REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur
Et de la Recherche Scientifique

Université Dr. YAHIA FARES **de** *Médéa*

Mémoire de Projet de Fin d'Etude pour
L'Obtention du Diplôme de Master (LMD)
En Génie électrique
Option: Systèmes Intelligents Et Communicant s

Thème :

Développement d'une application Androïd pour un système d'acquisition de données portatif

Proposé et dirigé par :

Mr : CHADLI Samir .

Réalisé par :

M^elle^ : RABAHIE Souhila.
M^elle^ : BENTEGRI Nesrine.

Promos : 2012 - 2013

L’étude est un trésor qui suivra son propriétaire Partout…..



Remerciements

Au terme de notre projet fin de cycle, nous tenons à Remercier toutes les personnes, qui par leurs conseils, leurs suggestions ou par leur simple présence.

Nous remercions spécialement mon encadreur Monsieur ***CHADLI SAMIR*** *pour son encadrement tout au long de ce projet, sa patience, sa disponibilité et ses conseils toujours avisés.*

Nous remercions très sincèrement, les membres de jury d'avoir bien voulu accepter de faire partie de la commission d'examinateur

DÉDICACE



... A Mes chérs parents « nadia » et« mohamed »

... A Ma grande mére

... A Ma chére tante « aziza »

... A Ma jumelle « imene »

... A Ma grande sœur « f.zohra»

... A Mon frére « walid »

... A Mon petit frére « yacine »

... A Mes chérs ancles

... A Mes fidéles amies : amina sihem, faiza , hadjer , meriem

... A Mon binome

... A T ous ceux qui compt pour moi

... A votre àme « baba » et « yema »

je dédies ce travail....

.....SOUHILA.....

.

Je dédie cette thèse à ...

A MES TRÈS CHERS PARENTS : FARIDA ET ABDELKADER

Affables, honorables, aimables : Vous représentez pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi.

A MES CHERS FRÈRES : MOHAMED , BENYOUCEF ET HAKIM

En témoignage de l'attachement et que vos étiez toujours présents t dans tous mes moments d'examens par vos soutien moral et vos belles surprises

Je vous souhaite un avenir plein de joie, de bonheur, de réussite et de sérénité

A TOUS MEMBRES DE MA FAMILLE, PETITS ET GRANDS

surtout Chahrazad

A TOUS LES AMIS

surtout Sihem

.....NESRINE.....

ملخص:

في هذا العمل قمنا بتصميم و برمجة تطبيق محمول ضمن نظام أندرويد ، يحمل على الهواتف الذكية ، المساعدات الشخصية الرقمية ، الألواح الرقمية أو أي جهاز يعمل بنظام أندرويد . التطبيق يسمح بالتحكم في نظام الكتروني - يعتمد على متحكم مصغـر - لاقتناء بيانات (قياسات حساسات) و يسمح باسترجاع هذه القياسات بواسطة وصلة بلوتوث . القياسات بعد ذلك تخزن على رقاقة ذاكرة الجهاز أو تعرض على المباشـــر على شاشة الجهاز.

كلمات مفتاحية: أندرو يد، اقتناء،بلوتوث ،الهواتف الذكية.

Résumé :

Dans ce projet nous avons conçus et programmé une application mobile sous Android qui peut être utilisée avec les Smartphones, les PDA, les tablettes ou autre dispositifs Android. L'application permet de configurer une carte électronique - à base d'un microcontrôleur - d'acquisition de données (de capteurs) et permet de récupérer les mesures via la liaison Bluetooth. Les données par la suite sont stockées sur la carte mémoire du dispositif Android ou bien visualisées en temps réel sur l'écran du dispositif.

Mots clé : Android, acquisition, Bluetooth, Smartphone.

Abstract :

In this project we have designed and programmed a mobile application on Android that can be used with smart phones, PDAs, tablets or other Android devices. The application allows you to configure an e-card - based on a microcontroller - data acquisition (sensors) and allows to recover the action via Bluetooth. The data is subsequently stored on the memory card of the Android device or viewed in real time on the screen of the device.

Key words: Android, acquisition, Bluetooth, Smart phone.

SOMMAIRE

Introduction generale ……01

Chapitre 1 :

Généralités autour d'Android

I. Introduction…….03

II. L'informatique mobile……03

II.1 Presentation……..03

II.2 Les Smartphones……..03

III. Le système d'exploitation Android ……...04

III.1 Définition…….04

III.2 Historique d'Android……...04

III.3 Avantage d'Android……....06

III.3.1 Open source……..06

III.3.2 Gratuit……...06

III.3.3 Facile à développer……...06

III.3.4 Flexible…….....06

III.4 Architecture d'Android……....…06

III.5 Les difficultés du développement pour des systèmes embarqués..07

IV. Développement d'application Android……08

IV.1 Langage…….08

- Java……08
- Interface XML……08

IV.2 Machine virtuelle……...09

IV.3 Mise en place de l'espace de travail……..…09

IV.4 Les outils mis à disposition par le SDK……..10

IV.5 Composant d'une application……12

IV.6 Cycle de vie d'une activité…….13
IV.7 Architecture de base d'un projet java Android……...15
V. Conclusion…….15

Chapitre 2 :
Conception de l'application Android

I. Cahier de charges ……..16
II. Analyse et spécification des besoins……..17
II-1- Spécification des besoins……..17
II-2- Analyse des besoins…….18
II-3- Description……..20
III. Conception de l'application……23
III-1- Liste des activités……..…23
III-2- Architecture logiciel……..24
IV. Implémentation de l'application……25
IV-1- Définition des droits…….25
IV-2- Recherche des équipements…….26
IV-3- Pairage et connexion de deux équipements…….…….27
IV-4- Envoi de la configuration……..……28
IV-5- Réception des données……..…….29
IV-6- Stockage des données…….29
IV-7- Visualisation des données…….30
V. Conclusion……31

Chapitre 3 :
mise en marche et tests

I. Introduction……32
II. Les cartes électroniques Android……..……..…32
III. Présentation de la carte utilisée ……..……...33

III.1 carte IOIO……33
III.2 Les avantages de la carte IOIO……..35
IV. Mise en marche de l'application……..…35
IV.1 Chargement de l’application dans le Smartphone …….…..35
IV.2 Démarrage de l'application: « my_pfe »…….…..36
IV.3 Configuration du système d’acquisitions…….…….37
IV.4 Réceptions de données……..….37
IV.5 Recherche des périphériques Bluetooth……40
V. Test et résultat……..……..42
VI. Conclusion ……..……..44
Conclusion général…….……..50
Références bibliographiques…….……51
Annexe……..……52

Liste des figures

Figure I- 1 : Evolution d'Android……04
Figure I- 2 : Siège social de Google……05
Figure I- 3 : Répartition actuelle des versions d'android……..…..05
Figure I- 4 : Architecture d'Android……..……07
Figure I- 5 : Exemple d'interface…….……..08
Figure I- 6 : outils de développement……...……..09
Figure I- 7 : Gestionnaire de l'émulateur……11
Figure I- 8 : Création d'un émulateur……...…….11
Figure I- 9 : affichage de l'émulateur……...….12
Figure I- 10 : Composants principaux d'une application Android…...…12
Figure I- 11 : cycle de vie d'une activité……...…14
Figure I- 12 : Projet java Android…….....15

Figure II- 1 : « Communication entre deux utilisateurs A et B »….……16
Figure II- 2 : « Diagramme de cas d'utilisation globale »…….20
Figure II- 3 : « Diagramme de cas d'utilisation de « gestion des configurations »……..……..21
Figure II- 4 : « Diagramme de cas d'utilisation de « gestion de visualisation »……..22
Figure II- 5 : «Diagramme de cas d'utilisation de « gestion de connexion Bluetooth » …….23
Figure II- 6 : « Liste des activités »…….……24
Figure II- 7 : « Fichier AndroidManifest.xml » …….…….26
Figure II- 8 : « Recherche des équipements » ……..27
Figure II- 9 : « Pairage et connexion de deux équipements » …….28
Figure II- 10 « Envoi des données »…….28

Figure II- 11 : « Réception des données »…….29
Figure II- 12 : « Stockage des données »……29
Figure II- 13 : « L'interface de la visualisation »……30
Figure II- 14 : « La méthode set-data »……31

Figure III- 1: « La carte IOIO associée à un AndroPhone»…….33
Figure III- 2: « Vue de face et d'arrière de la carte IOIO»…….35
Figure III- 3 :« Apparition de l'application dans le Smartphone »……...36
Figure III- 4« L'interface d'accueil de l'application Android réalisée »…36
Figure III- 5: « Les valeurs par défaut de la configuration »…….37
Figure III- 6: « Demande d'autorisation d'activation du Bluetooth local ».38
Figure III- 7: « L'utilisateur refuse l'activation de son Bluetooth »……38
Figure III- 8: « Sortie de l'interface de visualisation »……38
Figure III- 9: « L'interface de visualisation »……...39
Figure III- 10: « Le message d'alerte lié à l'établissement de la connexion entre les deux périphériques»……..39

Figure III- 11: « Liste des périphériques déjà sauvegardés »……40
Figure III- 12: « L'établissement de la connexion Bluetooth »…….40
Figure III- 13: « Message d'alerte de perte de la connexion Bluetooth»…..41
Figure III- 14: « Recherche des périphériques Bluetooth »……41
Figure III- 15: « Le test effectué avec un Générateur de fréquence »……..42
Figure III- 16: « Capture d'écran du premier Test »……43
Figure III- 17: « Capture d'écran du deuxième Test »…….43

Liste des abréviations

ADT	Android Development Tools
ADB	Android Debug Bridge
API	Application Programming Interface
APK	Android Package
CAN	Convertisseur Analogique Numérique
C.à.d.	c'est-à-dire
DDMS	Dalvik Debug Monitor Server
Ect	Et cetera
E/S	Entrée/Sortie
GPS	Global Positioning System
HTC	High Technology Computer
IDE	Integrated Development Environment
i.e	Id est
JRE	Java Runtime Environment
JDK	Java Development Kit
MacOS	Macintosh Operating System
MVC	Modèle-Vue-Contrôleur
OHA	Open Handset Alliance
OS	Operating System
P**WM**	Pulse Width Modulation
SDK	Software Development Kit
SMS	Short Message Service

SPP	Serial Port Profile
SPI	Serial Peripheral Interface
SD	Secure Digital
USB	Universal Serial Bus
UML	Unified Modeling Language
UART	Universal Asynchronous Receiver Transmitter
XML	Extensible Markup Language



Introduction Générale

Introduction générale :

Le domaine de la télécommunication a connu ces 20 dernières années des évolutions considérables dans le monde en générale et dans les pays en voie de développement en particulier. Il est indéniable que cette évolution, passe par les systèmes de communication sans fils (basés essentiellement sur la transmission radio) qui, se démarquent de plus en plus des autres systèmes de communication à travers les diverses recherches et investissements menés par les différents laboratoires et organismes spécialisés dans le domaine, qui poussent de plus en plus leurs recherches afin de développer des technologies encore plus innovantes dans ce secteur depuis la fin des années 1980.

Avec la multiplication des dispositifs mobiles (téléphones portables, tablettes, PDA, Smartphones, et bien d'autres systèmes) à travers le monde, les réseaux mobiles ont vu leur nombre d'abonnés et leurs revenus se multiplier de façon considérable depuis le début des années 1990. Ceci dit, plusieurs solutions de communication sans fil existent de nos jours parmi lesquelles : le GSM, le GPRS (General Packet Radio Service), l'UMTS (universal mobile télécommunication system), le Wi-Fi, l'infrarouge et le Bluetooth.

Les 3 premières technologies (GSM, GPRS, UMTS), sont beaucoup plus adaptées aux communications longues portées et nécessitent des équipements importants et des installations fastidieuses pour leur déploiement et pour leur mise en œuvre. Pour le cas du Wi-Fi et le Wi-Fi Direct, bien qu'offrant des services conséquents en matière de communication sans fil, ils ont tout de même de nombreuses limites : le Wi-Fi malgré son bon débit et sa longue portée est limité par son coût, son encombrement et sa forte consommation d'énergie; le Wi-Fi Direct, ayant hérité des propriétés du Wi-Fi classique a du même coup hérité des principaux inconvénients de ce dernier en plus du fait qu'il ne soit pas encore assez répondu dans le domaine de la téléphonie.

L'infrarouge pour sa part, bien que consommant très peu d'énergie électrique, est limité par sa très faible portée (de 5 à 10 mètres seulement) et par les perturbations dû aux interférences lumineuses (lumière du soleil, lumière des tubes à Néons) en plus les infrarouges ne peuvent passer à travers les objets c-à-d les appareils à reliés doivent donc être en contact visuel.

La dernière solution (le Bluetooth) dans le cadre des petits réseaux personnels (PAN) est une technologie qui regorge d'atouts et qui palie à certaines limites présentées par ses prédécesseurs de par la petite taille et le faible coût en prix et en énergie consommée de sa puce.

De plus, le Bluetooth offre plusieurs services (profils) de communication sans fils de proximité parmi lesquels le transfert de fichier, la connexion de périphérique sans fils entre eux et aussi de la téléphonie de proximité qui retiendra particulièrement notre attention dans la suite.

D'autre part le développement des applications électroniques embarquées sont de plus en plus demandées sur le marché des équipements électroniques pour satisfaire divers besoins de communication, de sécurité, de confort, de contrôle, de loisir, ...etc. Ces applications sont très appréciées par l'utilisateur lorsqu'ils sont pilotées par des tablettes, des PDA ou des Smartphone sui sont devenus des dispositifs incontournables, intelligents et de plus en plus moins chère. On avait l'intention de profiter de la présence de ce Smartphone dans tout le monde.

C'est dans ce cadre que nous avons voulus utilisé ces technologies dans un autre domaine a part celui des jeux et de loisir pour offrir une solution de mesure, de visualisation et de stockage de données d'une carte d'acquisition électronique.

Dans ce travail nous nous proposons de concevoir et de réaliser une application mobile sous Android destinée à un Smartphone, une tablette, un PDA ou autre dispositif de communication mobile fonctionnant sous Android pour la réception des données d'un ensemble de capteurs interfacés à un système électronique portatif embarqué. La communication entre le système embarqué et le dispositif mobile est réalisé via la liaison Bluetooth. Les données reçues sont stockées dans la mémoire du Smartphone (memoire micro SD) et visualisé en temps réel sur une interfacé graphique a deux dimension.

Notre mémoire est organisé comme suit :

- Le premier chapitre présente un ensemble de généralités liées à la programmation sous Android, les outils à utilisés, ...etc.
- Le deuxième chapitre présente la démarche suivie pour réaliser l'application.
- Le troisième chapitre montre les différents résultats obtenus après le test de l'application dans un téléphone Android.

Enfin on clôture ce document avec une conclusion générale et des perspectives.



Chapitre I : Généralités autour d'Android

Généralités

I. Introduction :

Ce chapitre portera sur une étude théorique sur les technologies de base adoptées pour réaliser notre projet. Nous allons commencer par introduire l'informatique mobile et ses dispositifs appropriés. Nous définirons par la suite le système d'exploitation Android : son historique, ces versions, son architecture et les outils nécessaires pour créer une application Android.

II. L'informatique mobile :

II.1 Présentation :

L'informatique mobile s'inscrit sous l'approche globale de l'informatique ubiquitaire. Historiquement, elle remonte au début des années 90. Depuis lors, elle constitue un outil de communications très puissant et en pleine évolution pour les entreprises et les utilisations personnelles.

II.2 Les Smartphones :

Un Smartphone est un téléphone « intelligent », appelé aussi ordiphone. C'est un appareil dédié aux communications mobiles, disposant d'un système d'exploitation ouvert et adoptant des applications tierces développées par le fabricant, par l'opérateur ou par un éditeur de logiciel.

Ce téléphone fournit un module de radiocommunication (pour la voix et l'échange de données), des fonctionnalités bureautiques (agenda, calendrier, navigation web, email, messagerie instantanée, GPS, ...etc.) ainsi que des fonctionnalités multimédia (photo, musique, vidéo et jeux).

En effet, les Smartphones sont dotés d'un système d'exploitation qui gère leurs fonctionnalités.

III. Le système d'exploitation Android :

III.1 Définition :

- Android est un système d'exploitation destiné pour les Smartphones. Ce projet est né d'un consortium de 34 entreprises (en 2008) [1], initié par la société Google le 5 novembre 2007, et appelé l'Open Handset Alliance ou OHA. Son but consistait à trouver une solution fiable pour concurrencer Apple avec l'Iphone IOS, Microsoft avec Windows Mobile, Nokia avec Symbian et Research In Motion avec Blackberry OS. En effet à cette époque, la quasi-totalité des Smartphones fonctionnait sur ces systèmes d'exploitation.

III.2 Historique d'Android: [2]

L'histoire d'Android commence en octobre 2003, où la société Android Inc. est créée officiellement, elle développe des logiciels pour mobiles. Mais en réalité, elle se préparait à sortir un tout nouveau système d'exploitation pour Smartphones. En 2005, Google rachète cette entreprise, et sort une première version béta en novembre 2007, avant de lancer la version 1.0 en septembre 2008 avec le HTC Dream.

À partir de ce moment là, le rythme des nouvelles sorties est très élevé : pas moins de 11 versions différentes sont sorties en 3 ans ! (figure I.1) On notera au passage que chaque version d'Android porte le nom d'un dessert.

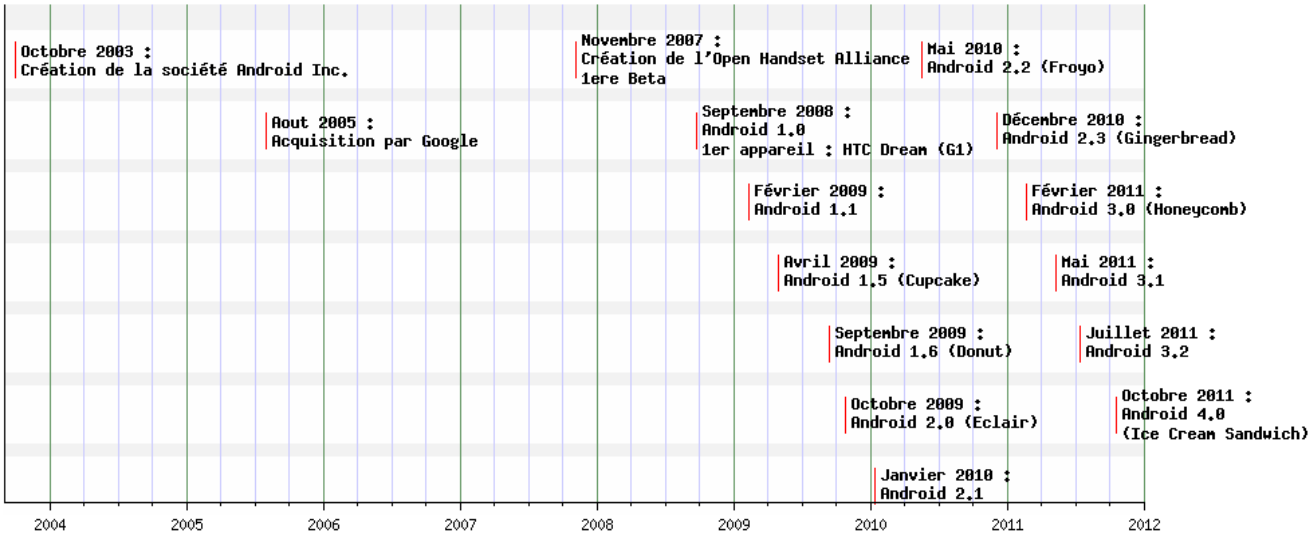


Figure I- 1 : « Evolution d'Android

Enfaite, la société Google n'a pas hésité à installer dans les jardins du Googleplex plusieurs représentations sous forme de statues représentant ces pâtisseries. Ces statues sont ajoutées au fur est à mesure des sorties des nouvelles versions ce qui donne une solution plutôt originale pour signaler les évolutions du système. (figure I-2) :



Figure I- 2 : « Siège social de Google »

La figure suivante montre la répartition actuelle (année 2013) des versions d'Android [3] :

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.7%
2.2	Froyo	8	3.7%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	38.4%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	27.5%
4.1.x	Jelly Bean	16	26.1%
4.2.x		17	2.3%

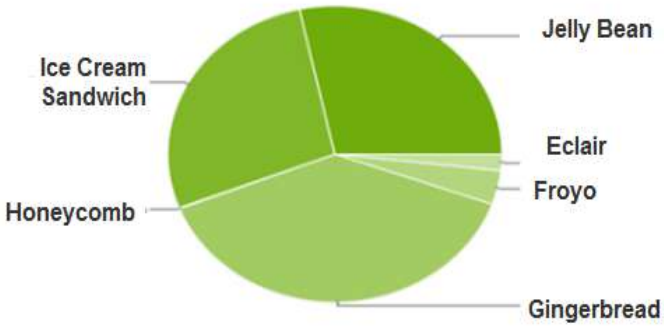


Figure I- 3 : « Répartition actuelle des versions d'android »

III.3 Avantage d'Android :

III.3.1 Open source :

Ce système d'exploitation est open source sous licence Apache 2.0. La licence Apache autorise l'utilisation du code source pour l'élaboration d'un produit libre, propriétaire, gratuit ou commercial.

III.3.2 Gratuit :

Android est gratuit, autant pour l'utilisateur que pour les constructeurs. Produire son propre téléphone sous Android ne nécessite aucunes dépenses si ce n'est un investissement personnel. Un utilisateur Android permet de poster plus facilement ses applications qu'un Iphone, le monde des applications Android étant plus accessible que l'Apple Store.

III.3 Facile à développer :

Le SDK (Software Development Kit) d'Android offre complètement les APIs pour développer les applications sous Android, ce qui facilite énormément la programmation.

III.4 Flexible :

Android s'adapte à de nombreux systèmes : les Smartphones, les tablettes, PDA, terminaux mobiles, ...etc.

III.4 Architecture d'Android: [3]

Android bénéficie d'une architecture en couche complète faisant de lui une plateforme riche, dédiée aux appareils mobiles.

Le noyau linux lui assure sécurité, gestion de mémoire, gestion de processus, etc. À un niveau supérieur se trouvent un ensemble de librairies écrites en C/C++ jouant le rôle d'un middleware (on en cite le système de bibliothèque C, les médiathèques, le SGL, etc.). C'est sur cette couche que se greffe l'Android Runtime, comprenant la machine virtuelle java et ses bibliothèques.

Vient ensuite la plateforme logicielle, nommée aussi Framework de développement, écrite en java et permettant de mutualiser les ressources entre applications Java. Elle offre aux développeurs la possibilité de produire des applications à travers un ensemble d'API.

Enfin, et à un niveau plus supérieur se situe un ensemble d'applications sous forme de paquets .apk.

Les applications fournies par Android se présentent sous divers formes d'applications telles qu'un navigateur web, un client mail, un calendrier, un gestionnaire de contacts, ...etc.

Cette architecture est illustrée sur la figure I-4.

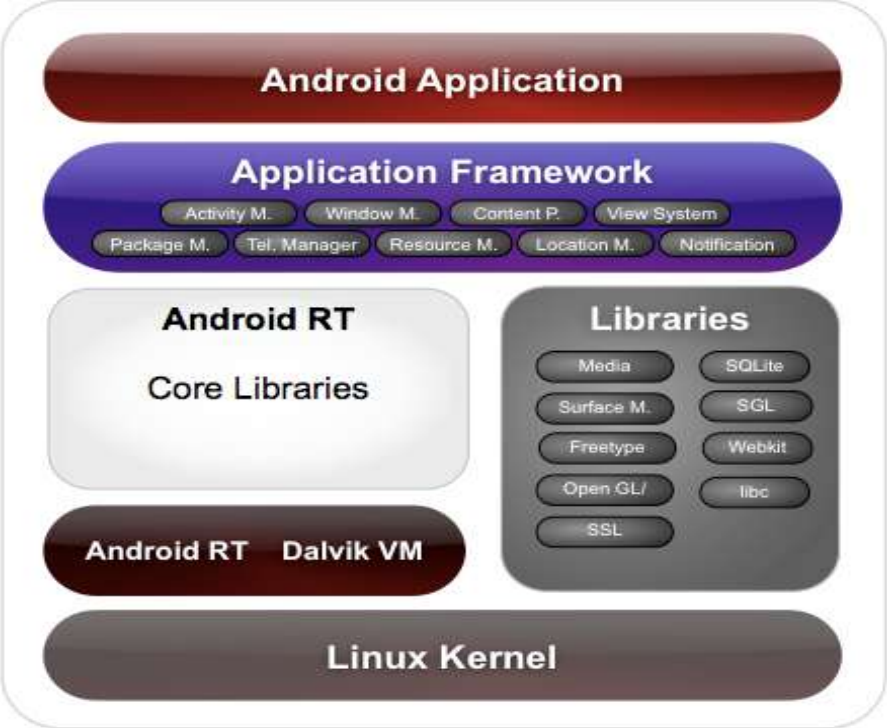


Figure I- 4 : « Architecture d'Android »

III.5 Les difficultés du développement pour des systèmes embarqués :

Il existe un certain nombre de contraintes pour le développement Android, qui ne s'appliquent pas au développement informatique habituel. Voici les principales contraintes à prendre en compte quand on développe pour un environnement mobile :

- Il faut pouvoir interagir avec un système complet sans l'interrompre. Android fait des choses pendant que l'application est utilisée, il reçoit des SMS et des appels, entre autres. Il faut respecter une certaine priorité dans l'exécution des tâches.
- La taille de l'écran est réduite, et il existe par ailleurs plusieurs tailles et différentes résolutions d'écrans. L'interface graphique doit s'adapter à toutes les tailles et toutes les résolutions.
- Enfin, en plus d'avoir une variété au niveau de la taille de l'écran, on a aussi une variété au niveau de la langue, des composants matériels présents et des versions d'Android. Il y a une variabilité entre chaque téléphone et même parfois entre certains téléphones identiques. C'est un travail en plus à prendre en compte.

IV. veloppement d'application Android :

IV.1 Langage :

• Java : Pour développer sur Android, Google à mis à disposition un SDK, Ce SDK, basé sur du Java, demande quelques mises à niveau, mais il est à la portée de tout développeur ayant des connaissances en développement orienté objet.

• Interface XML : Les interfaces utilisateurs sous Android sont définies dans un langage balisé de type XML, Cette approche donne la possibilité au développeur d'écrire son interface, d'utiliser un outil de création d'interface ou encore de le développer lui même. Ce format étant ouvert et humainement compréhensible.

La figure suivante nous montre comment générer une interface graphique en XML :

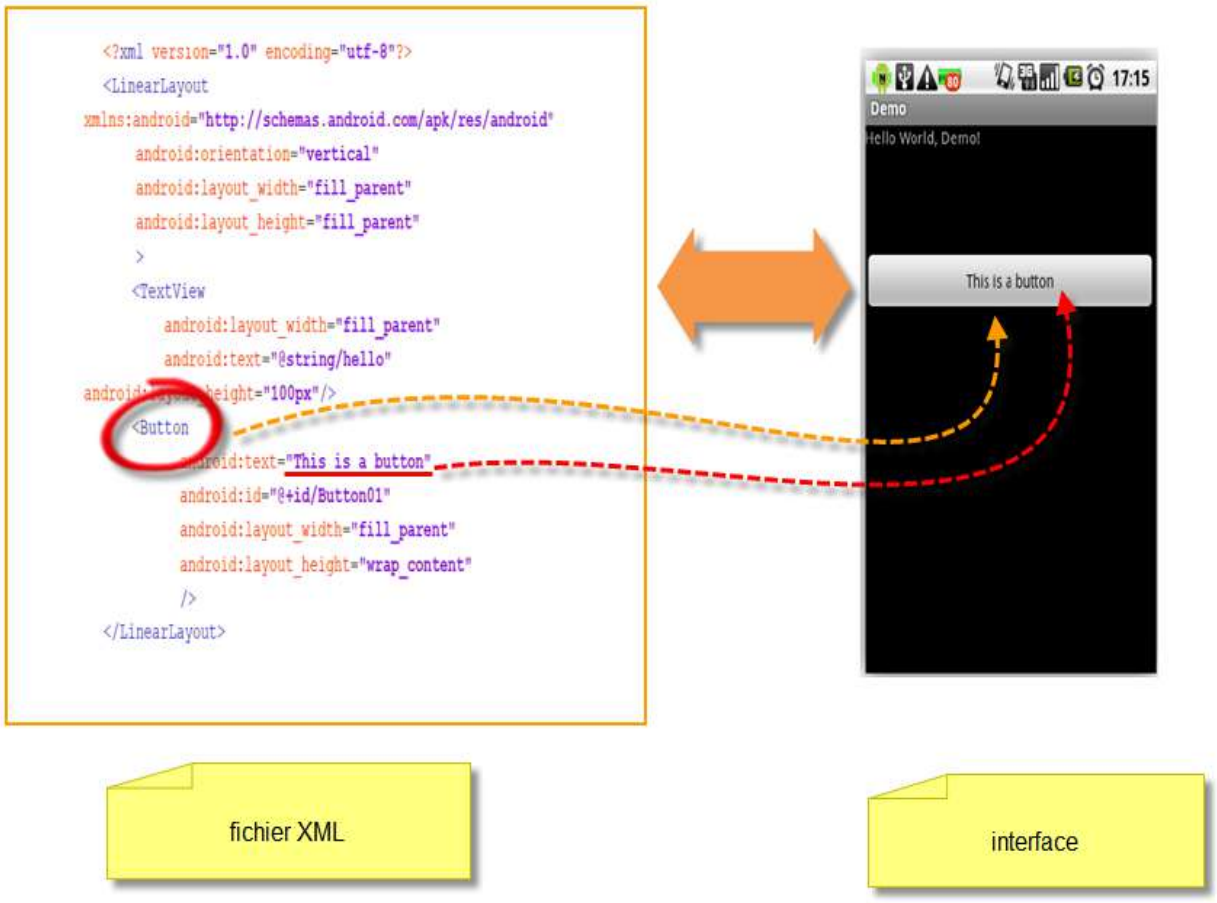


Figure I- 5 : « Exemple d'interface »

IV.2 Machine virtuelle :

La machine virtuelle Dalvik est une machine qui se basant sur celle de Sun pour le Java, mais optimisée pour Android. Sa particularité est de précompiler les applications dans un format, appelé Dalvik Dex Format, de manière à optimiser la gestion de la mémoire.

Pour anecdote, le nom Dalvik provient du nom d'un village d'Islande où auraient vécu certains ancêtres de Dan Bornstein, développeur de la machine virtuelle. Son code source étant ouvert c'est ce que Myriad (une société faisant partie de l'Open Handset Alliance) a optimisé pour promettre une vitesse d'exécution des applications trois fois supérieure qu'avec la Dalvik machine actuelle.

IV.3 Mise en place de l'espace de travail :

La mise en place d'un espace de travail pour développer sur Android est relativement aisée. Contrairement au développement pour l'IPhone qui exige l'utilisation d'un ordinateur Mac fonctionnant sous MacOS X, il est possible d'avoir un environnement de développement pour Android sous Linux, Microsoft Windows et Mac OS.

La figure suivante présente les outils nécessaires pour le développement sous Android :



Figure I- 6 : « Outils de développement »

Pour développer des applications Android il faut installer un certain nombre de kits de développements et configurer son système :

- ✓ Installation du JDK kit de développement JAVA :

JDK est le kit de développement proposé par Sun Microsystème. Ce kit en plus de contenir java Runtime Environement JRE, comprend aussi plusieurs outils de développement.

- ✓ Installation du SDK Android sous windows :

L'outil le plus important est le SDK Android. Facile à installer, il permet de télécharger tous les outils indispensables au développement d'applications. Un petit logiciel permet d'abord de télécharger les différentes versions du SDK (une version du SDK par version d'Android : 1.4, 1.5, 1.6, 2.0 etc.). Il permet également de télécharger les différentes versions des Google APIs (APIs pour intégrer des fonctionnalités liées aux services Google tels que Maps, ... etc.) ou de la documentation JavaDoc.

✓ Configuration de l'environnement du développement intégré (IDE):

Un environnement de développement (IDE : Integrated Development Environment) facilitera le développement d'applications. Eclipse est l'IDE le plus utilisé pour la programmation Java, il est très performant. On a téléchargé la version Juno Service Release 2. Il n'a pas besoin d'être installé, il s'exécute comme une application portable.

Afin d'intégrer les outils Android dans Eclipse il faut installer Android Development Tools (ADT) à partir du site « https://dl-ssl.google.com/Android /eclipse/». Ces derniers constituent un plugin pour Eclipse IDE pour l'installation du simulateur, en plus il permettant de compiler, d'exécuter du code Android, de créer des interfaces graphiques, ...etc. Après le redémarrage d'Eclipse IDE et la déclaration du chemin du répertoire d'installation d'Android dans les préférences, on obtient un environnement de travail, prêt à l'emploi.

IV.4 Les outils mis à disposition par le SDK :

❖ ADB (Android Debug Bridge) :

ADB est un outil disponible en ligne de commande. Comme son nom l'indique, il permet de se connecter à l'émulateur ou au téléphone à l'aide d'un câble USB standard. En outre, il donne la possibilité d'installer ou de désinstaller des applications, d'envoyer ou de récupérer des fichiers et de se connecter en ligne de commande sur l'appareil. Il est aussi utilisé par le SDK pour publier et débugger les applications en développement et peut-être aussi sollicité par certaines applications.

❖ L'émulateur :

Le SDK d'Android met aussi à disposition des développeurs un outil en ligne de commande appelé « émulateur ».

Cet outil permet de configurer puis de lancer un émulateur de Smartphone sous Android. L'avantage de cet outil est qu'il donne la possibilité de développer une application pour Android sans avoir de matériel particulier, comme un smartphone. De plus, il est entièrement intégré graphiquement sous Eclipse.

- Voici comment créer puis lancer un émulateur :
  - Pour lancer le gestionnaire d'émulateur, il suffit de cliquer sur l'icône représentant un téléphone sous éclipse.

Voir la figure I-7:



Figure I- 7 : « Gestionnaire de l'émulateur »

- Ensuite s'affiche la liste des émulateurs déjà configurés. Pour en crée un nouveau, il faut cliquer sur Creat new Android Virtual Device (AVD), une boite de dialogue s'affiche et qui doit etre remplie correctement permet la création d'un ADT (figure I-8):

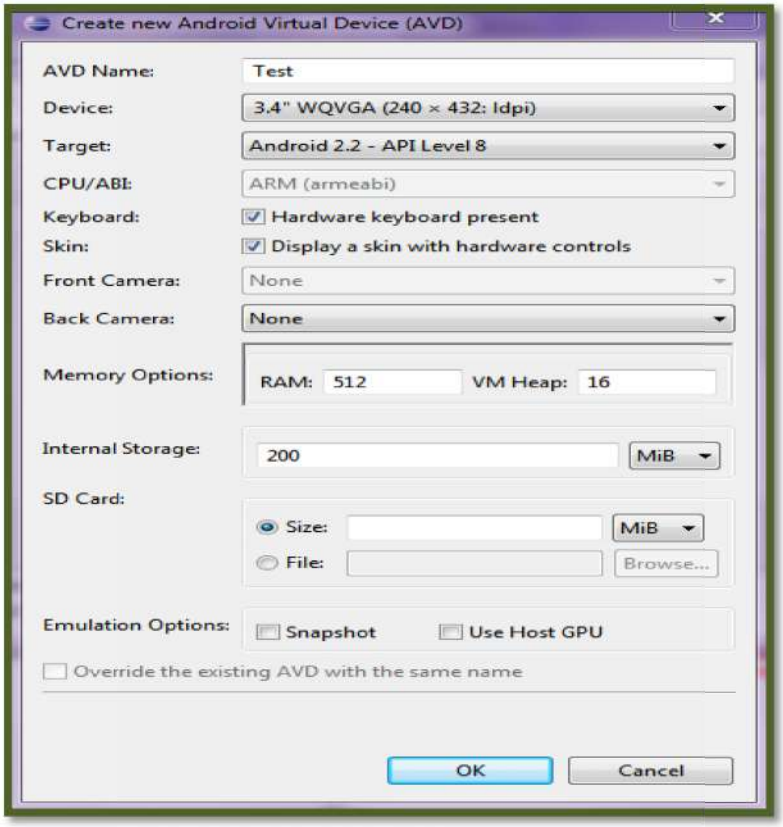


Figure I- 8 : « Création d'un émulateur »

- Après avoir spécifié un nom et la version d'Android souhaité, il est possible de créer l'émulateur qui s'affiche par la suite (figure I-9):



Figure I- 9 : « Affichage de l'émulateur »

❖ DDMS (Dalvik Debug Monitor Server):

Le DDMS est l'outil indispensable de tout développeur Android. Il est totalement intégré à Eclipse mais il est aussi possible de le lancer en lançant la commande « ddms ». Il est l'interface graphique de l'ADB. C'est grâce à cet outil que l'on pourra surveiller et manager les processus s'exécutant sur l'appareil ou l'émulateur, surveiller les journaux de log grâce au « logcat », envoyer et recevoir des fichiers, ou encore faire des captures d'écrans.

IV.5 Composant d'une application :

Une application Android hérite toujours d'un des quatre types définis sur la figure suivante :

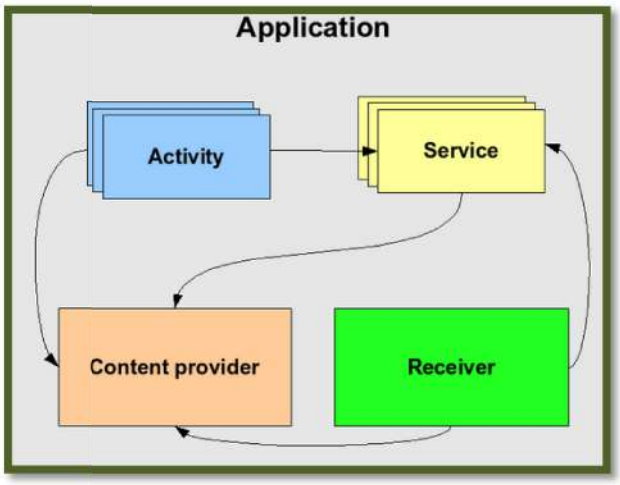


Figure I- 10 : « Composants principaux d'une application Android »

- Activity :

L'activity est l'application de base sous Android. Comme une application avec une interface pour l'utilisateur dépend toujours d'une activity, la plupart des applications en ont donc au moins une. L'activity va permettre d'afficher une vue contenants des contrôles. Ces contrôles sont simplement les listes, images, bouton, champs de saisie, ...Etc.

Dans le modèle MVC (Model, View, Controller), l'activity est le controller. C'est donc en quelque sorte le moteur de l'application. Une activity peut en ouvrir une autre, lui fournir des données et en récupérer à sa fermeture.

- Service :

Un service est une application sans interface pour l'utilisateur puisque fonctionnant en arrière-plan. C'est justement avec ces services que le multitâche d'Android à un sens.

Le service permettra par exemple de lire un morceau de musique tout en faisant autre chose. Ou encore continuer le téléchargement d'un fichier malgré la fermeture de l'application.

- Broadcast Receiver:

Les broadcast receiver sont une alternative aux services. C'est une sorte de service endormi qui se réveille suite à une demande spécifiée. Ils ont été créés pour répondre au besoin d'économie d'énergie que demandent les appareils comme les smartphones. Ils permettent aussi d'écouter des messages du système ou d'autres applications comme la réception d'un SMS.

- Content Provider :

Les Content Provider, comme leur nom l'indique, sont des gestionnaires de données. Ils permettent la mise à disposition de données filtrables aux autres applications. Ils sont utilisés pour récupérer la liste des SMS ou encore la liste des contacts.

IV.6 Cycle de vie d'une activité : [4]

Pour développer une application sur Android, on doit comprendre le cycle de vie d'une activité. Ce cycle de vie est exprimé par la figure suivante (figure I-10):

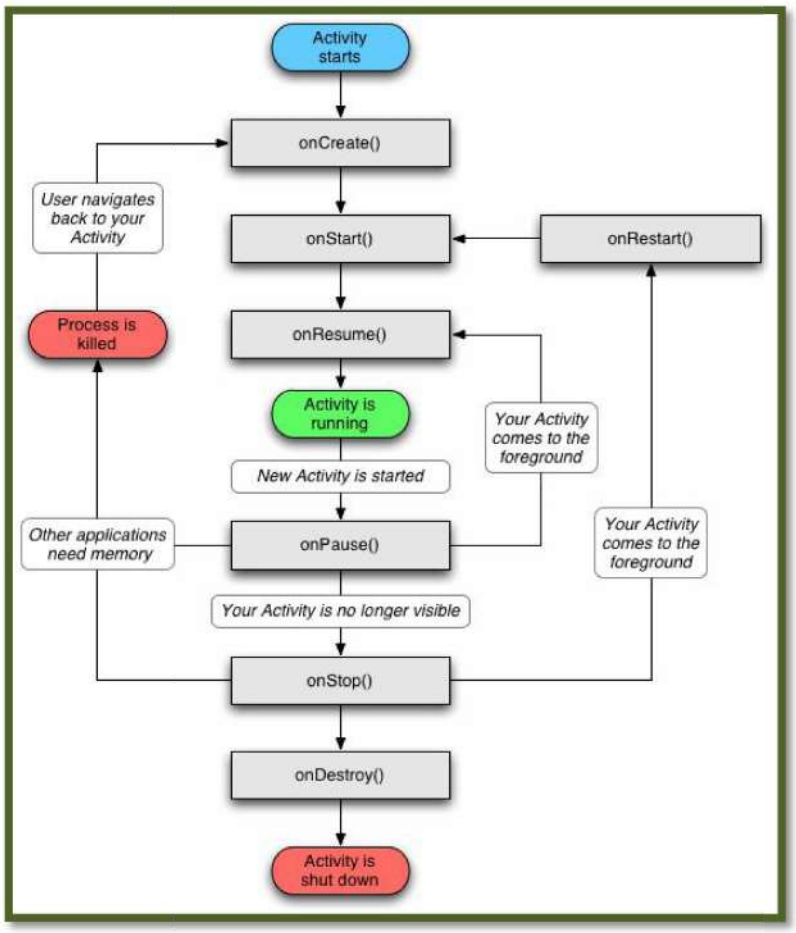


Figure I- 11 : « Cycle de vie d'une activité »

Voici un descriptif des différentes méthodes de callback que l'on peut implémenter dans une activité.

Méthode	Description
onCreate()	Appelée quand l'activité est crée la première fois. C'est ici que les déclarations statiques seront définies, les vues créées.
onRestart()	Appelée lorsque l'application a été stoppée et qu'elle est réactivée.
onStart()	Appelée juste avant que l'activité devienne visible à l'utilisateur.
onResume()	Appelée juste avant que l'activité interagisse avec l'utilisateur. L'activité est à cet instant tout en haut de la back stack
onPause()	Appelée quand le système est sur le point de rappeler une autre activité. Cette méthode est généralement utilisée pour sauvegarder les données non persistées, stopper les animations ou toute autre action consommatrice de CPU. Ces actions doivent être par contre exécutées très rapidement car la prochaine activité ne sera pas rappelée avant la fin de ces actions. Le système peut par contre couper le traitement s'il est trop long
onStop()	Appelée quand l'activité n'est plus visible de l'utilisateur, quand l'activité est détruite ou quand elle est mise en arrière plan au dépend d'une autre activité. Le traitement de cette méthode peut être coupé s'il est trop long
onDestroy()	Appelée avant que l'activité soit détruite. Peut être appelée si l'activité est finie ou si le système a besoin d'espace mémoire. Tout comme les deux précédentes méthodes, le système peut couper le traitement si ce dernier prend trop de temps

Tableau I- 1 : Différente méthodes de Callback

IV.7 Architecture de base d'un projet java Android :

Afin de voir les différents éléments composant un projet, nous allons détailler l'architecture d'un projet android. Examinons la figure I-12 :

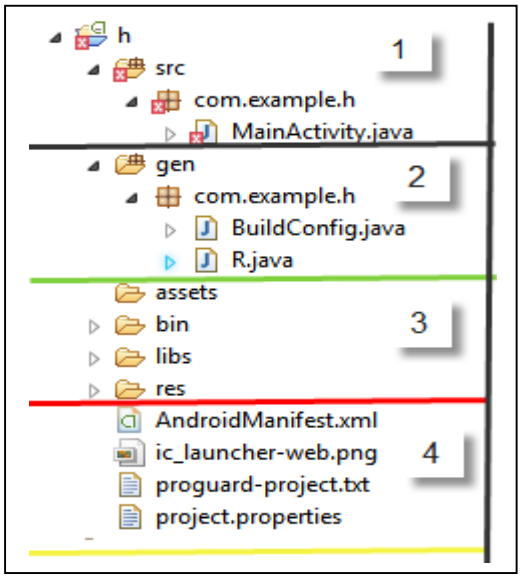


Figure I- 12 : « Projet java Android »

- La zone une correspond aux sources Java du projet. C'est dans ce dossier « src » que sont disposées les « Activity », les classes d'accès aux données et toute la partie moteur du projet.
- La zone deux comprend les fichiers générés par le SDK et les librairies qu'utilise le projet. Les fichiers générés ne doivent pas être modifiés puisqu'ils sont souvent mis à jour automatiquement par le SDK. Le fichier « R.java » permet au SDK de faire le lien entre les identifiants utilisés dans les sources et les ressources auquel ils correspondent.
- La zone trois contient les différentes ressources du programme. Il peut s'agir d'image, de son, de vue ou de n'importe quels fichiers devant être utilisés par l'application.

Plus précisément, le dossier « assets » permet la copie de fichiers de tous types alors que le dossier « res » correspond aux ressources identifiables par le SDK. Les ressources contrairement aux « assets » sont prises en compte par le SDK. Elles doivent être classées dans des sous-dossiers spécifiques et seront accessibles dans le code de manière simplifiée.

C'est dans le fichier « R.java » que le SDK se chargera de renseigner l'ajout ou la modification de ressources.

- La dernière zone, surlignée en jaune, correspond au fichier définissant les propriétés de l'application et du projet. Le fichier « AndroidManifest .xml » comprend toutes les propriétés de l'application comme son nom, son « package name » ou encore sa version. C'est aussi dans ce fichier que l'on va spécifier les permissions demandées par l'application. Le fichier « default.properties » est un fichier générer automatiquement par Eclipse et contient les propriétés du projet. Ce fichier n'est donc pas à prendre en considération.

V. Conclusion :

Après avoir énuméré les différents aspects liés à l'informatique mobile et les outils indispensable pour le développement sous Android, nous passons au chapitre suivant qui présente la conception de notre application Android.



Chapitre II : Conception de l'application Android





Conception de L'application Android

Dans ce qui suit nous présentons la démarche suivie pour le développement de notre application Android ainsi que les choix effectués et les outils utilisés.

I. Cahier de charges :

L'objectif du projet est de réaliser une application qui fonctionne sur la plateforme Android afin d'être installée sur un Smartphone, une tablette tactile, un PDA, … chargée de la réception des données (mesures) envoyées par une carte électronique doté d'un périphérique Bluetooth à base d'un microcontrôleur, d'une ASIC, d'un FPGA, ou autre.

Ces donnés serons par la suite représentées sur une interface graphique (en temps réel) et serons stockées dans la mémoire Flash du système. L'application permet aussi de configurer la carte électronique afin de contrôler la procédure d'acquisition des mesures.

L'application communique avec le système d'acquisition de données grâce au protocole Bluetooth selon le profil SPP (Serial Port Profile). Comme le montre la figure suivante :

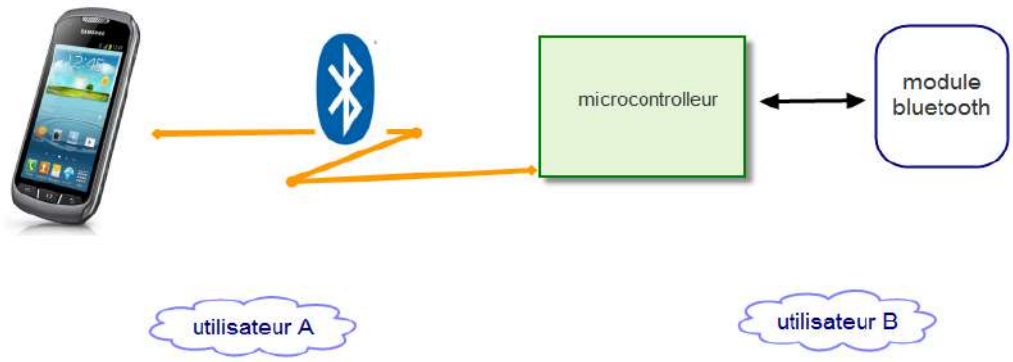


Figure II- 1 : « Communication entre deux utilisateurs A et B »

II. Analyse et spécification des besoins :

L'analyse et la spécification des besoins sert à identifier les acteurs réactifs du système et leur associer chacun l'ensemble d'actions avec lesquelles il intervient dans l'objectif de donner un résultat optimal et satisfaisant au client.

II-1- Spécification des besoins :

- Besoins fonctionnels :

Notre application doit être capable de configurer et d'acquérir des données venant d'une carte électronique d'acquisition via un périphérique Bluetooth utilisant le profil SPP selon un protocole simple de communication. Ceci afin d'exploiter l'application avec une grande variété de cartes électroniques utilisées en pratique.

Donc ce système doit permettre à l'utilisateur de :

- Faire la configuration de la carte d'acquisition :
  - ✓ sélectionner le nombre de canaux.
  - ✓ sélectionner la fréquence d'échantillonnage.
  - ✓ sélectionner la durée d'acquisition.
- Activer le Bluetooth local.
- Rechercher un équipement.
- Consulter le répertoire des contacts Bluetooth.
- Se connecter à un équipement.
- Envoyer la configuration.
- Débuter l'acquisition des données.
- Controller la visualisation des données acquises.
- Arrêter l'acquisition de données.
- Retourner d'un écran à un autre.
- Quitter l'application.

- Besoins non fonctionnels :

- Version d'Android : pour pouvoir utiliser l'application, il est nécessaire de posséder une version d'Android supérieure ou égale à la 2.1 pour la prise en compte du Bluetooth.
- En termes d'énergie, l'application doit avoir un cout d'utilisation de la batterie le plus faible possible.

- Ressources (matérielles et logiciel):

- Matériels :

- un ordinateur pouvant supporter l'environnement de développement Eclipse et l'émulateur Android.
- Un smart phone Android minimum avec Bluetooth.
- Une Catre électronique avec un module Bluetooth pour les tests.
- Une carte mémoire SD pour le stockage des données.

Logiciel :

- Eclipse : qui est la plateforme de développement utilisé, dont la version eclipse-jee-juno est disponible en téléchargement à l'adresse : http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/juno/SR2/eclipse-java-juno-SR2-win32.zip
- La machine virtuelle java JDK.
- Android SDK : la dernière version du SDK Android est (android_ r_r22.0.1-windows) et les différentes plugins associés. Notamment, l'ADT (Android Developper Toolkit) pour Eclipse.

II-2- Analyse des besoins :

• Généralisation des acteurs :

Si un ensemble d'acteurs communiquent de la même façon avec certains cas d'utilisation, on peut créer un acteur généralisé (souvent abstrait), qui permettra de factoriser ce rôle commun. Les acteurs spécialisés héritent alors des associations de l'acteur ancêtre [5].

En se basant sur cette définition ; Nous avons un seul acteur qui intervient sur le système et qui n'est rien d'autre que l'utilisateur de l'application.

• Organigramme :

L'organigramme présenté sur la figure II-2 décrit les composantes de notre application Android :

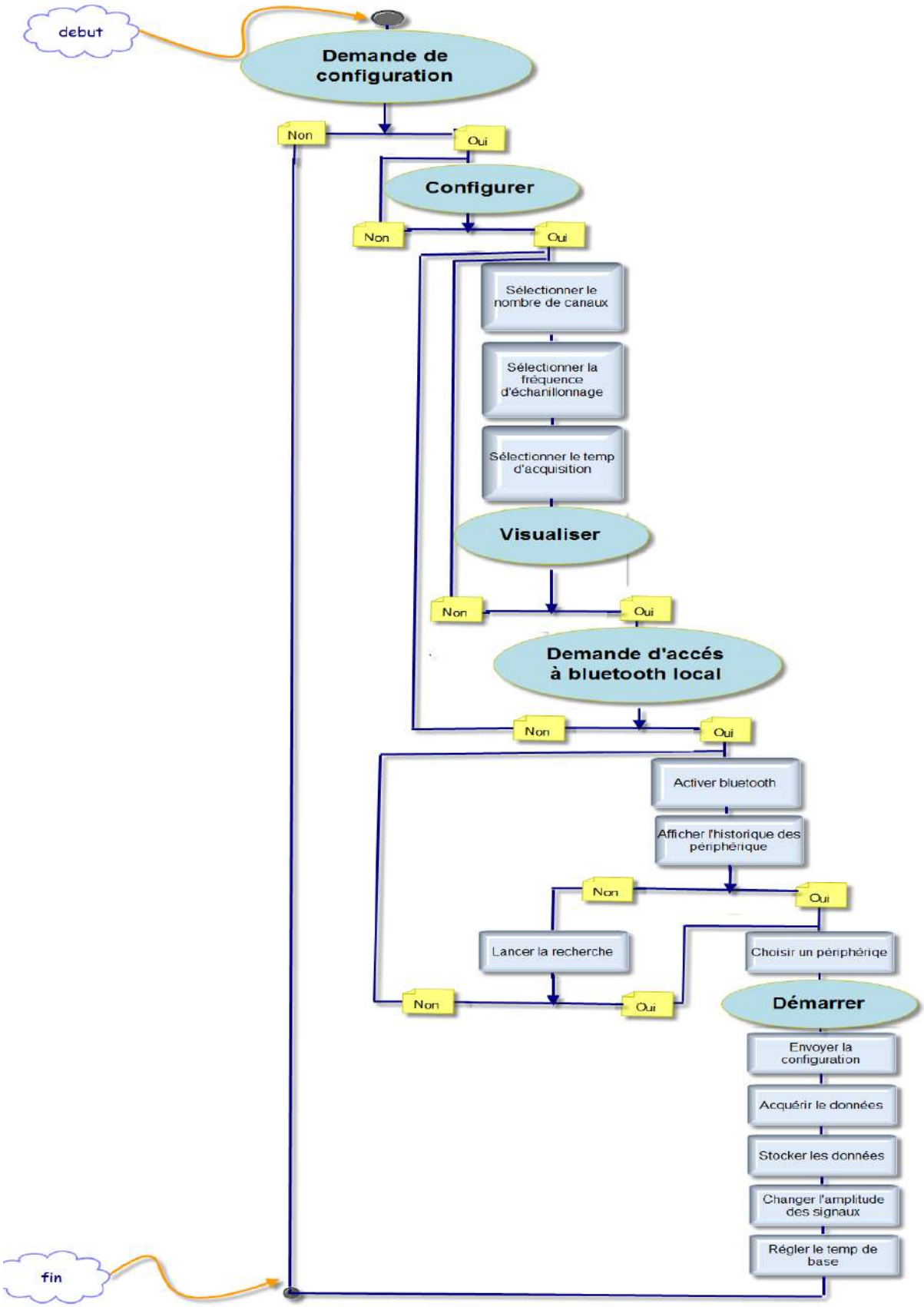


Figure II- 1 : « L'organigramme de l'application »

- Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation est un moyen qui permet de décrire les besoins des acteurs du système. C'est un diagramme qui sert à modéliser un des aspects statiques du système en passant du général au spécifique pour mettre en relief les besoins déjà énoncés d'une manière détaillée. [6]

Le diagramme illustré dans la figure suivante (Figure II-2) représente le cas d'utilisation globale de l'application sans en montrer les détails, l'analyse de ces cas d'utilisation sera détaillée plus bas. On vas détailler par la suite quelque cas d'utilisation.

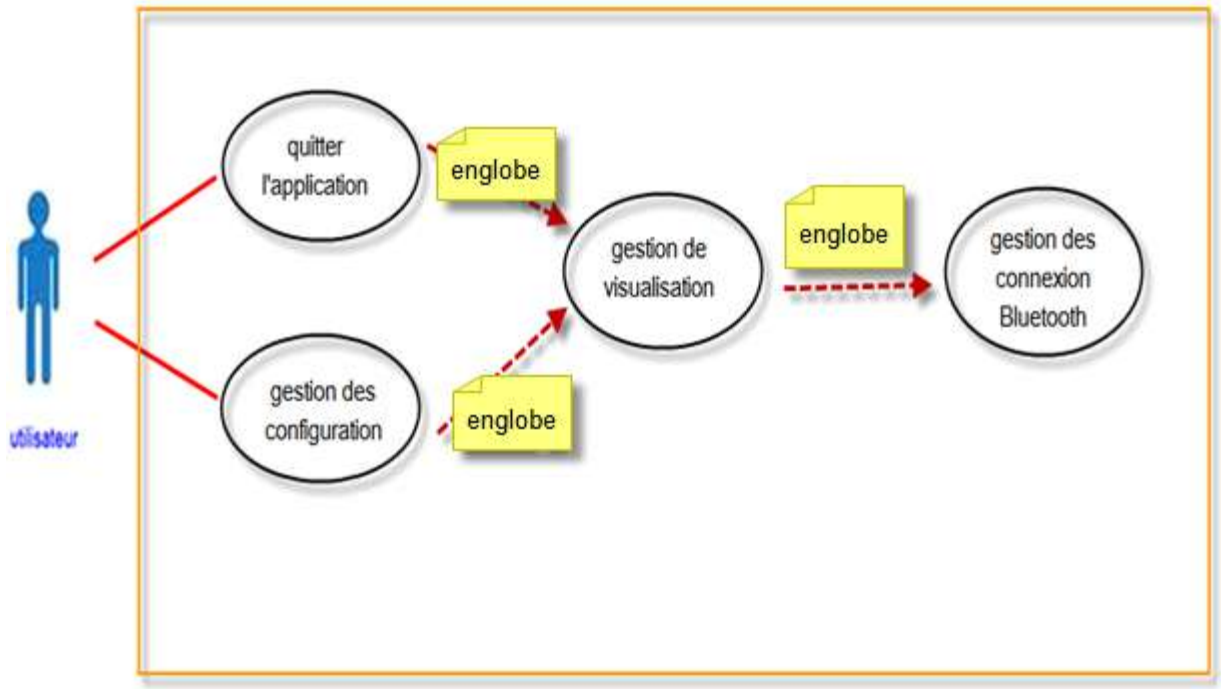


Figure II- 2 : « Diagramme de cas d'utilisation globale »

❖ Description des cas d'utilisation :

Afin de décrire les interactions entre les cas d'utilisation, nous présentons ces derniers de façon textuelle. Il s'agit donc d'associer à chaque cas d'utilisation un nom, un objectif, les acteurs qui y participent, les pré-conditions et des scénarios. Cependant ; il existe trois types de scénario :

- Les scénarios nominaux ;
- Les scénarios d'exceptions;
- Les scénarios alternatifs.

Dans notre description textuelle, nous présentons seulement les scénarios nominaux et alternatifs.

- Gestion des configurations :

- Analyse Cas d'utilisation « sélectionner le nombre de canaux » :

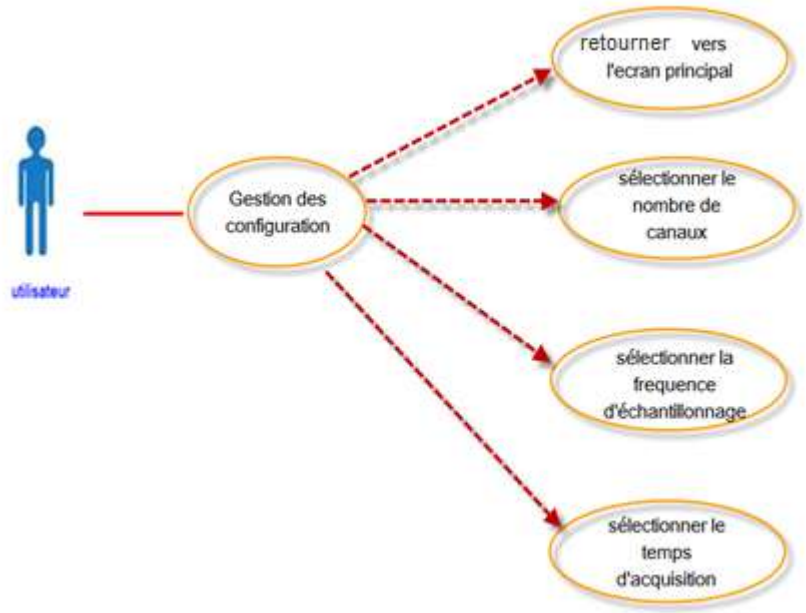


Figure II- 3 : « Diagramme de cas d'utilisation de « gestion des configurations »

Le tableau II.1 est un exemple de l'analyse de cas d'utilisation « Sélectionner le nombre de canaux » :

Titre :	Sélectionner le nombre de canaux
But :	Choisir le nombre de signaux à acquérir.
Acteur :	Utilisateur
Pré conditions :	Le nombre choisi doit être inferieur ou égale à 4.
SCENARIO NOMINAL :	
Ce cas d'utilisation commence lorsque l'utilisateur demande au système de choisir une configuration. - L'utilisateur entre un numéro dans le champ de saisi.	
ENCHAINEMENT ALTERNATIF :	
E1 : l'utilisateur peut garder le même nombre de canaux proposé par le système par défaut, et c'est la valeur 1 canal. E2 : l'utilisateur entre un nombre supérieur à 4 : - Le système affiche un message d'erreur : NB: « le nombre de canaux doit êtres inferieur à 4 »	

- Gestion de visualisation des données :

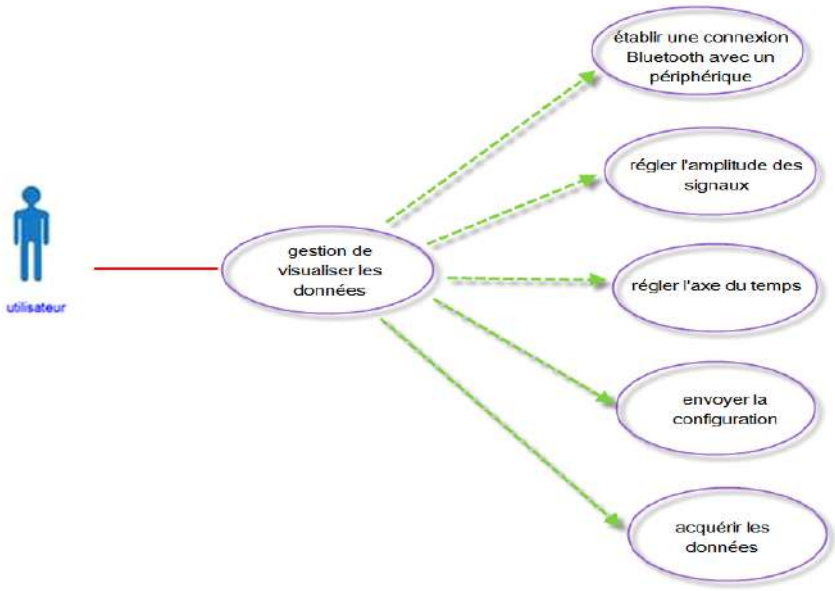


Figure II- 4 : « Diagramme de cas d'utilisation de « gestion de visualisation »

- Prenons l'Analyse Cas d'utilisation « Etablir une connexion Bluetooth avec le système d'acquisition de données. Le tableau II.2 est un exemple de l'analyse de cas d'utilisation « Etablir une connexion Bluetooth avec le système d'acquisition de données» :

Titre :	Etablir une connexion Bluetooth avec le système d'acquisition de données
But :	Connecter au système d'acquisition de données.
Acteur :	utilisateur
Pré conditions :	Le Bluetooth local du Smartphone doit être activé
SCENARIO NOMINAL :	
Ce cas d'utilisation commence lorsque l'utilisation demande au système de se connecter avec le système d'acquisition. 1- L'utilisateur clique sur le bouton « connecter ». 2- Le système affiche une liste des périphériques déjà existants dans son historique. 3- L'utilisateur lance une nouvelle recherche des équipements Bluetooth si l'historique ne contient pas l'équipement voulu. 4- L'utilisateur sélectionne le nom du Bluetooth du système d'acquisition de données. 5- La connexion entre l'application et le système est établie 6- Le system affiche un message : « reliée à : ... » NB : les point après « reliée à: ... » : sont le nom du Bluetooth du système d'acquisition.	
ENCHAINEMENT ALTERNATIF :	
E1 : l'utilisateur refuse l'activation du Bluetooth local avant qu'il sélectionne le nom du Bluetooth : - Le système affiche un message : « Bluetooth na pas été activée... » - L'application se ferme. E2 : l'utilisateur n'a pas été connecté au système d'acquisition de donnés : - Le système affiche un message : « non connecté ».	

- Gestion de connexion :

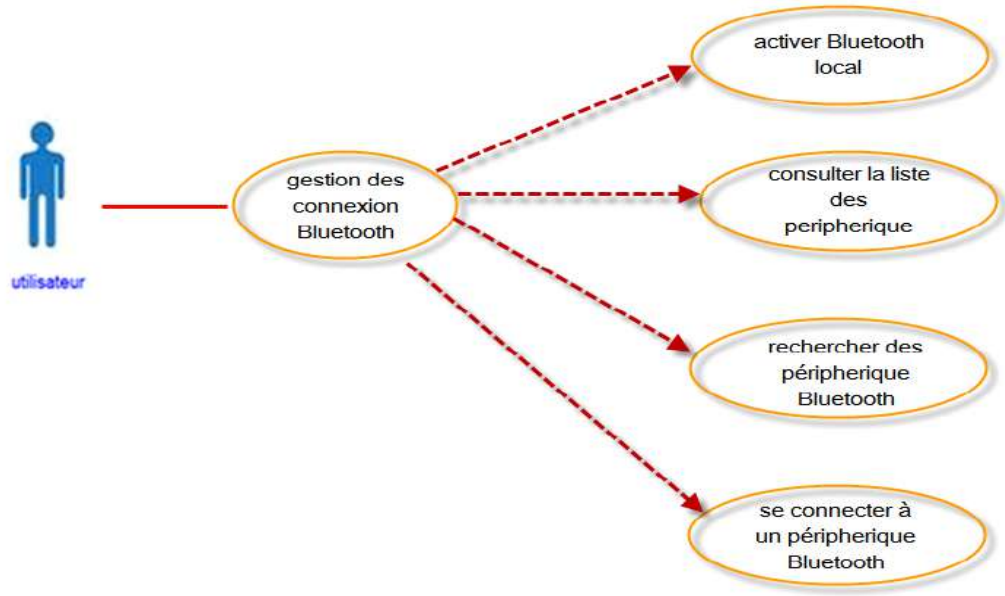


Figure II- 5 : «Diagramme de cas d'utilisation de « gestion de connexion Bluetooth »

II-3- Description :

Cette section nous a permis de couvrir les différents besoins fonctionnels et non fonctionnels des acteurs de notre système. Nous avons fourni une analyse plus détaillée de ces besoins grâce à un diagramme de cas d'utilisation relatif à l'acteur unique réagissant avec le système qui est le propriétaire du téléphone Android.

Nous essayerons dans les paragraphes qui suivent de décrire clairement l'architecture de notre système.

III.Conception de l'application :

Nous présentons à travers cette section, notre conception réalisée pour aboutir aux objectifs visés dans le cadre du projet.

III-1- Liste des activités :

La figure suivante (figure II-3) montre la liste des activités utilisées dans notre application Android :

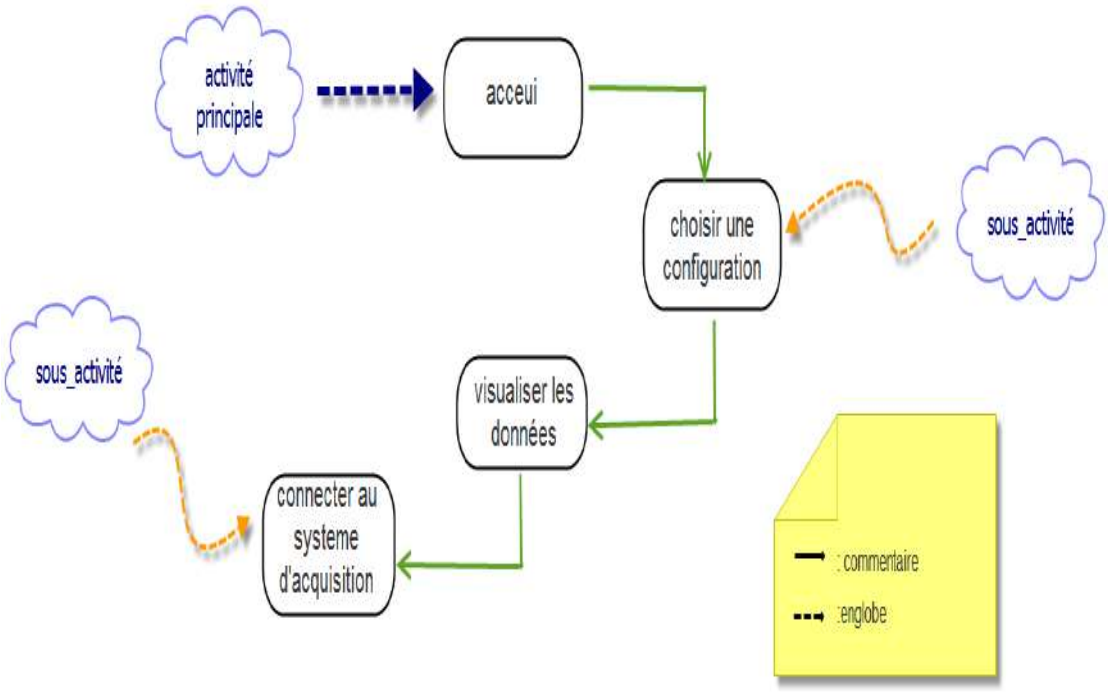


Figure II- 6 : « Liste des activités »

III-3- Architecture logiciel :

Nous allons faire ici une brève description des différents éléments de notre architecture logicielle :

- Main : Cette classe représente la vue principale de notre application. Elle contient un moteur qui permet aux classes dont elle dépend de pouvoir accéder et modifier indirectement les éléments de l'interface graphique.
- Configuration : cette classe permet à l'utilisateur de configurer le système d'acquisition de données.
- serviceBT : Classe réalisant les opérations liées à l'utilisation du Bluetooth et le stockage des données acquises.
- listeBT : activité utilisée avec son écran spécifique pour visualiser les listes des périphériques déjà apparus, ceux détectés dans le voisinage et demander la connexion.
- Visualisation : activité qui est responsable d'envoyer la configuration et visualisé les données acquises.

IV. Implémentation de l'application :

Dans cette partie, nous allons définir les points suivants :

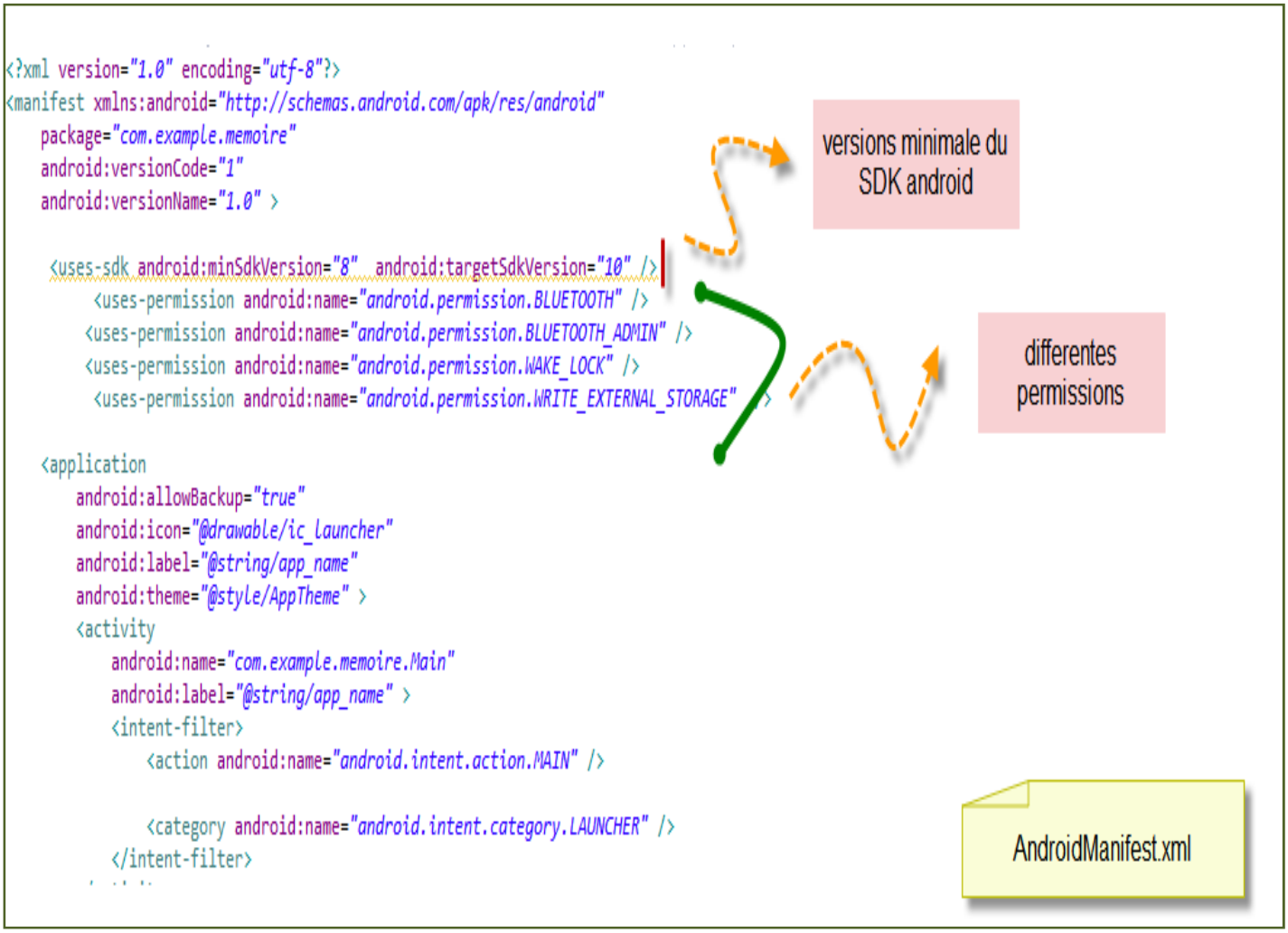
IV-1- Définition des droits :

Pour que notre application puisse utiliser le Bluetooth et l'accès à la carte mémoire du Smartphone pour le stockage des données, nous devons lui donner ces différentes permissions :

- La permission BLUETOOTH pour effectuer une quelconque communication Bluetooth, par exemple une demande de connexion, l'acceptation d'une connexion, le transfert des données, ...etc.
- La permission BLUETOOTH_ADMIN qui servent à initier la découverte de périphériques et la manipulation des paramètres Bluetooth comme l'activation ou la désactivation du Bluetooth, l'activation ou la désactivation de la visibilité du Bluetooth à partir de l'application et toute autre configuration liée au fonctionnement de la puce Bluetooth.
- La permission WAKE_LOCK pour désactiver le mode veille.
- La permission WRITE_EXTERNAL_STORAGE pour autoriser le stockage des données dans la carte mémoire externe du Smartphone.

Toutes ces différentes permissions sont définies dans le fichier manifest de notre application : AndroidManifest.xml.

La figure II-4 montre les différentes permissions et contraintes (versions minimale du SDK nécessaire pour faire fonctionner l'application) que notre application demande dans ce fichier.




```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.memoire"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk android:minSdkVersion="8"  android:targetSdkVersion="10" />
        <uses-permission android:name="android.permission.BLUETOOTH" />
        <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
        <uses-permission android:name="android.permission.WAKE_LOCK" />
        <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.example.memoire.Main"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

Figure II- 7 : « Fichier AndroidManifest.xml »

IV-2- Recherche des équipements :

Pour assurer la communication entre l'application et le système d'acquisition par Bluetooth, il faut d'abord que l'application initie la recherche de l'autre avant de lui envoyer une demande de connexion.

Pour cela nous avons crée la classe « listBT » afin de choisir le périphérique à connecter avec le Smartphone. Elle est ouverte par l'activité « visualisation » qui réagira aux choix effectués via l'interface "onClickListener".

L'activité liste les périphériques Bluetooth déjà associés et découvre sur demande les périphériques à portée. L'utilisateur choisit ensuite le périphérique dans les listes affichées (figure II.8):

```
    //Obtenir l'adaptateur Bluetooth local
    mBtAdapter = BluetoothAdapter.getDefaultAdapter();

   //-----------------------------------------------------------------------------------------

    // Obtenir un ensemble de dispositifs actuellement liés
    Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

  //-----------------------------------------------------------------------------------------
    //S'il ya des appareils appariés, ajouter chacun des ArrayAdapter
    if (pairedDevices.size() > 0) {
        findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
        for (BluetoothDevice device : pairedDevices) {
            mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    } else {
        String noDevices = getResources().getText(R.string.none_paired).toString();
        mPairedDevicesArrayAdapter.add(noDevices);
    }
}
//-----------------------------------------------------------------------------------------
```

Figure II- 8 : « Recherche des équipements »

IV-3- Pairage et connexion de deux équipements :

Le processus de connexion est défini dans le thread " tconnect ". Ce thread est responsable de gérer la mise en connexion Bluetooth du Smartphone avec le Bluetooth du système d'acquisition et de renvoyer des messages (Intent) à l'activité principale. Celle-ci indique l'évolution de la connexion dans son titre.

Pour ouvrir une connexion, nous utilisons la liste des adresses des périphériques appariés. On récupère donc une liste de BluetoothDevice qui va nous permettre de faire appel à la méthode createRfcommSocketToServiceRecord avec pour paramètre l'UUID (universally unique identifier) adéquat (c-à-d celui du périphérique avec lequel nous voulons nous connecté).

Cette méthode renvoie un BluetoothSocket que nous avons appelé SoketClient qui nous permet donc d'initier la connexion. (Figure II.9)

```
private class tconnect extends Thread {
    private final BluetoothSocket mmSocket;
    private final BluetoothDevice mmDevice;
    public tconnect(BluetoothDevice device) {
        mmDevice = device;
        BluetoothSocket SoketClient = null;

        //Obtenir un BluetoothSocket pour une connexion avec le BluetoothDevice
        try {

            SoketClient = device.createRfcommSocketToServiceRecord(MY_UUID);
        } catch (IOException e) {
            //
```

Figure II- 9 : « Pairage et connexion de deux équipements »

IV-4- Envoi de la configuration :

Le thread tconnected est chargé de gérer la connexion Bluetooth, notamment les échanges avec l'activité principale à la réception et la transmission des données.

Dans le cas ou l'application envoi la configuration au système d'acquisition on a utilisé la méthode sensMessage de la classe visualisation. Cette méthode utilise elle-même une autre méthode write de la classe serviceBT, comme la montre la figure II.10.

visualisation.java

```
private void sendMessage(String message) {
    //Assurez-vous que nous sommes réellement connecté
    if (mbtservice.getState() != serviceBT.STATE_CONNECTED) {
        Toast.makeText(this, R.string.not_connected, Toast.LENGTH_SHORT).show();
        return;
    }
//  Vérifier qu'il ya réellement quelque chose à envoyer
    if (message.length() > 0) {
        //Obtenez les octets du message et dire au mbtservice à écrire
        byte[] send = message.getBytes();
        mbtservice.write(send);
    }
}
```

serviceBT.java

```
public void write(byte[] out) {
    // Création d'un objet "tconnected" temporaire
    tconnected r;

    synchronized (this) {
        if (mState != STATE_CONNECTED) return;
        r = mtconnected;
    }
 // Ecriture effective du tableau d'octets
    r.write(out);
}
```

Figure II- 10 « Envoi des données »

IV-5- Réception des données :

La Méthode "run" du thread tconnected est appelée régulièrement jusqu'à la rupture de la connexion.

```
   // Lecture des octets reçus depuis le "InputStream"
   while ((bytes =  mmInStream.read(buffer)) > 0) {
  // bytes = mmInStream.read(buffer);

// Transmission du nb d'octets et du pointeur du buffer ds l'activité principale
// qui traite les octets reçus
   mHandler.obtainMessage(application.MESSAGE_READ, bytes, -1, buffer)
           .sendToTarget();
catch (IOException e) {
   //// Perte de connexion en cas d'exception
   connectionLost();
```

Figure II- 11 : « Réception des données »

IV-6- Stockage des données :

Les données reçus sont stoker dans un fichier d'extension « .txt » dans un dossier qui s'appel « acquis data ».

A chaque exécution de « run », un nouveau fichier est crée, avec un nom est la date du Smartphone ou le fichier à été crée.

Remarque : l'acquisition de données et le stockage sont exécutés en parallèle.

```
   // Lecture des octets reçus depuis le "InputStream"
   while ((bytes =  mmInStream.read(buffer)) > 0) {
  // bytes = mmInStream.read(buffer);

// Transmission du nb d'octets et du pointeur du buffer ds l'activité principale
// qui traite les octets reçus
   mHandler.obtainMessage(application.MESSAGE_READ, bytes, -1, buffer)
           .sendToTarget();
catch (IOException e) {
   //// Perte de connexion en cas d'exception
   connectionLost();
```

Figure II- 12 : « Stockage des données »

IV-7- Visualisation des données :

Après la réception des données, elles sont représentées sur une grille à deux dimensions (10 x 10 carreaux):

- Axe des amplitudes.
- Axe des temps.

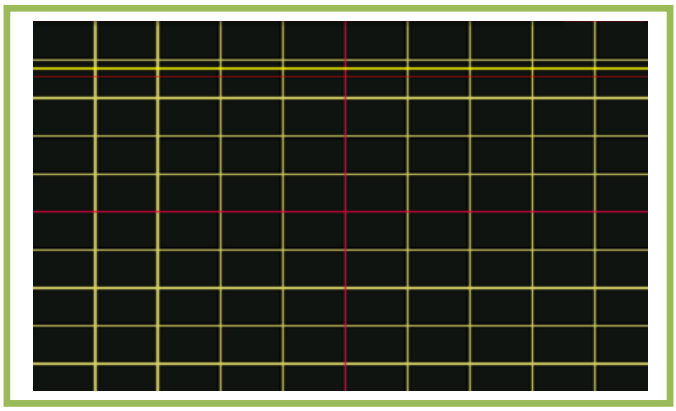


Figure II- 13 : « L'interface de la visualisation »

Pour pouvoir dessiner cette grille on a profité de la classe "Canvas" qui comporte toutes les méthodes de dessin.

```
  // dessiner les 2 axe
canvas.drawLine(0, (heigth/2)+1, width+1, (heigth/2)+1, cross_paint)
canvas.drawLine((width/2)+1, 0, (width/2)+1, heigth+1, cross_paint);

// bordure de la grille
canvas.drawLine(0, 0, (width+1), 0, outline_paint);
canvas.drawLine((width+1), 0, (width+1), (heigth+1), outline_paint);
canvas.drawLine(0, (heigth+1), (width+1), (heigth+1), outline_paint)
canvas.drawLine(0, 0, 0, (heigth+1), outline_paint);
```

Figure II- 14 : « Utilisation de la classe Canvas dans l'interface de visualisation »

On a utilisé le changement d'échelle d'amplitude et aussi celui de la base de temps afin de contrôler la visualisation des signaux.

- La classe WaveformView récupérer le tableau de échantillons de chaque canal grâce aux méthodes set_dat, set_data1, set_data2, et set_data3. Ces fonctions sont utilisées selon le nombre de canaux utilisé dans la configuration. Cette méthode à comme paramètre le/les tableaux d'échantillons et l'amplitude choisi dans la configuration.

```
public void set_data(int[] data1,double ampl){

    int x;
    plot_thread.setRunning(false);
    x = 0;
    while(x<width){
        if(x<(data1.length)){
            //ch1_data[x] = data1[x];
            ch1_data[x] = (int) ((heigth-data1[x]+1)/ampl);
        }else{
            ch1_data[x] = ch1_pos;
        }
        x++;
    }
```

Figure II- 14 : « La méthode set-data »

Remarque : La création de cette interface a été inspirée d'un projet d'un oscilloscope [7].

V. Conclusion :

- Au cours du développement, nous avons été confrontés à certaines difficultés liées à la programmation dans java Android concernant le passage des données entre la différente classe.
- Vu que l'émulateur Android ne supporte pas la fonctionnalité Bluetooth, nous étions obligées de faire nos tests avec un vrai téléphone.



Chapitre III: Mise en marche et Tests





Mise en marche et Tests

I. Introduction :

Nous présentons dans ce chapitre les captures d'écrans de quelques interfaces principales de notre application qui illustrent les différents cas d'utilisation déjà vus dans le chapitre précédent, en plus des tests effectués avec une carte électronique Android.

II. Les cartes électroniques Android :

Depuis quelques années les constructeurs de cartes électroniques proposent divers modèles cartes électroniques ineffaçables aux dispositifs fonctionnant sous Android, telles que les cartes présentées sur le tableau suivant :

Modèle	Constructeur	Processeur	Périphériques	Prix ($)
Arduino ADK	Arduino	ATmega 2560	Usb Host, E/S tout ou rien, sorties PWM, Entrées analogiques, liaison SPI, I2, UART...	55
Android Open accessory application	NXP	ARM Cortex M0 (LPC1769)	USB Host; USB device; Port Ethernet 10/100 Mbps; Interface CAN ; 8 ports d'E/S (dont 4 utilisables en entrées analogiques); connecteur pour carte mémoire micro SD; Connecteur JTAG	72
IOIO-OTG	Sparkfun Electronics	Pic 24FJ256	Usb OTG; sorties PWM, Entrées analogiques, E/S numériques ; liaisons: SPI, I2C, UART; ...	39
SBC CubieBoard Android/ Linux Ubuntu	SBC	ARM Allwinner A10	Video: HDMI; Audio: 3.5mm jack, Port Ethernet 10/100; Stockage: µSD, SATA; USB (Host/OTG); IR ; liaisons I2C; SPI; ADC; Touch-panel	52

Tableau III- 1 : Quels que modèles de cartes électroniques Android

III. Présentation de la carte utilisée [8]:

Dans notre projet nous avons opté pour l'utilisation de la carte électronique IOIO, (prononcé « yoyo ») qui est une carte électronique commercialisée par Sparkfun Electronics, elle se connecte à un périphérique Android via la prise USB ou via liaison Bluetooth. Cette carte permet de relier un périphérique Android (PDA, Smartphone, Tablette, ...etc) au monde extérieur, il peut ainsi récupérer des informations en provenance de capteurs, ou même piloter des éléments extérieurs. (Figure III.2).

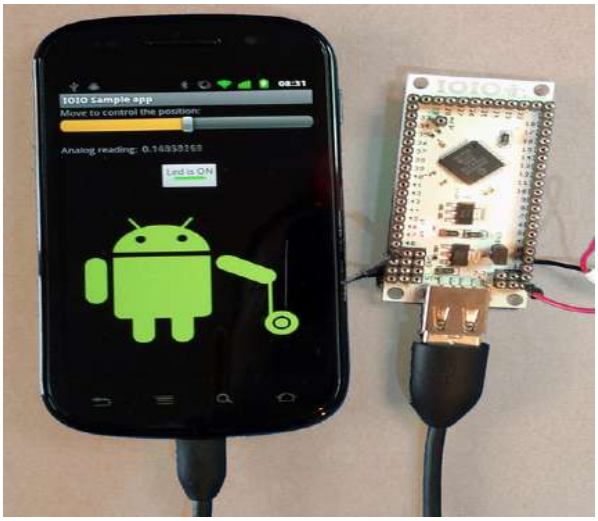


Figure III- 1: « La carte IOIO associée à un AndroPhone»

Ce choix à été due en grande partie aux raisons de disponibilités, car c'est la seule carte que nous avons pus l'acquérir, en plus son caractère Open source (hardware et Software), sa simplicité, ses faibles dimensions et ses divers périphériques.

La carte IOIO-OTG est un module d'interface spécialement conçu pour être piloté via un dispositif Android (avec version d'OS 1.5 et supérieur) par le biais de son port USB ou via une liaison Bluetooth. Ce pilotage s'effectuera via des API JAVA simples et intuitives sans avoir recours à une programmation embarquée complexe, ni au moindre programmateur hardware externe. Le processeur embarqué du module IOIO interprétera ainsi les commandes issues de l'application Android.

Cette carte électronique est construite autour d'un microcontrôleur PIC série 24F, qui dispose d'une connexion USB hôte. Il suffit donc de la relier à l'aide d'un câble USB à un périphérique Android (OS v1.5 minimum) pour que la carte IOIO interprète des commandes reçues par une application donnée.

La carte IOIO dispose d'un grand nombre d'entrées/sorties, on retrouve donc 48 pins d'entrées/sorties malgré sa petite taille (70 x 30 mm) dont :

- 16 entrées analogiques (CAN de résolution 10 bits);
- 9 sorties PWM;
- Jusqu'à 4 liaison série type UART;
- Jusqu'à 3 liaison série type SPI;
- Jusqu'à 3 liaison série type TWI (compatible I2C);

La carte dispose également d'un régulateur de tension 5V débitant un courant jusqu'à 1.5 A permettant de recharger le périphérique Android lors de la connection USB, cela peut permettre aussi d'alimenter des petits moteurs par exemple, en plus d'un régulateur 3.3 V.

Au niveau du fonctionnement sur le périphérique Android, il suffit d'installer une application (JAR) qui permet de piloter les entrées/sorties de carte IOIO. Il est également possible de réaliser ses applications dédiées à l'aide du SDK Android pour plus de souplesse.

L'intégrité de ce projet est 100% open-source se qui permet de disposer des moindres détails de la partie hardware et de la partie logicielle.

La figure suivante montre la vue de face et celle d'arrière de la carte IOIO :

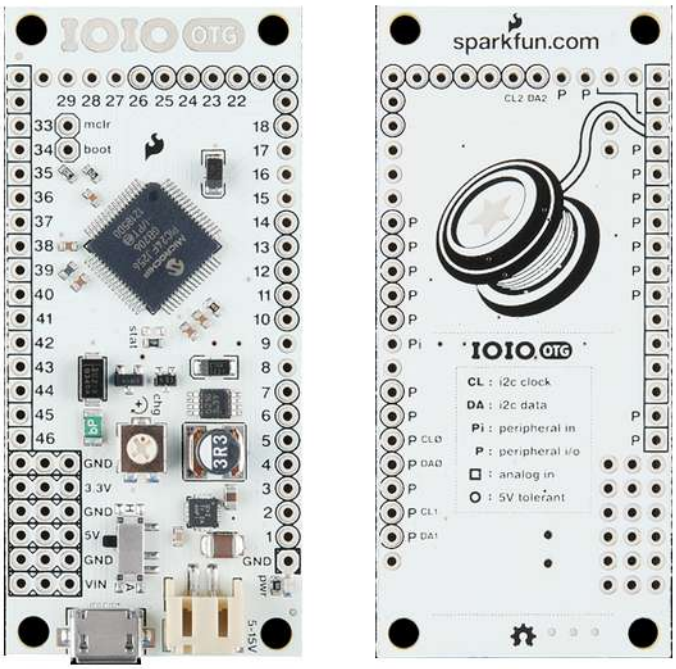


Figure III- 2: « Vue de face et d'arrière de la carte IOIO»

III.2 Les avantages de la carte IOIO :

Parmi les avantages de cette carte nous citons :

1. Nettement plus petit en taille que la plus part des cartes Android;
2. Faible cout;
3. Riches en fonctionnalités;
4. Open-source;
5. Puissance de calcul assez grande;
6. Package facile à utilisé;

IV. Mise en marche de l'application :

Afin d'utiliser notre application il faut suivre la démarche suivante :

IV.1 Chargement de l'application dans le Smartphone :

Apres l'exécution de l'application dans l'environnement de développement Eclipse, l'application s'installe automatiquement dans le Smartphone avec son nom comme la montre la figure III- 4.



Figure III- 3 :« Apparition de l'application dans le Smartphone »

IV.2 Démarrage de l'application: « my_pfe »

Lors de l'exécution de notre application une interface d'accueil démarre comme le montre la figure III.5.

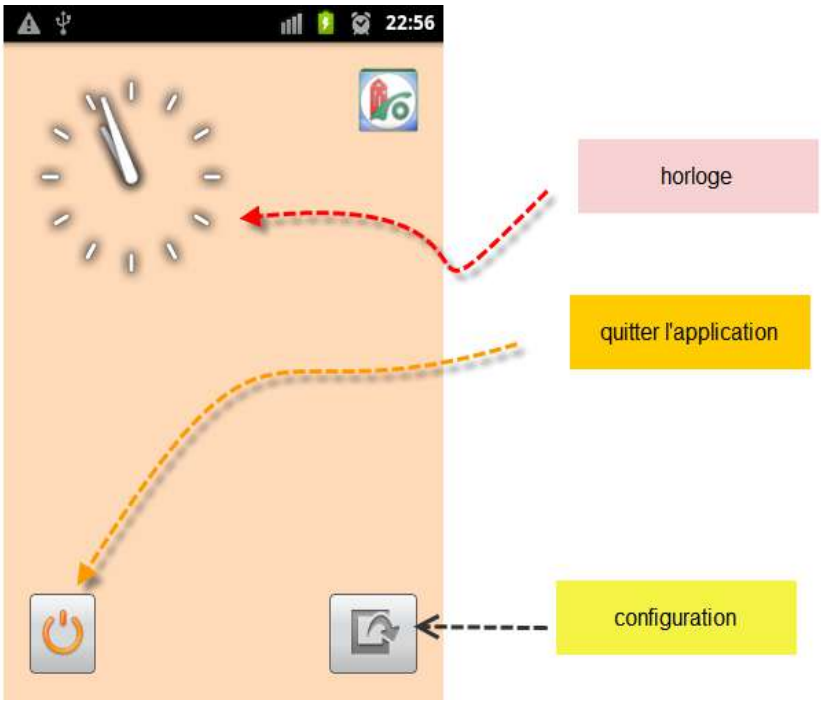


Figure III- 4 « L'interface d'accueil de l'application Android réalisée »

Dans cette interface on a deux choix, soit on quitte l'application, ou bien on accède à l'interface suivante. On a utilisé des boutons significatifs connus par la plus part des utilisateurs de Smartphones afin de la rendre plus souple et plus conviviale.

IV.3 Configuration du système d'acquisition :

Une fois l'utilisateur accède à l'interface de configuration il trouve trois champs à saisir, (figure III.6).

Le premier champ concernant le nombre de cannaux (de signaux) souhaité par l'utilisateur de l'application. Le deuxième est utilisé pour donner la fréquence d'échantillonnage en kHz. Et Le dernier champ permet de configurer la durée d'acquisition en minutes.

Si l'utilisateur ne fait pas ses choix de configuration et passe à l'écran suivant, l'application prend applique les valeurs par défaut.

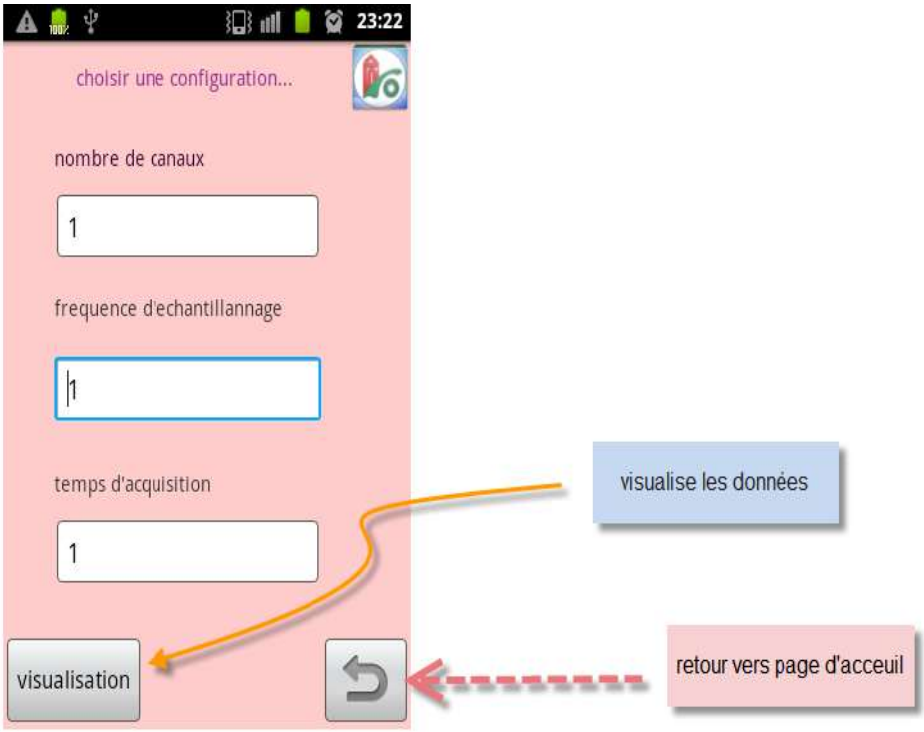


Figure III- 5: « Les valeurs par défaut de la configuration »

IV.4 Réceptions de données :

Dans un premier temps, l'application demande l'autorisation de l'activation du Bluetooth (figure III.7) :

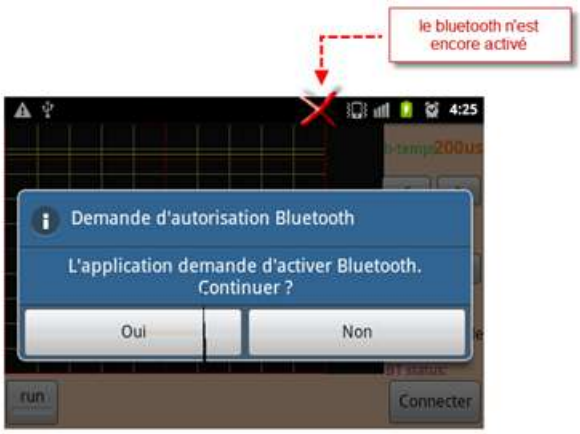


Figure III- 6: « Demande d'autorisation d'activation du Bluetooth local »

Si l'utilisateur refuse l'activation du Bluetooth (figure III.8) un message s'affichera pour quitter cette interface et l'application retourne vers l'interface de configuration (figure III.9).

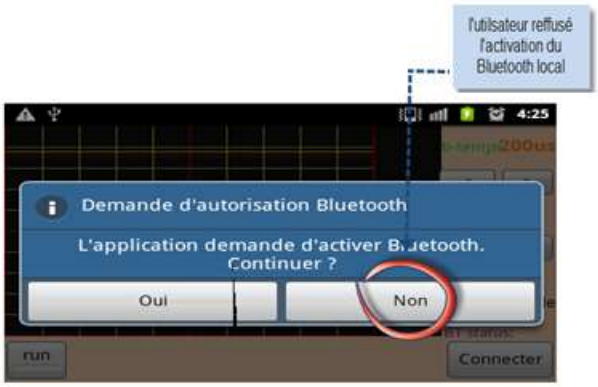


Figure III- 7: « L'utilisateur refuse l'activation de son Bluetooth »

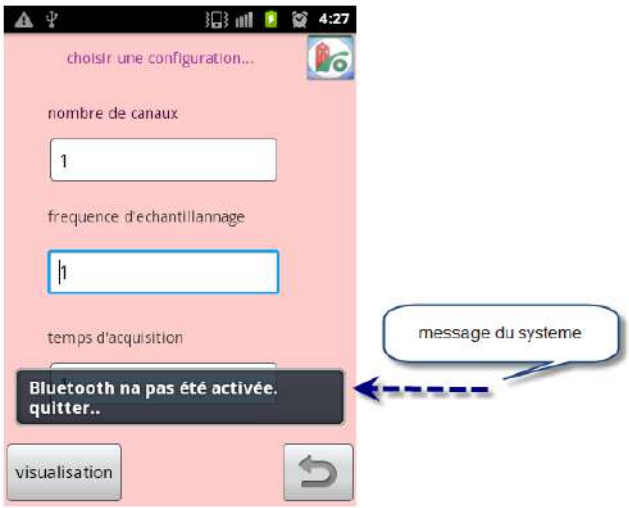


Figure III- 8: « Sortie de l'interface de visualisation »

Si le Bluetooth local du téléphone s'activé on obtient l'interface suivante : (Figure III.10)

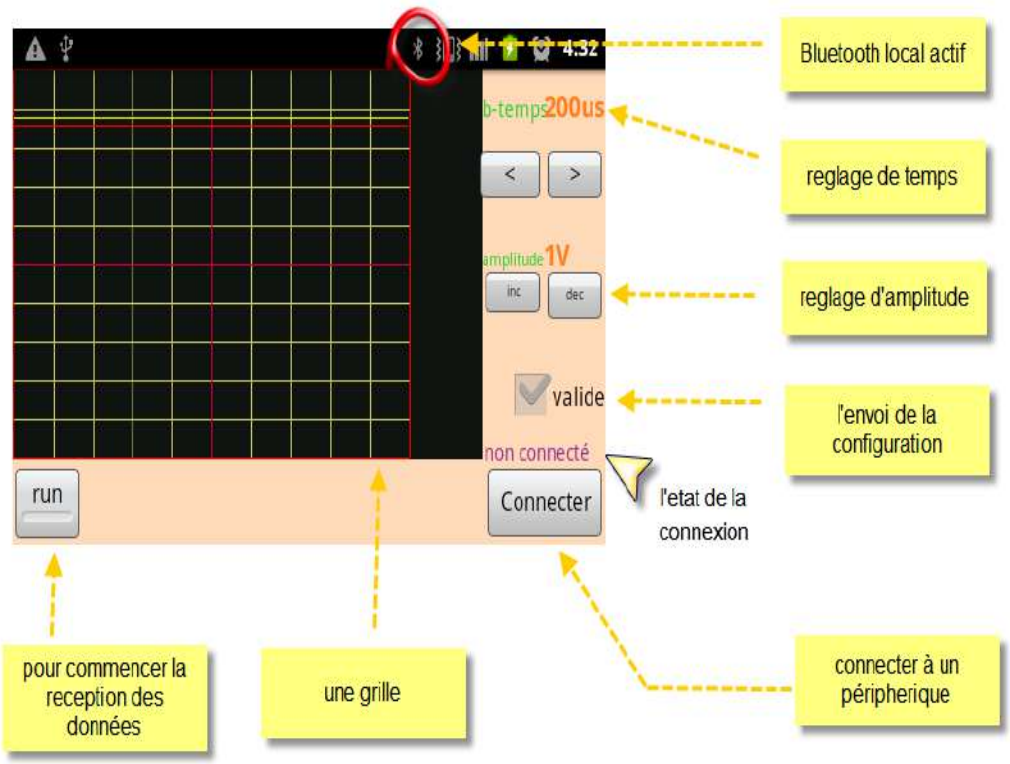


Figure III- 9: « L'interface de visualisation »

Si l'utilisateur appuie sur la touche « run » ou cocher le champ valide, le système affiche un message que le dispositif Android n'est pas encore reliée au périphériques Bluetooth d'acquisition (Figure III.10).

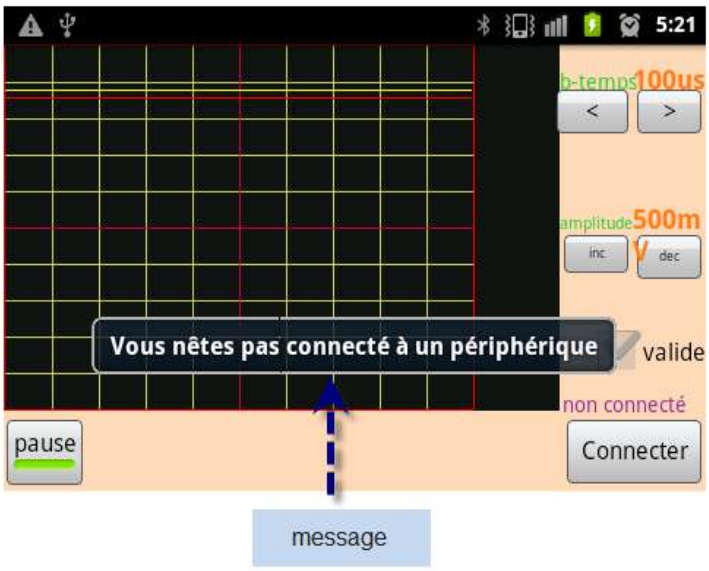


Figure III- 10: « Le message d'alerte lié à l'établissement de la connexion entre les deux périphériques»

IV.5 Recherche des périphériques Bluetooth :

Les figures III.12 et III.13 montrent les différentes démarches pour établir une connexion Bluetooth entre le Smartphone la carte Android d'acquisition des données:

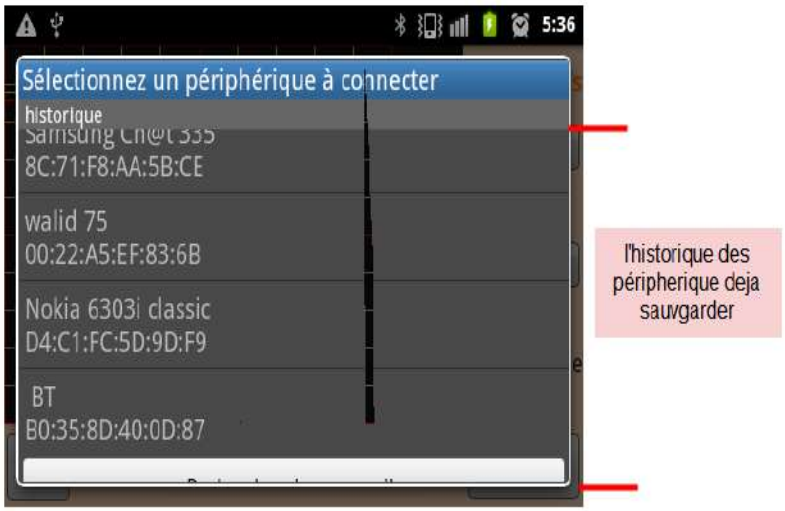


Figure III- 1: « Liste des périphériques déjà sauvegardés »

- L'utilisateur cherche un périphérique parmi la liste de son historique :

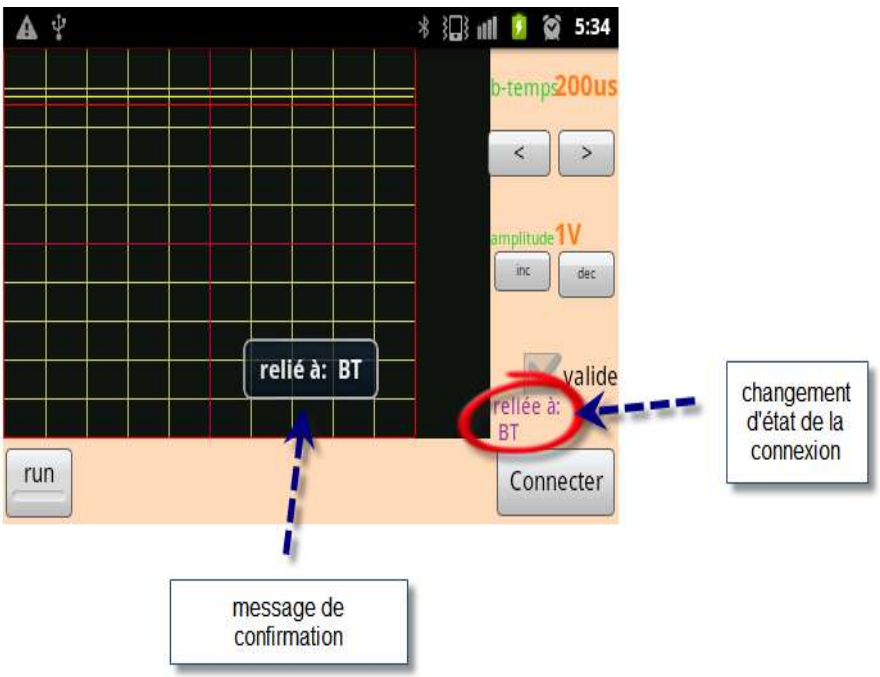


Figure III- 2: « L'établissement de la connexion Bluetooth »

- Dans le cas d'un problème hardware ou de perte de connexion, le système affiche un message et le retourne vers l'état « non connecté » (figure III.14).

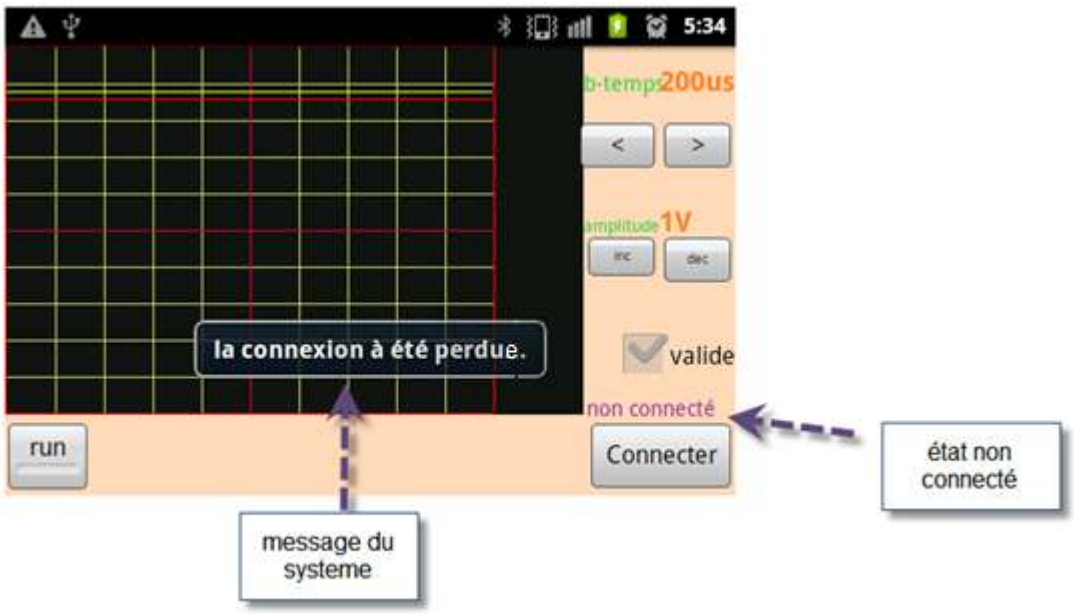


Figure III- 3: « Message d'alerte de perte de la connexion Bluetooth»

- Dans le cas ou l'utilisateur n'a pas trouver son bonheur dans l'historique il lance une recherche des appareils Bluetooth.



Figure III- 4: « Recherche des périphériques Bluetooth »

- Si le système n'a pas trouvé d'autres appareils Bluetooth, un message s'affiche pour informer l'utilisateur qu'aucun périphérique n'est trouvé.

V. Test et résultat :

Le matériel qui a été utilisé pour réaliser les tests préliminaires est :

- Un Smartphone : on a utilisé un téléphone Samsung « galaxy ace » qui possède une version d'Android 2.3.3.
- Une Catre électronique IOIO : (la version IOIO OTG).

Les premier tests que nous avons réalisés on consistés à injectés un signal de forme, d'amplitude et de fréquence connues à l'une des entrées analogiques de la carte IOIO. Pour cela nous avons utilisé un générateur de fonction afin de générer le signal analogique à numérisé par la carte IOIO. La figure suivante montre le câblage utilisé lors des tests :

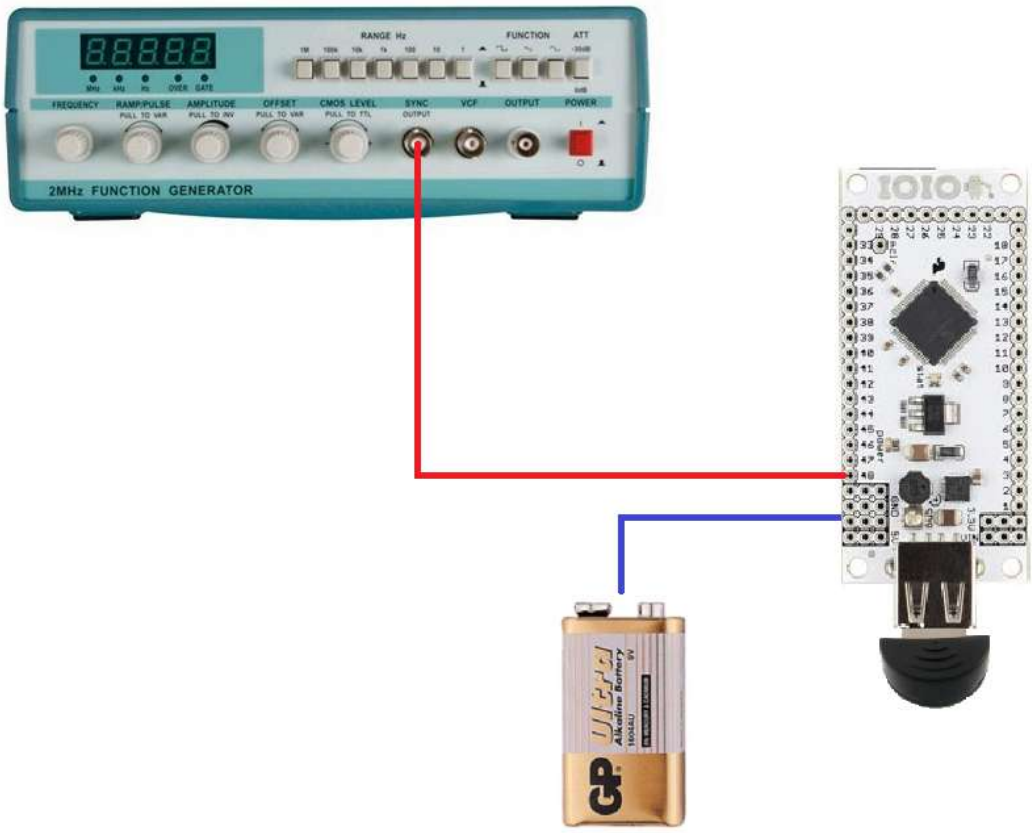


Figure III- 5: « Le test effectué avec un Générateur de fréquence »

Dans le premier test nous avons appliqué un signal sinusoïdal d'une amplitude de 2 V et d'une fréquence de 1 kHz, la figure III-17 montre la capture d'écran de visualisation sur le téléphone Android :

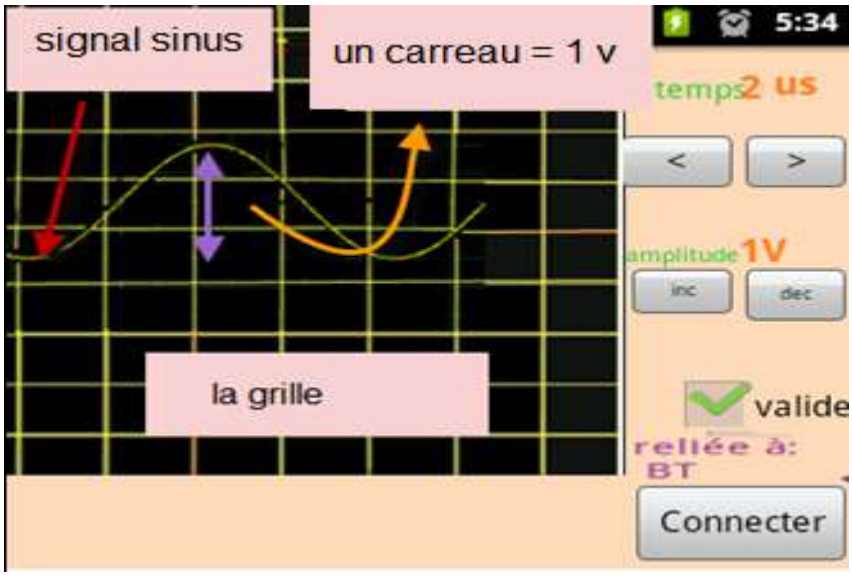


Figure III- 6: « Capture d'écran du premier Test »

Dans le deuxième test nous avons appliqué un signal carré d'une amplitude de 3 V et d'une fréquence de 10 kHz, la figure III-18 montre la capture d'écran de visualisation de ce signal :

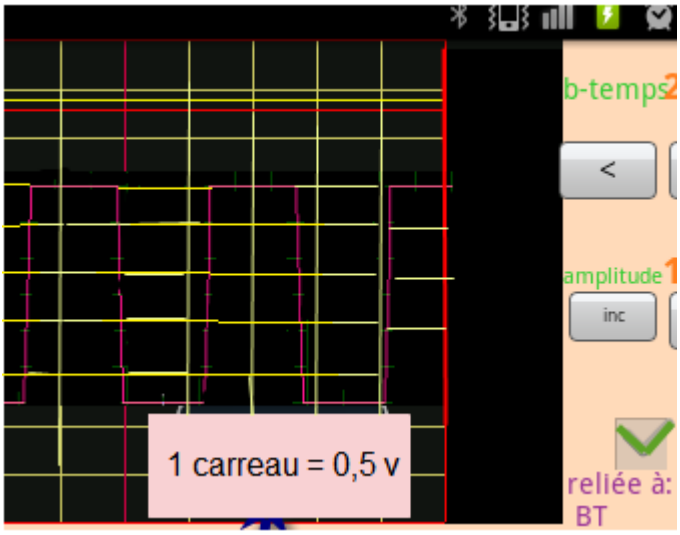


Figure III- 7: « Capture d'écran du deuxième Test »

VI. Conclusion :

Dans ce chapitre, nous avons présenté les choix techniques que nous avons adoptés, ainsi que l'environnement d'implémentation et quelques interfaces de l'application développée, en plus des résultats des premiers tests qui semblent satisfaisants en attendant d'effectués plus de tests pour connaitre les capacités et les limites de l'application.

Conclusion Générale

Notre projet a consisté à développer une application mobile sous Android pour la réception, la visualisation et le stockage de données de mesures d'un ensemble de capteurs interfacés à une carte électronique d'acquisition à base d'un microcontrôleur via la liaison Bluetooth.

Nous avons pus développer l'ensemble des fonctionnalités de base de l'application, l'intégration a été réalisée avec succès, c'est-à-dire que l'application est maintenant installée sur le Téléphone mobile et prête à être utilisée.

Ce projet nous a permis d'approfondir nos connaissances théoriques, acquises tous le long de notre formation, par la pratique de nouvelles technologies. Cette expérience nous a permis d'aborder le langage de modélisation UML pour les diagrammes de cas d'utilisation, les outils de développement Android à savoir le SDK Android, sous lequel, le développement n'a pas été une tâche facile. Il nous a également permis de découvrir comment se passe la communication Bluetooth entre deux périphériques Bluetooth.

Bien que les principaux objectifs de notre projet soient atteints, l'application que nous avons développée pourrait être enrichie par d'autres améliorations et fonctionnalités avancées pouvant être envisagées pour la perfectionner d'avantage, tel que la communication entre l'application et le système d'acquisition des données en utilisant d'autres moyens de communications tels que la liaison USB, la liaison Wifi,....etc.

Nous souhaitons, enfin, que ce modeste travail apporte aides et satisfactions aux étudiants des futures promos et toute personne intéressée de près ou de loin à la programmation d'applications électroniques sous Android.

Références bibliographiques

[1] J. KRUGER, « Opportunités d'entreprise avec Android », the international institue of sup info, promotion 2009.

[2] Site web: http://www-igm.univ mlv.fr/~dr/XPOSE2011/SDKAndroid/android.html#Historique

- Date de visite du site 10/06/2013 à 21.30.

[3] Site web: http://developer.android.com/about/dashboards/index.html

- Date de visite du site 15/05/2013 à 3 :15.

[4] N. KHYARI, « Rapport de stage sur le projet : Locate my car », école nationale des science de l'informatique, Tunisie, 2010.

[5] Résumé du sous-ensemblede la notation UML 2 http://webcache.googleusercontent.com/search?q=cache:QhF4o2_pUNwJ:izibook.eyrolles.com/extract/show/8382+R%C3%A9sum%C3%A9+du+sous ensemblede+la+notation+UML+2&cd=1&hl=fr&ct=clnk&gl=fr

- Date de visite du site 15/05/2013 à 10 :12.

[6] Site web : http://webcache.googleusercontent.com/search?q=cache:sycNcHSiGRQJ:www.redcad.org/members/tarak.chaari/cours/coursUML.pdf+%5B7%5D:+Cours+%C2%AB+Atelier+UML+%C2%BB+Tarak+Chaari&cd=1&hl=fr&ct=clnk&gl=fr

- Date de visite du site 16/05/2013 à 07 :50.

[7] Site web: http://projectproto.blogspot.com/2010/09/android-bluetooth-oscilloscope.html

[8] Site web: http://www.evola.fr/product_info.php/ioio-pour-android-p-177

- Date de visite du site 5/06/2013 à 22:30.

Annexe

code source de l'application :

Nous présentons quelque classe java et relier avec son interface XML :

```
package com.example.memoire;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.ImageButton;
import android.app.Activity;
import android.view.Menu;

public class Main extends Activity {

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		requestWindowFeature(Window.FEATURE_NO_TITLE);
		setContentView(R.layout.main);
		ImageButton exit = (ImageButton)findViewById(R.id.Button1);
		exit.setOnClickListener(new OnClickListener() {


			@Override
			  public void onClick(View arg0) {
			    finish();
			  }
			});

		ImageButton bouttonConfig =
(ImageButton)findViewById(R.id.Button2);
		bouttonConfig.setOnClickListener(new OnClickListener() {

		    @Override
		    public void onClick(View arg0) {
		      Intent i = new Intent(Main.this, configuration.class);
		      startActivity(i);
		    }
		});
```

Classe principale « main.java »

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background_color"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".Main" >

    <AnalogClock
        android:id="@+id/analogClock1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" />

    <ImageButton
        android:id="@+id/Button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:src="@android:drawable/ic_menu_set_as" />

    <ImageButton
        android:id="@+id/Button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignLeft="@+id/analogClock1"
        android:layout_alignParentBottom="true"
        android:layout_alignTop="@+id/Button2"
        android:src="@android:drawable/ic_lock_power_off" />

    <ImageView
        android:id="@+id/imageView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:src="@drawable/fac" />

</RelativeLayout>


}
}
```

- Le fichier XML associée à cette classe est :





I.2 La classe listBT.java :

```
package com.example.application;
import java.util.Set;
import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.AdapterView.OnItemClickListener;


public class listeBT extends Activity {
    // Debugging
    private static final String TAG = "listeBT";
    private static final boolean D = true;

    // Return Intent extra
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Member fields
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter<String> mPairedDevicesArrayAdapter;
    private ArrayAdapter<String> mNewDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Setup the window
        requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);
        setContentView(R.layout.device_list);

        // Set result CANCELED incase the user backs out
        setResult(Activity.RESULT_CANCELED);


        //Initialiser le bouton pour effectuer la détection des périphériques
        Button scanButton = (Button) findViewById(R.id.button_scan);
        scanButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                doDiscovery();
                v.setVisibility(View.GONE);
            }
        });
```





```
  //Initialiser adaptateurs de réseau. Un pour les appareils déjà jumelés et
        // Un pour les appareils nouvellement découverts
        mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);
        mNewDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);

        //Trouver et mettre en place le ListView pour les appareils associés
        ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);
        pairedListView.setOnItemClickListener(mDeviceClickListener);


        //Trouver et mettre en place le ListView pour les appareils nouvellement
découverts
        ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
        newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
        newDevicesListView.setOnItemClickListener(mDeviceClickListener);


        /// / Enregistrer des émissions lorsqu'un périphérique est découvert
        IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
        this.registerReceiver(mReceiver, filter);
/// / Enregistrer des émissions lors de la découverte a fini
        filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        this.registerReceiver(mReceiver, filter);

       //Obtenir l'adaptateur Bluetooth local
        mBtAdapter = BluetoothAdapter.getDefaultAdapter();


        /// / Obtenir un ensemble de dispositifs actuellement liés
        Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();


        //S'il ya des appareils appariés, ajouter chacun des ArrayAdapter
        if (pairedDevices.size() > 0) {
            findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
            }
        } else {
            String noDevices =
getResources().getText(R.string.none_paired).toString();
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }
@Override
    protected void onDestroy() {
        super.onDestroy();
//Assurez-vous que nous ne faisons pas découverte plus
        if (mBtAdapter != null) {
            mBtAdapter.cancelDiscovery();
        }
```






```
  //Initialiser adaptateurs de réseau. Un pour les appareils déjà jumelés et
        // Un pour les appareils nouvellement découverts
        mPairedDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);
        mNewDevicesArrayAdapter = new ArrayAdapter<String>(this,
R.layout.device_name);

          //Trouver et mettre en place le ListView pour les appareils associés
        ListView pairedListView = (ListView) findViewById(R.id.paired_devices);
        pairedListView.setAdapter(mPairedDevicesArrayAdapter);
        pairedListView.setOnItemClickListener(mDeviceClickListener);


        //Trouver et mettre en place le ListView pour les appareils nouvellement
découverts
        ListView newDevicesListView = (ListView) findViewById(R.id.new_devices);
        newDevicesListView.setAdapter(mNewDevicesArrayAdapter);
        newDevicesListView.setOnItemClickListener(mDeviceClickListener);


        /// / Enregistrer des émissions lorsqu'un périphérique est découvert
        IntentFilter filter = new IntentFilter(BluetoothDevice.ACTION_FOUND);
        this.registerReceiver(mReceiver, filter);
/// / Enregistrer des émissions lors de la découverte a fini
        filter = new IntentFilter(BluetoothAdapter.ACTION_DISCOVERY_FINISHED);
        this.registerReceiver(mReceiver, filter);
        //Obtenir l'adaptateur Bluetooth local
        mBtAdapter = BluetoothAdapter.getDefaultAdapter() ;
 /// / Obtenir un ensemble de dispositifs actuellement liés
        Set<BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();

    //S'il ya des appareils appariés, ajouter chacun des ArrayAdapter
        if (pairedDevices.size() > 0) {
            findViewById(R.id.title_paired_devices).setVisibility(View.VISIBLE);
            for (BluetoothDevice device : pairedDevices) {
                mPairedDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
            }
        } else {
            String noDevices =
getResources().getText(R.string.none_paired).toString();
            mPairedDevicesArrayAdapter.add(noDevices);
        }
    }
@Override
    protected void onDestroy() {
        super.onDestroy();
//Assurez-vous que nous ne faisons pas découverte plus
        if (mBtAdapter != null) {
            mBtAdapter.cancelDiscovery();
        }
// Unregister broadcast listeners
        this.unregisterReceiver(mReceiver);
    }
private void doDiscovery() {
        if (D) Log.d(TAG, "doDiscovery()");
```





```
     setProgressBarIndeterminateVisibility(true);
        setTitle(R.string.scanning);

        //Allumez le sous-titre de nouveaux périphériques
        findViewById(R.id.title_new_devices).setVisibility(View.VISIBLE);

        //Si nous sommes déjà en train de découvrir, arrêter
        if (mBtAdapter.isDiscovering()) {
            mBtAdapter.cancelDiscovery();
        }

     mBtAdapter.startDiscovery();
    }
    //L'auditeur sur-clic pour tous les appareils dans les ListViews
    private OnItemClickListener mDeviceClickListener = new OnItemClickListener()
{
        public void onItemClick(AdapterView<?> av, View v, int arg2, long arg3) {

             //Annuler découverte parce que c'est coûteux et nous sommes sur le
point de se connecter
            mBtAdapter.cancelDiscovery();


            //Obtenez l'adresse MAC de l'appareil, qui est les 17 derniers
caractères de la vue
            String info = ((TextView) v).getText().toString();
            String address = info.substring(info.length() - 17);
Intent intent = new Intent();
            intent.putExtra(EXTRA_DEVICE_ADDRESS, address);

            //Réglez résultat et terminer cette activité
            setResult(Activity.RESULT_OK, intent);
            finish();
        }
    };

    private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            //Lorsque la découverte trouve un dispositif
            if (BluetoothDevice.ACTION_FOUND.equals(action)) {
                // Get the BluetoothDevice object from the Intent
                BluetoothDevice device =
intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

                // Si elle est déjà jumelé, sauter, parce qu'il a été déjà
répertorié
                if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                    mNewDevicesArrayAdapter.add(device.getName() + "\n" +
device.getAddress());
                }
            //Lorsque la découverte est terminée, changer le titre de l'activité
            } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action))
{
                setProgressBarIndeterminateVisibility(false);
                setTitle(R.string.select_device);
                if (mNewDevicesArrayAdapter.getCount() == 0) {
```






```
        String noDevices = getResources().getText(R.string.none_found).toString();
                    mNewDevicesArrayAdapter.add(noDevices);
                }
            }
        }
    };

}
```

- son fichier XML :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView android:id="@+id/title_paired_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/title_paired_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/paired_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="1"
    />
<TextView android:id="@+id/title_new_devices"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
        android:text="@string/title_other_devices"
        android:visibility="gone"
        android:background="#666"
        android:textColor="#fff"
        android:paddingLeft="5dp"
    />
    <ListView android:id="@+id/new_devices"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:stackFromBottom="true"
        android:layout_weight="2"
    />
    <Button android:id="@+id/button_scan"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/button_scan"
    />
</LinearLayout>
```





I.3 Classe configuration.java :

```
package com.example.memoire;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.os.PowerManager;
import android.text.Editable;
import android.util.Log;
import android.view.View;
import android.view.Window;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageButton;
import android.widget.RadioButton;
import android.widget.TextView;
import android.widget.Toast;
import android.widget.ToggleButton;

public class configuration extends Activity implements  Button.OnClickListener{
        public static String bn_canaux = "nbr_canau";
        public static String frq = "frequence";
        public static String tem = "temps";
     protected PowerManager.WakeLock mWakeLock;
         @Override
           public void onCreate(Bundle savedInstanceState) {
               super.onCreate(savedInstanceState);
requestWindowFeature(Window.FEATURE_NO_TITLE);
              PowerManager pm = (PowerManager)
getSystemService(Context.POWER_SERVICE);
             this.mWakeLock =
pm.newWakeLock(PowerManager.SCREEN_BRIGHT_WAKE_LOCK, "My Tag");
              this.mWakeLock.acquire();
                Button vu = (Button) findViewById(R.id.vis);
              vu.setOnClickListener(new OnClickListener() {
```

```
@Override
                        public void onClick(View arg0) {
                           EditText canau = (EditText)findViewById(R.id.e1);
                            int c = Integer.valueOf(canau.getText().toString());
                                 EditText frequence =
(EditText)findViewById(R.id.e2);
                                 int f =
Integer.valueOf(frequence.getText().toString());
                                EditText temp = (EditText)findViewById(R.id.e3);
                                int t = Integer.valueOf(temp.getText().toString());
                                Bundle dataBundle = new Bundle();
                                dataBundle.putInt("canau", c);
                                dataBundle.putInt("frequence", f);
                                dataBundle.putInt("temp", t);
                                Intent myIntent = new Intent(configuration.this,
visualisation.class);
                                myIntent.putExtras(dataBundle);

                               startActivity(myIntent);
             }
                    });
//boutton retour
                ImageButton retour = (ImageButton)findViewById(R.id.image);
             retour.setOnClickListener(new OnClickListener() {

                      @Override
                        public void onClick(View arg0) {
                           finish();
                        }
                    });
  }
@Override
      public void onClick(View arg0) {
            // TODO Auto-generated method stub

      }

}
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="fill_horizontal"
    android:orientation="vertical"
    android:background="#FFCCCC" >

    <RelativeLayout
        android:id="@+id/Waveform"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentTop="true" >

        <ImageView
            android:id="@+id/imageView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentRight="true"
            android:layout_alignParentTop="true"
            android:src="@drawable/fac" />
<ImageButton
            android:id="@+id/image"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentRight="true"
            android:src="@android:drawable/ic_menu_revert" /><TextView
            android:id="@+id/textView3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@+id/e2"
            android:layout_alignLeft="@+id/e2"
            android:layout_marginBottom="14dp"
            android:text="temps d&apos;acquisition" /><EditText
            android:id="@+id/e3"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@+id/textView3"
            android:layout_alignLeft="@+id/textView3"
            android:layout_marginBottom="27dp"
            android:ems="10"
            android:text="1"
            android:inputType="number" /><TextView
            android:id="@+id/textView2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"android:layout_above="@+id/e3"
            android:layout_alignLeft="@+id/e3"android:layout_marginBottom="22dp"
            android:text="frequence d&apos;echantillannage" />
```

- Son fichier XML :




```
<EditText
            android:id="@+id/e1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@+id/textView2"
            android:layout_alignLeft="@+id/textView2"
            android:layout_marginBottom="18dp"
            android:ems="10"
            android:text="1"
            android:maxLength="3"
            android:inputType="number"
             />
<TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@+id/e1"
            android:layout_alignLeft="@+id/e1"
            android:layout_marginBottom="14dp"
            android:text="nombre de canaux"
            android:textColor="@color/g" />
<TextView
            android:id="@+id/TextView01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignBottom="@+id/imageView1"
            android:layout_alignRight="@+id/e1"
            android:layout_marginBottom="14dp"
            android:layout_marginRight="23dp"
            android:text="choisir une configuration..."
            android:textColor="@color/status_color" /><Button
            android:id="@+id/vis"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_alignParentBottom="true"
            android:layout_alignParentLeft="true"
            android:layout_alignTop="@+id/image"
            android:text="visualisation" />
<EditText
            android:id="@+id/e2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_above="@+id/image"
            android:layout_marginBottom="30dp"
            android:layout_toLeftOf="@+id/image"
            android:ems="10"
            android:text="1"
            android:inputType="number" />
</RelativeLayout>
  </RelativeLayout>
```



