UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Development of a Client-Server mobile App to collect biomedical data compatible with OpenMRS

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Enric Artús Suarez

In partial fulfilment

of the requirements for the degree in

Telecommunications Technologies and Services ENGINEERING

Advisor: Pere Joan Riu Costa

Barcelona, January 2020

Abstract

In this TFG it was intended to update and improve the performance of the client-server app based on OpenMRS, that is used in the BPUP project. However, for a request done by the doctor associated to the project, it was decided to change the objective to include the functionality of communication with a Bluetooth scale in order to collect more data about the patient.

To do so, reverse engineering was done to a Bluetooth scale to see and analyse its performance and the codification that was used to do the data sending. An application capable of connecting and receiving data from the Bluetooth scale, but when the merging and integration of the client-server was done, appeared an execution problem which hasn't been solved yet.

Resum

En aquest Treball de Fi de Grau es pretenia actualitzar i millorar l'aplicació client-servidor basada en OpenMRS utilitzada pel projecte BPUP. Malgrat tot, per una petició del metge associat al projecte, es va decidir canviar l'objectiu pel d'incloure la funcionalitat de comunicar-se amb una bàscula Bluetooth per recopilar més dades del pacient.

Per a fer-ho es va fer enginyeria inversa a una bàscula Bluetooth per tal de veure'n el funcionament i la codificació que utilitzava per realitzar l'enviament de dades. Es va desenvolupar una aplicació capaç de connectar-se i rebre dades de la bàscula però quan es va dur a terme la integració amb l'aplicació client-servidor, va sorgir un problema d'execució pel qual no s'ha trobat encara una solució.

Resumen

En este Trabajo de Fin de Grado se pretendía actualizar y mejorar la aplicación cliente-servidor basada en OpenMRS que utilizaba el proyecto BPUP. Sin embargo, por una petición del médico asociado al proyecto, se decidió cambiar el objetivo por el de incluir la funcionalidad de comunicar-se con una báscula Bluetooth para recopilar más datos del paciente.

Para hacerlo se realizó ingeniería inversa a una báscula Bluetooth para ver cómo funcionaba y que codificación utilizaba para el envío de los datos. Se desarrolló una aplicación capaz de conectarse y recibir datos de la báscula, pero cuando se llevó a cabo la integración con la aplicación cliente-servidor, surgió un problema de ejecución para el cual aún no se ha encontrado solución.

A la gent que m'ha hagut d'aguantar en aquest llarg viatge.

Agraïments

En primer lloc, m'agradaria agrair al meu tutor, Pere Riu, i a l'Eva Vidal, per donar-me la oportunitat de treballar en aquest projecte, que espero continuï durant uns anys més.

Agrair també als amics que m'han acompanyat durant tota la carrera, que m'han donat suport en els moments durs i han fet més memorables els moments bons.

En especial, vull agrair al meu company Sergio Gimenez, que em va proposar de treballar en el projecte de *Blood Pressure Under Pressure* i m'ha acompanyat i ajudat durant el transcurs d'aquest treball.

Finalment agrair-te a tu, el lector, per prendre't la molèstia de llegir aquest treball. Sense tu, haver escrit aquest treball no tindria cap sentit.

Historial de revisió i registre d'aprovacions

Revision	Date	Purpose
0	18/01/2020	Document creation
1	24/01/2020	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Enric Artús Suarez	enricarsu@gmail.com
Pere Joan Riu Costa	pere.riu@gmail.com

Written by:		Reviewed and approved by:	
Date	18/01/2020	Date	24/01/2020
Name	Enric Artús Suarez	Name	Pere Joan Riu Costa
Position	Project Author	Position	Project Supervisor

Índex

Llista de figures

Llista de Taules

1. Introducció

1.1. Situació actual

La hipertensió i els problemes i malalties cardiovasculars son un dels riscs més importants de la societat actual. Com que no té símptomes aparentment visibles en les fases inicials, la gent no sap que estan afectats per aquestes malalties. Últimament en països desenvolupats comença a no ser un problema tant greu, ja que ara ja es fan regularment controls a les visites mèdiques tot i que no s'hagi acudit al metge per aquest motiu. Així doncs, aquest projecte va començar perquè en els països del tercer món on no s'està al corrent d'aquests problemes, hi hagués una forma més senzilla i econòmica per començar a prevenir aquestes malalties.

Aquest, és la continuació d'un projecte ja començat pel grup del Dr. Pere Riu Costa, que forma part del projecte *Blood Pressure Under Pressure,* on es va desenvolupar una aplicació per mesurar la tensió mitjançant un tensiòmetre Bluetooth i una aplicació mèdica de software lliure anomenada OpenMRS.

1.2. Objectius inicials

L'objectiu inicial d'aquest projecte era corregir i millorar l'aplicació que ja es tenia, que utilitzava la versió 2.6 de l'app client-servidor OpenMRS, i actualitzar-la a la versió més recent (2.8), que corregia la majoria de *bugs* o errors. Aquesta actualització es realitzaria traslladant els mòduls Bluetooth i adaptant-los a la nova versió, millorant així la senzillesa amb la que gent no habituada a utilitzar aquestes aplicacions, com els habitants de països no tant desenvolupats, podia realitzar les mesures sense problemes.

1.3. Requeriments i especificacions

Aquest projecte requereix:

- Una manera de comunicar un dispositiu Bluetooth, en el cas del projecte un tensiòmetre, amb l'aplicació.
- Una manera de transformar el format utilitzat pel Bluetooth en un format OpenMRS.
- Una manera de penjar la nostra base de dades de pacients al núvol per donar aquesta informació als doctors per tal de diagnosticar als pacients. Aquesta base de dades ha de ser anònima, per tal de complir la llei de protecció de de dades sensibles.

Les especificacions que necessita aquest projecte son:

- Un telèfon mòbil intel·ligent amb sistema operatiu Android que disposi de connexió Bluetooth i internet.
- Un tensiòmetre Bluetooth
- Un servidor per la base de dades
- Un ordinador amb un programa capaç de programar i compilar un programa Android a un telèfon mòbil intel·ligent.

1.4. Mètodes i procediments

Per dur a terme el treball, es pretenia seguir la següent metodologia:

1. Estudi i anàlisi de l'anterior aplicació del projecte. Fent èmfasi en l'apartat Bluetooth que era el que s'havia de traslladar a la nova versió.
2. Implementació del sistema per suportar novament la configuració en la nova versió de l'app OpenMRS.
3. Provar el funcionament de l'app i corregir possibles errors o *bugs*.

Per realitzar-lo, així doncs, s'ha utilitzat el software de lliure distribució OpenMRS com a base, i el software desenvolupat anteriorment en el projecte BPUP per a continuar amb la seva feina i millorar el projecte.

1.5. Desviacions inicials

Durant el transcurs del projecte, el metge associat va demanar si també es podia afegir una bàscula a l'aplicació, paral·lelament al tensiòmetre. Per a actualitzar l'app es va considerar que primer s'havia d'afegir la bàscula Bluetooth. Es va comprar una bàscula de la mateixa marca que el tensiòmetre, pensant que utilitzarien la mateixa codificació i que es podria reutilitzar el codi. Tot i això, la bàscula va resultar utilitzar una altra codificació de dades i la implementació de la bàscula es va endur tots els esforços del projecte, canviant així l'objectiu de readaptar el codi a la nova versió d'OpenMRS, a realitzar una implementació correcta per comunicar la bàscula Bluetooth.

1.6. Planificació

- **Pla de treball**

Com s'ha comentat anteriorment, a la meitat del projecte es van haver de reordenar les prioritats i es va canviar l'objectiu principal del projecte. Per tal de dur-lo a terme es va generar aquest nou pla de treball.

Project: Establir connexió Bluetooth amb la bàscula	WP ref: 1	
Major constituent: Connexió Bluetooth	Sheet 1 of 1	
Short description: Aquest paquet de treball consisteix en la implementació del mòdul Bluetooth que es connectarà amb el dispositiu, en el nostre cas, la bàscula.	Planned start date: 15/11/2019 Planned end date: 17/12/2019	
	Start event:T1 End event:T4	
Internal task T1: App implementation Internal task T2: App testing Internal task T3: Bug and error fixing Internal task T4: App documentation	Deliverables: --	Dates: --

Project: Integració de les dues aplicacions	WP ref: 2	
Major constituent: Software	Sheet 1 of 1	
Short description: Aquest paquet de treball consisteix en la fusió de de l'aplicació original del projecte i la nova app de connexió amb la bàscula	Planned start date: 17/12/2019 Planned end date: 20/01/2020	
	Start event: T1 End event: T4	
Internal task T1: App merging Internal task T2: App testing Internal task T3: Bug and error fixing Internal task T4: App documentation	Deliverables: --	Dates: --

- **Diagrama Gantt**

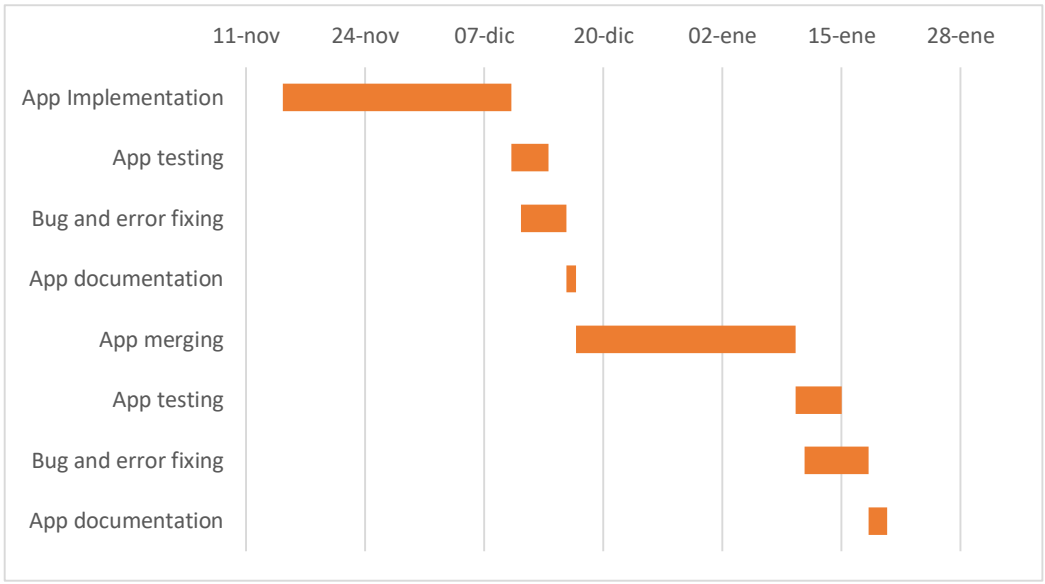

Figura 1: Diagrama Gantt

2. Estat de l'art

2.1. Continuació al projecte 'Sistemes autònoms de baix cost de suport a projectes educatius i de salut'

Aquest projecte neix com a continuació d'un altre projecte també realitzat en aquesta escola per l'Albert Sanchez.

El projecte anterior es pretenia demostrar la viabilitat de realitzar una implementació d'un software de gestió mèdica, a través d'un 'Single-Board Computer'. El sistema també comptava amb un dimensionat de sistema d'alimentació autònom basat en plaques fotovoltaiques i bateries per alimentar la Raspberry pi utilitzada i facilitar-ne el desplaçament.

Tant aquest projecte, com el projecte de l'Albert, pertanyen al projecte de *Blood Pressure Under Pressure*, un projecte social dedicat a conscienciar i prevenir els riscos cardiovasculars, en especial la hipertensió, que tot i que és una malaltia que es pot tractar adequadament segueix essent una de les principals causes de mortalitat mundial. Per aconseguir-ho s'intenta portar als països que no tenen els recursos per lluitar contra la malaltia una possibilitat de detectar-lo i prevenir-lo.

2.2. OpenMRS

OpenMRS és una comunitat que treballa per construir la plataforma electrònica d'emmagatzematge de dades mèdiques, en format de software de lliure distribució, del món. Són una comunitat de voluntaris d'arreu del planeta amb diferents formacions, tant en el sector de desenvolupament tecnològic com en el sector mèdic.

OpenMRS desenvolupa una plataforma de software lliure versàtil, que permet ser una base per a desenvolupament d'aplicacions web. També disposa de 'llibreries' que poden ser incorporades, permetent a l'usuari afegir les característiques i funcionalitats a l'aplicació segons el que necessiti.

És un software construït a partir de Java i MySQL (un software de bases de dades), pensat per a ser una eina a la comunitat mèdica arreu del món per oferir un servei millor.

2.3. Bluetooth Low Energy (BLE)

Bluetooth Low Energy és una forma ecològica del Bluetooth que s'ha desenvolupat recentment per a facilitar el futur *Internet of Things*. És important en aquest àmbit perquè és mes eficient energèticament que el Bluetooth clàssic o el Wi-Fi, i per tant millora la duració de les bateries dels dispositius utilitzats. A més la seva baixa velocitat de dades el

fa adequat per a casos on només s'ha de transmetre dades d'estat com en sensors, o en el cas que ens interessa, material mèdic.

2.4. Bàscules Bluetooth

Actualment ja son moltes les marques del sector que desenvolupen productes amb capacitat Bluetooth que permeten recollir les dades amb Bluetooth. Tot i així, no serveix només amb la bàscula, cada marca ha implementat la seva pròpia aplicació per a comunicar-se amb la seva bàscula, i per tal d'assegurar-se que el client utilitzarà la seva app, utilitzen protocols i xifrats diferents per transmetre la informació. A més a més, hi ha aplicacions que també requereixen d'una subscripció de pagament extra.

Cal remarcar també que, a més, aquestes aplicacions estan pensades per un ambient més domèstic i per tant, no estan preparades per a gestionar una base de dades com les que podria necessitar un hospital o un centre mèdic.

3. Metodologia i desenvolupament del projecte

3.1. Adquirir nocions d'Android i estudi del codi del projecte

Si bé la part del *back-end* de l'aplicació està programat en Java, per a programar la interfície gràfica calia editar fitxers .xml en el format adequat.

Per a poder programar es va utilitzar l'última versió de l'editor Android Studio, actualment la 3.5.3. Android Studio és l'IDE (*Integrated Development Environment*) oficial per al desenvolupament d'aplicacions Android. Ofereix un entorn flexible i unificat on pots desenvolupar per a tots els dispositius. Es pot vincular amb els dipòsits de GitHub. També permet compilar i depurar els programes sense requerir eines externes. A més també permet carregar l'aplicació al telèfon mòbil o en el cas de necessitar-ho, emular un telèfon mòbil a l'ordinador.

Per tot això, es va decidir utilitzar aquest software de desenvolupament. Per tal d'adquirir el coneixements necessaris per a desenvolupar una app i fer la posterior integració, es va recórrer a tutorials d'iniciació d'internet.

Un cop acabada la introducció a Android, es va procedir a revisar el codi de l'aplicació del projecte. El projecte constava de diversos mòduls per gestionar totes les funcionalitats amb les que ja comptava. Es va estudiar i depurar el mòdul Bluetooth per tal d'entendre el seu funcionament i adaptar-lo així a la versió més recent d'openMRS.

3.2. Addició d'una bàscula Bluetooth a l'aplicació.

Paral·lelament, el metge associat al projecte va demanar en una de les reunions si es podia afegir una bàscula Bluetooth de la mateixa manera que s'havia afegit el tensiòmetre. Es va comprar una bàscula Bluetooth marca Thompson igual que el tensiòmetre que ja s'havia connectat.

Es va intentar utilitzar el mateix mòdul utilitzat anteriorment però no va resultar.

3.3. Enginyeria inversa a la bàscula Thompson i anàlisi dels paquets

Investigant per internet es va trobar una *wiki* de GitHub, on el desenvolupador *Oliexdev* explica com realitzar enginyeria inversa a una bàscula que utilitzi el protocol Bluetooth 4.x utilitzant la opció de desenvolupador del registre de cerca HCI (*Host Controller Interface*) Bluetooth. Aquesta opció permet obtenir un registre anomenat 'btsnoop_hci.log' que conté les transmissions Bluetooth que s'han realitzat.

Seguint les instruccions, per obtenir un registre dels paquets que s'enviaven va caler esborrar els registres de comunicacions anteriors i activar de nou l'opció. Llavors es va realitzar una mesura amb la bàscula Thompson utilitzant l'aplicació pròpia de la bàscula i es van anotar totes les dades rellevants. Un cop acabada la comunicació, es va desactivar l'opció del registre per evitar possibles paquets no desitjats i es va guardar el fitxer a l'ordinador. Es van realitzar varies mesures seguint aquest procediment per obtenir vàries mostres i poder desxifrar els paquets.

A continuació, es va procedir a obtenir les característiques i el servei Bluetooth de la bàscula. Per això, es va utilitzar la pròpia aplicació de desenvolupament *d'Oliexdev*, anomenada *OpenScale*.

OpenScale és una aplicació de software de lliure distribució dedicada al seguiment del pes i altres mètriques corporals, que permet també connectar-s'hi amb diverses bàscules Bluetooth.

Utilitzant la opció d'activar el registre de depuració i seguidament buscant la bàscula Thompson.

Un cop trobada la bàscula, calia connectar-s'hi amb l'app de desenvolupament per tal de que pogués buscar i agafar la informació sobre els serveis i característiques. Després calia desactivar l'opció de registre de l'aplicació i guardar el nou registre associat amb el registre obtingut anteriorment amb el HCI.

```
2019-11-26 16:55:11.491 Debug [1] AboutPreferences: Debug log enabled, openScale (dev) v2.1.2-dev_49fc3aca_2019-10-20 (1571577752), SDK 25, Sony E6653
2019-11-26 16:55:11.493 Debug [1] AboutPreferences: Selected user id(1) name(mendes ) birthday(Wed Nov 10 00:00:00 GMT+01:00 1999) age(20) body height(176,00) scale unit(kg) gender(male
initial weight(63,00) goal weight(70,00) goal date(Tue May 26 16:53:40 GMT+02:00 2020) measure unt(cm) activity level(0)
2019-11-26 16:55:41.343 Info [1] BluetoothCentral: scan started
2019-11-26 16:55:46.198 Debug [1] BluetoothPreferences: Found unsupported device 1106A0 [CA:DC:A2:0B:C9:BA]
2019-11-26 16:55:53.650 Info [1] BluetoothCentral: scan stopped
2019-11-26 16:55:53.652 Debug [1] OpenScale: Trying to connect to bluetooth device [CA:DC:A2:0B:C9:BA] in debug mode
2019-11-26 16:55:53.656 Debug [1] BluetoothCommunication: Do LE scan before connecting to device
2019-11-26 16:55:53.672 Info [1] BluetoothCentral: scan started
2019-11-26 16:55:53.673 Debug [1] BluetoothCommunication: Stop machine state
2019-11-26 16:55:53.785 Debug [1] BluetoothCommunication: Found peripheral '1106A0'
2019-11-26 16:55:53.798 Info [1] BluetoothCentral: scan stopped
2019-11-26 16:55:54.802 Debug [1] BluetoothCommunication: Try to connect to BLE device CA:DC:A2:0B:C9:BA
2019-11-26 16:55:54.921 Info [1] BluetoothPeripheral: connect to '1106A0' (CA:DC:A2:0B:C9:BA) using TRANSPORT_LE
2019-11-26 16:55:56.030 Info [831] BluetoothPeripheral: connected to '1106A0' (BOND_NONE) in 1,1s
2019-11-26 16:55:56.040 Debug [1] BluetoothPeripheral: discovering services of '1106A0' with delay of 0 ms
2019-11-26 16:55:57.230 Info [830] BluetoothPeripheral: discovered 4 services for '1106A0'
2019-11-26 16:55:57.234 Debug [1] BluetoothCommunication: connected to '1106A0'
2019-11-26 16:55:57.236 Debug [1] BluetoothCommunication: Successful Bluetooth services discovered
2019-11-26 16:55:57.246 Debug [1] BluetoothCommunication: Invoke read bytes on 0x2a00 "device name"
2019-11-26 16:55:57.249 Debug [1] BluetoothDebug: Service 0x1800 "generic access"
2019-11-26 16:55:57.250 Debug [1] BluetoothDebug: |- characteristic 0x2a00 "device name" (#3): WRITE, WRITE_NO_RESPONSE', READ
2019-11-26 16:55:57.251 Debug [1] BluetoothDebug: |- characteristic 0x2a01 "appearance" (#5): READ
2019-11-26 16:55:57.252 Debug [1] BluetoothDebug: |- characteristic 0x2a04 "peripheral preferred connection parameters" (#7): READ
2019-11-26 16:55:57.253 Debug [1] BluetoothDebug: Service 0x1801 "generic attribute"
2019-11-26 16:55:57.254 Debug [1] BluetoothDebug: Service 0x180a "device information"
2019-11-26 16:55:57.254 Debug [1] BluetoothDebug: |- characteristic 0x2a25 "serial number string" (#11): READ
2019-11-26 16:55:57.255 Debug [1] BluetoothDebug: |--- descriptor 0x2904
2019-11-26 16:55:57.256 Debug [1] BluetoothDebug: |- characteristic 0x2a27 "hardware revision string" (#14): READ
2019-11-26 16:55:57.257 Debug [1] BluetoothDebug: |--- descriptor 0x2904
2019-11-26 16:55:57.258 Debug [1] BluetoothDebug: |- characteristic 0x2a26 "firmware revision string" (#17): READ
2019-11-26 16:55:57.258 Debug [1] BluetoothDebug: |--- descriptor 0x2904
2019-11-26 16:55:57.259 Debug [1] BluetoothDebug: |- characteristic 0x2a29 "manufacturer name string" (#20): READ
2019-11-26 16:55:57.260 Debug [1] BluetoothDebug: |--- descriptor 0x2904
2019-11-26 16:55:57.261 Debug [1] BluetoothDebug: |- characteristic 0x2a28 "software revision string" (#23): READ
2019-11-26 16:55:57.262 Debug [1] BluetoothDebug: |--- descriptor 0x2904
```

Figura 2: Informació sobre els serveis i característiques.

Finalment, per analitzar el registre dels paquets es va utilitzar l'analitzador de paquets *Wireshark*. Aplicant un filtre utilitzant la direcció Bluetooth de la bàscula es va aconseguir veure exclusivament els paquets resultants de la connexió desitjada, eliminant els paquets on el telèfon mòbil cercava els possibles dispositius.

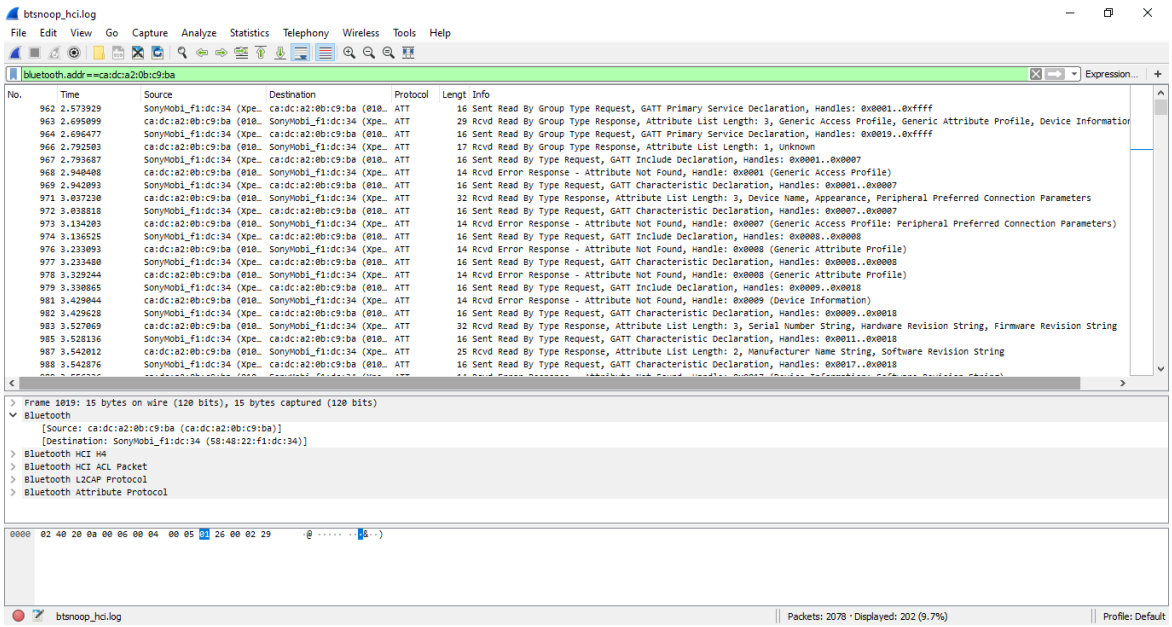

Figura 3: Filtrat dels paquets Bluetooth.

Per tal de desxifrar els paquets es va buscar el pes que havia guardat l'aplicació Thompson i fent al conversió a hexadecimal en format *little endian* es van filtrar els paquets que continguessin aquella informació. Es van obtenir diversos paquets amb aquells bytes. També es va filtrar el paquets amb el pes convertit en hexadecimal en format *big endian* però no es va trobar cap resultat, assumint així que la bàscula utilitzava codificació *little endian.*

Seguidament es va procedir a buscar l'altra informació rellevant anotada, com la UUID de la bàscula o la data i l'hora de la mesura en aquest cas sense èxit, ja que per exemple, és impossible saber el format que utilitza la bàscula per a codificar la data.

També es va provar de trobar els paquets de missatges previs a l'enviament del paquet que contenia el pes, per trobar els paquets que contenen les peticions cap a la bàscula per tal de que aquesta retorni el pes, també sense èxit.

3.4. Estudi de l'aplicació OpenScale i depuració.

Després de repetits intents sense èxit per tal de trobar tota la informació necessària, i en vista de que s'estava dedicant excessiu temps a l'anàlisi dels paquets, es va decidir no utilitzar la bàscula Thompson i utilitzar una de les bàscules a les quals *Oliexdev* ja havia fet exitosament l'enginyeria inversa. Per això es va comprar una bàscula marca Medisana model BS444 per a continuar amb el projecte.

Després de comprovar que la bàscula funcionava correctament amb l'aplicació, es va començar a estudiar el funcionament d'OpenScale. L'app disposa de moltes funcionalitats a part de rebre les dades de la bàscula Bluetooth, tals com disposar de diversos usuaris per tenir constància de les diferents situacions, la funció d'introduir manualment les mètriques com alçada, mesura del pit i cintura i d'altres, ja que la bàscula no les proveeix, per tal de calcular algunes constants de l'usuari com l'Índex de Massa Corporal (IMC) o el greix corporal.

Per tal d'estudiar l'aplicació OpenScale, es va depurar l'aplicació mentre s'utilitzava la funcionalitat Bluetooth que duia implementada per veure el camí que seguia l'execució del procés de cerca del dispositiu i l'enviament de les dades per part de la bàscula al telèfon mòbil.

Es va notar que un cop connectada per primer cop, l'aplicació ja no necessitava realitzar la cerca de dispositius ja que es connectava automàticament a la bàscula utilitzada, simplificant així la feina de l'usuari de l'app final. Aquesta funcionalitat és molt interessant en aquest projecte ja que permet que l'usuari final, que seria una persona sense massa coneixements tècnics, no haver de buscar el dispositiu cada cop que li calgui fer una nova visita.

3.5. Creació d'una app a partir dels mòduls Ble de OpenScale

En aquest punt del projecte, es va decidir utilitzar els mòduls Bluetooth de l'app OpenScale per dissenyar una petita app que permetés connectar-se amb la bàscula mencionada, que fos el pas previ a la integració en la app OpenMRS que utilitza el projecte. Es va decidir així, en comptes d'utilitzar els mòduls ja implementats en l'aplicació anterior degut a que l'estructura del mòdul Bluetooth utilitzada en OpenScale era més clara i entenedora.

Es va enfocar el problema des de dos punts de vista diferents, el primer, agafar l'aplicació d'*Oliexdev* i esborrar totes les funcionalitats no desitjades per obtenir una versió simplificada i senzilla d'integrar degut als pocs mòduls que resultarien al final. El segon punt de vista, consistia en continuar la depuració de l'app i seguir el fil d'execució per aconseguir trobar el punt exacte on les nostres dades eren rebudes i extreure'n els mètodes adequats per obtenir la funcionalitat desitjada.

Al principi es va optar per la primera, ja que semblava la solució més senzilla. Es van anar eliminant mòduls gradualment i comprovant, mitjançant mesures, que la comunicació es seguia rebent. A mesura que s'anaven esborrant mètodes i mòduls no necessaris augmentaven els errors de compilació i s'havien d'anar solucionant per tal de poder fer les comprovacions. A sobre, augmentaven els *bugs* o errors de funcionament ja que només

s'estaven eliminant parts del *back-end* sense tenir en compte com podria això afectar a la part de la interfície gràfica. Això va acabar provocant una dificultat excessiva en la manera de comprovar si les dades s'havien rebut correctament. Finalment es va arribar a la conclusió de que aquest mètode no portaria a una solució ràpida i senzilla, ja que moltes coses estaven bastant vinculades entre si i no es podia saber amb exactitud si era estrictament necessari per a la comunicació Bluetooth.

Així, es va decidir optar pel mètode de la depuració per a seguir el tractament de dades que es realitzava. Utilitzant la funcionalitat d'Android Studio, i col·locant *breakpoints* en diversos llocs prèviament estudiats per veure l'estat de les dades en tot moment, es va trobar el punt on les dades es processaven i es convertien al format decimal. Mirant el *callstack* del fil d'execució es van obtenir el mètodes cridats anteriorment i d'aquí es va deduir quines eren les classes útils per al nostre problema.

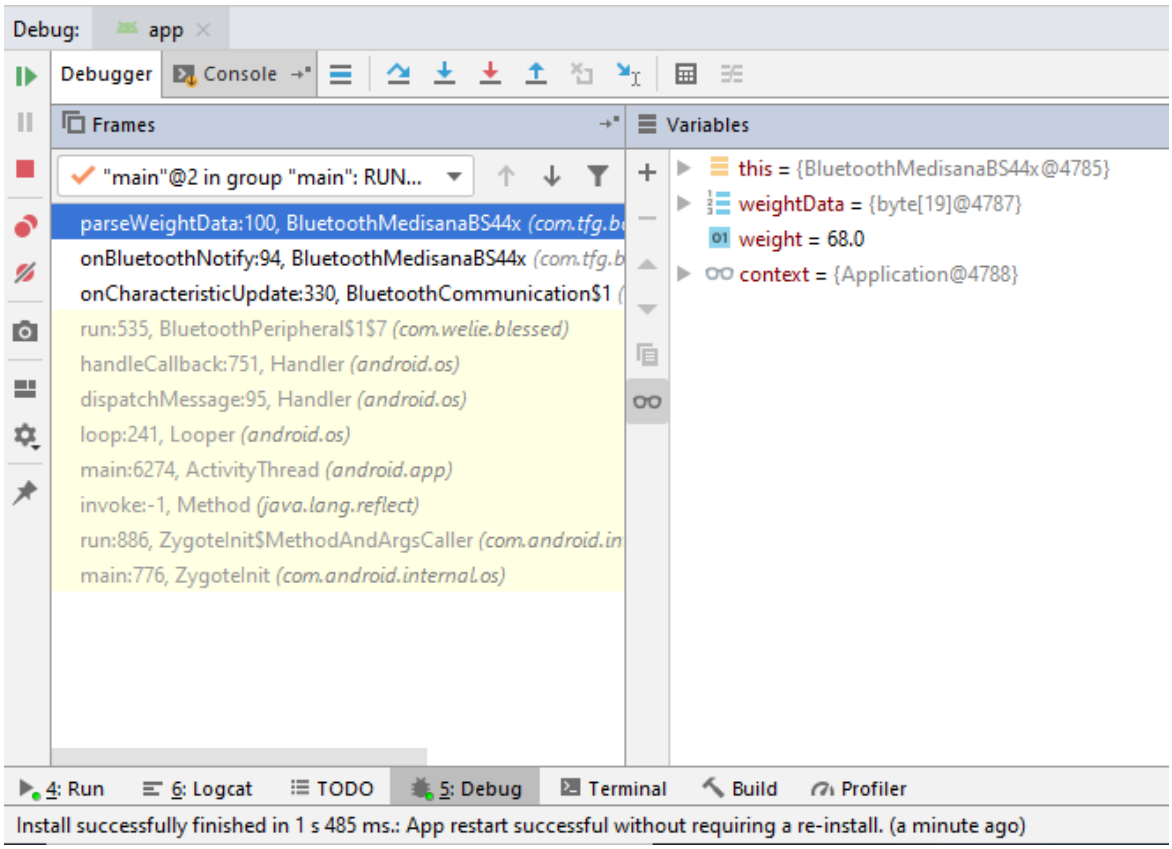

Figura 4: *Callstack* del punt on es converteixen els bytes en dades

Seguidament, es va procedir a crear una nova aplicació des de zero, amb una interfície gràfica senzilla que contenia un botó per a connectar-se i un botó que permet actualitzar el text per veure el resultat de la mesura. Amb aquest *front-end* es va procedir a agafar les classes obtingudes amb la investigació anterior, i a integrar-les amb la nova aplicació. Van aparèixer nous errors de compatibilitat i de referències que ja no eren utilitzades, i després

de solucionar-los, es va assignar els mètodes de cerca i connexió de la bàscula al botó. Un cop aquí, va aparèixer el problema de comunicar les dades del *back-end* al *front-end*. Tant OpenScale com OpenMRS disposen d'una base de dades basada en MySQL, però per a l'app de proves que després seria integrada al projecte no calia implementar una base de dades tant complexa per guardar un únic valor. En base a això, es va decidir guardar la variable del pes en un fitxer de text i consultar-lo quan s'hagués d'actualitzar la vista.

3.6. Integració d'openMRS i l'app pròpia.

Finalment, calia integrar l'aplicació desenvolupada amb l'app OpenMRS del projecte. Per a realitzar-ho es va seguir una metodologia semblant a la utilitzada per a integrar els mòduls Bluetooth a l'app pròpia. Es van importar les classes del *back-end* i es va procedir a solucionar els errors de compilació degut a les importacions de paquets externs i corregir les nomenclatures utilitzades anteriorment per les que es fan servir en el nou projecte.

Fet això, van aparèixer nombrosos errors de compatibilitat entre les versions d'Android que utilitzaven els nous mòduls Bluetooth i l'app del projecte. El problema venia donat per l'actualització de les llibreries de suport que oferia Android, que havien sigut renovades i canviades de nom a AndroidX. Això també afectava al dipòsit de Google.

Per tal de solucionar alguns problemes, Android Studio té la opció de fer una migració a AndroidX, tot i això, no va resultar gaire útil ja que molts paquets van continuar amb les dependències antigues. Així doncs, per continuar arreglant errors de compilació es va modificar el fitxer de construcció automàtica del projecte (*build.gradle*), per tal d'actualitzar totes les implementacions de llibreries antigues amb les noves. A continuació, es van actualitzar les importacions de les classes Java que així ho requerissin.

Després de solucionar tots els errors de compilació, es va muntar l'aplicació completa al telèfon mòbil per tal de comprovar el seu funcionament. A l'hora d'executar-la l'aplicació es bloquejava en una pantalla en blanc i no es podia utilitzar. Al ser incapaç d'accedir ni tan sols a la pantalla inicial de l'aplicació, resultava impossible depurar-la per intentar trobar quin era el problema.

4. Resultats

Malauradament, com s'ha explicat en el punt anterior, no es va aconseguir fer la integració final de les dues aplicacions desitjades, deixant això un resultat poc satisfactori ja que, tot i haver aconseguit una aplicació funcional capaç de connectar-se i realitzar un intercanvi satisfactori de dades amb la bàscula, no s'ha completat l'objectiu final d'obtenir una sola aplicació que tingui la capacitat de connectar-se amb el tensiòmetre i la bàscula.

En aquest apartat no s'analitzarà l'app OpenMRS del tensiòmetre perquè és l'aplicació que ja tenia el projecte BPUP i de la qual es partia a l'inici del projecte.

4.1. Aplicació desenvolupada per a la bàscula

A continuació es mostrarà el funcionament de l'aplicació que es pretenia integrar amb OpenMRS. L'app consta d'una pantalla amb 2 botons i una vista de text, tot i que a l'aplicació final no eren necessaris ni el botó d'actualitzar ni el text ja que les dades serien guardades directament a la base de dades i es podrien consultar al perfil del pacient.

Figura 5: Pantalla d'inici de l'aplicació

Per connectar-se a la bàscula només cal prémer el botó i es comença la cerca del dispositiu. Per configurar la bàscula, inicialment s'ha d'utilitzar l'app original per establir el primer aparellament. Per tant, no s'ha desenvolupat la funcionalitat de cerca de nous dispositius, ja que amb el codi implementat no s'ha de buscar cada cop el dispositiu sinó que s'utilitza la funcionalitat dels telèfons mòbils intel·ligents de guardar els dispositius amb els que s'han connectat algun cop. Notar que l'aplicació recorda a l'usuari que ha de mantenir la bàscula encesa durant tot el procés i que no val amb que estigui encesa al moment de donar-li al botó. Si l'usuari no té el Bluetooth del telèfon activat, l'aplicació mostra un missatge demanant permís per activar-lo. Després s'ha de tornar a prémer el botó de connectar-se

Figura 6: Pantalla de connexió de l'aplicació

Un cop connectada amb el dispositiu, l'aplicació notifica a l'usuari que ja pot pujar, (o fer pujar al pacient en el nostre cas) a la bàscula per prendre la mesura. També es sap que la connexió s'ha fet amb èxit observant la pantalla de la bàscula on surt una petita icona de Bluetooth indicant així que s'ha aparellat amb el telèfon correctament.

Figura 7: Pantalla quan l'aplicació s'ha connectat

Un cop realitzada la mesura, es pot veure com el número de la pantalla de la bàscula comença a mostrar-se intermitentment. Si es prem el botó d'actualitzar, es podrà veure la mesura també al telèfon mòbil, comprovant així, que s'ha rebut i s'ha guardat correctament al document esmentat al punt 3.5.

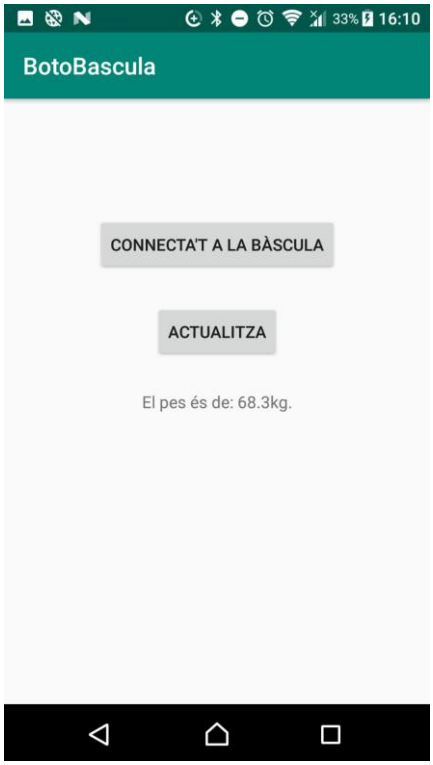

Figura 8: Pantalla de l'aplicació on es mostra el pes

Si la bàscula no està encesa en el moment de la cerca del dispositiu, l'aplicació mostra un missatge de connexió perduda. Igualment, si tornem a prémer el botó de connectar-se quan l'aplicació ja esta aparellada es mostra un missatge que mostra que l'aparellament s'ha desfet i es torna a cercar el dispositiu.

Figures 9 i 10: Pantalles on es mostra que la connexió no s'ha establert o s'ha tallat.

4.2. Integració de les dues aplicacions

Un cop integrades les dues aplicacions es va obtenir una aplicació que semblava correcta a priori, sense errors de compilació després d'haver substituït totes les llibreries per les noves d'AndroidX per arreglar els errors de compatibilitat, però el resultat no va ser l'esperat. Al muntar l'aplicació resultant al telèfon i executar-la per provar-la va quedar bloquejada mostrant una pantalla en blanc. Es va intentar depurar, però al estar bloquejada inclús abans d'entrar a la pantalla d'inici per entrar al servidor, no s'arribava a cap *breakpoint* dels que es van provar.

Aquest problema va deixar dues aplicacions per separat, l'app OpenMRS amb la versió antiga de llibreries d'Android, capaç de comunicar-se amb el tensiòmetre Bluetooth, i l'aplicació desenvolupada en aquest projecte, que utilitza la nova versió de llibreries d'AndroidX, capaç de comunicar-se amb la bàscula Bluetooth. Per altre banda tenim el resultat de la integració d'ambdues, que engloba les dues funcionalitats, però queda bloquejada al intentar executar-la, segurament per algun error en la portabilitat de llibreries d'Android a AndroidX, que devia generar algun problema en la muntatge de l'aplicació al telèfon.

5. Pressupost

Aquest projecte ha consistit únicament en la recerca i desenvolupament de software. Com que tots els programes utilitzats han sigut de lliure distribució, no s'ha hagut de comprar cap tipus de llicència o software addicional. Llavors, aquest pressupost contemplarà els costos de personal, i el hardware que s'ha requerit per al desenvolupament.

Cost del hardware

Component	**Cost (€)**
Ordinador portàtil*	100
Telèfon mòbil intel·ligent*	50
Total	150

Taula 1: Taula de costos de hardware

*Amortització dels dispositius per als 5 mesos de projecte considerant una vida útil de 5 anys.

Cost del personal

Empleat	**Total hores**	**€ / h**	**Cost (€)**
Enginyer Junior	540	10	5400
Total			5400

Taula 2: Taula de costos de personal

6. Conclusions i futurs desenvolupaments

Per resumir breument, aquest treball pretenia en un principi, realitzar la portabilitat de l'aplicació client-servidor OpenMRS que utilitzava el projecte BPUP, que tenia la funcionalitat del tensiòmetre (2.6), a la versió més recent desenvolupada per la plataforma OpenMRS (2.8), solucionant d'aquesta manera diversos errors i problemes que tenia la versió que s'utilitzava en aquell moment. A mitjans del projecte, degut a la petició del metge d'incloure una bàscula a l'aplicació, es va canviar l'objectiu principal d'actualitzar i millorar l'aplicació que ja es tenia per el d'implementar la nova funcionalitat que permetés connectar-se i rebre dades d'una bàscula Bluetooth. Veient els resultats obtinguts, podríem dir que tot i haver implementat correctament un mòdul Bluetooth capaç de fer la comunicació, no s'ha pogut integrar degut a la versió. Això fa pensar que, hagués sigut millor realitzar primer l'actualització, que hagués revelat molts d'aquests problemes de versions i llibreries antigues d'Android, i hagués fet mes senzilla la tasca d'integració total.

Podem concloure doncs, que no s'ha obtingut el millor resultat possible d'aquest projecte per una mala gestió dels objectius, pensant que obtenir més funcionalitats en l'app seria millor que dedicar més temps en arreglar i millorar els errors, que aparentment no semblaven tan problemàtics, de l'app de base.

Per a continuar el projecte, es podrien seguir dos camins, complir l'objectiu inicial actualitzant el client-servidor 2.6 al 2.8 i afegint el mòdul Bluetooth utilitzat pel tensiòmetre, i un cop actualitzat, integrar-hi l'aplicació desenvolupada en aquest projecte, sense problemes excessius de compatibilitat amb les noves llibreries AndroidX. Una altra manera de continuar seria continuar en el punt on s'ha arribat d'integració, intentant resoldre el problema d'execució de l'app. Finalment, després de realitzar la integració completa, es podrien resoldre els errors que genera la versió 2.6, o es podrien traslladar els mòduls Bluetooth a la versió 2.8. Si es vol acabar utilitzant la versió 2.8 d'OpenMRS (en la meva opinió, és la opció més adequada), recomanaria optar per la primera opció, que sembla més viable.

Bibliografia

Aquestes pàgines web van tornar a ser visitades el dia 24 de Gener de 2020 per assegurar que encara eren accessibles.

[1] S. Giménez, A. Sanchez "Blood Pressure Under Pressure" website, 2019. [Online] Available: http://bpup.eu/documentation/ [Last accessed: 24 January 2020].

[2] OpenMRS Community website, 2016. [Online] Available: https://openmrs.org/about/ [Last Accessed: 24 January 2020].

[3] M. Afaneh. Novel Bits website "What is BLE?", 2020. [Online] Available: https://www.novelbits.io/what-is-ble-bluetooth-low-energy-iot/. [Last Accessed: 24 January 2020].

[4] Android Community. "Developer Guides", 2019. [Online] Available: https://developer.android.com/guide. [Last Accessed: 24 January 2020].

[5] Android Community. "Class Mappings", 2019. [Online] Available: https://developer.android.com/jetpack/androidx/migrate/class-mappings [Last Accessed: 24 January 2020].

[6] Oliexdev. "How to reverse engineer a Bluetooth 4.x scale", 2019. [Online] Available: https://github.com/oliexdev/openScale/wiki/How-to-reverse-engineer-a-Bluetooth-4.x-scale [Last Accessed: 24 January 2020].

Annex – Codi de l'aplicació

En aquest annex es mostra el codi de l'aplicació desenvolupada. Es mostren només les classes i fitxers utilitzats per a la comunicació Bluetooth i la interfície gràfica, obviant així els fitxers generats per defecte en crear l'aplicació.

BluetoothCommunication.java

```
package com.tfg.botobascula.bluetooth;

import android.Manifest;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.le.ScanResult;
import android.content.Context;
import android.content.pm.PackageManager;
import android.location.LocationManager;
import android.os.Handler;
import android.os.Looper;
import android.widget.Toast;

import androidx.core.content.ContextCompat;

import com.tfg.botobascula.R;
import com.welie.blessed.BluetoothCentral;
import com.welie.blessed.BluetoothCentralCallback;
import com.welie.blessed.BluetoothPeripheral;
import com.welie.blessed.BluetoothPeripheralCallback;

import java.util.UUID;

import timber.log.Timber;

import static
android.bluetooth.BluetoothGattCharacteristic.WRITE_TYPE_DEFAULT;
import static
android.bluetooth.BluetoothGattCharacteristic.WRITE_TYPE_NO_RESPONSE;
import static android.content.Context.LOCATION_SERVICE;
import static com.welie.blessed.BluetoothPeripheral.GATT_SUCCESS;

public abstract class BluetoothCommunication {
    public enum BT_STATUS {
        RETRIEVE_SCALE_DATA,
        INIT_PROCESS,
        CONNECTION_RETRYING,
        CONNECTION_ESTABLISHED,
        CONNECTION_DISCONNECT,
        CONNECTION_LOST,
        NO_DEVICE_FOUND,
        UNEXPECTED_ERROR,
        SCALE_MESSAGE
    }

    private int stepNr;
    private boolean stopped;

    protected Context context;

    private Handler callbackBtHandler;
    private Handler disconnectHandler;
```

```
    private BluetoothCentral central;
    private BluetoothPeripheral btPeripheral;

    public BluetoothCommunication(Context context)
    {
        this.context = context;
        this.disconnectHandler = new Handler();
        this.stepNr = 0;
        this.stopped = false;
        this.central = new BluetoothCentral(context,
bluetoothCentralCallback, new Handler(Looper.getMainLooper()));
    }

    /**
     * Register a callback Bluetooth handler that notify any BT_STATUS
changes for GUI/CORE.
     *
     * @param cbBtHandler a handler that is registered
     */
    public void registerCallbackHandler(Handler cbBtHandler) {
        callbackBtHandler = cbBtHandler;
    }

    /**
     * Set for the openScale GUI/CORE the Bluetooth status code.
     *
     * @param status the status code that should be set
     */
    protected void setBluetoothStatus(BT_STATUS status) {
        setBluetoothStatus(status, "");
    }

    /**
     * Set for the openScale GUI/CORE the Bluetooth status code.
     *
     * @param statusCode the status code that should be set
     * @param infoText the information text that is displayed to the
status code.
     */
    protected void setBluetoothStatus(BT_STATUS statusCode, String
infoText) {
        if (callbackBtHandler != null) {
            callbackBtHandler.obtainMessage(
                    statusCode.ordinal(), infoText).sendToTarget();
        }
    }

    /**
     * Send message to openScale user
     *
     * @param msg the string id to be send
     * @param value the value to be used
     */
    protected void sendMessage(int msg, Object value) {
        if (callbackBtHandler != null) {
            callbackBtHandler.obtainMessage(
                    BT_STATUS.SCALE_MESSAGE.ordinal(), msg, 0,
value).sendToTarget();
        }
```

```
    }

    /**
     * Return the Bluetooth driver name
     *
     * @return a string in a human readable name
     */
    abstract public String driverName();

    /**
     * State machine for the initialization process of the Bluetooth
device.
     *
     * @param stepNr the current step number
     * @return false if no next step is available otherwise true
     */
    abstract protected boolean onNextStep(int stepNr);

    /**
     * Method is triggered if a Bluetooth data from a device is notified
or indicated.
     *
     * @param characteristic
     * @param value the Bluetooth characteristic
     */
    protected void onBluetoothNotify(UUID characteristic, byte[] value)
{}

    /**
     * Method is triggered if a Bluetooth services from a device is
discovered.
     *
     * @param peripheral
     */
    protected void onBluetoothDiscovery(BluetoothPeripheral peripheral)
{ }

    protected synchronized void stopMachineState() {
        Timber.d("Stop machine state");
        stopped = true;
    }

    protected synchronized void resumeMachineState() {
        Timber.d("Resume machine state");
        stopped = false;
        nextMachineStep();
    }

    protected synchronized void jumpNextToStepNr(int nr) {
        Timber.d("Jump next to step nr " + nr);
        stepNr = nr;
    }

    /**
     * Write a byte array to a Bluetooth device.
     *
     * @param characteristic the Bluetooth UUID characteristic
     * @param bytes          the bytes that should be write
     */
    protected void writeBytes(UUID service, UUID characteristic, byte[]
```

```
bytes) {
        writeBytes(service, characteristic, bytes, false);
    }

    /**
     * Write a byte array to a Bluetooth device.
     *
     * @param characteristic the Bluetooth UUID characteristic
     * @param bytes          the bytes that should be write
     * @param noResponse     true if no response is required
     */
    protected void writeBytes(UUID service, UUID characteristic, byte[]
bytes, boolean noResponse) {
        Timber.d("Invoke write bytes [" + byteInHex(bytes) + "] on " +
BluetoothGattUuid.prettyPrint(characteristic));

btPeripheral.writeCharacteristic(btPeripheral.getCharacteristic(service,
characteristic), bytes,
                noResponse ? WRITE_TYPE_NO_RESPONSE :
WRITE_TYPE_DEFAULT);
    }

    /**
     * Read bytes from a Bluetooth device.
     *
     * @note onBluetoothRead() will be triggered if read command was
successful. nextMachineStep() needs to manually called!
     *@param characteristic the Bluetooth UUID characteristic
     */
    void readBytes(UUID service, UUID characteristic) {
        Timber.d("Invoke read bytes on " +
BluetoothGattUuid.prettyPrint(characteristic));


btPeripheral.readCharacteristic(btPeripheral.getCharacteristic(service,
characteristic));
    }

    /**
     * Set indication flag on for the Bluetooth device.
     *
     * @param characteristic the Bluetooth UUID characteristic
     */
    protected void setIndicationOn(UUID service, UUID characteristic) {
        Timber.d("Invoke set indication on " +
BluetoothGattUuid.prettyPrint(characteristic));
        if(btPeripheral.getService(service) != null) {
            stopMachineState();
            BluetoothGattCharacteristic currentTimeCharacteristic =
btPeripheral.getCharacteristic(service, characteristic);
            btPeripheral.setNotify(currentTimeCharacteristic, true);
        }
    }

    /**
     * Set notification flag on for the Bluetooth device.
     *
     * @param characteristic the Bluetooth UUID characteristic
     */
    protected void setNotificationOn(UUID service, UUID characteristic)
```

```
{
        Timber.d("Invoke set notification on " +
BluetoothGattUuid.prettyPrint(characteristic));
        if(btPeripheral.getService(service) != null) {
            stopMachineState();
            BluetoothGattCharacteristic currentTimeCharacteristic =
btPeripheral.getCharacteristic(service, characteristic);
            btPeripheral.setNotify(currentTimeCharacteristic, true);
        }
    }

    /**
     * Disconnect from a Bluetooth device
     */
    public void disconnect() {
        Timber.d("Bluetooth disconnect");
        setBluetoothStatus(BT_STATUS.CONNECTION_DISCONNECT);
        try {
            central.stopScan();
        } catch (Exception ex) {
            Timber.e("Error on Bluetooth disconnecting " +
ex.getMessage());
        }

        if (btPeripheral != null) {
            central.cancelConnection(btPeripheral);
        }
        callbackBtHandler = null;
        disconnectHandler.removeCallbacksAndMessages(null);
    }

    /**
     * Convert a byte array to hex for debugging purpose
     *
     * @param data data we want to make human-readable (hex)
     * @return a human-readable string representing the content of
'data'
     */
    protected String byteInHex(byte[] data) {
        if (data == null) {
            Timber.e("Data is null");
            return "";
        }

        if (data.length == 0) {
            return "";
        }

        final StringBuilder stringBuilder = new StringBuilder(3 *
data.length);
        for (byte byteChar : data) {
            stringBuilder.append(String.format("%02X ", byteChar));
        }

        return stringBuilder.substring(0, stringBuilder.length() - 1);
    }

    protected float clamp(double value, double min, double max) {
        if (value < min) {
            return (float)min;
```

```
        }
        if (value > max) {
            return (float)max;
        }
        return (float)value;
    }

    protected byte xorChecksum(byte[] data, int offset, int length) {
        byte checksum = 0;
        for (int i = offset; i < offset + length; ++i) {
            checksum ^= data[i];
        }
        return checksum;
    }

    protected byte sumChecksum(byte[] data, int offset, int length) {
        byte checksum = 0;
        for (int i = offset; i < offset + length; ++i) {
            checksum += data[i];
        }
        return checksum;
    }

    /**
     * Test in a byte if a bit is set (1) or not (0)
     *
     * @param value byte which is tested
     * @param bit bit position which is tested
     * @return true if bit is set (1) otherwise false (0)
     */
    protected boolean isBitSet(byte value, int bit) {
        return (value & (1 << bit)) != 0;
    }

    private final BluetoothPeripheralCallback peripheralCallback = new
BluetoothPeripheralCallback() {
        @Override
        public void onServicesDiscovered(BluetoothPeripheral peripheral)
{
            Timber.d("Successful Bluetooth services discovered");
            onBluetoothDiscovery(peripheral);
            resumeMachineState();
        }

        @Override
        public void onNotificationStateUpdate(BluetoothPeripheral
peripheral, BluetoothGattCharacteristic characteristic, int status) {
            if( status == GATT_SUCCESS) {
                if(peripheral.isNotifying(characteristic)) {
                    Timber.d(String.format("SUCCESS: Notify set for %s",
characteristic.getUuid()));
                    resumeMachineState();
                }
            } else {
                Timber.e(String.format("ERROR: Changing notification
state failed for %s", characteristic.getUuid()));
            }
        }

        @Override
```

```
        public void onCharacteristicWrite(BluetoothPeripheral
peripheral, byte[] value, BluetoothGattCharacteristic characteristic,
int status) {
            if( status == GATT_SUCCESS) {
                Timber.d(String.format("SUCCESS: Writing <%s> to <%s>",
byteInHex(value), characteristic.getUuid().toString()));
                nextMachineStep();

            } else {
                Timber.e(String.format("ERROR: Failed writing <%s> to
<%s>", byteInHex(value), characteristic.getUuid().toString()));
            }
        }

        @Override
        public void onCharacteristicUpdate(final BluetoothPeripheral
peripheral, byte[] value, final BluetoothGattCharacteristic
characteristic, final int status) {
            resetDisconnectTimer();
            onBluetoothNotify(characteristic.getUuid(), value);
        }
    };

    // Callback for central
    private final BluetoothCentralCallback bluetoothCentralCallback =
new BluetoothCentralCallback() {

        @Override
        public void onConnectedPeripheral(BluetoothPeripheral
peripheral) {
            Timber.d(String.format("connected to '%s'",
peripheral.getName()));
            setBluetoothStatus(BT_STATUS.CONNECTION_ESTABLISHED);
            btPeripheral = peripheral;
            nextMachineStep();
            resetDisconnectTimer();
        }

        @Override
        public void onConnectionFailed(BluetoothPeripheral peripheral,
final int status) {
            Timber.e(String.format("connection '%s' failed with status
%d", peripheral.getName(), status ));
            setBluetoothStatus(BT_STATUS.CONNECTION_LOST);

            if (status == 8) {
                Toast.makeText(context.getApplicationContext(),
"Connectant a la bàscula", Toast.LENGTH_SHORT).show();
            }
        }

        @Override
        public void onDisconnectedPeripheral(final BluetoothPeripheral
peripheral, final int status) {
            Timber.d(String.format("disconnected '%s' with status %d",
peripheral.getName(), status));
        }

        @Override
        public void onDiscoveredPeripheral(BluetoothPeripheral
```

```
peripheral, ScanResult scanResult) {
            Timber.d(String.format("Found peripheral '%s'",
peripheral.getName()));
            central.stopScan();
            connectToDevice(peripheral);
        }
    };

    /**
     * Connect to a Bluetooth device.
     *
     * On successfully connection Bluetooth machine state is
automatically triggered.
     * If the device is not found the process is automatically stopped.
     *
     * @param macAddress the Bluetooth address to connect to
     */
    public void connect(String macAddress) {
        // Running an LE scan during connect improves connectivity on
some phones
        // (e.g. Sony Xperia Z5 compact, Android 7.1.1). For some scales
(e.g. Medisana BS444)
        // it seems to be a requirement that the scale is discovered
before connecting to it.
        // Otherwise the connection almost never succeeds.
        LocationManager locationManager =
(LocationManager)context.getSystemService(LOCATION_SERVICE);

        if (ContextCompat.checkSelfPermission(context,
Manifest.permission.ACCESS_FINE_LOCATION)
                == PackageManager.PERMISSION_GRANTED &&
(locationManager.isProviderEnabled(LocationManager.GPS_PROVIDER)
                ||
locationManager.isProviderEnabled(LocationManager.NETWORK_PROVIDER))
        ) {
            Timber.d("Do LE scan before connecting to device");
            central.scanForPeripheralsWithAddresses(new
String[]{macAddress});
            stopMachineState();
        }
        else {
            Timber.d("No location permission, connecting without LE
scan");
            BluetoothPeripheral peripheral =
central.getPeripheral(macAddress);
            connectToDevice(peripheral);
        }
    }

    private void connectToDevice(BluetoothPeripheral peripheral) {

        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Timber.d("Try to connect to BLE device " +
peripheral.getAddress());

                stepNr = 0;
```

```
                central.connectPeripheral(peripheral,
peripheralCallback);
            }
        }, 1000);
    }

    private void resetDisconnectTimer() {
        disconnectHandler.removeCallbacksAndMessages(null);
        disconnectWithDelay();
    }

    private void disconnectWithDelay() {
        disconnectHandler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Timber.d("Timeout Bluetooth disconnect");
                disconnect();
            }
        }, 60000); // 60s timeout
    }

    private synchronized void nextMachineStep() {
        if (!stopped) {
            Timber.d("Step Nr " + stepNr);
            if (onNextStep(stepNr)) {
                stepNr++;
                nextMachineStep();
            } else {
                Timber.d("Invoke delayed disconnect in 60s");
                disconnectWithDelay();
            }
        }
    }
}
```

BluetoothFactory.java

```
package com.tfg.botobascula.bluetooth;

import android.content.Context;



public class BluetoothFactory {

    public static BluetoothCommunication createDeviceDriver(Context
context, String deviceName) {

        // BS444 || BS440
        if (deviceName.startsWith("013197") ||
deviceName.startsWith("013198") || deviceName.startsWith("0202B6")) {
            return new BluetoothMedisanaBS44x(context, true);
        }

        //BS430
        if (deviceName.startsWith("0203B")) {
            return new BluetoothMedisanaBS44x(context, false);
        }
        return null;
```

```
    }
}
```

BluetoothGattUuid.java

```
package com.tfg.botobascula.bluetooth;

import java.lang.reflect.Field;
import java.util.Locale;
import java.util.UUID;

public class BluetoothGattUuid {
    private static final String STANDARD_SUFFIX = "-0000-1000-8000-
00805f9b34fb";

    public static final UUID fromShortCode(long code) {
        return UUID.fromString(String.format("%08x%s", code,
STANDARD_SUFFIX));
    }

    public static final String prettyPrint(UUID uuid) {
        if (uuid == null) {
            return "null";
        }

        String str = uuid.toString();

        if (str.endsWith(STANDARD_SUFFIX)) {
            String code = str.substring(0, str.length() -
STANDARD_SUFFIX.length());
            if (code.startsWith("0000")) {
                code = code.substring(4);
            }
            str = "0x" + code;
        }

        for (Field field : BluetoothGattUuid.class.getFields()) {
            try {
                if (uuid.equals(field.get(null))) {
                    String name = field.getName();
                    name = name.substring(name.indexOf('_') + 1);
                    str = String.format("%s \"%s\"", str,
                            name.replace('_', '
').toLowerCase(Locale.US));
                    break;
                }
            }
            catch (IllegalAccessException e) {
                // Ignore
            }
        }

        return str;
    }

}
```

BluetoothMedisanaBS44x.java

```
package com.tfg.botobascula.bluetooth;

import java.lang.reflect.Field;
import java.util.Locale;
import java.util.UUID;

public class BluetoothGattUuid {
    private static final String STANDARD_SUFFIX = "-0000-1000-8000-
00805f9b34fb";

    public static final UUID fromShortCode(long code) {
        return UUID.fromString(String.format("%08x%s", code,
STANDARD_SUFFIX));
    }

    public static final String prettyPrint(UUID uuid) {
        if (uuid == null) {
            return "null";
        }

        String str = uuid.toString();

        if (str.endsWith(STANDARD_SUFFIX)) {
            String code = str.substring(0, str.length() -
STANDARD_SUFFIX.length());
            if (code.startsWith("0000")) {
                code = code.substring(4);
            }
            str = "0x" + code;
        }

        for (Field field : BluetoothGattUuid.class.getFields()) {
            try {
                if (uuid.equals(field.get(null))) {
                    String name = field.getName();
                    name = name.substring(name.indexOf('_') + 1);
                    str = String.format("%s \"%s\"", str,
                            name.replace('_', '
').toLowerCase(Locale.US));
                    break;
                }
            }
            catch (IllegalAccessException e) {
                // Ignore
            }
        }

        return str;
    }

}
```

App.java

```
package com.tfg.botobascula;

import com.tfg.botobascula.bluetooth.*;
```

```
import android.content.Context;

import android.os.Handler;


import timber.log.Timber;

public class App {
    private static App instance;

    private BluetoothCommunication btDeviceDriver;


    private Context context;



    private App(Context context) {
        this.context = context;

        btDeviceDriver = null;


    }


    public static void createInstance(Context context) {
        if (instance != null) {
            return;
        }

        instance = new App(context);
    }

    public static App getInstance() {
        if (instance == null) {
            throw new RuntimeException("No App instance created");
        }

        return instance;
    }


    public boolean connectToBluetoothDevice(String deviceName, String
hwAddress, Handler callbackBtHandler) {
        Timber.d("Trying to connect to bluetooth device [%s] (%s)",
hwAddress, deviceName);

        disconnectFromBluetoothDevice();

        btDeviceDriver = BluetoothFactory.createDeviceDriver(context,
deviceName);
        if (btDeviceDriver == null) {
            return false;
        }

        btDeviceDriver.registerCallbackHandler(callbackBtHandler);
        btDeviceDriver.connect(hwAddress);
```

```
            return true;
        }

        public boolean disconnectFromBluetoothDevice() {
            if (btDeviceDriver == null) {
                return false;
            }

            Timber.d("Disconnecting from bluetooth device");
            btDeviceDriver.disconnect();
            btDeviceDriver = null;

            return true;
        }

}
```

Converters.java

```
package com.tfg.botobascula;


public class Converters {



    public static int fromSignedInt16Le(byte[] data, int offset) {
        int value = data[offset + 1] << 8;
        value += data[offset] & 0xFF;
        return value;
    }


    public static int fromUnsignedInt16Le(byte[] data, int offset) {
        return fromSignedInt16Le(data, offset) & 0xFFFF;
    }



    public static void toInt32Le(byte[] data, int offset, long value) {
        data[offset + 3] = (byte) ((value >> 24) & 0xFF);
        data[offset + 2] = (byte) ((value >> 16) & 0xFF);
        data[offset + 1] = (byte) ((value >> 8) & 0xFF);
        data[offset + 0] = (byte) (value & 0xFF);
    }


    public static byte[] toInt32Le(long value) {
        byte[] data = new byte[4];
        toInt32Le(data, 0, value);
        return data;
    }


}
```

MainActivity.java

```
package com.tfg.botobascula;

import androidx.appcompat.app.AppCompatActivity;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothManager;
import android.content.Intent;
import android.os.Bundle;

import android.os.Handler;
import android.os.Message;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

import com.tfg.botobascula.bluetooth.BluetoothCommunication;


import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;

import timber.log.Timber;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        App.createInstance(getApplicationContext());
        Button connectBtn = (Button) findViewById(R.id.connectbtn);
        Button updateBtn = (Button) findViewById(R.id.updatebtn);
        final TextView pesText = (TextView) findViewById(R.id.pesText);
        pesText.setText("Benvingut a l'app de mesura");
        connectBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //Botó que conecta a openscale
                invokeConnectToBluetoothDevice();
                Toast.makeText(getApplicationContext(), "Connectant a la
bàscula", Toast.LENGTH_SHORT).show();
                pesText.setText("Mantingui la bàscula encesa fins que es
realitzi la connexió");
            }

        });

        updateBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                updateText();
            }
        });

    }
```

```
    public void updateText(){
        String filePath =
getApplicationContext().getFilesDir().getPath().toString() +
"/measures.txt";
        File measures = new File(filePath);
        final TextView pesText = findViewById(R.id.pesText);
        pesText.setText("El pes és de: "+ readFromFile(measures) +
"kg.");
    }

    private void invokeConnectToBluetoothDevice() {

        final App app = App.getInstance();


        String deviceName = "0131986C2D3F38C1A4";
        String hwAddress = "A4:C1:38:3F:2D:6C";

        if (!BluetoothAdapter.checkBluetoothAddress(hwAddress)) {

            Toast.makeText(getApplicationContext(), "Dispositiu no
conectat", Toast.LENGTH_SHORT).show();
            return;
        }

        BluetoothManager bluetoothManager = (BluetoothManager)
getSystemService(BLUETOOTH_SERVICE);
        if (!bluetoothManager.getAdapter().isEnabled()) {

            Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, 0);
            return;
        }

        Toast.makeText(getApplicationContext(), "Trying to connect to
device", Toast.LENGTH_SHORT).show();


        if (!app.connectToBluetoothDevice(deviceName, hwAddress,
callbackBtHandler)) {
            Toast.makeText(getApplicationContext(), "Bàscula no
suportada", Toast.LENGTH_SHORT).show();
        }
    }

    private final Handler callbackBtHandler = new Handler() {
        @Override
        public void handleMessage(Message msg) {

            BluetoothCommunication.BT_STATUS btStatus =
BluetoothCommunication.BT_STATUS.values()[msg.what];

            switch (btStatus) {

                case INIT_PROCESS:

                    Toast.makeText(getApplicationContext(),"Bluetooth
initializing", Toast.LENGTH_SHORT).show();
```

```
                    Timber.d("Bluetooth initializing");
                    break;
                case CONNECTION_LOST:
                    Toast.makeText(getApplicationContext(), "Bluetooth
connection lost", Toast.LENGTH_SHORT).show();
                    Timber.d("Bluetooth connection lost");
                    break;
                case NO_DEVICE_FOUND:

                    Toast.makeText(getApplicationContext(), "No
Bluetooth device found", Toast.LENGTH_SHORT).show();
                    Timber.e("No Bluetooth device found");
                    break;
                case CONNECTION_RETRYING:
                    Toast.makeText(getApplicationContext(), "No
Bluetooth device found retrying", Toast.LENGTH_SHORT).show();
                    Timber.e("No Bluetooth device found retrying");
                    break;
                case CONNECTION_ESTABLISHED:
                    Toast.makeText(getApplicationContext(), "Bluetooth
connection successfully established", Toast.LENGTH_SHORT).show();
                    Timber.d("Bluetooth connection successfully
established");
                    break;
                case CONNECTION_DISCONNECT:
                    Toast.makeText(getApplicationContext(), "Bluetooth
connection successfully disconnected", Toast.LENGTH_SHORT).show();
                    Timber.d("Bluetooth connection successfully
disconnected");
                    break;
                case UNEXPECTED_ERROR:
                    Toast.makeText(getApplicationContext(), "Bluetooth
unexpected error", Toast.LENGTH_SHORT).show();
                    Timber.e("Bluetooth unexpected error: %s", msg.obj);
                    break;
                case SCALE_MESSAGE:
                    String toastMessage =
String.format(getResources().getString(msg.arg1), msg.obj);
                    Toast.makeText(getApplicationContext(),
toastMessage, Toast.LENGTH_LONG).show();
                    break;
            }
        }
    };

    public String readFromFile(File file){

        FileInputStream fis = null;
        String textWeight="";
        try {
            fis = new FileInputStream(file);

            System.out.println("Total file size to read (in bytes) : "
                    + fis.available());

            int content;
            while ((content = fis.read()) != -1) {
                // convert to char and display it
                textWeight=textWeight + (char) content;
            }
```

```
        } catch (IOException e) {
            e.printStackTrace();
        } finally {

            try {
                if (fis != null) {
                    fis.close();
                }
                return textWeight;
            } catch (IOException ex) {
                ex.printStackTrace();
                return textWeight;
            }
        }
    }


}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/pesText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Benvingut a l'app de mesura"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/connectbtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Connecta't a la bàscula"
        app:layout_constraintBottom_toTopOf="@+id/pesText"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <Button
        android:id="@+id/updatebtn"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Actualitza"
        app:layout_constraintBottom_toTopOf="@+id/pesText"
        app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/connectbtn" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Glossari

BPUP: Blood Pressure Under Pressure.

Single-Board Computer: Ordinador complet muntat únicament en una sola placa de circuits.

Raspberry Pi: SBC de baix cost molt comú per estimular l'ensenyament d'informàtica a les escoles.

MySQL: Sistema de gestió de bases de dades relacional, multi-fil i multiusuari. Utilitza el llenguatge *Structured Query Language.*

Back-end: Part de l'aplicació en què l'usuari no accedeix directament, normalment utilitzada per emmagatzemar i manipular la informació.

Front-end: Part de l'aplicació amb la que interactua l'usuari.

XML: Extensible Markup Language

IDE: Entorn de desenvolupament intel·ligent.

GitHub: Servei de hosting de repositoris Git.

OpenScale: Aplicació de software de lliure distribució, per monitoritzar l'estat cardiovascular de l'usuari.

WireShark: Analitzador de paquets de lliure distribució.

Little endian: Codificació dels bytes on el bit de menys pes és col·locat en primera posició.

Big endian: Codificació dels bytes on el bit de major pes és col·locat en primera posició.

Breakpoint: Punt on es genera una interrupció en la depuració d'una aplicació.

Callstack: Pla de crides realitzades en l'execució d'un codi. Permet veure els diferents mètodes que s'han anat cridant durant l'execució.