République Algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université Amar Télidji Laghouat

Faculté des sciences

Département de mathématique et d'informatique


Mémoire en vue de l'obtention du diplôme de licence en informatique

Thème :

# LES SYSTEMES EMBARQUES, ETUDE DE CAS : ANDROID

Présenté par :
BENMOUSSA Ahmed
MAHI Omar El Farouk

Encadré par : Guellouma Younes

**juin 2012**

# Table des matières

# Remerciements

D'abord, on remercie Dieu qui nous aide et qui reste toujours a nos côtés.

On remerie vivement notre encadreur Mr Guellouma Younes de nous avoir aidé pour achever ce modeste travail de recherche ainsi qu'aux membres du jury qui vont évaluer notre travail.

Un grand merci a tous nos enseignants, sans eux ce travail n'aura jamais vu le jour.

A vous tous merci.

# Dédicaces

Je dédie ce travail a mes parents, mon frére Amine et mes deux sœur, ainsi que mes cousins Rafik, Arslane, Mohamed et Ilyes.

Je dédie également ce travail a Si-3mar, Mohamed, Bachir, Sarah, Judith ainsi que tous mes amis et tous les étudiants de $3^{éme}$ année Informatique (promo 2011-2012).

Une spéciale dédicace a mon ami et mon cousin Seddik.

# Dédicaces

Je dédie ce travail a mes parents, mes deux fréres Youces et Sofiane et ma sœur, ainsi que mes cousins Hamid, Anes.

Je dédie également ce travail a mes amis : Hmez, Brahim, Abdou, Karim, Yacine, Tito et tous les étudiants de $3^{éme}$ année Informatique (promo 2011-2012).

# Introduction Générale

Après le développement massif des systèmes d'exploitation classiques comme Windows, Linux MacOS, un nouveau type de systèmes commence à prendre place parmi les produits high tech les plus marquants dans cette première décennie du $21^{eme}$ siècle. Avant, on entendait parler de Embedded Windows, ou bien Embedded Linux, mais maintenant, les systèmes d'exploitation pour l'embarqué font partie de notre vie quotidienne.

L'intérêt de ce mémoire est d'étudier de plus près un des systèmes d'exploitation embarqué les plus populaires, Android. Le but de cette étude est double, elle permet d'une part de mieux comprendre l'architecture logicielle de ce type de systèmes et d'autre part de définir comment on peut développer nos application dans un tel environnement.

Le présent mémoire est organisé comme suit :
Dans le premier chapitre, nous commençons par introduire ce que c'est qu'un système embarqué, son architecture matérielle..., ensuite nous introduisons les systèmes d'exploitation pour l'embarqué et nous donnons des exemples illustratifs.

Le deuxième chapitre est consacré à l'introduction du système d'exploitation Android, dans cette partie, nous donnons une brève définition des différentes versions de ce système, ensuite nous définissons son architecture logicielle.

Enfin, le troisième et dernier chapitre est réservé à la définition des techniques et méthode de développement dans l'environnement Android, pour cette raison nous donnons deux exemples d'applications avec le détail d'implémentation.

# Chapitre 1

# Les systèmes d'exploitations pour l'emabrqué

Dans ce chapitre, on introduit la notion des systémes embarqués, ensuite, on défini ce que c'est q'un système d'exploitation et quels sont ses composants, enfin, on décrit les systèmes d'exploitation pour l'embarqué et on donne quelques exemples illustratifs.

## 1.1 Les systémes emarqués

### 1.1.1 Définition

Les systèmes embarqué, appelés aussi systèmes enfouis (Embedded Systems en Anglais) sont des systèmes électroniques (matériels) et informatiques (logiciels) autonomes, conçus ensembles, afin d'exécuter des taches particulières. Ils fonctionnent généralement en ***temps réel*** [1]. Contrairement aux PCs, les systèmes embarqués ne possèdent pas des entrées/sorties standards.

### 1.1.2 Origine

Le premier système embarqué est le système de guidage de l'Appolo. C'est Appolo Guidance Computer. Il a été développé par Charles Stark Draper du MIT [2]

Apres d'autres systèmes embarqué sont apparus en 1971 avec l'apparition du Intel 4004. L'Intel 4004, le premier microprocesseur, était le premier circuit

1. Un système temps réel est une association logiciel/matériel où le logiciel permet, entre autre, une gestion appropriée des ressources matérielles en vue de remplir certaines tâches ou fonctions dans des limites temporelles bien précises.
2. MIT : Massachusetts Institute of Technology

intégré incorporant tous les éléments d'un ordinateur dans un seul boîtier : unité de calcul, mémoire, contrôle des entrées / sorties.

Alors qu'il fallait auparavant plusieurs circuits intégrés différents, chacun dédié à une tâche particulière, un seul microprocesseur pouvait assurer autant de travaux différents que possible. Très rapidement, des objets quotidiens tels que fours à micro-ondes, télévisions et automobiles à moteur à injection électronique ne tardèrent pas à être équipés de microprocesseurs. Ce sont alors les débuts de l'informatique embarquées.

### 1.1.3 Caractéristiques principales d'un système embarqué

Le système embarqué est un système principalement numérique. Il met généralement en œuvre un processeur. Il exécute une application logicielle dédiée précise (non pas une application scientifique ou grand public). Il n'a pas réellement de clavier standard et l'affichage est limité. Ce ne sont pas des PC, mais des architectures similaires (×86) basse consommation.

### 1.1.4 Domaines d'applications

- **Calcul généraliste :** Similaire aux applications bureau mais embarqué (assistant personnel, téléphone portable, etc...) Consoles de jeux vidéo, set-top box.
- **Contrôle de systèmes :** Moteur, voiture, avion, processus chimique, nucléaire, navigation, etc...
- **Traitement du signal :** Compression vidéo, radar, flux de données, etc...
- **Réseaux et communications :** Transmission de données, commutation, routage, téléphone, Internet, etc...

### 1.1.5 Architecture des systèmes embarqué

Les systèmes embarqués utilisent généralement des microprocesseurs à basse consommation d'énergie ou des microcontrôleurs.

La partie logicielle est en partie ou entièrement programmée dans la partie matériel de ce système, généralement dans une mémoire morte (ROM), EPROM, EEPROM, FLASH, etc...(c'est ce qu'on appelle un firmware).

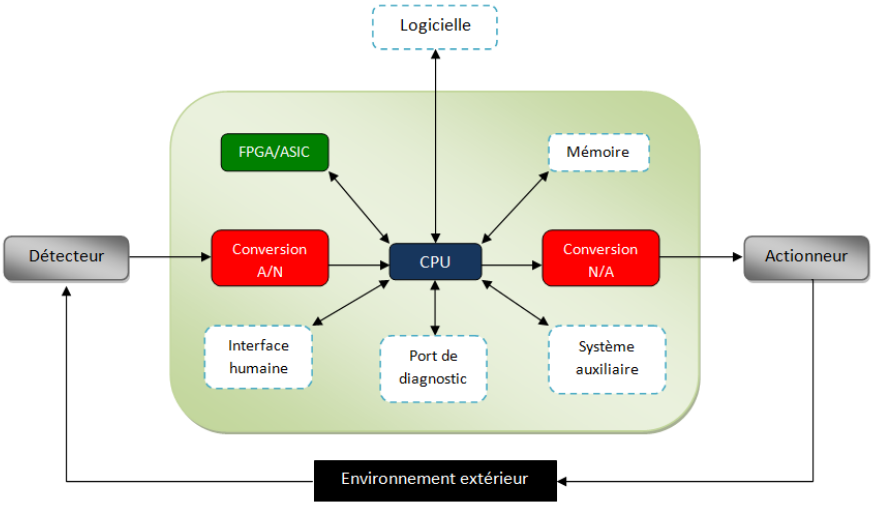



FIGURE 1.1 – Schéma représentant l'architecture des systèmes embarqué.

**Détecteurs(capteurs, etc...) :** sont associés avec des convertisseurs numérique/analogique.

**Actionneurs (LED, etc...) :** sont associés à des convertisseurs analogique/numérique.

**Calculateur**(processeur embarqué et ses E/S).

Possibilité d'avoir un/des FPGA set ou ASICs [3] pour jouer le rôle de coprocesseurs (accélération matérielle).

Cette architecture peut varier selon les systèmes : on peut par exemple, ne pas trouver de systèmes auxiliaires dans de nombreux systèmes embarqué autonome et indépendants. En revanche, l'architecture de base est la plupart du temps composée d'une unité centrale de traitement (CPU), d'un système d'exploitation qui réside parfois uniquement en un logiciel spécifique (ex : routeur), ou une boucle d'exécution (ex : ABS). De même l'interface humaine n'est pas souvent existante, mais est souvent utile pour reconfigurer le système ou vérifier son comportement.

### Comment fonctionne les systèmes embarqué ?

Le fonctionnement d'un système embarqué peut se résumer ainsi :

3. Application Specific Integrated circuit

- Il reçcoit des informations de l'environnement extérieur qu'il converti en signal numérique
- L'unité de traitement composée du CPU, de la mémoire, du logiciel, de l'ASIC et éventuellement de systèmes externes traite l'information
- Le traitement génère éventuellement une sortie qui est envoyée vers la sortie, les systèmes auxiliaires, les ports de monitoring ou l'IHM (l'interface humaine).

### 1.1.6 Quelques propriétés des systèmes embarqué

- **Ciblé :** domaine d'action limité aux fonctions pour lesquelles il a été créé.
- **Simple :** gage de robustesse.
- **Fiable :** fonctionnement complètement autonome.
- **Sécurisé :** plus il est sécurisé, mieux c'est.
- **Maintenable dans le temps :** certains produits sont censés durer jusqu'à 20 ans et plus, dans ce cas il doit être facile à maintenir (surtout dans le domaine militaire).
- **Interface spécifique :** approche matérielle à cause de contraintes d'optimisation.
- **Optimisé :** généralement les logiciels sont de petite taille car plus c'est grand, plus il y a de chance d'avoir des bugs. Ce sont des logiciels produits à grande échelle, le moindre centime compte.
- **Tolérant aux fautes.**

### 1.1.7 Contraintes

Chaque système embarqué exécute des taches prédéfinies. Pour cela il doit satisfaire certaines contraintes :

- **Le coût :** le prix de conception du système embarqué doit être le plus faible possible surtoût s'il est produit en grande quantité.
- **L'espace mémoire :** il convient de concevoir des systèmes embarqué qui répondent au besoin au plus juste pour éviter un surcoût.
- **Puissance de calcul :** il est préférable d'avoir des microprocesseurs qui répondent juste aux besoins de la tache prédéfinit, afin d'éviter des surcouts et une consommation excédentaire d'énergie.
- **Consommation en énergie :** plus la consommation est faible mieux c'est, car les systèmes embarqués utilisent souvent un ou plusieurs batteries et/ou, de panneaux solaires voire de pile à combustible.
- **La sureté de fonctionnement :** car s'il arrive que certains de ces systèmes embarqués subissent une défaillance, ils mettent des vies hu-

maines en danger ou mettent en périls des investissements importants. Ils sont alors dits "critiques" et ne doivent jamais faillir. Par "jamais faillir", il faut comprendre toujours donner des résultats justes, pertinents et ce dans les délais attendus par les utilisateurs (machines et/ou humains) des dits résultats.
- **La sécurité :** les systèmes embarqués doivent être sécurisés, et ça selon le domaine d'application de ce système (la sécurité des systèmes embarqués dans le domaine militaire ou le domaine spatiale doit être plus renfoncé que dans d'autres domaines).

## 1.2 Les systèmes d'exploitation (SE)

### 1.2.1 Définition

Un système d'exploitation (Operating System en Anglais) peut être définit comme un intermédiaire qui relie la partie matériele avec la partie logicielle. Ils sont chargés d'assurer la liaison entre les ressources matérielles, l'utilisateur et les applications.

### 1.2.2 Responsabilités des systèmes d'exploitation

#### Gestion du processeur

Les systèmes d'exploitation sont chargés de gérer l'allocation du processeur entre les différents programmes et processus grâce à un algorithme d'ordonnancement. Cet algorithme peut différer, selon le système d'exploitation et aussi l'objectif visé. Ils assurent aussi la Communication inter-processus et la synchronisation entre eux.

#### Gestion de la mémoire

Les systèmes d'exploitation sont chargés de gérer l'espace mémoire alloué à chaque application. Les SE peuvent aussi utiliser une partis du disque dur et la transformer en zone mémoire en cas d'insuffisance de mémoire physique (on parle alors de mémoire virtuelle).

#### gestion des fichiers

Les systèmes d'exploitation gèrent la lecture et l'écriture dans le système de fichiers et les droits d'accès aux fichiers par les utilisateurs et les applications.

#### Gestion des entrées/sorties et pilotes de périphériques

Les systèmes d'exploitation permettent de contrôler l'accès des programmes aux ressources matérielles par l'intermédiaire des pilotes (appelés également gestionnaires de périphériques ou gestionnaires d'entrée/sortie).

#### Gestion des tâches

Les systèmes d'exploitation sont également chargés de la bonne exécution des applications en leur affectant les ressources nécessaires à leur bon fonctionnement. Ils permettent aussi de "tuer" une application ne répondant plus correctement.

#### Gestion des informations

Le système d'exploitation fournit un certain nombre d'indicateurs permettant de diagnostiquer le bon fonctionnement de la machine.

#### Gestion des droits

Les systèmes d'exploitation sont chargés de la sécurité liée à l'exécution des programmes en garantissant que les ressources ne sont utilisées que par les programmes et utilisateurs possédant les droits adéquats.

### 1.2.3 Architectures des systèmes d'exploitation

#### SE à noyau monolithique

Un système d'exploitation est monolithique si l'ensemble des fonctions du système et des pilotes sont regroupés dans un seul bloc de code et un seul bloc binaire généré à la compilation. Les noyaux monolithiques ont été les premiers à être développés et mis en œuvre.

Pour répondre aux problèmes des noyaux monolithiques, ces derniers sont devenus **modulaires**. Dans ce type de noyau, seules les parties fondamentales du système sont regroupées dans un bloc de code unique (monolithique). Les autres fonctions, comme les pilotes matériels, sont regroupées en différents modules qui peuvent être séparés tant du point de vue du code que du point de vue binaire.

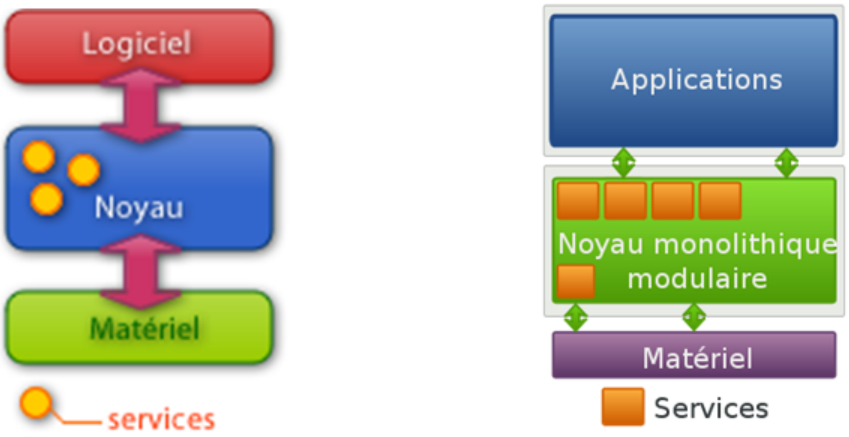
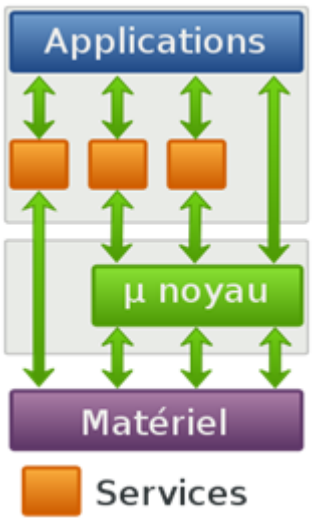



FIGURE 1.2 – Architecture d'un SE à noyau monolithique - monolitique modulaire

### SE à Micro-noyau




Les systèmes d'exploitation a micro-noyau ont la particularité d'avoir la plus grande partie des services du SE à l'extérieur du noyau, et celà pour limiter les fonctionnalités dépendantes du noyau.

Ces fonctionnalités sont alors fournies par de petits serveurs indépendants,déplacés en espace utilisateur, possédant souvent leur propre espace d'adressage. Donc seules les fonctions fondamentales sont dans l'espace noyau.

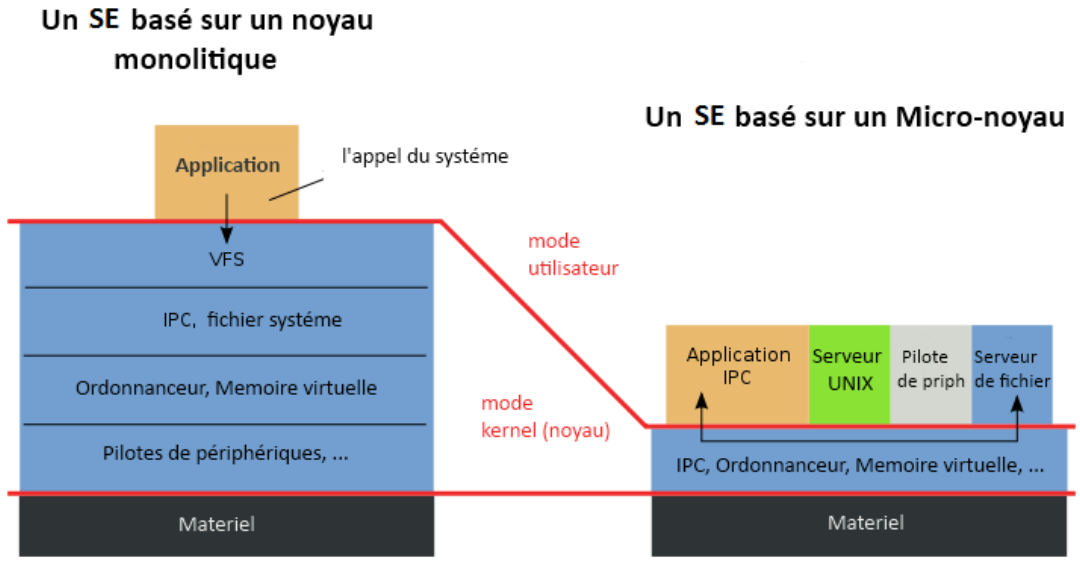
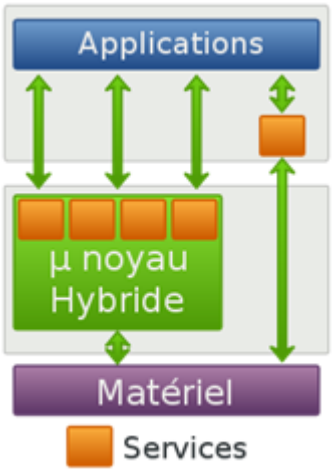



FIGURE 1.3 – Ce schéma représente la différence entre un SE monolithique et un SE micronoyau

### SE à noyau hybride




Les noyaux hybrides reprennent des concepts à la fois des noyaux monolithiques et des micro-noyaux, pour combiner les avantages des deux. Le noyau hybride a réintégré dans l'espace noyau, certaines fonctions non critiques mais qui sont très génératrices d'appels systèmes, ce qui permet d'améliorer les performances tout en conservant de nombreuses propriétés des systèmes à micro-noyaux.

#### SE multicouche

C'est un systéme d'exploitation organisé en hiérarchie de couches, chacune est construite sur la base des services offerts par la couche inférieure. Interface et gestion des interruptions similaire à celle des systémes monolithiques. Il est aisé à maintenir, et mieux structuré que monolithique. Configuration plus fine (modularité), meilleure utilisation de la mémoire, et une meilleure performance.

| Couche | Fonction |
|---|---|
| 5 | Opérateur |
| 4 | Programmes utilisateur |
| 3 | Gestion des E/S |
| 2 | Communication opérateur-processus |
| 1 | Gestion de la mémoire et du tampon |
| 0 | Allocation du processeur-multiprogrammation |

## 1.3 Les systémes d'exploitations pour l'embarqué

Pour fonctionner, chaque système embarqué doit avoir son propre logiciel embarqué, ce dernier peut être définit ainsi :

### 1.3.1 Logiciel embarqué

C'est un programme/application utilisé dans un équipement et complètement intégré dans ce dernier. Pour résumer, un système embarqué se constitue d'une partie matérielle, plus un ou plusieurs logiciels. Possibilité d'avoir aussi un système d'exploitation.

### 1.3.2 Les types de systèmes embarqué

#### Systèmes embarqué destinés à l'utilisateur (High-end)

Généralement une version dégradée d'un SE existant (ex : Linux). Ex : routeurs, PDA, Smartphones, etc...

#### Systèmes embarqué profondément enfouis

Peu de fonctions, très petite empreinte mémoire. Ex : Appareil photo numérique, téléphones portables, etc...

### 1.3.3 Les avantages d'avoir un SE pour un système embarqué

- Faciliter le développement de logiciels embarqué.
- Gain en temps de développement.
- Accès aux services du SE via des APIs (réutilisabilité du code, interopérabilité, portabilité, maintenance aisée).
- Possibilité de bénéficier des mêmes avancées technologiques que les applications classiques (TCP/IP, HTTP, etc...).
- Environnement de développement plus performant.

### 1.3.4 Les différents Système d'exploitation embarqué

**VxWorks**


VxWorks est un système d'exploitation temps réel multitâche. Il est aujourd'hui le noyau temps réel le plus utilisé dans l'industrie. Il a un noyau monolithique. Il est développé par la société Wind River (une firme acquise par Intel en 2009) qui a également racheté récemment les droits du noyau temps réel pSOS, un peu ancien, mais également largement utilisé.

Il est principalement utilisé par la recherche et l'industrie (aéronautique, automobile, transport, télécommunication).

VxWorks inclut en natif un support TCP/IP. Le gros inconvénient de ces deux systèmes est le coût important des licences. Il est également nécessaire d'utiliser un environnement de compilation croisé.

**QNX**


QNX est un noyau temps réel de type Unix et il est parfaitement conforme à Posix [4]. Il a été Développé par la société canadienne QNX Software. Il permet de développer directement sur la plate-forme cible et intègre l'environnement graphique Photon, proche de X-WindowSystem. Il peut occuper une très faible empreinte mémoire.

4. Posix : Portable Operating System Interface

### $\mu$ C/OS


$\mu$ C/OS est destiné à des environnements de très petite taille comme des micro-contrôleurs de type Motorola 68HC11. Il a été développé par le Canadien Jean J. Labrosse. Il est maintenant disponible sur un grand nombre de processeurs et peut intégrer des protocoles standards comme TCP/IP ($\mu$ C/IP). Il est utilisable gratuitement pour l'enseignement et de nombreuses informations sont disponibles sur `http://www.ucos-ii.com`.

### Windows CE


Windows CE [5] est un système embarqué développé par Microsoft Windows CE et ses cousins comme Embedded Windows NT n'ont pour l'instant pas Détrôné les systèmes embarqué traditionnels. Victime d'une réputation de fiabilité approximative, Windows CE est pour l'instant cantonné à l'équipement de nombreux assistants personnels (PDA, Personal Digital Assistant).

### LynxOS


LynxOS est un système temps réel conforme à la norme Posix. Il est développé par la société LynuxWorks, qui a arécemment modifié son nom en raison de son virage vers Linux avec le développementde BlueCat.

### Nucleus


Nucleus est développé par la société AcceleratedTechnology Inc. C'est un noyau temps réel qui inclut une couche TCP/IP, une interface graphique (Graphix) ainsi qu'un navigateur web (WebBrowse) et un serveur HTTP (Web-Server). Il est livré avec les sources et il n'y pas de royalties à payer pour la redistribution.

5. CE : Compact Embedded

#### eCos

eCos [6], fut developpé par Cygnuspuis, aquise ensuite par RedHat. C'est un noyau Temps réel, bien adapté aux très faibles empreintes mémoire. Son environnement de développement est basé sur Linux et la chaîne de compilation GNU avec conformité au standard Posix et disponibilité de protocoles standards comme TCP/IP.

### 1.3.5 Systèmes d'exploitation pour téléphone mobile

#### Windows Mobile (Windows Phone)


Windows Mobile est un système d'exploitation développé par Microsoft pour Smartphones (téléphones intelligent), tablette et Pc Pocket. Il est basé sur le noyau Windows CE.

#### IOS (Apple OS)


IOS (anciennement appelé IPhoneOS), est un système d'exploitation développé par Apple pour l'IPhone, l'Ipad, et l'Ipod. Il est basé sur un noyau hybride XNU.

6. eCos : Embeddable Configurable OS

**Blackberry OS**

BlackBerry OS est un système d'exploitation propriétaire pour téléphone mobile de la gamme BlackBerry, concu par la société canadienne Research In Motion (RIM). Il est basé sur un noyau QNX.

**MeeGo**

MeeGo est un système d'exploitation pour Smartphones, netbooks, tablette PC, et téléviseurs (Smart TV). le système MeeGo est open source et il est basé sur un noyau Linux.

**Android**

Android est un système d'exploitation open source conçcu pour Smartphones, tablettes et même des téléviseurs. Il basé sur un noyau Linux.

## 1.4 Conclusion

Nous avons présenté dans ce chapitre une courte introduction sur les systèmes d'exploitation pour l'embarqué commençant par la définition de l'environnement matériel jusqu'a la description de quelques systèmes utilisés. Le but du prochain chapitre est d'étudier un des systèmes d'exploitation les plus utilisés : Android.

# Chapitre 2

# Le systéme d'éxploitation Android

Les systèmes d'exploitation ont beaucoup évolué ces 15 dernières années surtout avec l'arrivée des Smartphones, qui ont bouleversé le monde spécialement les systèmes d'exploitation des téléphones. Une tendance à développer des systèmes d'exploitation pour Smartphone est devenue une obligation vu le développement énorme du monde de l'embarqué et du mobile. Heureusement les développeurs ont tenu le coup malgré l'architecture non normalisée des Smartphones. Par conséquent, beaucoup de systèmes ont vu le jour : PalmOs (1996), Windows PC Pocket, etc... Ensuite d'autres systèmes d'exploitation plus sophistiqués sont apparus, on s'intéresse surtout au système d'exploitation Android.

## 2.1 Définition

Comme définit dans le chapitre précèdent, Android est un système d'exploitation pour Smartphones, tablette PC, téléviseurs, etc. Annoncé officiellement le 5 novembre 2007, Android est développé par Google et l'Open Handset Alliance.

## 2.2 Historique

**Octobre 2003 :** Andy Rubin, Rich Miner, Nick Sears et Chris White ont fondé la société Android.

**17 Aout 2005 :** Google rachète la société Android.

**5 Novembre 2007 :** Naissance de l'Open Handset Alliance (un consortium qui regroupe differente compagnie : Google, T-Mobile, Intel, Nvi-

dia, HTC, LG, Motorola, et d'autres compagnie.)


FIGURE 2.1 – Logo de l'Open Handset Alliance

**5 Novembre 2007 :** Naissance du système d'exploitation Android.
**12 Novembre 2007 :** Publication du SDK Android, le Kit de développement Android.
**17 Avril 2008 :** Google lance l'Android Developer Challenge.
**28 Août 2008 :** Création de l'Android Market.
**21 Octobre 2008 :** Publication du code source d'Android : `http://source.android.com`
**22 Octobre 2008 :** Commercialisation aux USA du premier mobile sous Android : le T-mobile G1, produit par HTC. Ouverture de l'Android Market.

## 2.3 Version d'Android

### 2.3.1 Android 1.0

C'est la premiére version d'Android, sortie en 2008 avec le téléphone HTC Dream G1.

**Les caractéristiques de cette version :**

- Intégration avec Google Services.
- Explorateur Web qui supporte HTML & XHTML.
- Téléchargement Android Market.
- Multitâches, messagerie instantané, Bluetooth et Wi-Fi.

### 2.3.2 Android 1.5 [Cupcake]

Version sortie le 30 Avril 2009. Elle basé sur un noyau Linux 2.6.27

**Les caractéristiques et améliorations de cette version :**

- Démarrage plus rapide de la camera et de Image Capture.

- Un GPS plus développé et rapide.
- Un nouveau clavier virtuel avec la prédiction de mots.
- Envoie de vidéos sur YouTube et de photos sur Picasa depuis le téléphone.

### 2.3.3 Android 2.0 [Donut]

Version sortie le 15 Septembre 2009, basé sur un noyau 2.6.29

**Les caractéristiques et améliorations de cette version :**

- Recherche vocale plus rapide.
- Interface native pour l'appareil photo, la camera et la galerie.
- Mise à jour du support pour CDMA/EVDO [1], 802.1x, VPNs [2], et une synthèse vocale.
- Support des écrans avec une résolution WVGA [3].
- Amélioration de la vitesse pour les applications utilisant la camera.
- Framework de reconnaissance de gestes et outil de développement GestureBuilder.

1. ce sont des types de transmission de données.
2. VPN : Virtual Private Network
3. WVGA ou Wide VGA est une résolution d'écran plus large (wide en Anglais) que le VGA. Il est d'une hauteur fixe de 480 pixels et d'une largeur variable d'au moins 800 pixels.

### 2.3.4 Android 2.1 [Eclair]

Version sortie le 26 Octobre 2009, basé sur un noyau Linux 2.6.29

**Les caractéristiques et améliorations de cette version :**

- Vitesse hardware optimisée.
- Support de plus de taille d'écran et résolutions.
- Support de Bluetooth 2.1.
- Nouvelle interface du navigateur et support de l'HTML5.
- Support de Microsoft Exchange Server par Exchange ActiveSync 2.5.

### 2.3.5 Android 2.2 [Froyo]

Version sortie le 20 mai 2010, basé sur un noyau Linux 2.6.32

**Les caractéristiques et améliorations de cette version :**

- Amélioration additionnel de la vitesse des applications grâce à l'implémentation JIT [4].
- Intégration du moteur JavaScript V8 de Chrome dans le navigateur.
- Adobe Flash 10.1.

4. JIT : Just In Time compilation, ou compilation juste à temps est une technique utilisée pour simuler un processeur.

– Support des écrans à haute densité de pixels (320 dpi).
– multiples langues de clavier.

### 2.3.6 Android 2.3 [Gingerbread]

Version sortie en 6 decembre 2010, basé sur un noyau Linux 2.3.35

**Les caractéristiques et améliorations de cette version :**

– Passage au syst'eme de fichiers ext4.
– Support de la VoIP [5] et SIP [6].
– Support des formats vidéo WebM [7]/VP8, et l'encodage audio AAC.
– Amélioration de la fonction copier/coller et sélection du texte.
– Support du NFC [8].

5. VoIP : Voice over Internet Protocol

6. SIP : Session Initiation Protocol est un protocole standard ouvert de gestion de sessions souvent utilisé dans les télécommunications multimédia (son, image, etc.). Il est depuis 2007 le plus courant pour la téléphonie par internet (la VoIP).

7. WebM est un format multimédia ouvert principalement destiné à un usage sur le web.

8. NFC : Near Field Comunication (communications en champ proche), une Technologie de communication lancée par Sony et Philips. Elle permet d'échanger des données entre un lecteur et n'importe quel terminal mobile ou entre les terminaux eux-mêmes et ce, à un débit maximum de 424 Kbits/s.

### 2.3.7 Android 3.0 [Honeycomb]

Version sortie en 26 janvier 2011, basé sur un noyau Linux 2.6.36

**Les caractéristiques et améliorations de cette version :**

- Interface optimisée pour les tablettes et devices à écran large.
- Support des processeurs multi-cœurs.
- Multitâche revu.
- Bureau tridimensionnel avec widgets améliorés

Une autre version a vu le jour le 15 juillet 2011, il s'agit de la version 3.2

**Les caractéristiques et améliorations de cette version :**

- Support des tablettes de 7 pouces.
- Support des processeurs Qualcomm.

### 2.3.8 Android 4.0 [Ice Cream Sandwich]

Version sortie le 19 octobre 2011.

**Les caractéristiques et améliorations de cette version :**

- Unification des versions tablette et Smartphone d'Android.

- Nouvelle interface utilisateur.
- Disparition des boutons physiques au profit de boutons tactiles sur l'écran (réservé a certains terminaux [ex : Galaxy Nexus]).
- Capture d'écran native.
- Support natif des dossiers sur les bureaux.
- Support des écrans trés haute résolution *720.
- Voice Search amélioré.
- Compteur de consommation data.

Une autre version (4.0.4) est sortie le 29 mars 2012.

**Quelques améliorations qu'a apporté cette mise à jour :**

- Amélioration de la fluidité du système.
- Correction d'un bug dans les statistiques batteries.
- Nouveau menu Power.
- Optimisation du temps de démarrage.

## 2.4 Architecture Android

Le système d'exploitation Android est basé sur un noyau Linux 2.6. Le schéma suivant représente l'architecture de ce SE.

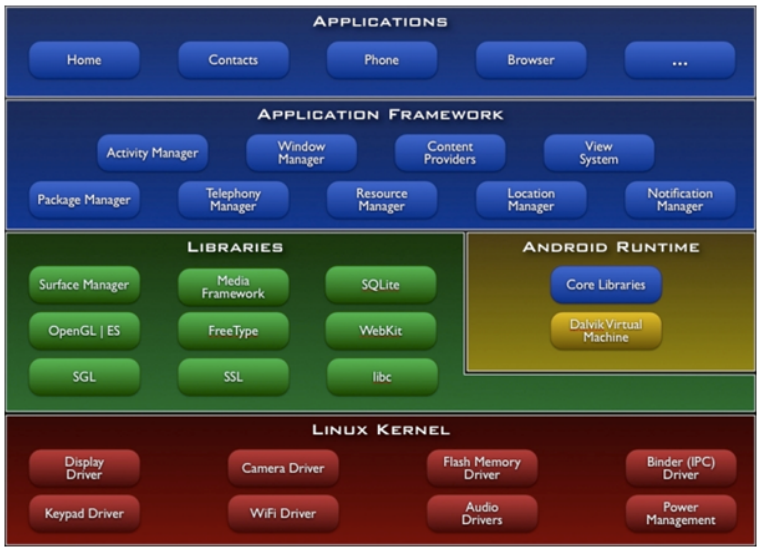



FIGURE 2.2 – Architecture d'Android. Source : `developer.android.com`

### 2.4.1 Noyau Linux (Linux Kernel)

La version du noyau utilisée avec Android est une version linux 2.6 conçue spécialement pour l'environnement mobile, avec une gestion avancée de la batterie et une gestion particulière de la mémoire. C'est cette couche qui fait en sorte qu'Android soit compatible avec tant de supports différents. Cette couche est la seule qui gère le matériel. Cependant, il ne s'agit pas d'un linux proprement dit car X-windows n'est pas implémenté (gestion de fenêtrage) tout comme la librairie glibc qui n'est pas supportée (Android utilise une bibliothèque libc customisée nommée Bionic).

### 2.4.2 Les Bibliothèques (Libraries)

Ces bibliothèques proviennent de beaucoup de projets open-sources, écris en C/C++ pour la plupart, comme SQLite pour les bases de données, WebKit pour la navigation web ou encore OpenGL afin de produire des graphismes en 2D ou en 3D.

### 2.4.3 Le moteur d'execution Android (Android Runtime)

Un moteur d'exécution ("runtime system" en anglais) est un programme qui permet l'exécution d'autres programmes. C'est cette couche qui fait qu'Android n'est pas qu'une simple "implémentation de Linux pour portables". Elle contient certaines bibliothèques de base du Java accompagnées de bibliothèques spécifiques à Android et la machine virtuelle "Dalvik".

### 2.4.4 Les frameworks pour les applications

Utilisant le langage JAVA le framework met à disposition un ensemble de classes utiles à la création d'applications sans oublier la mise à disposition de classes et méthodes permettant l'accés au matériel, à la gestion de l'interface graphique et aux ressources de l'application. Enfin, notons que le framework Android se compose des services applicatifs suivants :

- Activity Manager, gerant le cycle de vie des Activités (Activities).
- Views permettant de cr'eer les interfaces graphiques.
- Notification Manager founissant un mécanisme d'envois de message aux utilisateurs.
- Content Providers, permettant aux applications de partager des données entre elles.

– Resource Manager, offrant l'externalisation de ressources tels que les chaines de caractères pour la notion d'internationalisation, de style, de menus, etc...

### 2.4.5 Les applications (Applications)

C'est la couche la plus haute de l'architecture. Celle-ci contient l'ensemble des applications natives, tierces ou développées présentes sur l'appareil. Par exemple les fonctionnalités de base inclues un client pour recevoir/envoyer des emails, un programme pour envoyer/recevoir des SMS, un calendrier, un répertoire, etc.... Toutes les applications présentes dans cette couche sont, comme nous l'avons vu précédemment, exécutées par le moteur d'exécution Android.

## 2.5 Compilation d'un noyau Linux pour telephone mobile

Pour compiler un noyau Linux mobile il nous faut un système d'exploitation **32 bits**, qui tourne sur un kernel linux.

### 2.5.1 Modification des sources.list

On modifie sources.list pour avoir les binaires proprios :

*# nano /etc/apt/sources.list*
*deb http ://ftp2.fr.debian.org/debian/ lenny main contrib non-free*
*deb-src http ://ftp2.fr.debian.org/debian/ lenny main contrib non-free*
*deb http ://security.debian.org/ lenny/updates main contrib non-free*
*deb-src http ://security.debian.org/ lenny/updates main contrib non-free*
*deb http ://volatile.debian.org/debian-volatile lenny/volatile main contrib non-free*
*deb-src http ://volatile.debian.org/debian-volatile lenny/volatile main contrib non-free*

### 2.5.2 Mettre la machine à jour

Pour mettre la machine à jour :
*# aptitude install git-core gnupg sun-java5-jdk flex bison gperf libsdl-dev libesd0-dev libwxgtk2.6-dev build-essential zip curl libncurses5-dev zlib1g-dev libreadline5 libreadline5-dev readline-common*

### 2.5.3 Installation des dépendances

*# aptitude install git-core gnupg sun-java5-jdk flex bison gperf libsdl-dev libesd0-dev libwxgtk2.6-dev build-essential zip curl libncurses5-dev zlib1g-dev libreadline5 libreadline5-dev readline-common*

### 2.5.4 Compilation du Python 2.4

Pour finir d'installer les dépendances il faut compiler python 2.4 depuis les sources, car il faut le support readline[9] d'actif :

*# wget "http ://www.python.org/ftp/python/2.4.6/Python-2.4.6.tar.bz2"*
*# tar xvjf Python-2.4.6.tar.bz2 cd && Python-2.4.6/*
*# ./configure –disable-ipv6 –with-readline make make install*

### 2.5.5 Téléchargement des sources

*#mkdir -p /root/Build/bin/ /root/Build/luoAndroid*
*# export PATH="$PATH :/root/Build/bin"*
*# cd /Build && wget "http ://android.git.kernel.org/repo" -O bin/repo*
Ou :
*# curl http ://android.git.kernel.org/repo > /Build/bin/repo*
On continue :
*# chmod a+x /Build/bin/repo*
*# cd /Build/luoAndroid/*
*# repo init -u git ://android.git.us.kernel.org/platform/manifest.git -b cupcake*[10]
*# nano .repo/manifest.xml*
Remplacer la 4ème ligne par celle là : *fetch="git ://android.git.us.kernel.org/"*

### 2.5.6 Modification du local_manifest.xml

Après l'initialisation du repo git Android pour telecharger les sources. On ouvre le fichier local_manifest.xml avec un editeur de texte. On a choisit de travailler avec **nano**

*# cd /root/Build/luoAndroid && nano .repo/local_ manifest.xml*

9. la commande readline permet aux utilisateurs d'éditer des lignes de commande comme elles sont tapées
10. fait reference a la version 1.5 (Cupcake)

**Télécharger le Kernel Linux 2.6.27**

Pour avoir le Kernel 2.6.27, on copie ces lignes dans le fichier local_manifest.xml

*<?xml version="1.0" encoding="UTF-8"?>*
*<manifest>*
*<remove-project name="kernel/common"/>*
*<project path="kernel" name="kernel/msm" revision="refs/heads/android-msm-2.6.27"/>*
*<project path="vendor/xxxx*[11] *" name="platform/vendor/xxxx" revision="refs/heads/master"/>*
*<project path="hardware/msm7k" name="platform/hardware/msm7k" revision="refs/heads/master"/>*
*</manifest>*

**Télécharger le Kernel Linux 2.6.29**

Pour avoir le Kernel 2.6.29

*<?xml version="1.0" encoding="UTF-8"?>*
*<manifest>*
*<remove-project name="kernel/common"/>*
*<project path="kernel" name="kernel/msm" revision="refs/heads/android-msm-2.6.29"/>*
*<project path="vendor/xxxx" name="platform/vendor/xxxx" revision="refs/heads/master"/>*
*<project path="hardware/msm7k" name="platform/hardware/msm7k" revision="refs/heads/master"/>*
*</manifest>*

**Téléchargement des sources**

*# cd /root/Build/luoAndroid && repo sync*

### 2.5.7 La configuration

*# cd /root/Build/luoAndroid/kernel/*
*# export PATH="$PATH:/root/Build/bin:/root/Build/luoAndroid/prebuilt/linux-x86/toolchain/arm-eabi-4.3.1/bin"*
*# export ARCH=arm*
*# export SUBARCH=arm*
*# export CROSS_COMPILE=arm-eabi-*

11. xxxx fait reference au nom du telephone

#### Fichier de pre-configuration

Ce script permet d'écrire un fichier de pré configuration valide pour Android.
*# make msm_defconfig*

#### Configurer le noyau

*# nano .config*
*# make menuconfig*

### 2.5.8 La compilation

#### Compilation du noyau

*# make*
*# make INSTALL_MOD_PATH=modules modules_install*
*# cp arch/arm/boot/zImage /root/my_first_kernel*

#### Compilation du module WiFi

Pour compiler le module wifi lié au noyau wlan.ko, il faut taper les ligne de commandes suivantes :

#### Compilation du module WiFi pour le noyau 2.6.27

*# cd /root/Build/luoAndroid/system/wlan/ti/sta_dk_4_0_4_32/*
*# make KERNEL_DIR=/root/Build/luoAndroid/kernel*

#### Compilation du module WiFi pour le noyau 2.6.29

*# cd /root/Build/luoAndroid/*
*# repo download platform/system/wlan/ti 10344/1*
*# cd system/wlan/ti/sta_dk_4_0_4_32/*
*# make KERNEL_DIR=/root/Build/luoAndroid/kernel*

On verra une fois fini le chemin pour récupérer le wlan.ko. Chaque version de kernel est lié à un module wlan.ko.

## 2.6 Conclusion

Nous avons vu dans ce chapitre une courte description du système Android, ses version, son architecture et un exemple de la compilation du noyau linux pour Android. Le sujet du troisième et dérnier chapitre et d'étudier comment developper des applications pour android, pour cette raison on va définir la platte forme de développement et on va présenter quelques exemples d'applications.

# Chapitre 3

# La programmation sous Android

Afin de répondre aux besoins croissants des exigences des smartphones, les développeurs d'Android ont créé une plateforme spéciale très bien structurée. Le but de ce présent chapitre est décrire cette plateforme d'une manière simple et de montrer comment on peut développer aisément des applications pour Android.

## 3.1 Les application Android

### 3.1.1 Définition

Une application Android se compose de différents fichiers, reliès entre eux par un fichier de configuration. Les applications Android sont trés variables et visent plusieurs domaines. On peut trouver des applications de commerce, utilitaire, service d'information, des jeux, etc...

### 3.1.2 Les composants d'une application Android

Les applications Android se composent de plusieurs éléments. Plus l'application est complexe, plus le nombre d'éléments utilisé est grand. Voici donc les différents éléments qui constituent une application :

- Activités
- Services
- Fournisseurs de contenus
- Broadcast Receivers
- Objets Intent
- Récépteurs d'Intents
- Gadjets
- Notifications

### Les activités

Les activités [1] permettent d'afficher les éléments d'une application donnée.

### Les services

Un service est une sorte d'activité, ne possédant pas d'interface visuelle. Celle-ci est donc lancée en fond, permet donc par exemple, d'effectuer une vérification des e-mails périodiquement...

### Les fournisseurs de contenus

Les fournisseurs de contenus [2] permetent de récupérer un contenu. Ils permettent par exemple de récupérer la liste des contacts... chaque application peut ainsi partager des informations avec les autres applications.

### Broadcast receivers

Ce sont des composants chargés d'écouter les autres applications, et réagir lorsqu'un signal est reçcu (par exemple batterie faible, une photo a été prise, signal GSM,...)

### Les Intents

Les Intents permettent aux applications d'envoyer des messages pour déclencher une action, de fournir ou demander des services, des données, etc...

### Récepteur d'Intent

Comme son nom l'indique, il permet à une application d'être à l'écoute des autres afin de recevoir leurs Intents.

### Les Gadgets

Les gadgets [3] sont des composants graphiques qui s'installent sur le bureau Android.

### Notification

Une notification signale une information à l'utilisateur sans qu'elle interrompe ses actions en cours d'exécution.

1. Activity en Anglais
2. Content provider en Anglais
3. Widgets en Anglais

**Les permission**

Certaines opérations sont réalisables à condition d'en obtenir la permission. Ces actions sont de plusieurs formes. Il faut lui en donner l'autorisation par une balise *<uses-permission>*. Ils existent plusieurs permissions : l'accès a Internet, l'utilisation du GPS, l'accès aux données personnelles, l'accès aux matériels du téléphone, etc...

**Exemple :**
Pour avoir la permission d'accéder a Internet :
*<uses-permission android :name= "android.permission.INTERNET*

Géolocalisation (GPS) :
*<uses-permission android :name="android.permission.*
*ACCESS_COARSE_LOCATION"/>*
*<uses-permission android :name="android.permission.*
*ACCESS_FINE_LOCATION" />*
*<uses-permission android :name="android.permission.*
*ACCESS_MOCK_LOCATION" />*

### 3.1.3 Les ressources Android

Les ressources sont des fichiers externes ne contenant pas d'instruction qui sont utilisés par le code et liés à l'application au moment de sa construction. Android offre un support d'un grand nombre de fichiers ressources comme les fichiers images JPEG et PNG, les fichiers XML, etc...

Toutes ces ressources sont placées, converties ou non, dans un fichier de type APK qui constituera le programme distribuable de l'application. De plus, Android crée une classe nommée R qui sera utilisée pour se référer aux ressources dans le code.

### 3.1.4 Cycle de vie d'une application

Les applications Android ont un comportement particulier, ils n'ont pas le contrôle total sur leur cycle de vie car ils réagissent à des états imposés par le système (démarrage, pause, reprise, arrêt, ...). Donc le système peut arrêter ou mettre en pause une activité ou une application s'il juge nécessaire de le faire par exemple si l'application consomme trop de ressource processeur ou mémoire.

## 3.2 Les activités Android

### 3.2.1 Définition

Une activité est le point d'entrée principal pour une application Android. Elle peut être définie comme un écran de l'application. Donc pour chaque écran de l'application, il lui faut une ou plusieurs activités Une activité est composée de deux volets :

- La logique de l'activité et la gestion du cycle de vie de l'activité qui sont implémentés en Java dans une classe héritant de "Activity".
- L'interface utilisateur, qui pourra être définie soit dans le code de l'activité soit de fa,/con plus générale dans un fichier XML placé dans les ressources de l'application.

### 3.2.2 Cycle de vie d'une activité

Les états principaux d'une activité sont les suivants :

**active :** l'activité est visible, detient le focus et attend les entrées utilisateur.

**suspendue :** activité au moins en partie visible à l'écran mais qui ne détient pas le focus.

**arrêtée :** activité non visible.

#### Méthodes correspondant au cycle de vie

Les méthodes ci-dessous sont appelées au cours du cycle de vie. Si elles sont surchargées, elles doivent faire appel à leurs homologues de la classe supérieure(super.on...).

**onCreate(parametre) :** appelée à la création. Le paramètre permet de récupérer un état sauvegardé lors de l'arrêt de l'activité (si on a fait une sauvegarde).

**onStart() :** appelée quand l'activité démarre.

**onResume() :** appelée quand l'activité vient en premier plan.

**onRestart() :** appelée quand l'activité redémarre.

**onPause() :** appelée quand l'activité n'est plus en premier plan.

**onStop() :** appelée quand l'activité n'est plus visible.

**onDestroy() :** appelée quand l'activité se termine.

**finish() :** permet de terminer une activité.

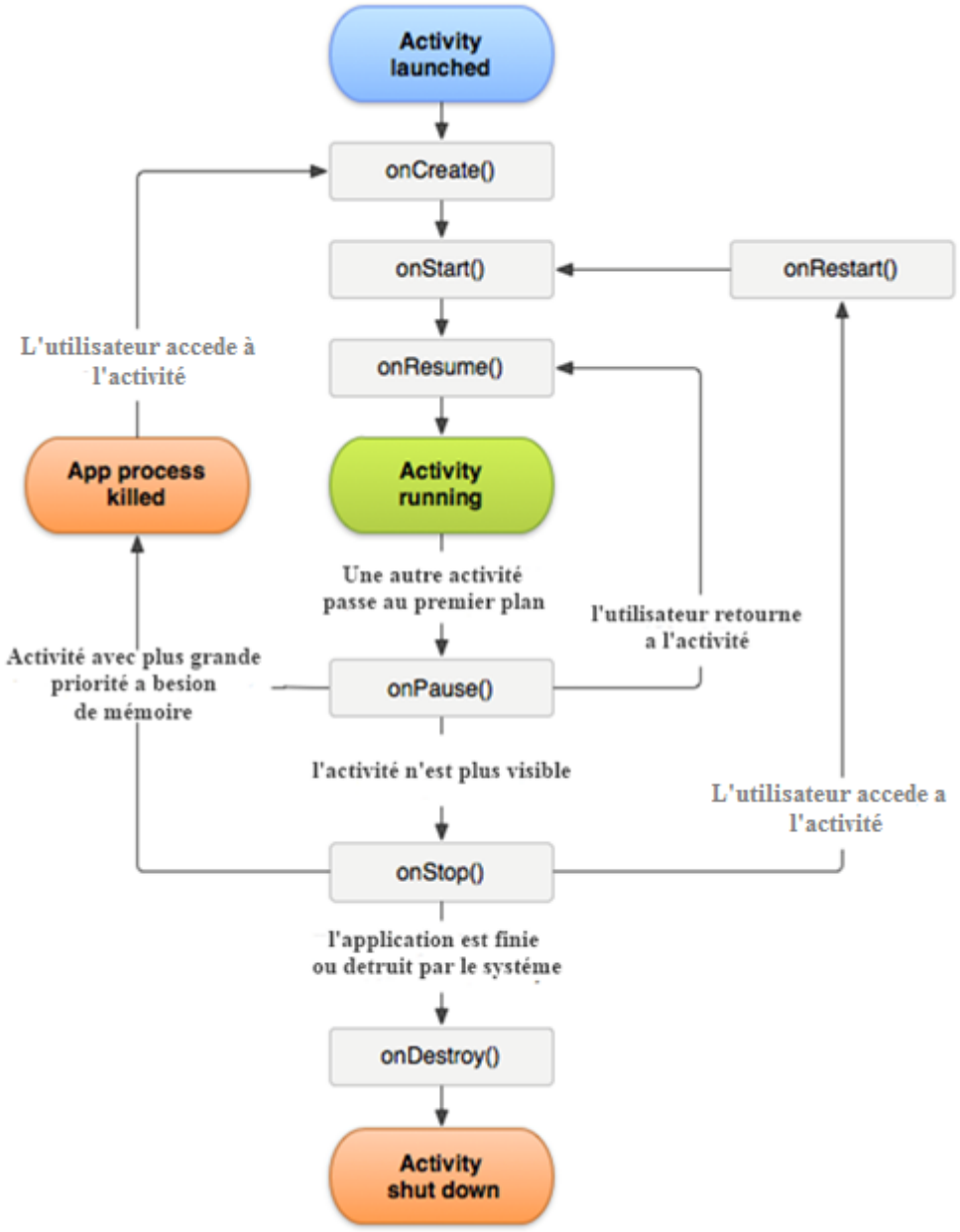
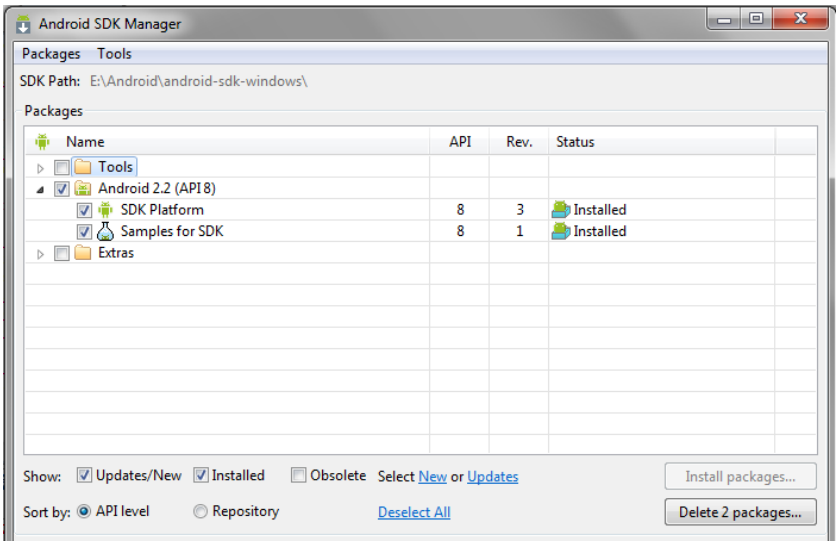



FIGURE 3.1 – Schéma représentant le cycle de vie d'une activité. Source : `developer.android.com`

## 3.3 Instalation du SDK

– Telecharger et installer Eclipse : `http://www.eclipse.org/downloads/`

- Il faut aussi installer JDK [4] et JRE [5]
- Rajouter le nom du site ("ADT plugin" par exemple). Dans la location rajouter : `https://dl-ssl.google.com/android/eclipse/`
- Apres l'installation du ADT [6], cliquer sur SDK Manager, cette fenêtre s'affiche :




- Choisir parmi les différentes versions qui s'affiche (la dernière version actuelle du SDK est 4.0.3)
- exemple : la version 2.2
  - SDK Platform Android 2.2, : Correspond tout simplement au SDK Android basique en version 2.x.
  - Samples for SDK API 8 : Correspond à quelques exemples.
  - Android+Google APIs : Correspond au SDK Android (1ère option)+Google Api qui inclut différentes fonctions comme GoogleMap, etc....
  - Galaxy Tab : C'est le SDK pour la tablette Samsung Galaxy Tab.

Une fois le SDK installé, ces diffèrents icônes (encadré en rouge) apparaissent :

FIGURE 3.2 – Barre d'outils de l'IDE Eclipse

4. JDK : Java Development Kit
5. JRE : Java Runtime Environement
6. ADT : Android Development Tool

 SDK Manager : Pour télécharger, ajouter, ou supprimer des composants du SDK[7].

 AVD[8] Manager : Pour créer, modifier, ou supprimer des émulateurs.

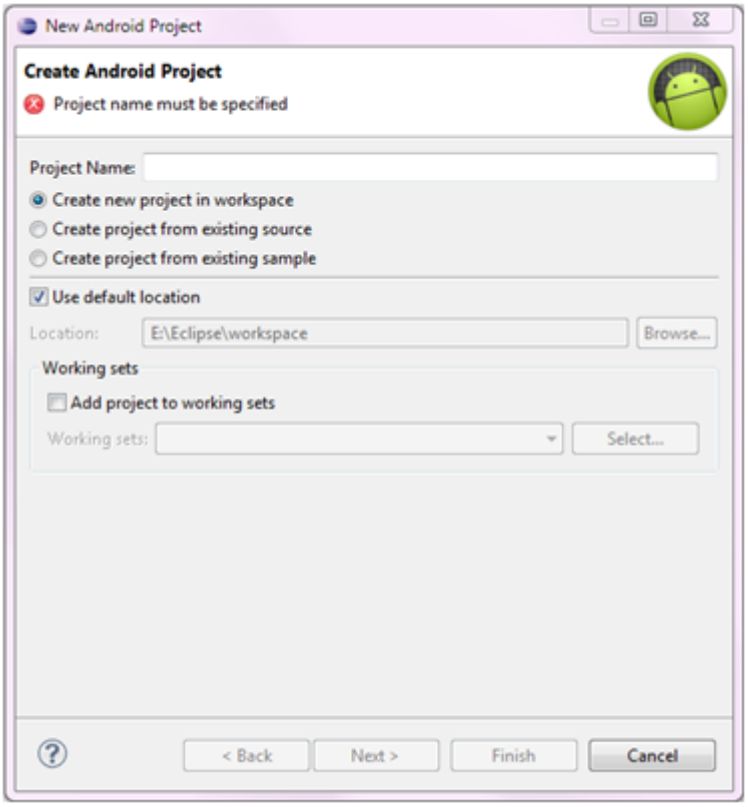
 Pour créer un nouveau projet Android.

 Pour créer un projet de test.

 Pour créer un fichier XML.

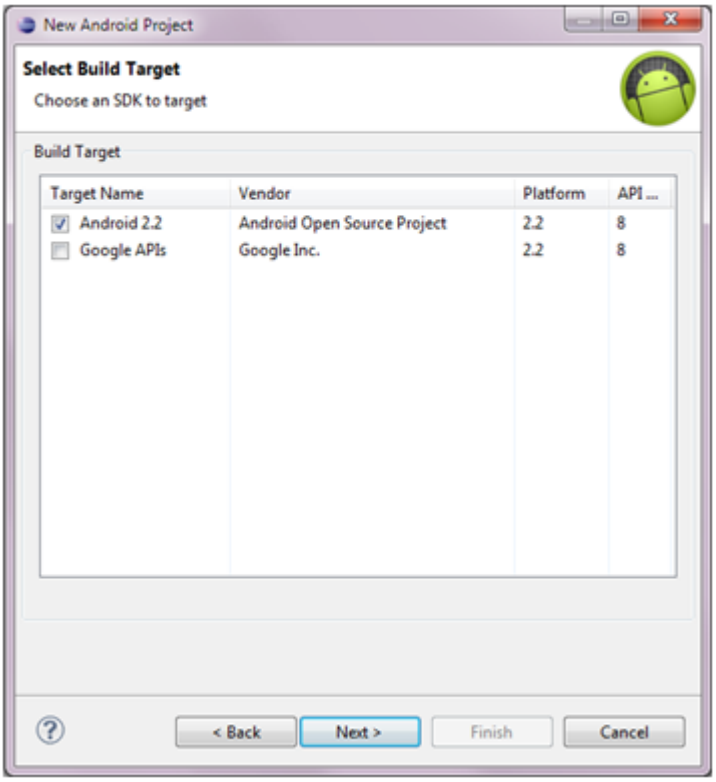
## 3.4 Création d'un projet Android

Pour créer un nouveau projet Android, on n'a qu'à suivre ces étapes : Cliquer sur le bouton "créer un nouveau projet Android", une fenêtre s'affiche :



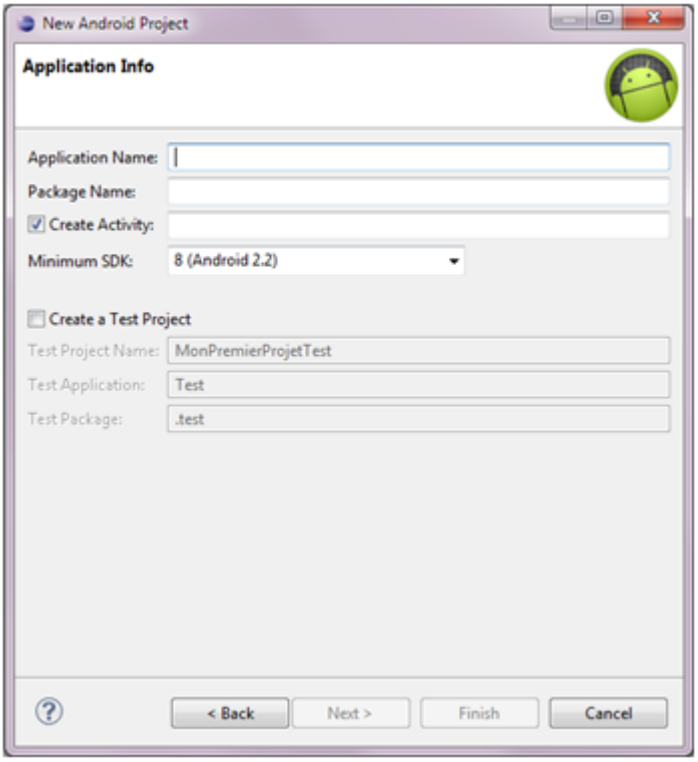

7. SDK : Software Development Kit ou kit de développement logiciel en Francais
8. AVD : Android Virtual Device

- Dans le champ Project Name, ajouter le titre du projet.
- cocher "Create new project in workspace".
- choisir le dossier réception, dans notre cas E :/Eclipse/workspace.
- cliquer sur Next. Une autre fenêtre s'affiche :



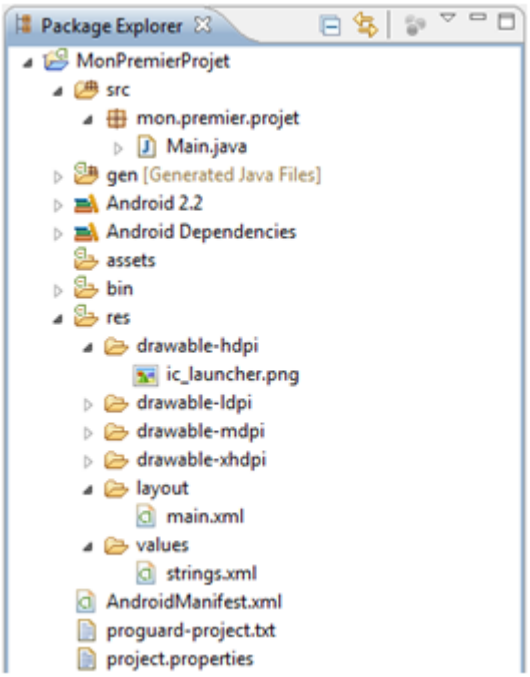

- Choisir la version du SDK. Plus vous avez de version installer, plus vous aurez le choix. (Dans notre cas, on a que la version 2.2 installée).
- Cocher la version choisie. Comme vous le voyez il y a un autre choix (Google APIs). Il est utilisé dans les projets où on utilise les API[9] de Google.
- Apres avoir choisi la version du SDK, cliquer sur Next. Une autre fenêtre s'affiche :

9. API : Application Program Interface




- **Application Name :** contient le nom du projet que vous avez choisi auparavant.
- **Package Name :** chaque projet Android à son propre Package. Le nom du package se compose de trois parties séparé entre elle par des points.(xxx.xxx.xxx) exemple : mon.premier.projet
- **Createactivity :** le nom de activité principale.
- **Minimum SDK :** mis selon la version du SDK que vous avez choisi à l'étape précédente.
- Pour créer un projet de test, vous n'avez qu'à cocher la case "Create a Test Projet" et remplir les différente cases.
- Pour finir, cliquer sur finish.

Une fois crée, notre projet se compose de ces différents éléments.



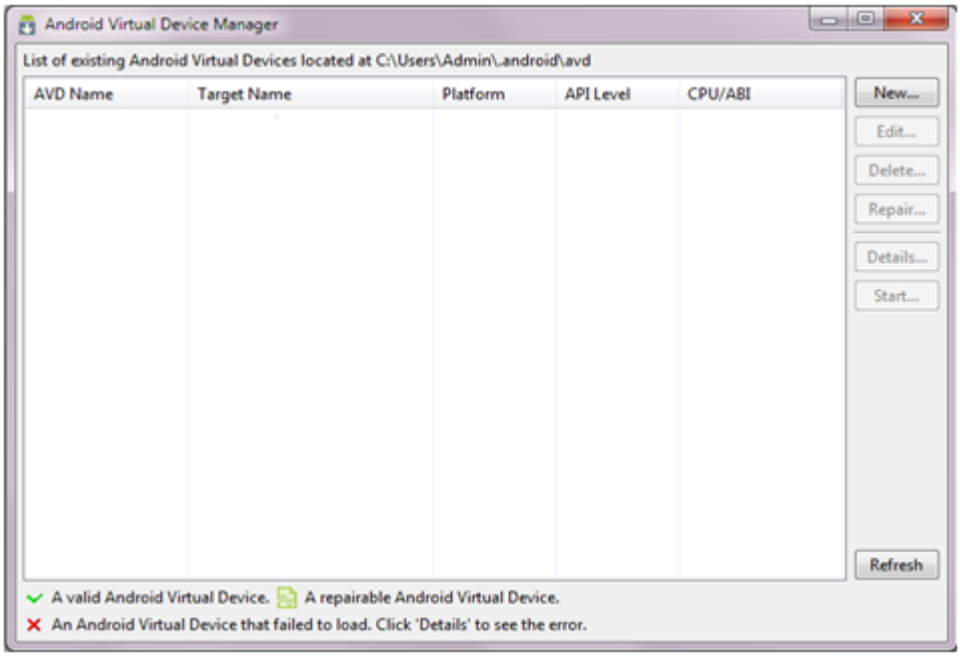

- Le nom du projet, dans notre exemple MonPremierProjet
- Le dossier **src** contient tous les fichiers écris en java y compris l'activité principale (Main.java dans notre exemple).
- Le dossier res(ressources) contient toutes les ressources qui seront mises dans le fichier application. Il est constituéde sous répertoires :
  - drawable-hdpi (images en haute définition)
  - drawable-ldpi (images en basse définition)
  - drawable-mdpi (images en moyenne définition)
  - drawable-xhdpi(images en tres haute definition)
  - layout (description en XML des interfaces)
  - values (définitions en XML de valeurs : chaînes, tableaux, valeurs numériques, etc...)

  On peut aussi trouver d'autres dossier :
  - anim (description en XML d'animations)
  - menu (description en XML de menus pour l'application)
  - xml (fichiers XML utilisés directement par l'application)
  - raw (tous les autres types de ressources : fichiers texte, vidéo, son, etc...)
- Le fichier AndroidManifest contient toutes les informations sur l'application, le nom, la version, le nom des différentes activités, les permissions, etc...

– Les autres dossiers contiennent des fichiers générés par l'IDE[10]. On peut aussi apercevoir un dossier nommé Androidx.x (ou x.x représente la version du SDK. Dans cet exemple 2.2).

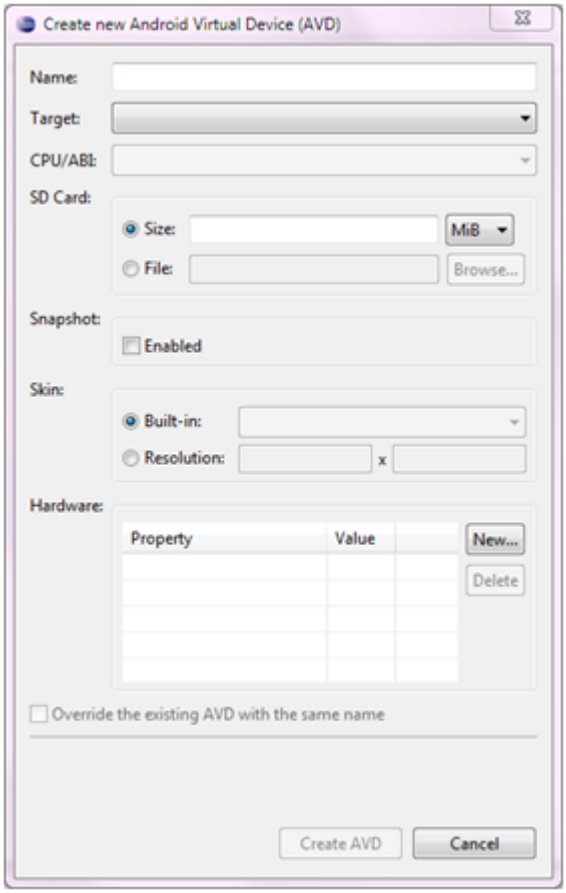
## 3.5 Creation d'un emulateur AVD

Pour créer un AVD (Emulateur), cliquer sur le bouton "creer un AVD", une fenêtre s'affiche :




Cliquer sur New. Une autres fenêtre s'affiche :

10. IDE : Integrated Development Environment, environnement de développement intégré

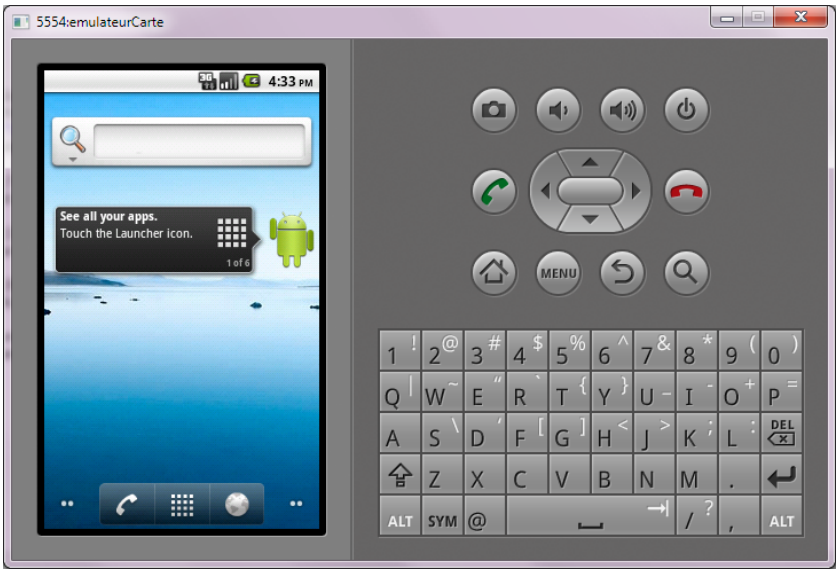



- **Name :** nom de l'emulateur.
- **Target :** version du SDK.
- **SD Card :** Pour créer un dossier qui joue le rôle d'une carte mémoire.
- Snapshot :
- **Skin :** choisir le skin de l'émulateur et sarésolution.
- **Hadware :** ajouter d'autres priorités à l'émulateur comme "activer l'appareil photo ou le GPS".

Une fois ces champs remplis, il ne reste plus qu'à cliquer sur "Create AVD". Exemple d'un AVD :

| AVD Name | Target Name | Platform | API Level | CPU/ABI |
|---|---|---|---|---|
| ✓ emulateur | Android 2.2 | 2.2 | 8 | ARM (armeabi) |

Capture d'écran d'un AVD en éxécution.




## 3.6 Les application réalisées

### 3.6.1 SpeakTra

Dans ce programme on a utilisè :
- Un EditText nommé editTextepour écrire.
- Un Spinner qui contient les differentes langues.
- 4 Boutons :
  - ecouteButton : pour écouter le texte saisie.
  - fermButton : pour fermer l'application.
  - effaceButton : poue effacer le texte saisie.
  - infoBouton : pour afficher les information de l'application.

1. On a créé une classe "Main" qui hérite de la classe *Activity* :
   `public class Main extends Activity`
2. On a affecté à chaque bouton un id (identificateur).
   ```
   fermButton=(Button)findViewById(R.id.ferm_button) ;
   ecouteButton=(Button)findViewById(R.id.button_ecoute) ;
   effaceButton=(Button)findViewById(R.id.efface) ;
   infoBouton=(Button)findViewById(R.id.info) ;
   ```

   Pour donner un id a chaque button, il faut ajouter : `android :id="@+id/xxxx"` (ou xxxx est le nom de id choisi).

@+id (pour dire a l'IDE de créer un id avec le nom xxxx).

3. On a crée un `OnClickListener` avec les differents boutons declarés :
   `fermButton.setOnClickListener(this) ;`
   `ecouteButton.setOnClickListener(this) ;`
   `effaceButton.setOnClickListener(this) ;`
   `infoBouton.setOnClickListener(this) ;`
   Elle prend en paramètre un contexte (dans notre exemple *this* veut dire la class Main)

   La méthode `OnClickListener` permet de rendre les boutons cliquables. Cette méthode doit être implémenter dans la class Main :
   `public class Main extends Activity implements OnClickListener`

   Pour associer une action à un clic, on utilise la méthode : `public public void onClick(View v)`

4. On déclare une variable de type ArrayAdapter(liste) pour y mettre les diffèrentes langues : `public ArrayAdapter listeLangue`

5. On rempli "listeLangue" avec les differentes langues disponibles. Pour cela, on utilise la fonction Locale. `public ArrayAdapter listeLangue = newArrayAdapter(this,android.R.layout.simple_spinner_item, Locale.getAvailableLocales()) ;`

   - android.R.layout.simple_spinner_item : le style du Spinner(liste déroulante).
   - Locale.getAvailableLocales() : pour ajouter toute les langues disponibles dans la bibliothèque du téléphone. On peut utiliser qu'une seul langue :`Locale.ENGLISH` par exemple.
   - android.R.layout.simple_spinner_dropdown_item : le style de l'intérieur du Spinner.

6. On déclare un Spinner : `langueSpinner=(Spinner)findViewById (R.id.spinner) ;`
   `langueSpinner.setAdapter(listeLangue) ;`
   `langueSpinner.setOnItemSelectedListener(this) ;`

   Avec *langueSpinner* comme nom et spinner comme id. Il contient la variable listeLangue. On utilise la méthode `setOnItemSelectedListener` :

Pour réagir. Cette methode doit être implémenter dans la classe *Main* :
`class Main extends Activity implements OnClickListener, OnItemSelectedListener`

7. On crée un Intent nommé "verifier" pour vérifier si la bibliothèque des langues existe ou non : `Intent verifier = newIntent() ;`
   `verifier.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA)` : est-ce qu'elle existe ou non ?

   `startActivityForResult(verifier, 0)` ; la méthode startActivity-ForResult prend deux paramètres : (un Intent, un result code).

8. On ajoute la méthode OnActivityResult qui prend en paramètres (un request code (d'un autre Intent), un result code, un Intent)

9. `if(requestCode==0)`
   `if(resultCode==TextToSpeech.Engine.CHECK_VOICE_DATA_PASS)`
   `TTS=newTextToSpeech(this,this) ;`

10. Pour utiliser le composant `TextToSpeech`, la `methodeOnInitListener` doit implementé la class Main. Ajouter la méthode `OnInit : public void onInit(intetat)`

11. Si le `requestCode==0`, cela veut dire que la bibliothèque existe. Exécuter *TextToSpeech*

12. `else`
    `Intent installer = newIntent() ;`
    `installer.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA) ;`
    `startActivity(installer) ;`
    Créer un autre Intent pour installer la bibliothèque manquante.

13. `public void onInit(intetat)`
    `if (etat == TextToSpeech.SUCCESS)`
    `Toast.makeText(this, "Le programme a été lancé avec succés",`
    `Toast.LENGTH_LONG).show() ;`

```
else

if (etat == TextToSpeech.ERROR)
Toast.makeText(this,"Une erreur s'est produite lors
de l'initialisation", Toast.LENGTH_LONG).show() ;
```

Si la variable `etat == TextToSpeech.SUCCESS` cela veut dire que l'application a été chargée avec succès. Il affiche un Toast (un message qui apparait à l'écran et qui a un délai de temps) avec le message "Le programme a été lancé avec succés" pendant un delai `LENGTH_LONG` (2 seconde).

Si la variable `etat == TextToSpeech.ERROR` cela veut dire qu'il y eu une erreur. Il affiche un Toast avec le message "une erreur s'est produite lors de l'itialisation" pendant LENGTH_LONG (2 seconde).

14. On retourne à la méthode OnItemSelected : `TTS.setLanguage (Locale.getAvailableLocales()[arg2]) ;`
    La fontion `setLanguage` permet de modifier la langue.

15. Maintenant, il nous reste plus qu'à paramétrer notre bouton pour qu'il réagit au clique. Pour cela : On utilise la méthode `speak`.
    `switch(v.getId())`

    ```
    caseR.id.button_ecoute :
    TTS.speak(editTexte.getText().toString(), TextToSpeech.QUEUE_ADD,
    null) ;
    ```

## Le code source

```
/**==============SpeakTra V1.0=================
 * |                                          |
 * ========================================== */


package com.android.transdroid;

import java.util.Locale;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.speech.tts.TextToSpeech.OnInitListener;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.Window;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class Main extends Activity implements OnClickListener, OnItemSelectedListener, OnInitListener {

    EditText        editTexte;
    Button          ecouteButton, fermButton, effaceButton, infoBouton;
    Spinner         langueSpinner;
    TextToSpeech    TTS;

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

    requestWindowFeature(Window.FEATURE_CUSTOM_TITLE);
    setContentView(R.layout.main);
    getWindow().setFeatureInt(Window.FEATURE_CUSTOM_TITLE, R.layout.bar_titre);


    fermButton=((Button)findViewById(R.id.ferm_button));
    fermButton.setOnClickListener(this);

    ecouteButton=(Button)findViewById(R.id.button_ecoute);
    ecouteButton.setOnClickListener(this);

    effaceButton=(Button)findViewById(R.id.efface);
    effaceButton.setOnClickListener(this);

    infoBouton=(Button)findViewById(R.id.info);
    infoBouton.setOnClickListener(this);


    editTexte=(EditText)findViewById(R.id.edit1);
    ArrayAdapter listeLangue = new ArrayAdapter(this,android.R.layout.simple_spinner_item ,Locale.getAvailableLocales());
    listeLangue.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

    langueSpinner=(Spinner)findViewById(R.id.spinner);
    langueSpinner.setAdapter(listeLangue);
    langueSpinner.setOnItemSelectedListener(this);


    Intent verifier = new Intent();
    verifier.setAction(TextToSpeech.Engine.ACTION_CHECK_TTS_DATA);
    startActivityForResult(verifier, 0);
}
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if(requestCode==0)
    {
        if(resultCode==TextToSpeech.Engine.CHECK_VOICE_DATA_PASS)
        {
            TTS=new TextToSpeech(this,this);
        }
        else{
            Intent installer = new Intent();
            installer.setAction(TextToSpeech.Engine.ACTION_INSTALL_TTS_DATA);
            startActivity(installer);
        }
    }
}

public void onClick(DialogInterface arg0, int arg1) {
    // TODO Auto-generated method stub

}


@Override
public void onItemSelected(AdapterView<?> arg0, View arg1, int arg2, long arg3) {
    if(TTS!=null)
        TTS.setLanguage(Locale.getAvailableLocales()[arg2]);
}


@Override
public void onNothingSelected(AdapterView<?> arg0) {
    // TODO Auto-generated method stub
}

@Override
public void onInit(int etat) {
    if (etat == TextToSpeech.SUCCESS) {
        Toast.makeText(this, "Le programme a été lancé avec succés", Toast.LENGTH_LONG).show();
        }
    else
    {
        if (etat == TextToSpeech.ERROR)
        Toast.makeText(this,"Une erreur s'est produite lors de l'initialisation", Toast.LENGTH_LONG).show();
    }
}

@Override
public void onClick(View v) {
    switch(v.getId())
    {

        //le bouton qui permet de declancher la methode speak
        case R.id.button_ecoute:
            String texte = editTexte.getText().toString();
            if (texte!=null && texte.length()>0) {
            Toast.makeText(Main.this, "Ecoutez : " + texte, Toast.LENGTH_LONG).show();
            TTS.speak(editTexte.getText().toString(), TextToSpeech.QUEUE_ADD, null);
            }
            else
            {
                Toast.makeText(Main.this, "Veuillez ajouter du texte", Toast.LENGTH_SHORT).show();
            }
        break;


        case R.id.efface:
             editTexte.setText("");
        break;
```

```
        // Un bouton pour fermer l'application
        case R.id.ferm_button:
            AlertDialog.Builder ad = new AlertDialog.Builder(this);
            ad.setMessage("Voulez-vous vraiment quittez le programme ?");
            ad.setTitle("Quitter");
            ad.setIcon(R.drawable.avert);
            ad.setCancelable(false);
            ad.setPositiveButton("Oui", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                        Main.this.finish();
                       }
            });
            ad.setNegativeButton("Non", new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                        dialog.cancel();
                       }
            });
            AlertDialog alert = ad.create();
            alert.show();
        break;

        // Un bouton pour afficher les information de l'application
        case R.id.info:
            final Dialog dialog = new Dialog(Main.this);
            dialog.setContentView(R.layout.fenetre_info);
            dialog.setTitle("Information sur le programme :");
            dialog.setCancelable(true);

            TextView text = (TextView) dialog.findViewById(R.id.TextView01);
            text.setText(R.string.TexteInfo);

            ImageView img = (ImageView) dialog.findViewById(R.id.ImageView01);
            img.setImageResource(R.drawable.speaktra_icon);

            Button button = (Button) dialog.findViewById(R.id.ButonAnnuler);
            button.setOnClickListener(new OnClickListener() {
                    public void onClick(View v) {
                        dialog.cancel();
                    }
                });

                dialog.show();

    }
  }
}
```

### 3.6.2 MapDroid

Dans cette application, on a utilisé :

**variables :**
*Carte* de type `MapView`. Pour afficher la carte.
*boutonClique* de type `clique` (nous allon voir ce que fait cette fonction).
Deux variable de type `long` : *debut, fin.*

**Methode :**
`isRouteDisplayed.`
`onTouchEvent.`
`MotionEvent.`
`AlertDialog.`

Pour créer une application qui utilise MapView, il faut une clé (API Key). Pour plus d'informations : `https://developers.google.com/maps/documentation/android/mapkey?hl=fr`

1. On a créé une classe "Main" qui hérite la classe `MapActivity` :
   `public class Main extends MapActivity`.
   La classe *MapActivity* requiert d'ajouter la méthode `isRouteDisplayed()` :
   ```
   protected boolean isRouteDisplayed()
   // TODO Auto-generated method stub
   return false ;
   ```

2. `carte=(MapView)findViewById(R.id.carteView) ;`
   Cette ligne permet d'identifier la variable *carte* avec son id *carteView* (qu'on a choisi). Elle permet aussi d'afficher la carte à l'écran.

   `carte.setBuiltInZoomControls(true) ;`
   Pour active le bouton zoom sur la carte.

3. On a créé la fonction `cliquer()` afin de créer un long click (un clic long) et de lui associer une action.
   Pour cela :
   Nous utiliserons non deux variable *debut* et *fin* pour leur affecter le temps du premier clique et le temps deuxième.

   `public boolean onTouchEvent(MotionEvent e, MapView m)`
   on utilise la methde `onTouchEvent`. Elle prend comme parametre un *MotionEvent* et un *MapView*

   ```
   if(e.getAction() == MotionEvent.ACTION_DOWN)
   debut = e.getEventTime() ;
   ```
   Si l'action est un clic de souris (bas). La variable debut recoit le moment où l'utilisateur a fait un clic vers le bas.

   ```
   if(e.getAction() == MotionEvent.ACTION_UP)
   fin = e.getEventTime() ;
   ```
   Si l'action est un clic de souris (relaĉher le bouton). La variable fin reçoit le temps ou le clic a terminé.

4. `if((fin-debut) > 2000)`
   Si le temps du clic a duré 2 seconde (2000 milliseconde).

On a créé un `AlertDialog` (une fenêtre qui contient des boutons).

```
AlertDialog boite = newAlertDialog.Builder(Carte.this).create() ;
boite.setTitle("Changer de vue") ; boite.setMessage("Ce bouton
vous permet de switcher entre la vue carte et la vue satellite") ;
boite.setIcon(R.drawable.icone) ;
```

- `boite.setTitle` : le titre de la fenêtre.
- `boite.setMessage` : le message qui s'affiche sous le titre.
- `boite.setIcon` : pour changer l'icone de la fenêtre.

5. `boite.setButton("Carte/Sattelite", new DialogInterface.OnClickListener()`
   Pour créer le bouton qui nous permet de switcher entre les vues.

```
public void onClick(DialogInterface arg0, int arg1)
// TODO Auto-generated method stub
if (carte.isStreetView())
carte.setStreetView(false) ;
carte.setSatellite(true) ;
```

Si la vue est en StreetView changer la en vue Satellite.

```
else
carte.setStreetView(true) ;
carte.setSatellite(false) ;
```

Sinon.

## Code source

```
package com.android.mapdroid;

import java.util.List;

import com.google.android.maps.MapActivity;
import com.google.android.maps.MapView;
import com.google.android.maps.Overlay;

import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.MotionEvent;

public class Carte extends MapActivity {

    MapView carte;
    clique boutonClique = new clique();
    long debut,fin;


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        carte=(MapView)findViewById(R.id.carteView);
        carte.setBuiltInZoomControls(true);

        List<Overlay> liste = carte.getOverlays();
        liste.add(boutonClique);

    }

    @Override
    protected boolean isRouteDisplayed() {
        // TODO Auto-generated method stub
        return false;
    }
class clique extends Overlay{
    public boolean onTouchEvent(MotionEvent e, MapView m){
        if(e.getAction() == MotionEvent.ACTION_DOWN)
            debut = e.getEventTime();


        if(e.getAction() == MotionEvent.ACTION_UP)
            fin = e.getEventTime();

        if((fin-debut) > 2000){
            AlertDialog boite = new AlertDialog.Builder(Carte.this).create();
            boite.setTitle("Options");
            boite.setMessage("Choisissez une option");

            boite.setButton("Carte/Sattelite", new DialogInterface.OnClickListener() {

                @Override
                public void onClick(DialogInterface arg0, int arg1) {
                    // TODO Auto-generated method stub
                    if (carte.isStreetView()){
                        carte.setStreetView(false);
                        carte.setSatellite(true);
                    }
                    else
                        {
                            carte.setStreetView(true);
                            carte.setSatellite(false);
                        }
                }
            });

            boite.show();
            return true;
        }
```

```
        return false;
        }
    }
}
```

## 3.7 Conclusion

Nous avons décri dans ce dérnier chapitre la démarche suivie pour développer des applications sous Android, et nous avons donné deux exemples concrés, le premier fait la transformation du texte en parole ; quand au second, il permet de consulter une carte en ligne.

# Conclusion générale

Nous avons étudié dans ce mémoire les systèmes d'exploitation pour l'embarqué, et plus spécialement le fameux système Android. Nous avons tout d'abord commencé à définir ce que c'est qu'un système d'exploitation, ce que c'est qu'un système embarqué. Ensuite nous avons étudié de plus près le système Android et son architecture logicielle. Enfin, nous avons montré comment développer une application pour Android, deux exemples illustratifs ont été programmés.

Ce projet de fin d'étude nous a été très bénéfique, il nous a permis de mieux comprendre ce que c'est qu'un système embarqué, comment comprendre le comportement d'un logiciel (ou plus précisément un système d'exploitation) dans un environnement qui n'est absolument pas classique (système embarqué). Nous avons aussi approfondi nos compétences dans la programmation objet spécialement dans Java.

Pour le futur, nous envisageons continuer dans ce domaine très prometteur, nous espérons pouvoir être de très fameux programmeurs de systèmes d'exploitation pour l'embarqué.

# Bibliographie


1. **Livre**
   [1] Linux embarqué 2e edition Pierre Ficheux EDITIONS EYROLLES 61, bd Saint-Germain 75240 Paris Cedex 05 `www.editions-eyrolles.com` ISBN : 2-212-11674-8. [2] Créez des applications pour Android Frédéric Espiau `www.siteduzero.com` Licence Creative Commons BY-NC-SA 2.0 - 2011. [3] Les Systèmes Embarqués Linux pour l'embarqué Patrice KADIONIK `http://www.enseirb.fr/~kadionik`. [4] Linux embarqué, Linux Temps Réel : présentation Patrice KADIONIK `http://www.enseirb.fr/~kadionik`. [5] Les systèmes d'exploitation pour l'embarqué J.Boukhobza Université de Bretagne Occidental.
2. **Site web**
   [1] http ://generation-droide.fr/2011/07/28/un-petit-peu-dhistoire-autour-dandroid/
   [2] http ://fr.wikipedia.org/wiki/Android
   [3] http ://fr.wikipedia.org/wiki/Historique_des_versions_dandroid
   [4] http ://www.javacodegeeks.com/
   [5] http ://stackoverflow.com/
   [6] http ://developers.google.com/
   [7] http ://www.Androideur.com
   [8] http ://android-er.blogspot.com/
   [9] http ://d.android.com/
   [10] http ://notesofgenius.com/
   [11] http ://en.wikipedia.org/wiki/Android_operating_system
   [12] http ://www.technologuepro.com/cours-genie-electrique/cours-12-systemes-embarques/
   [13] http ://fr.wikipedia.org/wiki/IPhone_OS

3. **Photo**
   [1] http ://www.arm.com/community/partners/display_product/rw/ProductId/2442/
   [2] http ://www.ilovetablette.com/la-connectivite-dans-l%E2%80%99automobile-avec-qnx-de-rim

[3] http ://www.accelerance.com/outsourced-software-development-companies-by-technology/
[4] http ://en.wikipedia.org/wiki/Windows_CE
[5] http ://www.ecoscentric.com/trademark_usage.shtml
[6] http ://www.gizmodo.fr/2012/01/23/blackberry-os-pourrait-etre-propose-sous-forme-de-licence.html
[7] http ://www.mobiflip.de/details-zur-hardware-der-meego-entwicklungsplattform/