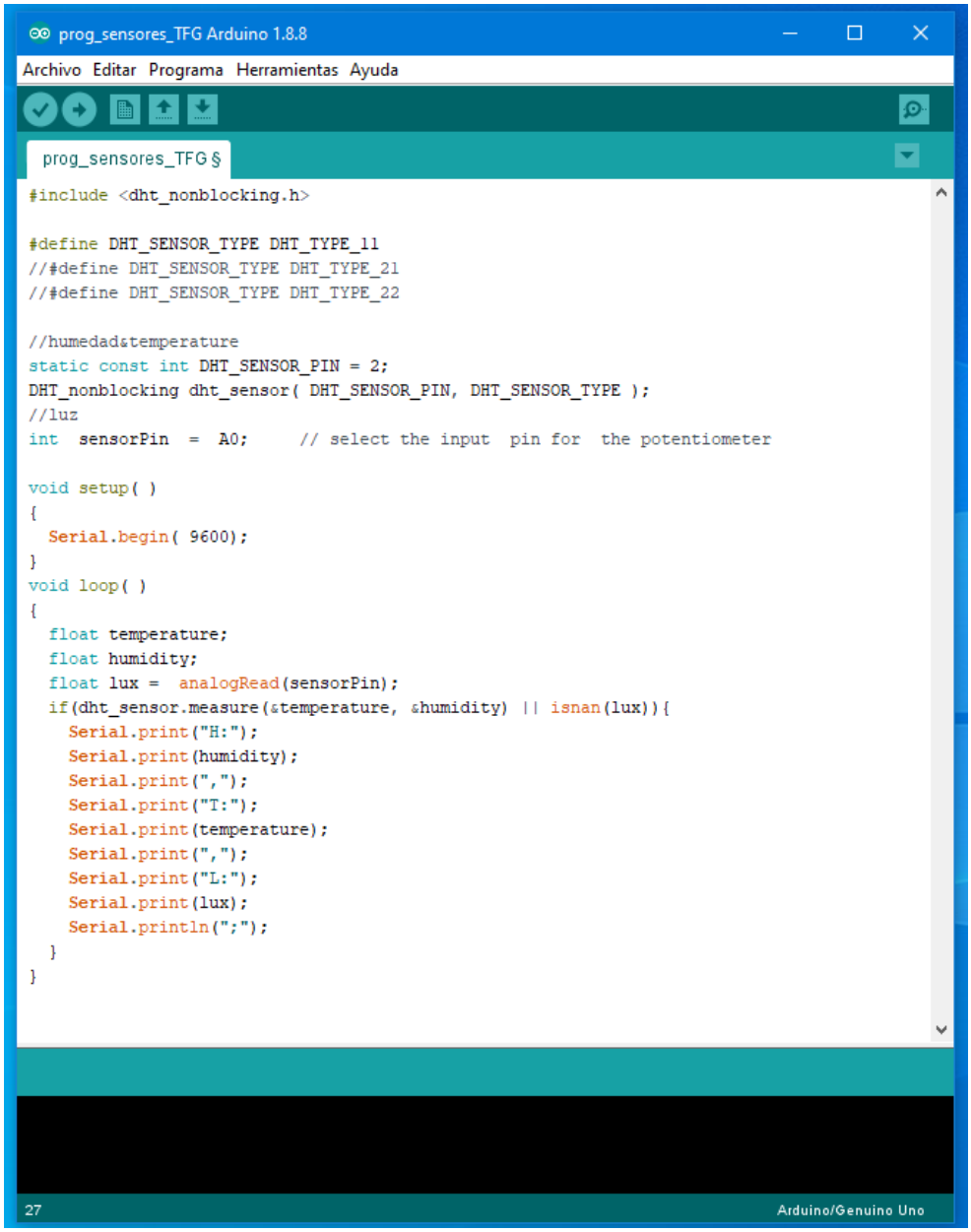


Ilustración 83. Código del programa a grabar en el arduito para el montaje del servidor.

ANEXOS III – MONTAJE ESCENARIO

En este anexo vamos a recopilar las diferentes secciones donde se explica cómo montar y configurar los diferentes dispositivos y herramientas software. De modo que siguiendo los pasos de forma secuencial se pueda recrear el escenario completo. Antes de nada, se adjunta en la ilustración 84 como sería el esquema final del escenario.

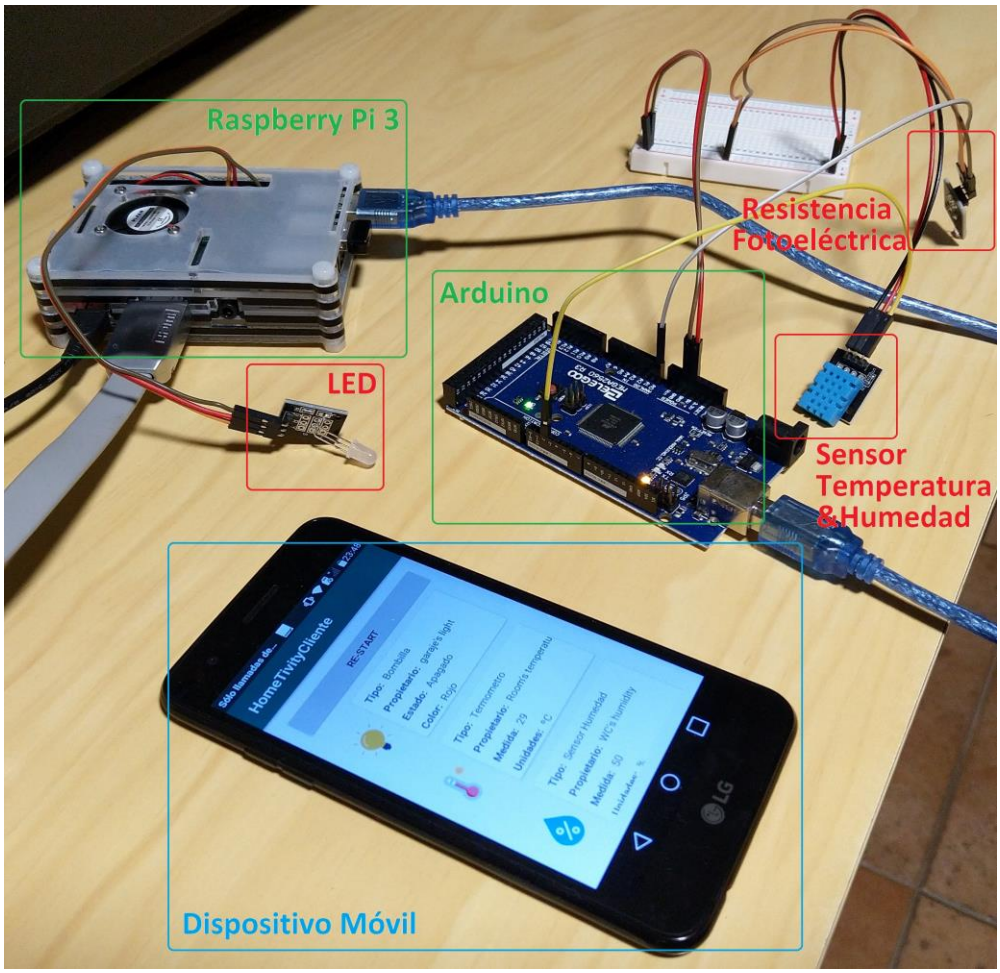


Ilustración 84. Escenario completo del proyecto.

Primero vamos a montar el núcleo de lo que sería el servidor, formado por la Raspberry Pi 3 y el Arduino:

-Seguimos el proceso descrito en el ANEXO I para instalar Android Thing en la Raspberry Pi 3.

-Ahora hacemos lo mismo con el ANEXO II para grabar el programa de lectura y comunicación de los sensores leídos.

-Por último, en este primer paso, conectamos los dispositivos entre sí como está descrita en la sección 6.1.

De modo que actualmente tendríamos el escenario representado en la ilustración 84 a falta de instalar la aplicación desarrollada con el framework IoTivity en los terminales Móvil y Raspberry Pi 3.

Para el desarrollo de las aplicaciones lo primero que se hará es obtener el proyecto IoTivity de su Gerrit y realizar el compilado de las librerías, además de crear el proyecto de Android studio y configurarlo adecuadamente como se explica en la sección 3.5.

Para terminar, solo tendríamos que crear nuestra aplicación cliente y servidor e instalarlas en sus respectivos dispositivos. Hay que recordar que con el siguiente comando se vincula la Raspberry Pi 3, ejecutando Android Thing, a Android Studio:

```
$ adb connect <ip-address>:<port> o $ adb connect Android.local
C:\platform-tools\adb connect 192.168.1.21:5555 %ejemplo real
```

NOTA: En el CD proporcionado están ambos proyectos, servidor y cliente, para Android Studio. Con todo el código utilizado para realizar las aplicaciones HomeTivity Servidor y Cliente.

ANEXOS IV – PROBLEMAS/SOLUCIONES

Error “couldn't find "libgnustl_shared.so”

-Causa: No se están exportando las librerías correctamente según la arquitectura del terminar.

-Solución: Exportar manualmente las librerías [15]

Abrir el siguiente archivo:

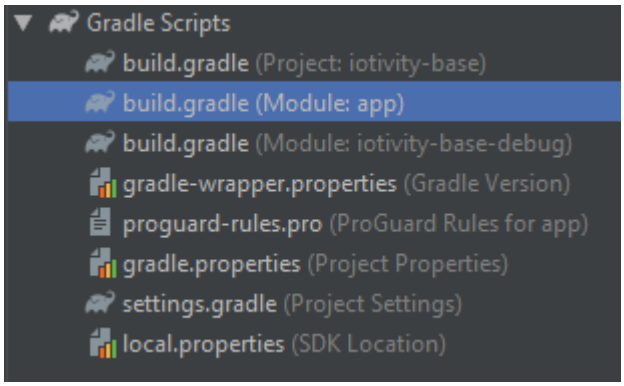


Ilustración 85. Localización de gradle para solucionar error

Añadir:

```
android {
....
defaultConfig {
   ....
   ndk {
      abiFilters "armeabi", "armeabi-v7a", "x86"
   }
}
```

Activar autoarranque de servidor

Copiar dentro del Android Manifest.xml,m a continuación del activity principal:

```
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />


            <category android:name="android.intent.category.HOME" />
            <category android:name="android.intent.category.DEFAULT" />
        </intent-filter>
```

Permisos de escritura y lectura para Android 6.0 o superior

-Causa: A partir de dicha versión de Android es necesario que el usuario confirme manualmente los permisos que usará la aplicación, además de añadirlo en el AndroidManifest.

-Solución: Implementaremos los siguientes fragmentos de código [15]

```
ActivityCompat.requestPermissions(MainActivity.this,
            new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},
            1);
```

```
@Override
public void onRequestPermissionsResult(int requestCode,
                        String permissions[], int[] grantResults) {
  switch (requestCode) {
    case 1: {

      // If request is cancelled, the result arrays are empty.
      if (grantResults.length > 0
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

          // permission was granted, yay! Do the
          // contacts-related task you need to do.
       } else {

          // permission denied, boo! Disable the
          // functionality that depends on this permission.
          Toast.makeText(MainActivity.this, "Permission denied to read your External storage",
Toast.LENGTH_SHORT).show();
       }
       return;
    }
  }
}
```

Installation failed with message Failed to establish session.

-Causa: No se puede establecer la conexión con el dispositivo desde AndroidStudio para instalar la APK.

-Solución: Desactivar Instant Run en AndroidStudio, para ello seguir los siguientes pasos [17]

File > Settings > Build,Execution,Deployment > Instant Run > Un-check (Enable Instant Run to hot swap code)

Instalar APK desde ADB

Si queremos instalar una APK directamente, debes de buscar la ruta donde tenemos el ejecutable **adb** e introducir el siguiente commando con las rutas adecuadas. A continuación se aporta un ejemplo:

```
C:\platform-tools\adb install C:\Users\carlo\Documents\Universidad\TFG\armeabi\examples-android/simpleserver/build/outputs/apk/simpleserver-armeabi-debug.apk
```