Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

ÁREA DE INGENIERÍA TELEMÁTICA

TRABAJO FIN DE MÁSTER Nº 201817

EVALUACIÓN DE LA TECNOLOGÍA WI-FI DIRECT EN ESCENARIOS DE MOVILIDAD

D. MIRAVALLES FUENTES, DANIEL

TUTORES: D. GARCIA FERNANDEZ, ROBERTO
D. SANCHEZ SANCHEZ, JOSE ANTONIO

FECHA: JULIO 2018

RESUMEN

En la actualidad cada vez son más comunes las comunicaciones entre dispositivos en cualquier tipo de ámbito, aumentando esta tendencia cada vez más en ciertos sectores como el de las comunicaciones vehiculares. Para llevar a cabo el intercambio de información en este ámbito se suele utilizar WAVE. Este estándar permite comunicaciones entre vehículos o entre vehículos e infraestructuras, pero para implementar esta tecnología se requiere de dos transceptores que encarecen en gran medida su uso. Es por ello que mediante el presente proyecto se pretende proporcionar una alternativa accesible a través de la cual pueda llevarse a cabo una comunicación entre vehículos.

Esta solución pasa por el estudio de la tecnología Wi-Fi Direct en escenarios de movilidad. Esta tecnología permite el intercambio de información entre dispositivos sin la necesidad de utilizar un punto de acceso, sino que uno de ellos adquirirá dicho rol. Esta característica resulta de gran utilidad, ya que no se depende de una conexión a Internet vía Wi-Fi o datos móviles para llevar a cabo el intercambio de información. Además Wi-Fi Direct está presente en la mayor parte de dispositivos inteligentes que hay hoy en día en el mercado. Esto es un punto a favor en cuanto a la accesibilidad de los usuarios a esta tecnología, ya que actualmente existe un incremento continuo del número de smartphones y dispositivos inteligentes de todo tipo. Por este motivo se ha escogido el desarrollo de una aplicación basada en el sistema operativo Android para llevar a cabo la parte práctica de este estudio. Esta aplicación será meramente una herramienta a través de la cual conseguir el objetivo principal del proyecto que no es otro que la evaluación de la tecnología en entornos vehiculares.

Para llevar a cabo el desarrollo de la aplicación será necesario realizar un estudio previo de esta tecnología, primero a nivel teórico, para conocer sus funcionalidades y, posteriormente, a nivel práctico para obtener un análisis de la misma en un entorno vehicular. Es por ello que será necesario realizar un primer acercamiento al entorno de desarrollo de este sistema operativo. También resultará fundamental conocer las librerías disponibles en Android para el uso de esta tecnología y que permitirán realizar tareas como el descubrimiento y conexión de peers.

Una vez se dispongan de las herramientas adecuadas para implementar la aplicación se fijaran las pruebas que se desean realizar teniendo en cuenta diferentes parámetros que permitan obtener información relevante para su posterior análisis. Para llevar a cabo estas pruebas se deberá escoger un emplazamiento que se adecúe a los requisitos de las mismas. Por último, tras realizar el análisis de esta tecnología se extraerán una serie de conclusiones a través de las cuales se pueda comprender su funcionamiento y utilidad en entornos vehiculares.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

LISTA DE ACRÓNIMOS

ADB	Android Debug Bridge
ADT	Android Developer Tools
API	Application Programming Interface
APK	Android Application Package
ART	Android runtime
AVD Manager	Android Virtual Device Manager
CET	Connection Establishment Time
CPU	Central Processing Unit
DHCP	Dynamic Host Configuration Protocol
DLNA	Digital Living Network Alliance
DSRC	Dedicated Short Range Communications
GO	Group Owner
HAL	Hardware Abstraction Layer
IEEE	Institute of Electrical and Electronics Engineers
IDE	Integrated Development Environment
HDMI	High-Definition Multimedia Interface
IP	Internet Protocol
Java SE	Java Standard Edition
JDK	Java Development Kit
JRE	Java Runtime Environment
Li-Fi	Light Fidelity
LTE	Long Term Evolution

MTU	Maximum Transmission Unit
NFC	Near Field Communication
OBU	On-Board Unit
P2P	Peer-to-Peer
P2P GO	Peer-to-Peer Group Owner
QR	Quick Response
RSU	RoadSide Unit
SDK	Software Development Kit
Soft Ap	Software enabled Access point
V2I	Vehicle to infrastructure
V2V	Vehicle to vehicle
V2X	Vehicle to everything
WAVE	Wireless Access Vehicular Environments
WLAN	Wireless Local Area Network
WPA2	Wi-Fi Protected Access 2
WPS	Wi-Fi Protected Setup
XML	eXtensible Markup Language

1.- INTRODUCCIÓN

Las comunicaciones entre dispositivos resultan fundamentales en multitud de proyectos y al final siempre son objeto de investigación en el ámbito de las telecomunicaciones. Es por ello que resulta de gran importancia que se vayan actualizando constantemente los protocolos que regulan estas comunicaciones haciendo que sean lo más accesibles y eficientes para todo tipo de usuarios.

1.1.- Motivación

Este proyecto surge con el fin de analizar una tecnología que pueda ser implementada en entornos vehiculares de manera sencilla y eficaz y que además no suponga un coste elevado para el usuario. Actualmente, salvo en países en condiciones de pobreza, la mayor parte de las personas disponen de al menos un dispositivo inteligente y en mayor medida el sistema operativo en el que está basado es Android (ya que cuenta con una mayor oferta de dispositivos) por lo que el uso de esta tecnología no supondría un coste adicional para el usuario.

De esta manera se aporta una alternativa al estándar de comunicaciones vehiculares WAVE el cual supone un coste adicional para el usuario por los transceptores que deben ser utilizados para implementar esta solución. Por su parte, mediante la solución presentada en este proyecto, solo sería necesario disponer de un dispositivo inteligente basado en el sistema operativo Android, y que cuente con la tecnología Wi-Fi Direct, presente en la mayor parte de dispositivos (de los dispositivos probados en este proyecto, todos contaban con esta tecnología, incluidos dispositivos con antigüedad superior a 5 años). Incluso con que solo uno de los dispositivos contase con la tecnología Wi-Fi Direct, sería suficiente para poder llevar a cabo el intercambio de información, siempre y cuando el otro dispositivo dispusiese al menos del interfaz Wi-Fi.

1.2.- Objetivos

El objetivo principal de este proyecto es el de realizar un análisis de la tecnología Wi-Fi Direct en escenarios de movilidad, comprendiendo de esta manera su utilidad así como definiendo los factores que influyen en su funcionalidad. Este objetivo se conseguirá a través de una serie de escenarios planteados que se basan en situaciones reales que tienen lugar durante la conducción.

Para llevar a cabo este objetivo habrá que ir cumpliendo otros objetivos como son el de familiarizarse previamente con esta tecnología, así como el desarrollo de una aplicación en Android, el cual simplemente será el medio para conseguir el fin anterior. De esta manera se establecerán unos parámetros a medir, los cuales definirán las características de la tecnología Wi-Fi Direct en este tipo de entornos.

1.3.- Estructura del Documento

En este documento en el **Capítulo 2 "Tecnología Wi-Fi Direct y Estado del Arte",** se comienza realizando una introducción a la tecnología Wi-Fi Direct. Para ello se define en que consiste esta tecnología, como tiene lugar la formación de los grupos, así como cuáles son sus características principales y servicios. Posteriormente, se comentan implementaciones de esta tecnología tanto en entornos vehiculares como no vehiculares.

Una vez realizado un primer acercamiento a esta tecnología, en el **Capítulo 3 "Entorno de Desarrollo",** se pasa a introducir el entorno de desarrollo de la misma. Como se ha comentado anteriormente la aplicación está basada en el sistema operativo Android, por lo que se hará una breve introducción al mismo, así como a su relación con Java, y se introducirá el IDE, explicando las diferentes partes que lo componen así como ciertas tareas relevantes como la creación de un proyecto o su ejecución en un dispositivo.

En el **Capítulo 4 "Desarrollo de la Aplicación",** se explicarán las principales partes del desarrollo de la aplicación. Estas son los pasos iniciales para el desarrollo de la misma, la explicación de cómo tiene lugar el descubrimiento y conexión de los peers, el intercambio de información, el uso de ficheros o el apartado de localización de dispositivos que permitirá conocer posteriormente la distancia a la que los dispositivos se desconectan.

Posteriormente en el **Capítulo 5 "Pruebas Realizadas"**, se indicarán una serie de pruebas que se van a llevar a cabo diferenciando entre pruebas iniciales, las cuales permiten una mejor comprensión de la tecnología Wi-Fi Direct, así como las pruebas basadas en entornos vehiculares. Para estas últimas se definirán una serie de escenarios, basados en situaciones reales que se pueden dar durante la conducción, así como los parámetros a medir en las mismas. Ya definidas las pruebas a realizar, en el **Capítulo 6 "Análisis y Discusión de los Resultados"** se indicaran los resultados obtenidos en las mimas y se procederá a obtener una serie de conclusiones para cada escenario

A continuación en el **Capítulo 7 "Planificación",** se comentará la planificación seguida para la realización del proyecto. Posteriormente en el **Capítulo 8 "Presupuesto"** se detallarán los costes del proyecto. Por último, en el **Capítulo 9 "Conclusiones y Trabajos Futuros"** se extraerán una serie de conclusiones finales, indicando también posibles líneas a seguir para la realización de futuros trabajos.

2.- TECNOLOGÍA WI-FI DIRECT Y ESTADO DEL ARTE

La tecnología Wi-Fi Direct permite que dispositivos Wi-Fi se conecten de manera "Directa", permitiendo realizar acciones como por ejemplo sincronizar, imprimir, compartir archivos o internet [1]. Con el término de manera "Directa" se quiere decir que, a diferencia de una conexión Wi-Fi normal, Wi-Fi Direct no necesita un punto de acceso intermedio para la gestión de la comunicación entre los dispositivos, sino que uno de estos dispositivos funcionará como punto de acceso. Es decir, Wi-Fi Direct es capaz de convertir mediante software (Soft AP), un dispositivo con Wi-Fi en un punto de acceso.

La organización sin ánimo de lucro Wi-Fi es la encargada de promover el uso de la tecnología Wi-Fi y certificar la interoperabilidad de productos WLAN basados en la especificación IEEE 802.11. Este organismo, anunció a finales de 2009 la llegada para el año 2010, de una nueva especificación que permitiría soportar conexiones Wi-Fi P2P entre dispositivos y la cual se encargarían de administrar y certificar. Esta especificación, denominada anteriormente Wi-Fi peer-to-peer, pasaría a ser llamada Wi-Fi Direct.

Como se ha comentado anteriormente los dispositivos Wi-Fi Direct se comunican estableciendo conexiones P2P. Mediante estas conexiones se crean grupos donde uno de estos dispositivos actuará como punto de acceso y será conocido como el "Propietario del Grupo P2P" o "Group Owner (GO)", mientras que el resto de dispositivos actuarán como clientes. Es decir, cuando se establece una comunicación por primera vez los dispositivos negocian y se determina cuál de ellos será el punto de acceso. La conexión puede tener lugar entre dos o más dispositivos, sólo siendo necesario que un dispositivo sea compatible con Wi-Fi Direct para formar un grupo. Este dispositivo activa Wi-Fi Direct de manera que los demás dispositivos se conectan a su Wi-Fi de la misma forma que lo harían a un router Wi-Fi. Por lo tanto los otros dispositivos no necesitan ser compatibles con Wi-fi Direct.

2.1.- Formación de Grupos

Existen tres técnicas diferentes para la formación de los grupos [2]:

- Estándar: en primer lugar se descubren los dispositivos P2P. Para ello se realiza un escaneo de la red Wi-Fi mediante el cual se descubren los grupos existentes así como las redes Wi-Fi. Posteriormente se negocia que dispositivo actuará como P2P GO. Con el fin de que los dispositivos se pongan de acuerdo sobre cual actuará como GO, éstos enviarán un valor numérico, el GO Intent, de tal manera que el dispositivo que declare el valor más alto se convierte en el GO. En caso de que éste valor coincida, existe un bit de desempate que se asigna aleatoriamente cada vez que se envía una solicitud de negociación GO.

- Autónoma: un dispositivo puede crear autónomamente un grupo P2P, convirtiéndose automáticamente en el GO. Otros dispositivos podrían descubrir el grupo mediante el escaneo de la red. En este caso no es necesaria llevar a cabo ninguna negociación entre dispositivos para elegir el GO porque ya se impone inicialmente.
- Persistente: durante el proceso de formación del grupo, los dispositivos P2P pueden declarar a un grupo como persistente usando un flag presente en las tramas Beacon, (engloba información sobre la red inalámbrica y son transmitidas periódicamente para anunciar la presencia de redes WLAN) y en las tramas de sondeo y de negociación del GO. De esta forma, los dispositivos que forman el grupo almacenan las credenciales de red y los roles del GO y de cliente asignados para volver a instanciar el grupo P2P. Por tanto, después de la fase de descubrimiento, si un dispositivo P2P reconoce haber formado un grupo persistente con el correspondiente par en el pasado, se puede utilizar la información recopilada anteriormente para volver a instanciar rápidamente el grupo.

Una vez se han descubierto los dispositivos y asignado los roles el siguiente paso será el de crear una comunicación segura mediante WPS. Este estándar está basado en WPA2, un sistema que protege las conexiones presentes en las redes Wi-Fi. Estas conexiones podrán llevarse a cabo para un solo intercambio de información, o con el fin de conservarlas en memoria, pudiendo vincularlas cada vez que los dispositivos se encuentren próximos. Por último el dispositivo que actúa como GO, tendrá también el papel de ser el servidor DHCP, asignando direcciones IP a los correspondientes clientes P2P.

Con el grupo ya establecido, los clientes pueden unirse mediante la red Wi-Fi tradicional. Cabe destacar que sólo el propietario puede conectar los dispositivos de su grupo a una red externa, además de que Wi-Fi Direct no permite transferir dicho rol a otro dispositivo del grupo. Por tanto, si el propietario del grupo P2P sale del grupo, entonces el grupo se descompone y tiene que volver a establecerse.

Las conexiones de dispositivos con esta tecnología pueden llevarse a cabo en cualquier lugar, incluso cuando no hay acceso a una red Wi-Fi. Los dispositivos Wi-Fi Direct emiten una señal a otros dispositivos del área, haciéndoles saber que se puede establecer una conexión. Los usuarios pueden ver los dispositivos disponibles y solicitar una conexión, o pueden recibir una invitación para conectarse a otro dispositivo.

2.2.- Características Principales

La tecnología Wi-Fi Direct tiene las siguientes características:

- Admite velocidades Wi-Fi típicas, que pueden llegar a 250 Mbps. Incluso a velocidades más bajas, Wi-Fi proporciona un gran rendimiento para transferir contenido multimedia con facilidad. El rendimiento de un grupo concreto de

dispositivos Wi-Fi Direct depende de los dispositivos por ejemplo si son 802.11g o n, así como de sus características particulares y del entorno físico.

- Cubre grandes distancias de hasta 200 metros, lo cual permite realizar conexiones entre dispositivos que se encuentren muy cerca, pero también por distancias relativamente grandes, lo cual es un punto a favor con respecto a otras tecnologías.
- Una red Wi-Fi Direct puede llevarse a cabo entre dos dispositivos, o entre muchos. El número de dispositivos soportados en un grupo con certificación Wi-Fi Direct es menor que el número admitido por un punto de acceso tradicional. De hecho la conexión con otros dispositivos es una función opcional que no se admite en todos los dispositivos con certificación Wi-Fi Direct, de tal manera que algunos dispositivos sólo podrán realizar conexiones con un dispositivo.
- La tecnología Wi-Fi Direct fue integrada en 2011 en dispositivos Android 4.0 (API 14), por lo que actualmente es soportado también por las versiones posteriores. Esto se produjo gracias a la API de "Wi-Fi Peer to Peer", la cual cumple el programa de certificación Wi-Fi Direct.
- Soporta el descubrimiento de servicios en la capa de enlace. De esta forma, antes del establecimiento de un Grupo P2P, los dispositivos pueden intercambiar solicitudes para conocer los servicios disponibles y, de esta manera decidir si se continúa o no con la formación del grupo. Esto representa un cambio importante con respecto a las redes Wi-Fi tradicionales, en las cuales los clientes solo están interesados en la conectividad a Internet.

Por tanto, con esta tecnología se amplía el concepto que se tenía de la tecnología Wi-Fi y no se reduce simplemente a una conexión a Internet, sino a conectar dispositivos Wi-Fi entre sí independientemente del lugar y sin la necesidad de dicha conexión a Internet. Esto permite que se pueda utilizar para todo tipo de aplicaciones como por ejemplo compartir contenido, sincronizar datos, jugar, reproducir audio y vídeo sin falta de tener acceso a Internet.

2.3.- **Servicios**

En cuanto a los servicios que ofrece esta tecnología, el programa de certificación Wi-Fi Direct destaca cuatro servicios principales:

- Envío Wi-Fi Direct: uno o más dispositivos pueden enviar y recibir contenido de forma rápida y sencilla con una mínima interacción por parte del usuario.
- Impresión Wi-Fi Direct: con un solo comando se pueden imprimir documentos directamente desde un smartphone, tableta o PC.
- Wi-Fi Direct para DLNA: los dispositivos que soportan DLNA (asociación de fabricantes de electrónica e informática que marca las directrices para la

compartición de medios digitales entre dispositivos) se descubren antes de conectarse y transmitir el contenido.

- Miracast: permite la implementación de mecanismos de descubrimiento de dispositivos y servicios de Wi-Fi Direct para habilitar el "Screen Mirroring", lo cual ofrece la posibilidad de visualizar el contenido de las pantallas de unos dispositivos en otros.

2.4.- ¿Wi-Fi Direct o Bluetooth?

Inicialmente la mayor parte de los usuarios de dispositivos inteligentes realizaban intercambios de información como por ejemplo fotos a través de la creación de redes por infrarrojos. Después, este sistema fue sustituido por el Bluetooth que actualmente sigue siendo utilizado para conectar dispositivos, pero en menor medida para la transferencia de archivos debido a la aparición de las aplicaciones de mensajería instantánea. Posteriormente surge la tecnología Wi-Fi Direct, la cual se consideraba que podía sustituir a de la tecnología Bluetooth, ya que ofrece velocidades superiores en un rango de distancia también mayor. Pero, realmente la tendencia no apunta a que la tecnología Bluetooth vaya a ser sustituida, principalmente porque tiene un punto a favor muy importante, el consumo de energía.

Wi-Fi Direct supuso con su aparición, un nuevo método de transmisión de archivos entre dispositivos dando lugar a velocidades muy superiores a las de Bluetooth, pasando de velocidades de 24 Mbps a velocidades de hasta 250 Mbps (velocidades 10 veces mayores). A finales de 2016 aparece la versión Bluetooth 5.0 la cual tiene el doble de velocidad y alcanza distancias cuatro veces mayores que su última versión [3]. En materia de velocidades los valores son inferiores a los que ofrece la tecnología Wi-Fi Direct, mientras que en cuanto al rango soportado los valores en ambos casos se situarían en torno a los 200 metros. Por otro lado, como se ha comentado anteriormente, la tecnología Wi-Fi Direct tiene una desventaja en materia de consumo energético con respecto a la tecnología Bluetooth, ya que el consumo continuado del Wi-Fi es superior al ocasionado por la tecnología Bluetooth.

Por tanto el uso de una tecnología u otra dependerá del marco de trabajo. Es decir, si lo que interesa es el envío de datos a gran velocidad o se trata de transmitir archivos de gran tamaño la tecnología Wi-Fi Direct sería la opción adecuada, independientemente de que se trate de distancias grandes o pequeñas. Si además se quieren enviar archivos o vincular dispositivos que se encuentren a distancias relativamente grandes también resulta más eficiente el uso de esta tecnología (ya que la versión Bluetooth 5.0 todavía está en proceso de introducción en los nuevos dispositivos). Por otro lado si simplemente se quiere tener una vinculación con otro dispositivo en el cual no haya un envío masivo de datos, como puede ser por ejemplo el uso de unos auriculares, tiene más sentido el uso de la tecnología Bluetooth debido al menor consumo de energía que supone. La tecnología Bluetooth no parece que vaya camino de desaparecer debido principalmente a la importancia que se le da hoy en día al consumo y duración de las baterías. Además con el auge existente actualmente

con el Internet de las cosas, esta tecnología seguramente esté muy presente en la vinculación de dispositivos debido al menor consumo de batería.

Para lograr el objetivo principal de este proyecto, que es el análisis de la tecnología Wi-Fi Direct en entornos vehiculares, se deberá producir un intercambio de información entre dispositivos en el que los dispositivos se encontraran en movimiento. Por tanto, interesa disponer del mayor tiempo posible para enviar información y hacerlo de la manera más rápida. Es decir, en este caso interesa el uso de la tecnología Wi-Fi Direct, ya que permite la vinculación de dispositivos en mayores rangos de distancia, otorgando más tiempo para la conexión y el envío de información y, además, posibilita que esa información se envíe más rápido debido a su mayor velocidad reduciendo así el tiempo de envío y recepción.

2.5.- **Estado del Arte**

Wi-Fi Direct está incluido actualmente en multitud de dispositivos como por ejemplo las impresoras, cámaras o tarjetas SD, Smart TVs, videoconsolas etc. Como se ha comentado en el apartado *2.3.- Servicios*, un uso muy común es el del servicio Miracast para el envío de screencast (enviar el contenido que se está visualizando en la pantalla de un dispositivo a otros dispositivos), así como el del protocolo DLNA para reproducir por ejemplo, contenidos del móvil en la pantalla de la televisión.

Por otro lado, actualmente también se usan dispositivos como el Google Chromecast [4] para transmitir contenido multimedia como fotos o películas a la televisión. Este dispositivo de tipo pendrive se conecta al HDMI de la televisión, permitiendo que desde cualquier móvil, tablet u ordenador se puede transmitir contenido vía Wi-Fi (no Wi-Fi Direct) al Chromecast y este lo reproduzca en la televisión. Google Chromecast tiene la ventaja de que permite hacer streaming con cualquier televisión a través de Wi-Fi, mientras que con Wi-Fi Direct sólo es posible transmitir ficheros si la televisión soporta DLNA. Sin embargo, una ventaja importante de la tecnología Wi-Fi Direct es que no se necesita conexión a una red Wi-Fi para compartir contenidos al contrario de lo que sucede con el Google Chromecast.

La tecnología Wi-Fi Direct está llegando a todo tipo de dispositivos, por ejemplo Samsung presentó las Monitorless [5] en el Mobile World Congress de 2017 en Barcelona. Esto consiste en unas gafas sobre las cuales se puede proyectar la pantalla del smartphone. Estas gafas constan de una batería y altavoz situados en una de las patillas de las gafas y un proyector, una CPU y un módulo Wi-Fi en la otra patilla. Además también permite la conexión con el ordenador usando el smartphone como intermediario. Para ello se establece una conexión LTE/4G mediante el ordenador y el smartphone para recibir la señal del escritorio y, mediante la conexión Wi-Fi Direct entre el smartphone y las Monitorless, se proyecta la información sobre estas últimas.

En el sector de los smartphones y las tablets Android, la tecnología Wi-Fi Direct se usa principalmente, para la comunicación y transferencia de archivos. El sistema operativo Android incluye Wi-Fi Direct a partir de la versión 4.0 mientras que los dispositivos con el

sistema operativo IOS no han adoptado Wi-Fi Direct, sino que utilizan otro tipo de aplicaciones como AirPlay o Airdrop. Estas aplicaciones usan una tecnología similar a Wi-Fi Direct pero que no cumple completamente con las especificaciones de los estándares. Por ejemplo Airdrop utiliza Wi-Fi para transmitir datos pero Bluetooth para sincronizar los dispositivos. Además cabe destacar que estas aplicaciones solo son compatibles entre dispositivos IOS, no permitiendo la comunicación con dispositivos Android. Sin embargo, si un dispositivo con Android activa Wi-Fi Direct los dispositivos con el sistema operativo IOS si se podrán conectar. Esto se debe a que los dispositivos que se conectan al Wi-Fi Direct de otro no tienen por qué contar con esta tecnología.

Probablemente uno de los principales motivos por los que esta tecnología no está todavía demasiado extendida entre los usuarios de smartphones y tablets, es que la transferencia de archivos no se puede realizar de forma nativa, es decir se necesita de una aplicación de terceros. Por tanto muchos clientes no saben usar Wi-Fi Direct, incluso desconocen esta tecnología, y no quieren instalar una aplicación de terceros para poder utilizarla. Es probable que en futuras versiones de Android se incluya esta aplicación. A continuación se muestran tres aplicaciones [6] actuales que permiten el uso de dicha tecnología para el intercambio de datos. Estas aplicaciones han sido probadas entre un dispositivo Sony Xperia Z3 y una tablet Samsung Galaxy S:

- SHAREit: aplicación utilizada para el intercambio de archivos mediante la tecnología Wi-Fi Direct. Permite enviar o recibir vídeos, imágenes, música etc. Para ello simplemente se debe crear un perfil que consiste en un nombre de usuario y una foto o icono de los que ofrece la aplicación. Posteriormente se selecciona la opción enviar o recibir, y se realiza una búsqueda de receptores o remitentes. Una vez iniciada la búsqueda se elige el usuario con el que se quiere intercambiar información y comienza el proceso de transferencia de archivos.

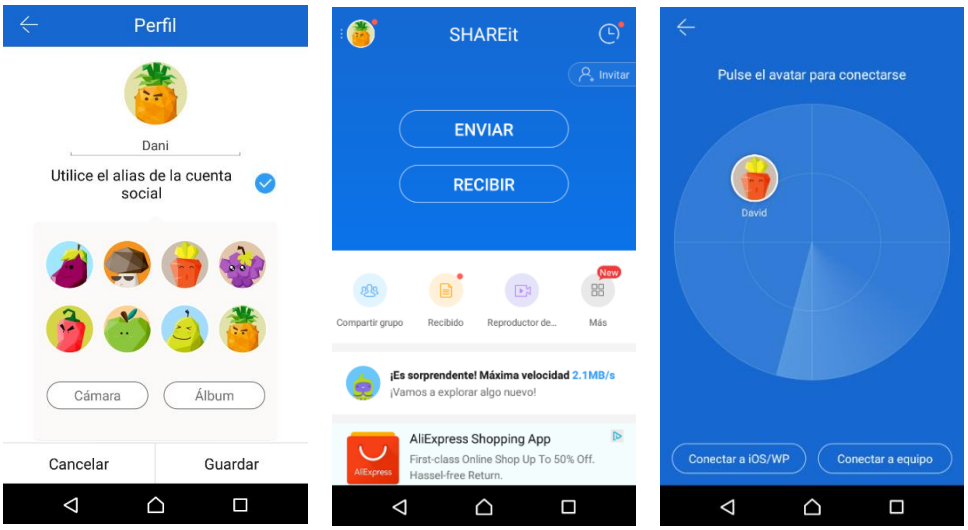

Figura 2.1.- Pruebas de la aplicación SHAREit

- SuperBeam: en esta aplicación para compartir archivos simplemente se deberá seleccionar el archivo que se desea enviar y posteriormente elegir la manera de emparejar ambos dispositivos, mediante la lectura del código QR o clave compartida (clientes de pago). Una vez emparejados comienza la descarga mediante la tecnología Wi-Fi Direct. También ofrece la opción de compartir los archivos mediante el uso de la tecnología NFC.

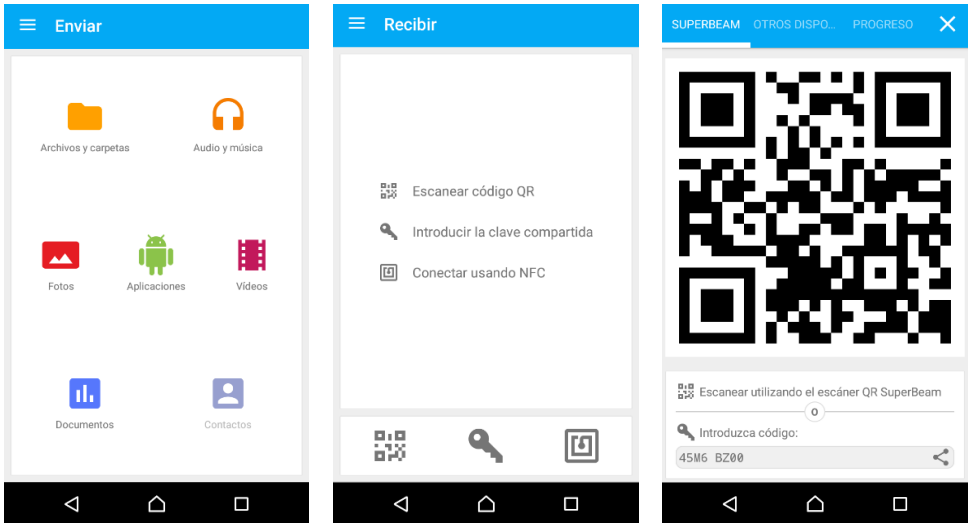

Figura 2.2.- Pruebas de la aplicación SuperBeam

- Send Anywhere: esta aplicación es parecida a SuperBeam pero tiene dos diferencias principales. Por un lado no ofrece la posibilidad de usar la tecnología NFC, pero sin embargo, permite el emparejamiento de los dispositivos mediante clave compartida sin la necesidad de ser cliente de pago. Para llevar a cabo un intercambio de archivos se selecciona los archivos que se desean enviar lo que genera una clave de 6 dígitos que caduca a los 10 minutos y un código QR. El receptor introduce dicha clave o escanea el código QR y comienza el proceso de transferencia.

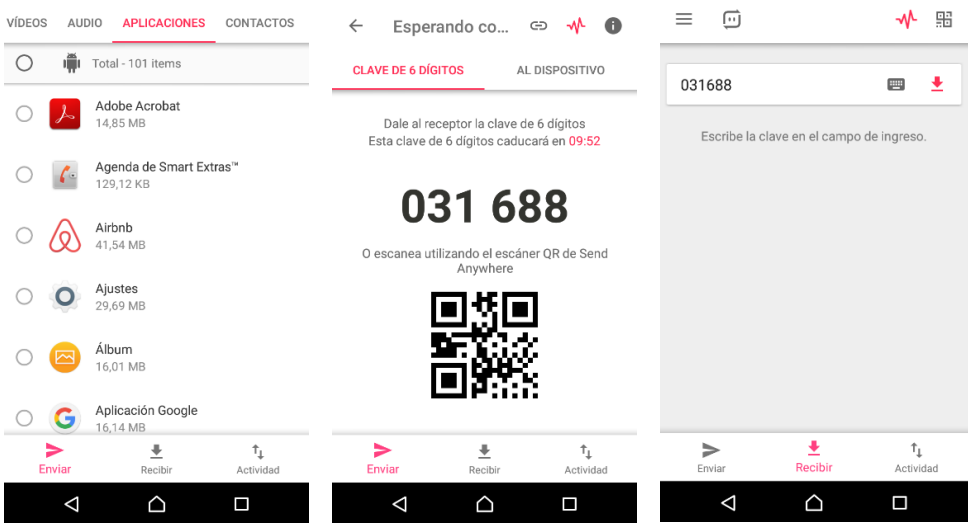

Figura 2.3- Pruebas de la aplicación Send Anywhere

2.5.1.- **Redes Vehiculares**

En relación al intercambio de información en entornos vehiculares, el IEEE introdujo WAVE para facilitar la comunicación entre sistemas de diferentes fabricantes de automóviles. Se trata de la estandarización de un conjunto de protocolos de acceso inalámbrico en entornos vehiculares desarrollado por la IEEE. Tiene como objetivo la comunicación entre vehículos, o vehículos e infraestructuras en un entorno cambiante. Existen muchos tipos de tecnologías inalámbricas que pueden utilizarse para transferir los datos del sistema WAVE, como por ejemplo la tecnología Wi-Fi, el uso de la red móvil, Bluetooth e incluso el uso de satélites.

Después de evaluar el coste, rendimiento y la disponibilidad, IEEE recomienda IEEE 802.11p, que es una extensión de la tecnología Wi-Fi que utiliza la banda 5.9G, la cual se asigna a las "comunicaciones dedicadas de corto alcance" o DSRC. Mediante este estándar, la latencia se reduce significativamente en comparación con la tecnología Wi-Fi debido a la eliminación de la vinculación y la autenticación, además de que el rango de trabajo se extiende hasta los mil metros. De esta manera la unidad cliente del vehículo puede enviar un mensaje directamente después de que escanee y encuentre el punto de acceso, no necesitando ninguna contraseña para usar la red. Corresponderá a la capa superior ocuparse de la cuestión de seguridad.

Para llevar a cabo la comunicación se necesita de dos transceptores, el RSU y el OBU. El RSU es un dispositivo fijo para comunicaciones DSRC que se encuentra generalmente en los laterales de las carreteras o en los sistemas de peaje. Por su parte el OBU es también un dispositivo para este tipo de comunicaciones que se encuentra instalado en los vehículos y, a diferencia del anterior, puede operar cuando está en movimiento. Este sistema hace que los vehículos formen una red en la cual los éstos junto con los RSUs pueden hacer de nodos y receptores. Existen dos roles principales, proveedor y usuario, que no dependen del tipo de dispositivo, pero en general el RSU actuará como el proveedor. Hay diferentes aplicaciones de este estándar como por ejemplo el cobro en peajes, vehículos de emergencia, aviso de frenada, estado del tráfico etc.

Existen diferentes parámetros que sirven para llevar a cabo una comparación entre el estándar WAVE y la tecnología Wi-Fi Direct [7]. Principalmente las ventajas de la tecnología Wi-Fi Direct se basan en que cuenta con una tasa de transferencia mucho mayor que el estándar WAVE (prácticamente 10 veces mayor) así como el hecho de que ésta tecnología está disponible en dispositivos habitualmente utilizados como pueden ser los teléfonos móviles. Por su parte el estándar WAVE tiene un radio de alcance cinco veces superior a la tecnología Wi-Fi Direct, soporta redes Ad-hoc además de que prácticamente no invierte tiempo en el establecimiento de la conexión entre dispositivos.

En definitiva, como se ha comentado anteriormente se necesita de equipos especiales para llevar a cabo la comunicación mediante el estándar WAVE, lo que supone de un coste adicional, motivo por el cual muchos conductores no estarían interesados. Sin embargo, debido al incremento del número de usuarios de smartphones y a que la mayoría de los

smartphones tienen la certificación Wi-Fi Direct, esta tecnología puede ser una solución para el intercambio de información V2V y V2I. Es por ello que en el presente proyecto se estudiará la viabilidad de utilizar esta tecnología en entonos vehiculares, usando como herramienta una aplicación en Android que permita obtener resultados que definen las características de esta tecnología en un entorno vehicular. De esta manera los conductores simplemente deberían instalar una aplicación basada en la tecnología Wi-Fi Direct para poder intercambiar mensajes con otros dispositivos inteligentes.

Por otro lado existen diferentes estudios relacionados con la tecnología Wi-Fi Direct en entornos vehiculares. Algunos de ellos son los siguientes:

- Device-to-Device Communications with WiFi Direct: Overview and Experimentation: en este estudio de la Universidad Carlos III de Madrid [2] se analiza esta tecnología en diferentes escenarios con el fin de obtener el rendimiento real de la misma. De esta forma se estudian parámetros como los retardos producidos en el descubrimiento y conexión de los diferentes dispositivos, así como diferentes mecanismos para el ahorro de energía. En el caso de los retardos, se concluye que existe un mayor retardo, además de una mayor variabilidad de los mismos para los grupos formados de manera estándar y persistente, mientras que los grupos formados de manera autónoma tienen un retardo menor ofreciendo a su vez valores prácticamente constantes del mismo. Por su parte en cuanto al ahorro de energía se analiza principalmente el protocolo Notice of Absence concluyendo que la política dinámica es la óptima teniendo en cuenta el rendimiento y coste energético en diferentes entornos, entre ellos el vehicular.

- Inter-vehicular communication for collision avoidance using Wi-Fi Direct: mediante esta tesis [9] se busca implementar un sistema para prevenir las colisiones mediante el uso de la tecnología Wi-Fi Direct como alternativa a la tecnología DSRC. Para ello se considera importante el intercambio de información, no solo entre vehículos, sino también con los peatones, lo cual reduciría el número de accidentes. De esta forma en el estudio se calcularon los diferentes retardos producidos observándose que se producían incrementos en la congestión de la red cuantos más dispositivos hubiese en un grupo Wi-Fi Direct. Esto se debía principalmente a retransmisiones de los mensajes de seguridad producidos por el GO. Por tanto, se propuso un método para solucionar estos retrasos de tal manera que el GO trasmitía mensajes a todos los clientes del grupo eliminando el tiempo producido por la retransmisión. Con este nuevo modelo se probó que los retardos en la transmisión se reducían y que los grupos P2P podían admitir más vehículos, aunque seguían siendo en cierta medida limitados.

- An Energy Efficient Vehicle to Pedestrian Communication Method for Safety Applications: en este documento al igual que en el anterior se analiza el intercambio de mensajes de seguridad entre vehículos y peatones, descartándose el uso del protocolo WAVE y proponiendo el uso de Wi-Fi Direct debido a su integración en los smartphones[10]. En este caso se busca un método para usar eficientemente la energía empleada en el intercambio de mensajes de seguridad V2P. Con el fin de ahorrar energía en los dispositivos móviles se escoge el P2P GO en función de la energía residual y la distancia. Además el P2P GO establece una serie de intervalos de tiempo que se pueden omitir según la información del tráfico para reducir la energía consumida por los clientes P2P. Finalmente se compara con el sistema WAVE concluyendo que se produce una mejora en el rendimiento de la entrega de mensajes así como de la eficiencia energética del 22% y el 37% con respectivamente.
- A Hybrid V2X System for Safety-Critical Applications in VANET: en este documento se proponen alternativas para el soporte de la comunicación V2X sin el uso del protocolo WAVE/DSRC [8]. En concreto se propone un sistema híbrido entre la red de telefonía móvil y la tecnología Wi-Fi Direct, de tal manera que los vehículos son monitorizados a través de la red móvil mientras que la transmisión local de datos se produce mediante Wi-Fi Direct. Para conseguir una solución óptima se busca reducir los altos periodos de tiempo de establecimiento de la conexión (CET) que se produce al usar la tecnología Wi-Fi Direct. En el estudio se consigue reducir el CET principalmente gracias a la reducción del tiempo de descubrimiento. Esto se debe a que el proceso de descubrimiento se lleva a cabo mediante el servidor que se encarga de la monitorización a través de la red móvil, en lugar de llevarse a cabo mediante la tecnología Wi-Fi Direct. Finalmente se deja abierto la creación de nuevos estudios para que esta propuesta sea puesta en práctica ya que se deben conseguir todavía resolver otros problemas como por ejemplo reducir los tiempos empleados en la formación de los grupos, los cuales todavía se consideran demasiado grandes.

Por otro lado existen otros estudios en los cuales se valoran diferentes tecnologías para su uso en entornos vehiculares las cuales requieren todavía de una mayor madurez, ya que se trata de tecnologías que están evolucionando o que todavía no se encuentran disponibles como por ejemplo Li-Fi [11] o la red 5G [12] [13].

3.- ENTORNO DE DESARROLLO

En este capítulo se explican las diferentes fases del proyecto para el desarrollo de la aplicación. Es por ello que se ha dividido en dos apartados principales:

- Definición y pasos iniciales.
- Android Studio.

3.1.- Definición y Pasos Iniciales

En este aparatado se hará una introducción al sistema operativo Android, explicando además su relación con Java. También se detallarán los procesos de instalación de los diferentes componentes del entorno de desarrollo, se escogerá el IDE y se comentarán las diferentes partes que lo componen.

3.1.1.- Introducción a Android

La aplicación objeto de este proyecto será desarrollada en Android, un sistema operativo basado en Linux y diseñado por Google. Android fue creado por la empresa Android Inc, adquirida posteriormente por Google en 2005. En noviembre de 2007 se presentó la primera versión de Android, junto con el SDK para que los programadores pudieran empezar a crear sus aplicaciones para este sistema. Inicialmente fue pensado para teléfonos móviles (haciéndose actualmente, con más del 85% del mercado), pero hoy en día se puede encontrar en prácticamente cualquier tipo de dispositivo como pueden ser automóviles, relojes, TV, etc. Su logotipo es el que se muestra en la *Figura 3.1*.

Figura 3.1.- Logotipo Android

Android posee múltiples funcionalidades entre las que destacan principalmente las siguientes:

- Código abierto.
- Aplicaciones escritas en el lenguaje de programación Java.
- Soporte Multitarea.
- Soporte Multitáctil.
- Dispone de un entorno de desarrollo propio.
- Se adapta a distintas resoluciones de pantalla.
- Soporte a diferentes tipos de conexiones (Wi-Fi, Bluetooth, LTE…).
- Permite el envío de mensajes (MMS y SMS),
- Soporte de información multimedia y en Streaming.

- Dispone de navegador web.
- Google Play.
- Soporte para hardware adicional.
- Aplicaciones comprimidas en formato APK.

La principal diferencia de Android con respecto al resto de sistemas operativos para dispositivos móviles es que se trata de un software libre, resultando la mayor parte de código abierto. Esto quiere decir que es gratuito, y que cualquiera puede añadirle mejoras, a diferencia de los sistemas propietarios, en los que sólo el fabricante puede realizar modificaciones.

Con respecto a las versiones disponibles en Android, cabe mencionar que se denominan con nombres de postres cuyas iniciales se ordenan alfabéticamente. Así pues, la primera versión de Android se llamó Apple Pie**,** la segunda Banana Bread y así sucesivamente. Esto permite reconocer las versiones y determinar cuáles son las más recientes de acuerdo a su letra inicial. En el momento de llevar a cabo la aplicación habrá que tener en cuenta para qué versiones se desea que ésta sea compatible, teniendo en cuenta el número de usuarios por versión así como las funcionalidades ofrecidas.

3.1.1.1.- Arquitectura

Android tiene la siguiente arquitectura [14]:

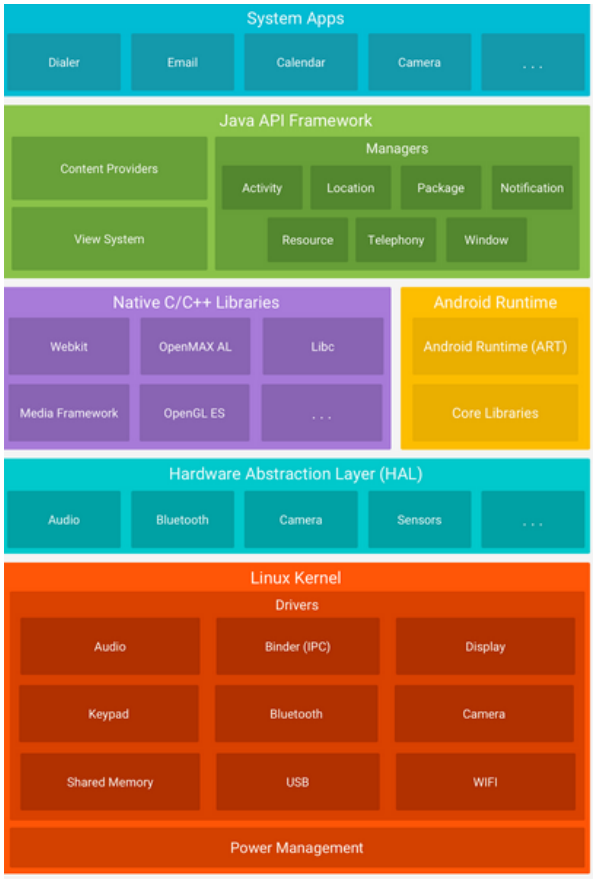

Figura 3.2.- Arquitectura Android

- **Kernel de Linux**: Es la base de la plataforma Android y depende de Linux para los servicios base del sistema. Permite aprovechar funciones de seguridad claves, además de que ofrece la posibilidad a los fabricantes de dispositivos de desarrollar controladores de hardware para un kernel conocido. Por ejemplo, el tiempo de ejecución de Android se basa en el kernel de Linux para la generación de subprocesos y la administración de memoria de bajo nivel.
- **Capa de abstracción de Hardware (HAL)**: consiste en varios módulos de biblioteca, implementando cada uno de éstos una interfaz para un tipo específico de componente de hardware, como el módulo de la cámara o de bluetooth. Cuando el framework de una API realiza una llamada para acceder a hardware del dispositivo, el sistema Android carga el módulo de biblioteca para el componente de hardware en cuestión.
- **Tiempo de ejecución de Android**: para los dispositivos con Android 5.0 o versiones posteriores, cada aplicación ejecuta sus propios procesos con sus propias instancias del tiempo de ejecución de Android. De esta manera se reemplaza a Dalvik que es la máquina virtual usada en versiones anteriores. El ART está escrito para ejecutar varias máquinas virtuales en dispositivos con poca memoria.
- **Bibliotecas C/C++ nativas**: Muchos componentes y servicios centrales del sistema Android, como el ART y la HAL, se basan en código nativo que requiere bibliotecas nativas escritas en C y C++.
- **Framework de la API de Java:** Todo el conjunto de funciones del sistema operativo Android está disponible mediante APIs escritas en el lenguaje Java. Estas APIs son la base para crear aplicaciones de Android simplificando la reutilización de componentes del sistema y servicios centrales.
- **Apps del sistema:** En Android se incluye un conjunto de apps centrales para correo electrónico, mensajería SMS, calendarios, navegación en Internet y contactos, entre otros elementos.

3.1.2.- Relación entre Android y Java

El sistema operativo Android está escrito en el lenguaje de programación Java, por lo tanto, será necesario instalar un software para ejecutar el código Java en el equipo en el que se va a desarrollar la aplicación. Con este propósito se deberá descargar el Java Development Kit, que consiste en un paquete con herramientas, utilidades, documentación y ejemplos para desarrollar aplicaciones Java.

Con respecto al sistema operativo a utilizar, Java se puede utilizar sobre cualquier sistema operativo (Windows, Mac OS, Linux). Es decir, Java permite escribir y compilar el programa una sola vez permitiendo ejecutar ese código en cualquier plataforma. Esto es así debido a que Java no se ejecuta sobre el sistema operativo del equipo, sino sobre su propia

máquina virtual, la cual cuenta con su propio hardware y sistema operativo. En definitiva en el lenguaje Java, se cambia mediante la compilación del código fuente a un código intermedio denominado "bytecode" entendible por la máquina virtual de Java, la cual es la encargada de interpretar el "bytecode" dando lugar a la ejecución del programa [15].

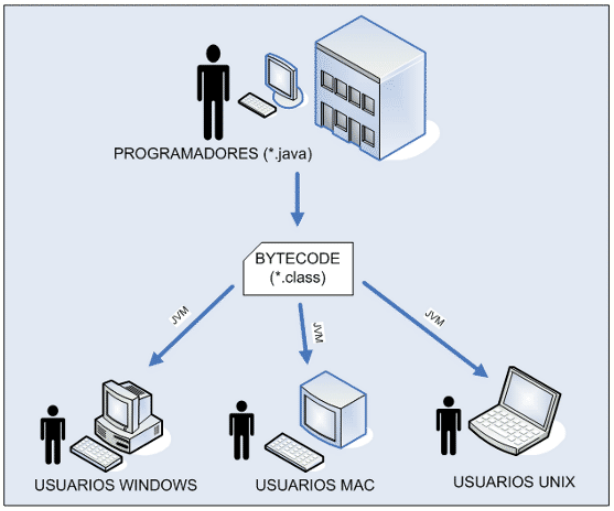

Figura 3.3.- Esquema compilado/ interpretado Java

3.1.2.1.- Instalación del Java Development Kit (JDK)

Como se ha mencionado anteriormente el JDK es el software que permite al equipo leer y trabajar en el lenguaje Java. Antes de proceder a la instalación de JDK se puede comprobar si éste ya se encuentra instalado en el equipo. Para ello simplemente se debe entrar en la línea de comandos e introducir el comando "*java –versión*". La versión se especifica en la primera línea y corresponde al número situado después del "1.", en el caso de la *Figura 3.4*, la versión es la "8.0_121".

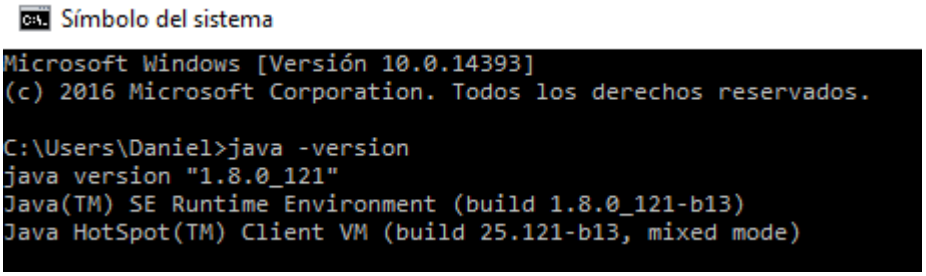

Figura 3.4.- Comando java -version

En cuanto a la versión Java, Oracle recomienda estar actualizado ya que cada actualización suele incluir importantes mejoras en la seguridad. Por tanto, si se tiene Java instalada en el equipo, y se quiere actualizar (en el caso de no tener marcada la opción de actualizar automáticamente), simplemente se debe introducir "java" en el buscador de la barra de tareas, y seleccionar la opción "Configurar Java". Posteriormente se selecciona la pestaña "Actualizar" y se escoge la opción "Actualizar ahora".

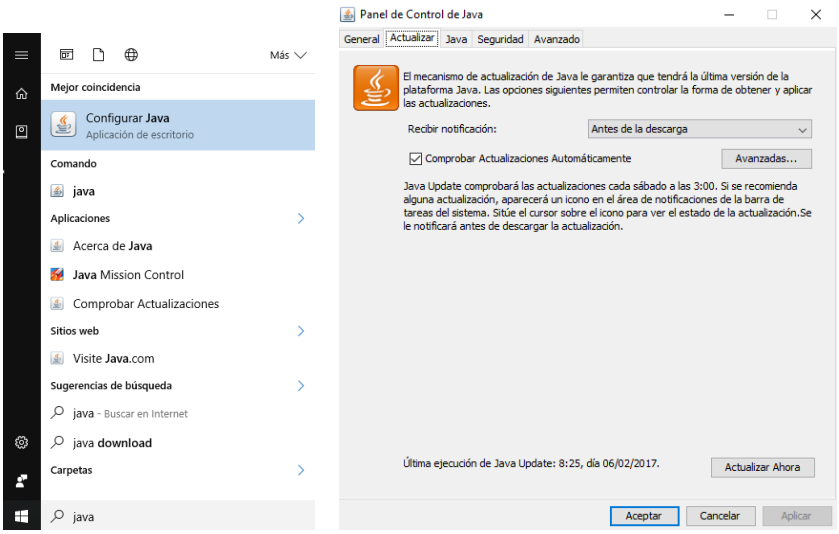

Figura 3.5.- Actualizar Java

Por otro lado, en el caso de no tener Java instalado en el equipo, se accede a la página de Oracle [16], y se selecciona la opción “Java para desarrolladores” presente en la pestaña “Descargas”.

Figura 3.6.- Descarga Java

Posteriormente aparecerá automáticamente la página de descarga de la versión Java SE. Esta versión es una distribución gratuita de la principal plataforma de programación en Java. Contiene un kit de desarrollo de software que se utiliza para escribir aplicaciones con el lenguaje de programación Java. Se trata de la versión más utilizada y la base de otras distribuciones Java. Una vez se accede a la página de descargas de Java SE, se debe escoger el paquete que se desea instalar, en este caso se instala el JDK. Este paquete está diseñado para desarrolladores de Java e incluye un JRE completo (el cual contiene la Máquina virtual) más herramientas para desarrollar, depurar y monitorizar aplicaciones Java.

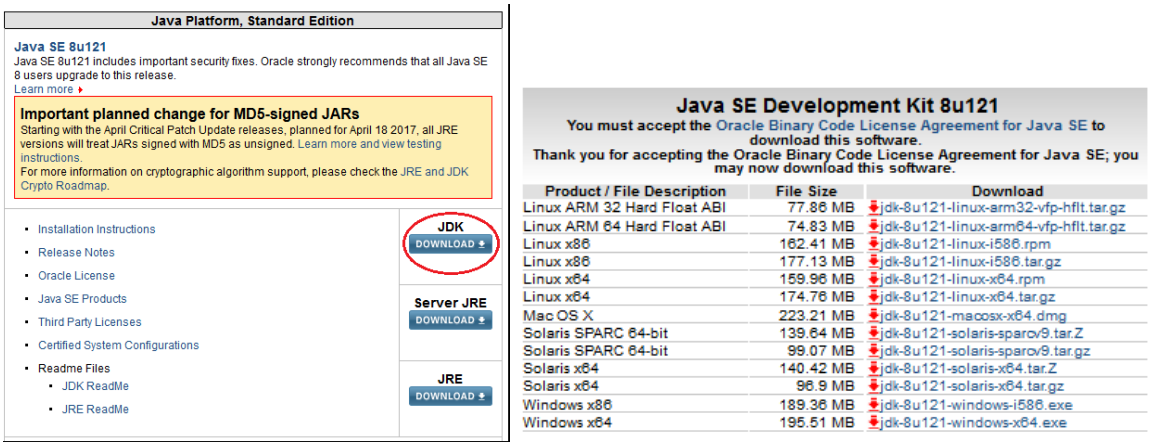

Figura 3.7.- Descarga del Java SE Development Kit

Por último se selecciona la opción adecuada al sistema operativo del equipo en el que se va a desarrollar la aplicación (en este caso, Windows x64), y se procede su descarga e instalación. Primero se abre el instalador de Java y se seleccionan las partes que se desean instalar (se mantienen las que aparecen por defecto) así como su ubicación.

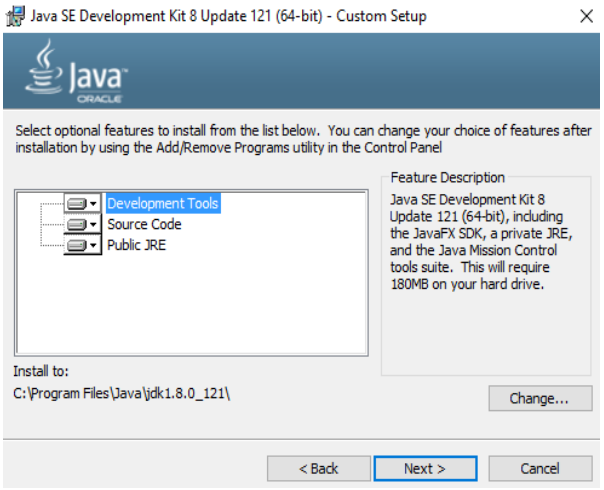

Figura 3.8.- Configuración de la Instalación del JDK

Posteriormente se permite cambiar la ubicación de la carpeta JRE, pero se deja la misma que la del JDK y se lleva a cabo la instalación. Una vez finalizado el paso anterior se verifica que la instalación se haya producido correctamente. Para ello se accede a la ruta antes especificada y se comprueba que existe una carpeta "jdk" y otra "jre" con el número de la versión Java descargada.

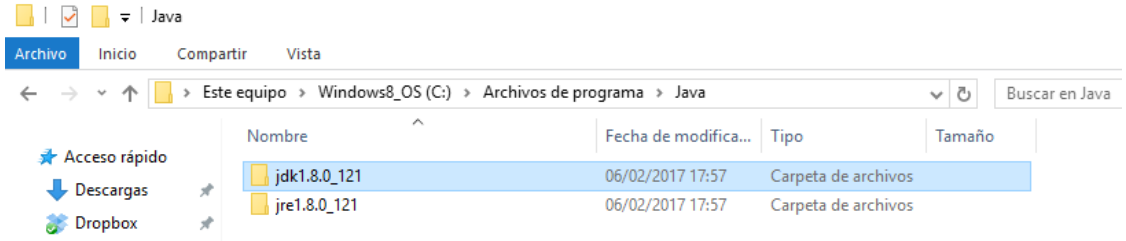

Figura 3.9.- Verificación de la instalación de Java

Con el fin de evitar posibles problemas en el futuro es recomendable realizar una pequeña configuración en las variables de entorno del equipo. Para ello botón derecho sobre "Equipo", se selecciona *Propiedades/ Configuración avanzada del sistema/ Opciones Avanzadas/ Variables de entorno*.

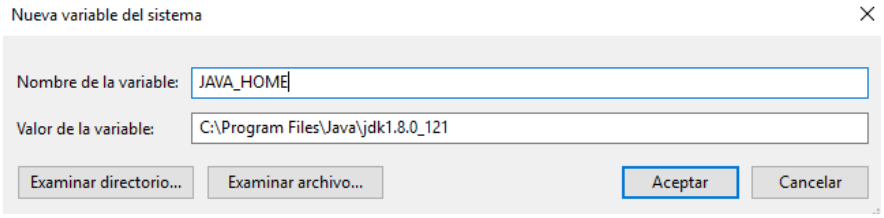

Figura 3.10.- Creación de la Variable JAVA_HOME

En primer lugar se crea la variable de entorno JAVA_HOME en el apartado "Variables del sistema", la cual informará al sistema operativo de la ruta en la que se encuentra instalado el JDK. Por tanto se copia esta ruta (en este caso "C:\Program Files\Java\jdk1.8.0_121").

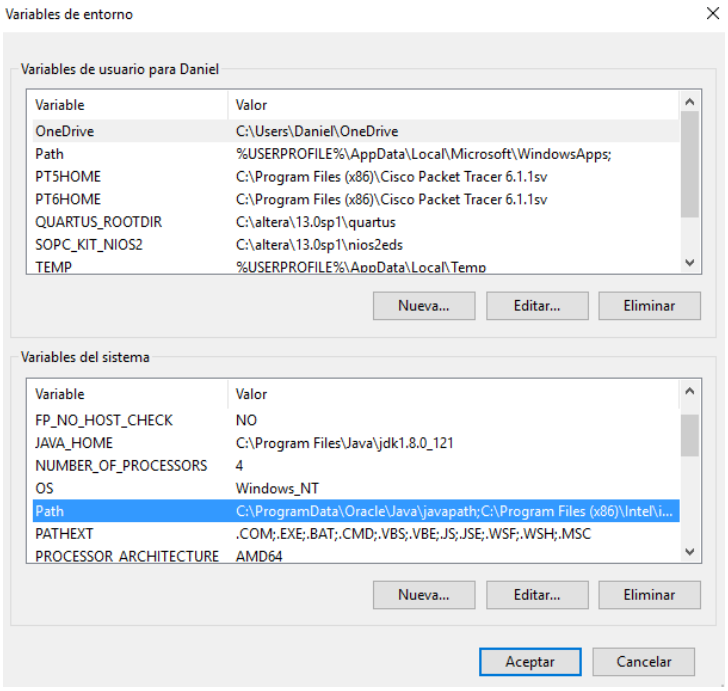

Figura 3.11.- Variables de entorno (Variable Path)

Por último se va a modificar la variable del sistema "PATH". Dicha variable informa al sistema operativo sobre la ruta de diferentes directorios importantes para el funcionamiento del ordenador. A esta variable le añadiremos la ruta donde se encuentran los ficheros ejecutables de Java necesarios para su ejecución, como el compilador (javac.exe) y el intérprete (java.exe). Para ello se busca la variable "Path" en "Variables del sistema" y se edita añadiendo al final del apartado "Valor de Variable" un punto y coma con el texto "%JAVA_HOME%\bin".

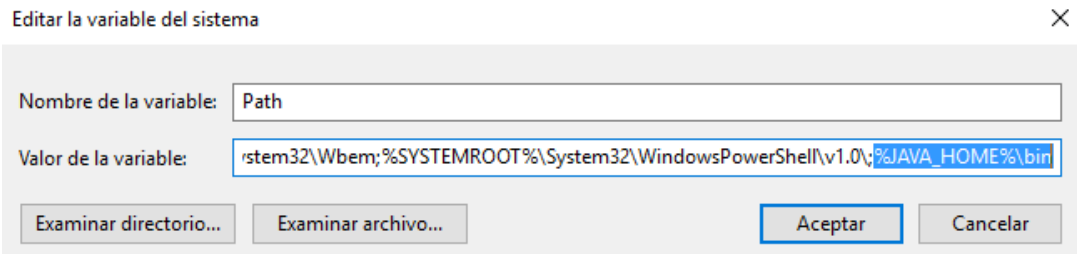

Figura 3.12.- Edición de la variable Path

A modo de comprobación se puede ejecutar un programa sencillo desde la línea de comandos. Para ello se accede a la ubicación del programa escrito en Java y se compila mediante el comando "*javac NombrePrograma.java*", siendo javac el nombre del compilador (javac.exe). Este comando transformará el código escrito en lenguaje Java a código entendible por la máquina virtual (bytecode) y por tanto genera un nuevo archivo NombrePrograma.class en esa misma ubicación. Para comprobar que se ha creado ese archivo en Windows se puede introducir el comando "*dir*". Por último, mediante el comando "*java NombrePrograma*", se pide al intérprete que transforme el código de la máquina virtual a código máquina (entendible por el ordenador), y lo ejecute.

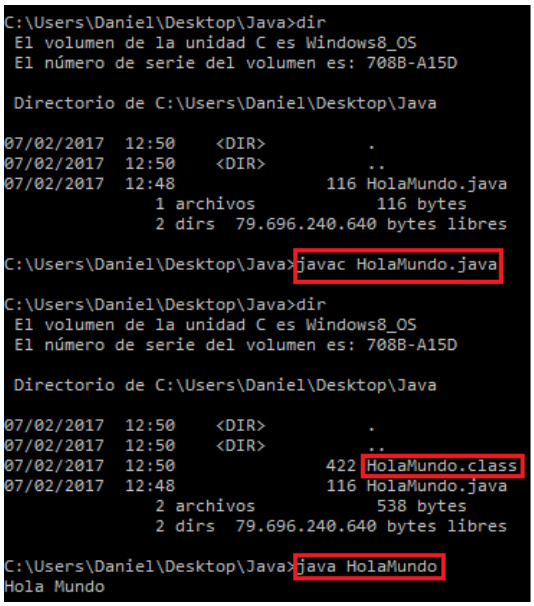

Figura 3.13.- Comprobación Java

3.1.3.- Elección del IDE

Una vez que se tenga instalado Java en el equipo se deberá hacer un estudio acerca de que IDE se empleará para el desarrollo de la aplicación. Principalmente existen dos opciones:

- Eclipse (añadiéndole el plugin de herramientas para desarrolladores de Android, conocido como ADT).
- Android Studio.

Después de la instalación de ambos IDEs y realizar pruebas con ellos y tras una búsqueda de información se concluye que se debe utilizar Android Studio (detallado en el apartado *3.2.- Android Studio*). Esto se debe a que cuenta con numerosas ventajas sobre Eclipse como por ejemplo que se trata de un IDE que se dedica exclusivamente a la programación en Android por lo que tiene un entorno más unificado así como mucho más veloz, y principalmente, por el hecho de que Google ha dejado de dar soporte oficial al plugin de ADT para Eclipse. Por tanto, Android Studio es el único entorno de desarrollo para aplicaciones Android con soporte oficial de Google.

3.2.- Android Studio

Android Studio es el IDE oficial de Android. Está diseñado para poder desarrollar aplicaciones para todos los dispositivos Android de manera rápida y sencilla. Ofrece herramientas personalizadas para programadores de Android e incluye herramientas de edición, depuración, pruebas y perfilamiento de códigos.

Este IDE dispone de herramientas bastantes útiles como por ejemplo Instant Run o el editor de código inteligente. Esta opción permite que al hacer clic en Run o Debug, se apliquen cambios en el código y en los recursos de la aplicación en ejecución de manera que, no hace falta reiniciar la aplicación ni volver a compilar el APK. Esto permite obtener los resultados de inmediato con el ahorro de tiempo que esto conlleva. El editor de código inteligente por su parte ofrece una serie de características que permiten escribir código de manera más eficaz. A medida que se escribe, Android Studio proporciona sugerencias en una lista desplegable.

Otra característica interesante es que cuenta con un emulador que ofrece bastantes funcionalidades. Éste instala e inicia las aplicaciones más rápido que un dispositivo real y permite crear prototipos de la aplicación y probarlos en todas las configuraciones de dispositivos Android: teléfonos, tablets y dispositivos Android Wear y Android TV. También se pueden simular varias funciones de hardware, como la localización de GPS, la latencia de red y las funciones multitáctiles.

3.2.1.- Instalación de Android Studio y Configuración Inicial

Una vez se ha elegido el IDE sobre el cual se va a trabajar, y se ha instalado el JDK de Java se procede a llevar a cabo la instalación de Android Studio. Para ello se debe acceder a la página principal de Android Developers [14], y descargar el entorno de desarrollo. En esta página siempre aparece la última versión disponible del IDE, así como las guías de uso y de instalación del mismo.

Figura 3.14.- Descarga Android Studio

Posteriormente se procede a llevar a cabo el proceso de instalación mediante el cual se navegará a través de diferentes ventanas en las que se escogerán las opciones que aparecen como predeterminadas.

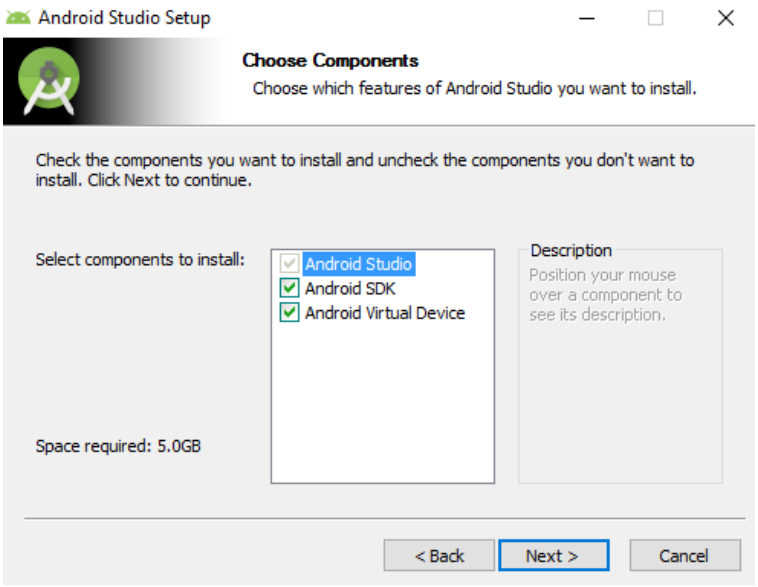

Figura 3.15.- Instalación Android Studio

Una vez finalizado el proceso de instalación aparecerá la pantalla de la *Figura 3.16*. En ella se muestran diferentes opciones, cómo empezar un nuevo proyecto, abrir uno ya existente o importar otros proyectos. Además también proporciona la opción de llevar acabo algunas configuraciones sobre el IDE, SDK Manager etc. Lo primero que se lleva a cabo es una familiarización con la plataforma de desarrollo Android Studio. Antes de crear un proyecto hay que comprobar que se disponga de las herramientas necesarias para llevarlo a cabo. Para ello se va a configurar el SDK Manager, el cual contiene todas las APIs disponibles. Por tanto, se hace clic sobre la opción “Configure” y se selecciona “SDK Manager”.

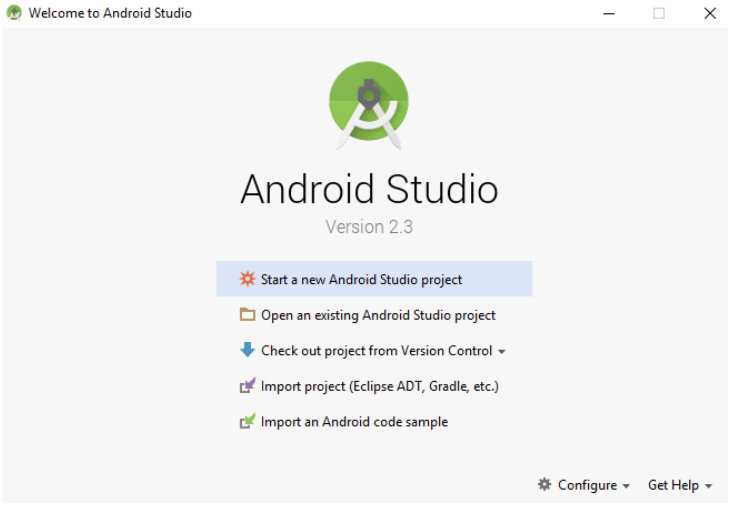

Figura 3.16.- Pantalla inicial Android Studio

Lo primero que se muestra en la pestaña SDK Platforms es una descripción de las APIs, es decir de las versiones de Android que se tienen ya instaladas. Conviene tener instaladas al menos 2 versiones, la correspondiente a la última versión disponible de Android, y la correspondiente a la mínima versión de Android que se quiere que soporte la aplicación. De esta manera se podrán probar las aplicaciones sobre ambas versiones para asegurar su correcto funcionamiento. Para el caso de la mínima versión, como se desea que esta aplicación pueda ejecutarse en la mayor parte de dispositivos Android, se instala la API 16 (Android 4.1 Jelly Bean). Esto se debe a que casi la totalidad de los dispositivos Android actuales tienen una versión mayor o igual a esta y que el dispositivo con menor API de los que se dispone para las pruebas se corresponde con esta versión (siempre teniendo en cuenta que soporte la tecnología Wi-Fi Direct, objeto de este proyecto, la cual está disponible desde la API 14).

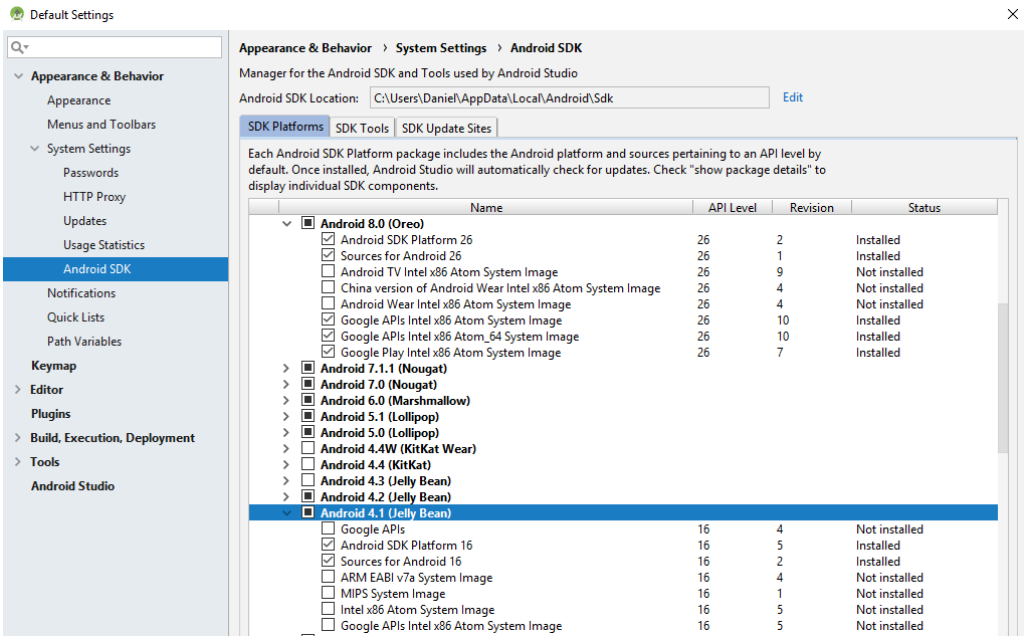

Figura 3.17.- Android SDK

Dentro de cada versión de la API no es necesario instalar todo, simplemente con instalar el SDK Platform (contiene todas las librerías de Android para poder desarrollar las aplicaciones), un emulador de dispositivo Android (si no se quiere utilizar el teléfono móvil para probar las aplicaciones) y Google APIs (si se desea utilizar alguna de las librerías de Google, como por ejemplo los mapas), sería suficiente. También se comprobará que se encuentren instalados el Android SDK Tools así como el Android SDK Build-Tools. El primero lleva a cabo el proceso de depuración (Debugging) permitiendo la localización y corrección de errores. El segundo por su parte se encarga de construir la aplicación a partir del código generado.

3.2.2.- Creación de un Proyecto

Para crear un proyecto se selecciona la opción "Start a new Android Studio Project" presente en la pantalla inicial de Android Studio (*Figura 3.16*). Esto llevará al usuario otra ventana en la cual se pide que se especifique el nombre de la aplicación así como el nombre de la compañía. El nombre de la compañía es una manera para identificar por quien son creadas las aplicaciones. Con esto se genera un nombre de paquete que será único para cada aplicación. Además en esa misma ventana se especifica la ubicación de la carpeta que contendrá los archivos generados por el proyecto.

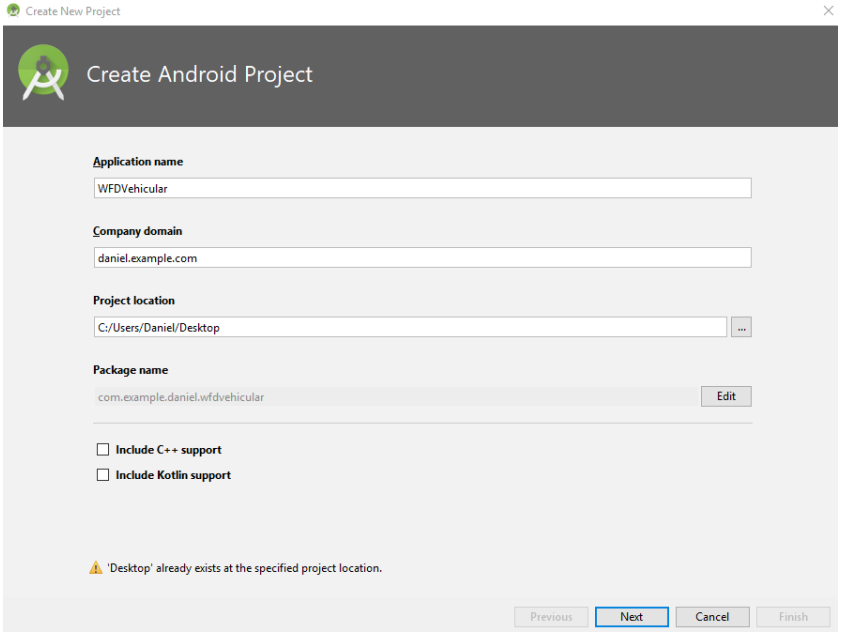

Figura 3.18.- Nombre del proyecto y localización

El siguiente paso consiste en indicar los dispositivos sobre los que funcionará, en este caso con seleccionar Teléfono y Tablet será suficiente. Cabe resaltar en este punto que se debe indicar el SDK mínimo. Como se dijo anteriormente se escoge la API 16 como la mínima (Android 4.1 Jelly Bean) Además debajo de esta opción se especifica que el hecho de escoger una API más baja permite que la aplicación sea usada en más dispositivos pero que a su vez ofrece menos prestaciones. También aparece especificado el tanto por ciento de dispositivos Android que soportarían dicha aplicación en función de la API escogida, en este caso el 99,2 % de los dispositivos.

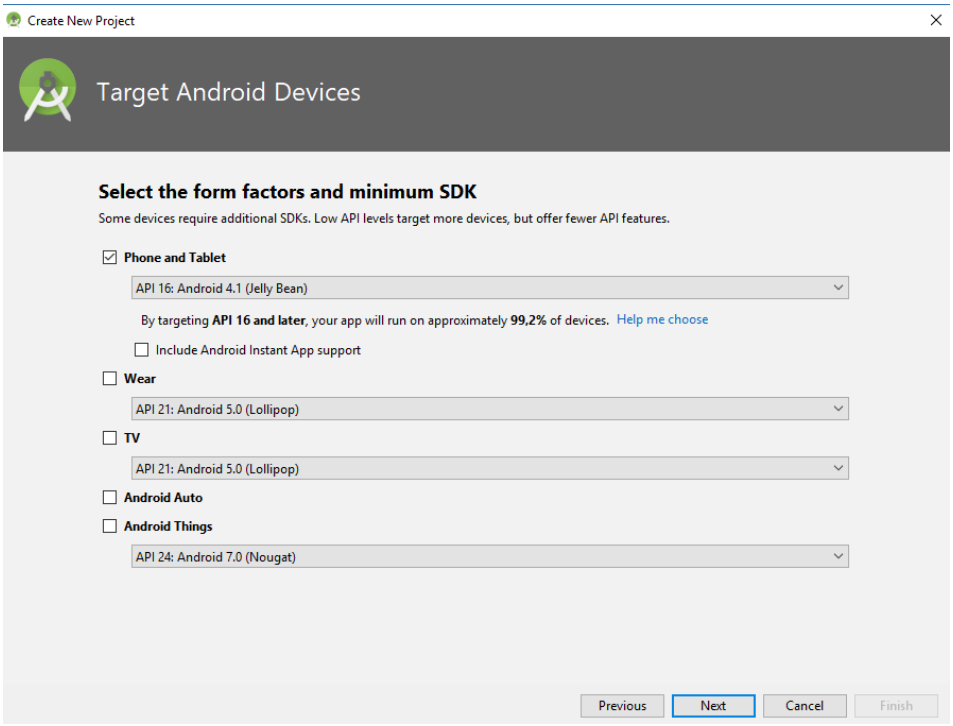

Figura 3.19.- Especificación del dispositivo y SDK mínimo

De hecho si se selecciona la opción "Help me choose" (*Figura 3.19*) se muestra un gráfico con los diferentes porcentajes para cada API así como descripción de las características principales soportadas (*Figura 3.20*).

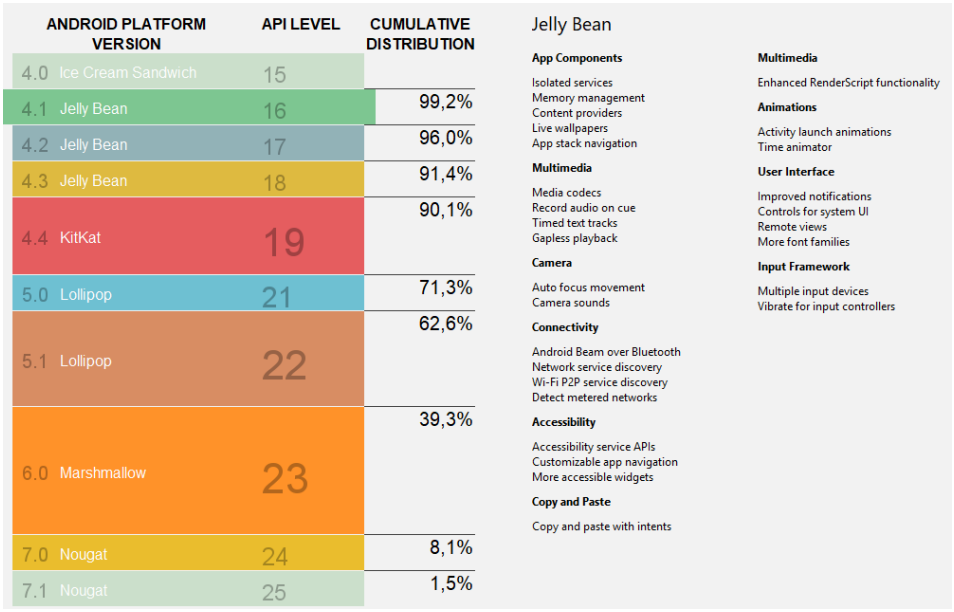

Figura 3.20.- Elección de la API mínima

A continuación se escoge el tipo de pantalla que se desea generar. En algunas pantallas ya se incorporan varios elementos por defecto, sin embargo se puede escoger la opción "Blank Activity" para comenzar la configuración desde cero. Por último se debe asignar un nombre a la actividad y al layout para crear el proyecto.

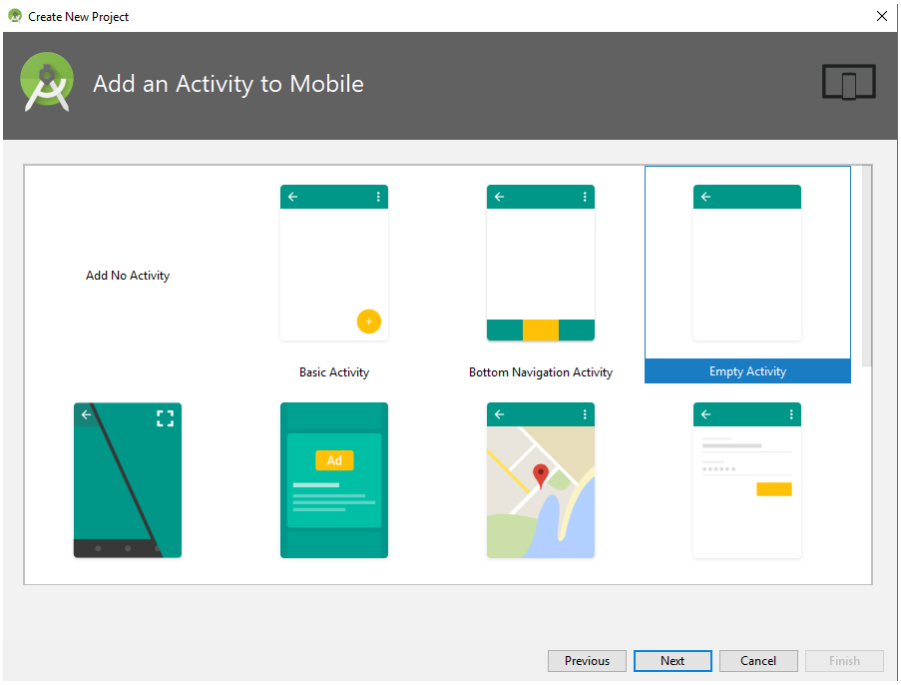

Figura 3.21.- Tipo de pantalla inicial

3.2.3.- Interfaz de Usuario en Android Studio

Una vez creado el proyecto aparecerá la ventana principal del interfaz de usuarios de Android Studio. En la parte central de esta interfaz aparecerán dos pestañas abiertos. Uno que contiene el código Java de la aplicación (nombre.java) y otro que contiene todo el código XML en el cual se definirá la interfaz de usuario para la aplicación que se está desarrollando. La ventana principal de Android Studio consta de diferentes áreas como aparece reflejado en la *Figura 3.22*:

1) Barra de herramientas: situada en la parte superior, permite llevar a cabo diferentes acciones, como ejecutar la aplicación, abrir el SDK Manager etc.

2) Barra de navegación: ayuda a explorar el proyecto y a abrir archivos para su edición. Ofrece una vista más compacta de la estructura del proyecto en la ventana Project.

3) La ventana del editor es el área en la que se puede crear y modificar código. Dependiendo del tipo de archivo, el editor puede ser diferente, por ejemplo, al visualizar un archivo de diseño, se muestra el editor de diseño.

4) Las ventanas de herramientas permiten acceder a tareas específicas, como la administración de proyectos, la búsqueda y los controles de versión, acceso al logcat, etc.

5) En la barra de estado se muestra el estado del proyecto y el IDE, además de diferentes tipos de advertencias o mensajes.

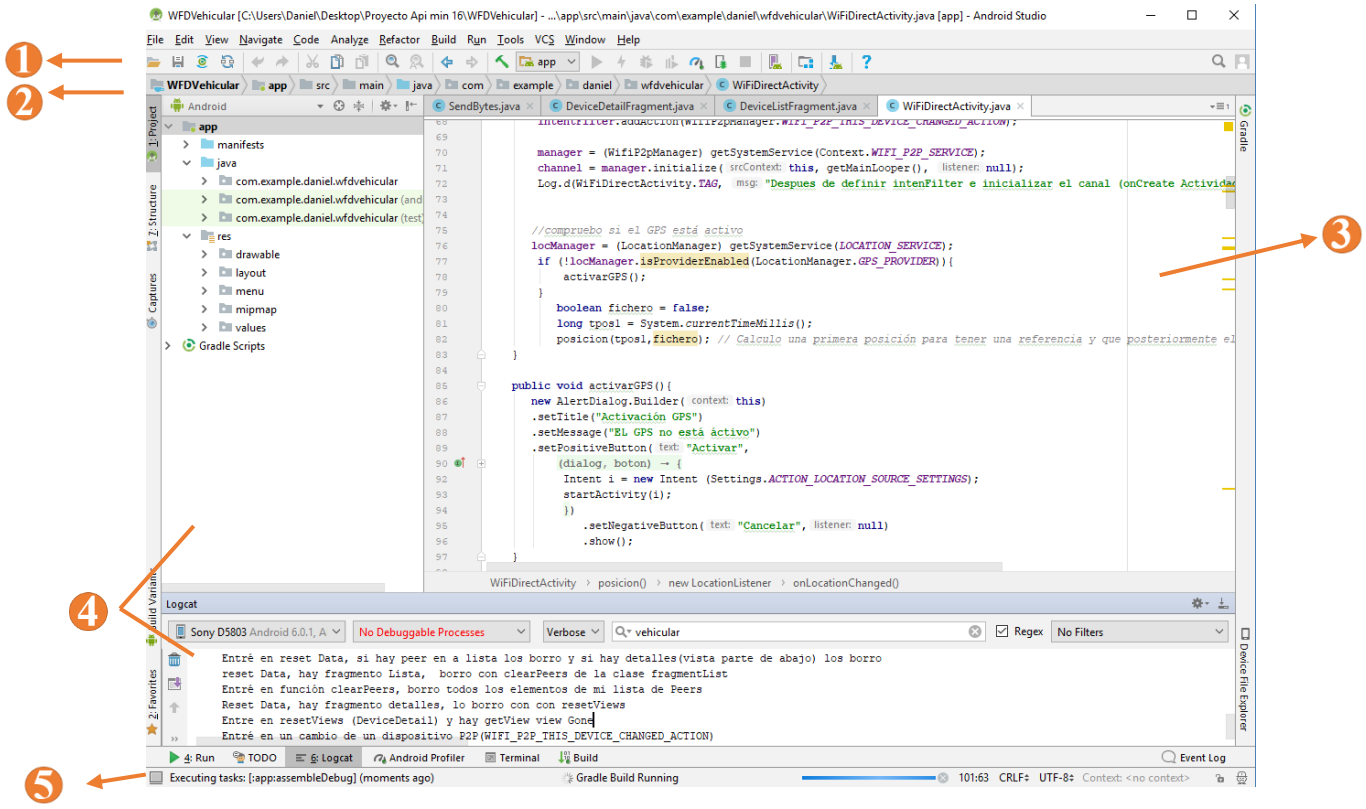

Figura 3.22.- Interfaz gráfica de Android Studio

En la parte izquierda de la *Figura 3.22* se muestra la estructura que sigue un proyecto en Android. Éste se divide en las siguientes partes principales:

- Carpeta manifest: contiene el archivo AndroidManifest.xml. Éste es el archivo de configuración de la aplicación, en él se encuentran definidos aspectos como el paquete de la aplicación, su icono y su nombre, las actividades, así como los permisos necesarios para su ejecución.
- Carpeta Java: contiene el código fuente de la aplicación.
- Carpeta res: contiene los ficheros de recursos necesarios para el desarrollo proyecto como imágenes, ficheros XML (layouts, colores, menús…), cadenas de texto etc.
- Gradle: es la herramienta que se encarga de construir el proyecto. Para ello utiliza todo lo que se tiene en el proyecto (librerías, recursos etc.) y lo une en una aplicación lista para ser ejecutada. Gradle cuenta con un sistema interno a la hora de ejecutar la aplicación que verifica si hubo un cambio en el código, de tal manera que si hubo un cambio recompila todo y genera la aplicación, pero si no hubo ningún cambio se evita estos pasos haciendo más eficiente y rápido ejecutar las aplicaciones móviles. Un archivo importante dentro de Gradle es "build.gradle (Module app)" ya que contiene configuraciones importantes como el SDK empleado para la compilación de la aplicación móvil, la versión mínima y máxima de SDK, etc.

3.2.4.- Ejecución de una Aplicación

En cuanto a la ejecución del proyecto, se puede realizar de diferentes maneras. Por un lado se puede crear un emulador mediante el AVD Manager mientras que por otro lado se puede ejecutar la aplicación en un dispositivo Android como puede ser el teléfono móvil o una tablet. A continuación se evaluarán ambos métodos.

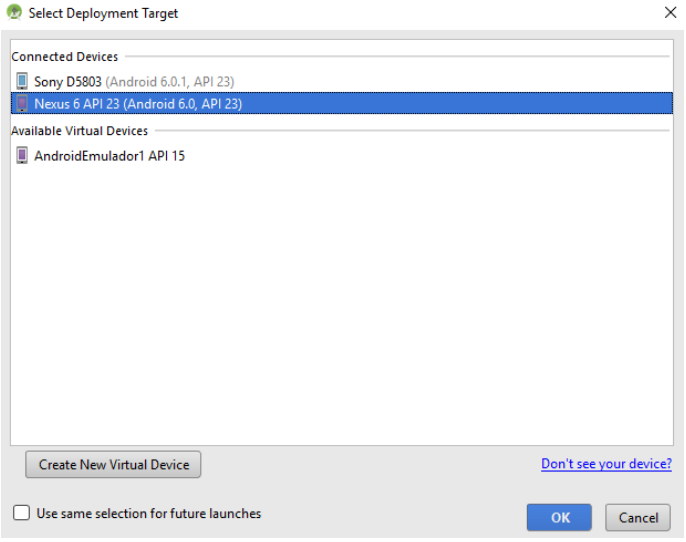

Figura 3.23.- Dispositivos disponibles para la ejecución de la aplicación

En primer lugar una vez se ha creado el programa que se desea ejecutar se debe seleccionar la opción "Run 'App' ". Este paso se puede llevar acabo bien mediante el icono en forma "play" situado en la parte superior de la pantalla o seleccionando dicha opción en la pestaña "Run". Una vez realizado el paso anterior aparecerá una nueva ventana en la cual se mostrarán los dispositivos que se encuentran conectados (AVD o teléfono físico) así como los emuladores que no se están utilizando en este momento como se muestra en la *Figura 3.23*. En esta ventana se escogerá el dispositivo sobre el cual se quiere llevar a cabo la ejecución de la aplicación. Una vez escogido, Android Studio generará un APK de depuración y lo implementará en este dispositivo.

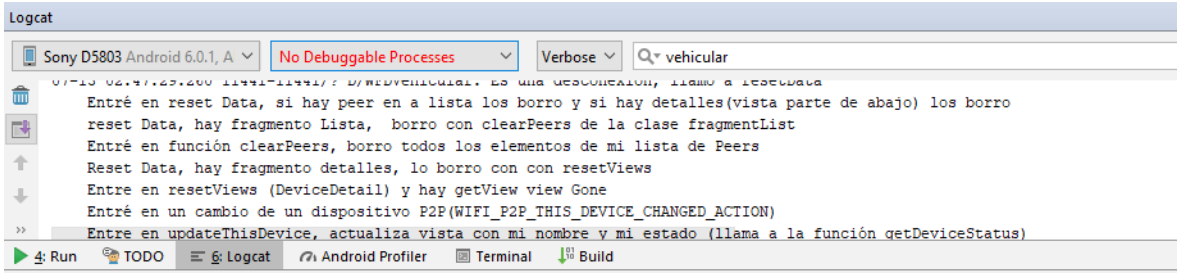

Figura 3.24.- Logcat

Una pestaña que resulta bastante importante a la hora de desarrollar y probar aplicaciones es la pestaña "Logcat". Se encuentra situada en la parte inferior de la pantalla principal. Esta herramienta de línea de comandos vuelca un registro de los mensajes del sistema en relación a los procesos que se están ejecutando en el emulador o dispositivo móvil. Estos mensajes incluyen los seguimientos de pila (llamadas a funciones), casos de

error del sistema y mensajes escritos en la aplicación con la clase "Log" que aporten información relevante a nivel de programación. Mediante estos registros, se indican que errores están ocurriendo y en muchas ocasiones proporciona consejos sobre cómo solucionarlo, por ejemplo revisar cierta línea de código o cierto archivo.

3.2.4.1.- AVD Manager

A través de este método se crea un teléfono, tableta u otro dispositivo Android virtual. Para ello se debe seleccionar la opción AVD Manager situada en la parte superior de la ventana principal del proyecto.

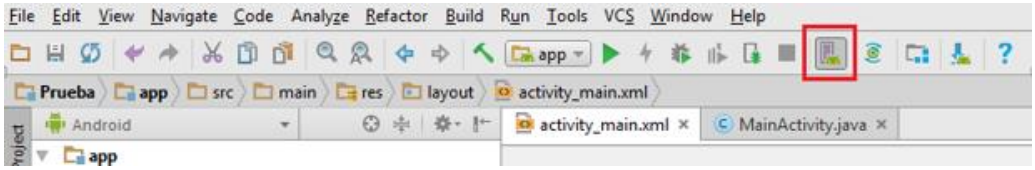

Figura 3.25.- Icono AVD Manager

Una vez hecho clic sobre el icono del AVD Manager, se selecciona la opción de crear un dispositivo virtual. A continuación se pide la especificación del tipo de dispositivo Android que se desea emular. Es decir el hardware que se desea escoger, bien un Android TV, un teléfono móvil, un Android Wear o una tablet. Una vez elegido, se ofrece la posibilidad de escoger uno de los dispositivos creados por defecto o bien generar uno propio. Si se desease generar uno propio simplemente se debe seleccionar la opción "New Hardware Profile".

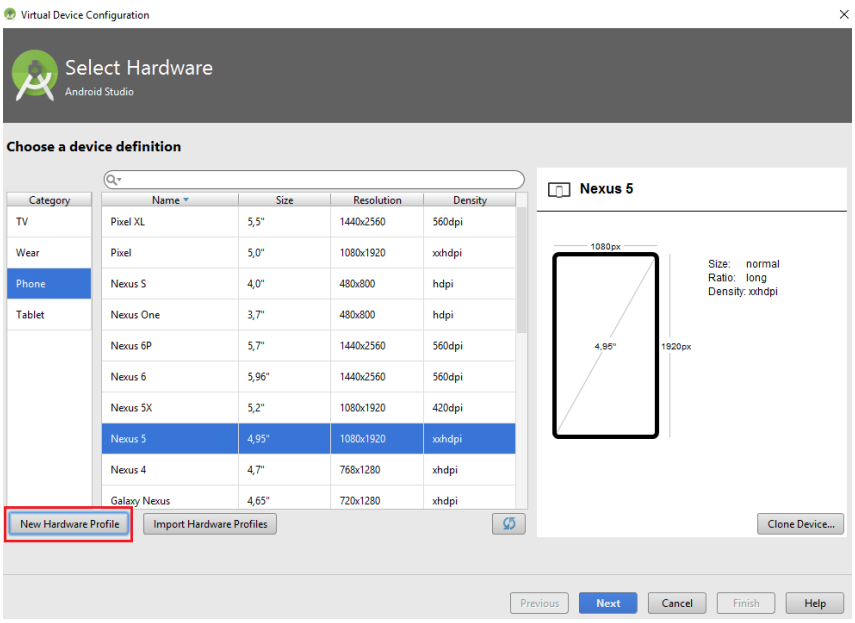

Figura 3.26.- Selección del Hardware del AVD Manager

A continuación aparece otra ventana en la cual se pueden escoger las especificaciones del dispositivo como el nombre, el tipo de dispositivo, las pulgadas de la pantalla, resolución, la cantidad de memoria RAM (habrá que tener en cuenta los recursos de los que se dispongan ya que esta cantidad de memoria será la que se use del dispositivo en el que se está emulando), si se desea que tenga botones y teclado etc.

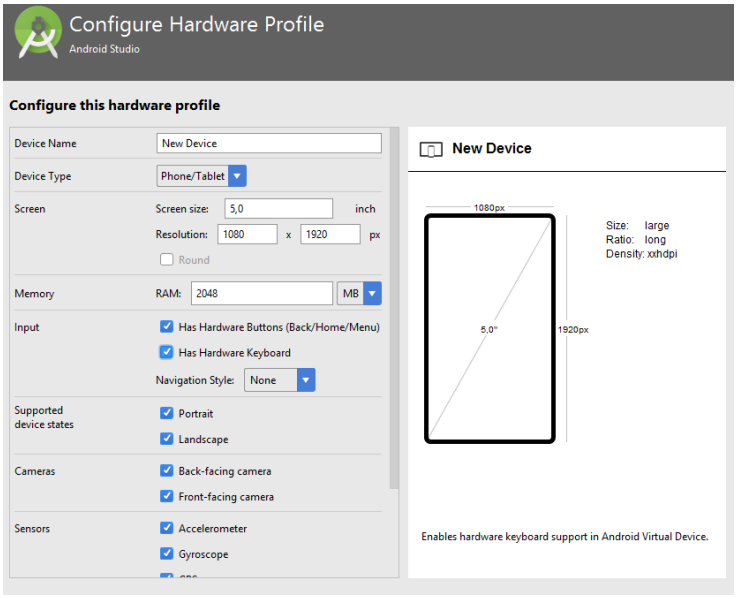

Figura 3.27.- Selección del Hardware del AVD Manager

Una vez escogido nuestro hardware, bien mediante la creación de un dispositivo o bien mediante el uso de uno de los dispositivos por defecto, el siguiente paso será el de escoger la versión de Android sobre la que se va a emular.

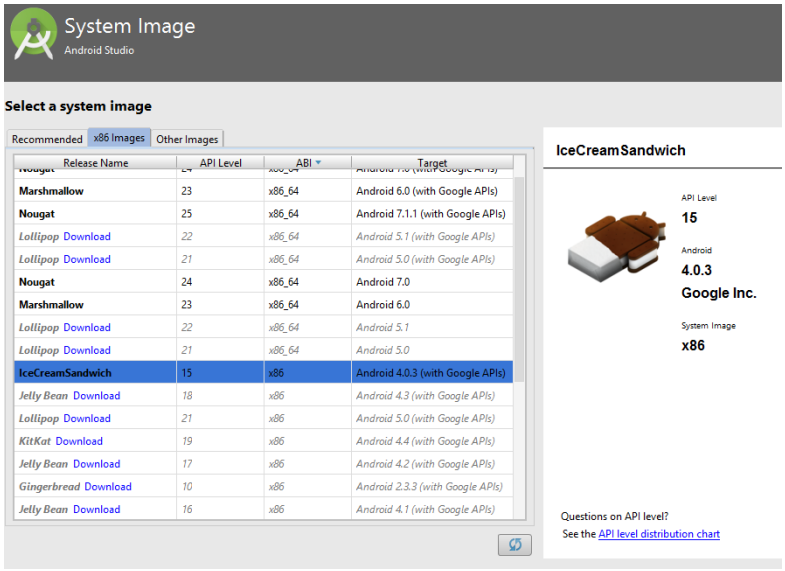

Figura 3.28.- Selección de la versión de Android del AVD Manager

En la *Figura 3.28* se muestran las versiones de los emuladores de Android descargados mediante el SDK, ofreciendo también la posibilidad de Descargar las otras versiones. Además se debe destacar el hecho de que algunas versiones incluyen las APIs de Google, que se deberán seleccionar en el caso de que vayan a ser utilizadas.

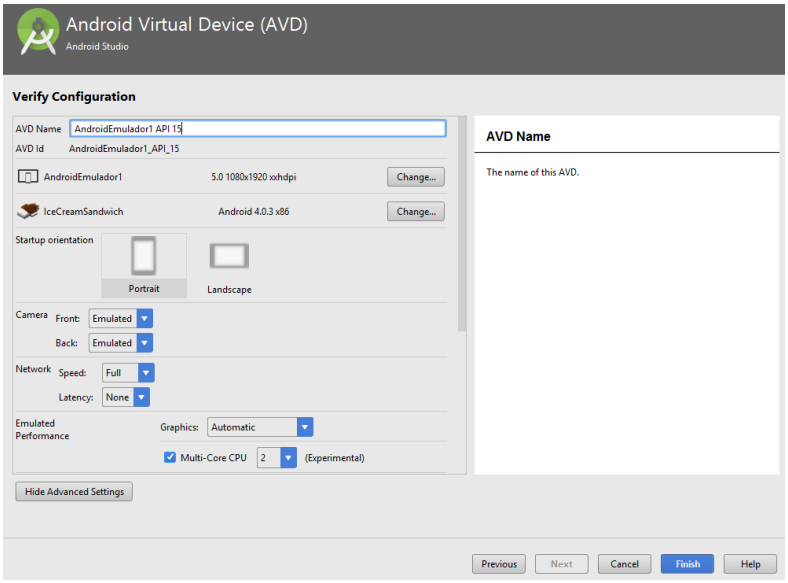

Figura 3.29.- Resumen del AVD

Por último aparecerá un resumen con todas las opciones seleccionadas, las cuales todavía pueden ser modificadas. Una vez seleccionada la opción "Finish", Android Studio comienza a crear el emulador. Para abrir dicho emulador simplemente se debe de hacer clic sobre el icono de Play. El AVD funciona igual que un dispositivo físico, se puede desbloquear, entrar al Menú, realizar llamadas a otro AVD etc.

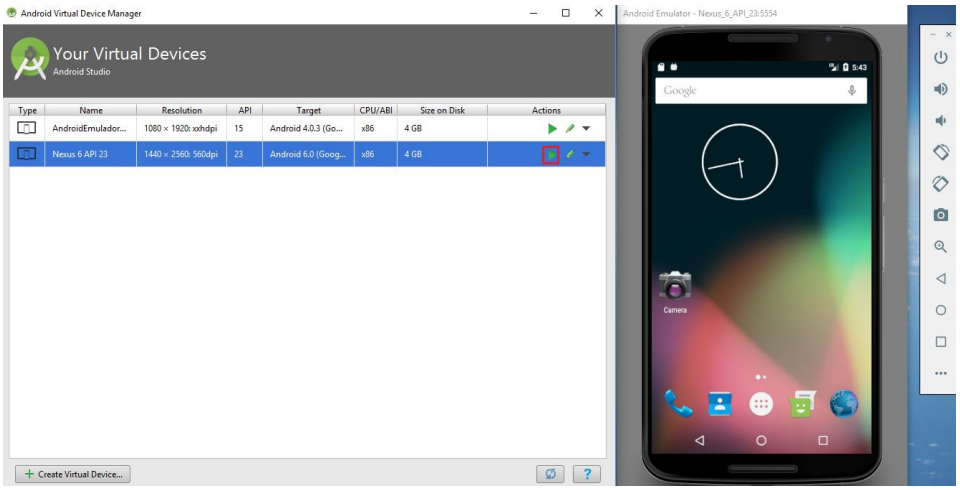

Figura 3.30.- AVD

3.2.4.2.- Dispositivo Android

Por otro lado se puede ejecutar la aplicación en un dispositivo Android real. Cuando se crea una aplicación en Android, es importante que se pruebe antes en un dispositivo real. Se puede usar cualquier dispositivo con tecnología Android para ejecutar, depurar y probar las aplicaciones. Las herramientas presentes en el SDK facilitan la instalación y la ejecución de las aplicaciones en el dispositivo cuando se realizan las compilaciones. De todas formas será necesario usar el emulador de Android para probar la aplicación en otras versiones que no son las del dispositivo real.

Los dispositivos con tecnología Android ofrecen diferentes alternativas a nivel de desarrollador. Éstas se encuentran disponibles en la opción "Opciones del desarrollador". Esta opción suele encontrarse en el apartado "Ajustes", pero en versiones Android 4.2 y posteriores se mantiene oculta de manera predeterminada. Para que se muestre, se deberá ir a "Ajustes" y posteriormente a "Información del teléfono". Una vez en este punto se debe buscar la opción "Número de compilación" y pulsarla siete veces. Al volver al menú de ajustes aparecerá la nueva opción "Opciones del desarrollador" que antes se encontraba oculta. El siguiente paso será el de activar la opción "Depuración USB" para que el dispositivo esté preparado para ejecutar aplicaciones (*Figura 3.31*).

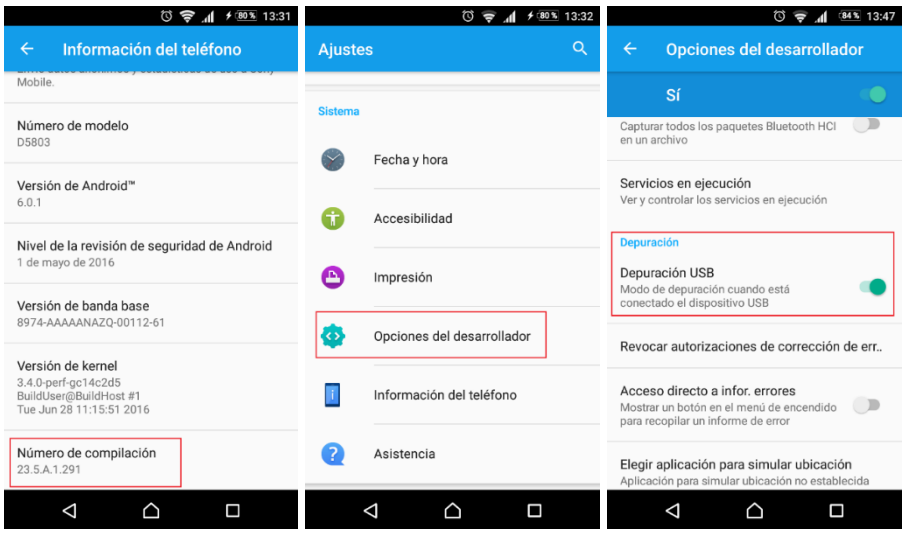

Figura 3.31.- Pasos para ejecutar aplicaciones en teléfono móvil

También es probable que sea necesario instalar un driver USB para que el sistema detecte el dispositivo móvil. En la página de Android Developers [14] hay una sección que se titula "Instalar controladores USB de OEM" en la cual se muestra una lista de los principales fabricantes de dispositivos, mediante la cual se accede a los drivers de cada uno.

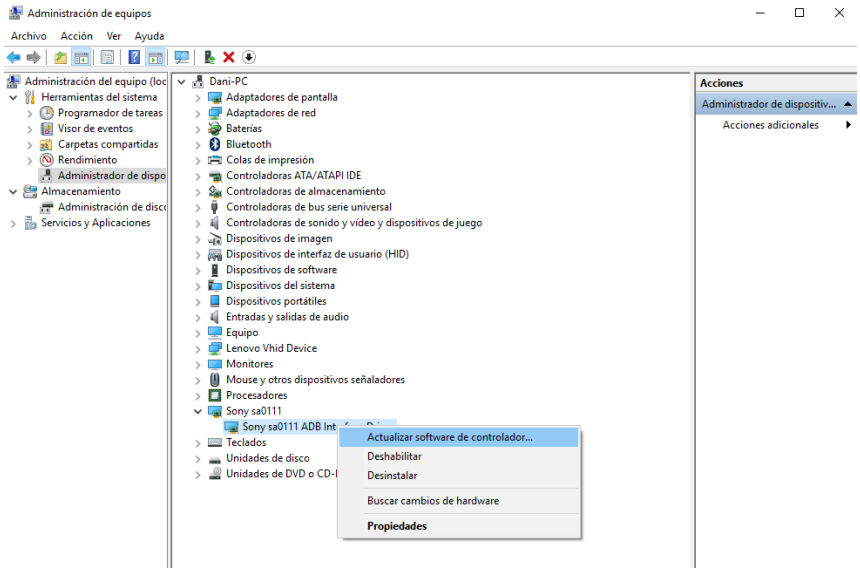

Figura 3.32.- Actualizar driver

El siguiente paso es el de acceder a la página del fabricante, escoger el modelo del dispositivo concreto en el cual se va a ejecutar la aplicación y descargar el driver. Una vez descargado, se conecta el dispositivo móvil al ordenador mediante el cable USB y se accede al "Administrador de dispositivos" (botón derecho sobre "Equipo" → "Administrar" → "Administrador de dispositivos") y se busca el dispositivo Android. A continuación se selecciona la opción "Actualizar software de controlador" y se escoge el driver que se descargó previamente (Figura 3.32).

Finalizada la actualización del controlador del dispositivo, éste ya podrá aceptar una aplicación del ordenador, es decir ya se pueden instalar en el teléfono las aplicaciones que se crean en Android Studio. Se puede comprobar el correcto funcionamiento de las mismas en el dispositivo móvil ejecutando una de ellas. Para ello se conecta el dispositivo móvil al ordenador mediante el cable USB si no lo estaba anteriormente, lo cual hará que aparezca en el dispositivo móvil la pantalla que se muestra en la *Figura 3.33* en cuanto se abra Android Studio. En esta pantalla habrá que aceptar el permiso para que se pueda llevar a cabo la depuración USB y de esta manera poder ejecutar las aplicaciones en el dispositivo móvil.

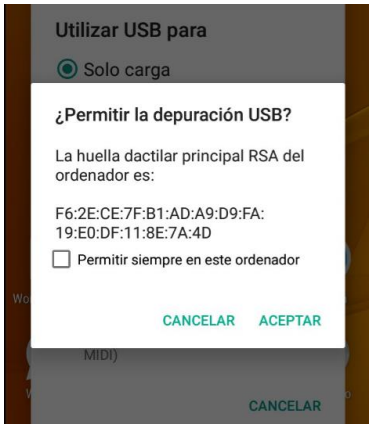

Figura 3.33.- Permiso para la depuración USB

Por último se debe seleccionar el botón de "Run 'app' " de Android Studio y escoger el dispositivo (en este caso el dispositivo físico) sobre el cual se quiere ejecutar la aplicación. Una vez elegido el dispositivo, la aplicación se debería ejecutar sin ningún problema.

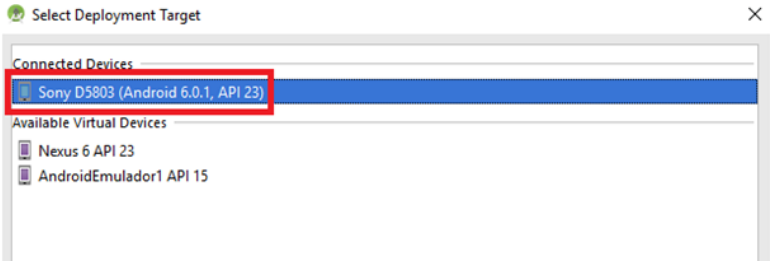

Figura 3.34.- Seleccionar dispositivo sobre el que ejecutar la app

4.- DESARROLLO DE LA APLICACIÓN

Mediante esta aplicación se pretende medir el rendimiento de una red y comprobar su viabilidad en un entorno vehicular. Para ello se analizarán parámetros como tiempo de descubrimiento, conexión, velocidad de envío, distancia etc.

En primer lugar resultará relevante la familiarización con las APIs Wi-Fi Peer-to-Peer relacionadas con la tecnología Wi-Fi Direct [14]. Éstas serán necesarias para llevar a cabo el proceso de descubrimiento y conexión con otros dispositivos. Dentro de las APIs de Wi-Fi P2P se distinguen las siguientes partes principales:

- Métodos para el descubrimiento, solicitud y conexión de peers los cuales están definidos en la clase WifiP2PManager.
- Listeners que envían notificaciones a cerca del éxito o fracaso de los métodos de la clase WifiP2PManager. Estos Listeners se pasan generalmente como parámetros en las funciones.
- Intents que informan sobre eventos especiales como por ejemplo la desconexión o descubrimiento de un nuevo peer.

4.1.- Pasos Iniciales para el Uso de las APIs P2P

Para el desarrollo de esta aplicación se establecerá como versión mínima la API 16 de Android (como se ha comentado anteriormente porque es la versión más baja de uno de los dispositivos de los que se dispone para las pruebas). Una vez escogida la API mínima el primer paso que a realizar será el de modificar el AndroidManifest.xml con el fin de añadir los permisos necesarios para poder hacer uso del Wi-Fi. Wi-Fi P2P no requiere de ninguna conexión a Internet pero utiliza el estándar de sockets de Java los cuales si necesitan este permiso.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Figura 4.1.- Permisos AndroidManifest.xml

Para poder utilizar los métodos de la clase WifiP2pManager se debe crear una instancia de la misma así como registrar la aplicación con el framework Wi-Fi P2P. De esta manera se obtiene una instancia del canal de transmisión, necesario para realizar cualquier operación P2P.

```
manager = (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
channel = manager.initialize( srcContext: this, getMainLooper(), listener: null);
```

Figura 4.2.- Instancia y registro WifiP2pManager

Posteriormente, una parte importante de la aplicación será la creación de la clase "BroadcastReceiver" para manejar los intents Wi-Fi P2P. De esta forma para cada evento detectado se actuará de una manera diferente. Para que el "BroadcastReceiver" pueda detectar estos "intents" será necesario crear un filtro con los eventos que se desean detectar.

```
intentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);
intentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);
intentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);
intentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);
```

Figura 4.3.- Intents Wifi P2P a detectar

Como se puede ver en la *Figura 4.3* se distinguen 4 "intents" principales:

- WIFI_P2P_STATE_CHANGED_ACTION: se habilita o deshabilita el Wi-Fi P2P en el dispositivo.
- WIFI_P2P_PEERS_CHANGED_ACTION: se produce algún cambio en la lista de peers.
- WIFI_P2P_CONNECTION_CHANGED_ACTION: se produce algún cambio en el estado de la conexión Wi-Fi P2P.
- WIFI_P2P_THIS_DEVICE_CHANGED_ACTION: se produce algún cambio relacionado con los detalles del dispositivo, como por ejemplo si se cambia el nombre o el estado del dispositivo (pasando por ejemplo a estar Available, Invited, Connected..)

Después de definir los intents que se desean recibir habrá que registrar el "intentFilter" y el "BroadcastReceiver" para que cuando se reciba uno de los eventos antes indicados se llame a este a último y se ejecuten las acciones deseadas para cada caso.

```
receiver = new WiFiDirectBroadcastReceiver(manager, channel, activity: this);
registerReceiver(receiver, intentFilter);
```

Figura 4.4.- Registro

Una vez obtenida la instancia del WifiP2pManager.Channel y configurado el "BroadcastReceiver" ya se podrán realizar llamadas a métodos Wi-Fi P2P así como recibir los intents Wi-Fi P2P. Por tanto, ahora se podrá hacer uso de todas las herramientas disponibles en la tecnología Wi-Fi P2P mediante la llamada a métodos de la clase WifiP2pManager.

4.2.- Descubrimiento y Conexión

En este apartado se explicará el procedimiento que se llevará a cabo para realizar el descubrimiento de peers así como la conexión. Para ello será importante comprender la utilidad de la clase WifiDirectBroadcastReceiver mediante la cual se manejarán los diferentes eventos que puedan tener lugar.

Dentro de la clase WifiDirectBroadcastReceiver se sobrescribirá el método "*onReceive"* perteneciente a la clase BroadcastReceiver. Este método es llamado cada vez que tiene lugar un evento de los filtrados anteriormente, y aquí será donde se indique la acción que se desea llevar a cabo para cada evento. Para ello se obtiene la acción que produjo la llamada a este método y se compara con los eventos filtrados.

```
@Override
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction(); // se obtiene la acción que ha tenido lugar para compararla y actuar en consecuencia

    if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) { // se comprueba si el WifiP2P está activo

        Log.d(WiFiDirectActivity.TAG,  msg: "Entré en un cambio en el estado del WIFI P2P(WIFI_P2P_STATE_CHANGED_ACTION), 2=Enabled, 1=Disabled");

        int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE,  defaultValue: -1);
        if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) { // Wifi Direct activo
            Log.d(WiFiDirectActivity.TAG,  msg: "El valor del estado del Wifi ha cambiado y ahora es " +state+ "= Enabled");
            activity.setIsWifiP2pEnabled(true); // guardo que está activo en la variable isWifiP2pEnabled

        } else { // Wifi direct no activo, estados: Disabled, enabling, disabling, unknown
            Log.d(WiFiDirectActivity.TAG,  msg: "El valor del estado del Wifi ha cambiado y no está habilitado (" +state+ ")");
            activity.setIsWifiP2pEnabled(false); // guardo que no está activo en la variable isWifiP2pEnabled
            activity.resetData(); // borro la información contenida en los fragmentos (lista de peers y fragmento detalles)
        }
```

Figura 4.5.- Verificación del estado Wi-Fi P2P

En primer paso será el de comprobar que el dispositivo soporta la tecnología Wi-Fi P2P y que ésta se encuentra activa. Para ello dentro de la clase BroadcastReceiver se obtiene el estado del Wi-Fi P2P y se guarda si éste se encuentra activo o no mediante la función setIsWifiP2PEnabled. Este es el paso previo a comenzar la búsqueda de peers, en caso de que el interfaz Wi-Fi no se encuentre activo se solicitará al usuario que lo active.

A continuación se comienza la búsqueda de Peers de manera automática. Para ello se hace uso del método "*discoverPeer*s" perteneciente a la clase P2PManager, el cual se encarga de comenzar el proceso de descubrimiento en búsqueda de otros pares que tengan el Wi-Fi P2P activo y con los cuales se pueda llevar a cabo el proceso de conexión. Como parámetros se pasan el canal de transmisión y un Listener que será el encargado de recibir si se ha podido iniciar o no la búsqueda.

```
manager.discoverPeers(channel, new WifiP2pManager.ActionListener() {
    //se comienza la busqueda de Peers hasta que se recibe el intent  WIFI_P2P_PEERS_CHANGED_ACTION, en ese momento se solicitará
    // la lista de Peers mediante requestPeers

    @Override
    public void onSuccess() { // se inicia el descubrimiento con exito
        Toast.makeText( context: WiFiDirectActivity.this,  text: "Búsqueda de Peer iniciada",
                Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onFailure(int reasonCode) { // error al iniciar el descubrimiento
        Toast.makeText( context: WiFiDirectActivity.this,  text: "Error en el proceso de búsqueda : " + reasonCode,
                Toast.LENGTH_SHORT).show();
    }
```

Figura 4.6.- Búsqueda de Peers

En el momento en el que aparezcan nuevos peers el sistema recibirá un evento del tipo "*WIFI_P2P_PEERS_CHANGED_ACTION*" que se detectará a través de la clase WifiDirectBroadcastReceiver. Una vez en este punto se solicitará la lista de peers a través del método "*requestPeers*" que tendrá cómo parámetros el canal y un listener que estará a la espera hasta que la lista se encuentre disponible.

```
else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {//Informa cuando hay cambios en la lista de peers

  Log.d(WiFiDirectActivity.TAG,  msg: "Entré en un cambio en la lista de peers(WIFI_P2P_PEERS_CHANGED_ACTION)");
  if (manager != null) {
      manager.requestPeers(channel, (PeerListListener) activity.getFragmentManager()
              .findFragmentById(R.id.frag_list)); // se solicita la lista de Peers
      Log.d(WiFiDirectActivity.TAG,  msg: "Solicité la lista de peers mediante requestPeers");
  }
```

Figura 4.7.- Solicitud lista de Peers

La lista solicitada anteriormente se recibirá como parámetro a través de la función "*OnPeersAvailable*". En este punto será donde se actualice la lista, para ello se borrarán los peers que pudiera haber con anterioridad y se añadirán los nuevos, recibidos como parámetros de la función. Además se notificará dicho cambio al adaptador para que se modifique la ListView y muestre la nueva lista de pares.

```
public void onPeersAvailable(WifiP2pDeviceList peerList) { // función que es llamada cuando la lista con los peers está disponible
    Log.d(WiFiDirectActivity.TAG,  msg: "Estoy dentro de OnPeerAvailable");

    peers.clear();//borro los elementos de la lista
    peers.addAll(peerList.getDeviceList());// añado la nueva lista de peers

    ((WiFiPeerListAdapter) getListAdapter()).notifyDataSetChanged(); //notifico el cambio al adaptador con el fin de modificar la ListView

    if (peers.size() == 0) {
        Log.d(WiFiDirectActivity.TAG,  msg: "No se ha encontrado ningun dispositivo");
        return;
    }
    Log.d(WiFiDirectActivity.TAG,  msg: " Peersize es: " + peers.size());
```

Figura 4.8.- Recepción de la lista de peers

Dentro de este método se llevarán a cabo los pasos necesarios para que la conexión se realice de manera automática. Es decir, cuando el usuario reciba la lista, éste se conectará automáticamente a uno de los peers especificados sin falta de que el usuario tenga que interaccionar con el dispositivo. En este caso se escoge el dispositivo Tablet Samsung Galaxy S2 utilizado para las pruebas de tal manera que, el otro dispositivo que realiza el descubrimiento de peers, permanecerá en un estado de búsqueda hasta que uno de los peers que aparezca en su lista sea "[Tablet] GTabS2", solicitando entonces su conexión automáticamente.

```
buscar =((DeviceActionListener) getActivity()).getBuscar();
if ( buscar ){
    for (int i=0; i<peers.size();i++) {
        WifiP2pDevice peerLista = (WifiP2pDevice) getListAdapter().getItem(i);

        if (peerLista.deviceName.equals("[Tablet] GTabS2")) {

        long tFinal = System.currentTimeMillis();
        ((DeviceActionListener) getActivity()).setTiempoDeteccion(tFinal);
        long tInicio= ((DeviceActionListener) getActivity()).getTiempoInicio();
        double tBusqueda = (tFinal - tInicio)/1000.0;
        Log.d(WiFiDirectActivity.TAG, msg: "Tiempo hasta que detecto el peer: " + tBusqueda );

            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss(); // quito el cuadro de dialogo una vez que  obtengo la lista de peers
            }
        configPeer(peerLista); // llamo a esta funcion para preparar la conexión
        }

        else if (peerLista.deviceName.equals("G6")) {

        long tFinal = System.currentTimeMillis();
        ((DeviceActionListener) getActivity()).setTiempoDeteccion(tFinal);
        long tInicio= ((DeviceActionListener) getActivity()).getTiempoInicio();
        double tBusqueda = (tFinal - tInicio)/1000.0; //añadido
        Log.d(WiFiDirectActivity.TAG, msg: "Tiempo hasta que detecto el peer: " + tBusqueda );

            if (progressDialog != null && progressDialog.isShowing()) {
                progressDialog.dismiss();
            }
```

Figura 4.9.- Análisis lista de peers

Por otro lado se utiliza la variable "*Buscar*" para comprobar si se están buscando peers o no, ya que hay ocasiones en las que la lista de peers se actualiza sin realizar la búsqueda. Si se están buscando peers se recorrerá la lista de peers en busca del dispositivo "[Tablet] GTabS2" y se llamará a la función configPeer. En esta función se completarán los campos necesarios de la clase WifiP2PConfig para que posteriormente, se pueda llevar a cabo la conexión entre dispositivos.

```
public void configPeer(WifiP2pDevice peerLista){

    Log.d(WiFiDirectActivity.TAG, msg: "Entré en conectarpeers");
    WifiP2pConfig config = new WifiP2pConfig(); // clase que permite llevar a cabo una configuración P2p
    config.deviceAddress = peerLista.deviceAddress; // se configura la dirección MAC
    config.wps.setup = WpsInfo.PBC; // Wi-Fi Protected Setup
    config.groupOwnerIntent = 15; // Este dispositivo tiene una prioridad para ser el GO de 15

    if (progressDialog != null && progressDialog.isShowing()) { //Si se está mostrando el Dialogo de Progreso
        progressDialog.dismiss(); // quitar el dialogo de progreso
    }
    progressDialog = ProgressDialog.show(getActivity(), title: "Presional atrás para cancelar",
            message: "Conectando con el dispositivo :" + peerLista.deviceName + " cuya MAC es :" + peerLista.deviceAddress,
    );
    ((DeviceActionListener) getActivity()).connect(config); // llamom a función connect mediante el interfaz para
    // llevar a cabo la conexión pasando como parametro la configuracion
}
```

Figura 4.10.- Configuración del Peer a conectar

Como último paso se lleva a cabo el intento de conexión mediante el método connect. Para ello se pasan como parámetros el canal, el objeto WifiP2PConfig que contiene información del dispositivo con el que se llevará a cabo la conexión, así como un listener a través del cual se indicará si el intento de conexión está realizándose correctamente o no.

```
public void connect(WifiP2pConfig config) {
    Log.d(WiFiDirectActivity.TAG,  msg: "Entré en connect");
    manager.connect(channel, config, new ActionListener() {

        @Override
        public void onSuccess() {
            Log.d(WiFiDirectActivity.TAG,  msg: "Conectado con exito");
            buscar= false;
        }

        @Override
        public void onFailure(int reason) {
            Toast.makeText( context: WiFiDirectActivity.this,  text: "Fallo en la conexión.Volver a intentar",
                    Toast.LENGTH_SHORT).show();
            Log.d(WiFiDirectActivity.TAG,  msg: "Fallo en la conexion Motivo:" + reason);
        }
```

Figura 4.11.- Conexión entre dispositivos

Cuando se produzca la conexión y también para el caso de la desconexión, el sistema recibirá un evento del tipo "*WIFI_P2P_CONNECTION_CHANGED_ACTION*" que, al igual que sucedía para otros eventos, será detectado a través de la clase "WifiDirectBroadcastReceiver". En este caso mediante la clase "NetworkInfo" se distinguirá si se trata de una conexión o una desconexión. En el caso de tratarse de una conexión, se solicitará información sobre la misma mediante el método "*requestConnectionInfo*".

```
else if (WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)) {

  Log.d(WiFiDirectActivity.TAG,  msg: "Cambio en el estado de la conexión con otros P2P(WIFI_P2P_CONNECTION_CHANGED_ACTION)");
  if (manager == null) {
      return;
  }
  NetworkInfo networkInfo = intent.getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);

  if (networkInfo.isConnected()) {
      Log.d(WiFiDirectActivity.TAG,  msg: "Es una conexión y voy a llamar a onConnectionInfo");
      DeviceDetailFragment fragment = (DeviceDetailFragment) activity
              .getFragmentManager().findFragmentById(R.id.frag_detail);
      manager.requestConnectionInfo(channel, fragment);
  } else {
      Log.d(WiFiDirectActivity.TAG,  msg: "Es una desconexión, llamo a resetData");
      activity.resetData();
  }
```

Figura 4.12.- Cambio en la conexión

En el momento en que esta información se encuentre disponible será recibida a través del método onConnectionInfoAvailable, invocado a través del listener ConnectionInfoListener. Posteriormente en este método se obtiene información acerca de que dispositivo actúa como group owner y con qué IP. De esta forma se establecerá que el dispositivo que actúe como group owner será el que reciba los mensajes mientras que el otro dispositivo será el que los transmita.

```
TextView view = mContentView.findViewById(R.id.group_owner);
view.setText(((info.isGroupOwner) ? getResources().getString(R.string.GO)
        : getResources().getString(R.string.no_GO)));

view = mContentView.findViewById(R.id.device_info);
view.setText("La IP del Group Owner es - " + info.groupOwnerAddress.getHostAddress());

if (info.groupFormed && info.isGroupOwner ) {
    Log.d(WiFiDirectActivity.TAG, msg: "Entré en Llamar ServerAsyncTask");
       Hilo = new ServerAsyncTask(getActivity(), mContentView.findViewById(R.id.status_text),
               mContentView.findViewById(R.id.texto_recibido));
       Hilo.execute();

   }else if (info.groupFormed) {
    Log.d(WiFiDirectActivity.TAG, msg: "No soy el Group Owner");
    ((TextView) mContentView.findViewById(R.id.status_text)).setText("Este dispositivo actuará como cliente");

    solicitudEnvio();
```

Figura 4.13.- Group Owner

Para hacer que un dispositivo actúe con cliente y otro como servidor se separa en dos casos principales, si el grupo está formado y el dispositivo es el Group Owner, se crea un nuevo hilo del tipo AsyncTask el cual actuará como servidor a la espera de recibir conexiones. En el caso de no ser el group owner, se actuará como cliente y se procederá a realizar el envío de información.

4.3.- Intercambio de Información

El intercambio de información entre dispositivos se llevará a cabo mediante sockets, dónde un dispositivo actuará como como servidor y otro como cliente. El dispositivo que actué como servidor (GO) permanecerá a la escucha hasta que le lleguen peticiones de conexión. Por su parte el equipo cliente será el encargado de realizar las peticiones de conexión para, posteriormente, llevar a cabo el envío de información.

Se han implementado diferentes pruebas relacionadas con el envío de información las cuales se explican en el *Capítulo 5 Pruebas Realizadas.* En primer lugar se ha creado el socket para, posteriormente, realizar el envío mediante el protocolo TCP. Se ha decidido utilizar TCP, teniendo en cuenta su fiabilidad, ya que si se desease enviar un mensaje importante mediante el protocolo UDP, no quedaría garantizada la recepción del mismo. La variable "*prueba*" guardará el valor de la prueba que se está llevando a cabo. Se han distinguido los siguientes casos:

- **tcp:** Envío de 100 MB mediante el protocolo TCP.
- **tcpbucle:** Pruebas para comprobar el tiempo de envío en TCP.
- **tcpShort:** Envío de tamaño pequeño (dos frases) mediante el protocolo TCP.

Comenzando por la parte del servidor, será necesario implementar un hilo secundario mediante la clase AsyncTask. Esto es así ya que en Android, las actividades, servicios, broadcast receivers, operaciones de gestión de la interfaz de usuario etc. son implementadas en el hilo principal, lo que implica que operaciones costosas computacionalmente pueden

hacer que se bloquee este hilo. De hecho el sistema impide acciones que requieran de operaciones de red (como por ejemplo peticiones al servidor) en el hilo principal, mostrándose una excepción del tipo NetworkOnMainThreadException.

Para el uso de la clase AsyncTask resulta fundamental comprender sus cuatro métodos principales, los cuales se sobrescribirán con el fin de implementar las funcionalidades propias de la tarea que se desea llevar a cabo. Los métodos son los siguientes:

- onPreExecute(): se ejecuta en el hilo principal antes del código de la tarea. Se suele utilizar para preparar la ejecución de la tarea, interfaz gráfica etc.
- doInBackground(): se ejecuta justo después del onPreExecute() en el hilo secundario (por tanto no se puede interactuar con la interfaz). Contendrá el código principal de la tarea, se suele utilizar para llevar a cabo los cálculos que puedan necesitar más tiempo. El resultado de estas operaciones se pasarán al método onPostExecute(). Este método también puede usar publishProgress() para ir publicando cambios en la interfaz gráfica.
- onProgressUpdate(): se ejecuta en el hilo principal justo después de que se produzca una llamada a publishProgress(). Este método se suele utilizar para mostrar actualizaciones en la interfaz gráfica mientras todavía se sigue ejecutando el hilo secundario.
- onPostExecute(): se ejecuta en el hilo principal una vez finalicen las tareas llevadas a cabo en segundo plano.

Inicialmente se utilizó onProgressUpdate, de tal manera que éste hilo secundario no se cerraba salvo que saltase alguna excepción, De esta forma se implementaba un bucle infinito en el cual el servidor estaba siempre a la escucha, recibía la información y volvía a ponerse a la escucha. Finalmente, y con el objetivo de añadir un tiempo de espera en el servidor se ha decidido que una vez los peers se conecten y envíen la información se cierre el socket mostrándose la información mediante onPostExecute, de tal manera que posteriormente también se finalice este hilo secundario.

Para realizar la recepción mediante TCP, se crea un ServerSocket que se encuentra a la espera de conexiones, en cuanto reciba una petición, la acepta y se genera el socket de datos a través del cual se recibirá la información. Para ello se reserva un tamaño de mensaje de 100 MB. Además, en el caso de que se produjese un error durante la recepción se lanzan las excepciones pertinentes (SocketTimeoutExcepction y SocketException) con el fin de obtener los datos recibidos durante el periodo anterior a que saltase la misma. El valor retornado será “Recibido” para los envíos de 100 MB y, la cadena enviada, para el caso de los envíos pequeños de información.

```
protected String doInBackground(Void... params) {
    try {
        Log.d(WiFiDirectActivity.TAG, msg: "Entré en doInBackground");

        if (tcpudp.equals("tcp")) {

            String msg_received = "";
            Log.d(WiFiDirectActivity.TAG, msg: "Entre en tcp");
            ServerSocket serverSocket = new ServerSocket(SERVERPORT); // socket escucha, se especifica el puerto
            Log.d(WiFiDirectActivity.TAG, msg: "Servidor: Socket escuchando");
            Socket cliente = serverSocket.accept(); // socket de datos en el servidor
            long tInicio = System.currentTimeMillis();
            Log.d(WiFiDirectActivity.TAG, msg: "Servidor: conexion establecida");

            TrafficStats.setThreadStatsTag(0xF00D);
            TrafficStats.tagSocket(cliente);

            DataInputStream DIS = new DataInputStream(cliente.getInputStream());
            int timeout = 10000;
            Log.d(WiFiDirectActivity.TAG, msg: "TimeOut 10 segundos");
            cliente.setSoTimeout(timeout);

            int tamMensaje = (100 * 1000 * 1000);
            byte[] msg_received2 = new byte[tamMensaje];

            long tInicioRecepcion = System.currentTimeMillis();

            try{
            DIS.readFully(msg_received2);
            }catch(SocketTimeoutException e) {
```

Figura 4.14.- Servidor TCP

El dispositivo que no sea el GO será el que actúe como cliente y llevará a cabo el envío utilizando un IntenService para crear una tarea en segundo plano. En este caso se pasarán las variables que interesen a través de intent y se comenzará el servicio.

```
public void solicitudEnvio (){
    Log.d(WiFiDirectActivity.TAG, msg: "Estoy dentro de SolicitudEnvio " );
    Intent serviceIntent = new Intent(getActivity(), SendBytes.class); //intent explicito indico en que
    serviceIntent.putExtra(SendBytes.EXTRAS_GROUP_OWNER_ADDRESS,
            info.groupOwnerAddress.getHostAddress()); // dirección del GO;

    serviceIntent.putExtra(SendBytes.EXTRAS_PROTOCOLO, tcpudp); // le paso los datos que me interesan
    long tInicioBusqueda=((DeviceActionListener) getActivity()).getTiempoInicio();
    long tInicioDeteccion = ((DeviceActionListener) getActivity()).getTiempoDeteccion();
    long tInicioConexion = ((DeviceActionListener) getActivity()).getTiempoConexion();
    serviceIntent.putExtra(SendBytes.EXTRAS_tInicioBusqueda, tInicioBusqueda);
    serviceIntent.putExtra(SendBytes.EXTRAS_tInicioDeteccion, tInicioDeteccion);
    serviceIntent.putExtra(SendBytes.EXTRAS_tInicioConexion, tInicioConexion);
    serviceIntent.putExtra(SendBytes.EXTRAS_GROUP_OWNER_PORT, SERVERPORT);
    serviceIntent.setType("text/plain");

    getActivity().startService(serviceIntent); //lanzo el intent e incio el servicio
```

Figura 4.15.- Preparación del cliente para el envío

Una vez se ha iniciado el servicio se crea el Socket y se realiza el intento de conexión a la dirección y puerto en la que el servidor se encuentra a la espera. Posteriormente se envía automáticamente un mensaje de 100MB de carga útil. Al igual que en el receptor, en el caso de que se produjese un error durante el envío se lanzan las excepciones pertinentes.

```
if (tcpudp.equals("tcp")) { //envio 100 MBytes mediante TCP
    Socket socket = new Socket();
    try {
        long tByteInicio = System.currentTimeMillis();
        Log.d(WiFiDirectActivity.TAG, msg: "Entre en Tcp");
        int tamanio = 100 * 1000 * 1000; // 100 MByte de tamaño de mensaje
        byte[] bytesMensaje = new byte[tamanio]; // byte es 1 byte de un valor de -128 a 127
        int tamMensaje = bytesMensaje.length;
        Log.d(WiFiDirectActivity.TAG, msg: "Bytes del mensaje a enviar " + tamMensaje);
        for (int x = 0; x < bytesMensaje.length; x++) {
            bytesMensaje[x] = 5;
        }

        long tByteFinal = System.currentTimeMillis();
        double tByte = (tByteFinal - tByteInicio) / 1000.0;
        Log.d(WiFiDirectActivity.TAG, msg: "El tiempo en cargar los 100MB es " + tByte);

        Log.d(WiFiDirectActivity.TAG, msg: "Abriendo el socket cliente - ");
        socket.bind( bindpoint: null);// sistema vincula puerto y dir.
        Log.d(WiFiDirectActivity.TAG, msg: "Tratando de conectarme a " + host + " " + port);
        socket.connect((new InetSocketAddress(host, port)), SOCKET_TIMEOUT);
        Log.d(WiFiDirectActivity.TAG, msg: "Mi ip es" + socket.getLocalAddress());

        TrafficStats.setThreadStatsTag(0xF00D);
        TrafficStats.tagSocket(socket);

        DataOutputStream DOS = new DataOutputStream(socket.getOutputStream());
        long tInicioEnvio = System.currentTimeMillis();
    try{
        DOS.write(bytesMensaje); // antes ponia 1024
        DOS.flush();
```

Figura 4.16.- Cliente TCP

4.4.- Ficheros

Una vez realizado el envío y recepción, de cara a facilitar el análisis posterior de la información obtenida en las pruebas se procede a guardar los datos que se consideran relevantes en un fichero ubicado en la memoria interna del dispositivo.

```
try
{
    OutputStreamWriter fout= new OutputStreamWriter(context.openFileOutput( name: "Pruebas.txt", Context.MODE_APPEND));
    Log.d(WiFiDirectActivity.TAG, msg: "Voy a escribir");
    fout.write( str: "Prueba realizada a las: " + Calendar.getInstance().getTime() + " " + tcpudp + "\n");
    fout.write( str: "Numero de Bytes recibidos hasta que se cortó la comunicacion: " + recibido + " bytes \n");
    fout.write( str: "Tiempo desde que selecciono buscar hasta que recibo es: " + tBusqueda + " segundos \n");
    fout.write( str: "Tiempo desde que detecto el peer hasta que recibo es: " + tDeteccion + " segundos\n");
    fout.write( str: "Tiempo desde que está conectado hasta que recibo es: " + tConexion + " segundos\n");
    fout.write( str: "Tiempo desde que el socket acepta la conexión(tiempo Tx-Rx) hasta que recibo es: " + tSocket + " segundos\n");
    fout.write( str: "El tiempo de recepcion es de " + tRx + " segundos\n");
    fout.write( str: "La velocidad desde que el socket recibe: " + velocidadMbpsSock + " Mbps\n");
    fout.write( str: "La velocidad de Rx es: " + velocidadMbps + " Mbps\n\n\n");

    fout.close();
}
catch (Exception fi)
{
    Log.e( tag: "Fichero", msg: "Error al escribir en el fichero");
}
```

Figura 4.17.- Escritura de variables en Fichero

Para ello mediante el método openFileOutput se especifica el nombre del fichero y el modo de acceso con el que se desea abrir el mismo. En este caso se selecciona "Mode_Append" para que de esta manera se puedan acumular datos sin sobrescribirlos. Este fichero se almacena en la memoria interna del teléfono, concretamente en la siguiente ruta:

>data/data/nombre_del_paquete/files/nombre del fichero.

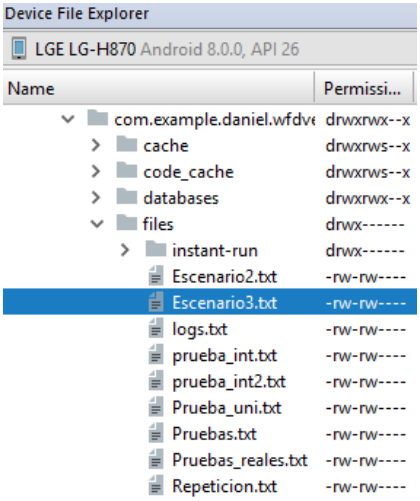

Figura 4.18.-Ubicación Ficheros

Cabe mencionar que en algunos dispositivos Samsung no está permitido el acceso como se muestra en la siguiente figura:

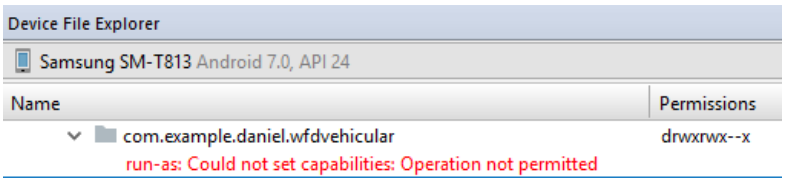

Figura 4.19.- Error acceder al fichero dispositivos Samsung

Para poder acceder al fichero en estos dispositivos se puede hacer una copia de seguridad a través del ADB de Android. Para ello habrá que acceder a la ruta que se muestra en la *Figura 4.20* e introducir el comando especificado.

```
C:\Users\Daniel\AppData\Local\Android\sdk\platform-tools>adb backup -f data.ab com.example.daniel.wfdvehicular
Now unlock your device and confirm the backup operation...
```

Figura 4.20.- Comando para realizar copia de seguridad de la aplicación

Una vez introducido el primer comando, se pedirá que se acepte el la copia de seguridad en el dispositivo.

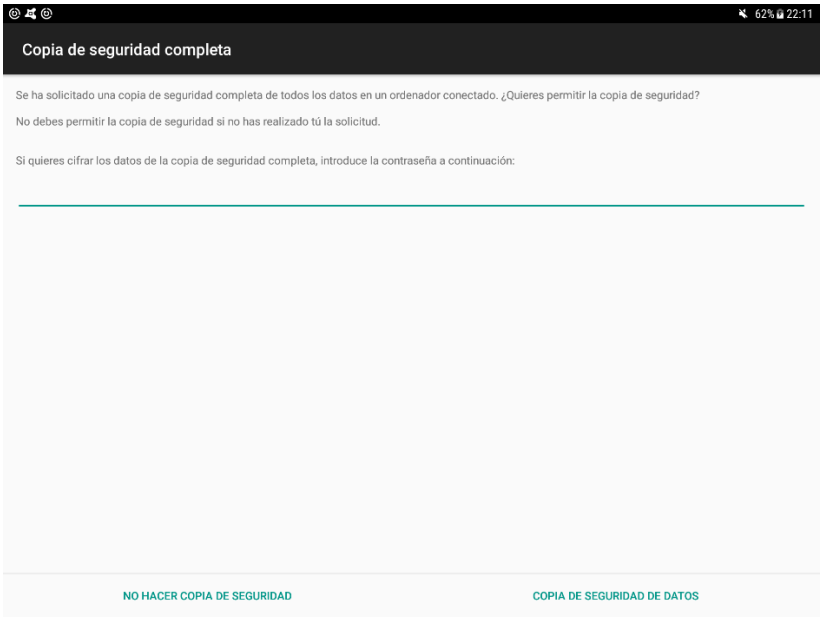

Figura 4.21.-Copia de seguridad en dispositivos Samsung

Por último para acceder al archivo se puede introducir el comando que aparece en la *Figura 4.22*. Para poder introducir este comando ha tenido que descargarse el Cygwin [17] ya que de otra manera no se reconocía el comando openssl.

```
Daniel@Dani-PC /cygdrive/c/users/Daniel/AppData/Local/Android/sdk/platform-tools
$ dd if=data.ab  bs=24 skip=1 | openssl zlib -d > data.tar
35+1 registros leídos
35+1 registros escritos
860 bytes copied, 0,000370348 s, 2,3 MB/s
```

Figura 4.22.- Comando para acceder a la copia de seguridad a través de SSL

4.5.- Localización

Para conocer la ubicación del dispositivo en primer lugar se deberá escoger que sistema de localización se va a utilizar el basado en satélites o GPS o el basado en Redes inalámbricas (telefonía móvil, Wi-Fi). En este caso, al tratarse de una aplicación en exteriores y para la cual se requiere la máxima precisión, el sistema escogido es el GPS. Después de haber elegido el proveedor del servicio el siguiente paso será el de comprobar si el GPS se encuentra activo.

```
    //compruebo si el GPS está activo
    locManager = (LocationManager) getSystemService(LOCATION_SERVICE);
     if (!locManager.isProviderEnabled(LocationManager.GPS_PROVIDER)){
         activarGPS();
     }
        boolean fichero = false;
        long tpos1 = System.currentTimeMillis();
        posicion(tpos1,fichero); // Calculo una primera posición para tener
}

public void activarGPS(){
    new AlertDialog.Builder( context: this)
    .setTitle("Activación GPS")
    .setMessage("EL GPS no está áctivo")
    .setPositiveButton( text: "Activar",
        new DialogInterface.OnClickListener(){
         public void onClick(DialogInterface dialog, int boton){
         Intent i = new Intent (Settings.ACTION_LOCATION_SOURCE_SETTINGS);
         startActivity(i);
         }})
            .setNegativeButton( text: "Cancelar", listener: null)
            .show();
```

Figura 4.23.- Comprobación del estado del GPS

Una vez comprobado el estado del GPS se calculará una primera vez la posición del GPS a través de la función *posición (tpos, fichero)*, meramente por tener un dato y que la siguiente vez que se calcule la localización, que será cuando los dispositivos se desconecten, la localización se produzca de una manera más rápida. A través de esta función se pasan dos parámetros, uno es el momento exacto en el que se solicitó la posición con el fin de saber cuánto se tarda en recibirla, y el otro es un booleano que tiene la función de indicar si se quiere escribir en el fichero.

```
public void posicion (final long tpos1, final boolean fichero) {
    if (ActivityCompat.checkSelfPermission( context: this,
        Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {

         Log.d(WiFiDirectActivity.TAG, msg: "No hay permisos Fine Location");
         ActivityCompat.requestPermissions( activity: this,
                 new String[]{Manifest.permission.ACCESS_FINE_LOCATION},
                 PERMISO_LOCALIZACION);
    } else{
        Log.d(WiFiDirectActivity.TAG, msg: "Hay permisos para acceder a la ubicacion de este dispositivo");
        locManager.requestSingleUpdate(LocationManager.GPS_PROVIDER, new LocationListener() {
            public void onLocationChanged(Location location) {
                long tpos2 = System.currentTimeMillis();
                double tposicion=(tpos2-tpos1)/1000.0;
                Log.d(WiFiDirectActivity.TAG,  msg: "Tiempo en obtener la posicion: " + tposicion + " segundos");
                Location Coordenadas = location;
                float precision = location.getAccuracy();
                Log.d(WiFiDirectActivity.TAG, msg: "Precision de " + precision + " metros" );
                double longitud=Coordenadas.getLongitude();
                double latitud= Coordenadas.getLatitude();
                Log.d(WiFiDirectActivity.TAG, msg: "Coordenadas obtenidas ; Latitud : " + latitud + " Longitud: " + longitud );
```

Figura 4.24.- Cálculo de la posición

Para obtener la posición es necesario haber concedido los permisos de localización en el "AndroidManifest.xml". En este caso como se desea la máxima precisión se conceden los permisos de alta precisión (Access_Fine_Location). Además a partir de Android 6.0 es necesario pedir algunos permisos en tiempo de ejecución, por lo que se le solicita al usuario

los permisos de localización mediante *checkSelfPermission()*. Una vez solicitados y aceptados los permisos de localización se solicita la posición mediante la función requesSingleUpdate, pasando como parámetros el proveedor, en este caso el GPS, y el interfaz LocationListener. En este interfaz, el método que se utilizará para obtener la posición será onLocationChanged como se muestra en la *Figura 4.24*.

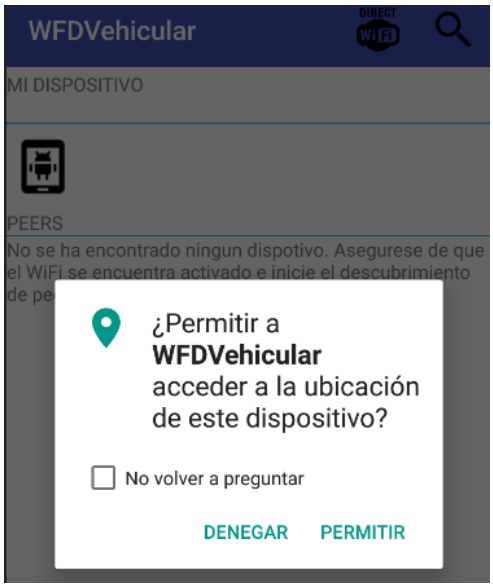

Figura 4.25.- Permiso de localización

Por último, para calcular la posición del dispositivo en el momento de la desconexión se han introducido varios logs en el código con el fin de comprender como actúan los eventos en la desconexión. Una vez analizados, se puede concluir que el primer evento en el que se detecta un cambio en el estado del peer, pasando de Connected a Available (Peer Status: 3), es WIFI_P2P_THIS_DEVICE_CHANGED_ACTION. Será entonces, en este momento cuando se haga la llamada a la función *posición(tpos,fichero)* calculando así la ubicación.

```
D/WFDVehicular: Entré en un cambio de un dispositivo P2P(WIFI_P2P_THIS_DEVICE_CHANGED_ACTION)
D/WFDVehicular: Entre en updateThisDevice, actualiza vista con mi nombre y mi estado (llama a la función getDeviceStatus)
D/WFDVehicular: Peer status :3

D/WFDVehicular: Entré en un cambio en el estado de la conexión con otros P2P(WIFI_P2P_CONNECTION_CHANGED_ACTION)
D/WFDVehicular: Es una desconexión, llamo a resetData
```

Figura 4.26.- Logs generados en la desconexión de los peers

5.- PRUEBAS REALIZADAS

En este capítulo se definirán las diferentes pruebas a realizar durante la implementación de este proyecto. Para ello este apartado se subdividirá en dos partes principales, uno con las primeras pruebas llevadas a cabo con el fin de entender ciertos aspectos relacionados con la tecnología Wi-Fi Direct, y un segundo apartado en el cual se plantearan una serie de escenarios (en parado y en movimiento, variando velocidades) sobre los que se producirá un intercambio de datos entre dispositivos mediante esta tecnología.

5.1.- Pruebas Iniciales

En primer lugar, en este apartado se van a realizar una serie de pruebas que ayuden a comprender mejor el uso de la tecnología Wi-Fi Direct. Para ello se tomará contacto con conceptos como GO Intent, grupo persistente, direccionamiento asociado o comunicación entre varios dispositivos.

Como se ha comentado anteriormente en el *Capítulo 2*, cuando se forma un grupo P2P, se negocia cuál de los dispositivos actuará como Group Owner. Esta negociación se lleva a cabo mediante teniendo en cuenta la variable GO Intent, la cual puede tomar un valor entre 0 a 15 siendo 15 la máxima prioridad y 0 la mínima. Por tanto, el dispositivo que tenga esta variable con el valor más alto será el GO. En caso de coincidir estos valores el GO será escogido aleatoriamente.

Con el fin de verificar que esto sucede así se han realizado una serie de pruebas con diferentes dispositivos Android:

- Samsung Galaxy S2 (Android 4.1.2, API 16)
- Samsung Galaxy tab S (Android 5.0.2, API 21)
- Samsung Galaxy tab S2 (Android 7.0, API 24)
- Sony Xperia Z3 compact (Android 6.0.1, API 23)
- Sony Xperia Z3 (Android 6.0.1, API 23)
- Bq Aquaris M5 (Android 7.1.2, API 25)
- LG G6 (Android 7.0, API 24) (8.0.0,API26)

Para indicar el valor del GO Intent que se desea dar al dispositivo se debe introducir el siguiente comando:

config.groupOwnerIntent = *<nº entero de 0 a 15>*

Durante estas pruebas se han conectado los dispositivos y se ha comprobado que dispositivos actuaban como GO en diferentes situaciones; especificando un GO Intent y sin especificarlo.

En cuanto a la formación de grupos se ha comprobado la posibilidad de implementar un grupo persistente, es decir un grupo que sea memorizado y que, en futuras conexiones, no necesite llevar a cabo una negociación, si no que mantenga los roles que anteriormente

se habían asignado. De igual modo resulta relevante intentar realizar una conexión entre varios dispositivos para verificar que está puede llevarse a cabo y que existe comunicación entre los mismos.

Otro punto que se ha analizado es el del direccionamiento IP, comprobando qué dirección es asignada a cada dispositivo, así como el interfaz que se utiliza para ello obteniendo valores relevantes para el envío de información como pueden ser los de la MTU, especialmente para el caso de enviar grandes cantidades de información. En este caso se han hecho diferentes envíos de 100 MB, para comprobar si influye de alguna manera el hecho de enviar la información de golpe, o mediante un bucle de diferentes dimensiones. Siguiendo en esta línea se han hecho varios envíos de 100 MB con el fin de comprobar el valor de diferentes parámetros como tiempos de detección, conexión y recepción así como las velocidades obtenidas. Además se ha comprobado cómo influyen los obstáculos intermedios en la velocidad.

Por último se ha hecho una prueba en la cual se ha enviado la frase "Por favor, reduzca la velocidad. Accidente en 1km sentido Oviedo-Gijon" (prueba = tcpShort) con el fin de probar una posible aplicación de esta tecnología en una situación en la que se quisiera alertar de un accidente, analizando los tiempos obtenidos de detección conexión y recepción.

5.2.- Pruebas en Entornos Vehiculares

Las pruebas a realizar se llevaran a cabo en diferentes escenarios vehiculares, buscando valorar la utilidad de esta tecnología y comprobando su funcionalidad en diferentes situaciones reales. Para llevar a cabo estas pruebas ha sido necesario contar con los siguientes elementos:

- Dos dispositivos con sistema operativo Android. Concretamente se ha usado la tablet Samsung Galaxy Tab S2 y el smartphone LG G6.
- Dos vehículos.
- Un ordenador, desde el cual se han monitorizado los resultados obtenidos.

La ubicación elegida para el desarrollo de las pruebas ha sido la Escuela Politécnica de Ingeniería de Gijón. Se ha escogido este emplazamiento debido a que tiene una extensión suficiente para llevar a cabo las pruebas así como por su facilidad para la repetición de las mismas.

Figura 5.1.- Ubicación elegida para el desarrollo de las pruebas.

Para evaluar las prestaciones de la tecnología Wi-Fi Direct se hará un envío masivo de 100 MB mediante el protocolo TCP. Se escogen 100 MB porque se considera una cantidad bastante elevada, que permite acotar sobradamente la cantidad máxima de información que puede ser transmitida en los escenarios en los que, al menos un vehículo, se encuentre en movimiento. Además al enviar este caudal de información también se consigue obtener un valor más estabilizado de la velocidad a la que se recibe la misma.

A modo de apunte, también se realizará un pequeño envío de información, concretamente la frase “Por favor, reduzca la velocidad. Accidente en 1km, sentido Oviedo-Gijón.” Para visualizar lo que podría ser una aplicación de esta tecnología.

5.2.1.- Escenarios

Para la realización de las pruebas se analizaran principalmente tres escenarios diferentes que tienen lugar mientras se conduce un vehículo. El primer escenario sería aquel en el que dos vehículos se encontrasen en reposo. El siguiente escenario comprendería a un vehículo en movimiento y otro en reposo y, el último escenario, se llevaría a cabo entre dos vehículos que se encontrasen en movimiento.

5.2.1.1.- Escenario 1: Vehículos en reposo

En primer lugar, como se ha comentado anteriormente se realizaran las pruebas con ambos vehículos en reposo, situación que podría tener lugar entre vehículos que estuviesen parados en un semáforo, o entre un vehículo y un dispositivo instalado en el propio semáforo o en un poste cercano. En este escenario se llevarán a cabo envíos a diferentes distancias (5m, 35m y 80m) con el fin de comprobar la influencia de la misma en el intercambio de información.

Figura 5.2.- Pruebas realizadas con vehículos en reposo

5.2.1.2.- Escenario 2: Un vehículo en movimiento y otro en reposo

El segundo escenario planteado sería aquel en el cual un dispositivo se encontrase en movimiento, y el otro estuviese parado sin variar su posición. Este escenario podría tener lugar cuando un vehículo en movimiento se cruza con un semáforo, poste, panel informativo u otro vehículo parado en el arcén, el cual por ejemplo, buscase alertar de un accidente, congestión del tráfico u otro tipo de situación. Estas pruebas se llevarán a cabo a diferentes velocidades para comprobar la influencia de la misma sobre los resultados obtenidos. Concretamente se realizarán a 30 y 50 Km/h.

Figura 5.3.- Pruebas realizadas con vehículos en reposo

5.2.1.3.- Escenario 3: Ambos vehículos en movimiento.

El último escenario planteado sería aquel en el cual ambos vehículos se estuvieran desplazando. Esta situación podría darse bien entre dos vehículos que se cruzaran en calzadas de doble sentido, así como entre vehículos que se estén desplazando en el mismo sentido de circulación. Al igual que en el escenario anterior se llevarán a cabo pruebas a diferentes velocidades para comprobar la influencia de este parámetro sobre los resultados finales. Se han elegido las mismas velocidades que para el escenarios anterior, 30 y 50 Km/h.

5.2.2.- Parámetros a Medir

En este apartado se especificaran los parámetros a medir durante las pruebas en entornos vehiculares. Para ello se escogerán aquellos parámetros que se consideren más adecuados para comprender y acotar el funcionamiento de la tecnología Wi-Fi Direct. A continuación se especifican los parámetros escogidos así como la nomenclatura que se utilizará durante la discusión de los resultados obtenidos.

- $\mathbf{T_{BR\text{-}RX}}$ = Tiempo desde que se inicia la búsqueda hasta que se recibe.
- $\mathbf{T_{DC\text{-}RX}}$ = Tiempo desde la detección del peer hasta la conexión en el receptor.
- $\mathbf{T_{DC\text{-}TX}}$ = Tiempo desde la detección del peer hasta la conexión en el transmisor.
- $\mathbf{T_{CR\text{-}RX}}$ = Tiempo desde que los peers están conectados hasta que se recibe.
- $\mathbf{T_{SR}}$ = Tiempo desde que se el servidor acepta la conexión a través del socket hasta que se recibe.
- $\mathbf{B_{CU}}$ = Bytes de carga útil recibidos.
- $\mathbf{V_{Tx\text{-}Rx}}$ =Velocidad transmisión-recepción.
- $\mathbf{D_D}$ = Distancia de desconexión.

6.- ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS

En este Capítulo se analizaran los resultados obtenidos durante la realización de las pruebas ejecutadas en el apartado anterior. En primer lugar se comentarán los resultados obtenidos durante las pruebas iniciales y, posteriormente, se hará lo propio con los resultados obtenidos en los diferentes escenarios vehiculares.

6.1.- Pruebas Iniciales

La primera prueba llevada a cabo ha sido la conexión y desconexión entre dispositivos para comprobar y tratar de esclarecer que dispositivo actúa como GO. Durante esta prueba cabe resaltar que en algunos puntos los resultados obtenidos no son siempre los mismos para todos los dispositivos, encontrando un comportamiento diferente en los dispositivos Samsung con respecto al resto y especialmente, en el smartphone Samsung Galaxy S2 el cual cuenta con la API 16 (Android 4.1, Jelly Bean) que es la más baja entre los dispositivos probados.

En primer lugar, se ha optado por realizar las pruebas sin especificar ningún GO Intent. De esta manera, se comprueba qué dispositivos actúan como GO en caso de que no se indiquen preferencias. En la mayor parte de las pruebas realizabas este rol se alternaba pero en algunos casos puntuales el rol de GO lo tenía siempre un dispositivo concreto. Esta situación será explicada a continuación.

La siguiente prueba realizada se ha hecho especificando el GO Intent el cual influye en la elección del Group Owner. Cabe mencionar que el GO Intent solo se puede especificar en el dispositivo que realiza la conexión, siendo el que usa la función connect (WifiP2pConfig). Al establecer un valor de 0 se está indicando que ese equipo tiene la menor prioridad para ser el GO y, por tanto, se comprueba que el otro dispositivo actúa como GO. Por el contrario sí se establece un valor de 15 para ese dispositivo, éste siempre actúa en el rol de GO. Se ha observado, analizando los logs de Android Studio que el otro dispositivo usa un valor por defecto de GO Intent (Por ejemplo en la Tablet Samunng Galaxy S2 y en el smartphone Sony Xperia Z3 compact, es de 6 y, en el LG G6 es 5). Es por eso que cuando no se especificaba ningún valor se llevaba a cabo la negociación teniendo en cuenta el valor por defecto, coincidiendo para algunos dispositivos, lo cuales alternaban este rol y siendo superior para otros, los cuales funcionaban siempre como GO.

```
I/wpa_supplicant: P2P: Request to start group negotiation - peer=be:6e:64:dd:d6:8c  GO Intent=6
```

Figura 6.1.- LOG GO Intent

Por tanto se ha comprobado que, dependiendo de si el valor GO Intent especificado en el dispositivo que realiza el connect (WifiP2pConfig) es menor o mayor, este dispositivo

actúa como GO o no. Por ejemplo entre el Sony y la tablet, los cuales tienen el GO Intent por defecto de 6, si se especificaba un GO Intent para uno de ellos por encima de 6 este siempre iba a ser el GO y si por el contrario se especificaba por debajo siempre lo sería el otro dispositivo. En caso de coincidir este valor ambos dispositivos se alternaban el rol de GO. Para el caso del LG G6 sucedía de igual manera pero en este caso el otro dispositivo tenía que establecer un valor menor o igual a 5 para que el LG G6 llegase a actuar como GO.

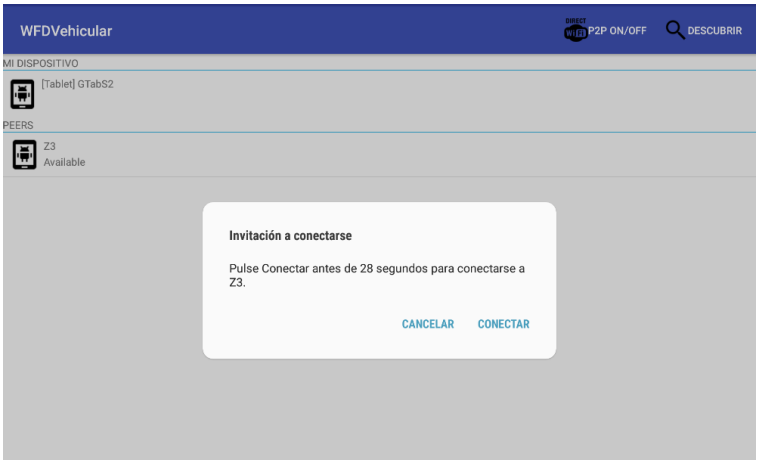

Figura 6.2.- Solicitud de conexión

Para llevar a cabo la conexión entre dos dispositivos, como se ha comentado anteriormente en el *Capítulo 4*, uno de ellos es el que realiza la petición de conexión mediante la función connect (WifiP2pConfig) y el otro dispositivo simplemente acepta la conexión. La primera vez que un dispositivo pide la conexión a otro, en el dispositivo en que se recibe la petición, se muestra un cuadro de diálogo consultando al usuario si desea aceptar dicha conexión *(Figura 6.2)*. Una vez aceptada la solicitud, el grupo es memorizado formándose un grupo persistente y, en sucesivas conexiones entre esos dispositivos, no se mostrará este cuadro de diálogo consultando al usuario manteniéndose los roles anteriormente asignados. Los grupos memorizados pueden verse, según la versión de Android, en: Ajustes→Wi-Fi→ Wi-Fi Avanzado →Wi-FI Direct

Figura 6.3 Grupos Memorizados

Cabe destacar que esto es así aunque posteriormente se cambie el GO Intent. Por ejemplo se probó a establecer un GO intent 15 en el dispositivo Sony Xperia Z3 y este era el GO. Posteriormente, sin eliminar el grupo se cambió a GO Intent a 0, por lo que el otro

dispositivo debería ser el GO, pero al tener el grupo memorizado el Z3 seguía siendo el GO. También es importante resaltar que el grupo persistente es soportado por todos los dispositivos probados anteriormente a excepción de los Samsung, los cuales no memorizan el grupo. De esta forma por ejemplo, en las pruebas entre la Tablet S2 y el Sony Z3 o entre la Tablet S y el Sony Z3, se llegaba a guardar el grupo en el Z3, pero solo cuando este actuaba como GO. De esta forma cuando el Z3 era el GO solo se pedía la primera vez aceptar la conexión y en sucesivas ocasiones se conectaban automáticamente. Sin embargo si el GO era el dispositivo Samsung el grupo nunca era guardado.

Otra prueba que se ha realizado es la conexión entre varios dispositivos, para ello simplemente se ha establecido que todos los dispositivos que tengan en su lista a la Tablet Samsung Galaxy S2 sean los encargados de realizar las solicitud de conexión estableciendo como GO Intent 0 (hay que olvidar los grupos persistentes en caso de existir, ya que dará igual el GO Intent si el grupo está memorizado), es decir haciendo que la Tablet pase a ser el GO y se encargue de la gestión del grupo.

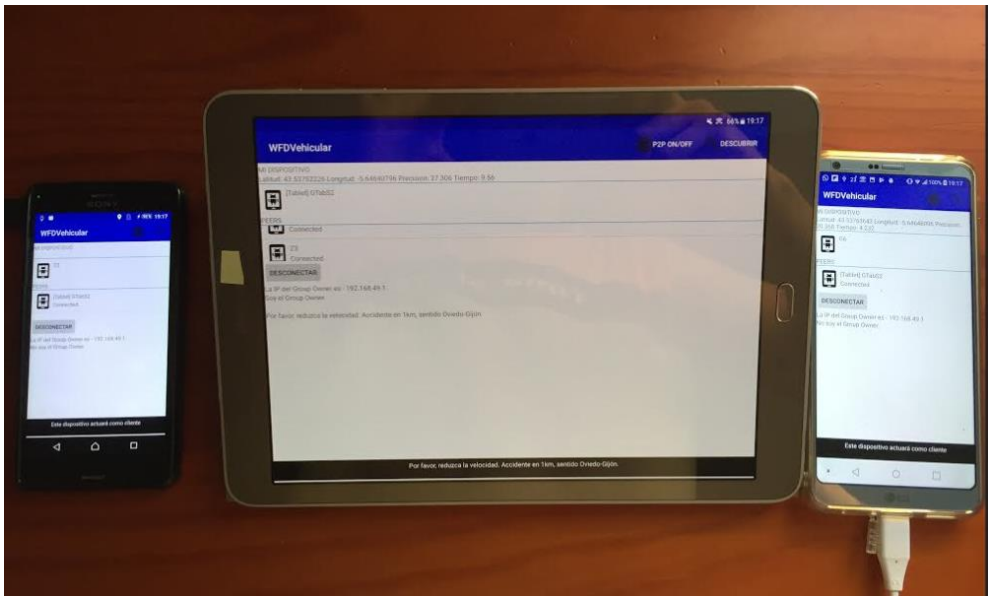

Figura 6.4.-Conexión entre varios dispositivos

En cuanto al direccionamiento IP el GO siempre adquiere la dirección IP 192.168.49.1 y asigna a los demás mediante DHCP una dirección aleatoria del rango 192.168.49.X. Algunas de las IPs obtenidas fueron 192.168.49.112, 192.168.49.200, 192.168.49.201, 192.168.49.212, 192.168.49.245, repitiéndose la misma IP en caso de formar un grupo persistente.

Se han obtenido también algunos de los datos de la interfaz utilizada por Wi-Fi Direct. Entre ellos destacar el valor de la MTU que es de 1500 bytes.

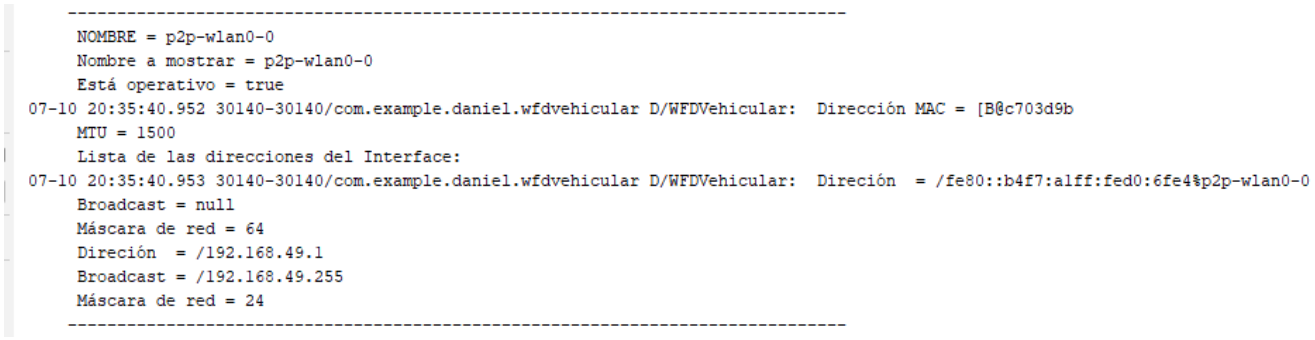

Figura 6.5.- Interfaz Wi-Fi Direct

Para realizar el envío de 100MB mediante TCP, se puede enviar todo de golpe mediante un único array de bytes (*prueba = tcp*), o enviarlo mediante un bucle "for" (*prueba = tcpBucle*). Realmente estos paquetes serán segmentados, teniendo en cuenta la MTU en capas inferiores. Con el fin de comprobar si realmente existe diferencia de enviar la información mediante un único array de bytes o bien mediante un bucle, se han realizado una serie de pruebas con el envío TCP de 100 MB, donde:

- **T_{100MB}**: es el tiempo que se tarda en crear la carga útil a enviar de 100MB.
- **Ttx y Trx**: son los tiempos de transmisión y recepción respectivamente.
- **Vtx y Vrx**: son las velocidades de transmisión y recepción.

T_{100MB} (seg.)	Ttx (seg.)	Vtx (Mbps)	Trx (seg.)	Vrx (Mbps)
0.264	3.663	218.400	3.675	217.687
0.268	3.587	223.027	3.589	222.903
0.274	3.643	219.599	3.646	219.419

Tabla 6-1.- Envío de 100 MB mediante un único array de bytes

T_{100MB} (seg.)	Ttx (seg.)	Vtx (Mbps)	Trx (seg.)	Vrx (Mbps)
0.256	3.528	226.757	3.544	225.734
0.265	3.634	220.143	3.628	220.507
0.267	3.595	222.531	3.614	221.361

Tabla 6-2.- Envío de 100 MB mediante un bucle de 65 400 bytes

T_{100MB} (seg.)	Ttx (seg.)	Vtx (Mbps)	Trx (seg.)	Vrx (Mbps)
0.259	3.617	221.178	3.543	225.797
0.264	3.558	224.845	3.615	221.300
0.26	3.646	219.419	3.56	224.719

Tabla 6-3.- Envío de 100 MB mediante un bucle de 1400 bytes

T_{100MB} (seg.)	Ttx (seg.)	Vtx (Mbps)	Trx (seg.)	Vrx (Mbps)
0.268	4.559	175.477	4.555	175.631
0.261	4.524	176.835	4.517	177.108
0.263	4.489	178.213	4.483	178.451

Tabla 6-4.- Envío de 100 MB mediante un bucle de 500 bytes

A la vista de los resultados anteriores se puede concluir que las velocidades así como los tiempos de envío y recepción no varían mucho de unas pruebas a otras a excepción de la que hace el envío de información en un bucle de 500 bytes, aumentando en 1 segundo los tiempos de envío y recepción y disminuyendo la velocidad en torno a 50Mbps. Esto se debe a que a pesar de que en la capa de aplicación no se lleve a cabo la segmentación, como los bytes se envían de 500 en 500, tienen un menor tamaño que la MTU, por lo tanto o bien los paquetes enviado son menores o bien tienen que esperar más tiempo para rellenar los 1500 bytes (menos cabeceras) que componen la MTU (no tiene por qué ser el tamaño de la MTU puede ser algo menor) y realizar los envíos.

De las tablas anteriores también se pueden obtener los datos de velocidades ideales, ya que las pruebas se han realizado con los dispositivos separados apenas medio metro. Estas velocidades se acercan a los valores máximos teóricos comentados previamente en el *Capítulo 2* (250Mbps). En estos casos los tiempos de que se tarda en transmitir y recibir son inferiores a los 4 segundos a excepción del caso del bucle de 500 bytes. Hay que tener en cuenta que no se estaba conectado a ninguna otra red Wi-Fi a la vez, ya que cuando el dispositivo se encuentra conectado a una red Wi-Fi las velocidades de transmisión y recepción alcanzan como máximo valores cercanos a los 65 MB. Además puede observarse en la comparativa presente en la *Figura 6.6* como la velocidad cae al introducir elementos intermedios como paredes u obstáculos de gran tamaño.

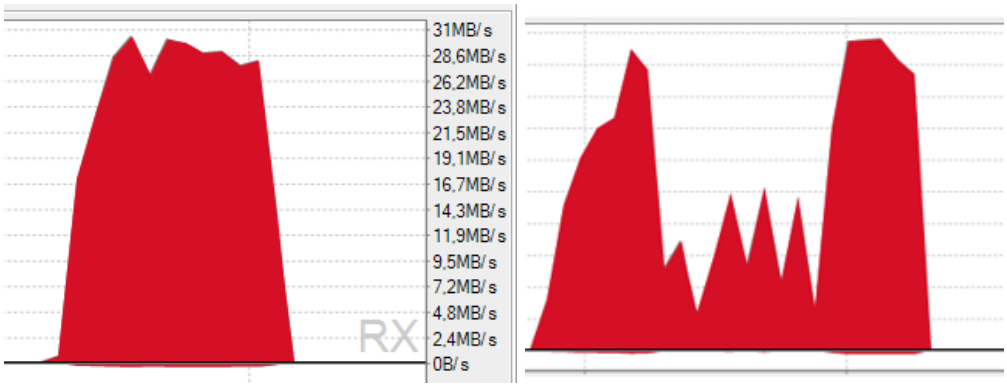

Figura 6.6 Comparativa de la velocidad al introducir obstáculos intermedios

Continuando en el escenario anterior, en situaciones ideales en las cuales los teléfonos están separados en torno a medio metro y sin obstáculos, los tiempos desde que se produce la búsqueda hasta que se detecta ($T_{BD,}$) son de entre 2 y 4 segundos y desde que se produce la búsqueda hasta que se conecta (T_{BC}), son en torno a medio segundo superiores a los anteriores, como se muestra en la *Tabla 6.5*.

T_{BD} (seg.)	**T_{BC} (seg.)**	**T_{BR} (seg.)**
2.025	2.706	7.713
2.974	3.487	7.816
3.882	4.496	9.144

Tabla 6-5.- Tiempos de búsqueda

En esta situación el tiempo total que se tarda en enviar 100 MB a velocidades por encima de 200 Mbps desde que comienza el proceso de búsqueda hasta que se recibe la información (T_{BR}) es de entre 8 y 9 segundos aproximadamente. Para los casos en los que las velocidades fueran de entre 40 y 60 Mbps estos tiempos serían entre 9 y 17 segundos mayores por el incremento producido en el envío de la información.

Además del envío de 100 MB, también se prueba también a transmitir la frase “Por favor, reduzca la velocidad. Accidente en 1km sentido Oviedo-Gijón”, (prueba = tcpShort) para probar una posible funcionalidad en la cual se alertara a un vehículo de un accidente. El tiempo desde que se inicia la búsqueda hasta que se recibe el mensaje es prácticamente el mismo tiempo que invierten los peers en conectarse, tardando aproximadamente 0.2 segundos más desde que los peers están conectados hasta que se recibe la información. Es decir, desde que se inicia la búsqueda hasta que se puede leer el mensaje se tarda entre 3 y 5 segundos.

6.2.- Pruebas en Entornos Vehiculares

En este apartado se estudiarán los resultados obtenidos durante las pruebas realizadas en entornos vehiculares. Para ello se analizaran los parámetros especificados en el apartado *5.2.2 Parámetros a medir*, para cada uno de los diferentes escenarios planteados (*5.2.1 Escenarios*). En primer lugar se comentaran los resultados de las pruebas realizadas brevemente y de manera individual para, posteriormente, realizar una discusión de los resultados de la cual se puedan extraer una serie de conclusiones.

6.2.1.- Escenario 1: Vehículos en Reposo

En este escenario se ha establecido el envío de información entre dos vehículos que se encuentran en reposo. Los envíos se han realizado variando la distancia de separación entre mismos (5, 35 y 80 metros).

6.2.1.1.- Vehículos separados 5 metros

A continuación se muestran los resultados obtenidos durante esta prueba:

T_{BR-RX} (seg.)	T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
7.946	2.000	0.623	4.104	3.628	100	220.751
9.559	2.629	0.697	5.166	4.616	100	173.611
8.152	0.669	0.412	4.561	4.054	100	197.970

Tabla 6-6.- Vehículos en reposo separados 5 metros

Analizando los datos cabe destacar que los tiempos desde que se inicia la búsqueda hasta que se reciben los 100 MB van desde los 8 hasta los 10 segundos, invirtiendo sobre 4 segundos en el periodo de conexión. El tiempo transcurrido desde que se detecta el peer hasta que se conecta al mismo se sitúa por debajo del segundo a excepción del dispositivo transmisor. Esto puede deberse a que el transmisor haya detectado al receptor antes (porque haya empezado a buscar antes o por la capacidad de escaneo del propio dispositivo) y por tanto haya tenido que esperar a que el otro dispositivo lo detecte. Por otro lado los tiempos desde que los peers están conectados hasta que se recibe se sitúan entre 4 y 5 segundos, reduciéndose en medio segundo aproximadamente el tiempo desde que se acepta el socket y se comienza la transmisión, hasta que se recibe.

En cuanto a la velocidad se puede observar como esta varía entre 174 y 221 Mbps aproximadamente, valores cercanos a la velocidad máxima obtenida en pruebas anteriores entre estos dispositivos. En la *Figura 6.7* se puede ver una gráfica con las variaciones de la velocidad (en MegaBytes) en función del tiempo en uno de los envíos realizados. En este

caso, se trata de una transmisión de pocos segundos, pero parece que la velocidad es bastante estable, especialmente después del primer segundo, situándose en valores cercanos a los 200Mbps.

Figura 6.7.- Velocidad/Tiempo envío a 5 metros

6.2.1.2.- Vehículos separados 35 metros

A continuación se muestran los resultados obtenidos durante esta prueba:

T_{BR-RX} (seg.)	T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
27.468	0.662	0.616	24.942	24.337	100	32.872
14.934	0.355	0.493	12.847	12.366	100	64.694
22.034	0.594	0.488	19.925	19.247	100	41.565

Tabla 6-7.- Vehículos en reposo separados 35 metros

En este caso los tiempos desde que se inicia la búsqueda hasta que se reciben los 100 MB, van desde los 15 hasta los 27 segundos, invirtiendo en torno a 2 segundos en el periodo de conexión. Los tiempos desde que detecta el dispositivo hasta que se conecta, se sitúan por debajo del segundo tanto para el transmisor como para el receptor. En cuanto a los tiempos desde que los peers establecen la conexión hasta que se recibe la información, varían bastante situándose entre los 13 y los 25 segundos. Resulta lógico que a mayor distancia pueda variar en mayor medida los tiempos, ya que es más probable que influyan otros factores como interferencias o la aparición de obstáculos intermedios. Por último, las velocidades en este caso van relacionadas con los tiempos, oscilando entre los 33 y los 65 MB.

Por último, en la *Figura 6.8* se puede apreciar como varía la velocidad en función del tiempo, manteniéndose más estable en la zona central y produciéndose más picos en los extremos, especialmente en la parte en la que se inicia la transmisión.

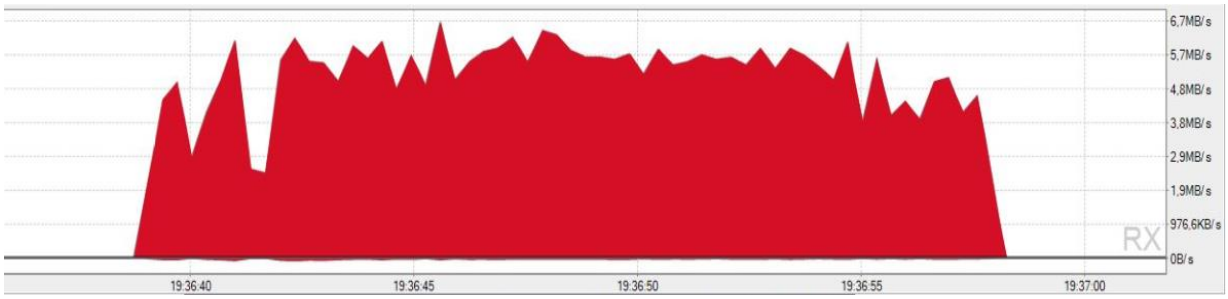

Figura 6.8.- Velocidad/Tiempo envío a 35 metros

6.2.1.3.- Vehículos separados 80 metros

A continuación se muestran los resultados obtenidos durante esta prueba:

T_{BR-RX} (seg.)	T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
27.453	0.461	0.516	25.013	24.491	100	32.665
27.21	1.045	0.679	24.152	23.479	100	34.073
34.755	1.134	0.726	31.887	31.288	100	25.569

Tabla 6-8.- Vehículos en reposo separados 80 metros

Para este último caso los tiempos desde que se inicia la búsqueda hasta que se reciben los 100 MB, van desde los 27 hasta los 35 segundos, invirtiendo aproximadamente 3 segundos en el periodo de conexión. Los valores desde la detección del peer hasta su conexión, se sitúa por debajo del segundo para el receptor y cercanos al segundo para el transmisor. Respecto a los tiempos desde que los peers están conectados hasta que se recibe la información estos varían desde los 24 hasta los 32 segundos, obteniéndose las velocidades más altas para los menores tiempos de envío (de 25 a 34 Mbps). A continuación se muestra una gráfica con la variación de la velocidad respecto del tiempo en la cual, se aprecia un pequeño incremento de en torno a 1MB en la parte final del envío con respecto de la velocidad media de la parte inicial.

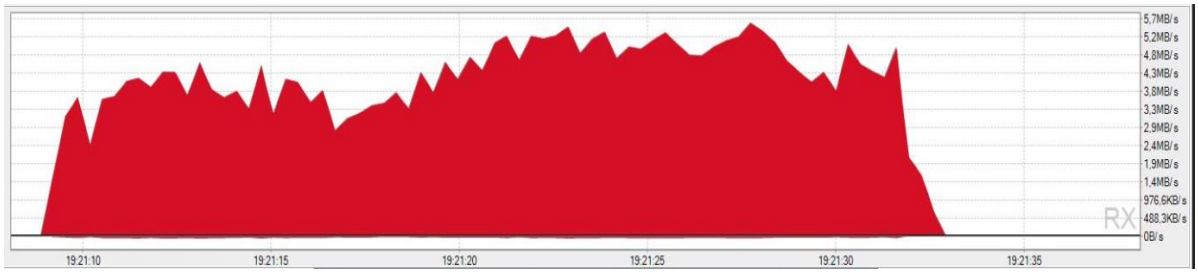

Figura 6.9.- Velocidad/Tiempo envío a 80 metros

6.2.1.4.- Discusión de los resultados en el Escenario 1

Una vez analizadas las distintas pruebas llevadas a cabo en el Escenario 1 se pueden obtener las siguientes conclusiones:

- Los tiempos totales, es decir, desde que se comienza la búsqueda hasta que se recibe, son de entre 8 y 10 segundos para la distancia de 5 metros, de entre 15 y 27 segundos para la distancia de 35 metros y de entre 27 y 35 segundos para la distancia de 80 metros. Estos valores varían con la distancia, sin tener gran afección en la parte en la que se establece la conexión, e influyendo en gran medida en la que se realiza el envío.
- Los tiempos invertidos en el descubrimiento y conexión, es decir desde que los dispositivos inician la búsqueda hasta que se conectan oscilan entre los 2 y los 4 segundos, sin que la distancia sea un factor determinante para este parámetro. Incluso los peores tiempos se obtuvieron a menor distancia (puede que se haya seleccionado buscar antes en un dispositivo que en otro). Sí habría que tener en cuenta que a mayor distancia podrían haber más obstáculos intermedios o interferencias, y en ese caso, este parámetro si se vería afectado.
- Los tiempos desde la detección de los peers hasta su conexión se sitúan en valores inferiores o cercanos al segundo para el receptor, mientras que el transmisor, en algunos casos se sitúa en valores superiores al segundo. Esto, puede ser debido a las propias características del dispositivo, o a que la prueba se haya iniciado antes en el transmisor.
- Los tiempos desde que los peers están conectados hasta que se recibe la información aumentan considerablemente con la distancia obteniendo valores entre 3 y 6 veces superiores para la distancia de 35 metros y entre 6 y 8 veces superiores para la distancia de 80 metros. Estos valores obtenidos fueron de entre 4 y 5 segundos para la distancia de 5 metros, de entre 13 y 25 segundos para la de 35 metros y de entre 24 y 32 segundos para la de 80 metros
- Este aumento de los tiempos es consecuencia de la caída de la velocidad con respecto a la distancia, disminuyendo la misma con respecto a la velocidad máxima obtenida a 5 metros, hasta 5 veces para la distancia de 35 metros y, 9 veces para una distancia de 80 metros. Las velocidades obtenidas fueron de entre 174 y 221 Mbps para la distancia de 5 metros, de entre 33 y 65 Mbps para la de 35 metros y de entre 25 y 34 Mbps para la de 80 metros.
- Mediante las gráficas de la velocidad con respecto al tiempo puede observarse como en términos generales la velocidad es constante, produciéndose alguna variación únicamente en la parte inicial de la transmisión.

6.2.2.- Escenario 2

En este escenario se han realizado pruebas entre un vehículo en movimiento y otro en situación de reposo. Para comprobar la influencia de la velocidad en el envío de la información se han hecho pruebas a dos velocidades, 30 Km/h y 50 Km/h. Además también se ha medido la distancia a la que se desconectan los peers una vez que han transmitido información. Esta distancia es un valor aproximado y se ha calculado mediante la *ecuación 6.1*, donde T_{OP} es el tiempo invertido desde que se solicita la información hasta que se recibe:

$$D_D = (Distancia\ entre\ coordenadas - (T_{OP} * Velocidad)) \pm Precisión \qquad (6.1)$$

Es decir, la distancia estimada de desconexión se ha calculado como resultado de obtener la distancia entre coordenadas en el momento de la desconexión y, restarle el tiempo invertido en obtener este dato por la velocidad a la que el vehículo se mueve. Además a este valor final se le sitúa en un rango en función de la precisión obtenida en la consulta de la posición.

6.2.2.1.- Un vehículo en movimiento a 30 Km/h

A continuación se muestran los valores obtenidos durante esta prueba:

T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
2.431	0.716	23.517	22.977	31.326	10.907
1.599	0.696	25.029	23.549	25.486	8.741
4.727	0.73	38.021	37.483	36.084	7.701

Tabla 6-9.- Un vehículo en movimiento a 30 Km/h

D_D (metros)	Distancia inicial (metros)	Precisión (metros)	Tiempo (seg.)	Velocidad (m/s)
154.587 (±13.653)	175.27	13.653	2.483	8,33
104.98 (± 15.17)	136.89	15.17	3.831	8,33
140.082 (±4.288)	164.63	4.288	2.947	8,33

Tabla 6-10.- Distancias obtenidas al desplazarse un vehículo a 30 Km/h

A partir de los resultados obtenidos en la *Tabla 6.9*, se puede observar como los tiempos desde la detección del peer hasta su conexión se sitúan por debajo del segundo para el receptor mientras que, para el transmisor, llegan a alcanzar casi los 5 segundos. Respecto a los tiempos desde que los peers están conectados hasta que se produce la desconexión (ya que no se llegan a enviar los 100MB), estos varían desde los 23 hasta los 38 segundos. Los bytes de caga útil recibidos se sitúan en valores de entre 25 y 36 MB. Estos valores, dependen directamente de la distancia a la que consigan desconectarse los peers (analizada posteriormente), así como de la velocidad a la que transmitan durante ese intervalo de tiempo que, en este caso, fue desde los 7 hasta las 11Mbps.

En la *Figura 6.10* se puede observar cómo cambia la velocidad de recepción con el tiempo. En este caso si se aprecian grandes variaciones de la misma que se deben fundamentalmente a la distancia con el receptor. Cuanto más cerca se encuentra el transmisor del receptor mejor es la conexión entra ambos dispositivos y por tanto mayor es la velocidad y la carga útil transmitida, como se aprecia en la parte central de la *Figura 6.10*.

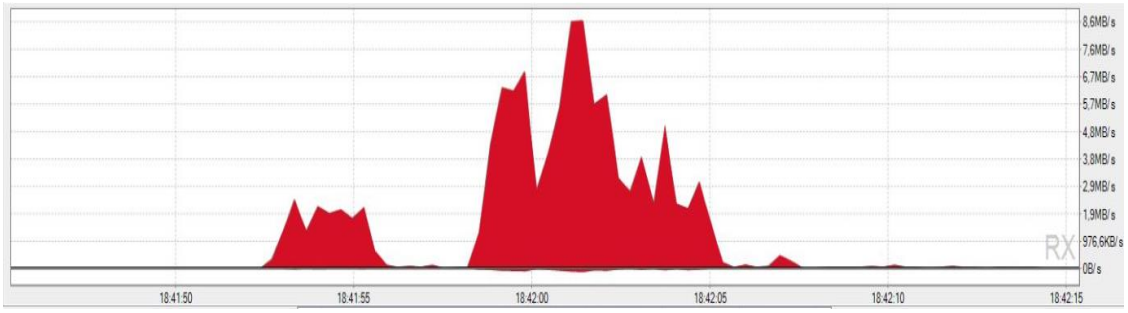

Figura 6.10.- Velocidad/Tiempo de un vehículo a 30Km/h

Con respecto a la distancia, esta posición, como se ha especificado anteriormente, ha sido corregida (*Tabla 6.10*) teniendo en cuenta el tiempo que tarda en recibirse la ubicación, así como la velocidad de desplazamiento, obteniendo un dato más restrictivo y que proporciona una aproximación más precisa de la posición de desconexión. En este caso se puede observar como los valores aproximados a los que se produce la desconexión se sitúan entre los 105 y los 155 metros. En la *Figura 6.11* se muestra la ubicación de los dos vehículos cuando se recoge la posición a causa de la desconexión.

Figura 6.11.- Distancia de desconexión de un vehículo desplazándose a 30Km/h

6.2.2.2.- Un vehículo en movimiento a 50 Km/h

A continuación se muestran los valores obtenidos durante la realización de esta prueba:

T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
1.574	0.645	17.804	17.224	17.763	8.251
1.915	0.538	17.717	17.164	21.969	10.240
1.407	1.79	28.719	26.98	11.421	3.387

Tabla 6-11.- Un vehículo en movimiento a 50 Km/h

D_D (metros)	Distancia inicial (metros)	Precisión (metros)	Tiempo (seg.)	Velocidad (m/s)
142.429 (±12.136)	178.39	12.136	2.589	13.89
146.012 (±15.17)	188.94	15.17m	3.09	13.89
167.644 (±12.136)	204.3m	12.136	2.639	13.89

Tabla 6-12.- Distancias obtenidas al desplazarse un vehículo a 50 Km/h

En la *Tabla 6.11* se puede observar como los valores desde que se produce la detección del peer hasta su conexión se sitúan por debajo del segundo para el receptor mientras que para el transmisor se encuentran entre uno y dos segundos. En cuanto los tiempos desde que los dispositivos están conectados hasta que se produce la desconexión, estos varían entre los 17 y los 29 segundos, variando también carga útil recibida entre los 11 y los 22 MB. Esta última, como se ha comentado anteriormente, además de depender del tiempo que los peers estuvieron conectados, depende también de la velocidad de recepción, que en este caso se ha tomado valores entre los 3 y 10 Mbps.

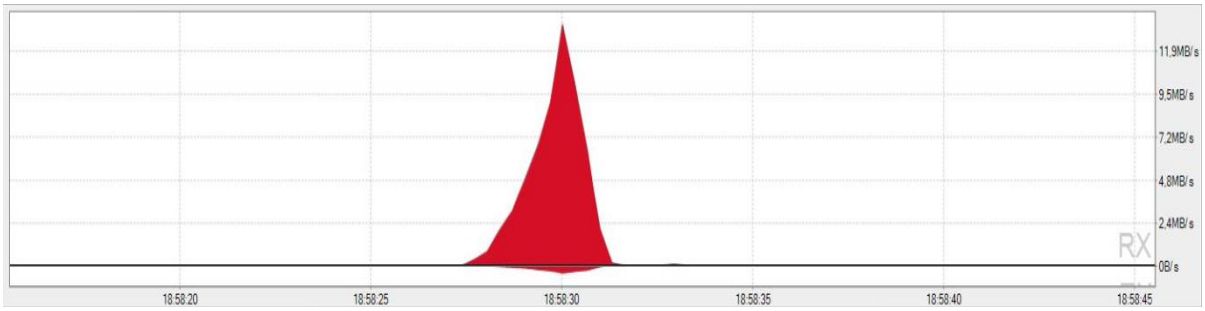

Figura 6.12.- Velocidad/Tiempo de un vehículo a 50km/h

En la *Figura 6.12* se aprecia como la gráfica de la velocidad con respecto al tiempo tiene forma de pico. Esto como se ha comentado anteriormente se debe a que la mayor tasa de transferencia de bits por segundo se produce cuando los dispositivos están más cerca. A los lados de ese pico se producen pequeños envíos ya ha velocidad muy baja y casi inapreciables en la gráfica cuando los dispositivos se encuentran alejados.

En cuanto a la distancia, los valores aproximados a los que se produce la desconexión para este caso se sitúan entre los 142 y los 167 metros. En la *Figura 6.13* se muestra la ubicación de los dos vehículos cuando se recoge la posición a causa de la desconexión.

Figura 6.13.- Distancia de desconexión de un vehículo a 50Km/h

6.2.2.3.- Obtención de la distancia de desconexión

Para obtener un valor más preciso de la distancia a la que los peers se desconectan se ha realizado otra prueba adicional. En esta prueba, dos vehículos se situaron inicialmente en paralelo y, posteriormente, uno de ellos comenzó la marcha manteniéndola a una velocidad aproximada de 15 km/h. Esta prueba se ha realizado en dos ubicaciones diferentes (*Figuras 6.14 y 6.15*, ubicadas en la parte delantera y trasera de la escuela de Marina). A continuación se muestran los valores de distancia obtenidos durante esta prueba:

D_D (metros)	**Distancia inicial (metros)**	**Precisión (metros)**	**Tiempo (seg.)**	**Velocidad (m/s)**
207.033 (±19.72)	215.29	19.72	1.91	4.17
172.158 (±12.136)	182.12	12.136	2.389	4.17
222.376 (±7.585)	227.89	7.585	1.109	4.17
196.209 (±10.619)	207.33	10.619	2.667	4.17

Tabla 6-13.- Pruebas de localización

En la *Tabla 6.13* se puede observar como las distancias de desconexión van desde los 222 hasta los 172 metros. Esto proporciona una idea aproximada de las distancias soportadas por esta tecnología en situaciones de poco tráfico en valores cercanos a los 200 metros En las *Figuras 6.14 y 6.15* se muestran algunos puntos de desconexión durante las pruebas realizadas en las dos ubicaciones elegidas.

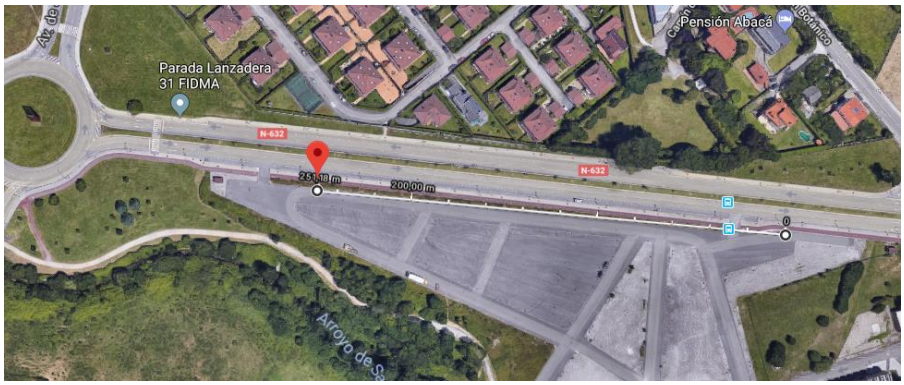

Figura 6.14.- Distancia de desconexión en la parte trasera de Marina

Figura 6.15 Distancia de desconexión en la parte delantera de Marina

6.2.2.4.- Discusión de los resultados en el Escenario 2

Una vez analizados las distintas pruebas llevadas a cabo en el Escenario 2 se pueden obtener las siguientes conclusiones:

- Al igual que en el Escenario 1 los tiempos entre la detección de los peers y la conexión son menores en el dispositivo receptor que en el transmisor, situándose en términos generales en valores menores al segundo y cercanos a los dos segundos respectivamente. Esto como se ha comentado anteriormente puede deberse a las prestaciones del dispositivo o al momento en el que realizó la búsqueda el mismo.
- Los tiempos que los peers estuvieron conectados fueron de entre 23 y 38 segundos cuando un vehículo se desplazaba a 30 Km/h y de entre 17 y 29 cuando lo hacía a 50 Km/h. Como se observa, estos tiempos se ven reducidos con la velocidad, la mayor diferencia recogida en los resultados anteriores, se

da en uno de los valores de la prueba de 30 Km/h que llega a ser del doble de tiempo que a 50Km/h. Por otro lado también existe un valor en la prueba de 50 km/h que llega a ser igual o incuso un poco superior al tiempo de conexión de la prueba realizada a menor velocidad. Es lógico, quitando esta última excepción (puede deberse a factores como la aparición de otros vehículos, interferencias o el funcionamiento del propio dispositivo) que los tiempos de conexión se reduzcan ya que, a mayor velocidad menor es el tiempo invertido en recorrer una distancia y por tanto menor será el tiempo que se permanece conectado.

- La carga útil que fue capaz de ser transmitida durante estas pruebas también disminuyó con la velocidad puesto que el tiempo de conexión es menor. Estos valores recibido fueron, para el caso de la prueba realizada a 30 Km/h, de entre 25 y 36 MB mientras que para la prueba de 50Km/h se recibe entre 11 y 21 MB. Es decir en ciertos casos se llegó a transmitir el doble de carga útil. Se puede concluir entonces que los bytes de caga útil recibidos dependen de la distancia a la que consigan desconectarse los peers, de la velocidad del propio vehículo, así como de la velocidad a la que transmitan los bytes durante ese intervalo de tiempo.
- La velocidad de recepción, sin embargo no parece que se reduzca tanto cuando el vehículo se desplaza a 30 o a 50 Km/h, de hecho la máxima obtenida para las dos pruebas se diferencia en 0.700 Mbps aproximadamente, tomando valores de entre 7 y 11 Mbps para el primer caso y de entre 3 y 10 Mbps para el segundo. Incluso en el caso comentado anteriormente en el que se recibió el doble de carga útil a 30 Km/h que a 50 Km/h la velocidad de recepción fue mayor a 50 Km/h que a 30 Km/h. Por lo tanto no parece que este incremento de la velocidad de desplazamiento de los vehículos afecte en gran medida a la velocidad de recepción.
- En cuanto a las distancias de desconexión en las pruebas realizadas a 30Km/h y 50Km/h estos valores oscilan situándose entre los 150 y los 200 metros. Sin embargo, en las pruebas realizadas exclusivamente para la obtención de la distancia (en las cuales los vehículos estaban en paralelo y uno de ellos comenzaba la marcha aproximadamente a 15 Km/h), estos valores parecen situarse entre los 170 y los 200 metros. Hay que tener en cuenta que a mayor velocidad mayor también es la probabilidad de recibir un dato erróneo.
- También es importante resaltar que la posición depende del momento en el cual se produzca la desconexión y esta a su vez puede verse influida por otros factores como interferencias, vegetación, cruces con otros vehículos e incluso el tiempo en detectar la desconexión. En definitiva, según los resultados obtenidos se puede establecer un rango de distancias a las que los peers se desconectan de entre los 150 y 200 metros en situaciones en las que no hay un tráfico excesivo y las velocidades van desde los 15 hasta los 50 Km/h.

6.2.3.- Escenario 3

En este escenario se han realizado pruebas entre dos vehículos en movimiento. Para ello se han organizado situaciones en las cuales dos vehículos se cruzan a diferentes velocidades así como, situaciones en las que los dos vehículos se encuentran desplazándose en el mismo sentido de circulación.

6.2.3.1.- Dos vehículos en sentidos opuestos a 30 Km/h

A continuación se muestran los resultados obtenidos durante esta prueba:

T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
4.094	0.711	17.854	17.305	24.598	11.371
0.399	0.748	22.475	21.887	26.715	9.765
1.545	0.592	20.214	19.525	28.679	11.751

Tabla 6-14.- Cruce de dos vehículos a 30 Km/h

Como puede observarse en la *Tabla 6.14* los valores desde la detección del peer hasta su conexión se sitúa por debajo del segundo para el receptor mientras que para el transmisor alcanza valores de hasta 4 segundos. Con respecto a los tiempos desde que se produce la conexión hasta la desconexión, estos varían desde los 17 hasta los 22 segundos. Los bytes de caga útil que se han conseguido transmitir durante el cruce de los vehículos ha sido de entre 24 y 29 MB mientras que las velocidades de transmisión alcanzaron valores de entre los 10 hasta los 12 Mbps.

6.2.3.2.- Dos vehículos en sentidos opuestos a 50 Km/h

A continuación se muestran los valores obtenidos durante esta prueba:

T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
3.218	0.664	12.71	12.005	12.842	8.558
4.658	0.582	18.848	17.993	21.914	9.744
3.606	0.732	15.756	15.194	5.858	3.085

Tabla 6-15.- Cruce de dos vehículos a 50 Km/h

En este caso al igual que en el caso anterior, los valores desde la detección del peer hasta su conexión se sitúan por debajo del segundo para el receptor mientras que para el transmisor alcanzan valores de hasta casi 5 segundos. Con respecto a los tiempos desde que se produce la conexión hasta la desconexión, estos varían desde los 12 hasta los 19 segundos. Los bytes de caga útil que se han conseguido transmitir durante el cruce de los vehículos ha sido de entre 6 y 22 MB, dónde más que el tiempo de conexión, ha influido la velocidad de transmisión situándose en 3 Mbps para el peor caso y, sobre 10 Mbps para el mejor.

6.2.3.3.- Dos vehículos en la misma dirección a 30 Km/h

Durante esta prueba se ha rodado a 30 Km/h manteniendo una distancia relativamente cercana al vehículo delantero de en torno a unos 5-15 metros. A continuación se muestran los resultados obtenidos:

T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
0.409	0.747	31.898	31.383	100	25.492
0.438	0.479	40.692	40.173	100	19.914
0.440	0.635	38.59	38.105	100	20.995

Tabla 6-16.- Vehículos en el mismo sentido a 30 Km/h.

Los tiempos desde la detección hasta la conexión están, tanto para el receptor como para el transmisor, por debajo del segundo (los vehículos están cerca al iniciar la búsqueda). En cuanto a los tiempos desde que los peers están conectados hasta que se reciben los 100 MB, estos se sitúan entre los 30 y los 40 segundos con unas tasas de transmisión de entre 20 y 25 Mbps.

6.2.3.4.- Dos vehículos en la misma dirección conducción variable

A continuación se muestran los resultados obtenidos durante esta prueba:

T_{DC-TX} (seg.)	T_{DC-RX} (seg.)	T_{CR-RX} (seg.)	T_{SR} (seg.)	B_{CU} (MB)	V_{Tx-Rx} (Mbps)
1.626	0.630	66.292	65.724	100	12.172
0.622	0.608	79.724	78.664	100	10.169
0.533	0.778	24.718	24.117	100	33.172

Tabla 6-17.- Vehículos en el mismo sentido (velocidad y distancia variable)

En este caso se ha rodado a una velocidad variable de aproximadamente entre 10 y 50 Km/h, modificando también la distancia entre vehículos y dejando, en general, mayores distancias entre los mismos principalmente para las dos primeras pruebas que se muestran en la *Tabla 6.17*. También en algún momento puntual se han acercado bastante los vehículos entre sí.

Para estas pruebas, los tiempos desde la detección hasta la conexión se sitúan para el receptor por debajo del segundo. Para el transmisor se da la misma situación a excepción de uno de los casos en que el tiempo se encuentra cercano al segundo y medio. En cuanto al tiempo desde que los peers se han conectado hasta que se reciben los 100MB, es bastante variable. Para las pruebas en las que más se han modificado las distancias y velocidades (Filas 1 y 2 de la *Tabla 6.17*) se sitúa entre los 65 y 78 segundos con unas tasas de transmisión de entre 10 y 12 Mbps. Por otro lado, el tiempo de conexión de la tercera prueba, en la cual los vehículos acabaron acercándose bastante, es de 25 segundos con una tasa de transmisión de 33MB, la cual es bastante superior a las otras dos pruebas. Esto es lógico puesto que las distancias entre vehículos eran inferiores y por tanto existía mayor estabilidad en el enlace.

6.2.3.5.- Discusión de los resultados en el Escenario 3

Una vez analizados las distintas pruebas llevadas a cabo en el Escenario 3 se pueden obtener las siguientes conclusiones:

- Los tiempos entre la detección de los peers y la conexión en general son menores en el dispositivo receptor que en el transmisor, aunque existen algunos casos en los cuales son bastante parejos. Esto como se ha comentado anteriormente parece deberse a las prestaciones del dispositivo unido al momento de comienzo de la búsqueda.
- Los tiempos que permanecen conectados los peers en las pruebas realizadas entre vehículos que se cruzan, son menores cuando éstos se desplazan a mayor velocidad, reduciéndose la conexión un máximo de 10 segundos y, llegando incluso, en un caso concreto, a ser superiores en el dispositivo que se desplaza a mayor velocidad. Esta situación puede deberse a diferentes factores como la aparición de otros vehículos, interferencias o el funcionamiento del propio dispositivo. Los tiempos que los peers estuvieron conectados fueron de entre 12 y 19 segundos cuando los vehículos se cruzaban a 50 Km/h y de entre 18 y 22 cuando lo hacían a 30 Km/h.
- Los tiempos desde que se conectan los peers hasta que se reciben los 100 MB en la prueba realizada con vehículos que circulan en el mismo sentido dependen de diferentes factores como tráfico, distancias y velocidades. Para el caso en el que la velocidad son de 30 Km/h y la distancia de entre 10 y 15 metros, los tiempos de conexión se encuentran entre 30 y 40 segundos. Si por el contrario se aumentan las distancias y se varía la velocidad incrementándola, estos tiempos también se ven incrementados. Si la distancia con el vehículo

situado delante se ve reducida también lo harán estos tiempos de conexión en los que se recibe la carga útil.

- La velocidad de recepción no se reduce en gran medida al aumentar la velocidad de los vehículos que se cruzan siendo de entre 10 y 12 Mbps para la prueba en la que se cruzan a 30 Km/h y de entre 3 a 10 Mbps para los que lo hacen a 50 Km/h. Esta diferencia no es demasiado llamativa en la mayor parte de los casos pero si en caso puntual en el cual la velocidad de transmisión a 30 Km/h es prácticamente 4 veces superior a la velocidad a 50Km/h.
- La velocidad recepción en el caso en el que los vehículos se mueven en el mismo sentido depende en gran medida de la distancia de separación de los mismos así como de otros factores como el tráfico. A partir de los resultados obtenidos esta velocidad llega en algunas ocasiones a triplicarse. Las velocidades para el caso en que ambos vehículos se desplazaban a 30 Km/h a una distancia de entre 5 y 15 metros va desde los 20 a los 25 Mbps, mientras que en el caso en que se variaba la velocidad y las distancias estos valores se mueven entre los 10 y los 33 Mbps.
- Al igual que en el Escenario 2 la carga útil recibida en los casos en los que los vehículos se cruzan se ve afectada por la velocidad de los vehículos ya que esto influye en el tiempo de conexión de los mismos. Los valores de carga recibida fueron de entre 25 y 29 MB cuando los vehículos circulaban a 30 Km/h y de entre 6 y 22 MB cuando lo hacían a 50 Km/h. Por ejemplo para uno de los casos, la carga útil recibida a 50 Km/h es la mitad que la recibida a 30 Km/h y, para otro caso, llega incluso a ser entre 5 y 6 veces menor.

7.- PLANIFICACIÓN

En la siguiente tabla se muestran las tareas que se deben realizar para llevar a cabo el proyecto así como el plazo de tiempo estimado de cada una:

ID Tarea	Nombre de la Tarea	Duración	Fecha de Inicio	Fecha de finalización
1	Estudio de la tecnología Wi-Fi Direct	3 semanas	12/02/2018	04/03/2018
2	Desarrollo de la aplicación	12 semanas	05/03/2018	27/05/2018
3	Estudio del IDE	1 semana	05/03/2018	11/03/2018
4	Estudio de las APIs de la tecnología Wi-Fi Direct	3 semanas	12/03/2018	01/04/2018
5	Planteamiento de pruebas	2 semanas	21/05/2018	03/06/2018
6	Realización de pruebas iniciales	1 semana	04/06/2018	10/06/2018
7	Análisis de los resultados de las pruebas iniciales	1 semana	11/06/2018	17/06/2018
8	Realización de pruebas en entornos vehiculares	1 semana	18/06/2018	24/06/2018
9	Análisis de los resultados en entornos vehiculares	1 semana	25/06/2018	01/07/2018

Tabla 7-1.- Planificación

A continuación se muestra el diagrama de Gant con las tareas a realizar:

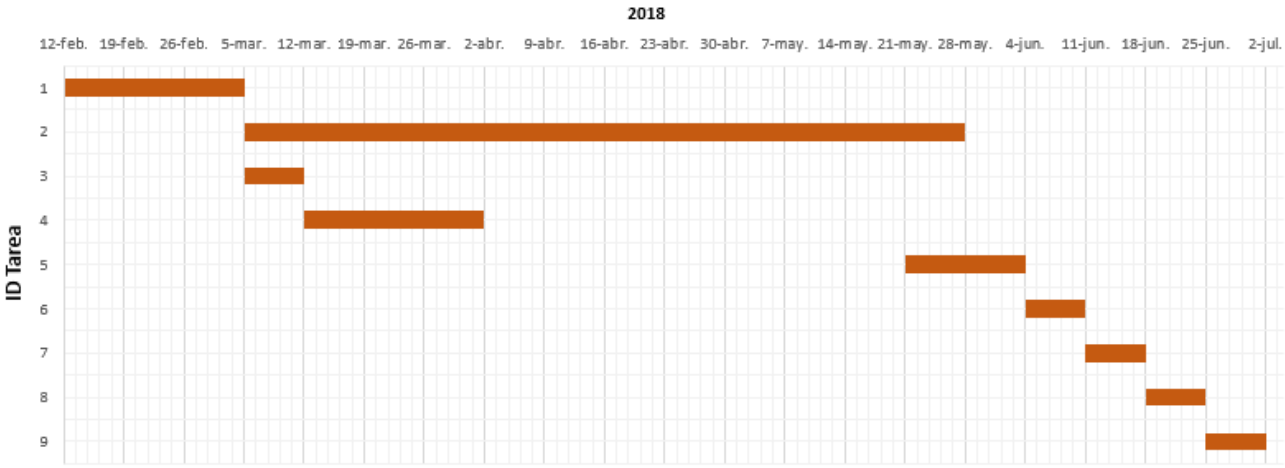

Figura 7.1.- Diagrama de Gant

8.- PRESUPUESTO

Este proyecto se basa en el análisis de la tecnología Wi-Fi Direct en escenarios de movilidad para lo cual, se ha tenido que desarrollar una aplicación en Android. Para poder llevar a cabo este análisis se han realizado las pruebas en las instalaciones de la Escuela Politécnica de Ingeniería de Gijón.

Para la elaboración del proyecto se han utilizado dos dispositivos inteligentes basados en el sistema operativo Android, así como un ordenador y dos vehículos sobre los que realizar las pruebas los cuales se incluirán en el presupuesto como si de un alquiler de una semana se tratara. En la siguiente tabla se muestran los precios de cada material utilizado en la elaboración del proyecto:

Material Utilizado	Unidades	Precio(€)/ Unidad	Subtotal (€)
Dispositivo Inteligentes Android	2	300	600
Ordenador Lenovo Z50 Intel Core i5	1	699	699
Alquiler Vehículo	2	300	600
	Total Material (€) : 1899		

Tabla 8-1.- Presupuesto del material

A continuación se muestra el coste de la mano de obra utilizada en este proyecto:

Mano De Obra	Unidades (horas)	Precio(€)/ Unidad	Subtotal (€)
Ingeniero en Telecomunicaciones	800	15	12000
	Total Mano De Obra (€): 12000		

Tabla 8-2.- Presupuesto de la mano de Obra

Por último se muestra una tabla resumen en la cual se suman todos los gastos obteniendo, finalmente, el coste del proyecto:

RESUMEN	
PRESUPUESTOS	**TOTAL(€)**
Presupuesto material	1899
Presupuesto salario	12000
Presupuesto de ejecución material	13899
Gastos generales (13%)	1806.87
Beneficio industrial (6%)	833.94
Coste total sin IVA	**16539.81**

Tabla 8-3.- Presupuesto total

El **Presupuesto Total** del proyecto (IVA no incluido) asciende a la cantidad de **dieciséis mil quinientos treinta y nueve euros con ochenta y un céntimos (16539.81 €).**

9.- CONCLUSIONES Y TRABAJOS FUTUROS

A través de este proyecto se ha realizado una primera aproximación a la tecnología Wi-Fi Direct es escenarios de movilidad. Para ello se han llevado a cabo una serie de pruebas, inicialmente en parado y posteriormente en movimiento, que han permitido un mayor conocimiento de las características de esta tecnología en entornos vehiculares. Para poder realizar las pruebas se ha decidido implementar una aplicación en Android por la facilidad de acceso a un dispositivo con estas características (tanto por razones económicas, como por el número de dispositivos presentes en el mercado). Las pruebas realizadas han tenido lugar en condiciones de poco tráfico y sin demasiados obstáculos intermedios por lo que los resultados obtenidos se encontrarán cerca de los valores ideales que se podrían obtener con esta tecnología en escenarios vehiculares.

Destacar que para la realización de las pruebas en entornos vehiculares, se distinguieron tres escenarios principales en los cuales los vehículos se intercambiaron información. Un primer escenario en el que ambos vehículos están parados y simplemente se realizan envíos a diferentes distancias, un segundo escenario en el que un vehículo se encuentra parado y otro se encuentra en movimiento y, un tercer escenario en el que ambos vehículos se encuentran en movimiento. En este último escenario, se ha hecho que los vehículos intercambien información tanto cuando se cruzan, como cuando se mueven en el mismo sentido de circulación. Para todos los escenarios se ha realizado un envío de 100 MB. Se ha escogido esta cantidad porque se considera un valor suficiente del que poder obtener conclusiones acerca de la carga útil recibida, así como porque permite obtener valores estabilizados de la velocidad.

Antes de analizar las conclusiones en entornos vehiculares (estudio detallado en el apartado *6.2 Pruebas en Entornos Vehiculares*) resulta interesante revisar el apartado *6.1 Pruebas Iniciales*, en el cual se hace referencia a algunas pruebas realizadas en entorno de laboratorio que, aunque no son el objetivo fundamental del proyecto, ayudan a comprender mejor el funcionamiento de la tecnología Wi-Fi Direct.

- En el primer escenario, los tiempos totales, es decir los tiempos desde que se comienza la búsqueda hasta que se reciben los 100 MB, son de entre 8 y 10 segundos para la distancia de 5 metros, de entre 15 y 27 segundos para la de 35 metros y de entre 27 y 35 segundos para la de 80 metros. Estos valores como se puede observar varían con la distancia, sin tener apenas afección en la parte en la que se establece la conexión (entre 2 y 4 segundos en el establecimiento de la conexión), e influyendo en gran medida en la parte que se realiza el envío a causa de la disminución de la velocidad en el mismo. Esta disminución de la velocidad llega a ser hasta 5 veces menor para la distancia de 35 metros y hasta 9 veces menos para una distancia de 80 metros.

- Comparando los tiempos anteriores, con los tiempos empleados para enviar los 100 MB cuando los vehículos se desplazan en el mismo sentido de circulación, existen ciertas equivalencias. Para el caso en el que la velocidad de los vehículos es de 30 Km/h y la distancia es de entre 10 y 15 metros, esta situación equivaldría aproximadamente al envío en parado a una distancia de 80 metros, situándose los tiempos entre 30 y 40 segundos. Si se reduce la distancia con el vehículo situado delante variando las velocidades, estos tiempos también se verán reducidos, en este caso para la prueba realizada, el tiempo fue de 24 segundos, lo que equivaldría a uno de los peores envíos a 35 metros en parado o, casi a uno de los mejores a 80 metros. Si por el contrario se aumentan las distancias y se varía la velocidad incrementándola, estos tiempos también se ven incrementados, tomando valores de entre 65 y 79 segundos para las pruebas realizadas, siendo bastante superiores a todos los envíos anteriores.
- Continuando con la comparativa anterior, la velocidad recepción para la prueba en la que los vehículos se mueven en el mismo sentido, depende en gran medida de la distancia de separación de los mismos, así como de otros factores como el tráfico. Las velocidades para el caso en que ambos vehículos se desplazaban a 30 Km/h a una distancia de entre 5 y 15 metros va desde los 20 a los 25 Mbps (parecidas a las velocidades alcanzadas en los intercambios en parado a 80 metros), mientras que en el caso en que se variaba la velocidad y las distancias, estos valores se mueven entre los 10 y los 33 Mbps siendo mayores cuando los dispositivos se encontraban más cercanos (en este caso las velocidades se parecen a uno de los peores envíos a 35 metros en parado o a uno de los mejores a 80 metros).
- Para los casos en lo que un vehículo se encontraba parado y otro estaba en movimiento o, cuando ambos se cruzaban, no se llegaron a transmitir los 100 MB. En estos casos, los tiempos que los peers estuvieron conectados intercambiando información fueron, para el Escenario 2, de entre 23 y 38 segundos cuando un vehículo se desplazaba a 30 Km/h y, de entre 17 y 29, cuando lo hacía a 50 Km/h. Para el Escenario 3 por su parte los tiempos fueron de entre 18 y 22 segundos cuando los vehículos se cruzaban a 30 Km/h y de entre 12 y 19 lo hacían a 50 Km/h .

  De los resultados anteriores se puede concluir que los tiempos de conexión disminuyen cuando los vehículos se mueven a mayor velocidad. Esto es lógico ya que se tarda menos tiempo en recorrer la distancia a la que se desconectan los dispositivos. Por la misma razón se aprecia una disminución en el tiempo de conexión si se desplaza un único vehículo o si se desplazan los dos.
- A raíz de las pruebas realizadas para medir las distancias de desconexión de los dispositivos, se puede concluir que éstos se desconectan en un rango de entre 150 y 200 metros.

- Con respecto a la carga útil recibida, si los tiempos de conexión son menores también disminuirá la carga útil recibida, ya que se cuenta con menos tiempo para recibir la información. No obstante la carga útil no dependerá solo del tiempo que los vehículos estén conectados, sino que también lo hará de la velocidad a la que se reciban los bytes durante ese intervalo de tiempo. Además estos parámetros pueden variar también debido a factores externos como la aparición de otros vehículos, interferencias o el funcionamiento del propio dispositivo.

  Las carga útil recibida para el caso en que se desplaza un solo vehículo, es algo superior que para el caso en que ambos se desplazan cruzándose. Estos valores a 30 Km/h son de entre 25 y 36 MB para el primer caso, y de entre 24 y 29 MB para el segundo caso. Lo mismo sucede cuando los vehículos van a 50 Km/h siendo de entre 11 y 22 MB para el primer caso y de entre 6 y 21 MB para el segundo. Como se ha comentado anteriormente esto es lógico, ya que si se desplazan los dos vehículos en direcciones opuestas se tarda menos en recorrer la distancia de desconexión que si lo hace uno solo.

- Las velocidades de recepción, en general, no se ven afectadas en gran medida por ese incremento de unos 20 Km/h de desplazarse a 30 Km/h o a 50 Km/h, pero si afecta a la carga útil recibida (tarda menos en recorrer la distancia de desconexión). Por ejemplo, en uno de los casos del segundo escenario, el tiempo que se mantuvieron conectados los peers fue el doble a 30 Km/h que a 50 Km/h, sucediendo lo mismo con la cantidad de carga útil recibida. Sin embargo, en este caso, la velocidad de recepción fue superior incluso cuando el vehículo iba a mayor velocidad.

  Las velocidades de recepción tanto cuando se desplaza un solo vehículo como cuando los vehículos se cruzan son bastante parejas tomando valores que oscilan entre los 8 y 12 Mbps cuando los vehículos van a 30 Km/h y de entre 3 y 10 Mbps cuando van a 50 Km/h.

- De este modo, ha quedado demostrado que la tecnología Wi-Fi Direct se puede utilizar perfectamente en entornos vehiculares, no limitándose solo al envío de avisos, si no que se puede abrir un amplio abanico de posibles aplicaciones ya que, en las situaciones más complejas analizadas en este proyecto se llegaron a intercambiar entre 24 y 36 MB a 30 Km/h y entre 6 y 22 MB a 50 Km/h.

  Por otro lado, también resulta interesante las grandes cantidades de información que se pueden compartir en poco tiempo cuando los vehículos están parados. Incluso encontrándose éstos a la mayor distancia estudiada en este proyecto (80 metros), para la cual, en el peor caso, se tardó sobre medio minuto en transmitir 100 MB.

  Al igual que en el caso anterior, se ha comprobado la gran cantidad de información que se puede intercambiar entre vehículos que se desplacen en el

mismo sentido de circulación. Por ejemplo, en el peor de los casos estudiados se tardó 1 minuto y 20 segundos en transmitir 100 MB. Es por ello, que una aplicación interesante de esta tecnología podría ser aquella en que ambos vehículos se desplacen hacia el mismo destino o, simplemente, que coincidan en una pequeña parte del trayecto, permitiendo intercambiar entre sí grandes cantidades de información.

En cuanto a los caminos a seguir para continuar con el estudio de esta tecnología, quedan abiertas diferentes vías de trabajo. Por un lado, se podría continuar con esta investigación mediante el estudio de la tecnología Wi-Fi Direct en un entorno desfavorable en el que hubiese bastante tráfico así como numerosos obstáculos intermedios. En este caso las pruebas se han realizado entre dos vehículos, por lo que también se podría comprobar el comportamiento de esta tecnología cuando se conecta un mayor número de vehículos. Además, en este estudio las velocidades máximas a las que se han recogido datos son de 50 Km/h por lo que también se podría analizar el rendimiento de la tecnología Wi-Fi Direct a velocidades superiores como, por ejemplo las que tienen lugar en carreteras secundarias (velocidad máxima sobre 100 Km/h) o autovías/autopistas (velocidad máxima 120 Km/h).

10.- REFERENCIAS

[1] Información de la tecnología Wi-Fi Direct: http://www.wi-fi.org

[2] Daniel Camps-Mur, Andres Garcia-Saavedra y Pablo Serrano, "Device to Device communications with WiFi Direct: overview and experimentation", 2012.

[3] Información de la tecnología Bluetooth: https://www.bluetooth.com/

[4] Información sobre el dispositivo Chomecast: https://www.google.es/intl/es_es /chromecast/

[5] Información sobre las Monitorless: https://news.samsung.com/global/samsung-c-lab-to-exhibit-new-vr-projects-at-mobile-world-congress-2017

[6] Obteneción de las aplicaciones a probar: https://play.google.com

[7] Nurain Izzati Shuhaimi, Heriansyah y Tutun Juhana, "Comparative Performance Evaluation of DSRC and Wi-Fi Direct in Vanet", 2015.

[8] Sangsoo Jeong, Youngmi Baek y Sang H. Son, "A Hybrid V2X System for Safety-Critical Applications in VANET", 2016.

[9] Chaitra Satish, "Inter-vehicular communication for collision avoidance using Wi-Fi Direct", 2014.

[10] Sungwon Lee, Dongkyun Kim, "An Energy Efficient Vehicle to Pedestrian Communication Method for Safety Applications", 2015

[11] K.Kalidhas, Jerin Ninan, Jubin Mathew Chacko, Sooraj Saseendran, Ullas Chandran, "Implementation of Li-Fi Technology for Home Automation and Vehicle Communication", 2016.

[12] Giuseppe Araniti, Massimo Condoluci, Pasquale Scopelliti, Antonella Molinaro, y Antonio Iera, "Multicasting over Emerging 5G Networks: Challenges and Perspectives", 2017.

[13] Javier Gozalvez (IEEE vehicular technology magazine), "5G Worldwide Developments", 2017.

[14] Información sobre Android y sus APIs: https://developer.android.com

[15] Información sobre Java: www.aprenderaprogramar.es

[16] Descargar Java: www.oracle.com

[17] Descargar Cygwin: www.cygwin.com