Michael MUELLER, Cyrille SAVY

Octobre 2013 à Janvier 2014

Résumé

Ce projet consiste à développer une application mobile sur la plate-forme Android qui peut communiquer avec une carte embarquée via la technologie Bluetooth. La carte embarquée est une petite carte de développement qui fonctionne avec un microcontrôleur. L’application peut se connecter à la carte et ensuite allumer ou éteindre des LEDs, et visualiser l’état de certains éléments, tels que des boutons poussoirs et un potentiomètre.

Bluetooth sur Android

Rapport de conception



Summary

[2. Introduction 2](#_Toc378350826)

[3. Cahier des charges 3](#_Toc378350827)

[3.1. Buts principaux du projet 3](#_Toc378350828)

[3.2. Tâches à effectuer 3](#_Toc378350829)

[4. Généralités 4](#_Toc378350830)

[5. Application Android 5](#_Toc378350831)

[5.1. Architecture de l’application 5](#_Toc378350832)

[5.2. Interface graphique 6](#_Toc378350833)

[6. Système embarqué 7](#_Toc378350834)

[6.1. Matériel 7](#_Toc378350835)

[6.2. Architecture de l’application 7](#_Toc378350836)

[7. ProtocoleS de communication 9](#_Toc378350837)

[7.1. Protocole Bluetooth 9](#_Toc378350838)

[7.2. Protocole de l’application 9](#_Toc378350839)

[8. Conclusion 10](#_Toc378350840)

[9. Bibliographie 11](#_Toc378350841)

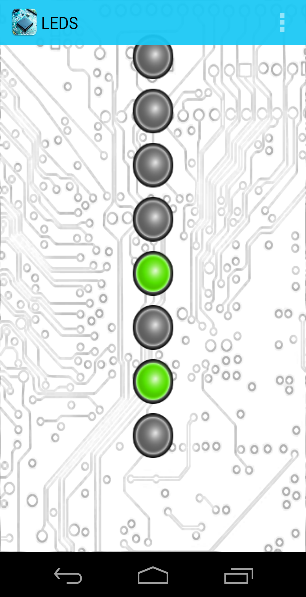
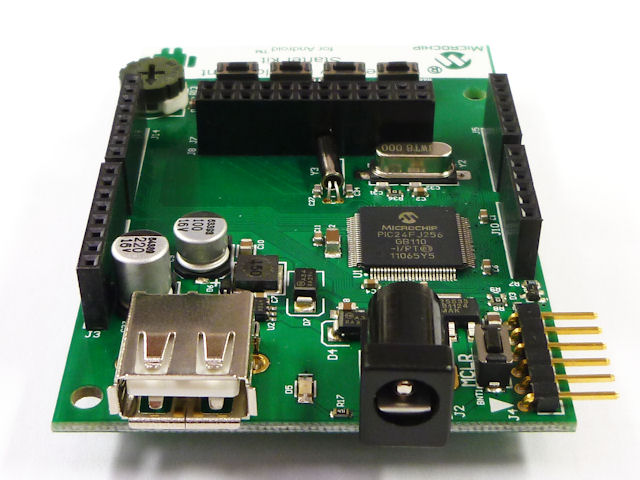
[10. Annexes 12](#_Toc378350842)

# Introduction

Le projet s’est déroulé dans le cadre du cours « Développement mobile » en troisième année de la filière informatique de la Haute École Arc. Il consistait à développer une application mobile sur la plate-forme Android en utilisant au moins un capteur disponible sur l’appareil utilisé.

Le sujet du projet était laissé à choix, avec la contrainte d’utiliser un capteur du côté appareil mobile. Le sujet que nous avons donc choisi pour ce projet consiste à utiliser le module Bluetooth disponible sur les appareils mobiles pour communiquer avec un système embarqué fonctionnant avec un microcontrôleur afin de lui envoyer des paramètres et de visualiser son état.

L’application peut se connecter et se déconnecter de la carte, et si elle est connectée elle peut allumer et éteindre les LEDs, visualiser l’état des boutons poussoirs et le niveau du potentiomètre.



LEDs + LCD

Pot + Boutons

Figure 1 : principe de fonctionnement

L’utilité du projet n’est pas grande en soi (allumer des LEDs, voir l’état d’un potentiomètre, …), mais la finalité visée est surtout de montrer que l’on peut paramétrer et consulter facilement un système embarqué à microcontrôleur à l’aide d’un appareil mobile.

# Cahier des charges

Afin de réaliser le projet que nous avons choisi, voici les objectifs, contraintes et tâches à effectuer que nous nous sommes fixés.

## Buts principaux du projet

Voici les différents objectifs en début de projet :

* Adapter la partie embarquée pour communiquer par Bluetooth.
* Adapter l’application Android pour la communication par Bluetooth.
* Développer une interface graphique pour visualiser les différents paramètres sur Android.
* Lecture/écriture des différents capteurs et actionneurs de la carte embarquée
* Affichage d’un texte sur un petit écran LCD (optionnel : si le temps le permet)

## Tâches à effectuer

Voici la liste des tâches qui ont été effectuées durant le projet :

* Comprendre l’existant et poser les spécifications
* Etude du protocole de communication Bluetooth au niveau embarqué
* Etude du contrôle du « Dongle » USB-Bluetooth
* Etude du module Bluetooth au niveau mobile
* Implémentation de la couche de communication Bluetooth sur la carte embarquée
* Implémentation de la couche de communication Bluetooth sur l’appareil mobile
* Lecture/écriture des différents capteurs et actionneurs de la carte au niveau du PIC
* Adaptation de l’application Android afin de communiquer par Bluetooth et utilisation des différents composants de la carte de manière dynamique
* Maquettage et implémentation de l’interface graphique utilisateur

# Généralités

La carte embarquée que nous avons choisie, sur une proposition de Mme Pazos, est la carte « Android Accessory Development Board » développée par Microchip, et qui fonctionne avec un microcontrôleur PIC24F. Les composants utilisés sur la carte sont les huit LEDs, les quatre boutons poussoirs et le potentiomètre. Nous utilisons également l’interface USB, dans laquelle nous avons branché petit adaptateur USB-Bluetooth. L’écran LCD ajouté a été interfacé avec un circuit de développement « Arduino », afin qu’il puisse être contrôlé par une communication série RS232.

Le dongle USB-Bluetooth cité plus haut ( De la marque « Trust ») fonctionne avec le blutooth Version 4, qui est une version basse consommation.

L’application « Android » a été développée pour fonctionner avec l’API 15 au minimum (Android 4.0.3).



Figure 2 : montage final, avec LCD

# Application Android

L’application est basée sur une activité principale qui gère les données échangées, un thread séparé qui s’occupe de l’envoi et de la réception par Bluetooth, et l’utilisation de fragments pour l’interface utilisateur.

## Architecture de l’application

L’architecture de l’application est faite sur différents niveaux d’abstraction du matériel utilisé, afin de facilité la gestion des données et améliorer la modularité des différentes parties du code source.



Figure 3 : Architecture de l’application mobile

Tout en bas il y a un thread séparé de l’application qui s’occupe uniquement de la transmission et de la réception Bluetooth. Son rôle est d’envoyer sur le Bluetooth les données qui sont contenues dans un buffer d’envoi, et aussi de remplir un buffer de réception avec les données reçues sur le Bluetooth.

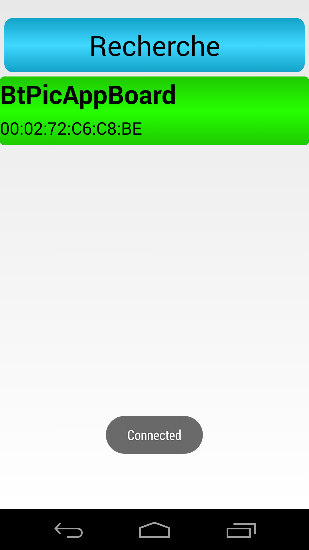
Ensuite il y a au-dessus une classe gérant le protocole de communication que nous avons implémenté. Elle décode les trames reçues sur le Bluetooth et encode les commandes envoyées par la couche supérieure.

La couche supérieure permet donc d’accéder à des méthodes d’envoi de commandes, qui sont appelées avec un « listener » lorsque l’on appuie par exemple sur une LED pour l’allumer. Pour ce qui est de la réception, les composants ont des méthodes permettant de leur fixer une valeur (par exemple fixer la valeur du potentiomètre) qui sont appelées par la classe réception lorsqu’un message est reçu depuis la carte.

Et tout en haut il y a notre application, avec l’interface graphique, qui gère les interactions avec l’utilisateur (si il éteint une LED, quand il écrit du texte dans le champ de l’écran LCD) ainsi que de l’appareil mobile (par exemple un changement d’orientation de l’appareil).

Et bien évidemment, il y a encore au-dessus de notre application, le système d’exploitation Android qui gère le cycle de vie de l’application (ouverture, fermeture, pause, …, avec les méthodes « onCreate », « onDestroy », « onResume », …).

## Interface graphique

L’interface graphique est faite avec des fragments. On peut facilement glisser d’un fragment à un autre, suivant l’action que l’on veut réaliser.

Un premier fragment permet de rechercher les Bluetooth disponibles, de s’y connecter et s’y déconnecter. Le bouton rechercher va lancer un scan de tous les Bluetooth dans la portée de l’appareil mobile. Lorsque la recherche est terminée ou qu’on l’a interrompue en touchant l’écran, on peut se connecter au Bluetooth souhaité simplement en appuyant sur son nom ! Il faut bien évidemment se connecter à la carte PIC (« BluetoothPicAppBoard ») et non à un autre Bluetooth, ce qui ne serait pas approprié… Pour se déconnecter, il suffit d’appuyer à nouveau sur le nom du Bluetooth connecté.

Figure 4 : fragment des connections bluetooth

Une fois que l’on est connecté à la carte on peut commencer à naviguer dans les trois autres fragments, qui sont mis à jour avec les valeurs actuelles de la carte PIC. Le premier d’entre eux permet d’allumer et d’éteindre les LEDs en appuyant celle dont on souhaite changer l’état. Le second fragment permet de visualiser l’état des boutons poussoirs et le niveau du potentiomètre. Le dernier permet quant à lui de configurer le texte qui est affiché sur l’écran LCD connecté à la carte. On peut écrire deux lignes de texte de seize caractères chacune, et il est envoyé lorsque l’on presse sur envoyer. La case à cocher « Backlight » permet d’allumer ou non le rétroéclairage de l’écran LCD.

Figure 7 : fragment du LCD

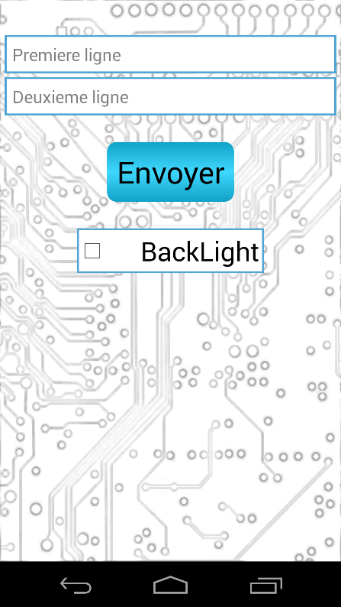
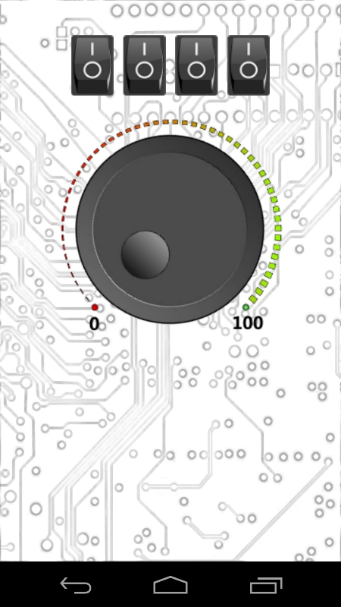
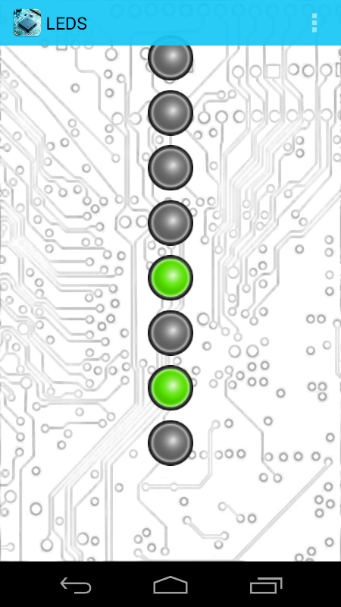


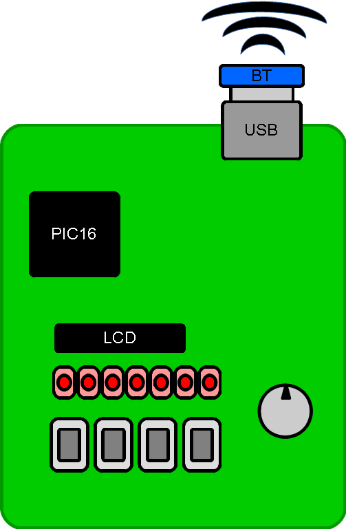
Figure 6 : fragment des boutons et du potentiomètre

Figure 5 : fragment des LEDs

# Système embarqué

L’application du système embarqué fonctionne comme esclave de l’application Android. Elle reçoit des commandes par le Bluetooth et exécute l’action associée à la commande, par exemple : « éteindre la LED 1 ».

## Matériel

La carte embarquée est une carte de développement produite par Microchip dans le but de faire des développements avec Android, mais par USB : l’ « Android Accessory Development Board ».

Elle fonctionne avec un PIC24FJ256, qui est un microcontrôleur 16bits de Microchip. Les composants auxquels le microcontrôleur a accès sont huit LEDs, quatre boutons poussoirs, un potentiomètre et une interface USB. Toutes les autres entrées/sorties ne sont pas utilisées mais elles sont à disposition sur des connecteurs.

Nous avons donc connecté la sortie UART du microcontrôleur sur un kit Arduino qui permet de contrôler facilement un écran LCD. Le texte à afficher sur le LCD est donc envoyé sur cette sortie série.

Figure 8 : materiel sur la carte

La carte n’ayant elle-même pas d’interface Bluetooth, nous avons ajouté un dongle USB-Bluetooth sur son port USB.



Figure 9 : montage avec l’écran LCD

## Architecture de l’application

L’architecture du logiciel embarqué est également basée sur différents niveaux d’abstraction du matériel utilisé, toujours pour rendre le programme modulaire et faciliter la gestion des données.

Il y a tout en bas les stacks USB et bluetooth, qui permettent de contrôler le dongle « USB-Bluetooth » ainsi que les connexions, transmission et réception de données (voir fig. 10). Ces couches basses permettent de voir le Bluetooth comme une communication série standard.

Juste en-dessus, il y a une couche qui gère notre protocole de communication. Elle décode les trames reçues et encode les données à envoyer.

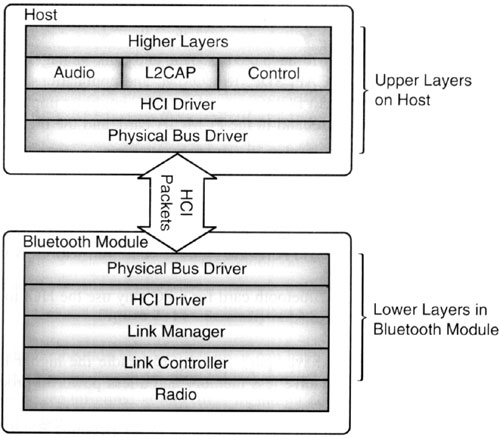
Et pour finir, tout en haut il y a la couche application, qui effectue la bonne action en fonction des messages reçus. Juste après la connexion, elle transmet les valeurs de tous ses paramètres afin que l’application mobile puisse mettre à jour ses données. Après ça, elle transmet la valeur du potentiomètre et des boutons poussoirs uniquement si leur état a changé. Pour les LEDs et l’écran LCD, elle attend de recevoir une commande les concernant pour changer leur valeur.

Figure 10 : couches USB et Bluetooth

La première stack Bluetooth qui nous avait été conseillée n’était malheureusement pas utilisable pour notre architecture. Nous avons finalement utilisé la stack fournie par « Google inc. » (Android Development Kit). Nous n’avons pas pu l’utiliser sans modification, car elle était portée pour un processeur ARM, alors que nous avons un processeur « Microchip » 24Fxxx. Nous n’utilisons que les couches utiles au projet et en voici une liste exhaustive :

* SDP : Donne les informations quant aux services Bluetooth qui sont fournie par la carte embarquée. Dans notre cas, un simple port sériel.
* L2CAP : C’est la couche de transmission des données la plus basse. Voir le chapitre suivant
* RFCOMM : C’est la couche qui fournit le service de port série virtuel. On lit et on écrit directement les données au travers de celui-ci
* HCI : C’est la couche qui gère les événements lancés par le dongle USB et permet de configurer les modalités de communications (par ex. : avec ou sans mots de passe, etc.). Par exemple si un périphérique désire se connecter à la carte.

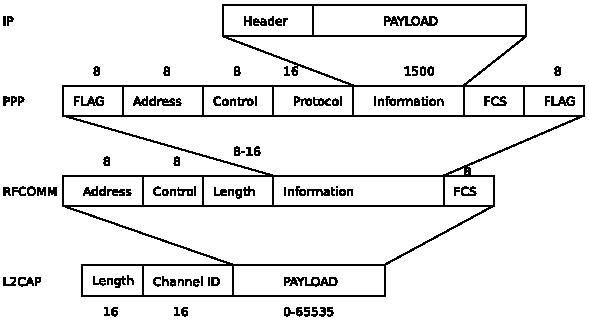
# ProtocoleS de communication

## Protocole Bluetooth

Ce chapitre vous donne rapidement une petite description du fonctionnement du protocole de communication Bluetooth.

La couche « RFCOMM » permet d’empaqueter les informations que nous voulons transmettre au smartphone (cf. chapitre 7.2). Il est composé d’un champ d’adresse, de contrôle, de la longueur des données totale et d’une sorte de CRC. Ce paquet est ensuite transmis à la couche « L2CAP » qui elle empaquette toute la trame « RFCOMM » dans son champ de données (PAYLOAD). Le premier champ de la trame est sa longueur totale, le deuxième est le numéro de canal utilisé par le périphérique Android (Il peut y avoir plus d’un smartphone connecté) et le dernier champ, comme dit précédemment, est la trame « RFCOMM » complète.

Voici ci-dessous un petit diagramme qui illustre ce principe de couche.



## Protocole de l’application

Afin de communiquer de manière simple avec le smartphone, nous avons dû établir un petit protocole de communication. En voici un petit diagramme :

**<CR><LF>**

**Séparateur**

**Paramètre x**

**Séparateur**

**Paramètre 1**

**Séparateur**

**$**

**Commande**

Voici un exemple concret pour allumer la LED numéro 5 et envoi des boutons poussoirs :

**$**

**1**

**\_**

**5 (No. LED)**

**\_**

**1 (Etat)**

**\_**

**<CR><LF>**

**2**

**$**

**\_**

**1(Etat btn 1)**

**\_**

**1(Etat btn 2)**

**\_**

**0(Etat btn 3)**

**1(Etat btn 4)**

**\_**

**<CR><LF>**

Nous utilisons le même principe pour lire et écrire sur l’écran LCD et pour lire la valeur du potentiomètre.

# Conclusion

Malgré les difficultés rencontrées, notamment pour adapter les stacks USB et Bluetooth pour le PIC, initialement prévues pour un processeur ARM, nous avons réussi à atteindre tous les objectifs que nous nous étions fixés en début de projet. L’application est terminée et fonctionnelle, la communication fonctionne sans problème et est suffisamment rapide pour avoir grande réactivité de l’application du changement d’état du potentiomètre et des boutons dans un sens, et des LEDs et du LCD dans l’autre sens.

Les résultats ont vraiment dépassé toutes nos attentes et nous remercions chaleureusement nos enseignantes qui nous ont si bien encadrés pour ce projet, particulièrement Mme. Nuria Pazos.

Ce projet nous a permis d’approfondir et de consolider nos connaissances apprises au cours théorique à propos de la plateforme de développement Android.

# Bibliographie

1. [*http://stackoverflow.com/*](http://stackoverflow.com/)
2. [*http://www.android.com/*](http://www.android.com/)
3. <http://fr.openclassrooms.com/informatique/android/cours>

# Annexes

* Cahier des charges
* Planification et suivi de projet