TREBALL FINAL DE GRAU

Títol: Avaluació de dispositius wearables i disseny d'una aplicació per a la comunicació amb un smartphone

Autor: Òscar Azuara Domínguez

Directora: Elena López Aguilera

Co-Director: Eduard Garcia Villegas

Data: 28 de Maig de 2015

Títol: Avaluació de dispositius wearables i disseny d'una aplicació per a la comunicació amb un smartphone

Autor: Òscar Azuara Domínguez

Directora: Elena López Aguilera

Co-Director: Eduard Garcia Villegas

Data: 28 de Maig de 2015

Resum

Aquest Treball Final de Grau té com a objectiu estudiar les aplicacions per a dispositius wearables.

Conté una guia de desenvolupament dirigida a proporcionar la informació suficient per programar una aplicació per a un dispositiu wearable utilitzant el sistema operatiu Android Wear.

El projecte comença explicant els motius de l'elecció del dispositiu wearable, comparant diferents dispositius del mercat, així com l'elecció del sistema operatiu Android Wear respecte altres sistemes operatius utilitzats en wearables.

Així doncs, es mostra una aplicació que mesura la freqüència cardíaca utilitzant el sensor òptic que porta el dispositiu wearable i que transmet aquesta informació des del dispositiu wearable cap al smartphone, on es desenvolupa una aplicació Android que dóna suport a la del wearable.

Aquesta aplicació, s'ha de dotar de comunicació Bluetooth per la connexió entre tots dos dispositius, així com de mètodes per obtenir la informació del sensor de freqüència cardíaca.

En la línia d'explicar com s'ha desenvolupat aquesta aplicació, s'inclou informació detallada sobre les llibreries que es poden utilitzar per programar amb l'entorn escollit, així com els mètodes utilitzats. S'inclou també, una guia de com preparar l'entorn de programació pas a pas per que l'usuari pugui reproduir una aplicació com la del projecte pel seu compte.

Finalment, es realitza un estudi de consum de bateria del dispositiu wearable. Gràcies a l'avaluació dels diferents escenaris, es podrà comprovar aquest consum relacionant els diferents escenaris amb la utilització del sensor de freqüència cardíaca i amb les diferents maneres d'enviament de les mostres pel canal Bluetooth.

Title: Wearable devices evaluation and smartphone-wearable communication application

Author: Òscar Azuara Domínguez

Director: Elena López Aguilera

Co-Director: Eduard Garcia Villegas

Date: May 28th of 2015

Overview

The objective of this Treball Final de Grau is studying wearable devices and developing applications for them.

This document contains a few concepts about Android applications and basic guidelines to program an application for Android Wear O.S. that could be followed by the reader for programming his own application.

First of all, there is an explanation about the reasons for the device choice, thus comparing commercially available devices and performing an overview of the different O.S. used nowadays.

The document presents an application that measures the heart rate through an optical sensor embedded in the wearable device. This application sends the data obtained from the sensor to another application developed in a smartphone. Bluetooth is employed for this purpose.

With the objective of understanding how the application works, the document explains the methods used in the application. A guideline for application developers is also included so that a user could be able to develop a new application step by step using the Android Wear O.S.

Finally we present some battery level tests on the wearable. We study the impact of the developed application on the battery level, thus evaluating the influence of the heart rate sensor and the use of the Bluetooth link between the applications, along with different scenarios for sending heart rate samples.

ÍNDEX

INTRODUCCIÓ

Avui dia, i cada cop més, la gent es preocupa pel seu estat físic i per la seva salut, i requereixen de més mètodes per mesurar els seus senyals vitals. Volem, doncs, un sistema que ens permeti portar el control del nostre estat de salut i que a la vegada sigui accessible a tants usuaris com sigui possible.

Per altra banda, i gràcies a que la tecnologia avança a passos agegantats, cada cop comptem amb més dispositius que mesuren aquestes constants. Dispositius que tenen integrats sensors, que permeten mesurar, per exemple, la freqüència cardíaca.

¿Podem comptar, llavors, amb aquests dispositius per portar un control de la nostra salut?

La resposta es sí, i aquest projecte es farà al voltant d'un d'aquests dispositius, concretament amb un rellotge intel·ligent, que ens permetrà tenir accés a un sensor de freqüència cardíaca, i, a la vegada, tenir connectivitat amb altres dispositius intel·ligents.

Com l'objectiu d'aquest projecte és que pugui ser utilitzat pel màxim nombre de persones, i que sigui un coneixement el més útil possible, s'ha buscat un rellotge intel·ligent que sigui compatible amb la majoria de dispositius smartphones del mercat. En l'actualitat els dispositius mòbils més venuts són els que porten un sistema operatiu Android, com es pot veure en la Figura I.1, i sembla que continuarà sent així [1].

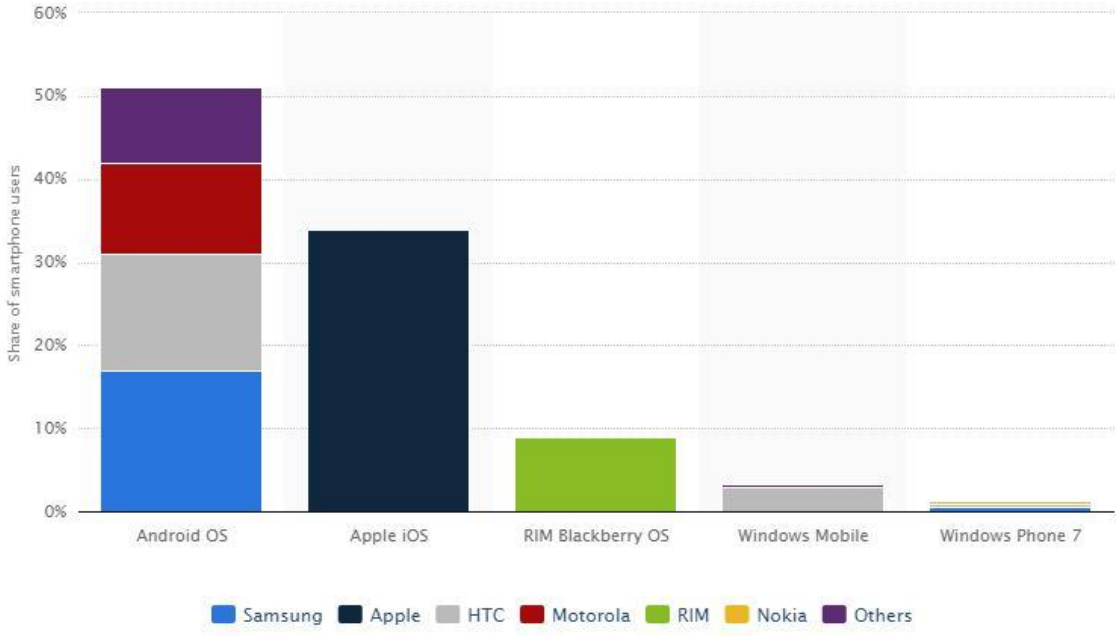

Fig.I.1 Comparativa de la utilització de sistemes operatius per a dispositius mòbils [2].

A part de ser el sistema operatiu més utilitzat, es un sistema lliure on tothom pot tenir accés a les eines de programació i a les llibreries de manera gratuïta. A més, es compta amb una gran comunitat d'usuaris que investiga amb aquest sistema operatiu per treure-li el màxim rendiment.

Hem de pensar, llavors, com hauria de ser una aplicació per aquest tipus de dispositius intel·ligents que fos intuïtiva i fàcil d'emprar. Qualsevol tipus de públic ha de poder utilitzar-la i així tenir el control de la seva freqüència cardíaca. En aquest projecte explicarem tot el necessari per fer una aplicació Android per a aquests rellotges intel·ligents (a partir d'ara els anomenarem dispositius 'wearable').

També centrarem la darrera part d'aquest projecte en fer un estudi del consum de bateria que tenen aquests dispositius wearable en diferents situacions que tindran a veure amb el sensor de ritme cardíac i amb l'enviament d'aquestes dades a un altre dispositiu smartphone amb sistema operatiu Android utilitzant la connexió Bluetooth.

Quins són, doncs, els objectius d'aquest projecte?

L'objectiu principal d'aquest projecte serà adquirir tot el coneixement necessari sobre el desenvolupament d'aplicacions per poder portar a terme amb èxit una aplicació Android, així com conèixer els mètodes per obtenir informació del dispositiu on s'executa l'aplicació i tot el referent a enviament de dades entre dispositius a través de la connexió Bluetooth.

Veurem com crear una aplicació des de zero comporta un cert grau de dedicació i uns coneixements previs necessaris, però que està a l'abast de qualsevol persona amb ganes d'aprendre i gràcies a la participació de tots els desenvolupadors de la gran comunitat que és Internet.

Per marcar un objectiu final a aquest aprenentatge sobre programació d'aplicacions Android, es desenvoluparà una aplicació per a Android Wear que posi en pràctica tots els coneixements adquirits en aquesta memòria per comprovar com totes les funcions poden funcionar de manera simultània en l'aplicació.

I, finalment, es faran unes proves de bateria en el dispositiu wearable monitoritzant la informació d'aquest sensor. Això, ens permetrà saber si un wearable compta amb la bateria suficient per ser utilitzat durant el dia a dia amb el sensor obtenint dades contínuament, o bé, si amb unes mesures puntuals, podem obtenir la mateixa informació amb menys consum.

Aquesta memòria doncs està classificada en diferents apartats, que seguiran un ordre lògic pensat per a ser llegit de principi a final començant pel capítol 1 i, quan en el document hi faci referència llegint altres capítols del mateix projecte, així com de l'Annex, i accedint als mitjans i portals web inclosos en les referències que estan situats al final del document.

CAPÍTOL 1. ENTORN

Per portar a terme aquesta aplicació s'han de tenir controlats uns elements imprescindibles. Parlem tant d'elements hardware, per exemple, un rellotge que contingui un sistema operatiu Android per poder descarregar i executar l'aplicació, com d'elements software i dels entorns de programació necessaris per a poder programar-la.

Comencem parlant del hardware que recomanem utilitzar per portar a la pràctica el desenvolupament del projecte, i després continuem explicant quin entorn s'utilitza i perquè.

1.1. Hardware

Són imprescindibles tres elements físics o suports on executar l'aplicació. Per una banda, es necessita un dispositiu wearable (és on s'executa l'aplicació principal), un smartphone que doni suport al dispositiu wearable, i un ordinador PC per poder programar i desenvolupar l'aplicació.

1.1.1. Android Wearable

Per poder provar l'aplicació és necessari un dispositiu wearable, un element d'ús quotidià que portem sempre amb nosaltres, com una polsera o un rellotge, i que tingui, com a element integrat, un sensor de polsos cardíacs. Per fer una aplicació que sigui amigable per l'usuari, és molt important que aquest dispositiu tingui una petita pantalla tàctil per tal que l'usuari pugui interactuar amb l'aplicació.

Com es pot comprovar en la Figura 1.1.1.1, hi ha una àmplia gamma de fabricants de dispositius wearables al mercat amb aquest tipus de sensor integrat, ja que ens pot donar una informació molt útil per a ús mèdic i d'investigació.

Fig.1.1.1.1 Principals fabricants de wearables al mercat

Per fer la nostra aplicació utilitzarem el dispositiu *Samsung Gear Live* ja que, a diferència d'altres dispositius disponibles en el mercat, com el *Galaxy Gear 2*, que funciona amb el sistema operatiu propi Tizen, o l'*iWatch*, que treballa amb sistema operatiu també propi, Watch OS, aquest utilitza un sistema operatiu Android anomenat Android Wear, el que ens permet programar les nostres aplicacions amb llenguatge Java, juntament amb XML, així, partim amb cert avantatge, ja que s'han estudiant durant la carrera.

Un cop clar que treballarem amb un dispositiu amb Android, i mirant el mercat de dispositius wearables amb sistema operatiu Android Wear [3], les característiques són semblants entre tots els dispositius. Si ens fixem en el preu, el *LG G Watch*, el *Asus ZenWatch* i el *Samsung Gear Live* són una mica més barats que la resta, i descartant el *ZenWatch*, que no té sensor de polsos cardíacs, ens podríem quedar amb qualsevol dels altres dos.

Finalment es tria el dispositiu de *Samsung* per la fiabilitat de la marca i per que compta amb una pantalla amb millor resolució, i és una mica més gran que la resta.

Així doncs, les característiques principals del Samsung Gear Live són les següents [12]:

- Pantalla de 1.63 polzades, 320 x 320 píxels, SuperAMOLED
- Processador a 1.2GHz, RAM de 512MB, 4GB de capacitat d'emmagatzematge
- Bateria de 300mAh
- Compatibilitat amb qualsevol dispositiu amb Android 4.3 o superior.
- Google Services disponibles: Google Now, Google Voice, Google Maps & Navigation, Gmail, Hangouts
- Sensor òptic de mesurament de polsos cardíacs, GPS, acceleròmetre, Compàs, Giroscopi i podòmetre.
- Connectivitat Bluetooth 4.0, microUSB

Els punts forts d'aquest dispositiu són; la seva connectivitat mitjançant Bluetooth 4.0, que inclou Bluetooth Low Energy i Smart Bluetooth, la seva bateria i el seu ampli ventall de sensors.

El mesurador òptic de polsos cardíacs, juntament amb l'ampli ventall de llibreries que ens ofereix el seu sistema operatiu Android Wear que, a més, el permet ser compatible amb tots els dispositius Android 4.3 o superior, ens permetrà poder fer un estudi d'enviament de dades d'un sensor a un dispositiu Android mitjançant Bluetooth.

1.1.2. Android Smartphone

Tots aquests dispositius wearables estan preparats per la comunicació amb mòbils que també tinguin instal·lat sistema operatiu Android. Per aquest motiu

existeix una aplicació per a smartphones Android, *Android Wear* [4], que ens facilita aquesta comunicació entre el dispositiu wearable i el smartphone.

L'únic requisit que ha de tenir aquest smartphone per instal·lar l'aplicació és que ha de treballar amb una API d'Android superior a la versió 4.4.2 (API 19).

En aquest cas utilitzarem un *Sony Xperia M* on li hem actualitzat la versió d'Android a la 4.4.2 per no tenir problemes de compatibilitat amb les llibreries de l'aplicació del wearable, així com amb la tecnologia Bluetooth 4.0 que porta incorporada.

1.1.3. Windows PC

En aquest projecte s'utilitzarà un ordinador amb sistema operatiu Windows 8, encara que es pot fer amb qualsevol altre sistema operatiu on pugui funcionar l'Android Studio.

Aquest ordinador ha de tenir la memòria RAM suficient per poder fer funcionar els emuladors de dispositius Android que ens proporciona l'Android Studio amb fluïdesa, sobretot en l'etapa de desenvolupament de l'aplicació (a partir de 2GB és suficient tenint en compte que l'emulador de dispositiu Android consumeix fins a 512MB de RAM).

Així mateix, també és necessari que aquest ordinador tingui disponible un compte d'administrador amb els permisos d'obertura necessaris per poder fer proves de Debug en els dispositius directament, especialment important sobretot en l'etapa de desenvolupament, així com la possibilitat d'accedir a Internet per descarregar-se les llibreries SDK de la xarxa.

1.2. Software

A continuació comentarem tot el relacionat amb els entorns que hem de preparar, abans de començar a desenvolupar la nostra aplicació, tenint en compte que aquesta es desenvoluparà en un sistema operatiu Android.

1.2.1. Sistema Operatiu

¿Perquè fem servir Android? La resposta es fàcil, Android és un sistema operatiu basat en el nucli Unix, que està molt estès arreu del món, aproximadament un 80% dels dispositius mòbils l'utilitzen [5]. Per aquest motiu, i ja que Android és un entorn obert on tothom pot accedir a totes les llibreries i codis necessaris i hi ha una gran comunitat de desenvolupadors, utilitzar un dispositiu wearable amb un entorn Android és la millor opció.
A més, podem comptar amb el sistema operatiu Android Wear, que és una modificació que es va fer del sistema Android 4.3 específicament dissenyat pels dispositius wearable.

Aquest sistema operatiu Android Wear treballa de forma similar a l'Android però inclou funcionalitats addicionals per a dispositius wearables, i elimina algunes funcionalitats d'Android que no són necessàries en els wearables permetent centrar-se més en el desenvolupament de l'aplicació on es poden aplicar conceptes d'Android.

Com veiem en la Figura 1.2.1.1, els diferents fabricants de dispositius wearables han pres diferents estratègies a l'hora de triar el sistema operatiu dels seus dispositius. Però Android Wear ofereix compatibilitat amb tots els dispositius Android 4.3 o superior, el que el permet connectar-se amb la majoria de dispositius mòbils del mercat. Utilitzar Android Wear també li permet utilitzar els serveis de Google [6] així com el seu mercat d'aplicacions.

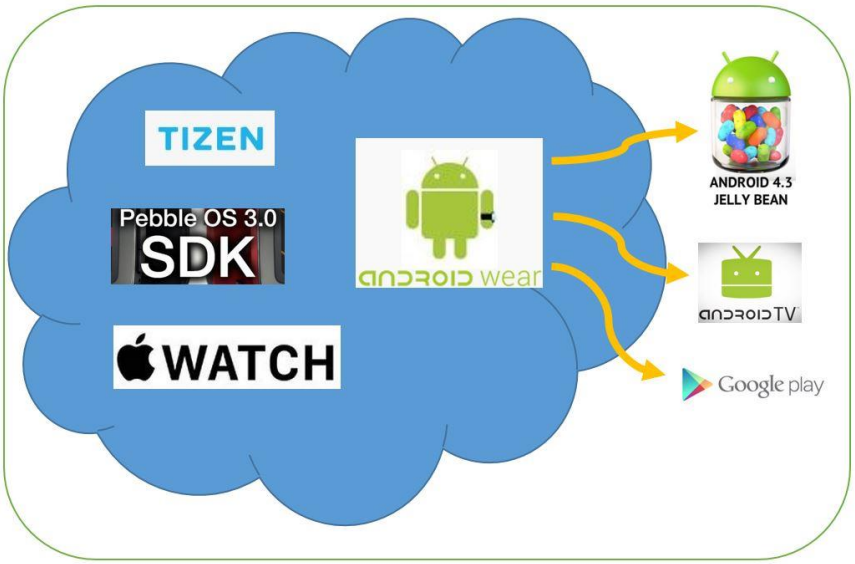

Fig.1.2.1.1 Connectivitat Android respecte altres S.O.

És un encert, per part de Samsung, haver preparat el dispositiu Gear Live per treballar amb Android ja que la plataforma compta amb un gran ventall d'aplicacions per donar suport a la connectivitat amb dispositius wearables i, segurament, sigui més utilitzat ja que, per a molts desenvolupadors d'aplicacions, és més fàcil treballar amb un sistema operatiu més estès i amb més suport com Android que amb un sistema propi com Tizen o Watch OS.

Es molt important tenir l'última versió d'Android per al wearable Android Wear, ja que aquesta utilitza els serveis de Google, per tant ens demanarà que descarreguem les ultimes versions per tal de tenir sempre el dispositiu wearable actualitzat. En cas que s'actualitzi la versió dels serveis de Google o de l'aplicació Android Wear, s'ha de tornar a configurar el wearable, activant el mode Debug, com s'explica a la secció 6 de l'Annex, i instal·lant l'aplicació de nou.

1.2.2. Entorn de programació

Per preparar el desenvolupament de l'aplicació necessitem un entorn de desenvolupament que coneguem, per això la millor opció és utilitzar Android Studio [7].

Android Studio és un entorn de desenvolupament integrat (IDE), molt semblant visualment i per característiques a Eclipse, però que està dissenyat específicament per a Android. En aquest cas ens és molt útil perquè ja que hem treballat amb Eclipse i, per tant, ja tenim coneixement previ.

Pels desenvolupadors d'aplicacions, Android Studio és molt útil, ja que disposa també d'un sistema per crear emuladors d'Android. Aquests emuladors, a part d'emular un dispositiu mòbil Android, també permeten l'emulació de dispositius wearables.

1.2.3. SDK

El SDK, o kit de desenvolupament de software, és un conjunt d'eines de desenvolupament software que ajuden al programador a poder desenvolupar la seva aplicació. Es tracta d'una interfície que permet la programació d'aplicacions creada per poder programar en un llenguatge de programació, en aquest cas Java i XML.

Per portar a terme aquesta aplicació comptarem amb dos dispositius Android (mòbil i wearable) que tenen instal·lats l'API 19 d'Android KitKat versió 4.4.2 en el mòbil i l'API 20 d'Android Lollipop en el wearable.

La API 19 KitKat d'Android ja s'està instal·lant a tots els nous dispositius mòbils amb Android i un dels aspectes mes destacables d'aquesta nova versió és que ha estat feta pensant en els dispositius wearables. Així que inclou un conjunt de llibreries específicament per a sensors que permeten obtenir la informació dels mateixos, com és el cas del sensor de freqüència cardíaca i altres llibreries que faciliten la connexió Bluetooth amb aquests dispositius.

1.2.4. Android Debug Bridge

Com ja hem dit, treballarem amb Android Studio per fer l'aplicació. Aquest IDE ens permet fer proves amb emuladors de dispositius Android, tant de wearables com de mòbils. També ens facilita el poder fer proves de Debug en els nostres dispositius directament gràcies a l'Android Debug Bridge (ADB), en la secció A.6 de l'Annex s'explica en què consisteix el mode Debug.

Aquest ADB està inclòs dins del SDK d'ajuda que ens proporciona Android (un altre avantatge de la utilització d'Android) i que podem descarregar fàcilment des d'Android Studio mitjançant l'eina Android SDK Manager dins l'apartat d'eines.

L'ADB és necessari per obrir els ports en l'aplicació Android i així crear un canal abstracte entre l'Android Studio, on estem programant l'aplicació, i el dispositiu mòbil, que està connectat a l'ordinador via USB. Només hem d'obrir

un port TCP qualsevol (per exemple el tcp:4444) de manera que es pugui instal·lar l'aplicació al dispositiu connectat per USB.

A continuació s'ha de veure com fer la connexió Bluetooth entre el dispositiu wearable i el dispositiu mòbil. Es pot connectar el dispositiu wearable via USB a l'ordinador, i executar l'aplicació del wearable directament en el rellotge, però interessa poder executar l'aplicació en els dos dispositius simultàniament.

Per fer-ho és necessari instal·lar l'aplicació Android Wear (es pot descarregar de forma gratuïta des de la plataforma Google Play!) i configurar l'aplicació per a que emparelli el dispositiu mòbil amb el wearable, com s'explica en la secció 7 de l'Annex.

Després només s'ha de configurar el dispositiu wearable per a permetre proves de Debug mitjançant ADB en el mateix i ja podrem connectar els dos dispositius (també mitjançant l'ADB en el nostre PC) i fer les proves de l'aplicació directament sobre el nostre dispositiu wearable, a més del dispositiu mòbil, és clar.

En aquest cas, i per facilitar la feina hem fet un petit script que s'ha d'executar cada vegada que es volen fer proves per obrir aquest port de l'ADB.

```
cd "Program Files <x86>/Android/android-studio/sdk/platform-tools"
adb forward tcp:4444 localabstract:/adb-hub
adb connect localhost:4444
adb devices
```

Seguint la primera línia, anem a la carpeta on ens hem descarregat la carpeta de *platform-tools* del SDK, dins d'on tenim instal·lat l'Android Studio, i obrim un port TCP qualsevol (tcp:4444). Després connectem l'ADB al port localhost:4444 i la darrera comanda serveix per mostrar una llista del dispositius connectats.

Dins de l'Annex d'aquest document s'explica més detalladament el procés d'instal·lació i de preparació de l'entorn de programació amb Android Studio. En aquest Annex s'ha intentat proporcionar al lector una petita guia de com instal·lar l'entorn correctament i com s'han de preparar els dispositius abans de començar a programar qualsevol aplicació per no tenir problemes afegits i disposar d'un entorn controlat.

CAPÍTOL 2. DEFINICIÓ DE L'APLICACIÓ

En aquest capítol presentem l'aplicació concreta que hem programat i que utilitza els conceptes que expliquem més detalladament en el capítol 3. Per desenvolupar l'aplicació s'han seguit les guies generals que proporciona Android [8], i específicament la guia per a desenvolupar aplicacions per a wearables [9] així com fòrums de consultes sobre programació i errors de compilació [10] i comunitats de desenvolupadors [11].

L'aplicació desenvolupada serveix per mesurar la freqüència cardíaca. Aquesta es compta en pulsacions per minut (ppm) i correspon al nombre de vegades per minut que el nostre cor batega o es contrau.

Es pretén fer una aplicació que sigui el més simple i lleugera possible, només es posarà en marxa quan l'usuari vulgui monitoritzar la seva freqüència cardíaca. De manera simultània, aquests resultats seran enviats al dispositiu mòbil on es mostren per pantalla, de manera que l'usuari pot saber quina és la seva freqüència cardíaca mirant-ho tant en el wearable com en el dispositiu mòbil.

2.1. Vistes de l'aplicació

Com que es pretén fer una aplicació senzilla, només s'utilitzarà una vista principal en el wearable on anirem mostrant missatges conforme vagin succeint els esdeveniments, i una altra vista secundària per mostrar una gràfica amb els resultats del sensor.

Així doncs, quan obrim l'aplicació, apareix un missatge de benvinguda, com es veu en la Figura 2.1.1, juntament amb tres botons (Start, Stop i View Results) per a que l'usuari interaccioni amb l'aplicació. Addicionalment, mostrem el nivell de bateria del dispositiu wearable, que es va actualitzant conforme va variant, en la part superior de la vista.

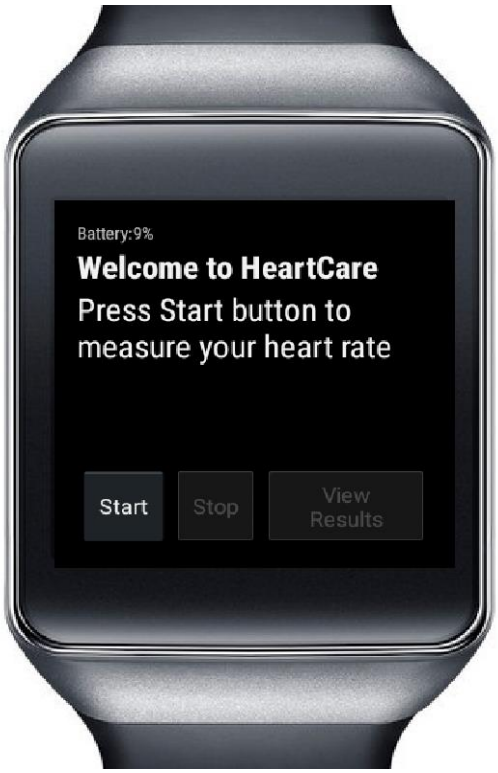

Fig.2.1.1 Pantalla principal de l'aplicació en el wearable

Paral·lelament, l'aplicació en el dispositiu mòbil també reacciona als diferents estats de l'aplicació i, com ja hem dit, utilitzarem aquests missatges per monitoritzar els estats de l'aplicació del wearable.

Si observem la Figura 2.1.2, es pot veure el moment en que l'aplicació mòbil i el wearable es connecten via Bluetooth per començar la transmissió de dades, ja que, en el mètode OnConnected(), s'envia un missatge de benvinguda a l'altre dispositiu quan la connexió es satisfactòria.

Fig.2.1.2 Dispositius connectats

Respecte l'aplicació del wearable, en la part inferior del missatge de benvinguda apareix un segon missatge on se'ns explica com interactuar amb l'aplicació. Es pot comprovar que només tenim una opció per interactuar amb l'aplicació, que és prement el botó de Start, ja que els altres dos botons es troben desactivats en aquest moment.

Quan premem el botó de Start, veure Figura 2.1.3, el missatge de l'aplicació s'actualitza, i ens mostra un missatge informant-nos que s'està mesurant la freqüència cardíaca.

Mentre estem rebent les mostres del sensor veiem com la situació dels botons de la vista principal ha canviat. Ara el botó Start es troba inhabilitat, mentre que els altres dos s'han habilitat.

Fig.2.1.3 Aplicació del wearable al clicar Start

Automàticament, quan l'aplicació ja comença a obtenir les mostres del sensor s'actualitza aquest missatge per mostrar-nos immediatament quin és el valor instantani de freqüència cardíaca, es pot veure en la Figura 2.1.4.

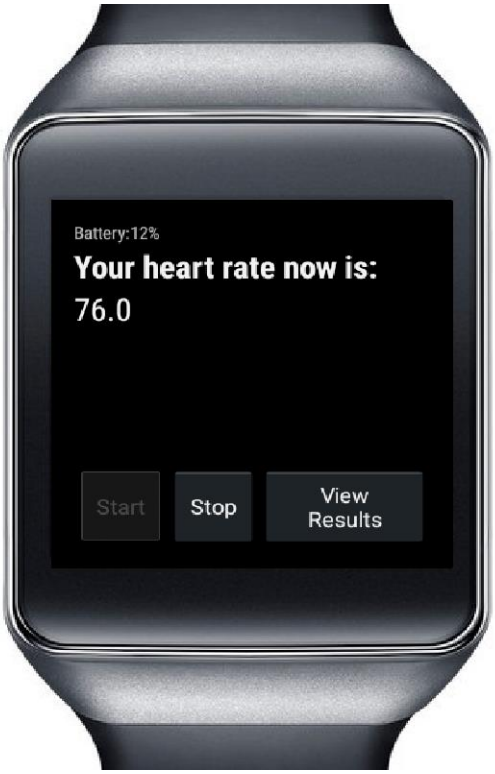

Fig.2.1.4 Visualització mostres en el wearable

El botó Stop farà que cesi la presa de mesures del sensor, i, per tant, aquest s'apagarà, canviant la vista principal de l'aplicació a l'estat inicial, com quan l'obrim per primer cop però, amb una petita diferència, que el botó de View Results aquest cop resta habilitat i es mostra l'última mostra rebuda del sensor, com es pot veure en la Figura 2.1.5.

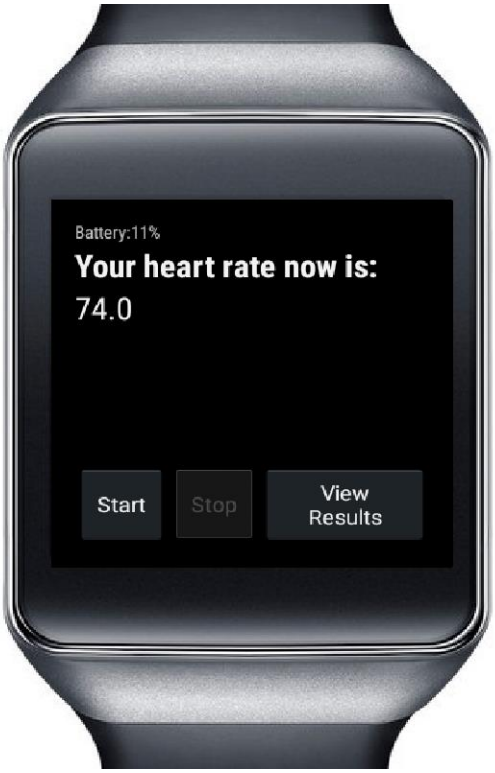

Fig.2.1.5 Aplicació del wearable al clicar Stop

El botó de View Results és el que ens porta a la vista on es visualitza la gràfica amb les mostres del sensor. Com que ja hem obtingut unes mostres de freqüència cardíaca, des de que hem polsat el botó de Start fins que hem parat les mesures amb el botó de Stop, l'aplicació ja ha començat a dibuixar en la gràfica els resultats obtinguts.

Llavors, quan es prem el botó, s'accedeix a la vista on està la gràfica amb els resultats. Com veiem en la Figura 2.1.6, hem enquadrat els resultats dins del rang de valors possibles de freqüència cardíaca en les persones adultes, i es va emplenant amb els valors de la gràfica indefinidament, conforme anem obtenint dades del sensor.

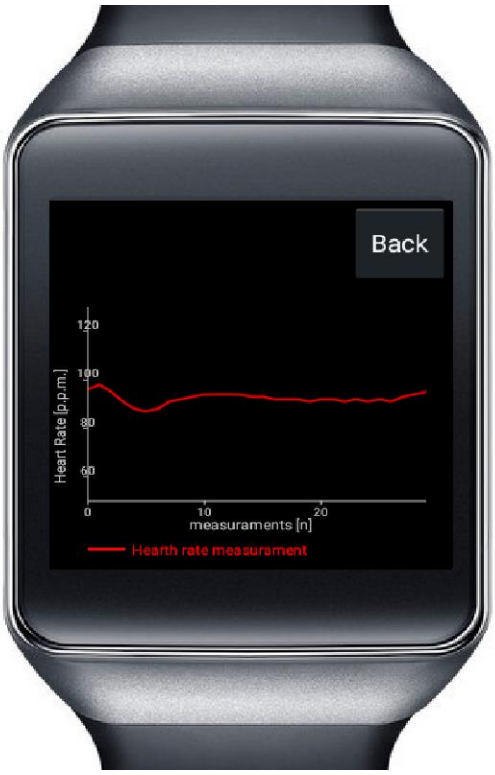

Fig.2.1.6 Gràfica amb els resultats en el wearable

En la vista corresponent a la gràfica hem inclòs un botó per tornar enrere, Back, que fa canviar la vista que es mostra a l'usuari i tornar a la vista principal en el mateix estat en què l'havíem deixat abans d'accedir a veure la gràfica. D'aquesta manera es pot anar canviant a la vista de la gràfica per veure els resultats, sense parar de prendre mostres dins de l'activitat principal.

Entre l'aplicació que s'executa en el dispositiu wearable i la que ho fa en el mòbil, es van enviant missatges per conèixer els diferents estats de l'aplicació, i així actualitzar les interfícies gràfiques dels dos dispositius.

Així doncs, quan es prem el botó Start del wearable, s'envia un missatge a l'aplicació del mòbil que l'interpreta i també actualitza els seus botons.

Fig.2.1.7 Mostres en el wearable i en el mòbil

Com que la comunicació és bidireccional, veure capítol 3.6, quan es premen els botons en el mòbil, també s'envia un missatge al dispositiu wearable, per actualitzar els seus botons a més d'apagar o encendre el sensor depenent del botó clicat.

Quan es clica el botó de Stop, de qualsevol dels dos dispositius, també s'envia un missatge, d'un a l'altre, que actualitza els botons corresponents i apaga el sensor del dispositiu wearable.

2.2. Diagrama de Classes

A l'hora de programar l'aplicació, és molt útil comptar amb el Diagrama de Classes de l'Aplicació. Aquest diagrama presenta les relacions entre les diferents classes de l'aplicació i com es criden, amb els mètodes, entre elles en l'aplicació.

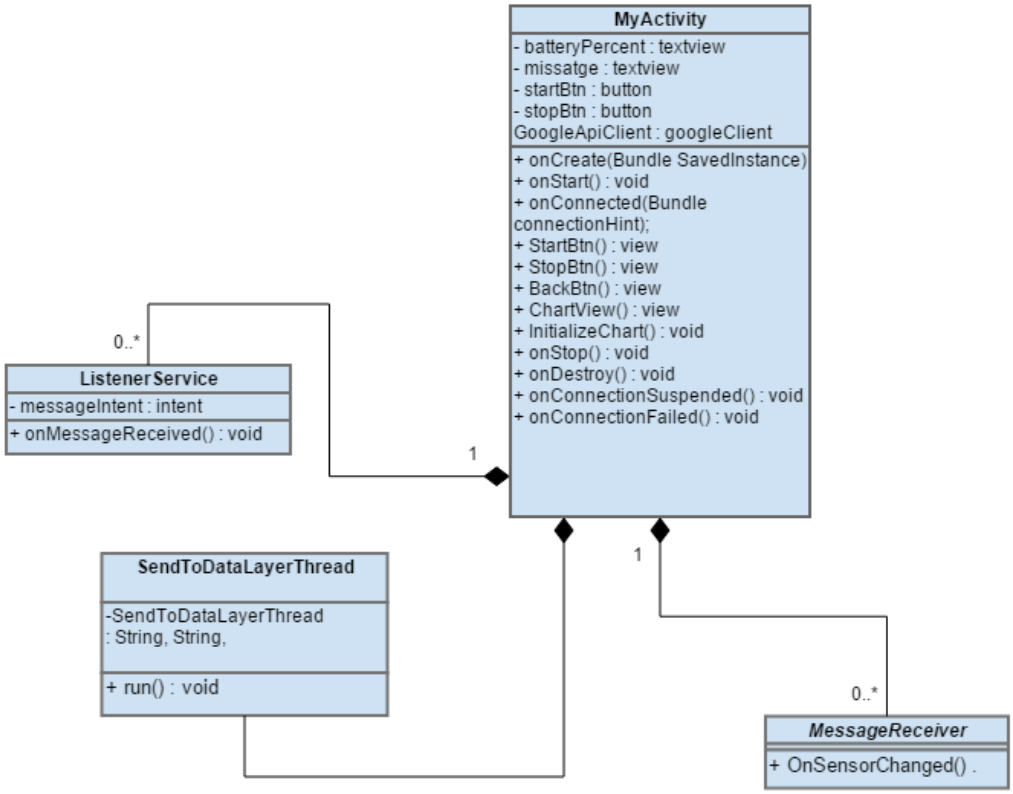

Fig.2.2.1 Diagrama de classes

Com indica el diagrama de classes, l'aplicació consta d'una classe principal MyActivity que es comunica amb les altres tres classes, ListenerService, MessageReceiver i SendToDataLayerThread, per fer algunes de les seves funcions. Comencem parlant de la classe principal.

MyActivity és la classe que es crida només començar l'aplicació. Inclou la declaració dels elements del Layout (vistes de pantalla), i es connecta amb el client de Google API (connectivitat Bluetooth).

En aquesta classe principal, es defineixen els mètodes principals del cicle de vida de qualsevol aplicació Android (veure la secció 3.1, Figura 3.1.2): onCreate(), onStart(), onStop(), onResume(), onDestroy() i també els mètodes onClick() corresponents als botons de l'aplicació.

Dins del mètode onCreate() es crida al Layout de l'aplicació i es connecta l'aplicació amb la corresponent al mòbil. En el mètode onConnected(), per verificar que l'aplicació del wearable i la del mòbil estan connectades, s'envia un missatge a l'altra aplicació, utilitzant el mètodes explicats en el capítol 3.6. També es crida als mètodes onConnectionSuspended() i onConnectionFailed() en cas que la comunicació es talli o sigui errònia, respectivament.

Per enviar i rebre les dades es requereixen les altres classes de l'aplicació. En concret, per enviar dades a través del canal Bluetooth a l'altra aplicació s'ha de cridar a la classe SendToDataLayerThread, veure secció 3.6 per entendre com funciona. Mentre que per rebre dades s'ha definit la classe ListenerService, que

ens permet escoltar els missatges que arriben pel canal Bluetooth, enviant-los a la classe principal amb el mètode OnMessageReceived().

Per rebre les dades del sensor, s'utilitza la classe MessageReceiver que, cridant-la amb el seu mètode OnSensorChanged(), explicat en la secció 3.3, ens permet obtenir la informació del sensor en la mateixa classe principal per mostrar-la per pantalla.

Per tal d'aconseguir una comunicació bidireccional entre les dues aplicacions, aquesta relació entre classes es compleix en totes dues aplicacions, exceptuant que l'aplicació del mòbil no inclou el mètode MessageReceiver() amb el que s'obtenen les mostres del sensor.

La comunicació bidireccional entre els dos dispositius permetrà, a part de passar la informació corresponent al sensor de ritme cardíac, d'un dispositiu a l'altre, poder canviar els estats de l'aplicació del wearable des del mòbil i viceversa, enviant missatges que són interpretats en el ListenerService, com es veu en el capítol 3.6.

2.3 Diagrama de seqüència

Per fer-nos una idea més clara d'aquest enviament de missatges entre les dues aplicacions, ens ajudem d'un diagrama de seqüència com el de la Figura 2.3.1, on es veuen les accions i els missatges que desencadenen les accions de l'usuari.

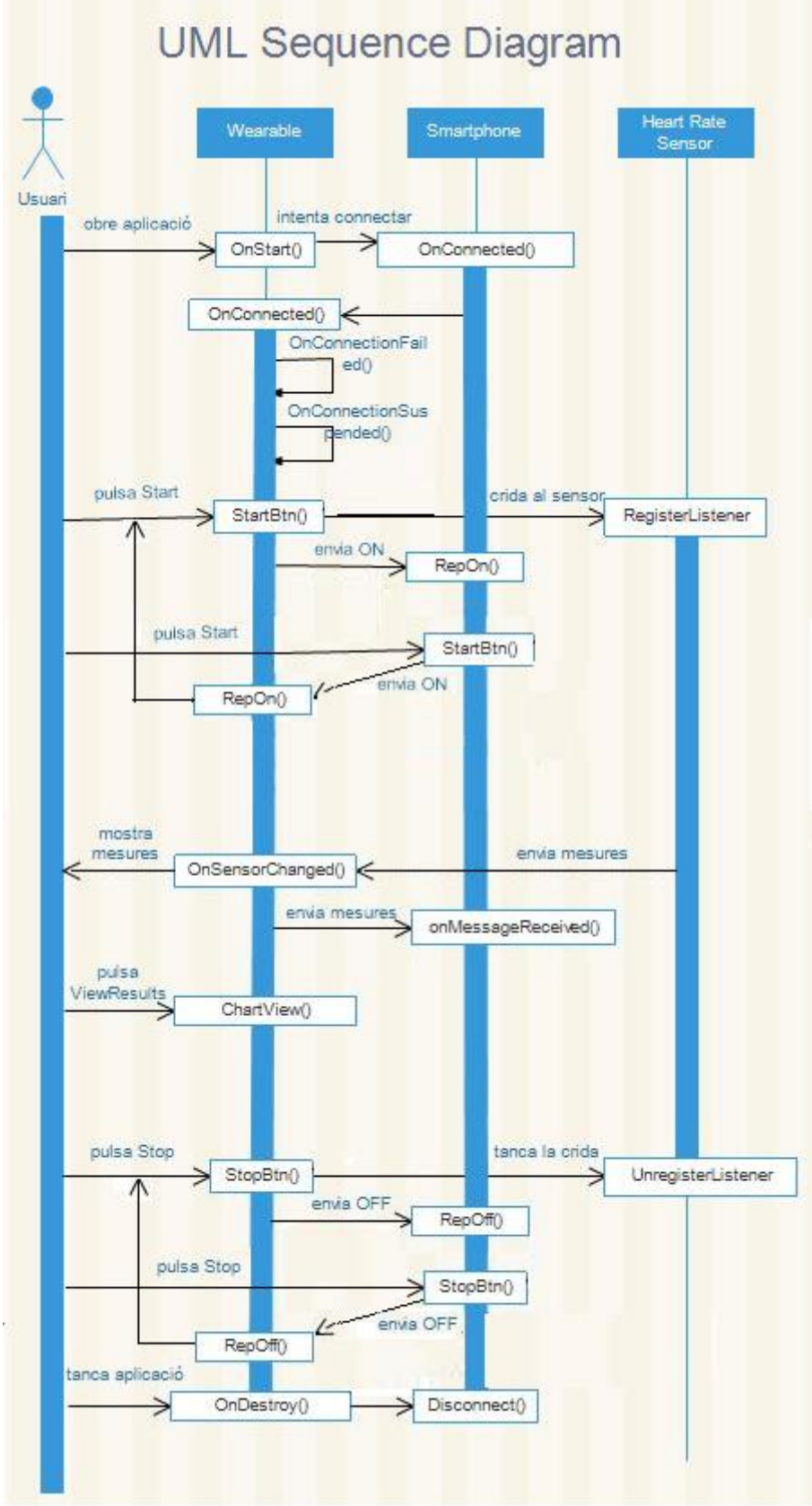

Fig.2.3.1 Diagrama seqüencial

Com es pot veure, en tot moment s'envien missatges amb l'estat de l'aplicació per poder monitoritzar els canvis que es produeixen en el wearable des del mòbil i viceversa.

Com veiem en el diagrama seqüencial, l'usuari només pot realitzar 3 accions en l'aplicació wearable, sense comptar la relativa a obrir/tancar la pròpia aplicació, que consisteixen en prémer els botons de Start, Stop i View Results. També pot clicar els botons de Start i Stop en l'aplicació mòbil.

En el moment que l'usuari obre l'aplicació al wearable, es crida el mètode OnStart() de la classe principal (MyActivity) que connecta els dos dispositius pel canal Bluetooth i es verifica si aquesta connexió s'ha realitzat correctament, en el mètode OnConnected(), o si la connexió no ha sigut satisfactòria cridant els mètodes OnConnectionFailed() o OnConnectionSuspended() en cas que s'hagi tallat, o cancel·lat la comunicació respectivament.

Quan es prem el botó de Start en l'aplicació wearable es crida al sensor amb el mètode RegisterListener per obtenir mostres, i s'envia un missatge ON al mòbil que el llegeix amb el mètode RepOn() a l'Activity principal. El mateix passa amb el botó Stop, que tanca la connexió amb el sensor, deixant de rebre mostres, i envia un missatge OFF al mòbil que es llegeix en el mètode RepOff(). En aquests mètodes del mòbil simplement s'actualitza l'estat dels botons.

Quan es cliquen els botons de l'aplicació mòbil, aquests també envien missatges al wearable. S'envia ON quan es clica el botó Start i OFF quan es clica el Stop. Aquests missatges, son interpretats en l'aplicació del wearable, en els mètodes RepOn() i RepOff(), igual que en el mòbil, i canvien els estats dels botons en l'aplicació del wearable. Però no només això, sinó que criden als mètodes de StartBtn() i StopBtn(), del wearable, per reproduir el mateix cas en el que es cliquen els botons directament en el wearable.

En clicar el botó de View Results en el wearable, aquest crida el mètode ChartView() del wearable que mostra la gràfica per pantalla.

CAPÍTOL 3. APLICACIÓ ANDROID

Per començar a fer l'aplicació Android és necessari tenir uns coneixements bàsics, i no tant bàsics, de com treballa una aplicació Android i tots els elements que tenim a la nostra disposició per poder desenvolupar l'aplicació al nostre gust.

Primer, hem de saber com treballen les aplicacions Android, per això es bàsic conèixer el model-vista-controlador i el cicle de vida de les activitats en Android. Es començarà explicant aquests dos conceptes, per després veure diferents funcionalitats que cal entendre per poder-les utilitzar dins de l'aplicació desenvolupada en el projecte i detallada en el capítol 2.

3.1. MVC i Lifecycle

Les aplicacions en Android utilitzen un patró d'arquitectura de software que coneixem com a model-vista-controlador, en el qual, les parts d'una aplicació es diferencien en tres grups:

- El Model: és la representació de la informació que utilitzem
- La Vista: és on mostrem la informació a l'usuari
- El Controlador: fa d'intermediari entre la vista i el model responent a la informació que demana l'usuari

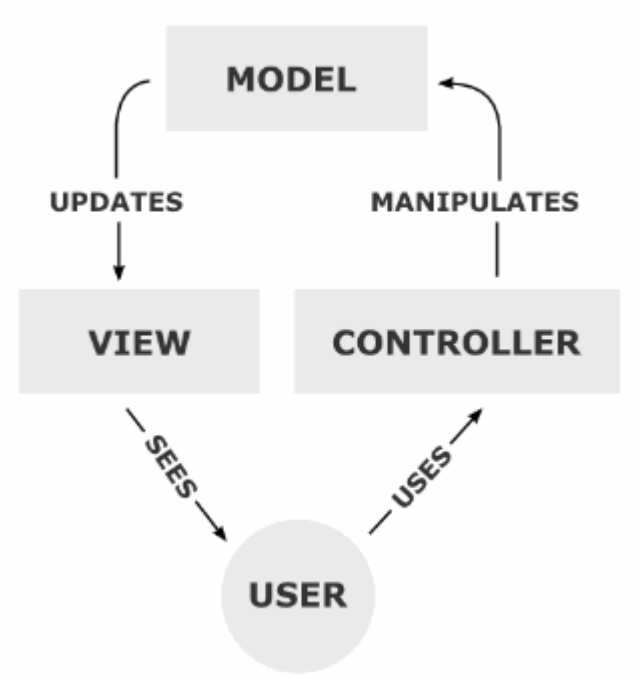

Fig. 3.1.1 Patró model-vista-controlador

Per tal d'aplicar els coneixements que hem adquirit al llarg de la carrera, per fer la part del controlador i del model, utilitzarem llenguatge Java, i per fer les vistes i les interfícies d'usuari utilitzarem llenguatge XML, ja que també són els més utilitzats per fer aplicacions Android.

Una aplicació Android segueix un cicle de vida, en aquest cicle de vida, es pot veure un exemple en la Figura 3.1.2, es veu com l'Activity principal de l'aplicació, que és on es descriu el que l'usuari pot fer i que normalment estan relacionades amb vistes, segueix un cicle d'estats de l'aplicació que depèn de les accions que faci l'usuari o es facin automàticament.

Quan l'aplicació s'obre per primera vegada, comença aquesta Activity principal i crida el mètode OnCreate(), on s'inicien alguns elements de l'aplicació i es passa al mètode OnStart(), on normalment es genera una primera vista de l'aplicació.

Aquesta aplicació es manté en primer plànol fins que l'usuari la tanca (l'Activity rebrà una crida al seu mètode OnStop()) o la deixa en segon terme (es cridarà el mètode OnPause() de l'Activity).

Mentre l'aplicació està en segon terme consumeix menys capacitat del dispositiu, i si l'usuari només l'ha deixada en segon terme i no l'ha tancada, es pot recuperar l'aplicació en el punt en que s'havia deixat mitjançant una crida al mètode OnResume(), que fa passar l'aplicació a primer terme per a que l'usuari pugui interactuar-hi.

El mètode OnRestart() funciona semblant al mètode OnResume() però cridant de nou la funció OnStart(), inicialitzant així les variables de l'aplicació. Finalment, el mètode OnDestroy() tanca completament l'aplicació i esborra els estats de l'aplicació, així com la informació emmagatzemada.

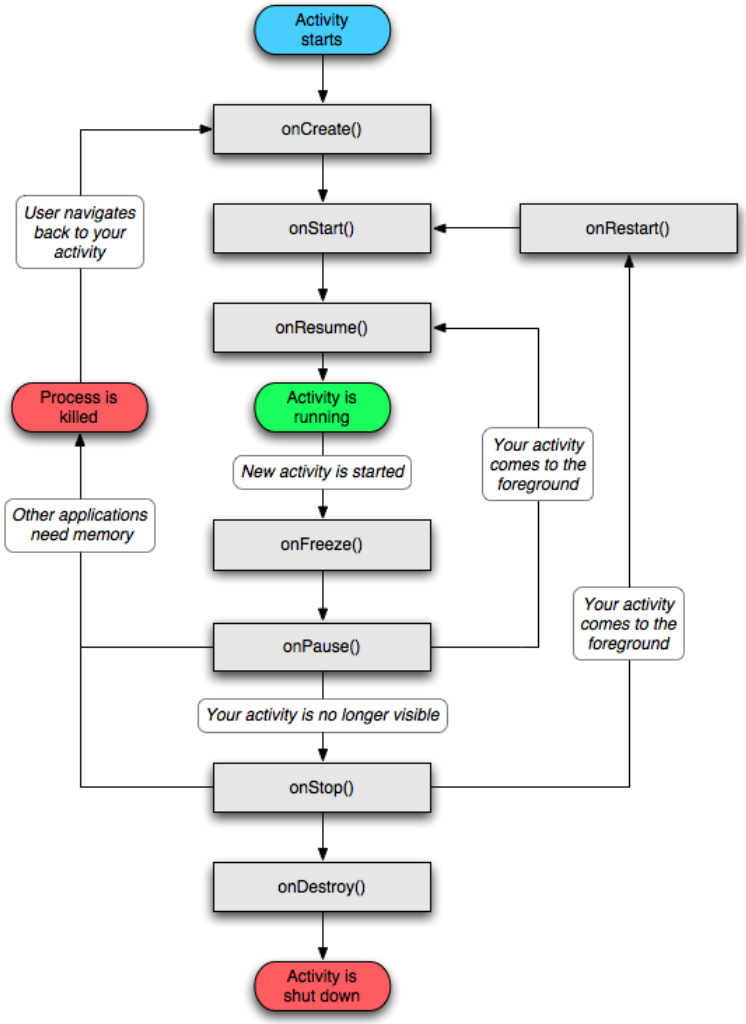

Fig.3.1.2 Lifecycle bàsic de l'Activity principal en Android

Amb aquests dos conceptes de com s'ha de treballar en aplicacions Android clars, podem passar a detallar els mecanismes necessaris per desenvolupar una aplicació com la que s'explica al capítol 2.

3.2. Elements del projecte

Quan comencem un nou projecte, utilitzant Android Studio tenim l'opció de començar a fer una aplicació tenint en compte que estem considerant dues aplicacions, una per al dispositiu wearable i l'altra per al smartphone que estarà vinculat amb aquest wearable i que treballaran de manera coordinada i simultània. Encara que Android Studio també ens permet programar l'aplicació per al Wearable de forma individual.

Per això, quan creem el projecte, tenim l'opció de començar directament amb dos mòduls principals, un per al wearable i l'altre per al mòbil, com es veu en la Figura 3.2.1. Per cadascun dels mòduls del projecte es creen, automàticament, una sèrie d'arxius per ajudar a l'usuari.

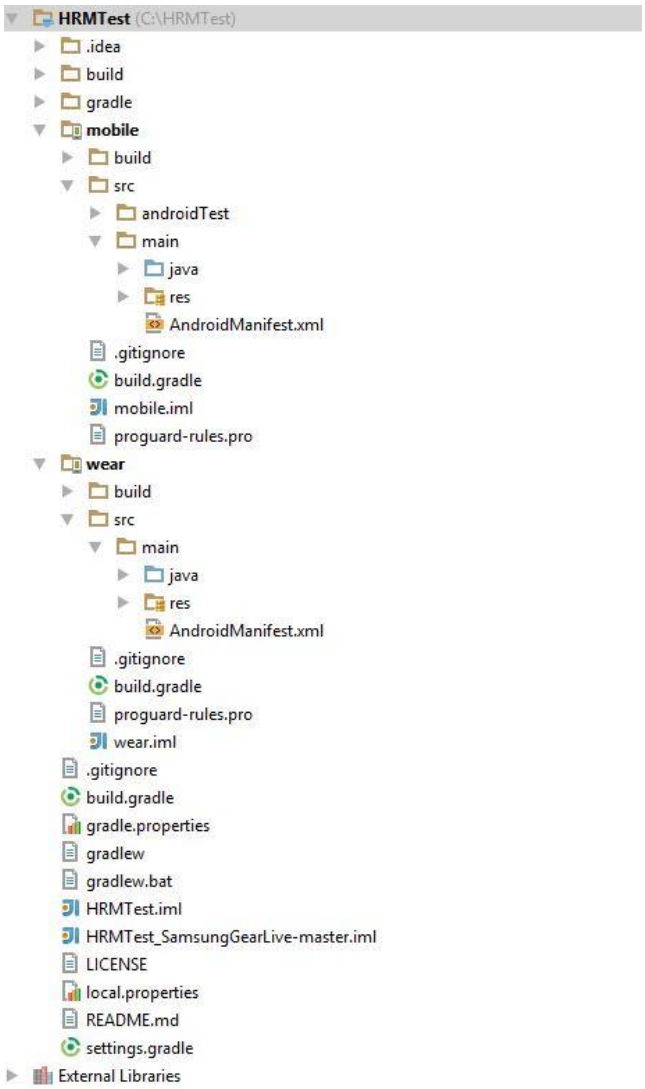

Fig.3.2.1 Visualització de dos mòduls en el mateix projecte en Android Studio

Android Studio crea un arxiu Java en la carpeta res/main/java, veure Figura 3.2.1, on es pot programar l'Activity principal de l'aplicació, una sèrie de carpetes en la ruta source/main/res, veure Figura 3.2.1, per crear el Layout, que és on es programa com serà la vista de l'aplicació i la relació amb l'Activity que li correspon, veure capítol 3.4 on es descriu com funcionen els Layouts.

Cadascun dels mòduls, també conté el fitxer AndroidManifest on es declaren els permisos que es volen donar a l'aplicació i on es declara l'Activity principal amb les relacions amb les altres classes. En l'Annex, secció 4, es dóna un exemple d'aquest fitxer AndroidManifest.

També conté el fitxer *build.gradle*, que és una eina que ens proporciona Android Studio [7] per compilar les llibreries necessàries en el projecte directament del repositori d'Android a Internet, així, només cal descarregar aquestes llibreries el primer cop que s'executa l'aplicació i després aquest fitxer ens anirà descarregant les últimes versions d'aquest repositori a mesura que s'actualitzin, veure secció 3 de l'Annex per entendre com funciona.

Android Studio també dóna l'opció de treballar sense connexió, en el qual s'utilitzen les llibreries descarregades sense intentar descarregar les últimes versions del repositori d'Android. Pot ser molt útil per programar sense connexió a Internet.

En el cas concret de la nostra aplicació, el mòdul del mòbil ens servirà, bàsicament, per mostrar la informació que obtenim de l'aplicació del wearable per la pantalla del mòbil. Això ens permetrà fer una aplicació més amigable de cara a l'usuari, que té diverses opcions per utilitzar l'aplicació i per visualitzar la informació.

Per altra banda, en el mòdul del wearable és on tenim la part més rellevant de l'aplicació ja que és d'on obtenim la informació del sensor de polsos cardíacs.

Típicament l'aplicació per al wearable ha de ser lleugera, amb poc cost computacional, ja que no disposem de gran capacitat de processament en el *wearable*, i deixem aquestes operacions que requereixen més processament per al dispositiu mòbil, que normalment té més capacitat.

3.3. Sensor

Un punt important és saber com obtenir la informació dels diferents sensors. Per això comptem amb diverses llibreries específiques d'Android que permeten obtenir, tractar i monitoritzar les dades d'aquests.

Per obtenir aquesta informació, comptem amb les llibreries i les APIs que Android té a l'abast de qualsevol usuari per utilitzar-les lliurement, i només hem d'indicar en el Manifest a quins sensors volem accedir. En aquest exemple volem accedir als sensors del cos:

```
<uses-permission android:name="android.permission.BODY_SENSORS" />
```

Un cop tenim els permisos, s'utilitza l'objecte SensorManager que obté informació del sistema amb els mètodes getSystemService(), així com el sensor que interessa amb el mètode getDefaultSensor().

Li indiquem doncs, en el mètode OnStart() de l'Activity principal, que s'executa al iniciar l'aplicació, quin és el sensor al que volem accedir, en el nostre cas el sensor de polsos cardíacs:

```
mSensorManager = ((SensorManager)getSystemService(SENSOR_SERVICE));
mHeartRateSensor = mSensorManager.getDefaultSensor(Sensor.TYPE_HEART_RATE);
```

Com que volem que la nostra Activity principal sigui la que obté els valors del sensor, només hem de fer la següent crida quan iniciem l'aplicació (mètode OnStart()),

```
mSensorManager.registerListener(this, this.mHeartRateSensor, 3);
```

per a que l'aplicació agafi el valor del sensor.

Amb el mètode explicat, l'Activity rep la informació del sensor cada cop que aquest canvia a través del mètode OnSensorChanged().

```
public void onSensorChanged(SensorEvent sensorEvent) {
    try {
        latch.await();
        if (sensorEvent.values[0] > 0) {
            String message = String.valueOf(sensorEvent.values[0]);
        }

    } catch (InterruptedException e) {
        Log.e(TAG, e.getMessage(), e);
```

Aquest mètode permet guardar en una variable tipus string (variable message) la informació dels polsos cardíacs cada cop que s'obté un nou valor del sensor.

3.4. Layouts

Els Layouts són els fitxers on s'especifica com s'han de mostrar les vistes gràficament. En ell es declaren tots els objectes de l'aplicació com els botons o el textos que volem mostrar a l'usuari.

Aquests fitxers Layout, que es descriuen mitjançant el llenguatge XML, estan relacionats amb la intel·ligència de l'aplicació, programada en l'arxiu Java, de forma que les accions de l'usuari que es recullen en el Layout es rebin en el controlador de l'aplicació per processar la informació.

En l'aplicació del capítol 2, s'utilitzen, bàsicament, dos elements d'aquest Layout; botons i quadres de text.

Es pot dotar d'activitat als botons de l'aplicació fàcilment en Android, ja que, quan es defineix el botó en el fitxer Layout, incorpora un camp OnClick on se li pot indicar una funció a realitzar quan l'usuari el clica.

```
  <Button
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Start"
      android:id="@+id/StartBtn"
      android:onClick="StartBtn"
      android:layout_alignParentBottom="true"
      android:layout_alignParentStart="true" />
```

En aquest exemple de botó utilitzat en l'aplicació es pot comprovar com el mètode OnClick d'aquest botó crida el mètode StartBtn() de la nostra Activity principal.

En l'aplicació es treballa amb quadres de text, *TextView*, per que cada cop que el missatge que l'aplicació mostra a l'usuari es vagi actualitzant el text amb els diferents estats d'aquesta i el quadre es vagi ajustant a la mida del text.

Només s'ha de dotar el quadre de text amb un identificador per tal que, en el moment que es desitgi canviar el text des del programa Java, només s'hagi d'especificar el nou text i l'identificador del quadre de text on volem afegir-lo.

```
rate = (TextView) stub.findViewById(R.id.rate);
rate.setText("");
```

Aquesta aplicació compta amb dues vistes principals, la de l'aplicació en sí, i la que mostra la gràfica amb les mostres a l'usuari, per tant, l'usuari ha de poder obrir la vista de la gràfica i tornar a la vista principal de l'aplicació de forma fàcil.

També interessa obrir la gràfica amb la informació dels polsos cardíacs en el màxim espai possible, ja que conté molta informació i es interessant veure-la amb precisió. Una manera de fer això és obrint aquesta gràfica en un nou Layout.

Tot i això, per fer aquesta gràfica, el mètode necessita obtenir els valors del sensor de polsos cardíacs, i aquests s'obtenen en la classe principal; no és recomanable obrir dos arxius XML, o dos vistes, per la mateixa Activity.

Una possibilitat seria fer-ho amb fragments ja que aquests permeten l'intercanvi de dades entre classes, i passar la informació d'una classe a l'altra, però realment no és necessari obrir un nou fragment només per mostrar la gràfica. Així que es pot utilitzar una tècnica que ens proporciona l'API d'Android anomenada ViewSwitcher.

El ViewSwitcher, tal i com el seu nom indica, ens permet canviar d'una vista a l'altra en una mateixa Activity. Això estalvia el procés de tenir que obrir una nova Activity, amb una nova vista, i passar la informació d'una a l'altra.

En aquest cas, pot ser molt útil ja que aquesta segona vista només contindrà la gràfica amb els resultats i un botó per tornar enrere a la primera vista. Fer-ho d'aquesta manera simplifica molt el codi i el fa més lleuger.

Només cal definir en la vista principal, dins de l'arxiu XML, el ViewSwitcher i incloure els dos Layouts dins d'aquest.

```
<?xml version="1.0" encoding="utf-8"?>
<ViewSwitcher xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/profileSwitcher"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

<RelativeLayout
…
</RelativeLayout>
```

```
<RelativeLayout
…
</RelativeLayout>

</ViewSwitcher>
```

Després només s'ha de fer constar en l'aplicació que s'està fent servir aquest ViewSwitcher. Per això, en el mètode OnCreate(), on es defineix el Layout de l'Activity principal, es crea aquest objecte que farà referència al XML.

```
switcher = (ViewSwitcher) findViewById(R.id.profileSwitcher);
```

Ara ja només cal cridar els mètodes que incorpora aquest objecte. Principalment, s'utilitzen els dos mètodes que permeten anar a la següent vista i tornar a l'anterior.

```
switcher.showNext();
switcher.showPrevious();
```

Aquests mètodes permeten passar d'un Layout a l'altre, i per tant, han de ser cridats quan es vulgui canviar de vista, obrint un nou Layout que ocupi tota la pantalla, i també quan es vulgui tornar a la vista de l'aplicació principal.

3.5. Gràfica amb resultats

Aquesta informació que s'obté del sensor ens pot ésser molt útil, però s'han d'agafar aquestes mostres de pols cardíac, que ens va donant el sensor, i processar-les per mostrar-li a l'usuari. D'aquesta manera, l'usuari pot tenir coneixement de l'evolució que ha seguit el seu pols cardíac des de que ha començat a mesurar.

Una manera de mostrar els resultats és amb una gràfica on es visualitzi el valor de pols cardíac (ppm) al llarg del temps o de les mostres obtingudes.

Per crear gràfiques dins d'una aplicació per a wearable existeix una llibreria molt simple i molt utilitzada en aplicacions Android anomenada AChartEngine que s'haurà d'incloure en el fitxer AndroidManifest (veure secció 4 de l'Annex).

Aquesta llibreria treballa amb 3 conceptes molt bàsics que hem de controlar:

- Dataset: És on es guarden les dades que es volen mostrar en la gràfica, en aquest cas, les mostres del sensor de freqüència cardíaca.

- The view: Es refereix al tipus de gràfica que es vol crear, la llibreria conté diversos tipus de gràfics, com el de barres o el de variacions. En aquest cas s'utilitzarà el lineal.

- Renderer: És el que s'encarrega de controlar com es dibuixa la gràfica i el que s'hi mostra. Hi ha dos tipus de *Renderer* el que controla quina és

la informació que s'ha de mostrar en la gràfica, i el que controla com s'ha de mostrar aquesta informació.

- Chart Factory: S'encarrega de combinar la informació del *Dataset* amb les indicacions del *Renderer* per mostrar la gràfica en una nova vista.

En el nostre cas, el *Dataset* el generem a partir de les mostres que s'obtenen del sensor en forma de string (variable message), així doncs, s'ha d'indicar, en el OnSensorChanged(), que, cada cop que el sensor obtingui un nou valor, s'afegeixi al *Dataset.*

```
HRSeries.add(mostres, Double.parseDouble(message));
mostres++;
```

Estem afegint la informació del sensor en l'eix Y i la mostra en la que s'ha rebut en l'eix X de la sèrie de mostres. La següent mostra que obtinguem es dibuixarà a continuació en l'eix X.

Ara que ja tenim el Dataset, per crear la gràfica s'ha de definir el *Renderer* per obtenir la informació del *Dataset* i mostrar-la en la vista. Per tant, en l'Activity principal definim el *Renderer* i li indiquem com volem que es visualitzi la línia en la gràfica.

```
HRRenderer renderer = new XYSeriesRenderer();
renderer.setLineWidth(2);
```

Ara que ja obtenim la informació i s'ha creat el *Renderer*, s'ha de crear la vista. Per això s'utilitza el mètode, definit en la llibreria de Achartengine, initializeChart() cridant-lo en l'Activity principal de l'aplicació per que dibuixi la gràfica en sí.

```
private void initializeChart()
{
    HRRenderer=new XYSeriesRenderer();
    myRenderer=new XYMultipleSeriesRenderer();
    myRenderer.addSeriesRenderer(HRRenderer);
    HRSeries=new XYSeries("Hearth rate measurament");
    mySeries=new XYMultipleSeriesDataset();
    mySeries.addSeries(HRSeries);
  }
```

Un cop inicialitzada, s'ha de mostrar aquesta gràfica en un Layout que ocupi tota la pantalla del dispositiu wearable, veure Figura 2.1.6, per poder apreciar-la clarament. Això implica crear un nou Layout per la gràfica que es mostri a l'usuari, en el nostre cas, en el mètode OnClick() del botó corresponent a la visualització dels resultats (View Results).

```
  LinearLayout layout=(LinearLayout)findViewById(R.id.chart);
  if(myChart==null)
```

```
{
   myChart=ChartFactory.getLineChartView(this, mySeries, myRenderer);
   layout.addView(myChart);
}
else
{
   myChart.repaint();
}
```

Com es pot observar, es crea un nou Layout MyChart per afegir-hi la gràfica, però està en la mateixa Activity, en la següent secció 3.4 s'explica com resoldre aquesta casuística.

3.6. Comunicació wearable/mòbil

La comunicació entre els dos dispositius és la part més complexa de la nostra aplicació. Aquesta es realitza utilitzant Bluetooth i, per tant, s'ha de dotar la nostra aplicació d'uns serveis per escoltar el canal de Bluetooth.

La versió 5 dels serveis de Google Play, amb la que es treballa en aquest projecte, inclou una API especial per a missatges del wearable. Gràcies a aquesta API, aquests missatges que es volen enviar des del wearable al mòbil via Bluetooth segueixen el procés indicat (veure Fig. 3.6.1) per enviar el missatge pel canal Bluetooth. En aquest procés es simplifica el missatge que es vol enviar per tal que pel canal Bluetooth s'hi transporti la mínima quantitat d'informació possible.

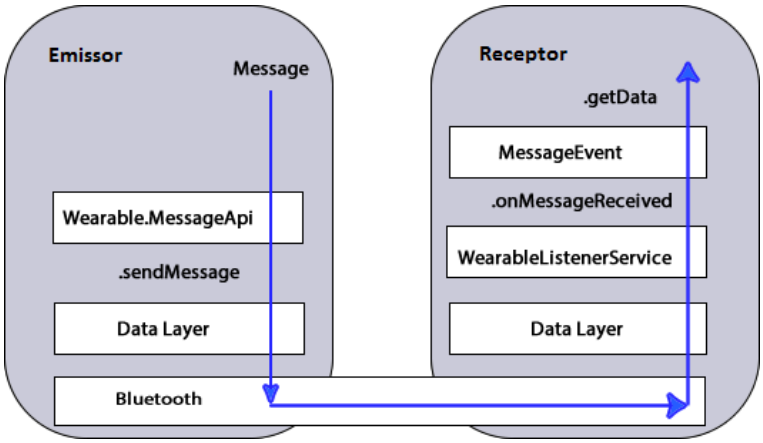

Fig.3.6.1 Enviament de dades per Bluetooth

En la Figura 3.6.1 es veuen les classes que intervenen en l'enviament de dades del Wearable al mòbil. S'ha de puntualitzar que, com que la comunicació és bidireccional i s'envien missatges en els dos sentits, aquests mètodes també es produeixen en l'altra direcció, del mòbil al wearable. Així que el diagrama de la Figura 3.6.1 és aplicable al mòbil enviant els missatges al wearable.

Seguint fixant-nos en la Figura 3.6.1, l'emissor envia el missatge mitjançant el mètode SendMessage de l'API que utilitzem per al wearable. En el receptor, es crea un WearableListenerService que escolta el missatge que arriba des de l'altra part del canal Bluetooth, i invoca un mètode OnMesssageReceived() per tractar aquest missatge.

Per crear aquests mètodes que envien i reben els missatges, s'hauran de crear dues classes que utilitzen l'API d'Android, específicament la part per a wearables.

A la primera d'aquestes classes li direm SendtoDataLayer; bàsicament serveix per enviar un missatge a tots els nodes connectats. Com que el mètode SendMessage d'aquesta classe podria bloquejar l'Activity principal, és recomanable crear un nou Thread (un fil d'execució paral·lel).

```
class SendToDataLayerThread extends Thread {
    String path;
    String message;
    GoogleApiClient googleClient;

    // Constructor to send a message to the data layer
    SendToDataLayerThread(String p, String msg, GoogleApiClient gc) {
        path = p;
        message = msg;
        googleClient = gc;
    }

    public void run() {
        NodeApi.GetConnectedNodesResult nodes =
          Wearable.NodeApi.getConnectedNodes(googleClient).await();
        for (Node node : nodes.getNodes()) {
            MessageApi.SendMessageResult result =
                    Wearable.MessageApi.sendMessage(googleClient, node.getId(),
                    path, message.getBytes()).await();
```

Un cop feta aquesta classe, l'únic que s'ha de fer és crear un nou client de GooglePlay Services quan es creï l'aplicació, en el mètode OnCreate() de l'Activity principal.

```
  googleClient = new GoogleApiClient.Builder(this)
    .addApi(Wearable.API)
    .addConnectionCallbacks(this)
    .addOnConnectionFailedListener(this)
    .build();
```

Al començament de l'aplicació, típicament en el OnStart(), s'ha de connectar aquest client de Google utilitzant,

```
  googleClient.connect();
```

Finalment, en el mètode de l'aplicació que es desitgi s'ha d'omplir la variable 'message' amb la informació que es vulgui enviar. En el cas de la nostra aplicació, els valors del sensor de polsos cardíacs que s'han obtingut amb els mètodes del sensor, a més d'afegir quin és el 'path' al que es vol enviar el missatge, fent la següent crida

```
new SendToDataLayerThread("/message_path", message).start();
```

Aquest enviament del missatge es pot fer un cop per rebre el valor del sensor de polsos cardíacs, però el que ens interessa és que s'enviï cada cop que el valor del sensor canvia. Per tant s'ha de fer aquest enviament, a part de dins del mètode OnConnected() explicat en el capítol 2.1, cada cop que es cridi al mètode OnSensorChanged(), explicat a la secció 3.3.

És molt important que quan es tanqui l'aplicació, es tanqui també el client de Google, ja que si no aquest es quedaria obert i suposaria un consum de bateria innecessari per l'aplicació. Només s'ha de cridar al mètode de desconnexió en el OnStop() de l'aplicació,

```
if (null != googleClient && googleClient.isConnected()) {
  googleClient.disconnect();
  }
```

Anem ara a veure com es fa la recepció d'aquest missatge a l'altre costat del canal Bluetooth.

Quan es crida el mètode SendToDataLayer(), l'emissor envia el missatge a tots els nodes del canal Bluetooth, però només agafa aquest missatge aquell node que estigui interessat en aquesta informació. En aquest cas, l'aplicació del mòbil és el receptor d'aquesta informació, per tant s'ha d'indicar d'alguna manera que l'aplicació vol rebre aquesta informació.

Amb aquest objectiu, a l'aplicació del receptor, s'ha de crear una segona classe (apart de l'Activity principal), que anomenarem ListenerService,

```
public class ListenerService extends WearableListenerService {

  @Override
  public void onMessageReceived(MessageEvent messageEvent) {

    if (messageEvent.getPath().equals("/message_path")) {
      final String message = new String(messageEvent.getData());
    }
    else {
      super.onMessageReceived(messageEvent);
    }
```

S'assigna un 'path' i quan l'emissor envia el missatge amb la informació i el 'path' corresponent, aquest receptor veu que coincideixen els valors

corresponents al 'path' i per tant entén que les dades contingudes dins d'aquest missatge van dirigides a ell. El receptor guarda aquest missatge en una variable de tipus string; en el nostre cas aquest conté els valors corresponents al pols cardíac enviat des del wearable.

S'ha de tenir en compte però, que el ListenerService encara no és capaç de mostrar aquest missatge en l'Activity principal, ja que s'està executant en un Thread diferent. El que s'ha de fer és enviar el missatge rebut cap a l'Activity principal. Això s'aconsegueix creant un objecte LocalBroadcastManager.

Dins de la classe on es rep el missatge (ListenerService) s'afegeix,

```
      // Broadcast message to wearable activity for display
      Intent messageIntent = new Intent();
      messageIntent.setAction(Intent.ACTION_SEND);
      messageIntent.putExtra("message", message);
      LocalBroadcastManager.getInstance(this).sendBroadcast(messageIntent);
```

Ara s'ha de crear la classe MessageReceiver que estengui la Broadcast Receiver i que implementi el mètode OnReceive() per rebre els missatges que reenvia internament el ListenerService.

```
  public class MessageReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
      String message = intent.getStringExtra("message");
      // Display message in UI
      mTextView.setText(message);
    }
```

Després aquesta classe es registra dins l'Activity per a que sigui la responsable de gestionar els missatges que arribin.

```
  // Register the local broadcast receiver
  IntentFilter messageFilter = new IntentFilter(Intent.ACTION_SEND);
  MessageReceiver messageReceiver = new MessageReceiver();
  LocalBroadcastManager.getInstance(this).registerReceiver(messageReceiver,
messageFilter);
```

S'ha de tenir en compte, molt important, que un cop l'aplicació s'atura, en el OnStop() o en el OnDestroy(), s'ha de tancar aquesta 'subscripció' al canal de Broadcast, ja que, en cas contrari l'aplicació seguiria connectada a aquest servei de Broadcast i s'estaria malgastant la bateria del dispositiu amb l'aplicació.

```
 googleClient.disconnect();
```

3.7. Bateria

Per mesurar el consum de bateria de la nostra aplicació es necessita un mètode capaç d'obtenir el nivell de bateria del dispositiu wearable. Aquesta informació ha de ser visible en tot moment i s'ha d'anar actualitzant en cas de canvis.

El mètode per obtenir el nivell de bateria d'un dispositiu està molt relacionat amb el mètode utilitzat per obtenir els missatges del canal de Bluetooth de l'altra aplicació, descrit en la secció 3.6.

S'utilitza un objecte BroadcastReceiver en el OnCreate() de l'Activity principal que estigui 'subscrit' a la informació del dispositiu, en aquest cas la bateria, per després només implementar el mètode getBatteryPercentatge() en l'Activity principal, que agafi aquesta informació, que la mostri per pantalla, i que es vagi actualitzant conforme va canviant aquest nivell de bateria.

```
private void getBatteryPercentage() {
BroadcastReceiver batteryLevelReceiver = new BroadcastReceiver() {
       public void onReceive(Context context, Intent intent) {
           int currentLevel = intent.getIntExtra(BatteryManager.EXTRA_LEVEL, -1);
           int scale = intent.getIntExtra(BatteryManager.EXTRA_SCALE, -1);
           int level = -1;
           if (currentLevel >= 0 && scale > 0) {
               level = (currentLevel * 100) / scale;
           }
           batteryPercent.setText("Battery Level: " + level + "%");
       }
   };
   IntentFilter batteryLevelFilter=new IntentFilter(Intent.ACTION_BATTERY_CHANGED);
   registerReceiver(batteryLevelReceiver, batteryLevelFilter);
}
```

Es pot apreciar com l'API ja permet agafar el valor del nivell de bateria amb aquest BroadcastReceiver, després només s'ha de tractar la informació rebuda, escalant el nivell de bateria que s'obté amb el màxim possible per obtenir el percentatge de bateria que resta en el dispositiu. Aquest servei s'ha de mantenir funcionant sempre que l'aplicació estigui oberta, així es va actualitzant el valor de la bateria cada cop que aquest canviï.

3.8. Bluetooth 4.0

Una part important de la nostra aplicació consisteix en la comunicació entre el dispositiu wearable i el mòbil mitjançant un canal Bluetooth.

S'ha de tenir en compte, però, que Bluetooth 4.0 incorpora la nova tecnologia emergent Bluetooth Low Energy pensada per enviar la informació amb menys potència i dins d'un radi de cobertura menor, tenint això un impacte directe en la bateria.

Per aquest motiu es comença a utilitzar en grans xarxes de sensors on es busca el mínim consum de bateria possible tenint en compte que aquests dispositius sensors acostumen a ser petits i no se'ls pot dotar de grans bateries.

Com que la obtenció de dades en la nostra aplicació es realitza amb un sensor, el sensor òptic de polsos cardíacs del dispositiu wearable, es podria estudiar si fer la comunicació entre el dispositiu mòbil i el wearable utilitzant Bluetooth Low Energy. Això queda fora de l'abast d'aquest projecte i es deixa com a treball futur.

CAPÍTOL 4. CONSUM DE BATERIA

Com a punt final d'aquest projecte, i per comprovar el correcte funcionament de l'aplicació, es realitza un estudi del consum de bateria del dispositiu wearable. Un dels objectius és veure quin és el consum que afegeix l'aplicació que hem dissenyat.

4.1. Quines proves?

Es consideren 5 escenaris diferents per avaluar el consum de bateria.

El primer d'aquests consisteix en mesurar el consum quan el wearable està en mode repòs. Bàsicament, el dispositiu wearable actua com un rellotge normal, i només activem la pantalla per mirar el nivell de bateria i l'hora.

El segon escenari consisteix en sincronitzar el mòbil amb el dispositiu wearable a través de l'aplicació Android Wear. En aquest cas, totes les aplicacions que estiguin incloses dins d'aquesta aplicació d'Android Wear poden enviar informació al wearable de manera que ens apareguin notificacions amb informació de diferents aplicacions, com per exemple un missatge de Whatsapp que rebem o una alarma que teníem programada en el mòbil.

Al wearable només arriba una notificació i en cap cas s'obre cap programa en el dispositiu; en el cas d'un missatge de Whatsapp es pregunta a l'usuari si vol obrir el missatge rebut en el mòbil. Definim llavors un comportament d'ús realista de l'aplicació, de manera que aquest sigui el més fidedigne a la realitat possible. Aquest s'utilitzarà en la resta d'escenaris.

Aquest ús realista, quan el dispositiu wearable i el mòbil estan connectats, consisteix en que, ja que el mòbil envia notificacions generades per aplicacions com Whatsapp o correus electrònics, aquestes notificacions apareixen en el wearable però no s'obre l'aplicació. La notificació simplement es descarta un cop llegida i el rellotge torna a l'estat de repòs en el qual s'apaga la pantalla mostrant només el rellotge.

El tercer escenari consisteix en tenir el dispositiu wearable sincronitzat amb el mòbil i amb l'aplicació de mesura de freqüència cardíaca oberta i funcionant. En aquest cas, a més d'obtenir les mostres de ritme cardíac, aquestes s'envien pel canal Bluetooth al dispositiu mòbil immediatament.

El quart escenari consisteix en utilitzar l'aplicació però enviant només una de cada tres mostres que obtenim amb el sensor pel canal Bluetooth.

El cinquè i últim escenari és similar al quart, però en comptes d'enviar una de cada tres mostres que ens proporciona el sensor, n'enviem una de cada deu.

Per reproduir cadascun dels escenaris, s'ha de carregar el dispositiu wearable fins al 100% de la seva bateria i, cada cert temps, anotar quin és el nivell

restant. Es comprova la bateria cada 15 minuts fins que s'esgota. La mesura s'acaba quan queda un 5% de bateria, ja que llavors el dispositiu tanca processos per reduir el consum de bateria el màxim possible.

4.2. Resultats

Els resultats es veuen millor si ens ajudem d'una gràfica per representar les mostres i agafant un punt en comú entre tots els resultats, per això es fixa l'eix de les X a temps = 0:00h. Després podem veure com es va consumint la bateria del dispositiu en cada una de les proves i al llarg de les hores.

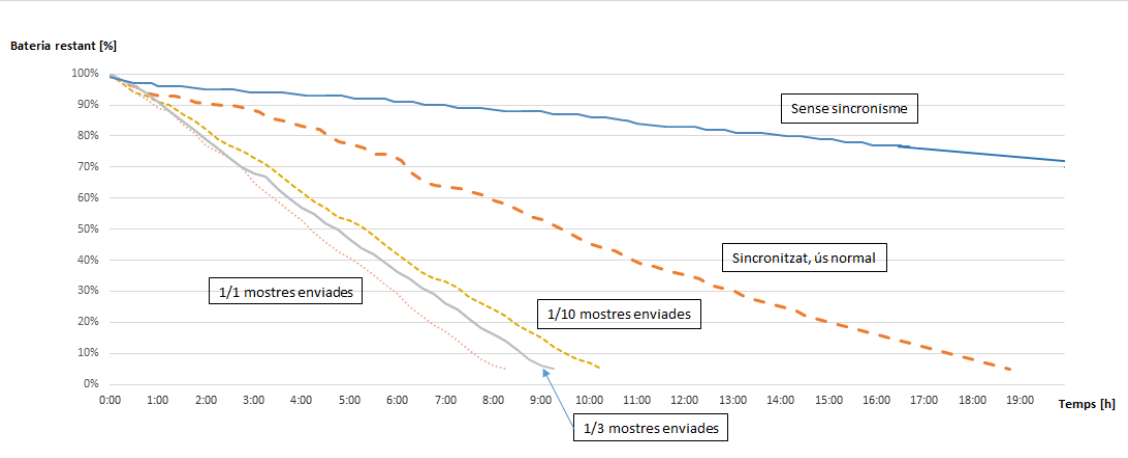

Fig.4.2.1 Resultats proves realitzades

Com es pot observar en la Figura 4.2.1, hi ha uns escenaris en els quals el dispositiu ha esgotat la bateria abans de les 12 hores de funcionament, mentre que en altres casos la bateria inclús podria aguantar alguns dies, veient la seva progressió.

En el primer cas on el wearable i mòbil no estan sincronitzats, es pot apreciar com el consum de bateria es aproximadament d'un 2% per hora. Aquest fet ens indica que podríem estar més de dos dies sense carregar el dispositiu.

En el cas corresponent a tenir els dispositius sincronitzats i enviant notificacions de les aplicacions desitjades, el que anomenem ús realista, veiem com hi ha un increment del consum i, aproximant a una recta, podríem dir que té un consum d'un 5% per hora.

Els tres escenaris restants tenen un comportament lleugerament diferent. Quan es transmeten per Bluetooth totes les mostres de ritme cardíac tenim un consum d'un 12% per hora, mentre que quan enviem una de cada deu mostres aquest és d'un 9% per hora, quedant el cas quan s'envia una de cada tres mostres entremig de les dues amb un consum aproximat d'un 10-11% per hora.

Quines conclusions extraiem, llavors, de les proves que hem realitzat?

Les proves mostren com, quan es fa l'ús realista del wearable, el dispositiu és capaç de romandre més temps obert amb una diferència notable respecte a quan l'aplicació està mesurant i enviant dades. Podem dir, llavors, que el consum del dispositiu amb l'aplicació funcionant respecte a quan l'aplicació no està funcionant, seria aproximadament d'entre un 4% i un 7% de bateria cada hora depenent de si la prova envia totes les mostres per canal Bluetooth o només algunes de les mostres.

Si ens fixem en la relació entre els tres casos en els que es vol mirar el consum que comporta l'enviament de les dades del sensor a través del canal Bluetooth, ens donem compte de que realment sí que afecta el fet d'enviar menys mostres.

Concretament, observem que enviar totes les dades implica un consum de bateria d'un 3% més cada hora respecte a enviar una de cada deu mostres, o d'entre un 1,5% i un 2% més si s'envien una de cada 3 mostres.

CAPÍTOL 5. CONCLUSIONS

En el començament d'aquest projecte, es va voler fer un estudi sobre el desenvolupament d'aplicacions per a dispositius wearable que comportés un enviament de dades d'aquest dispositiu a un altre.

Això implicava fer dues aplicacions, una aplicació per al dispositiu wearable, on s'haurien de prendre mostres del sensor i enviar-les pel canal Bluetooth, i una aplicació per al mòbil que estigués rebent les mostres i que les mostrés també per pantalla. S'havia de tenir en compte tot el necessari per desenvolupar-la.

Evidentment, es necessitava un dispositiu wearable que tingués algun tipus de sensor, preferiblement un sensor de freqüència cardíaca, amb capacitat per comunicar-se amb altres dispositius per mitjà del canal Bluetooth.

Per això es va mirar en el mercat i es va veure, dins de l'àmplia gamma de dispositius, quin ens podia ser de més utilitat en el nostre cas. Veient els diferents dispositius wearables es va comprovar com hi havia una part d'aquests dispositius que utilitzaven llenguatges de programació i sistemes operatius propis i sota llicència que feien pensar que no seria l'opció més fàcil.

D'entre tots aquests dispositius que utilitzaven diferents sistemes operatius es va pensar que la millor opció seria programar en un llenguatge conegut i vam acabar elegint un dispositiu amb el sistema Android Wear integrat que, a part, és el més utilitzat en el mercat.

Es va elegir utilitzar el dispositiu Samsung Gear Live per que portava sistema operatiu Android Wear integrat, que compta amb Bluetooth 4.0, i per que el dispositiu comptava amb un sensor òptic integrat per mesurar la freqüència cardíaca.

Amb el dispositiu wearable, i els coneixements sobre Android per al desenvolupament d'aplicacions per a dispositius mòbils adquirits durant la carrera, ja es podia començar a fer l'aplicació que havia d'utilitzar el wearable per obtenir les mesures del sensor i enviar-les.

En aquest punt és on més aprenentatge s'adquireix fent aquest projecte. La majoria del treball que comporta aquest projecte, és degut al desenvolupament de les aplicacions.

Aprendre com integrar les llibreries d'Android en el nostre projecte així com aprendre el correcte ús d'aquestes ha aportat un coneixement global sobre aplicacions Android i, més concretament, sobre les aplicacions Android Wear que s'integren en els dispositius wearables i la seva comunicació amb altres dispositius.

Per això es podria dir que una de les conclusions d'aquest projecte és que amb una bona recerca d'informació sobre les aplicacions Android, uns coneixements

previs bastant bàsics sobre llenguatges i entorns de programació i esforç per obtenir resultats es pot portar a terme el desenvolupament d'una aplicació Android que es comuniqui amb altres dispositius i faci les funcions que ens proposem sense problemes.

A part d'aquesta primera conclusió, amb l'aplicació Android que s'ha desenvolupat, s'ha pogut fer un estudi de la bateria que comporta per al dispositiu wearable el fet d'estar mesurant contínuament amb un sensor, en aquest cas de freqüència cardíaca, i el consum de bateria que comporta l'enviament de dades a través del canal Bluetooth entre les nostres aplicacions.

Amb les proves realitzades s'ha arribat a la conclusió que l'enviament de dades a través del canal Bluetooth entre un dispositiu i l'altre implica un consum de bateria que, possiblement en aquest cas no acabi de ser determinant, pot ser decisiu per l'autonomia del dispositiu i com, amb tècniques d'enviament de dades, com per exemple enviar una de cada 10 mostres, es pot arribar a reduir aquest consum sense implicar un elevat cost computacional.

Un cop arribats amb aquesta conclusió, i pensant en millorar aquest projecte, seria interessant observar que passa amb el consum de bateria utilitzant altres tecnologies d'enviament de dades com per exemple Bluetooth Low Energy.

ANNEX

En aquest annex, es proporciona una guia que pot seguir el lector per preparar l'entorn de desenvolupament d'una aplicació Android, tant per a dispositius mòbils com per a wearables, des del principi.

A continuació es descriuen els passos que s'han de seguir, abans de començar a desenvolupar una aplicació d'aquest tipus, amb l'objectiu de preparar l'entorn de programació en el nostre ordinador. Per això ens hem basat principalment en la web de desenvolupadors que ens proporciona Android [].

A.1. Nou projecte

En primer lloc s'ha de descarregar l'entorn de desenvolupament integrat. En aquest projecte s'ha utilitzat Android Studio que es pot descarregar gratuïtament de la web de Android Studio [].

Un cop descarregat i instal·lat, començarem creant un nou projecte que disposi de dos mòduls o aplicacions diferenciades, ja que una ha d'anar al mòbil i l'altra al rellotge.

Per això, s'ha d'obrir l'Android Studio, anar a File→New Project. Com veiem en la figura A.1, podem triar el nom de l'Activity principal i indicar per a quines plataformes es vol fer el projecte. S'indica l'opció *Phone and Tablet* i *Wear*, i s'elegeix l'opció 'Blank Activity' per les dues.

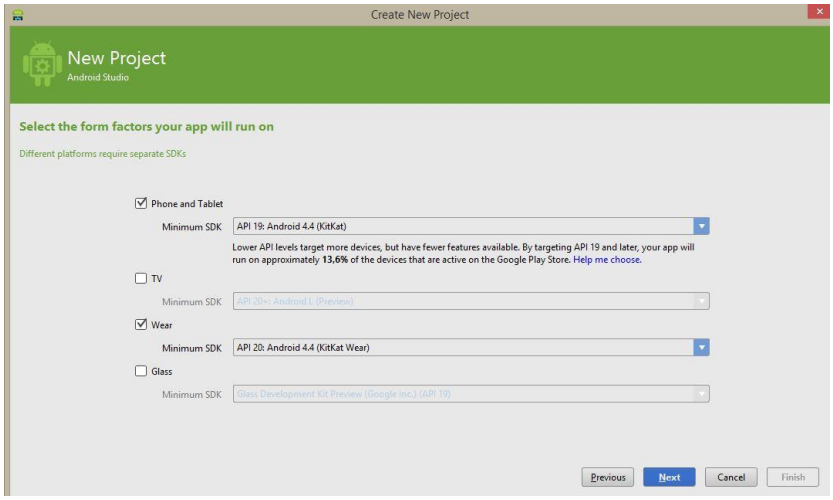

Fig.A.1 Creació nou projecte per a mòbil i wearable

A.2. SDK

El següent pas consisteix en descarregar-se les llibreries del SDK d'Android. S'instal·la l'API 20 per l'aplicació del wearable, i la API 19 d'Android per l'aplicació del mòbil. En el menú de l'Android Studio accedim a

Tools→Android→SDK Manager. Descarreguem les llibreries que ens interessen, veure figura A.2.

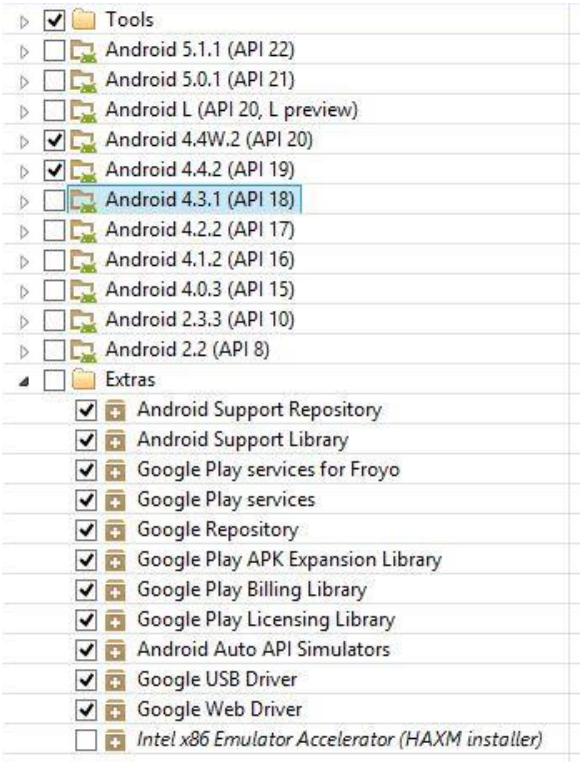

Fig.A.2 Diferents SDK i eines a descarregar

A part de les API d'Android amb les que es vol treballar, s'han d'instal·lar les eines que faciliten el desenvolupament, per això cal instal·lar totes les llibreries que estan dins de la carpeta Tools i les de la carpeta Extras, com es pot veure a la Figura A.4. Les corresponents a la carpeta Extra ens permetran utilitzar Google Play Services [6] a més d'oferir-nos ajudes pel desenvolupament.

A.3. Build.gradle

Anem a veure com incloure aquestes llibreries instal·lades en el projecte. Per això cal accedir als fitxers *build.gradle* que es generen quan creem el nou projecte.

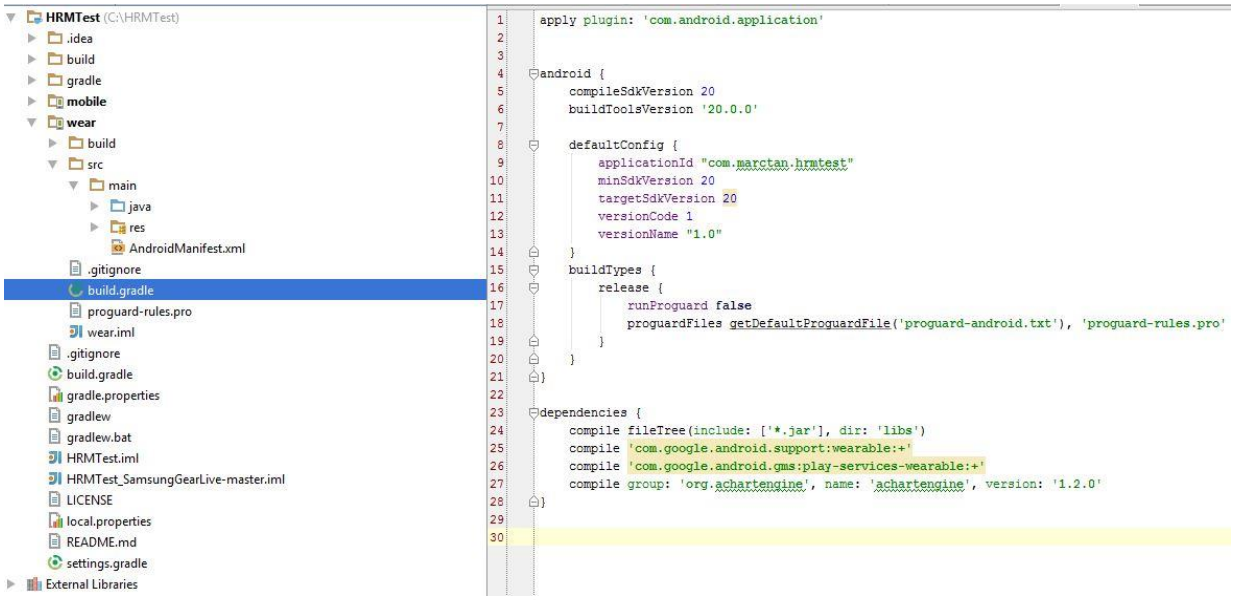

Fig.A.3 Declaració SDK en el *build.gradle*

En aquest fitxer *build.gradle*, s'especifiquen les llibreries a les que ha de fer referència el mòdul que estem definint, com es pot veure a la Figura A.3. S'ha d'indicar quin SDK es vol utilitzar en el mòdul, en el cas del wearable el 20, i després la ruta on es troba l'aplicació i quina llibreria SDK utilitza aquesta.

També s'ha d'indicar en l'apartat de *dependencies* les diferents llibreries conforme les anem utilitzant en l'aplicació. Així, s'ha d'incloure aquí la llibreria de Achartengine que s'utilitza en l'aplicació per fer gràfics, com s'indica en el capítol 3.5 de la memòria, i les llibreries per les aplicacions que utilitzin Google Services.

A.4. AndroidManifest

Un cop s'han creat els dos mòduls, i s'han inclòs les llibreries necessàries, hem d'especificar quins són aquests dos mòduls en el fitxer AndroidManifest, així com els permisos amb els que es vol dotar a l'aplicació. En el cas del wearable, s'haurà d'indicar, en aquest arxiu Manifest, quina és l'Activity principal, així com els permisos per accedir al sensor del dispositiu i per comunicar-se amb altres dispositius mitjançant Bluetooth.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.marctan.hrmtest" >

    <uses-feature android:name="android.hardware.type.watch" />
    <uses-permission android:name="android.permission.BODY_SENSORS" />
    <!--OAD-->
    <uses-feature android:name="android.hardware.bluetooth_le" android:required="true"/>
    <uses-permission android:name="android.permission.BLUETOOTH"/>
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
    <!--OAD-->

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="HRMTest"
        android:theme="@android:style/Theme.DeviceDefault" >
        <activity
            android:name=".MyActivity"
            android:label="HRMTest" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <meta-data android:name="com.google.android.gms.version"
            android:value="6587000" />
        <service android:name=".ListenerService">
            <intent-filter>
                <action android:name="com.google.android.gms.wearable.BIND_LISTENER" />
            </intent-filter>
        </service>

    </application>
</manifest>
```

Fig.A.4 Arxiu AndroidManifest del wearable

En el cas del dispositiu mòbil, aquestes declaracions s'hauran de fer tal i com s'ha indicat pel mòdul del wearable, però sense incloure els permisos per obtenir dades del sensor.

A.5. Execució Aplicació

L'aplicació es pot testejar utilitzant els simuladors que proporciona Android Studio. Per crear un simulador de dispositiu wearable s'ha d'accedir a Tools→Android→AVD Manager. S'ha de crear un nou simulador i triar les seves característiques.

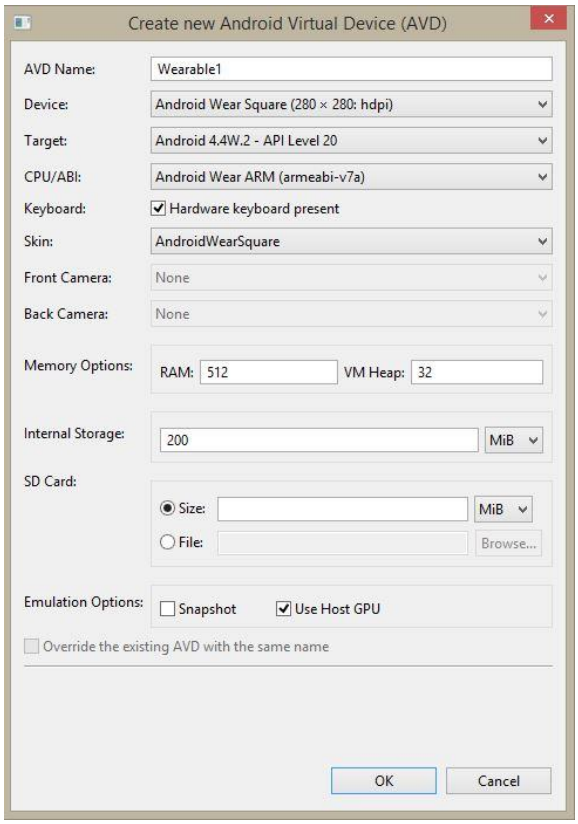

Fig.A.5.1 Opcions del simulador Android Wear

En aquesta Figura A.5 es veuen les opcions marcades que més s'escauen amb el dispositiu wearable en el que després s'ha d'instal·lar l'aplicació, i ens proporciona una aproximació bastant real del que després observarem en el wearable.

Evidentment, si es treballa amb una altra API, amb un altre dispositiu o es vol dotar al simulador amb més capacitat RAM, es poden canviar aquests valors. S'ha de tenir en compte que s'utilitza la memòria RAM del PC on s'executa el simulador.

Un cop acceptem les opcions es crearà aquest dispositiu simulat. Després, quan es vulgui testejar l'aplicació, es pot configurar l'execució de forma que es triï el simulador que s'ha creat. Per modificar les opcions d'execució s'ha d'anar, en la barra d'eines, a Run→Edit Configurations.

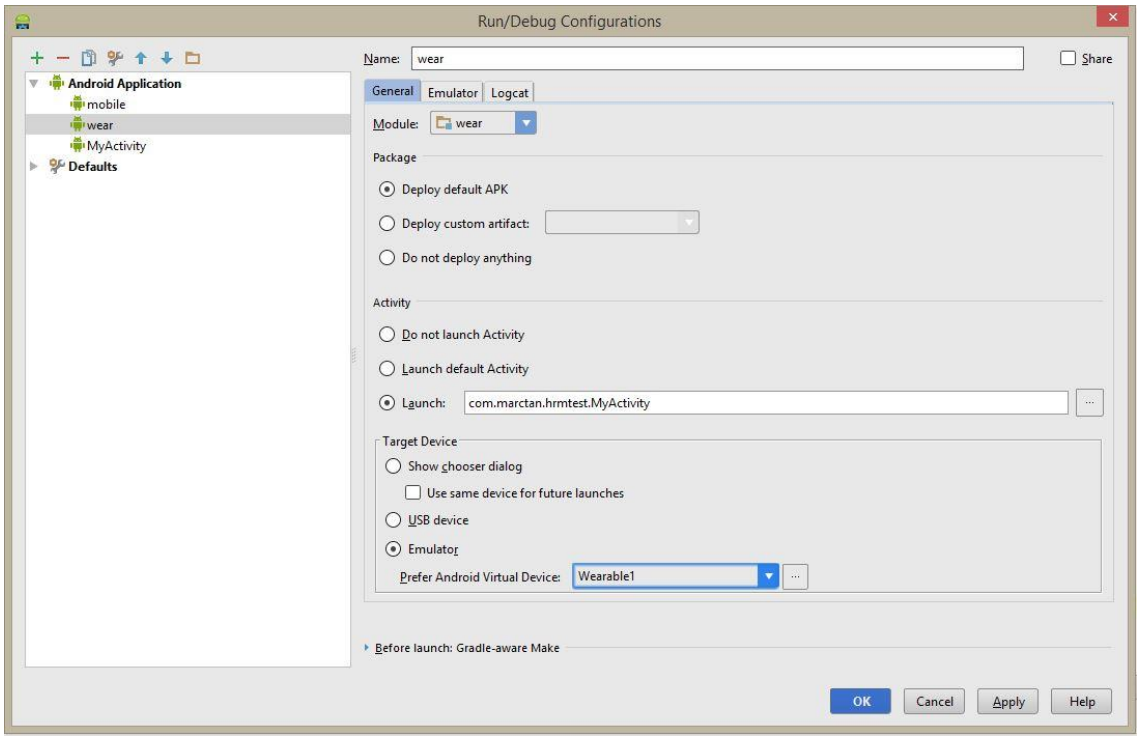

Fig.A.5.2 Opcions per executar l'aplicació

En la finestra que s'obre, Figura A.5.2 es poden triar diferents formes d'executar l'aplicació. Inicialment vam utilitzar l'emulador i després vam fer servir l'opció USB Device per instal·lar-la directament en el dispositiu wearable.

Executant l'aplicació directament per USB, es poden executar els dos mòduls de l'aplicació directament ens els dos dispositius. Això permet provar les aplicacions, com reaccionen a les accions de l'usuari i com interactuen entre elles.

A.6. Mode Debug

El següent pas, doncs, per comprovar el bon funcionament de l'aplicació, és testejar-la en els dispositius activant el mode de Debug tant en el dispositiu mòbil com en el wearable. Aquest mode es pot trobar en les opcions per a desenvolupadors que incorpora el sistema operatiu Android.

El mode Debug permet que el dispositiu descarregui i executi l'aplicació directament en el dispositiu objectiu. Mentre s'executa l'aplicació en el dispositiu amb el mode Debug funcionant, es poden visualitzar els missatges de l'aplicació amb el seu estat, així com els missatges d'error, en l'Android Project.

Per activar el mode Debug, en el cas d'Android Wear cal anar al menú de *settings* i seleccionar l'opció *About*, en la qual apareix informació sobre el dispositiu. És necessari prémer 7 vegades consecutives sobre el número de muntatge del dispositiu per a que aparegui un missatge conforme ja estem en el mode de desenvolupador. Llavors es pot activar el mode Debug anant a

opcions per a desenvolupadors i activant l'opció de Debug per Bluetooth, veure Figura A.6.1.

Fig.A.6.1 Opcions per a desenvolupador al wearable

En el cas del dispositiu mòbil fem un procés semblant. S'ha d'anar a *settings* accedir al número de muntatge i prémer 7 vegades fins que aparegui un missatge conforme ja estem en el mode de desenvolupador.

Per habilitar el pont Bluetooth entre el dispositiu mòbil i el wearable, s'ha d'utilitzar el ADB, com s'explica en la secció 1.2.4 de la memòria.

A.7. Android Wear App

Per connectar els dos dispositius des de l'aplicació Android Wear del mòbil, primer s'ha d'instal·lar l'aplicació Android Wear en el mòbil [13].

Un cop instal·lada s'ha d'obrir amb el wearable encès. Ens apareixeran unes instruccions d'ús de l'aplicació i seguidament, un llistat amb els dispositius wearables propers. Seleccionem, llavors, el *Samsung Gear Live*, el nom del dispositiu apareix en la pantalla principal del mateix, i vinculem els dos dispositius després d'introduir el codi de vinculació que es mostra ens els dos dispositius.

Apareixerà un missatge de confirmació, i ja tindrem els dos dispositius connectats, com es veu en la Figura A.7.1.

Fig. A.7.1 Aplicació Android Wear per a mòbils

Ara que estan emparellats, s'ha d'habilitar el wearable per a que mostri notificacions de les aplicacions que s'indiquin en l'aplicació Android Wear. Per això cal anar al menú d'opcions de l'aplicació (tres punts en vertical) i seleccionar l'opció que habilita les notificacions del telèfon en el wearable.

Si obrim el menú de Settings en l'aplicació Android Wear del mòbil (la roda dentada) podrem veure com apareix l'opció de depuració de Bluetooth. Quan el wearable i el mòbil estiguin connectats, apareixerà l'opció destí connectat mentre que l'opció Host estarà en desconnectat. Per connectar amb el Host, que serà l'ordinador on es desenvolupa l'aplicació, s'ha d'utilitzar l'ADB com s'explica en el capítol 1.2.4 d'aquesta memòria.

REFERÈNCIES

[1] "Worldwide Smartphone Growth Forecast to Slow from a Boil to a Simmer as Prices Drop and Markets Mature, According to IDC", FRAMINGHAM, Mass. December 1, 2014.

http://www.idc.com/getdoc.jsp?containerId=prUS25282214

[2] "Global smartphone sales from 2009 to 2014, by operating System (in milions)", Statista, the statistics portal. 2015

http://www.statista.com/statistics/271994/smartphone-users-by-operating-system-and-device-in-the-us/

[3] "Comparativa: todos los smartwatches con Android Wear, enfrentados", ElAndroidLibre, Redeslibre Networks S.L., 2015

http://www.elandroidelibre.com/2014/09/comparativa-todos-los-smartwatches-con-android-wear-enfrentados.html

[4] "Android Wear Application for Android devices downloader", Google Inc, 18 de mayo de 2015

https://play.google.com/store/apps/details?id=com.google.android.wearable.app&hl=es

[5] "Market Share, Mobile operating System", Wikipedia, the free encyclopedia.

http://en.wikipedia.org/wiki/Mobile_operating_system

[6] "Setting Up Google Play Services", Android Open Source Project, 2015.

http://developer.android.com/google/play-services/setup.html

[7] "Developing tools for Android Apps", Android Open Source Project, 2015.

https://developer.android.com/sdk/index.html

[8] "Getting Started, Android Training" Android Open Source Project, 2015.

http://developer.android.com/training/index.html

[9] “Building Apps for Wearables”, Android Open Source Project, 2015.

http://developer.android.com/training/building-wearables.html

[10] “Stack Overflow”, Stack Exchange, Inc., 2015.

http://stackoverflow.com

[11] “Git Respository hosting service”, GitHub Inc, 2015.

https://github.com

[12] “The Samsung Gear Live Review”, Jerry Hildenbrand, July 08 2014.

http://www.androidcentral.com/samsung-gear-live-review

[13] “Cómo vincular el reloj con el teléfono”, Google Inc, 2015.

https://support.google.com/androidwear/answer/6056630?hl=es