2011
2012

Projet tutoré : application pour C4

Emmanuel PENOT
License pro SARI
2011 2012

Sommaire

Présentation du projet

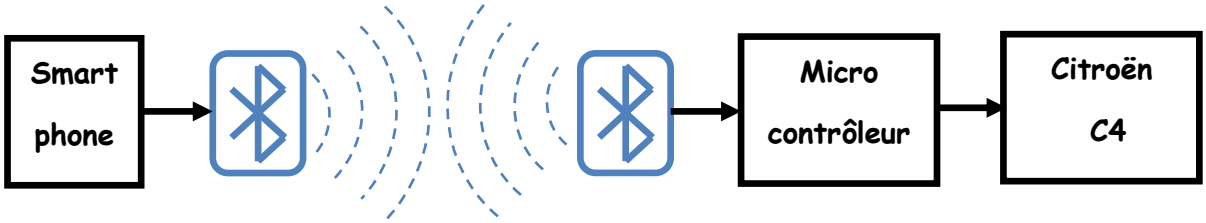
I. Contexte du projet :

Suite aux applications réalisées sur le véhicule et présentées lors des manifestations (fêtes de la science et journées portes ouvertes), il s'agit de trouver de nouveaux équipements à interroger ou à commander sur le véhicule. Pour cela, on analysera les trames du BSI (Boitier de Servitude Intelligent) avec un analyseur CAN de chez Vector, on les décodera et elles serviront à constituer une base de données. Une fois les principales trames récupérées, on contribuera à améliorer l'interface du téléphone portable afin d'y inclure des fonctions de diagnostic et d'autres fonctions issues des trames décodées précédemment.

Sur le terrain, nous avons pu découvrir qu'une base de données existait déjà et que son contenu avait permis de créer une table de transcodage. Après présentation des équipements en activité (Citroën C4, téléphone sous OS Windows mobil, carte de connexion Bluetooth et interface de dialogue Bluetooth-micro-contrôleur), le projet s'orienta davantage à améliorer l'interface graphique et les fonctions de l'application existantes. Ces améliorations s'inscrivent dans le cadre d'un portage de l'application depuis un OS Windows mobile vers un smart phone sous OS Androïd.

II. Fonctionnement global du projet :

Après étude des projets et applications précédents nous avons déduit le fonctionnement suivant :



Lors d'un contact sur l'écran tactile du smart phone, les paramètres et les données sont mis dans une trame appelée « trame maison », puis envoyés via une connexion sans fil de type Bluetooth. On retrouve la « trame maison » dans la partie données de la trame Bluetooth. Une fois cette dernière reçue par le micro contrôleur, ce dernier envoie les ordres, sous forme de trame CAN, sur les réseaux de la Citroën C4.

III. Organisation pour la réalisation :

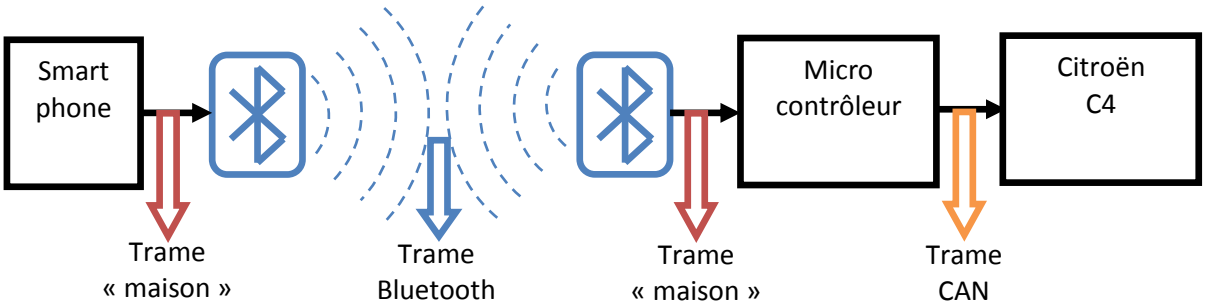
Afin de répondre à cette demande nous avons choisi de diviser le projet en 3 parties qui seront autonomes mais complémentaires. Ainsi les parties seront :

- compréhension de la transmission pour développement du fonctionnement théorique de l'application.
- création de l'interface graphique pour smart phone avec les inter-connectivités entre les différents écrans.
- programmation fonctionnelle de l'application : construction de trame et configuration du port série Bluetooth.

Fonctionnement théorique

I. Transmission de commande depuis un smart phone

Pour créer un fonctionnement théorique d'envoi de trame, il m'a fallu préalablement comprendre ce fonctionnement de façon précise. Ainsi, si l'on reprend le schéma du fonctionnement global de façon complète, on obtient cela :



II. Localisation du protocole :

Suite à une 1ère analyse, on distingue 2 parties dans la transmission. La partie smart phone en amont de la connexion Bluetooth et la partie C4 en aval. On peut donc exprimer le schéma 1. Pour notre projet nous devons comprendre la partie smart phone pour la migration sous Androïd, et particulièrement sur la composition de la trame « maison » qui sera le format unique pour communiquer avec le véhicule par la fonction de transcodage du micro contrôleur.

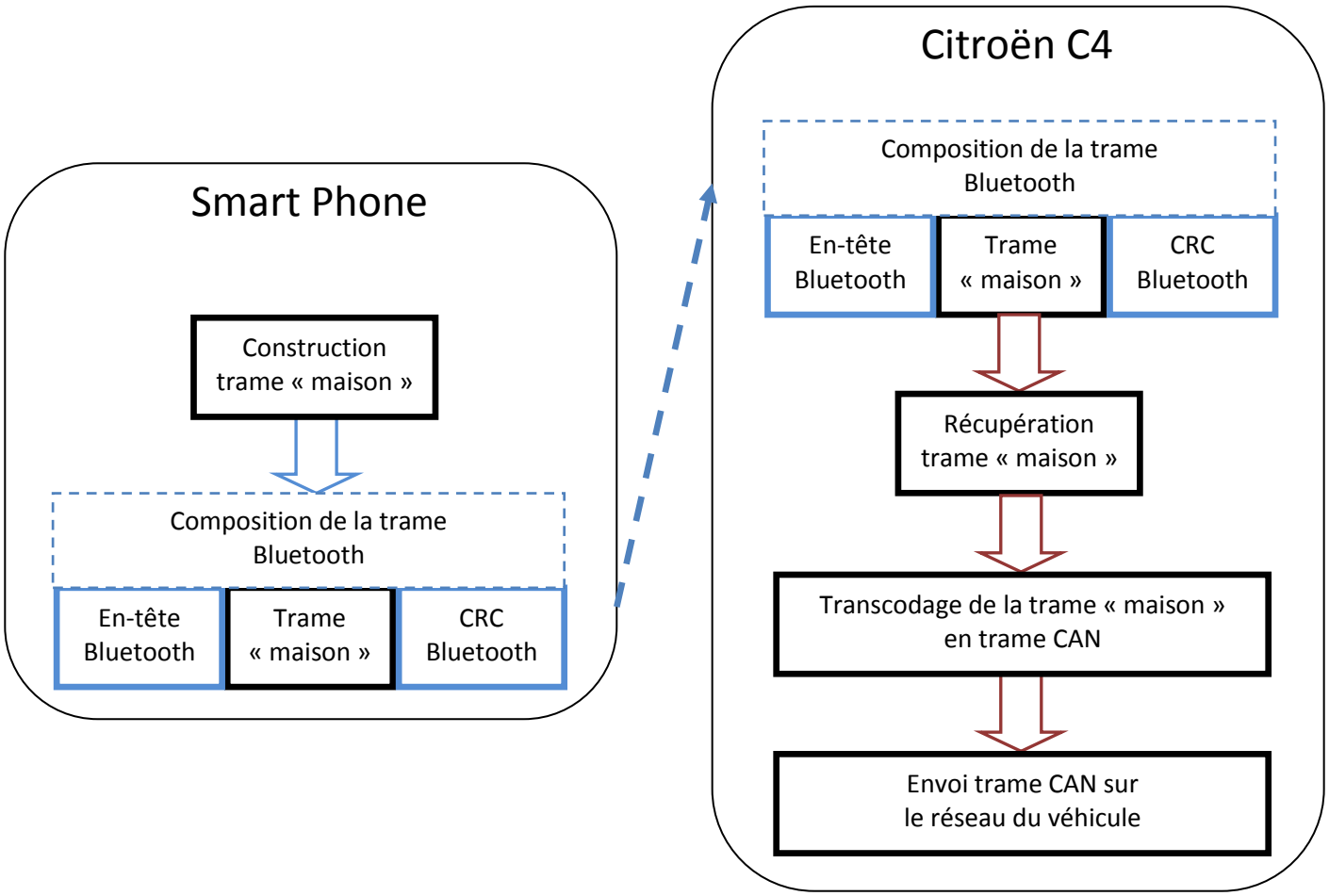


Schéma 1

III. La trame « maison » :

Après quelques recherches dans le programme développé sous Labwiew, on distingue 2 types de trames ainsi que le format de ces dernières. En fonction si ce sont des actions d'activation/désactivation ou des transferts de données, on note des différences de contenu mais pas de format. J'ai donc constitué un tableau de référence qui regroupe format de la trame, composition et contenu pour les différents cas.

	1 Octet de start	2 octets de données		1 octet d'activation	1 octet séparateur	ID
Format	0xFF	0x_ _	0x_ _	0 ou 1	0	1 à 255
Trame booléenne	0xFF	0x00	0x00	0 ou 1	0	
Trame de données	0xFF	0x_ _	Valeur>0	1	0	
Variable dans le programme	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6

ID : Identificateur, il identifie chaque fonction avec un numéro unique qui permet au micro contrôleur ensuite de faire la liaison avec le réseau CAN.

Pour une trame de type booléenne, les paramètres importants seront son ID et l'état de l'octet d'activation. En revanche, pour une trame de type données, ce sera : son ID et sa valeur sur 2 octets, l'octet d'activation étant toujours à 1.

Il sera donc obligatoire de prévoir 2 procédures distinctes pour construire ces trames.

De plus, pour construire une trame, on a besoin d'utiliser une chaine de caractères. Qui elle-même est le résultat d'une concaténation des différents membres préalablement transformés d'entier en caractères ascii.

IV. Communication via Bluetooth :

Dans notre projet, la communication entre les appareils sera configurée comme une communication port série. Il faut néanmoins, avant de procéder à tout échange, réaliser la connexion selon le protocole Bluetooth.

Pour cela, il faudra réaliser les opérations suivantes sur le mobile :

- Activation du module Bluetooth du smart phone
- Recherche du module du véhicule
- Affiliation au véhicule
- Connexion au véhicule
- Création de socket de communication
- Déroulement du programme

Interface graphique

I. Choix de la technique :

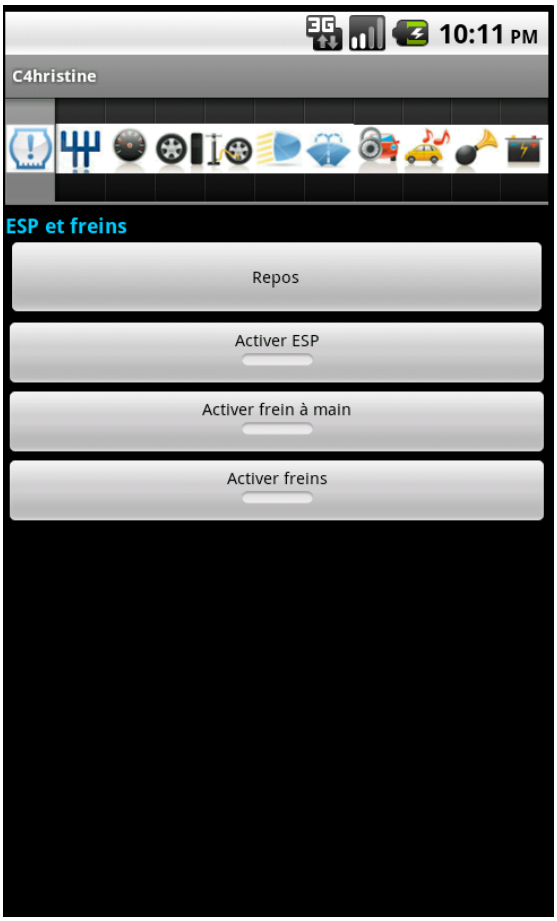
Dans la programmation sur Android, il y a 2 possibilités pour réaliser l'interface graphique : une interface fixe réalisée dans le fichier « main.xml » ou avec un parseur XML dont la ressource sera importée. C'est cette 2nde option technique qui fut choisie dans la réalisation du projet.

Quand on analyse les commandes qui sont référencées, on peut les répartir dans 12 groupes, sous la forme d'onglets dans l'interface graphique. Chacune de ces commandes se verra, en fonction de son rôle, affecter un type de composant graphique le plus approprié pour son rôle dans le programme *(voir annexe I).*

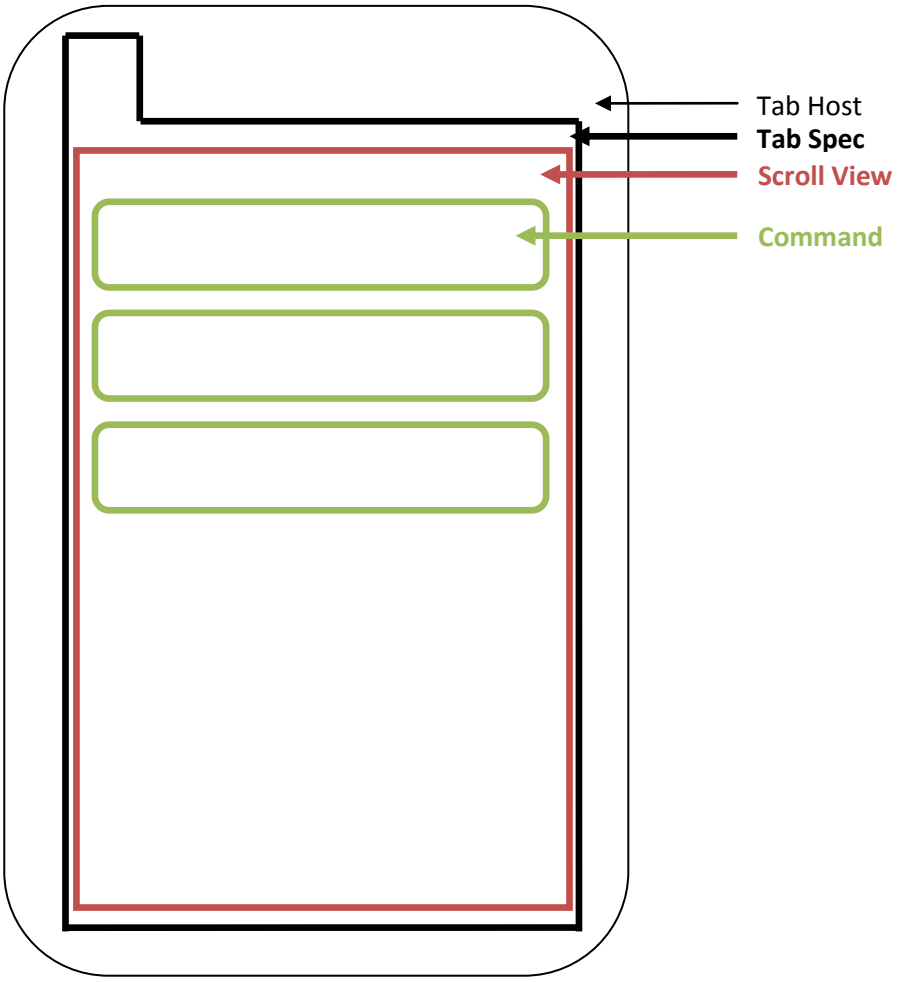
Dans la perspective d'évolution et de mise à jour constante des fonctionnalités de l'application, l'interface dynamique reste la solution la plus appropriée.

II. Les interfaces sous Androïd :

Les interfaces sous OS Androïd se basent le fonctionnement en « modèles de boites » ; ainsi un bouton mis dans un onglet hérite des caractéristiques de dimension et d'ajustement de ce dernier. Voici le principe choisi pour ce projet :



(Ecran d'accueil)



(Schéma théorique)

Comme le schéma l'explique, y a une page à onglet (Tab Host), qui est hôte des différents onglets (Tab Spec). Dans les onglets, on place une page dont la vue est « scrollable » (Scroll View), enfin dans cette dernière sont insérées les différentes commandes (Command) les unes sous les autres. On retrouve donc en lien d'emboitement :

Command → Scroll View → Tab Spec → Tab Host

III. Parseur XML :

Le parseur XML permet l'analyse décomposée d'un fichier XML. Il identifie les différentes sections (éléments, attributs) et permet de brancher des fonctions qui assureront le comportement souhaité.

Ici, ces fonctions assureront la création des composants graphiques selon les règles d'emboitement établies par la ressource. Cette méthode permet de construire une interface de manière dynamique et parfaitement modulable au fil des améliorations et mises à jour.

Dans notre application, nous avons défini plusieurs balises qui se décomposent de la manière suivante :

	Eléments lus	DataC4	Groupe	Choix	Command
Fichier XML	Attributs associés		Identifiant interne, titre, logo	Identifiant interne, titre	Identifiant interne, type, titre.
Interface graphique	Objet construit	Tab Host	Tab Spec	Radiogroup	Button, toggleButton, EditText, TextView

(*fichier source : annexe II*)

D'un point de vue programmation, les fonctions de constructions d'objet sont surchargées afin de pouvoir conserver l'identification des commandes et leurs modalités d'exécution. De plus une fois l'objet affiché, on viendra lui associer les procédures événementielles, ces dernières appelleront les fonctions de construction de trame correspondants à leur type.

Les commandes en plus des propriétés génériques possèdent un « type » et en fonction de ce dernier des paramètres supplémentaires.

Type	Id	textOn	textOff	label	Commentaire
bool	Oui	Oui	Oui	Non	Les ID des commandes servent dans le programme pour les identifier, mais sont aussi un paramètre de la trame
Pulse	Oui	Non	Non	Oui	
State	Oui	Non	Non	Oui	
Data	Oui	Non	Non	Oui	

(fichier source : annexeII)

Programmation fonctionnelle

I. Configuration du port série Bluetooth :

Pour établir la communication entre le smart phone et le véhicule par le réseau Bluetooth, il nous faut paramétrer celle-ci. Pour les OS Androïd, le système Bluetooth est divisé en 3 parties dans la programmation :

- L'Adapter : qui défini les paramètres et les actions sur le port local
- Le Device : qui défini les propriétés des périphériques extérieurs
- Le Socket : qui gère les connexions ainsi que les protocoles d'émission/réception.

1) Activation du Bluetooth du smart phone :

Il est possible sous Androïd de pouvoir activer le Bluetooth du smart phone sans demander l'avis de l'utilisateur. Cela permet ainsi une meilleure fluidité et prise en main de l'application. Pour cela, il faut prévoir les permissions dans le fichier « AndroidManifest », en insérant les lignes suivantes:

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

(voir annexeIV)

Comme le forçage d'une activation Bluetooth ne peut se réaliser sur un smart phone qui ne possède pas cette technologie, il sera réalisé un test de présence avant toute activité. Ainsi un utilisateur sans connexion pourra tout de même parcourir les menus sans problème. C'est la fonction « enable() » du Bluetooth adapter qui déclenchera l'activation du port Bluetooth.

2) Récupération du périphérique Bluetooth du véhicule :

Puisqu'on connait l'adresse Mac du véhicule (stockée dans une variable DEVICE_ADRESSE), il est possible de se connecter directement à celui-ci sans devoir effectuer une recherche du périphérique (fonction getRemoteDevice()). L'utilisateur n'aura qu'à accepter la demande d'association en tapant le code 1234 lors de la 1ere utilisation. De cette façon le bloc Bluetooth se connectera uniquement au module du véhicule.

3) Connexion et paramétrage de la communication :

Pour finir, on va créer un socket, issu et conçu spécialement pour cette communication. Pour cela la fonction « createRfcommSocketToServiceRecord()», sera utilisée. On lui passera en paramètre un type UUID. L'UUID est un identifiant unique qui défini le type de communication, dans le projet on souhaite une communication port série.

Dans le projet, on a une situation client/serveur, respectivement smart phone/véhicule. On va donc demander une connexion au serveur par la fonction «connect() ». Si ce dernier accepte, on pourra alors établir les canaux de transmission. Dans ce cas les fonctions « getInputStream » et « getOutStream », configure respectivement les canaux d'entrée et de sortie.

Maintenant la connexion est configurée, on peut dialoguer avec le véhicule.

II. Construction et envoi des trames:

1) Construction :

Dans cette partie fonctionnelle du projet, la contrainte principale est d'avoir en fin de processus une variable qui soit au format de la trame « maison ». Ce sera donc une chaine de 6 caractères ascii, dont les valeurs en octets représentent les paramètres de chaque commande. Rappel, l'id est un identificateur unique pour un transcodage que réalise le micro contrôleur à réception de la trame « maison ».

Voici donc le fonctionnement qui est mis en place pour remplir ces contraintes :

Cas de commande Booléen	Octet de start	2 Octets de données		Octet d'état	Octet séparateur	Octet ID
Evénement	L'utilisateur touche une commande, appel de la fonction sendCommandBool(id, etat)					
Valeur en integer	0xFF	0	0	Oct_etat = etat	0	ID = id
Transformation en caractères ascii	«ÿ»	« null »	« null »	« null » ou « SOH »	« null »	Tous caractères possibles
Concaténation	Trame = «ÿ» «null» «null» «null» « null ou SOH » «null» « »					
Envois en bytes successif	255	0	0	1 ou 0	0	0-255

Cas de commande Data	Octet de start	2 Octets de données		Octet d'état	Octet séparateur	Octet ID
Evénement	L'utilisateur touche une commande, appel de la fonction sendCommandData(id, value)					
Valeur en integer	0xFF	0	0	1	0	ID = id
Evénement	Pré-traitement de value, en fonction des commandes (exemple la vitesse)					
Evénement	Mise à l'échelle de value sur 2 octets de données, défini par le programme sur Labview					
Transformation en caractères ascii	«ÿ»	« »	« »	« SOH »	« null »	Tous caractères possibles
Concaténation	Trame = «ÿ» « » « » «null» «SOH» «null» « »					
Envois en bytes successif	255	0-255	0-255	1	0	0-255

(Programme complet : annexe V)

2) Envoi / réception de trame :

Comme on utilise une communication par port série, il faut prévoir des buffers en envoi et en réception. Le protocole est déjà établi dans la classe BluetoothSocket, il suffit d'utiliser les fonctions : « write() » et « read() ».

L'envoi étant réalisé octet par octet, il nous faudra placer la chaine de caractère « trame » dans un tableau de byte puis les envoyer 1 à 1. Néanmoins, les bytes sont signés et par conséquent l'octet de start qui a pour valeur 255 devra être envoyé avant et le reste de la trame ensuite.

Pour la réception, le processus sera l'inverse de l'envoi. Récupérer les octets 1 par 1 puis les placer sous forme de trame maison pour les traiter et faire interagir l'application en conséquence.

Conclusion

Durant ce projet, j'ai pu apprendre à développer sous java en découvrant l'IDE Eclipse dans le cadre d'une application Android, créer une interface graphique dynamique et une communication port série via Bluetooth.

La démarche de recherche et de compréhension des fonctionnements et des protocoles reste pour moi la partie qui m'a le plus enthousiasmé et apporté sur le plan personnel et professionnel.

Actuellement, l'application est capable de se connecter au véhicule, créer une interface graphique dynamique et fonctionnelle et commander 61 fonctions sur le véhicule ; malheureusement la récupération d'information telle que la pression des pneus n'a pas était menée à bien.

Les perspectives d'avenir de ce projet peuvent s'orienter sur 2 axes :

- Compléter les fonctionnalités du programme en programmant de nouvelles commandes
- Améliorer l'interfaces graphiques avec l'intégration de nouvelles commandes et l'optimisation de l'esthétique

Remerciements et sources

Je tiens à remercier, Christian PECOSTE ; pour son aide sur les comportements de la C4 ainsi que pour les documents et applications mis en place précédemment ; l'équipe de Master constituée de Roth MVOULA, Jean-Pierre CONAN, Mohandas LAOUROU et Abdi GUELLEH ; qui ont pris le temps de m'expliquer les bases du module Bluetooth et leur motivation qu'ils ont su me faire partager, sans compter les nombreuses sources d'information qu'ils m'ont fait parvenir. Je remercie aussi toutes les personnes qui ont su m'aider à résoudre mes difficultés ponctuelles : Timothee LEVI, Serge BOUTER, et tous les autres ; merci encore.

Pour mettre en place ce projet, voici les sources qui me furent des plus utiles :

Liaison Bluetooth :

- http://developer.android.com/resources/samples/BluetoothChat/index.html
- http://sberfini.developpez.com/tutoriaux/android/bluetooth/
- http://nononux.free.fr/index.php?page=elec-brico-bluetooth-android-microcontroleur

Langage Java :

- http://www.siteduzero.com/tutoriel-3-10601-apprenez-a-programmer-en-java.html
- http://www.oracle.com/fr/technologies/java/index.html
- http://java.developpez.com/cours/

Annexes

I. Tableaux de références des entrées sorties :

Numéro ID	Nom de la commande	Entrés / sorties	Type	Objet dans le programme
0	Repos	Sortie	pulse	Button
1	ESP	Sortie	bool	Toggle button
2	Frein à main	Sortie	bool	Toggle button
3	Frein	Sortie	bool	Toggle button
4	Mode neige	Sortie	bool	Toggle button
5	Mode sport	Sortie	bool	Toggle button
6	Activer levier de vitesse	Sortie	select	Radiogroup
7	Position parking	Sortie	select	Radiogroup
8	Position neutre	Sortie	select	Radiogroup
9	Position arrière	Sortie	select	Radiogroup
10	Désactiver boite auto	Sortie	select	Radiogroup
11	Position 1ere vitesse	Sortie	select	Radiogroup
12	Position 2nde vitesse	Sortie	select	Radiogroup
13	Position 3ime vitesse	Sortie	select	Radiogroup
14	Position 4ime vitesse	Sortie	select	Radiogroup
15	Position 5ime vitesse	Sortie	select	Radiogroup
16	Position 6ime vitesse	Sortie	select	Radiogroup
17	Angle du volant	Sortie	data	Edittext
18	Vitesse	Sortie	data	Edittext
19	Distance roue	Sortie	data	Edittext
20	Accélération	Sortie	data	Edittext
21	Roue AVG normal	Sortie	select	Radiogroup
22	Roue AVG fortement dégonflée	Sortie	select	Radiogroup
23	Roue AVG faiblement dégonflée	Sortie	select	Radiogroup
24	Roue AVG crevée	Sortie	select	Radiogroup
25	Roue AVD normal	Sortie	select	Radiogroup

26	Roue AVD fortement dégonflée	Sortie	select	Radiogroup
27	Roue AVD faiblement dégonflée	Sortie	select	Radiogroup
28	Roue AVD crevée	Sortie	select	Radiogroup
29	Roue ARG normal	Sortie	select	Radiogroup
30	Roue ARG fortement dégonflée	Sortie	select	Radiogroup
31	Roue ARG faiblement dégonflée	Sortie	select	Radiogroup
32	Roue ARG crevée	Sortie	select	Radiogroup
33	Roue ARD normal	Sortie	select	Radiogroup
34	Roue ARD fortement dégonflée	Sortie	select	Radiogroup
35	Roue ARD faiblement dégonflé	Sortie	select	Radiogroup
36	Roue ARD crevée	Sortie	select	Radiogroup
37	Pression pneu AVG	Entrée	data	Textview
38	Pression pneu AVD	Entrée	data	Textview
39	Pression pneu ARG	Entrée	data	Textview
40	Pression pneu ARD	Entrée	data	Textview
41	Eteindre les feux	Sortie	pulse	Button
42	Allumage automatique	Sortie	pulse	Button
43	Appel de phare	Sortie	pulse	Button
44	Feux de position	Sortie	bool	Toggle button
45	Feux de croisement	Sortie	bool	Toggle button
46	Feux de route	Sortie	bool	Toggle button
47	Antibrouillard arrière	Sortie	bool	Toggle button
48	Antibrouillard avant	Sortie	bool	Toggle button
49	Clignotant gauche	Sortie	bool	Toggle button
50	Clignotant droit	Sortie	bool	Toggle button
51	Essuie-glace avant vitesse 2	Sortie	bool	Toggle button
52	Essuie-glace avant vitesse 1	Sortie	bool	Toggle button
53	Essuie-glace position 1	Sortie	bool	Toggle button
54	Essuie-glace position auto	Sortie	bool	Toggle button
55	Lave-glace avant	Sortie	bool	Toggle button
56	Essuie-glace arrière	Sortie	bool	Toggle button
57	Lave-glace arrière	Sortie	bool	Toggle button
58	Commande OBD	Sortie	bool	Toggle button
59	Limiteur de vitesse	Sortie	bool	Toggle button

60	Régulateur de vitesse	Sortie	bool	Toggle button
61	Augmenter vitesse régulateur	Sortie	pulse	Button
62	Réduire vitesse régulateur	Sortie	pulse	Button
63	Etat des fonctions du véhicule	Sortie	pulse	Button
64	Klaxon	Sortie	pulse	Button
65	Mute autoradio	Sortie	bool	Toggle button
66	Monter volume	Sortie	pulse	Button
67	Baisser volume	Sortie	pulse	Button
68	Changer de station	Sortie	pulse	Button
69	Son clignotant 1	Sortie	pulse	Button
70	Son clignotant 2	Sortie	pulse	Button
71	Son oubli clef	Sortie	pulse	Button
72	Fonction libre	Sortie		
73	Fonction libre	Sortie		
74	Fonction libre	Sortie		
75	Fonction libre	Sortie		
76	Fonction libre	Sortie		
77	Fonction libre	Sortie		
78	Fonction libre	Sortie		
79	Fonction libre	Sortie		
80	Fonction libre	Sortie		
81	Fonction libre	Sortie		
82	Fonction libre	Sortie		
83	Fonction libre	Sortie		
84	Fonction libre	Sortie		
85	Fonction libre	Sortie		
86	Fonction libre	Sortie		
87	Fonction libre	Sortie		
88	Fonction libre	Sortie		
89	Fonction libre	Sortie		
90	Fonction libre	Sortie		
91	Mute autoradio	Sortie	bool	Toggle button
92	Monter volume	Sortie	pulse	Button
93	Baisser volume	Sortie	pulse	Button

94	Changer de station	Sortie	pulse	Button
95	Repos	Sortie		
96	Repliement rétroviseur	Sortie	bool	Toggle button
97	Sélection rétroviseur gauche	Sortie	select	Radiogroup
98	Sélection rétroviseur Droit	Sortie	select	Radiogroup
99	Ouverture complète vitre AVG	Sortie	select	Radiogroup
100	Ouverture vitre AVG	Sortie	select	Radiogroup
101	Fermeture complète vitre AVG	Sortie	select	Radiogroup
102	Fermeture vitre AVG	Sortie	select	Radiogroup
103	Ouverture complète vitre ARD	Sortie	select	Radiogroup
104	Ouverture vitre ARD	Sortie	select	Radiogroup
105	Fermeture complète vitre ARD	Sortie	select	Radiogroup
106	Fermeture vitre ARD	Sortie	select	Radiogroup
107	Ouverture complète vitre ARG	Sortie	select	Radiogroup
108	Ouverture vitre ARG	Sortie	select	Radiogroup
109	Fermeture complète vitre ARG	Sortie	select	Radiogroup
110	Fermeture vitre ARG	Sortie	select	Radiogroup
111	Verrouillage vitres	Sortie	bool	Toggle button
112	Menu	Sortie		
113	Mode	Sortie		
114	Reconnaissance vocale	Sortie		
115	ESC	Sortie		
116	OK	Sortie		
117	Téléphone	Sortie		
118	Molette souris	Sortie		
119	Verrouillage portières	Sortie	select	Radiogroup
120	Déverrouillage portières	Sortie	select	Radiogroup
121	Maintien du dialogue	Sortie		
122	Initialisation dialogue	Sortie		
123	Initialisation dialogue	Sortie		

II. FichierSource.xml :

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<dataC4>
	<groupe id="g1" title="ESP et freins">
		<command id="0" type="pulse" label="Repos" />
		<command id="1" type="bool" textOn="Activer ESP"           textOff="Désactiver ESP"/>
		<command id="2" type="bool" textOn="Activer frein à main"  textOff="Désactiver frein à main" />
		<command id="3" type="bool" textOn="Activer freins"        textOff="Désactiver freins" />
	</groupe>
	<groupe id="g2" title="Boite de Vitesse">
		<command id="4" type="bool" textOn="Activer le mode neige"     textOff="Désactiver le mode neige"  />
		<command id="5" type="bool" textOn="Activer le mode sport"     textOff="Désactiver le mode sport" />
		<command id="6" type="bool" textOn="Activer levier de vitesse" textOff="Désactiver levier de vitesse" />
		<choix id="sg21" title="Boite Automatique">
			<command id="7"  type="state" label="Activer position parking" />
			<command id="8"  type="state" label="Activer position neutre" />
			<command id="9"  type="state" label="Activer position arrière" />
			<command id="10" type="state" label="Activer position boite auto" />
		</choix>
		<!--
		<choix id="sg22" title="Boite Manuelle" >
			<command id="11" type="state" label="Activer position 1ère vitesse manuelle" />
			<command id="12" type="state" label="Activer position 2nde vitesse manuelle" />
			<command id="13" type="state" label="Activer position 3ème vitesse manuelle" />
			<command id="14" type="state" label="Activer position 4ème vitesse manuelle" />
			<command id="15" type="state" label="Activer position 5ème vitesse manuelle" />
			<command id="16" type="state" label="Activer position 6ème vitesse manuelle" />
		</choix>
		 -->
	</groupe>
	<!--
	<groupe id="g3" title="Angle volant">
		<command id="17" type="data" label="angle du volant" value="0"/>
	</groupe>
	 -->
	<groupe id="g4" title="Vitesse">
		<command id="18" type="data" label="Vitesse" />
		<!--
```

```
	<command id="19" type="data" label="Distance roue" />
	<command id="20" type="data" label="Accélération" />
	 -->
</groupe>
<groupe id="g5" title="Etat des roues">
	<choix id="sg23" title="Roue Avant Gauche">
		<command id="21" type="state" label="Pression normale" />
		<command id="22" type="state" label="Fortement dégonflée" />
		<command id="23" type="state" label="Faiblement dégonflée" />
		<command id="24" type="state" label="Crevée" />
	</choix>
	<choix id="sg23" title="Roue Avant Droite">
		<command id="25" type="state" label="Pression normale" />
		<command id="26" type="state" label="Fortement dégonflée" />
		<command id="27" type="state" label="Faiblement dégonflée" />
		<command id="28" type="state" label="Crevée" />
	</choix>
	<choix id="sg24" title="Roue Arriere Gauche">
		<command id="29" type="state" label="Pression normale" />
		<command id="30" type="state" label="Fortement dégonflée" />
		<command id="31" type="state" label="Faiblement dégonflée" />
		<command id="32" type="state" label="Crevée" />
	</choix>
	<choix id="sg24" title="Roue Arriere Droite">
		<command id="33" type="state" label="Pression normale" />
		<command id="34" type="state" label="Fortement dégonflée" />
		<command id="35" type="state" label="Faiblement dégonflée" />
		<command id="36" type="state" label="Crevée" />
	</choix>
</groupe>
<groupe id="g6" title="Pression des pneus">
	<command id="37" type="dataRead" label="Pression Avant Gauche" />
	<command id="38" type="dataRead" label="Pression Avant Droit"   />
	<command id="39" type="dataRead" label="Pression Arrière Gauche" />
	<command id="40" type="dataRead" label="Pression Arrière Droit" />
</groupe>
<groupe id="g7" title="Feux et clignotants">
	<command id="41" type="pulse" label="Eteindre" />
```

```
            <command id="42" type="pulse" label="Allumage Automatique" />
            <command id="43" type="pulse" label="Appel de phares" />
            <subgroupe id="sg44" title="Feux">
                  <command id="44" type="bool" textOn="Activer feux de position" textOff="Désactiver feux de position" />
                  <command id="45" type="bool" textOn="Activer feux de croisement" textOff="Désactiver feux de croisement" />
                  <command id="46" type="bool" textOn="Activer feux de route" textOff="Désactiver feux de route" />
                  <command id="47" type="bool" textOn="Activer Antibrouillard arrière" textOff="Désactiver Antibrouillard
arrière" />
                  <command id="48" type="bool" textOn="Activer Antibrouillard avant" textOff="Désactiver Antibrouillard avant"
/>
            </subgroupe>
            <subgroupe id="sg49" title="Clignotants">
                  <command id="49" type="bool" textOn="Activer Clignotant droit" textOff="Désactiver Clignotant droit" />
                  <command id="50" type="bool" textOn="Activer Clignotant gauche"  textOff="Désactiver Clignotant gauche" />
            </subgroupe>
      </groupe>
      <groupe id="g8" title="Essuie-glaces et lave-glaces">
            <subgroupe id="sg51" title="Avant">
                  <command id="51" type="bool" textOn="Activer essuie-glace avant vitesse 2"   textOff="Désactiver essuie-
glace avant vitesse 2" />
                  <command id="52" type="bool" textOn="Activer essuie-glace avant vitesse 1"   textOff="Désactiver essuie-
glace avant vitesse 1" />
                  <command id="53" type="bool" textOn="Activer essuie-glace avant position 1" textOff="Désactiver essuie-glace
avant position 1" />
                  <command id="54" type="bool" textOn="Activer essuie-glace avant position auto" textOff="Désactiver essuie-
glace avant position auto" />
                  <command id="55" type="bool" textOn="Activer lave-glace avant"  textOff="Désactiver lave-glace avant" />
            </subgroupe>
            <subgroupe id="sg57" title="Arrière">
                  <command id="56" type="bool" textOn="Activer essuie-glace arrière" textOff="Désactiver essuie-glace arrière"
/>
                  <command id="57" type="bool" textOn="Activer lave-glace arrière" textOff="Désactiver lave-glace arrière" />
            </subgroupe>
            <command id="58" type="bool" textOn="Activer commande ODB" textOff="Désactiver commande ODB" />
      </groupe>
      <groupe id="g9" title="Commandes au volant">
            <!--
            <command id="59" type="bool" textOn="Activer limiteur de vitesse" textOff="Désactiver limiteur de vitesse" />
```

```
            <command id="60" type="bool" textOn="Activer regulateur de vitesse" textOff="Désactiver regulateur de vitesse" />
             -->
            <command id="61" type="pulse" label="Augmenter vitesse régulateur" />
            <command id="62" type="pulse" label="Réduire vitesse régulateur" />
            <!--
            <command id="63" type="pulse" label="Etat des fonctions du vehicule" />
             -->
            <command id="64" type="pulse" label="Avertisseur sonore" />
      </groupe>
      <groupe id="g10" title="Autoradio">
            <command id="65" type="bool" textOn="Couper volume" textOff="Restaurer volume" />
            <command id="66" type="pulse" label="Monter volume" />
            <command id="67" type="pulse" label="Baisser volume" />
            <command id="68" type="pulse" label="Changer station" />
      </groupe>
      <groupe id="g11" title="Sons">
            <command id="69" type="bool" textOn="Activer son Clignotant 1" textOff="Désactiver son Clignotant 1" />
            <command id="70" type="bool" textOn="Activer son Clignotant 2" textOff="Désactiver son Clignotant 2" />
            <command id="71" type="bool" textOn="Activer son oubli clef" textOff="Désactiver son oubli clef" />
      </groupe>
      <groupe id="g12" title="Vitres et Portières">
            <command id = "96" type ="bool" textOn = "Replier les rétroviseurs" textOff = "Déployer les rétroviseurs"
value="0" />
         <choix id="sg121" title="Rétroviseurs">
           <command id = "97" type ="state" label = "Gauche"/>
           <command id = "98" type ="state" label = "Droit" />
         </choix>
         <choix id="sg122" title="Vitre avant gauche">
             <command id = "99"  type ="state" label = "Ouverture complète"/>
             <command id = "100" type ="state" label = "Descendre" />
             <command id = "101" type ="state" label = "Fermeture complète" />
             <command id = "102" type ="state" label = "Monter" />
         </choix>
         <choix id="sg123" title="Vitre arrière droite">
           <command id = "103" type ="state" label = "Ouverture complète"/>
           <command id = "104" type ="state" label = "Descendre" />
           <command id = "105" type ="state" label = "Fermeture complète" />
           <command id = "106" type ="state" label = "Monter" />
```

```
        </choix>
        <choix id="sg124" title="Vitre arrière gauche">
          <command id = "107" type ="state" label = "Ouverture complète"/>
          <command id = "108" type ="state" label = "Descendre" />
          <command id = "109" type ="state" label = "Fermeture complète" />
          <command id = "110" type ="state" label = "Monter" />
        </choix>
        <command id = "111" type ="bool" textOn ="Verrouiller les vitres" textOff ="Déverrouiller les vitres" />
        <choix id="sg124" title="Portières">
          <command id = "119" type ="state" label = "Verrouiller"/>
          <command id = "120" type ="state" label = "Déverrouiller" />
        </choix>

      </groupe>
</dataC4>
```

III. Parseur Xml dans d'Activity :

```
/* Gestion de l'interface graphique */
    public class IHM {
       private ParserXmlData    parserXML;

       public IHM( Context context) {
              parserXML = new ParserXmlData();
              parserXML.parse( context);
              if (bltCommand.sendStream != null) this.initializeData();
       }

       public void initializeData() {
              /* TODO : lire l'état des commandes du véhicule pour initialiser correctement chaque composant graphique */
              bltCommand.sendCommandBool(37, true);
              /*TODO Affectation des pression au penu correspondant*/

       }

    }

/* ----------------------------------------------------------------------------*/
/* Définition des différentes classes utilisées pour la représentation graphique */
/* ----------------------------------------------------------------------------*/

       /* Gestion des composants graphiques de type bouton on/off */
       public class IHMBtnOnOff extends ToggleButton {

              public IHMBtnOnOff( Context context, String id, String textOn, String textOff, boolean etat)
              {
                     super( context /*, null, R.style.button */);
                     this.setId( Integer.parseInt( id));
                     this.setTextOn( textOff);
                     this.setTextOff( textOn);
                     this.setChecked( etat);
                     this.setOnClickListener( IHMBtnOnOffListener); // surcharge du listener pour emettre la commande Bluetooth
              }
```

```
}

   private OnClickListener IHMBtnOnOffListener = new OnClickListener() {
          IHMBtnOnOff IBOO;
          public void onClick(View v) {
                IBOO = (IHMBtnOnOff)v;
                IBOO.setChecked( IBOO.isChecked());
                bltCommand.sendCommandBool( IBOO.getId(), IBOO.isChecked());
       }
   };

   /* Gestion des composants graphiques de type bouton  */
   public class IHMBtn extends Button {

          public IHMBtn( Context context, String id, String label)
          {
                super( context);
                this.setId( Integer.parseInt( id));
                this.setText( label);
                this.setOnClickListener( IHMBtnListener); // surcharge du listener pour emettre la commande Bluetooth
          }
}

   private OnClickListener IHMBtnListener = new OnClickListener() {
          IHMBtn IBtn;
          public void onClick(View v) {
                IBtn = (IHMBtn)v;
                bltCommand.sendCommandBool( IBtn.getId(), true);
       }
   };


   /* Gestion des composants graphiques de type groupe de sélection */
   public class IHMSelectChoice extends RadioGroup {
          TextView grpTitle;

          public IHMSelectChoice( Context context, String title)
          {
```

```
                super( context);
                grpTitle = new TextView( context);
                grpTitle.setText( title);
                this.addView(grpTitle);
            }

            public void addItem( String id, String label)
            {
                RadioButton rdButton = new IHMSelectItem( this.getContext(), id);
                rdButton.setText( label);
                rdButton.setButtonDrawable(null);
                //this.setBackgroundResource(R.drawable.radio_button_selector);
                rdButton.setOnClickListener( IHMSelectChoiceListener); // surcharge du listener pour emettre la commande
Bluetooth
                this.addView( rdButton);
            }
    }

      public class IHMSelectItem extends RadioButton {
            private int id;
            private IHMSelectItem( Context context, String id)
            {
                  super( context);
                  this.id = Integer.parseInt( id);
            }
      }

      private OnClickListener IHMSelectChoiceListener = new OnClickListener() {
            IHMSelectItem Rb;
            public void onClick(View v) {
                  Rb = (IHMSelectItem)v;
                  bltCommand.sendCommandBool( Rb.id, true);
          }
      };


      /* Gestion des composants graphiques de type saisie de données */
      public class IHMEditText extends LinearLayout {
```

```
        private TextView textView;
        private EditText editText;
        private Button   button;

        public IHMEditText( Context context, String id, String label, String value) {
              super( context);
              this.setId( Integer.parseInt( id));
              this.editText = new EditText( context);
              this.textView = new TextView( context);
              this.button   = new Button( context);

              this.editText.setId( Integer.parseInt( id));
              this.editText.setInputType( 2); // number
              this.editText.setWidth(100);

              this.button.setWidth(70);
              this.button.setText("OK");
              this.button.setId( Integer.parseInt(id) + 500);
              this.button.setOnClickListener( IHMEditTextListener);

              this.textView.setText( label);

              this.setValue( value);
              this.addView( this.editText);
              this.addView( this.button);
              this.addView( this.textView);
        }

        public void setValue( String value) {
              this.editText.setText( value);
        }

        public String getValue() {
              return this.editText.getText().toString();
        }
}

   private OnClickListener IHMEditTextListener = new OnClickListener() {
```

```
            IHMEditText IEt;
            public void onClick(View v) {
                  IEt = (IHMEditText)findViewById( v.getId() - 500);
                  bltCommand.sendCommandData( IEt.getId(), IEt.getValue());
            }
      };


      /* Gestion des composants graphiques de type affichage de données */
      public class IHMTextView extends TextView {
            private String label, value;

            public IHMTextView(Context context, String id, String label, String value) {
                  super( context);
                  this.label = label;
                  this.setId( Integer.parseInt( id));
                  this.setValue( value);
            }

            public String getValue() {
                  return this.value;
            }

            public void setValue( String value) {
                  if ( value == null) value = "--";
                  this.value = value;
                  this.setText( this.label + " : " + this.value);
            }

}


      /* ------------------------------------------------ */
      /* parsing XML du descripteur de données dataC4     */
      /* -> tranformer le XML en représentation graphique */
      /* ------------------------------------------------ */

      public class ParserXmlData {
```

```
        Context       context;
        XmlPullParser parseur;
        InputStream   fichier;
        private TabHost         tabHost;
        private TabHost.TabSpec tabSpec;
        private TabWidget       tabWidget;
        private FrameLayout     frameLayout;
        private ScrollView        scrollView;
    private HashMap<String,ScrollView> tabScroll;
        private LinearLayout    layout;
        private TextView        txtView, txtGroup;
        private String          id, type, title, label;
        private Boolean         etat;
        IHMSelectChoice ihmSelectChoice;
        IHMBtnOnOff     ihmBool, ihmBoolR;
        IHMBtn          ihmBtn;
        IHMEditText     ihmData;
        IHMTextView     ihmDataR;

        public ParserXmlData( ) {
        }

        public void parse( Context context) {

         try {
              this.context = context;
                    XmlPullParserFactory factory = XmlPullParserFactory.newInstance();
               factory.setNamespaceAware(true);
              parseur = factory.newPullParser();
              fichier = getResources().openRawResource( R.raw.datac4);
              parseur.setInput( fichier, "iso-8859-1");

              int eventType = parseur.getEventType();
             while (eventType != XmlPullParser.END_DOCUMENT) {
              if( eventType == XmlPullParser.START_DOCUMENT) {
                    startDocument();
              } else if( eventType == XmlPullParser.START_TAG) {
                    startElement( parseur.getName());
```

```
                } else if( eventType == XmlPullParser.END_TAG) {
                      endElement( parseur.getName());
                } else if( eventType == XmlPullParser.TEXT) {
                      // rien
                }
                eventType = parseur.next();
               }
            } catch ( XmlPullParserException e) {
                e.printStackTrace();
            } catch ( IOException e) {
                e.printStackTrace();
            }
          }

          private void startDocument() {
                tabScroll = new HashMap<String, ScrollView>();
          }

          private void startElement( String qName) {
          if ( qName.equalsIgnoreCase("datac4")) {
                buildTabHost();
          } else if ( qName.equalsIgnoreCase("groupe")) {
                buidlTabSpec();
          } else if ( qName.equalsIgnoreCase("choix")) {
                buidlSelectChoice();
          } else if ( qName.equalsIgnoreCase("subgroupe")) {
                txtGroup = new TextView( context);
                txtGroup.setText( parseur.getAttributeValue( null, "title"));
                layout.addView( txtGroup);
          } else if ( qName.equalsIgnoreCase("command")) {
                id     = parseur.getAttributeValue( null, "id");
                type   = parseur.getAttributeValue( null, "type");
                label  = parseur.getAttributeValue( null, "label");
                if ( type.equals( "bool")) {
                      etat    = Boolean.valueOf( parseur.getAttributeValue( null, "etat"));
                      ihmBool = new IHMBtnOnOff( context, id, parseur.getAttributeValue( null, "textOn"),
parseur.getAttributeValue( null, "textOff"), etat);
                      layout.addView( this.ihmBool);
```

```
                } else if ( type.equalsIgnoreCase( "data")) {
                        ihmData = new IHMEditText( context, id, label, parseur.getAttributeValue( null, "value"));
                        layout.addView( this.ihmData);
                } else if ( type.equalsIgnoreCase( "dataread")) {
                        ihmDataR = new IHMTextView( context, id, label, parseur.getAttributeValue( null, "value"));
                        layout.addView( this.ihmDataR);
                } else if ( type.equalsIgnoreCase( "boolread")) {
                        etat     = Boolean.valueOf( parseur.getAttributeValue( null, "etat"));
                        ihmBoolR = new IHMBtnOnOff( context, id, label, "", etat);
                        ihmBoolR.setEnabled( false);
                        layout.addView( ihmBoolR);
                } else if ( type.equalsIgnoreCase( "state")) {
                ihmSelectChoice.setEnabled( false);
                        ihmSelectChoice.addItem( id, label);
                } else if ( type.equalsIgnoreCase( "pulse")) {
                        ihmBtn = new IHMBtn( context, id, label);
                        layout.addView( this.ihmBtn);
                }
        }

        }

        private void endElement( String qName) {
        if ( qName.equals("dataC4")) {
                setContentView( tabHost);
        } else if ( qName.equalsIgnoreCase("groupe")) {
                tabSpec.setContent( new TabHost.TabContentFactory() {
                public View createTabContent(String tag)
                {
                        return tabScroll.get(tag);                    }
          });
                tabHost.addTab( tabSpec);
        } else if ( qName.equalsIgnoreCase("choix")) {
                layout.addView( ihmSelectChoice);
        }
        }

    private void buildTabHost()
```

```
	{
			tabHost = new TabHost( context);
			tabHost.setLayoutParams( new LinearLayout.LayoutParams( LinearLayout.LayoutParams.FILL_PARENT,
LinearLayout.LayoutParams.FILL_PARENT));
			tabWidget = new TabWidget( context);
			tabWidget.setVerticalGravity( Gravity.TOP);
			tabWidget.setId(android.R.id.tabs);
			tabHost.addView(tabWidget, new LinearLayout.LayoutParams( LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT));
			frameLayout = new FrameLayout( context);
			frameLayout.setId(android.R.id.tabcontent);
			frameLayout.setPadding(0, 100, 0, 0);
		tabHost.addView(frameLayout, new LinearLayout.LayoutParams( LinearLayout.LayoutParams.FILL_PARENT,
LinearLayout.LayoutParams.WRAP_CONTENT));
		tabHost.setup();
	}

	private void buidlTabSpec() {
		id	= parseur.getAttributeValue( null, "id");
		title	= parseur.getAttributeValue( null, "title");

		txtView = new TextView( context);
		txtView.setTextAppearance( context, R.style.groupeTitle);
		txtView.setText( this.title);

		layout = new LinearLayout( context);
		layout.setOrientation(LinearLayout.VERTICAL);
		layout.setLayoutParams( new LinearLayout.LayoutParams( LinearLayout.LayoutParams.FILL_PARENT,
LinearLayout.LayoutParams.FILL_PARENT));
		layout.addView( this.txtView);

		tabScroll.put(id, new ScrollView( context));
		scrollView = tabScroll.get( id);
		scrollView.setLayoutParams( new LinearLayout.LayoutParams( LinearLayout.LayoutParams.FILL_PARENT,
LinearLayout.LayoutParams.FILL_PARENT));
		scrollView.addView( this.layout);

		tabSpec = tabHost.newTabSpec( id);
```

```
                int idIcon = getResources().getIdentifier( id , "drawable", "calmos.app.namespace");
                this.tabSpec.setIndicator( "", getResources().getDrawable( idIcon));
        }

        private void buidlSelectChoice() {
                ihmSelectChoice = new IHMSelectChoice( context, parseur.getAttributeValue( null, "title"));
        }

        }
}
```

IV. AndroidManifest.xml :

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

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

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
         >
        <activity
            android:theme="@style/Theme.Custom"
            android:name=".CalmosActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

V. Bluetooth et constructeur de trames :

```
/* ---------------------------------------------------------------------------- */
/* Classe qui gère la construction et l'envoi / réception de trames en Bluetooth */
/* ---------------------------------------------------------------------------- */

    public class BlueToothCommand {
            private BluetoothAdapter blueAdapter;
            private BluetoothDevice  blueDevice;
            private BluetoothSocket  blueSocket;
            private String trame;

            private UUID         Apuuid         = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
        private String       DEVICE_NAME    = "Free2move WU"; // nom du périphérique Bluetooth C4
        private String       DEVICE_ADRESSE = "00:0B:CE:04:09:DA"; //adresse MAC C4
            private InputStream  receiveStream; // Canal de réception
          private OutputStream sendStream;    // Canal d'émission

            public BlueToothCommand( Context context)
            {
                  try {
                        activate();
                        if ( blueAdapter != null) {
                              searchDeviceC4();
                              if ( blueSocket != null) {
                                    connectToC4();
                                    initCommunication();
                              }
                        }
                  } catch (Exception e) {
                        alertMessage( "Erreur grave: " + e.getMessage());
                        e.printStackTrace();
                  }
            }

          private void activate ()
          {
            //Vérification de la présence bluetooth
```

```
   blueAdapter = BluetoothAdapter.getDefaultAdapter();
         if ( blueAdapter != null)  {
               if ( !blueAdapter.isEnabled())
                     blueAdapter.enable();
               else
                     alertMessage( "Equipement Bluetooth non disponible.");
         } else
               alertMessage( "Erreur tentative usage Bluetooth.");
}

private void searchDeviceC4()
{
   try {
         blueDevice = blueAdapter.getRemoteDevice( DEVICE_ADRESSE);
   blueAdapter.cancelDiscovery();
         if ( blueDevice != null) {
               blueSocket = blueDevice.createRfcommSocketToServiceRecord( Apuuid);
               alertMessage( "Connexion avec C4 réussie.");
      }
   } catch ( IOException closeException) {
         alertMessage( "Echec recherche périphérique C4.");
   }
}

private void connectToC4()
{
   try {
         blueSocket.connect();
   } catch (IOException connectException) {
               try {
                     blueSocket.close();
               } catch (IOException closeException) {
                     alertMessage( "Echec fermeture connexion.");
               }
               alertMessage( "Echec connexion au périphérique C4");
   }
}
```

```
private void initCommunication()
{
   try {
          receiveStream = blueSocket.getInputStream();
          sendStream    = blueSocket.getOutputStream();
          modeDebug     = false; // La comunication est établie, pas besoin de mode trace
   } catch(IOException e){
          alertMessage( "Echec initialisation communication C4.");
   }

}

   public void sendCommandBool( int id, boolean etat)
   {
          this.trame = this.buildTrameBool(id, etat);
          this.sendTrame( this.trame);
   }

   public void sendCommandData( int id, String value)
   {
          this.trame = this.buildTrameData(id, Integer.parseInt(value));
          this.sendTrame( this.trame);
   }

   public String receiveCommandData( int id)
   {
          this.trame = this.buildTrameData(id, 0);
          return this.getTrame();
   }

   private String buildTrameBool( int id, boolean etat)
   {
          int oct_bit_act;
          if ( etat == true)
                oct_bit_act = 1;
          else
                oct_bit_act = 0;
```

```
            String data2 = Character.toString((char) 0);
            String data3 = Character.toString((char) 0);
            String data4 = Character.toString((char) oct_bit_act);
            String data5 = Character.toString((char) 0); //séparateur
            String data6 = Character.toString((char) id);

            this.trame = data2 + data3 + data4 + data5 + data6;
            if ( modeDebug) alertMessage( "Trame bool : " + trame) ;
            return this.trame;
     }

    private String buildTrameData( int id, int value)
    {
      String chaineTot;
       int    oct_val1, oct_val2;

      if (id==18) value = (value - 1) * 100; // commande vitesse

      // Mise au format des données
      chaineTot = String.format("%04X",value);
      oct_val1  = Integer.parseInt( chaineTot.substring(2),   16);
      oct_val2  = Integer.parseInt( chaineTot.substring(0,2), 16);

      // Transcription en caractères ASCII pour concaténation
      String data2 = Character.toString((char) oct_val1);
      String data3 = Character.toString((char) oct_val2);
      String data4 = Character.toString((char) 1);  //act
      String data5 = Character.toString((char) 0);  //séparateur
      String data6 = Character.toString((char) id);

      trame = data2 + data3 + data4 + data5 + data6;
      if ( modeDebug) alertMessage( "Trame data : " + trame);
      return trame;
    }

       public void sendTrame( String trame) {
      byte[] writeBuf = trame.getBytes();
      if ( sendStream == null) return;
```

```
        try {
         sendStream.write(255);
         sendStream.write(writeBuf);
        } catch (IOException closeException) {
         alertMessage( "Echec envoi trame de données.");
        }
         }

         public String getTrame() {
        int nbCar;
        int i;
        byte readBuf[] = new byte[24];
        byte rawdata[] = new byte[6];
        String data = null;

        if ( receiveStream != null) {
         try {
                nbCar = receiveStream.read( readBuf,0,24);
                if ( nbCar > 0) {
                      for ( i = 0; i == nbCar; i++ )
                            rawdata [i] = readBuf[i];
                      data = new String(rawdata);
                }

         } catch ( IOException e) {
                alertMessage("Echec réception message.");
         }
        }
        alertMessage( data);
        nbCar = 0;
        return data;
         }
   }


  public class ReadPresPneu {
     String[] tabPresPneu = new String[4];
     String dataReceiv = bltCommand.getTrame();
```

```
    public String[] ChoiTraitData(){
          if( Integer.getInteger( dataReceiv.substring(5,6) ) == 37)
                tabPresPneu = checkPresPneu(dataReceiv);
          return tabPresPneu;
    }

    public String[] checkPresPneu( String data) {

         tabPresPneu[0] = analyPresPneu(  data.substring(  0,  6));
         tabPresPneu[1] = analyPresPneu(  data.substring(  6, 12));
         tabPresPneu[2] = analyPresPneu(  data.substring( 12, 18));
         tabPresPneu[3] = analyPresPneu(  data.substring( 18, 24));

         return tabPresPneu;
     }

     private String analyPresPneu( String tramePneu){
         DecimalFormat df = new DecimalFormat( "#,##");
          float dataPneu = 0;

         if ( Integer.getInteger( tramePneu.substring( 0, 1)) == 255 )
            dataPneu = (Integer.getInteger(tramePneu.substring(1,2))) / 10;
         return df.format( dataPneu);
     }
} }
```

VI. Librairie à importer et création d'Activity :

```
package calmos.app.namespace;
import android.app.Activity;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserException;
import org.xmlpull.v1.XmlPullParserFactory;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.IOException;
import android.content.Context;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.FrameLayout;
import android.widget.RadioGroup;
import android.widget.RadioButton;
import android.widget.ScrollView;
import android.widget.TabWidget;
import android.widget.TextView;
import android.widget.TabHost;
import android.widget.ToggleButton;
import android.widget.Toast;
import android.os.Bundle;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import java.text.DecimalFormat;
import java.util.HashMap;
import java.util.UUID;


public class CalmosActivity extends Activity {
    /** Called when the activity is first created. */
      IHM             ihmElement;
```

```
   BlueToothCommand bltCommand;
boolean          modeDebug = true;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
         try {
               bltCommand  = new BlueToothCommand( this);
               if (bltCommand.blueSocket!=null)
                     ihmElement  = new IHM( this);

         } catch (Exception e) {
               alertMessage( "Erreur grave : " + e.toString());
               e.printStackTrace();
         }
}

public void alertMessage( String txtMessage) {
         Toast.makeText( this, txtMessage, Toast.LENGTH_SHORT).show();
  }
```