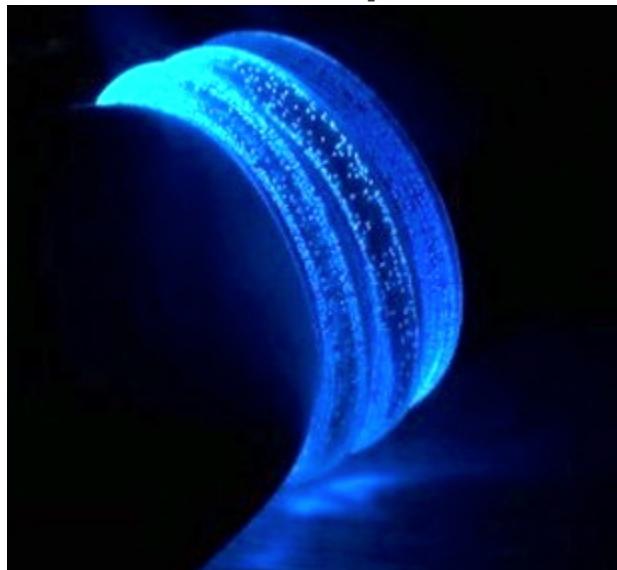


PACT
2013-2014

Rapport d'avancement du groupe "PACT 4.1"
PAN 4 : Cahier des Charges
"The Bumpers"



Date de mise à jour: 01/05/2014

Damien Auffert
Arturo Castellanos Salinas
Cédric Meston
Julien Roméro

Charles Deschard
Marc Faddoul
Félix Richart
Alexandre Tadros

<i>Nom des encadrants:</i>	<i>Fonction:</i>
Jean Le Feuvre	Tuteur
Mathieu Cholet	Encadrant informatique/Expert JAVA
Guillaume Duc	Expert électronique/GPIO
Tarik Ghraba	Expert électronique/GPIO
Jean Claude Dufourd	Expert Androïd
Thomas Robert	Expert Communication
Rémi Maniak	Expert SES

Sommaire:

1.	Résumé du sujet choisi.....	p.3
2.	Summary.....	p.4
3.	Description fonctionnelle du prototype allégé.....	p.5
4.	Architecture informatique du prototype allégé.....	p.10
5.	Planification temporelle.....	p.12
6.	Perspectives pour le prototype final.....	p.12
7.	Résumé de l'état des modules.....	p.16

1. Résumé du sujet choisi:

Le reproche principal que l'on peut formuler à l'égard des réseaux sociaux et de la prolifération des Smartphones est qu'ils détruisent le contact humain de proximité. Un paradoxe quand on pense que ces outils ont réussi à rapprocher des inconnus écartés à des milliers de kilomètres...

L'idée du BumpBand est simple. Il s'agit d'exploiter cette capacité incroyable des réseaux sociaux à rapprocher les inconnus mais à l'échelle d'un évènement festif. Il s'agit, via un bracelet ludique, surfant sur la vague des objets connectés utiles et du quantified self, de transformer le contact humain en composante d'un réseau social local. Ajoutez à cela un côté éphémère et une architecture qui protège efficacement la vie privée des utilisateurs et vous obtenez un objet amusant, pratique et qui peut ajouter une vraie valeur ajoutée à une soirée.

L'utilisateur doit simplement mettre son BumpBand autour du poignet et le synchroniser avec une application Smartphone préalablement téléchargée : il est prêt à l'emploi. Doté d'un petit écran LCD et pouvant être illuminé de la couleur souhaitée par l'utilisateur, on pourra communiquer avec les autres bumpers via un code couleur établi ou simplement le personnaliser à son goût. Deux personnes deviennent des BumpFriends (BF) par simple contact entre leurs bracelets, le Bump. Une fois ce geste effectué, ils auront accès à un certain nombre d'interactions : un service de messagerie qui durera le temps de l'événement et proposer à un BF de le rechercher par clignotement lumineux avec des couleurs spécifiques. De plus, chacun possède un fil d'actualité qui s'affiche sur le Smartphone et qui recense les activités des BF en temps réel.

Le gérant de la soirée peut envoyer un message à tous les Bumpers ainsi que lancer un Flash MoB (Flash Meet other Bumpers). Ce jeu a pour but de rassembler des participants qui ne se connaissent pas forcément. Lorsque le Flash MOB commence, 5 bumpers choisis aléatoirement se voient attribuer une même couleur. Il faut parvenir à bumper les 4 autres personnes avant la fin du jeu afin de gagner un crédit boisson. De cette manière les 5 Bumpers feront connaissance.

En plus de ces fonctionnalités ludiques, le BumpBand simplifie réellement la vie de ses utilisateurs. Avant l'événement, il est possible d'acheter la place via internet et de l'enregistrer sur le bracelet. Le passage par une file prioritaire permet alors d'entrer dans la salle via un bump signalant sa présence à l'entrée. De même plus besoin de ticket pour le vestiaire : un simple bump suffit à enregistrer son numéro. On peut enfin créditer un compte au bar au début de l'événement. On peut ensuite le débiter en prenant des consommations en bumpant au bar. On peut choisir sa boisson sur le Smartphone qui sélectionne automatiquement la couleur correspondante pour le bracelet. Le barman peut ainsi plus facilement savoir quelles boissons préparer et en quelles quantités. Finalement l'utilisateur peut suivre sa consommation approximative d'alcool et utiliser ses nouvelles relations pour partager un taxi pour rentrer, histoire de rentrer en sécurité.

2. Summary:

One of the main forms of criticism that can be directed at the overwhelming presence of Smartphones and the generalized use of Social Networks is that they destroy authentic face-to-face social interactions. A paradoxical reality when one considers how social networks have helped give birth to contacts between human beings thousands of miles apart...

The idea behind the BumpBand is simple. It aims to take advantage of social networks' incredible ability to bring two perfect strangers to interact, by applying it at the scale of a party. Via a recreational bracelet, surfing on the current trends of connected objects and of the idea of the quantified self, it aims to convert physical interactions into components of the social networks' architecture. Add to that idea a construction that guarantees user privacy, the ephemeral life span of the network and a series of practical apps and you get a entertaining object which can really enrich the user's experience of the event.

The user simply wears the bracelet around his wrist and syncs it with a smartphone app he previously downloaded and it is ready to use. The BumpBand has an embedded LCD display and can be lit up with virtually any colour. The user can therefore communicate via an existing color code with other users or simply customize his bracelet. Two people become BumpFriends (BF's) by bumping each other's bracelets. Once they are BF's, they will have access to a certain amount of features, which include a closed messaging system, and the possibility to request the two bracelets flash the same color, allowing BF's to regroup. Furthermore the user can access a live News Feed, which will keep him informed of his BF's activities.

The party's organizer can also message all users and start a Flash MoB (Flash Meet other Bumpers). The game, which strives to bring bumpers to meet each other, randomly selects five people who aren't already Bf's and sets their bracelet a specific color. These 5 users will have to Bump in the set time to win free drinks.

In addition to these features, the BumpBand offers several apps that aim to ease the user's experience. You can purchase tickets to the event online and simply Bump the bouncer to access the party. The bracelet can also save your coat hanger number, preventing the easy loss of a paper ticket. Finally it can be used to pay for drinks by bumping the bar and can use the subsequent data to analyze your alcohol consumption. From there, why not use your new social network to safely share a cab back home?

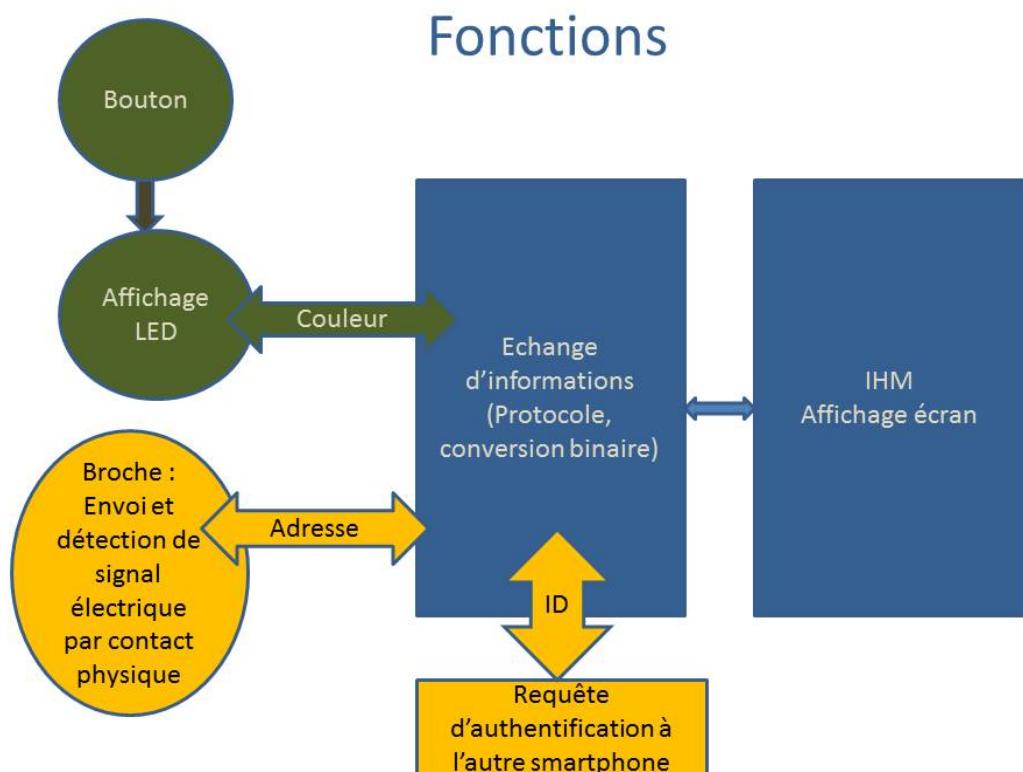
3. Description fonctionnelle du prototype allégé:

A. Description littérale:

Le bracelet permettrait via “Bump”(infrarouge) de bracelet à bracelet d’ajouter une personne en ami(qui sera stockée sur le bracelets), ainsi que de changer de couleur via boutons sur le bracelet ou virtuellement par l’interface du smartphone.

B. Description fonctionnelle du prototype:

- Communication bracelet – portable
- Communication portable – portable
- BUMP** incluant:
 - Communication infrarouge avec échange d’identifiants personnels (IP) entre 2 bracelets
 - Définition et mise en place du protocole de transmission d’identité (adresse IP) au Smartphone depuis le bracelet
 - Définition et mise en place du protocole d’authentification
- Réglage de la couleur sur le bracelet :
 - Via le bouton intégré
 - Via le smartphone
- Mise en place d’une IHM simplifiée pour contrôler les quelques fonctionnalités (liste de BF)
- Pas d’administrateur serveur à cette étape, ni d’envoi de fil d’actualités



C. Liste des modules et répartition des tâches :

Liste des modules :

Module SES - Business model

Expert : Rémi Maniak

Elèves : Marc Faddoul, Cédric Meston, Alexandre Tadros

Description : Dans ce module, nous étudierons l'aspect économique du projet. Nous chercherons à identifier les acteurs, les réseaux de valeurs associés et établir un business plan pour une éventuelle exploitation commerciale (stratégie de vente, pricing, branding...)

Module électronique - GPIO/Smartphone

Experts : Guillaume Duc, Tarik Graba

Elèves : Damien Auffret, Charles Deschard, Marc Faddoul

Description : Ce module a pour objectif d'établir la communication Bluetooth entre le Smartphone et le bracelet et la communication IR entre deux bracelets. La réalisation de ce module permettra de piloter le bracelet via le Smartphone (demander l'allumage des LED, vibrer...) et de transformer le bracelet en interface pour le réseau social. Il est central pour le projet, sa valeur ajoutée venant dans l'interaction sociale enrichie par les fonctionnalités du bracelet. D'un point de vue pédagogique, il nous permettra d'appréhender des protocoles de communication répandus et les entrées/sorties d'un module embarqué.

Module électronique - Bracelet

Experts : Guillaume Duc, Tarik Graba

Elèves : Damien Auffret, Charles Deschard

Description : Ce module nous permettra de connecter et de gérer les différents composants autour du micro-controleur afin de pouvoir le programmer. Il s'agit de construire le BumpBand et d'en assurer le fonctionnement interne. D'un point de vue pédagogique, il permettra de nous introduire à quelques notions d'électronique et à la programmation protocolaire d'un module embarqué.

Module Android - outils de communication

Expert : Jean-Claude Dufourd

Elèves : Arturo Castellanos Salinas, Félix Richart, Julien Roméro

Description : Le but de ce module est de créer les modules qui vont permettre aux appareils de communiquer entre eux. Il se découpera principalement en deux parties : la communication Bluetooth (avec le bracelet) et la communication Wifi (avec le portable et l'administrateur).

Module Androïd - services internes et interface graphique

Expert : Jean-Claude Dufourd

Elèves : Félix Richart, Alexandre Tadros

Description : Dans ce module, nous nous occuperons de coder toute la partie de l'application qui n'utilise que des données locales au téléphone, ainsi que l'interface graphique de l'application.

Module programmation distribuée

Expert : Thomas Robert

Elèves : Arturo Castellanos Salinas, Cédric Meston, Julien Roméro

Description : Le but de ce module est de créer les protocoles de communication entre les différents modules. Il faudra donc sans cesse communiquer avec les différents individus de groupe afin d'unifier les moyens de communication. Il faudra donc maîtriser différentes notions aidant à décrire les procédés et réussir à les expliquer.

Répartition des tâches :

- A. Contact (**Module électronique-Bracelet**)
 - a. Transmission physique via IR courte portée
 - b. Authentification à l'aide de l'horaire et d'adresse
- B. Affichage (**Module Electronique-Bracelet**)
 - a. Affichage LED
 - b. Bouton de commande pour piloter à travers le micro-contrôleur
- C. Communication bluetooth du bracelet (**Module Electronique-GPIO + Module programmation distribuée**)
 - a. Transmission infos bump
 - b. Pilotage du bracelet
- D. Envoi et réception depuis le smartphone:
 - a. Protocole et socket avec établissement de la communication et interprétation des messages (**Module programmation distribuée + Module Androïd-Communication**)
 - b. Gestion interne de la communication avec stockage des listes d'amis (**Module Androïd-Communication + Module Androïd-Services internes**) pour :
 - i. Communiquer en bluetooth avec le bracelet(réception d'une demande d'ami, envoi de couleur)
 - ii. Communiquer en wifi avec le smartphone(certifier l'identification)

E. Interaction avec l'écran (***Module Android-Services internes***)

- a. IHM pour :
 - i. choisir une couleur
 - ii. voir ami ajouté et liste
- b. avec gestion parallèle des changements d'écrans et des changements dans les autres applications(modifications de données)

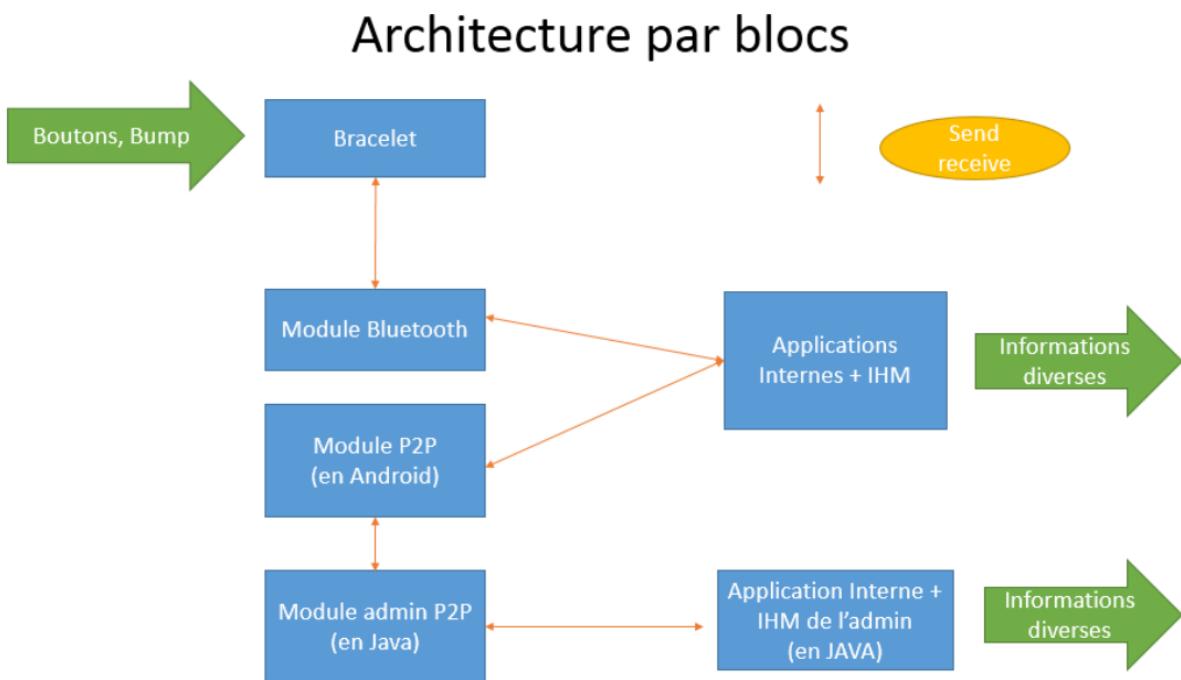
D. Liste du matériel demandé :

Voici la liste du matériel demandé afin réaliser trois bracelets ainsi qu'un administrateur (le bar par exemple). Il est également possible que l'on demande à emprunter des téléphones Android.

- | | |
|---|--|
|  | Carte Arduino mini : ATmega 328 / 16 MHz / 3,0 * 1,8 cm
/ 6 sorties PWM X3 |
|  | Bluetooth Mate Gold : module de classe 1 RN- 41 Bluetooth / se branche directement sur la carte Arduino / 4,4 * 1,6 cm X3 |
|  | Arduino Mini USB Adapter : conçu pour parler à un ordinateur / Pins TX et RX pour le contact de l'admin. X1 |
|  | Fibres optiques dénudées |
|  | Mini moteur à vibration : facilement intégrable X3 |
|  | LED RGB Cree PLCC4 : 3.2 *2.8 mm X6 |
|  | Mini écran LCD LMBD0820 : 5,8 * 1,1 cm X3 |
|  | Bouton poussoir X6 |
|  | LED émettrice infrarouge X3 |
|  | Récepteur infrarouge X3 |

4. Architecture informatique du prototype allégé:

Lors de la journée de Génie Logiciel, nous avons défini les différents blocs qui définiront notre projet. La principale difficulté était le nombre d'appareils entrant en jeu. Il nous fallait donc un découpage prenant en compte cela. De plus, les entrées/sorties sont très peu nombreux mais le nombre de communications internes est très important. Il fallait donc définir les différents protocoles de communication entre les blocs. Voici le découpage auquel nous sommes arrivés :

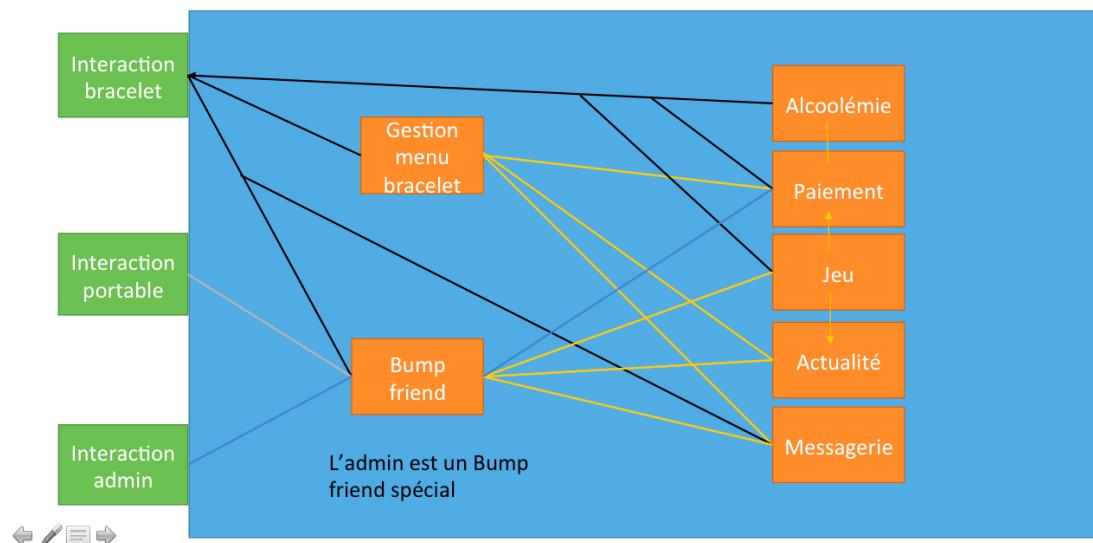


En fait, les entrées de l'utilisateur se font principalement sur le bracelet (afin d'éviter la sortie du téléphone de manière répétée) mais l'utilisateur pourra interagir aussi avec le téléphone. L'administrateur aura lui aussi une interface graphique et une borne de synchronisation. Concernant les sorties, on a au niveau du bracelet l'indication lumineuse, l'écran et le vibreur. Ensuite, au niveau du portable et de l'administrateur, des informations diverses.

Chaque appareil va comporter une partie communication avec les autres agents. Du fait des différents dispositifs de communication (Bluetooth et Wifi), des blocs différents seront nécessaires. De plus, si l'administrateur possède un ordinateur, il lui faut son propre module de communication.

Une fois ce schéma réalisé, il nous a paru bien pour décrire de manière générale notre système mais n'indiquait guère les fonctionnalités futures. Nous avons donc décidé de faire un schéma des applications internes :

Zoom sur les applications internes Android



Les entrées (autre que celles de l'utilisateur sur l'écran) se font par le biais des modules de communication. Le module gestion des menus permettra principalement de gérer les interactions avec les boutons et à affecter la bonne action. Ensuite, la gestion de liens sociaux passe par un module Bump Friend qui répartira les actions. Ensuite viennent les différents modules qui traduisent les fonctionnalités.

5. Planification temporelle:



6. Perspectives pour le prototype final:

En plus des fonctionnalités du prototype allégé (affichage de couleur et ajout en bump-friend), le prototype final :

- Sera muni d'un vibreur et d'un écran (en plus des LED)
- Permettra à l'utilisateur d'envoyer des messages à ses bump-friends via un système de messagerie locale, via le smartphone.
- Inclura un administrateur, qui pourra à tout moment lancer le jeu de rencontre (Tous les bracelets se colorent. Chaque utilisateur doit trouver et Bumper les autres utilisateurs ayant la même couleur que lui pour gagner une réduction par exemple, favorisant ainsi les rencontres entre inconnus.)
- Permettra de payer ses consommations à l'administrateur, ainsi que son vestiaire. Inclura un fil d'actualité sur le smartphone, qui nous indiquera ce que nos bump-friends sont en train de faire. (Un code couleur sera instauré. Par exemple, jaune signifierait "je vais

bientôt partir". Alors, lorsqu'un utilisateur met son bracelet en jaune, ses bump-friends sont avertis sur le fil d'actualité qu'il va bientôt partir.)

Nous avons jugé ces fonctions comme constituant une bonne bibliothèque capables de démontrer l'esprit du projet. Elles pourront ensuite être enrichies par un tissu applicatif personnalisé par l'utilisateur final. Si le projet est commercialisé à grande échelle, les fonctions que nous allons implémenter ne sont qu'un exemple des fonctionnalités que le bracelet pourrait avoir.

Tout d'abord, des fonctionnalités simples peuvent être ajoutées, comme la possibilité de voter pour la musique via le bouton du bracelet, ou utiliser le micro du smartphone pour faire bouger la lumière du bracelet en rythme avec la musique.

On peut également envisager la création d'un réseau social sur le web reposant sur le bracelet. On serait alors obligé d'avoir rencontré la personne pour pouvoir être bump-friend avec lui sur le réseau social.

Et si le bracelet est utilisé en extérieur, alors on peut se servir des coordonnées GPS de l'utilisateur et faire vibrer le bracelet lorsqu'un autre utilisateur partageant des points communs avec le premier passe à proximité de lui. Cela renforcerait d'autant plus le côté social, allié à la proximité. En plus en extérieur, il n'y a pas le problème d'y avoir trop de gens aglutinés les uns aux autres, donc la détection avec la couleur sera plus facile.

Structure de Test:

Afin de savoir si un programme fonctionne correctement, nous avons du mettre en place un séquence de tests. Celle-ci vise à parcourir toutes les fonctionnalité de l'appareil pour vérifier le bon fonctionnement après chaque mise à jour.

Il va comprendre cinq acteurs :

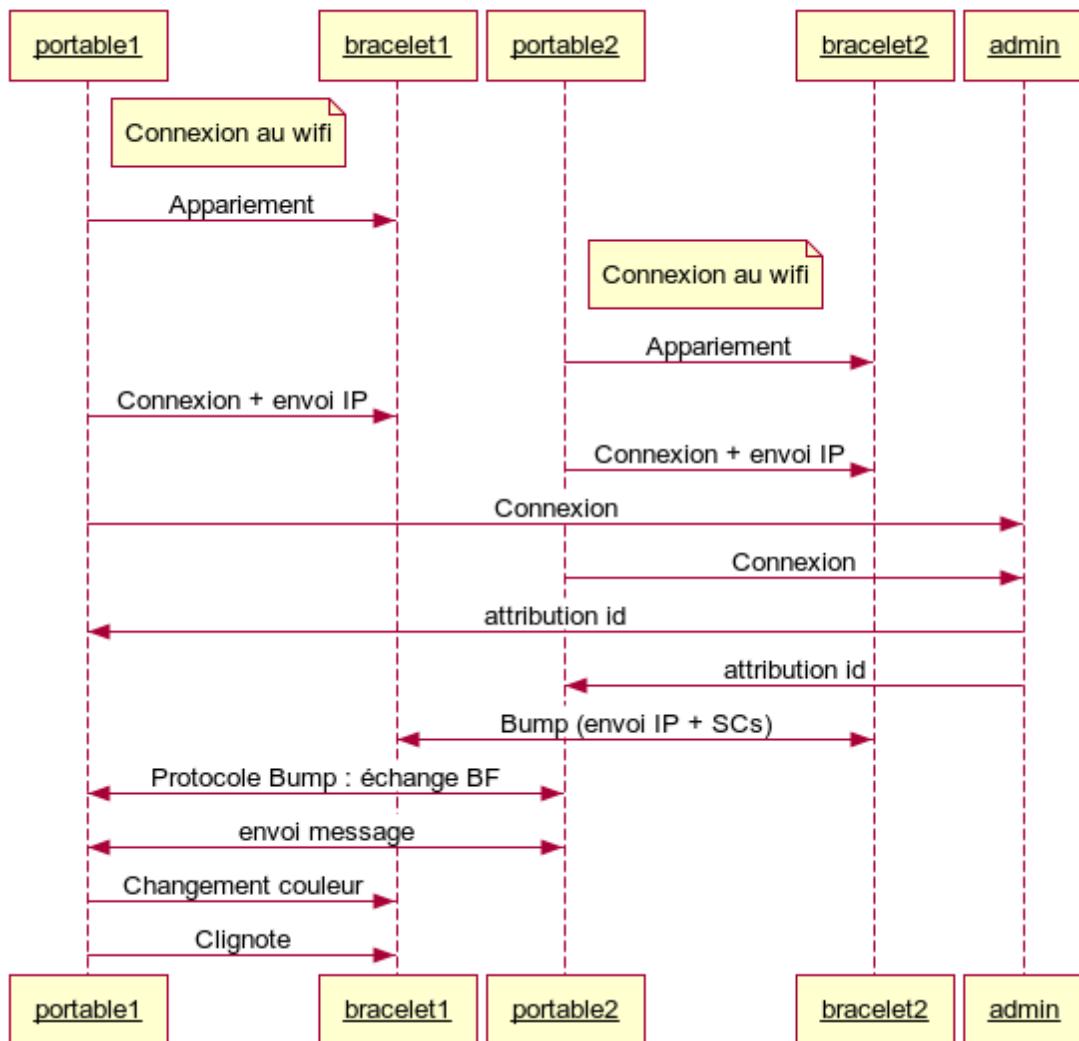
- deux portables. En effet, la communication est a coeur de notre projet. Dans un premier temps, nous utilisons deux portables pour la simplicité de l'écriture du diagramme mais il est tout à fait possible et même recommandé de faire les tests avec au moins trois portables.

- deux bracelets. Ils sont inséparable de leurs portables.

- un administrateur. C'est lui qui gère la liste des participants avec leur identification. Il est unique et sera en général contrôlable depuis le bar.

Pour illustrer la situation, nous avons choisi de nous appuyer sur un diagramme de séquence. Celui-ci s'adaptera à l'avancée des fonctionnalité dans les divers acteurs.

Test



Pour tester les classes d'Android, nous utilisons une librairie appelée Robolectric.



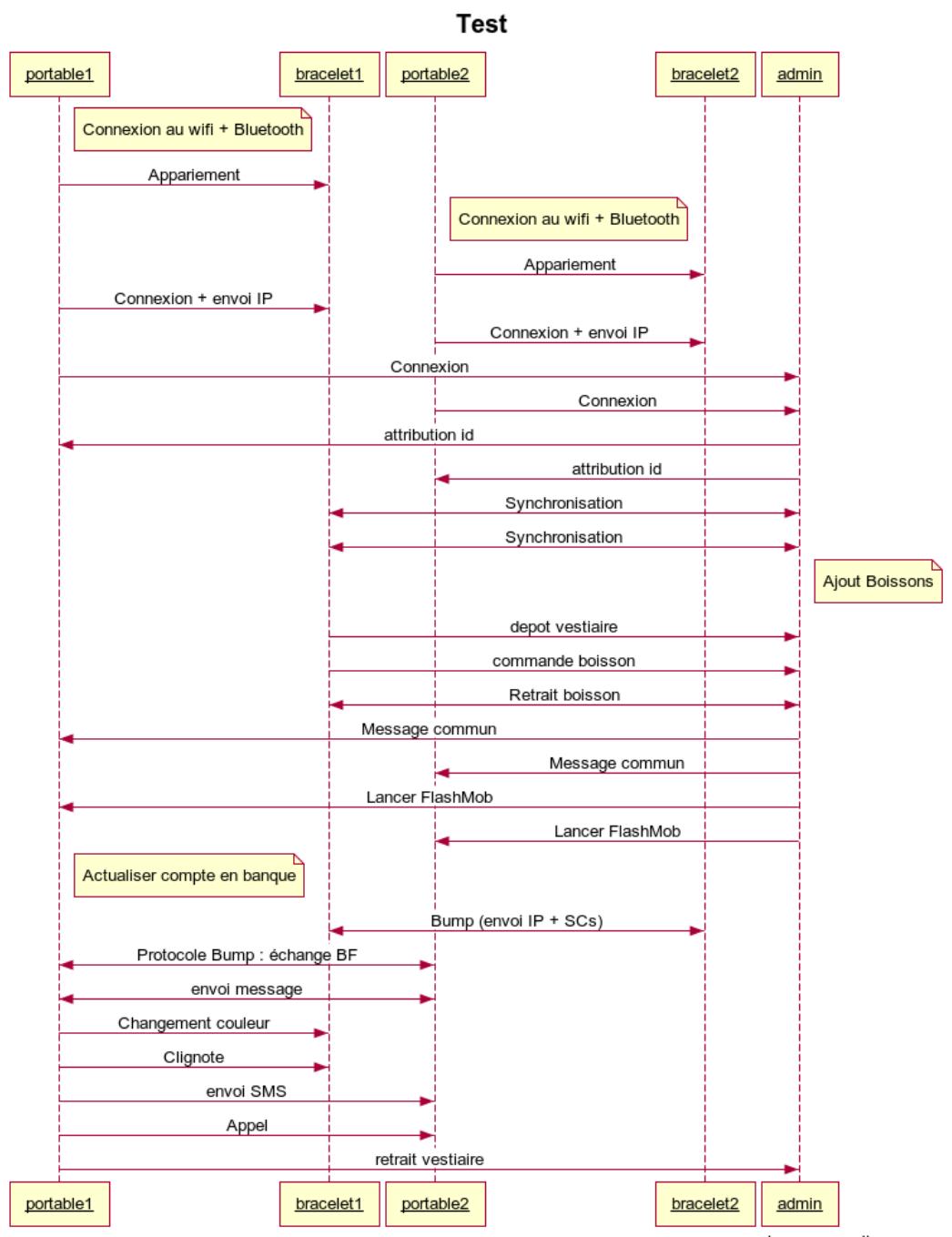
Que propose robolectric ? Tout d'abord, il est open source. Ensuite il permet de réaliser des tests très rapidement. En effet, le programme n'utilise pas Dalvik mais directement la JVM. Il est aussi plus simple à utiliser que Junit.

L'installation de robolectric fut des plus périlleuse : il s'intègre assez mal dans Android Studio, ou du moins il n'existe pas de méthode directe pour le faire. Il nous a donc fallut manipuler gradle, ce qui est toujours compliqué et au final, il ne peut se lancer qu'à travers la

console.

Grâce à cette librairie, nous pouvons tester de nombreuses manières nos classes. Cependant, il est difficile de tester les connexions réseaux en TCP. Or il s'agit là d'une grande partie de notre projet. Il faut donc une approche plus manuelle, qui suit le schéma de Test décrit plus haut.

Pour trouver nos fichiers de test, vous pouvez vous rendre sur notre dépôt git, sur la branche la plus récente (<https://github.com/projetpact41/bumpband>).



www.websequencediagrams.com

La structure de tests comporte cinq acteurs :

- deux portables. En effet, la communication est à cœur de notre projet. Dans un premier temps, nous utilisons deux portables pour la simplicité de l'écriture du diagramme mais il est tout à fait possible et même recommandé de faire les tests avec au moins trois portables.
- deux bracelets. Ils sont inséparables de leurs portables.
- un administrateur. C'est lui qui gère la liste des participants avec leur identification. Il est unique et sera en général contrôlable depuis le bar.

Après synchronisation avec l'administrateur, le bracelet est connecté avec le portable par bluetooth. Ensuite les deux bracelets communiquent par infrarouge lors d'un Bump. Il est important de noter que les interactions avec le compte en banque de l'utilisateur passent par l'admin.

7.Résumé de l'état des modules:

MODULE GPIO/ÉLECTRONIQUE:

(i) Sommaire du module :

Description : Le groupe PACT ambitionne de réaliser un réseau social, éphémère et de proximité, basé sur un bracelet connecté à un Smartphone Androïd via Bluetooth. L'application java doit être en mesure de piloter une LED RGB et un vibreur installés sur le bracelet et de recevoir via Bluetooth l'état des boutons poussoirs qui y sont connectés. Le contact physique (« bump ») entre deux bracelets doit permettre aux deux Smartphones associés de pouvoir communiquer entre eux (échanges de messages, liste d'amis)

Choix techniques : Le bracelet est centré autour d'une carte Arduino Pro Mini. Elle communique via Bluetooth avec le Smartphone à l'aide d'une carte Bluetooth Mate Gold de Sparkfun (Simulation d'une liaison sérielle via Bluetooth). Nous y ajoutons un vibreur, 2 boutons poussoirs et une LED RGB. Afin de permettre le « bump », chaque bracelet stocke l'adresse IP (sur le réseau local) du portable associé et fait de la communication par champ proche en permanence via infrarouge (Protocole Sony Remote (porteuse 38 kHz)). Chaque bracelet est donc équipé d'un émetteur et d'un récepteur IR (Diode 950nm SFH415_U et TSOP 1738). Lors du « bump », les bracelets se communiquent mutuellement ces IP et les retransmettent à leur Smartphone respectifs, ce qui leur permet à leur tour de communiquer.

Calendrier d'avancement sommaire :



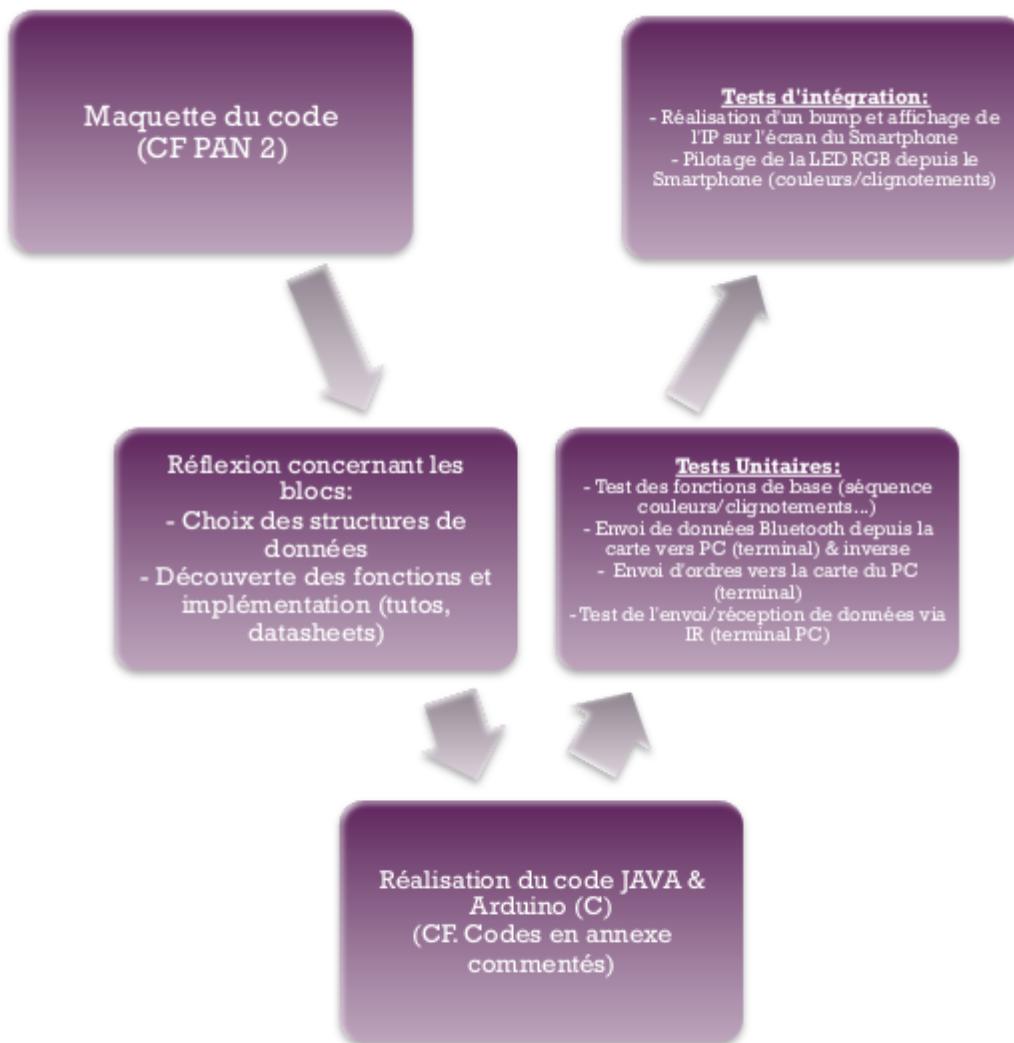
(ii) Etat d'avancement/ Réalisations :

Tous les objectifs du PAN 3 (CF diagramme de Gant du cahier des charges) ont été réalisés. Nous avons réussi à réaliser :

- La communication via infrarouge d'une IP entre deux bracelets
- La communication de cette IP reçue à l'application Androïd
- L'envoi d'instructions (pilotage LED RGB et vibrer) au bracelet via Bluetooth et leur réalisation.

Toutes ces fonctions ont été testées. Le développement réalisé au PAN 3 (pour ce module) est précisé dans le diagramme en V ci dessous :

Diagramme des tâches:



Comme prévu une démonstration du prototype allégé sera réalisée lors de la présentation du PAN 3.

ANNEXES: Veuillez également trouver dans le rapport mail le code Arduino (commenté) ainsi que la gestion des I/O dans l'application Androïd (sur notre GIT). Le schéma du montage électronique est également inclus.

(iii) *Ressenti/ Difficultés rencontrées :*

Le groupe s'est bien entendu et nous avons pu réaliser les tâches efficacement. L'écrasante majorité du code a été créée et testé à Télécom en groupe (pendant et en dehors des horaires consacrés), ce qui nous permettait d'efficacement échanger nos idées. Chacun a également su se renseigner individuellement (lecture et assimilation de tutoriels (très nombreux sur le net) et des datasheet). Nous avons également essayé d'avoir plus de contacts avec nos experts qu'au PAN 2.

Nous avons également su court-circuiter une partie de l'intégration en travaillant sur la partie I/O du code JAVA avec Julien Romero (module Androïd) ce qui nous a permis de nous mettre bien d'accord sur les protocoles. (Recommandation du jury lors du PAN 2). Cette réflexion en amont a beaucoup simplifié l'assemblage des différents blocs.

Les principales difficultés rencontrées concernaient particulièrement le format et la structure des données manipulées (conversions et intégration assez minutieuses).

La mise en place du Bluetooth a été assez difficile, notamment à cause d'une erreur sur la compilation (type de carte destinataire). De même l'utilisation de la lib. IR de Ken Sherrif a été assez complexe et l'aide de T. Graba nous a été précieuse. Nous avions du mal à comprendre comment alterner émission et réception (présence de timers dans la lib). Le déclanchement de fonctions dans la loop() risque de ralentir l'émission/réception IR. Utiliser les timers de la carte ou de la lib IR pourrait être une solution lors du PAN4 pour améliorer les performances.

(iv) *Perspectives pour PAN 4 :*

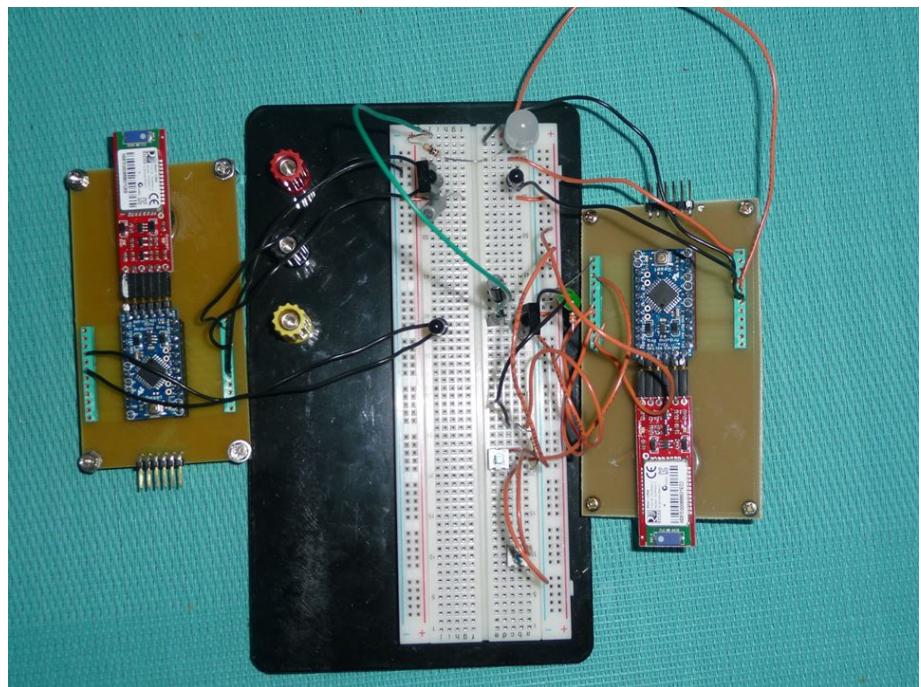
Il reste deux grandes tâches à réaliser : l'assemblage des composants pour former un prototype de bracelet ainsi que la validation du travail du groupe GPIO dans des conditions utilisateurs (alpha). Il s'agira également de régler la puissance d'émission de la diode IR, afin de ne permettre que les communications en champ (très) proche (calibration empirique).

Nous envisageons de créer un petit PCB pour le raccordement des composants, créer un bracelet à l'aide d'une imprimante 3D et de travailler sur une solution d'alimentation portable. Il s'agira alors de tester la portabilité et le fonctionnement de l'ensemble.

L'alpha test consistera en une simulation d'une « soirée », où nous vérifierons que le bracelet permet bien de réaliser toutes les tâches de notre scénario d'usage.

Annexes :

(i) Montage électronique:



TEST D'UN BUMP

A gauche : Bracelet uniquement équipé du Bluetooth et de l'IR

A droite : 'Bracelet' complet

(ii) Code Arduino :

(Cf. Code joint dans le mail)

(iii) Code JAVA :

(Cf. Git groupe)

1. Module programmation concurrente et/ou distribuée:

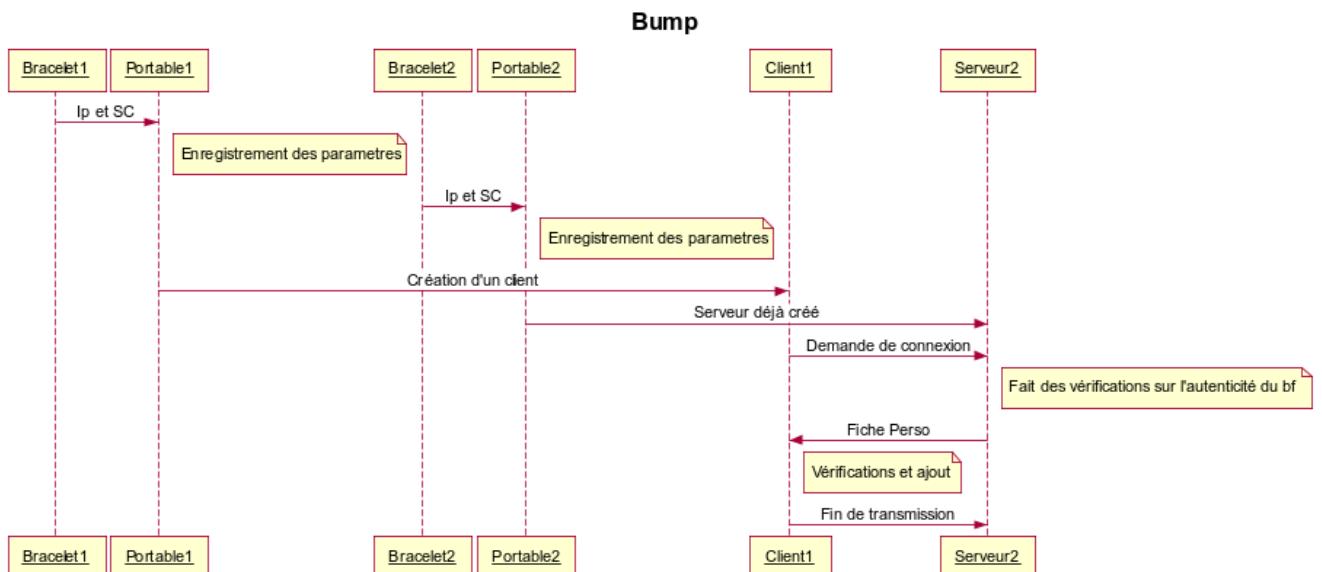
Définitions :

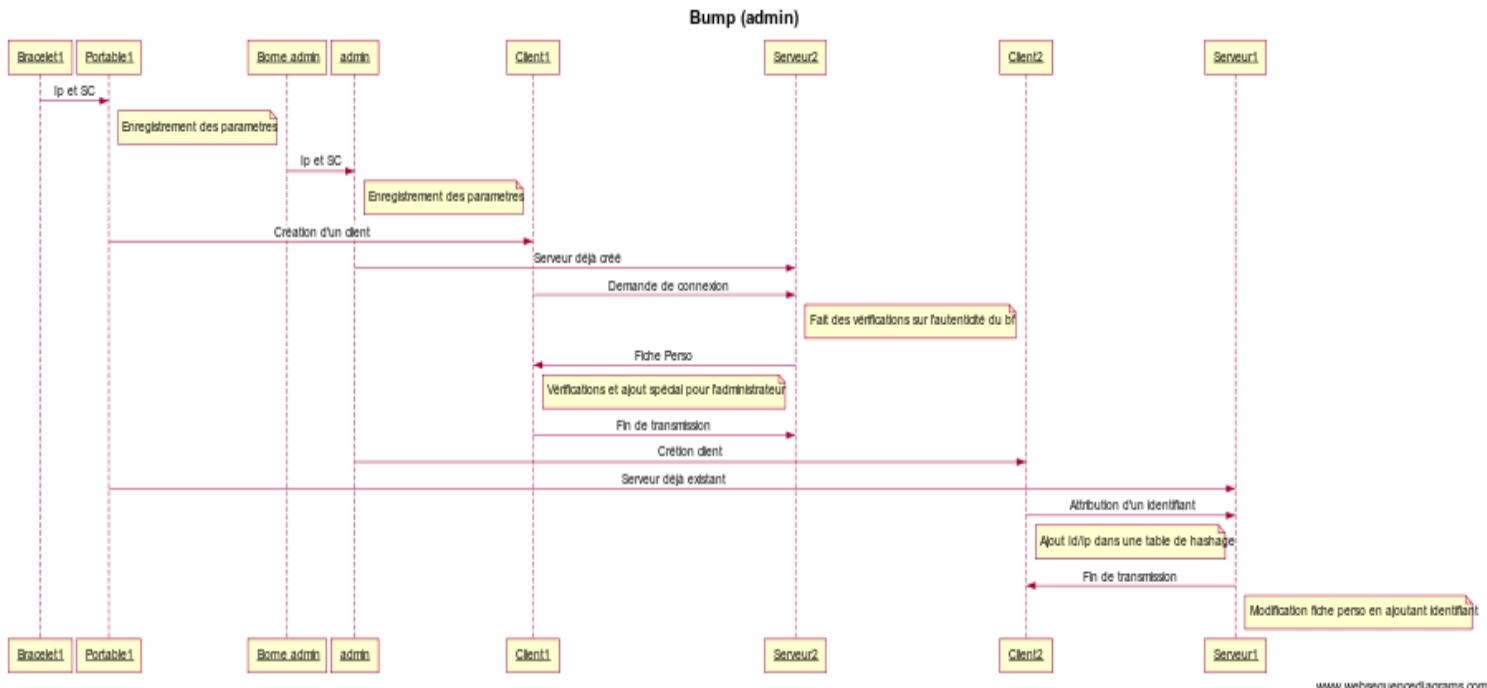
- Thread : similaire à un processus, il permet de séparer le programme en plusieurs files d 'exécution et ainsi permet d'effectuer des actions en parallèle.
- Variable partagée : variable commune à plusieurs threads/processus. Il faut faire attention lorsqu'on les modifie/lit depuis plusieurs threads car des problèmes peuvent apparaître.

- Messages : entités transmises au cours d'une communication.
- Protocole : définit la façon de coder les informations dans les messages transmis.
- Interface de communication : technologie utilisée pour transmettre l'information.

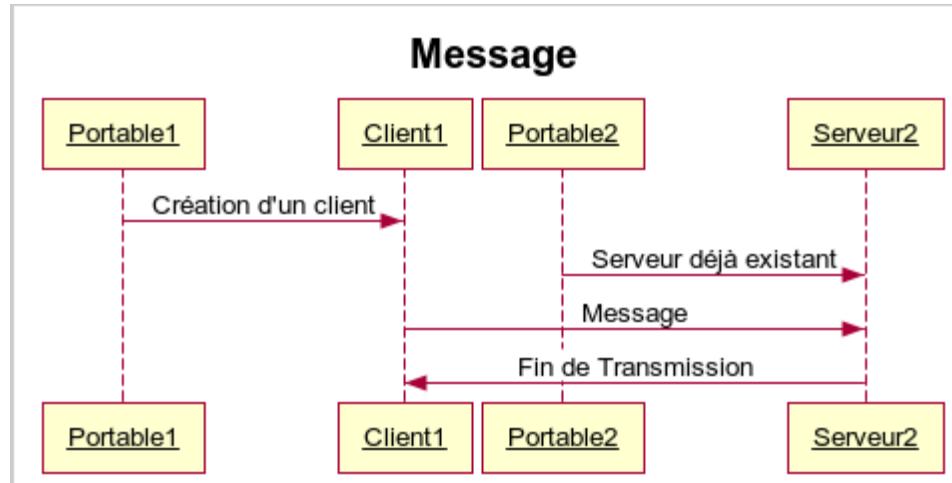
Ce module est le lien entre plusieurs modules : GPIO, programmation Android, et programmation de l'administrateur. Il définit les différents protocoles de communication pour les interfaces de communications suivantes : wifi, Bluetooth et infrarouge. Il aiguille les différents acteurs sur l'utilisation de différents threads pour permettre l'utilisation d'applications concurrentes.

Pour chaque actions, il nous faut définir des diagrammes de séquence. Ceux-ci nous permettent de connaître l'agencement d'une communication pour arriver à un certain objectif. En voici la liste :

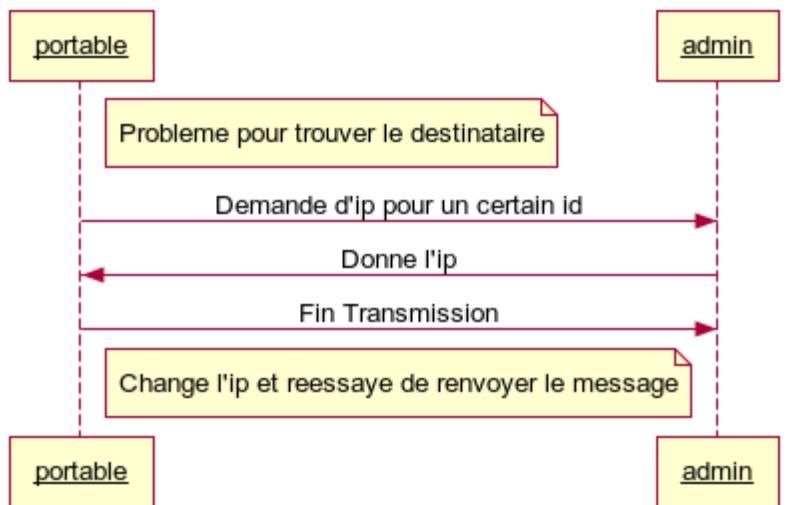




(Il nous faut considérer les cas du jeu de couleur.)

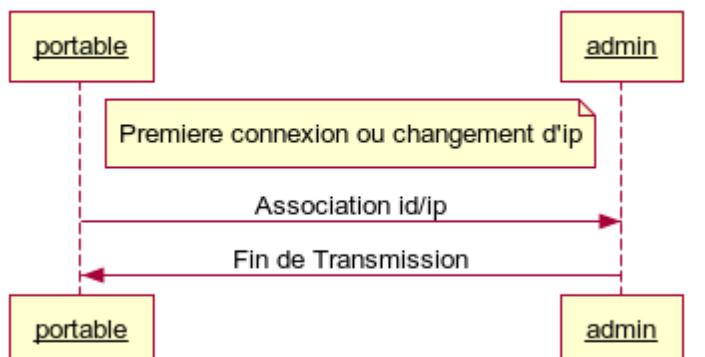


Demande d'identifiant



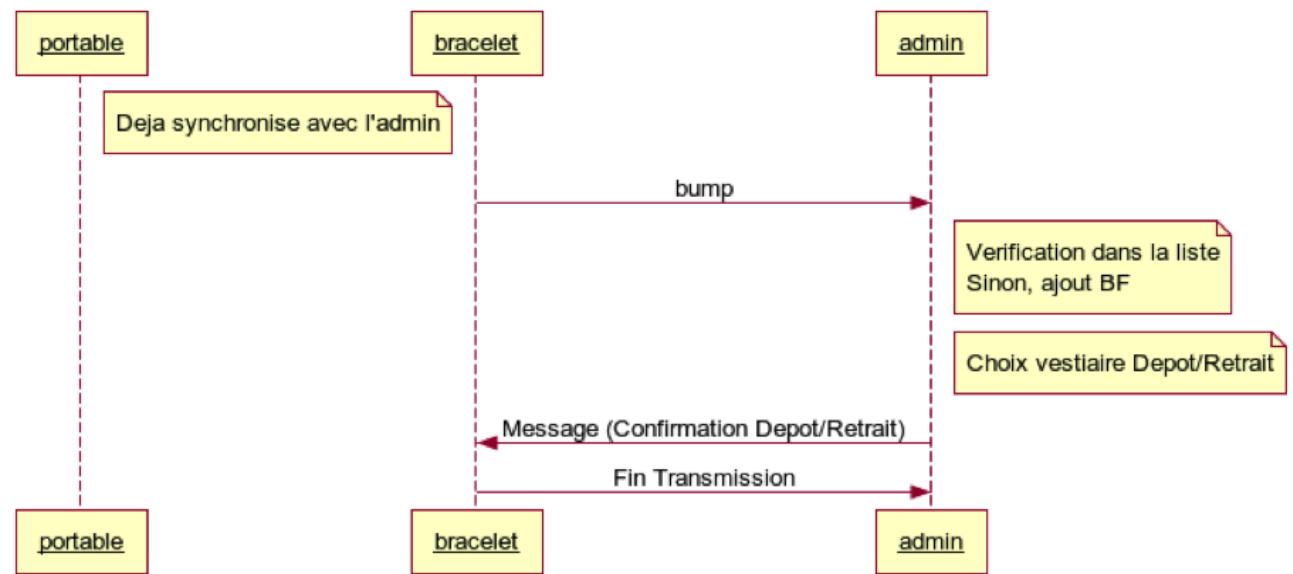
www.websequencediagrams.com

Attribution ip



www.websequencediagrams.com

Vestiaire



www.websequencediagrams.com

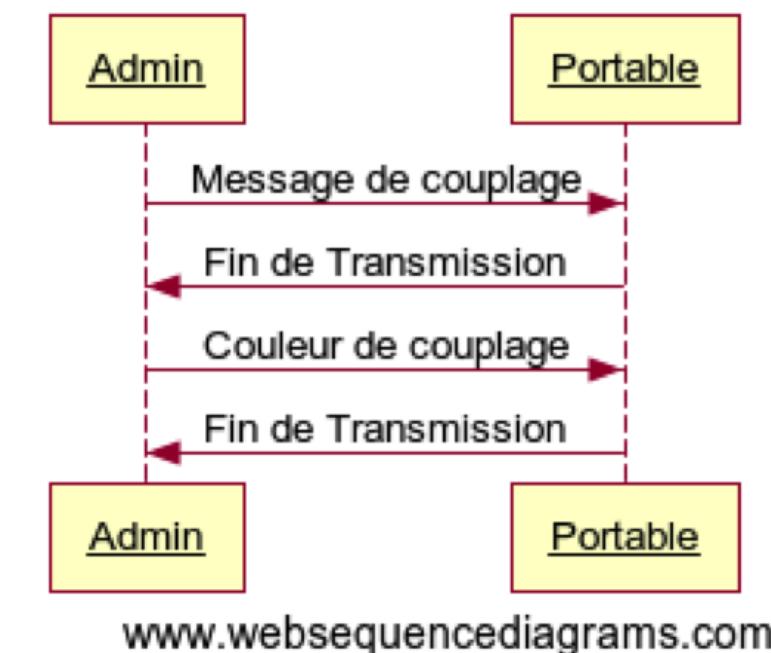
Dans notre projet, nous utilisons de nombreux threads afin de pouvoir faire plusieurs actions en parallèle. Le serveur est sur un thread pour qu'il puisse tourner en tâche de fond. A chaque fois qu'un client se connecte, un nouveau thread est créé afin de pouvoir traiter plusieurs clients simultanément. Lorsque l'on crée un nouveau client, un thread est aussi créé. Il y a aussi un thread créé pour le serveur Bluetooth.

Il y a quelques variables partagées, qui se révèlent surtout être des fichiers. Tout d'abord, la liste des bump friend est partagée, ainsi que la fiche personnelle, l'identité de l'administrateur, la liste des messages. Il faudrait penser à utiliser des sémaphores afin de gérer l'accès à ces fichiers. Ce sera utile si l'on bumpe trop rapidement ou que l'on reçoit trop de messages d'un coup.

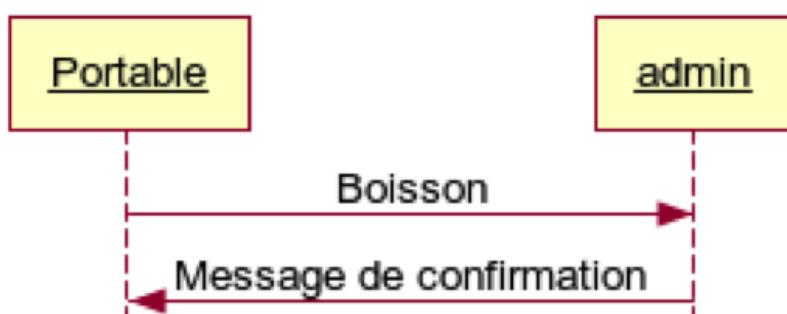
Fin de commande



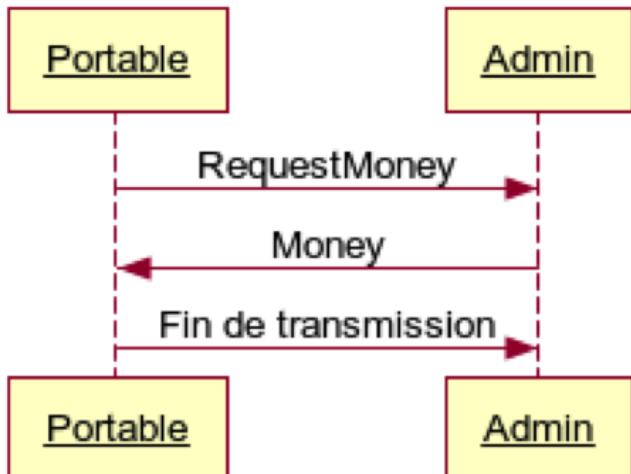
Debut FlashMob



Commande

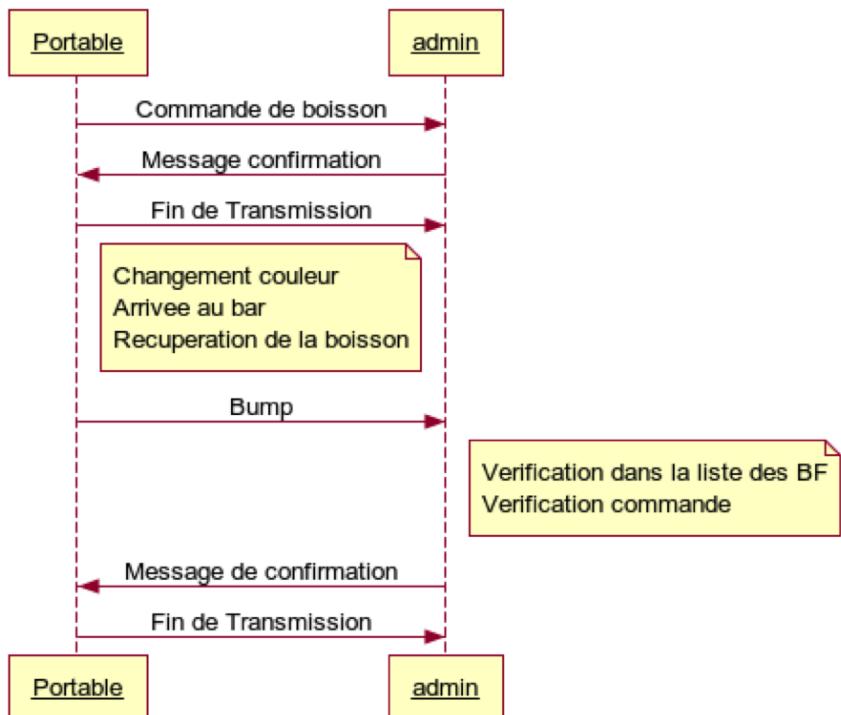


Demande solde Banque



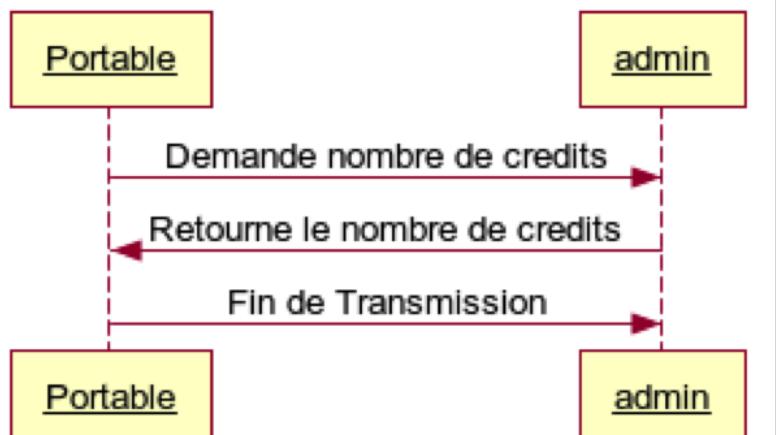
www.websequencediagrams.com

Bar



www.websequencediagrams.com

Demande credit



www.websequencediagrams.com

Comme nous l'avons déjà précisé plus haut, il y a trois interfaces de communication. D'abord, le wifi. L'envoi se fait à l'aide du protocole TCP, mais cela se révèle assez transparent pour nous. Ensuite, il y a le Bluetooth, avec son protocole. Enfin, il y a aussi l'infrarouge entre les deux bracelets.

Ce qui nous intéresse plus, c'est la manière dont sont codés les messages transmis, c'est à dire le protocole de communication que nous mettons en place. Pour faire simple, nous résumons cela dans des tableaux.

Pour le Bluetooth :

La séquence de base est la suivante :

Nbr de bytes transmis(sur un byte) + message.

Voici le tableau servant à coder le message :

Bracelet vers portable		Envoyez des bytes depuis le bracelet de la forme			
Fonctions	Bump	Bouton1	Bouton2		
Code	0	1	2		
Envoie	0+IPadresse+SC1+SC2	1	2		
Codage	IPadresse : 4 bytes, SC1 et SC2 : 1byte chacun pour l'instant				

Portable vers bracelet		Envoie des bytes au bracelet			
Fonctions	sendMessage	vibre	sendColor	clignote	sendIp
Renvoie la string	0+message+durée	1+durée	2+numeroCanal+couleur	3+fréquence+duree	4+IP
	message est une string (codé en ascii?) transformée en bytes		couleur sera une string de 3 bytes	fréquence aura une valeur entière (en bytes)	IP sous forme de 4 bytes.
	durée est un byte (en secondes)		numeroCanal sera un tableau de bytes	duree en s sur un byte	

Pour le Wifi :

Le motif de base est le suivant :

Taille + numéro fonction + arguments

La taille représente le nombre de bytes envoyés. Cela facilite la réception. Le tableau suivant précise l'encodage :

BumpFriend	0+nom+μ+ip	Nom est le nom du bf, μ sert de séparateur, ip est une chaîne de caractères représentant l'ip. Tout est en ascii.
Color	1+rouge+vert+bleu	Rouge, vert et bleu sont des octets, codant la couleur.
Connexion	2+ip+SC1+SC2	Ip est l'ip de celui qui fait la demande, SC1 et SC2 sont les

		codes de sécurités, codés sur un octet chacun
Message	3+message+ +expéditeur	Message et expéditeur sont les chaines de caractères, sert de séparateur.
Transmission	4+ConnexionReussie (+erreur)	Connexion réussie = 1 si tout s'est bien passé, 0 sinon. Erreur est une chaîne de caractères codant l'erreur.
Identifiant (spécial admin)	5+identifiant	Identifiant sur 4 bytes
IdentifierAnswer	6+ID+IP	id (int) ip (String)
Menu(spécial admin)	7+(n x boisson)	boisson=prix(1 byte)+Color(rvb 3)+degre(1 byte)
Boisson(spécial admin)	8+prix+rouge+vert+bleu+n om	
RequestMoney	9	
Money (spécial admin)	10+money	money = 1 byte
FMConfirmation	11+ip	
Vestiaire_ajout	12	
Vestiaire_retrait	13	
RetirerBoisson	14	

Ce tableau sera à compléter à chaque fois qu'une nouvelle fonction est ajoutée.

Infrarouge :

C'est le groupe GPIO qui s'en occupe.

Nous avons codé les parseurs pour le Bluetooth et pour le Wifi. Ceux-ci vont être utilisés par les modules programmation android et administrateur. Il a cependant été nécessaire d'adapter selon que le programme marche sur Android ou sur l'ordinateur. En effet, certaines fonctions et classes existent sur l'une et non sur l'autre, comme par exemple les logs ou le context.

En pratique :

Pour le wifi, nous auront besoin d'utiliser les classes Sockets et SocketServer, autant pour la programmation android que pour la programmation de l'admin en java « pur ». Les communications vont se faire en P2P (Peer-to-Peer), c'est à dire que chaque téléphone est à la fois serveur et client. Nous imposons donc aux modules programmation android et admin de respecter cette norme.

Pour le Bluetooth, on commence par chercher dans la liste des appareils appareillés celui se nommant « Bracelet ». Ensuite, on se connecte à une socket et on récupère les flux d'entrée et de sortie. Un thread est lancé pour qu'un serveur écoute ce qu'il se passe.

Le code est dans un état plutôt avancé. Il est disponible sur le serveur git. Nous avons créé les communications wifi et les communications Bluetooth.

Au départ nous avions opté pour un envoi d'objets directement entre les plateformes android. Cependant, cela posait quelques problèmes de portabilité, et l'impossibilité de développer l'application sur IPhone ou Windows Phone. Nous avons donc du refaire le protocole et recoder ce qui l'avait déjà été.

Les objets pouvant être transmis implémentent l'interface Transmissible. Celle-ci demande à la classe de réaliser deux fonctions : execute (qui est une fonction onReceive) et toBytes qui permet de transformer l'objet en bytes pour être transmis. Le protocole est donc en partie codé dans chaque objets.

Nous venons d'intégrer en partie le bracelet dans le portable. Nous avons réussi à transmettre de l'information et à générer le début des protocoles. Cependant, nous nous sommes heurté à un obstacle : certains messages Bluetooth étaient scindés en plusieurs parties, rendant alors la lecture impossible. Il a donc fallut modifier le protocole bluetooth : maintenant, les séquences commencent par la taille du message transmis (ce qui n'était pas le cas avant).

Cette phase d'intégration s'est révélée plus longue que prévue et nous as permis d'exhiber des erreurs que nous n'avions pas décelé plus tôt.

Dans ce module, nous sommes en avance par rapport aux attendus du pan3 : le code est proche d'être complet même certains ajustement sont nécessaire. Nous devrons sans doute chercher à optimiser d'avantage le programme durant le pan4. Le protocole est déjà intégré dans le téléphone et dans le bracelet. Il nous restera à intégrer l'administrateur au protocole.

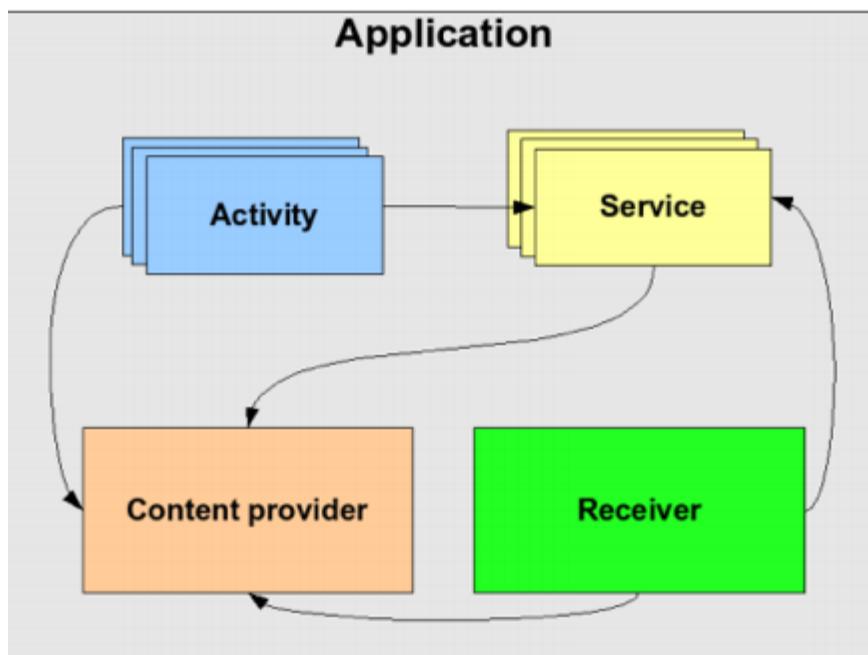
Pour ce qui concerne les tests, nous suivrons le processus décrit dans le cahier des charges. Il est difficile d'automatiser ces tests car ils font intervenir beaucoup de connexions à des réseaux très différents.

2. Module Android : Outils de communication

Dans un premier temps, nous avons mis en place les outils nécessaire au développement. Nous avons choisi d'utiliser Android Studio et d'émuler sur GenyMotion. Nous avons rencontré un certain nombre de problème, notamment pour porter un projet d'un ordinateur à l'autre.

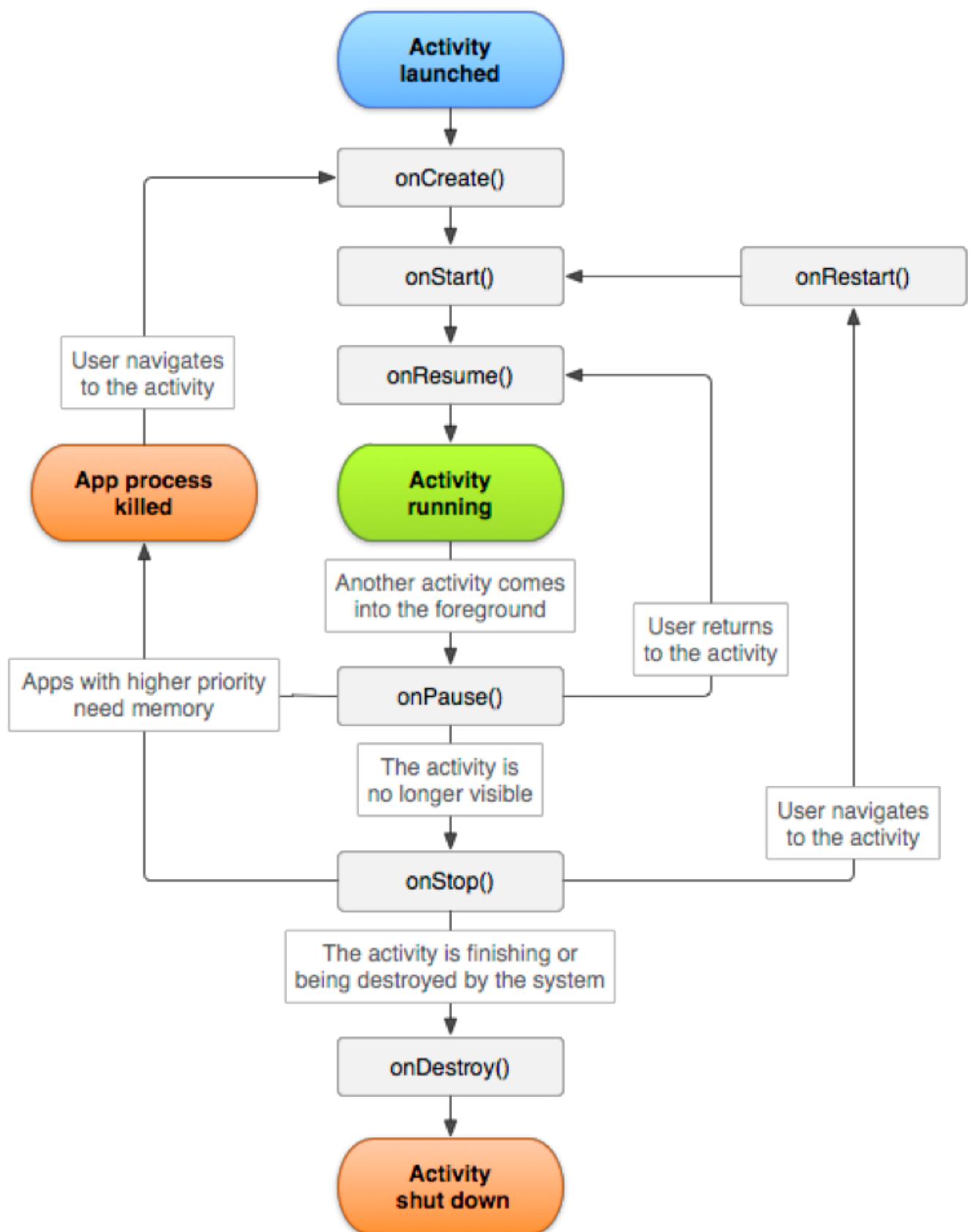
Nous nous sommes ensuite intéressé au cycle de vie d'une application, d'une activité et des Services.

Une application est composé de quatre composants : les « Activity », les « Service », les « BroadcastReceiver » et les « ContentProvider ».



(<http://www.iro.umontreal.ca/~dift1155/cours/ift1155/communs/Cours/1P/02CycledeVie.pdf>)

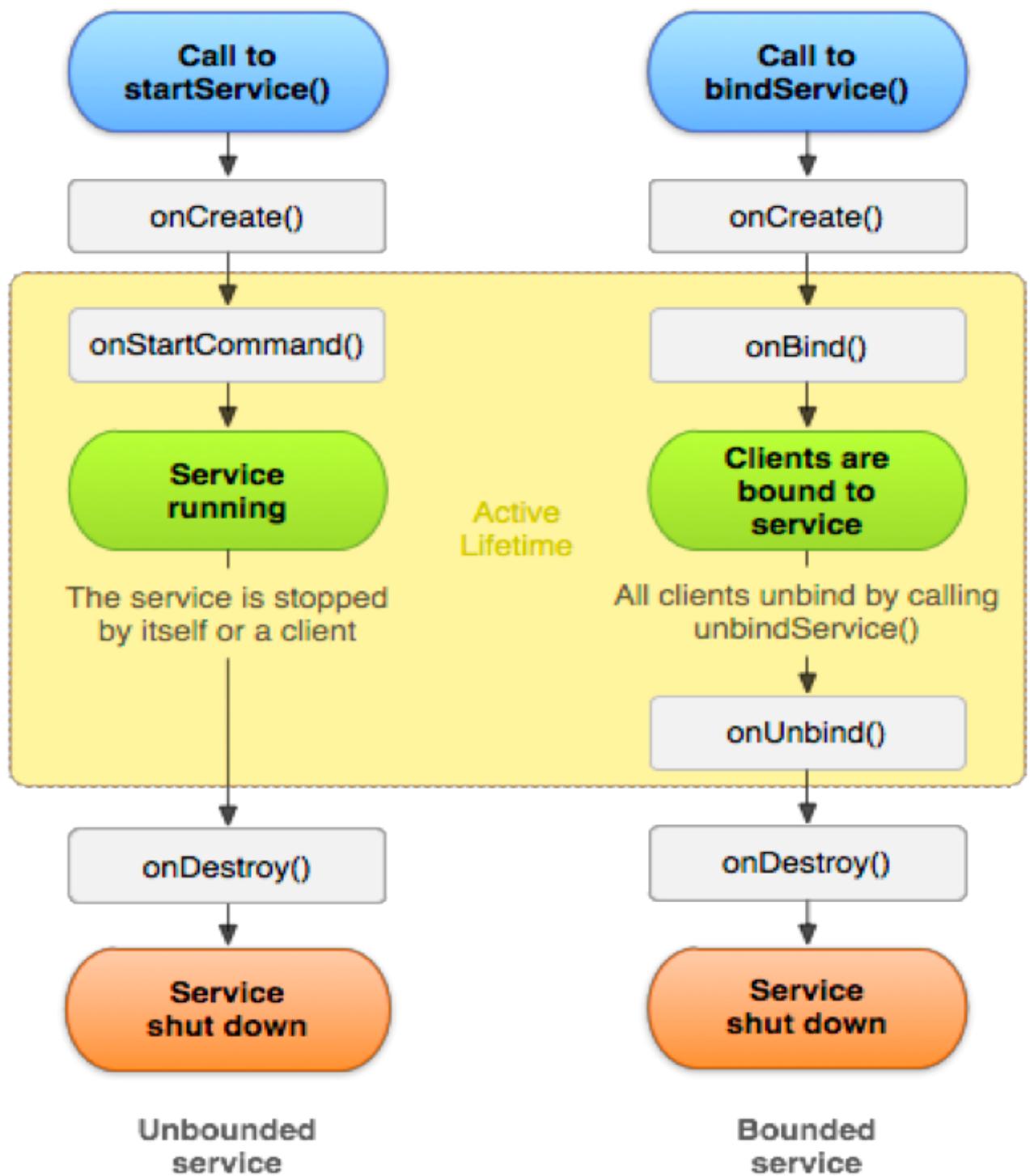
Une Activité représente l'interaction entre l'utilisateur et le téléphone. Il s'agit de ce qui s'affiche à l'écran. Il n'y a qu'un activité par page et chacune représente une page différente. Voici le cycle d'une activité :



(<http://developer.android.com/reference/android/app/Activity.html>)

Nous avons pu implémenter certaines de ces fonctions (qui héritent de la classe Activity).

Un service est une action qui est effectuée en arrière plan et qui n'est pas visible. Il prodigue un certain « service », c'est à dire qu'il effectue une action donnée sur demande. Voici son cycle de vie :



(<http://developer.android.com/guide/components/services.html>)

Un BroadcastReceiver est un récepteur d'intention. Lorsqu'il se produit un événement sur le portable, comme la réception d'un SMS ou une alerte de batterie faible, Android émet une intention pour informer le reste du système. Le BroadcastReceiver l'intercepte et agit en conséquence.

Les ContentProvider permettent de gérer des bases de données.

Nous avons ensuite créé l'application. Nous avons crée une activité principale qui nous demande un identifiant et une autre page afin de vérifier si nos communications se passent bien. Nous nous sommes ensuite attelé à programmer la communication entre deux téléphones. Dans un premier temps il s'agissait simplement d'un serveur écho.

Pour stocker la liste des BumpFriend, nous avons du nous intéresser au stockage des données persistantes. Nous avons essayé les Préférences, mais ils se sont révélés trop lourd pour ce genre de données. Nous avons ensuite utilisé une simple classe stockant un tableau de bump friend. Cependant, si l'application venait à se fermer, les données étaient réinitialisées. Nous avons finalement opté pour les fichiers textes. Nous aurions pu utiliser des bases de données, mais celles ci ont été jugées inutiles par un expert en début de projet et elles impliquaient la maîtrise de la SQL.

Nos avons ensuite programmé la communication Bluetooth. Le module Protocole de communication nous a fourni les parseurs pour le protocole à utiliser. Nous les avons intégrer. De plus, ils nous ont aussi fournis des diagrammes de séquence qui nous ont permis de savoir comment se passe chaque action.

Dans un premier temps, nous envoyions directement des objets par WiFi. Cependant, cela posait quelques problèmes, notamment pour la portabilité. Nous avons donc suivi un nouveau protocole, en envoyant des octets. De l'ancienne programmation subsiste des objets implémentant l'interface Transmissible, possédant une fonction execute() (fonction du type onReceive) qui effectue la bonne action. Il y a aussi une fonction toBytes() traduisant l'objet pour le transmettre. A chaque extrémité d'une communication, l'objet est transformé en bytes et ensuite retransformé en objet. Ainsi, nous avons pu conserver une partie de notre ancien travail tout en réduisant la charge transportée et en augmentant la portabilité.

Une grosse étape a été celle de l'intégration du bracelet. Nous nous sommes retrouvé avec le module GPIO et nous avons mis au point la communication bracelet portable durant toute une après midi. Des erreurs sont apparus mais grâce aux messages du Log, elles ont pu être identifiées. Un problème est aussi apparu avec l'utilisation des bytes. Sous Java, ces derniers sont signés donc vont de -128 à 127. Il nous a fallut manipuler des bits avec des >> et des & pour pouvoir les retransformer correctement en int.

Concernant l'avancement du module, les attentes du pan3 sont validées : les parties Bluetooth et Wifi ont été implémentée et intégré aux portables et aux bracelets. Pour le pan4, il nous restera à intégrer le module administrateur et à optimiser le code. Il pourra aussi apparaître

de nouveaux objets Transmissible qui devront être intégré.

3. Module Administrateur

En accord avec le module protocole de communication, les protocoles ont été adaptés pour donner une priorité à l'administrateur. Au début de l'initialisation du programme sous Android, l'utilisateur doit se synchroniser avec l'administrateur. Ceci fait, il garde l'adresse ip de l'admin, qui sera supposée fixe, en mémoire. Quand il reçoit un message prioritaire, il vérifiera l'identité de l'administrateur.

Nous avons réalisé les premiers essais de communication WiFi entre deux ordinateur. Cela nous a permis de comprendre le fonctionnement des Sockets. Ensuite, le programme a été adapté pour Android et des essais ont été réalisés entre un ordinateur et un smartphone. Après cela, l'application Android a subit de gros changements. Initialement, les communications se faisaient par envoi d'objets, mais cela impliquait trop de contraintes. Le programme a été revu dans son intégralité pour conserver le travail fait tout en changeant le mode de communication. Suite à cette phase de changement, un programme a été retransféré sur l'administrateur. Les protocoles ont du être adaptés et certains même créés.

Nous avons ensuite géré les identifiants. Les ip n'étant pas forcément fixes, il fallait trouver un moyen de quand même pouvoir retrouver un destinataire en cas de changement d'ip. Après la synchronisation avec l'administrateur, le smartphone obtient un identifiant unique. L'administrateur contient une table de correspondance entre l'id et l'ip. En cas de problème, un smartphone peut redemander l'id à l'administrateur.

On avait pensé à associer à chaque individu (type BumpFriend) un numéro aléatoire dans une table de hachage. On a aussi commencé une interface graphique sous forme de tableau pour l'ordinateur, qui en est à l'état de test mais auxquels on pourra rajouter les différents attributs comme la couleur et l'accomplissement de la mission plus tard.

4. Avancées diverses

- Implémentation d'un récepteur de SMS : quand l'utilisateur reçoit un SMS, le bracelet s'allume dans une certaine couleur et clignote pendant 5 secondes. Ainsi, malgré le bruit ambiant, un stimulus visuel attire son attention.

- Implémentation d'appel : quand le téléphone reçoit un appel, de la même manière que pour les SMS, le bracelet s'allume d'une certaine manière et clignote.

5. Module programmation Android : interface graphique

L'essentiel des services intérieurs avaient été programmé pour le PAN2.

Les activités sont organisées selon l'architecture suivante:

- **Connexion** : taper le pseudo et appuyer sur le bouton

|
|
V

- **Menu principal** : le format était une listview mais pour plus de liberté graphique et sachant que le nombre d'item sur cette activité est fixe, nous avons changé pour mettre une succession de boutons.

|
V

- **Liste des BF**: la taille de la liste pouvant changer, on utilise une listview.

|
V

- **Messagerie(envoi)**: taper le message et appuyer sur le bouton envoyer.

- **Choix couleur**: un bouton par couleur. Appuyer sur le bouton envoie un message au module bluetooth de l'application

- **About us** : lance une simple activité avec du texte.

- **Messagerie(récéption)**: Notification android qui ouvre une activité affichant le message.

Les modifications par rapport au pan2 dans l'interface graphique sont le fond d'écran et les boutons. Pour cela, il a fallu utiliser le format xml.

Pour modifier un bouton selon une image, il faut:

- Copier l'image dans le dossier drawable-hdpi
- Créer un dossier drawable qui contiendra les modèles de bouton

- Dans ce dossier, créer un fichier xml selector
- Rajouter dans la balise selector deux balises item : elles auront pour attributs l'état du bouton (appuyé ou pas) et l'image que l'on veut lui donner dans chaque état.
- Dans le fichier xml correspondant à l'activité : rajouter dans la balise button un attribut android:background qui aura pour valeur @drawable/nom_du_bouton

Pour modifier le fond d'écran et mettre une image, il faut :

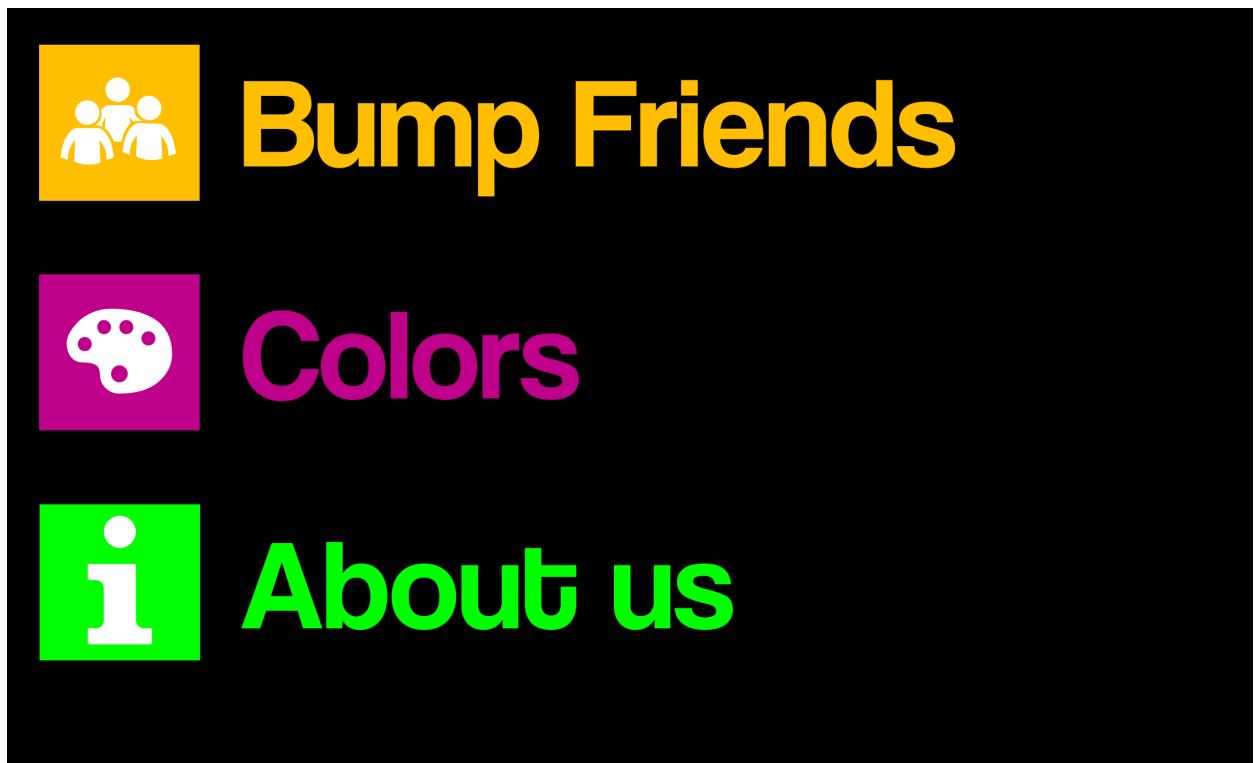
- Copier l'image dans le dossier drawable-hdpi (qui contiendra toutes les images utilisées par l'application)
- Dans le fichier xml correspondant à l'activité : dans la balise principale, rajouter un attribut android:background qui aura pour valeur @drawable/nom_image

Pour placer les boutons:

Dans le menu principal, pour une plus grande liberté de placement nous avons utilisé un absolut layout qui permet de placer le bouton où on veut. Nous avons ensuite joué sur le format xml et sur les attribut des boutons qui correspondaient à leur tailles et à leur positions pour une parfaite précision. (layout_x layout_y layout_height layout_width avec pour unité le dp)

Dans les activités qui ne contiennent qu'un seul bouton, nous utilisons plutôt un frame layout et choisissons de positionner le bouton centré en bas, il n'est pas possible de modifier sa position, mais elle est plus précise (bien centrée).

Design de l'interface graphique du menu :



Toute l'application suit la même charte graphique : Coloré sur fond noir

6. Module Business Model:

Direction générale du projet:

Entre le PAN 2 et le PAN 3 nous avons réellement intégré que le concept le plus valorisable dans notre projet est celui de **réseau social éphémère de proximité**.

Ce secteur est en plein développement et offre aujourd'hui de nombreuses opportunités.
Notre concept repose sur les points suivants:

-créer des réseaux sociaux ayant pour seul but **faciliter les relations physiques**. Le contact numérique n'est plus une fin en soi. Les contacts sont donc les personnes à proximité immédiate. Les "amitiés" sont provisoires et la liste s'adapte en permanence.

-réduire l'empreinte numérique au maximum , pour contourner les problématiques actuelles portant sur la vie privée.

Pour l'entreprise c'est un réel choix stratégique.

En effet, si la confidentialité est un réel argument pour l'utilisateur, on se prive d'un bien très précieux, à savoir les informations sur les clients, qui constituent la richesse principale des réseaux sociaux actuels tels que facebook.

-Provoquer des rencontres autour de nous. Si les reseaux sociaux classiques permettent de rencontrer des personnes à l'autre bout du monde, cela se concrétise presque jamais par un réel face à face. Notre valeur ajouté est de provoquer des rencontres, mais avec un premier contact toujours physique.

(par exemple le flash-mob dans un contexte festif)

Nous développons donc un "réseau social de test", dont **le concept est réellement novateur sur la marché**.

Avec Bump Band, nous nous sommes lancés sur le marché de l'évenementiel nocturne. Mais nous sommes persuadés de pouvoir exporter notre concept sur **d'autres secteurs, tels que les croisières, les resort hôtels ou les séminaires**, qui seront sans doute plus stables et plus lucratifs.

Livrables du module:

1) La vidéo commerciale sera inspiré des formats que l'on peut trouver sur la fameuse plateforme de crowdfunding [kickstarter](#).

Elle sera divisée en trois:

-On tâchera dans un premier temps de mettre en lumière les points soulevés dans la première partie de ce document. Le contenu étant relativement conceptuel, nous avons choisi de l'illustrer par des animations. Les icônes ont été pour beaucoup trouvées sur thenounproject.com et sont animées sur le logiciel wimeo.

-Une seconde partie présentera les fonctionnalités du Bump Band. Comme le prototype ressemble pour l'instant plus à un oscilloscope éventré qu'à un bracelet, nous avons choisi de réaliser le film par succession de photos, ce qui permet de faire changer de couleur un bracelet inerte.

-Enfin, une dernière partie, qui sort d'un cadre promotionnel classique, montrera la bonne ambiance de travail qui règne au sein du groupe 4.1. Cette partie comporte un réel intérêt pédagogique, car des extraits vidéos du travail au quotidien permettent de créer un sentiment de proximité avec le spectateur. Cela est bénéfique pour obtenir un financement par crowdfunding.

2) Business model:

Il s'agit du Business model des bracelets dans le cas d'un procédé de vente en B2B au monde de l'évènementiel (discothèques). Nous avons cependant pris conscience que ce marché est le

moins porteur pour notre bracelet. Par conséquent nous nous concentrerons lors du prochain pan sur le Business Model des marchés: Bateau de croisière, mariages, séminaires, etc.

I. Coût d'un bracelet :

Carte arduino : 7euros

Carte Bluetooth : 18 euros

Batterie 10 euros

Infrarouge + reste : 2 euros

prix à l'unité des composants: 37 euros

+ Cout de production du bracelet : environ 3 euros (comparaison au prix d'autres bracelets et de la PS4...)

Coût de fabrication d'un Bracelet : 40 euros

Conséquences :

- B2C a priori impossible. Dans le sondage, les consommateurs avaient dits être prêts à dépenser 25 euros.
 - Vente aux boîtes de nuit / évènementiel :

II. Projet:

Location des bracelets à la boîte de nuit:

Hypothèse de départ: entrée à 15 euros.

coque en caoutchouc pour garantir la solidité du bracelet.

700 personnes dans la boîte

Sur un site de prévente, on vend des préventes pour la soirée à 18 euros après un accord avec la boite de nuit: on gagne 3 euros par bracelets, et cela ne coutent rien aux organisateurs. A cela s'ajoute une caution électronique de 10 euros (prix à évaluer précisément). De plus, on met à disposition de la boite 100 ou 200 bracelets que les gens peuvent louer en entrant dans la boite. Prix alors plus cher, 4.50 euros + caution de 10 euros.

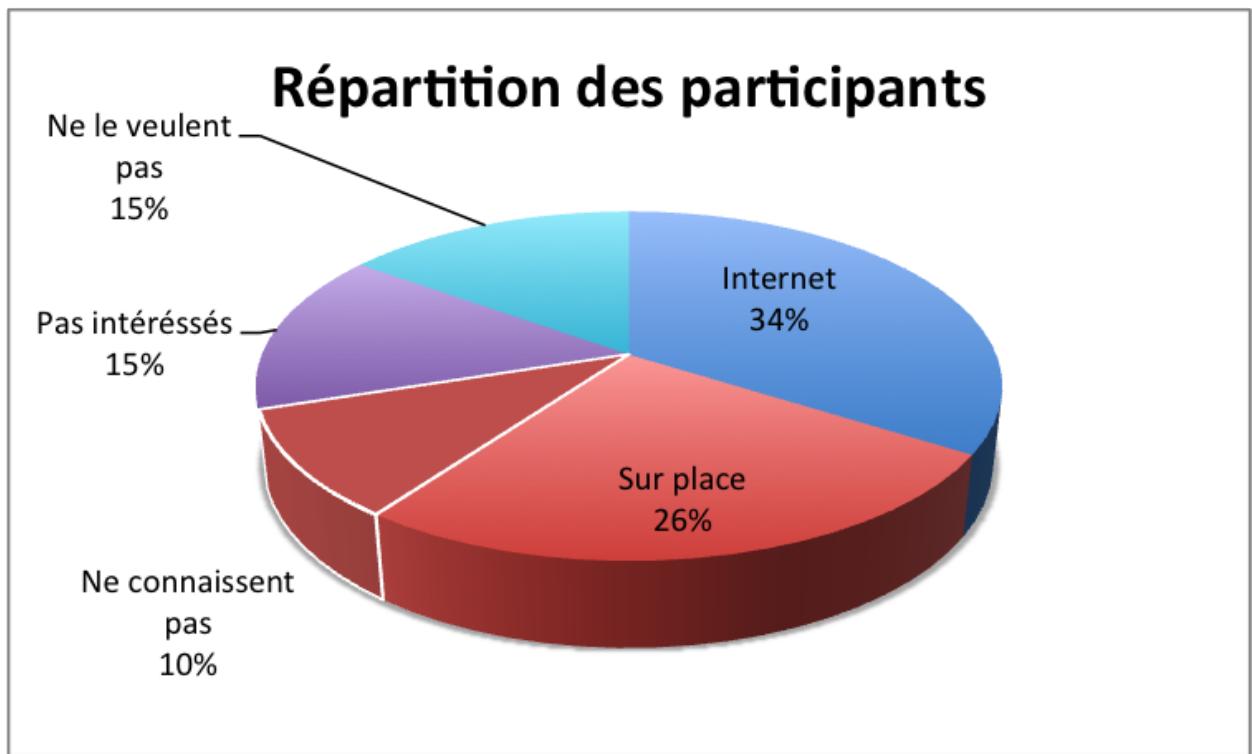
A ceci pourrait s'ajouter des revenus grâce aux taxis. On les néglige pour l'instant car pas de données exploitables.

Eviter les vols: Les clients doivent rendre le bracelet au vestiaire en échange de leur manteau et de leur caution électronique (bump sur la borne).

Revenu approximatif:

On se base sur les hypothèses de notre sondage: 85% des gens sont prêt à en prendre un. Sur ces 85%, on suppose (car 40% des préventes sont prises sur internet) que: 34% des gens allant à la soirée prennent un bracelet sur internet

Revenus: (sur 700 entrées en boîte): 490 bracelets dans la boîte. 70 % des gens.



$$R = 714 \text{ (préventes)} + 1134 \text{ (sur place)} - \text{Destructions}$$

Destruction + vols : 1% (hypothèse) = 4.9 bracelets = 196 euros - 49 euros (cautions) = 147 euros

Pas de fiscalité tant qu'on est en perte nette.

1ère année: CAPEX: 40 000 euros

WAAC: 4200 euros

PUB: CAPEXPUB: 2000 euros

Salaires: 0 euros

Logistique/stock: 2000 euros

Investissement RnD (telecom) 3000 euros

2ème année et plus: WAAC: 12 780 euros d'après lafinancepourtous.com

Salaires: 30 000 euros (commercial)

16 300 euros (smic)

+ frais de transport: 800 euros de frais de transports par mois -> 9 600 par an

Pub: 10 000 euros

Logistique et stock: 4 000 euros (location d'une box type Rue Barrault)

Revenus: Variable: 1700 euros par soirée, 6 max par semaine

Scénario optimiste: 1 le samedi soir, 1 le vendredi, 4 ou 5 en semaine.

Scénario pessimiste: 1 le samedi soir, 0,5 le vendredi et 1 ou 2 en semaine

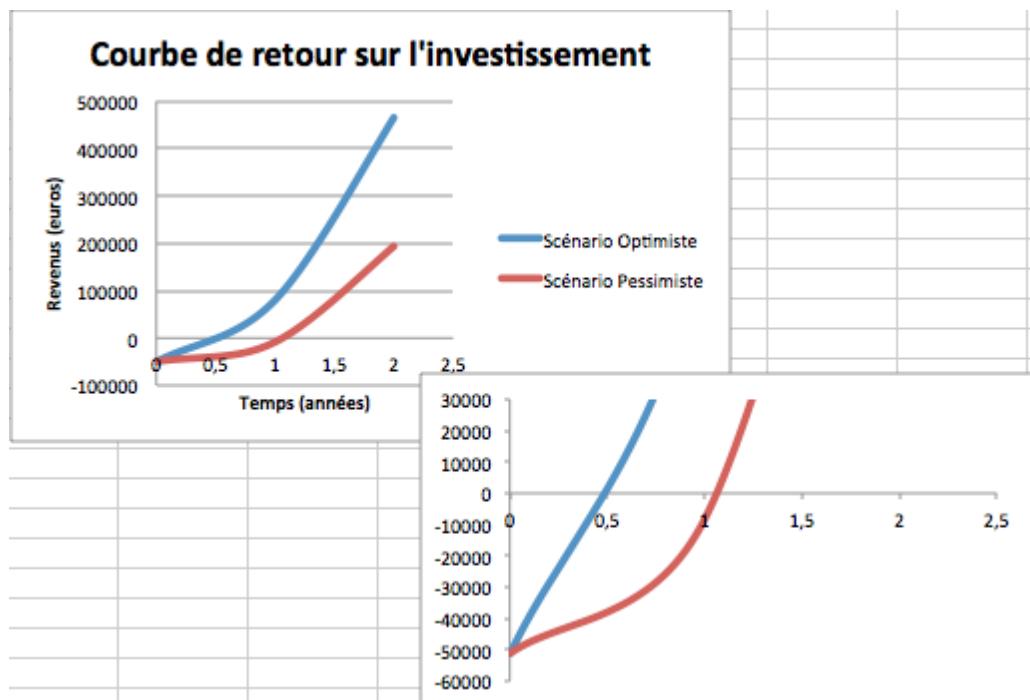
Samedi/Vendredi: 1700 euros

Semaine: 900 euros

TVA: 20%

O2: 4 750 euros par semaine: 247 000 par an

P2: 2625 euros, 136 500 par an



AVANCEMENT PAN 4:

Concernant la vidéo, nous avons réalisé le script et nous allons la tourner entre **lundi matin et mercredi après midi**. Elle se découpe en trois parties distinctes:

Tout d'abord, une explication du concept (environ 45s), faite par un orateur dans la nouvelle salle numérique (mur d'images) du CRDN. Nous la tournons **lundi matin de 10h à 11h**.

Ensuite, une présentation du bracelet. Nous la faisons également au CRDN, au même moment. (durée: 30s).

Enfin, une présentation des fonctionnalités. Nous souhaitons montrer chaque fonctionnalité du bracelet dans son contexte. Par conséquent, **mercredi après midi**, nous allons nous déplacer dans Paris afin de tourner de brèves séquences vidéo à plusieurs endroits: Parc des Expositions, Palais des Congrès, Hotel, Eglise, etc. Nous mettrons ensuite l'ensemble en forme entre **mercredi soir et jeudi matin** afin de vous envoyer la vidéo avant **jeudi midi**.

Nous avons également décidé de réaliser un site web afin de présenter le BumpBand. Je le réaliserais la semaine prochaine et il sera opérationnel **jeudi** matin. A priori l'adresse sera www.bumpband.com. La mise en page sera semblable à celle d'un site que j'ai réalisé cette année: www.lesbigdata.com

Concernant le Business Model, nous nous sommes focaliser entre les PAN 3 et 4 sur l'exportation de notre concept à d'autres secteurs, tels que les mariages, les croisières, les resort hôtels ou les séminaires, qui seront sans doute plus stables et plus lucratifs. Par conséquent, nous n'avons pas refait de Business Model mais nous avons interrogé des entrepreneurs et des salariés de ces secteurs afin de s'interroger sur la viabilité de notre projet dans les domaines des mariages, croisières et resort hôtels. Quant aux séminaires, nous ne savions pas à qui nous adresser, par conséquent nous n'avons pas d'information, mais les réunions que nous avons eu avec vous au cours de l'année nous font croire que le Bumpband pourrait marcher dans ce domaine.

Mariages:

Lorsque l'on organise un mariage, on travaille souvent avec la même entreprise tout le long, qui comporte à la fois une salle des fêtes, un hotel et un traiteur. C'est à cette entreprise qu'il nous faudrait louer les bracelets. Il peut alors proposer aux fiancés d'opter pour la location des Bumpband.

Nous avons contactés Dominique Denis, qui possède ce type d'entreprise à Anger. Il se dit prêt à proposer ces bracelets aux clients si notre projet voit le jour.

Croisières:

Nous avons contacté un conseillé de vente, Eric COIA, de Costa Croisières (l'une des compagnies de croisière les plus grandes d'Europe). Il souhaite qu'on lui transfère la vidéo du projet une fois terminée. Il se dit "pourquoi pas intéressé" par ces bracelets sur le Costa Pacifica, un bateau qui est orienté sur l'aspect festif du voyage.

Resort Hotel:

Nous avons contactés Laurianne Payen, qui travaille dans un Club Med. Elle voit une vraie viabilité du projet dans les Clubs Med pour adultes. Cependant elle ne pense pas que cela marcherait dans un Club Med Famille.

Ces trois contacts très différents montrent la viabilité de notre projet dans d'autres secteurs que celui des boites de nuits. Ces secteurs sont d'ailleurs sûrement plus lucratifs que le marché nocturne.

Nous avons ainsi pris le Business Model "par le pôle Nord": nous nous sommes au départ focalisé sur le marché de l'évènementiel, alors que les secteurs les plus intéressants et les plus lucratifs ont été laissés sur le côté dans un premier temps. Cependant, comme nous avons gardé

une ligne directrice de notre bracelet très général, il est très simple de l'exporter aux domaines les plus lucratifs.

A l'avenir, il faudra donc se rappeler de deux choses:

Ne pas se focaliser sur un domaine en particulier lors d'une étude de marché mais plutôt voir comment l'appliquer à d'autres domaines.

Garder une ligne directrice très générale afin de pouvoir facilement changer de client si des contraintes s'imposent.

Descriptif des classes :

Admin [Module Admin / protocoles de communication]

Package admin :

Ce package contient des informations spécifiques à l'administrateur

Banque

Cette classe va jouer le rôle de banque. Les crédits de chaque individus y sont stockés. Pour cela on utilise un HashTable qui associe à une adresse ip le nombre de crédits (on suppose que l'ip reste fixe). On dispose ensuite de fonctions pour ajouter un client (create), retirer un client (remove), obtenir son solde (getMoney), lui ajouter de l'argent (addMoney) et lui en retirer (removeMoney). Quand on retire de l'agent, on vérifie qu'il y en a assez. On renvoie 0 si c'est le cas, -1 sinon.

Cette classe possède toutes ses méthodes et attributs statiques afin de pouvoir y accéder depuis n'importe quelle classe. De plus, elle est unique, ce qui ne pose pas de problème.

Boisson [Protocole de communication]

Cette classe va représenter une boisson. Pour cela, elle contiendra ses caractéristiques : son nom, prix, code couleur et degré d'alcool (pour pouvoir faire des graphes de consommation). Elle implémente l'interface Transmissible afin de montrer que c'est un objet que l'on peut envoyer. Elle possède les getters habituels. Lorsque l'administrateur reçoit un objet Boisson, il considère qu'il s'agit d'une commande et ajoute une commande à la liste. Avant cela, il vérifie que le client a assez d'argent et n'a pas déjà commandé. La fonction toBytes code l'envoi (voir protocoles de communication).

Commande

Cette classe va gérer les commandes au bar. Pour cela, elle dispose de trois attributs : la liste des commandes (listeCommande) qui va contenir les ips des personnes qui ont passé une commande, et deux HashMap qui vont lier cette ip à la boisson et à l'état de préparation correspondant. Il existe deux états : en attente (false) et préparée (true).
Ensuite, nous avons comme pour la banque tous les attributs et les méthodes en statique. Nous disposons d'une méthode add qui permet d'ajouter une boisson à la liste des commandes. Cette fonction empêche un individu de faire plusieurs commandes. Elle renvoie 0 si la commande est confirmée, -1 sinon.

La fonction remove permet de supprimer la commande quand le client vient la chercher. Elle lui

envoie un message de confirmation et retire les credits correspondants à son compte en banque.

La méthode prepare permet de changer l'état de la commande.

La méthode enCommande renvoie si l'ip a une commande en cours ou pas.

FlashMob

Cette classe va permettre de gérer le flashMob, le jeu où tout le monde doit retrouver la personne qui lui a été associée.

Tout comme Banque et Commande, les attributs et méthodes sont statiques.

Elle possède deux attributs : une ArrayList de couleurs et une ArrayList de BumpFriend. En fait, deux clients vont devoir se retrouver s'ils sont côté à côté par indice pair/impair dans la liste.

Ensuite, pour connaître leur couleur associée, il suffit de diviser par deux leurs indices.

La méthode initialiser va faire débuter le jeu. Pour cela, elle récupère la liste des bumpFriends et les classe de manière aléatoire. Elle génère en même temps des couleurs aléatoires pour chaque couple.

La méthode vérification vérifie si le couplage est bien effectué.

Menu [Protocoles de communication]

Cette classe va permettre de transmettre le menu que va pouvoir utiliser le client. Il est transmis la première fois après la synchronisation initiale.

Elle possède un seul attribut : la liste des boissons dans le menu. Du côté de l'admin, les méthodes et attributs sont statiques afin de pouvoir les modifier à tout moment.

On a trois méthodes : une pour ajouter et l'autre pour enlever une boisson. Après cette opération, il est important de mettre à jour les Menus. C'est à cela que sert la méthode maj.

Vestiaire

Cette classe permet de gérer le vestiaire. Là encore, la plupart des attributs et méthodes sont statiques.

Tout d'abord, il contient une hashmap associant à chaque ip son numéro de vestiaire. Ensuite, l'attribut compt avance à chaque entrée au vestiaire, permettant ainsi de ne pas avoir deux numéros identiques.

Quand on identifie une interaction avec le vestiaire (pas de commande de boisson et ce n'est pas le premier bump), la méthode TraitementVestiaire est appelée. Elle commence par vérifier si le client est déjà dans le vestiaire. Si c'est le cas, cela signifie qu'il souhaite retirer son vêtement. Un message apparaît alors à l'écran, demandant à l'admin de confirmer ce retrait. Un message de confirmation est alors envoyé au client selon le choix de l'admin. Un scénario identique se produit lors du dépôt d'un vêtement.

Les méthodes afficheEntree et afficheSortie permettent d'afficher à l'écran le numéro de vestiaire et propose un choix à l'admin, qui n'a qu'à cliquer sur un des trois boutons.

[Package bracelet](#)

Ce package contient ce qui est nécessaire pour communiquer avec la carte arduino et les Verrous utilisés pour contrôler les données partagées et les points de rendez vous.

[SerialTest \[Protocoles de communication\]](#)

Cette classe a été récupérée sur le site de arduino et permet de communiquer avec ce dernier. Quelques attributs ont été ajoutés. La méthode envoieBF() exécute une tâche après la réception d'une information de la part de l'arduino. Pour commencer, on commence par vérifier si la personne est déjà dans la liste d'ami. Si ce n'est pas le cas, on vérifie si elle vient récupérer une boisson. Sinon, c'est qu'elle vient pour le vestiaire. Si elle n'est pas dans la liste d'ami, on l'ajoute. On suit le protocole pour les opérations qui suivent.

Des sémaphores sont implémentés dans cette fonction afin d'avoir des points de rendez-vous avec d'autres Threads et protéger des ressources partagées.

La méthode getIpAddr sert à obtenir l'ip de l'ordinateur.

[Verrous \[Protocoles de communication\]](#)

Cette classe contient les verrous et sémaphores utilisés dans le projet. Les Lock (Verrous) protègent des données partagées et les sémaphores servent à implémenter des points de rendez-vous.

[Package Client – Classe Destinataire \[Protocoles de communication\]](#)

Ici, on programme la partie client. La méthode envoieObjet permet d'envoyer un objet au destinataire. Pour commencer, on ouvre un nouveau Thread afin de ne pas bloquer le reste du programme. Ensuite on crée une nouvelle socket et on ouvre les flux d'input et d'output pour pouvoir communiquer. Ensuite l'échange se fait en échangeant les tableaux de bytes à chaque fois. On vérifie qu'une fin de transmission symbolisée par un objet Transmission ne passe pas. Si c'est le cas, on sait qu'on a affaire à la dernière communication.

[Package action \[Protocoles de communication\]](#)

Dans ce package sont regroupées les classes qui entrent en jeu dans la communication et donc

représentent les différentes actions possibles.

BFList

Cette classe permet de gérer l'enregistrement de la liste des clients dans un fichier texte et d'y accéder à n'importe quel moment. Quand on crée une instance de cette classe, on lui donne le nom du fichier texte à utiliser. La méthode ajoutBF permet d'ajouter un client à la liste. On commence d'abord par vérifier que le client n'est pas déjà dans la liste puis on l'ajoute. La méthode index renvoie l'index d'un bumpFriend dans la liste des bumpfriends. La fonction isBF est surchargée. Quand on lui donne un bumpfriend, elle utilise la méthode index pour dire si oui ou non il est vraiment un Bumpfriend. Quand on lui donne une ip, comme index, elle parcourt la liste des bumpfriend jusqu'à trouver (ou pas) l'ip. La méthode getBFListe renvoie la liste des bumpfriends. Pour cela, on parcourt le fichier et on décrypte ce que l'on a écrit. La méthode effacerBF est aussi surchargée. En fait, elle se ramène à effacer un ligne. Pour cela, on utilise un fichier temporaire.

BumpFriend [Protocoles de communication]

Cette classe va permettre de s'échanger les premières informations nous concernant juste après le bump. Elle a besoin de deux informations de base : le nom et l'ip du bumpfriend. Cette classe (comme presque toutes celles qui vont suivre) implémente l'interface Trasmissible. On y retrouve donc les deux méthodes execute et toByte.

La méthode execute commence par un point de rendez-vous. En effet, il faut pour effectuer cette méthode que la méthode envoieBF de SerialTest soit terminée. Ensuite, on effectue une série de vérifications qui vont nous permettre de confirmer l'identité du bumpFriend.

La méthode toString est utilisée pour transformer un BumpFriend en String afin de l'enregistrer dans le fichier texte dans la classe BFList

Color [Protocoles de communication]

Cette classe permet de modéliser une couleur. Elle n'a pas beaucoup d'utilité côté communication pour l'admin.

Connexion [Protocoles de communication]

Cette classe est la première échangée après le bump. Elle permet d'effectuer quelques vérifications sur le bump. Nous avons une fonction verifyCorrespond qui fait toutes les vérifications. Dans la méthode execute, on a aussi un point de rendez-vous car là encore, il faut attendre que la fonction envoieBF dans SerialTest soit finie. Après les vérifications, on est d'accord pour envoyer notre fiche d'identité.

ErreurTransmission [Protocoles de communication]

On retrouve ici une série d'erreur que l'on peut envoyer si jamais il y a un problème lors de la communication.

FMConfirmation [Protocoles de communication]

Cette classe permet de vérifier que le FlashMob c'est bien réalisé avec la bonne personne. Si c'est le cas on lui attribue un certain nombre de crédit et on lui signale son gain. Sinon, on lui indique qu'il y a un problème.

HashList [Protocoles de communication]

Cette classe permet de gérer l'attribution des identifiants. Elle contient une hashMap entre les bumpFriends et les identifiants. La méthode add génère un identifiant aléatoirement et l'attache à un bumpfriend.

Identifiant [Protocoles de communication]

Cette classe permet de transmettre l'identifiant à une personne. Elle ne sert à rien d'autre du côté de l'admin pour l'instant. On peut y mettre une demande d'ip correspondant à un identifiant.

IdentifiantAnswer [Protocoles de communication]

Cette classe permet de répondre à une demande d'ip en fonction d'un identifiant. Elle n'est pas utilisée dans notre prototype.

Message [Protocoles de communication]

Cette classe permet d'envoyer des messages en utilisant un système de messagerie. L'admin n'est pas censé recevoir de message. Nous envoyons aussi l'expéditeur.

Money [Protocoles de communication]

Cette classe répond à une RequestMoney (Demande d'argent). Elle envoie au client le nombre de crédit qu'il lui reste.

RequestMoney [Protocoles de communication]

Cette classe est celle utilisée quand on demande notre solde. L'admin répond en envoyant ce solde via la classe précédente.

Transmissible [Protocoles de communication]

Cette interface caractérise les objets que l'on peut transmettre. Deux méthodes doivent impérativement être implémentées : une méthode execute (qui est un onReceive) et une méthode toBytes. Execute explique ce qui se passe quand on reçoit l'objet. La méthode toBytes transforme l'objet en tableau de bytes afin de pouvoir le transmettre.

Transmission [Protocole de communication]

Cette classe permet de clôturer les communications. Elle transmet si la conversation s'est bien passée et si ce n'est pas le cas quel problème est intervenu.

Package Serveur [Protocole de communication]

Dans ce package, la partie serveur de l'admin est implémentée. On y retrouve aussi le parseur des connexions wifi.

EcoutConnexion [Protocole de communication]

Cette classe étend la classe Thread. Ainsi, on va pouvoir écouter les connexions sans bloquer le reste de l'application. La méthode run est une boucle infinie qui dès qu'un client vient se connecter au serveur, appelle la classe TraitementClient.

TraitementClient [Protocole de communication]

Comme son nom l'indique, cette classe va traiter les clients qui viennent se connecter. Elle étend la classe Thread afin de ne pas bloquer l'écoute des clients.

On commence ensuite par écouter ce qui arrive. Ensuite, tout ce passe comme dans la classe Destinataire.

Parseur [Protocole de communication]

Cette classe permet de décoder le tableau de byte reçu. La méthode paser lit le premier byte et en déduit l'objet auquel elle a affaire. Ainsi, elle appelle la bonne fonction pour décoder le reste du message. Pour chaque objet que l'on peut recevoir, il y a donc la méthode associée pour la décoder.

Package graphique

Ce package correspond à l'interface graphique en Java pour l'ordianteur de l'admin

BarModel

Cette classe implémente une JTable utilisé par FenetreBar, modèle de tableau pour gérer la liste des commandes dont les caractéristiques (en colonnes) sont le nom du client, la boisson demandée, et si l'état de la commande. La liste est actualisable.

FenetreMessage

Fenêtre avec un champ de texte et un bouton Envoyer pour envoyer des messages.

FenetreTableau

Lance une fenêtre avec une liste de BumpFriend, implémenté à partir de TableModel.

FlashMobFenetre

Fenêtre où l'on peut entrer le nombre de crédits à gagner et lancer le jeu.

MainTest

Test pour pouvoir lancer la FlashMobFenetre et FenêtreMessage.

TableModel

Tableau de Test contenant des BumpFriends dont les colonnes sont l'adresse IP et le nom.

BoissonModel

Tableau des boissons utilisé par l'interface graphique MenuTableau pour afficher la liste des boissons avec leurs attributs.

MenuTableau

Lance la fenêtre graphique du Menu avec possibilité de créer de nouvelles boissons ou d'en supprimer.

9. Bibliographie:

Sites Web :

- <http://fr.openclassrooms.com/>
- <http://developer.android.com/>
- <http://docs.oracle.com/javase/tutorial/networking/>
- <http://www.cc.gatech.edu/fac/Russell.Clark/Classes/6250/PracticalSocketJava.pdf>
- <http://arduino.cc/en/Main/ArduinoBoardMini>
- <https://www.sparkfun.com/products/12580>
- <http://arduino.cc/en/Main/MiniUSB>
- http://www.evola.fr/product_info.php/lcd-8x2-retro-eclairage-bleu-p-83
- http://www.evola.fr/product_info.php/mini-moteur-vibration-p-90
- <http://www.digikey.com/catalog/en/partgroup/plcc-4-tri-color-rgb-round-with-flat-top-leds/37338>
- http://www.evola.fr/product_info.php/mini-interrupteur-bouton-poussoir-p-235
- <https://learn.sparkfun.com/tutorials/ir-communication/getting-started>
- <https://learn.sparkfun.com/tutorials/using-the-bluesmirf>

Livre :

- Programmation Android, W. frank Ableson, Robi Sen, Chris King, C. Enrique Ortiz

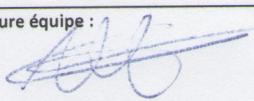
Annexe 1 : fiche d'identité du groupe:

Le groupe PACT4.1 s'est constitué, comme tous les autres groupes, par le hasard des algorithmes de Télécom Paristech. Mais le hasard a bien fait les choses. Les membres se complètent les uns les autres pour former une équipe solidaire, et solide dans tous les domaines. En effet, chacun a ses propres centres d'intérêts scolaires dans lesquels il est investi et pour lesquels il réussit à partager ses connaissances et sa passion au reste du groupe. En effet, ce dernier est composé de Damien Auffret (intérêt : électronique), Cédric Meston (aspect économique et social), Charles Deschard (communications sans fil, économie), Julien Romero (programmation), Arturo Castellanos (programmation), Félix Richart (informatique), Marc Faddoul (leadership et organisation), Alexandre Tadros (aspect économique et design). Cet éventail de préférences disciplinaires, allié à une idée de projet que chacun porte et en laquelle chaque membre se reconnaît, permet un travail en groupe efficace et une répartition des tâches facile à effectuer (bien que chaque membre soit déterminé à profiter de PACT pour s'intéresser aussi aux domaines qu'il ne maîtrise pas et a donc de lui-même l'envie de participer à la réalisation d'autres modules que celui de son domaine de prédilection).

Nos méthodes de travail en groupe se définissent en deux catégories : présence physique et travail à distance. Nous organisons au moins un rendez-vous hebdomadaire en plus des TH consacrées à PACT pour une mise au point collective des idées de chacun et des tâches à effectuer, et la répartition de celles-ci. Pour ce qui est du travail à distance, nous avons créé un groupe Facebook et un google drive, ce qui nous permet, d'une part de garder un historique de tout ce qui a été fait, de partager et modifier facilement (par chacun) les travaux de chacun, d'autre part de permettre au tuteur de se tenir précisément au courant de l'avancée du projet.

Annexe 2 : fiches modules:

Module SES

Suivi module PACT 2013-2014	Business Model - Rémi Maniak (SES - Sciences de Gestion)	
	Groupe	4.1.
	Projet	Réseau social de soirée
	Contact	projetpact4.1@gmail.com
	Contenu générique à réaliser	Contenu spécifique
Descriptif	Le module Business Model permet de se sensibiliser aux approches "tirées par le système client". Il s'agit à la fois de préciser la proposition de valeur et de la positionner par rapport à des utilisateurs (et potentiellement des clients) finaux, mais aussi d'expliquer l'ancrage de l'offre au sein d'un réseau d'acteurs plus large (partenaires, fournisseurs, collectivités,...)	
Objectifs d'apprentissage	Initiatition au positionnement d'une offre Initiation aux business models d'ecosystème	
Résultats attendus	Scénarios d'offre, parcours utilisateur Réseau de valeur intégrant l'ensemble des partenaires pertinents et les échanges qui les relient : flux financiers, flux de prescription, flux d'action / logistiques	
PAN1 :	Explicitation de un ou deux scénarios d'offre (client typique, contraintes sur la technique,...). "Pitch" du projet, explicitation des éléments qui, dans le projet, amènent à penser qu'il y a ici potentiellement une offre qui pourrait interesser plusieurs acteurs : l'utilisateur final, mais pas seulement.	Explicitier deux ou trois scénarios variés de fonctionnement et de valeur apportée pour l'utilisateur et les partenaires de l'offre (bar, taxis,...)
PAN2 :	Présentation des investigations poussées sur "qui pourrait gagner quoi?". Etude d'une population large de bénéficiaires potentiels : bauche du réseau de valeur sur sa partie "flux financiers".	Présenter les données receuillies (interviews,...) sur les utilisateurs et partenaires : usage, propension à payer, valeur perçue, économie de coût, difficultés levées, leviers nécessaires, modalités concrètes de partenariats
PAN3 :	Finalisation du réseau de valeur, éléments de chiffrage Storyboard du film sur le parcours utilisateur	
PAN4 :	Réseau de valeur finalisé. Film sur le parcours utilisateur.	
		Signature expert :
		
		Signature équipe :
		

Module – Arduino + Bluetooth

Projet PACT 4.1

Experts :

- Guillaume Duc - Tarik Graba

Périmètre du module :

Le module est centré sur un micro-contrôleur. Ce micro-contrôleur fait tourner de façon autonome un programme qui pilote une ou plusieurs LED RGB. Le programme sur le micro-contrôleur peut également dialoguer, grâce à un module Bluetooth, avec une application sur un appareil Android. Enfin, deux dispositifs doivent pouvoir échanger des informations entre-eux via une liaison filaire.

Matériel nécessaire :

- Carte Arduino Pro Mini x 2 - Module Bluetooth Mate Gold (à commander) x 2

Objectifs pédagogiques :

- Savoir ce qu'est un micro-contrôleur - Découvrir l'utilisation d'un module Bluetooth - Programmer un micro-contrôleur et utiliser ses entrées/sorties

Déroulement :

- PAN 1 (décembre) : Savoir ce qu'est un micro-contrôleur, quelles sont ses possibilités et ses limites
- PAN 2 (janvier) : Choix du module Bluetooth, Étude de sa documentation, Première mise en œuvre de la carte Arduino (mise en place de l'environnement de développement, communication avec la carte, utilisation des entrées/sorties)
- PAN3 (mars) : Module opérationnel

Élèves concernés : Charles Deschard, Damien Auffret, Marc Faddoul

Module Android - Interface graphique et services internes

Proposé par: Jean-Claude Dufourd (jean-claude.dufourd_at_telecom-paristech.fr, 7733, bureau DA413)

Réalisé par : groupe 4.1

Elèves : Félix Richart, Alexandre Tadros

Descriptif

L'utilisation de smartphones ou de tablettes est devenue depuis quelques années presque incontournable. Parmi les différents systèmes d'exploitation permettant de contrôler ces smartphones et tablettes, le système Android représente près de la moitié des périphériques. La programmation des smartphones ou tablettes Android fait appel au langage Java, vu lors du cours INF103. Elle ne devrait pas poser de difficulté majeure, néanmoins une des difficultés souvent rencontrées est la prise en main de l'environnement de développement et des spécificités des applications Android. Ce module a pour vocation d'aider les élèves à surmonter cette difficulté.

Ressources dont le module dépend

Le site de référence pour l'environnement de développement Android (bibliothèques, simulateur, environnement de développement, etc.) et pour l'auto-formation est le site <http://developer.android.com> , Transparents du Mini-cours Android.

Exemples d'utilisation du module:

- Application Android avec une interface graphique simple
- Application Android utilisant la caméra d'un téléphone mobile
- Application Android utilisant l'écran tactile d'une tablette

Objectifs d'apprentissage

- Informatique :

prendre en main l'environnement de développement Android, comprendre le fonctionnement d'une application Android, savoir lire et utiliser la documentation des API Android, savoir créer une application Android de base et utiliser certaines fonctionnalités avancées (affichage graphique, réseau, multimédia ...), savoir debugger une application.

Résultats attendus:

PAN1 :

- Démontrer que l'environnement de développement est en place et que des applications simples fonctionnent sur un émulateur.
- Expliquer le cycle de vie d'une application et la notion d'activité

PAN2 :

- Réaliser une application Android sur machine virtuelle mettant en œuvre le cycle de vie, l'enchaînement de plusieurs activités, et le lancement d'une autre application.
 - Expliquer le fonctionnement du stockage de données persistantes.
 - Réaliser une application spécifique. Cette application fonctionnera sur un smartphone ou une tablette et pourra mettre en œuvre.

PAN3 :

- Bluetooth/WiFi

PAN4 :

- Réaliser l'application finale
- Analyser comment le module est intégré dans le prototype, quelles pistes d'améliorations seraient à envisager (performance, simplicité)

Module Android - Outils de communication:

Proposé par: Jean-Claude Dufourd (jean-claude.dufourd_at_telecom-paristech.fr, 7733, bureau DA413)

Réalisé par: groupe 4.1

Elèves : Arturo Castellanos Salinas, Félix Richart, Julien Roméro

Descriptif:

L'utilisation de smartphones ou de tablettes est devenue depuis quelques années presque incontournable. Parmi les différents systèmes d'exploitation permettant de contrôler ces smartphones et tablettes, le système Android représente près de la moitié des périphériques. La programmation des smartphones ou tablettes Android fait appel au langage Java, vu lors du cours INF103. Elle ne devrait pas poser de difficulté majeure, néanmoins une des difficultés souvent rencontrées est la prise en main de l'environnement de développement et des spécificités des applications Android. Ce module a pour vocation d'aider les élèves à surmonter cette difficulté.

Ressources dont le module dépend

Le site de référence pour l'environnement de développement Android (bibliothèques, simulateur, environnement de développement, etc.) et pour l'auto-formation est le

site <http://developer.android.com> , Transparents du Mini-cours Android.

Exemples d'utilisation du module

- Application Android avec une interface graphique simple
- Application Android utilisant la caméra d'un téléphone mobile
- Application Android utilisant l'écran tactile d'une tablette

Objectifs d'apprentissage

- Informatique :

Prendre en main l'environnement de développement Android, comprendre le fonctionnement d'une application Android, savoir lire et utiliser la documentation des API Android, savoir créer une application Android de base et utiliser certaines fonctionnalités avancées (affichage graphique, réseau, multimédia ...), savoir debugger une application

Résultats attendus

PAN1 :

- Démontrer que l'environnement de développement est en place et que des applications simples fonctionnent sur un émulateur
- Expliquer le cycle de vie d'une application et la notion d'activité

PAN2 :

- Réaliser une application Android sur machine virtuelle mettant en œuvre le cycle de vie, l'enchaînement de plusieurs activités, et le lancement d'une autre application.
- Expliquer le fonctionnement du stockage de données persistantes.
- Réaliser une application spécifique. Cette application fonctionnera sur un smartphone ou une tablette et pourra mettre en œuvre.

PAN3 :

Bluetooth/WiFi

PAN4 :

- Réaliser l'application finale
- Analyser comment le module est intégré dans le prototype, quelles pistes d'améliorations seraient à envisager (performance, simplicité)

Module Programmation Concurrente et/ ou distribuée

Expert : Thomas Robert

Périmètre du module :

Le module est centré sur l'élaboration de protocoles de communication, afin de permettre aux différents éléments de notre projet de se comprendre. De plus, certaines communications devront être sécurisées. Les différentes interactions sont la communication bracelet-portable, portable-portable et portable-administrateur. Nous devrons alors imposer aux programmeurs la manière dont les informations doivent être envoyées.

Objectifs pedagogiques :

- Assimilation du vocabulaire associé.
- Être capable d'utiliser les bibliothèques Java/Android pour implémenter les threads et les communications.
- Création d'un protocole de communication, avec les contraintes associées.

Déroulement :

- PAN 1 (décembre) : Connaitre le protocole associé, et l'utiliser pour expliquer l'intérêt du module dans notre projet.
- PAN 2 (janvier) : identification des interfaces utilisées et des bibliothèques offrant le motif socket + socket serveur, code des premières connexions, définition du protocole de communication entre les différents agents.
- PAN 3 (mars) : code Java/Android réalisant le protocole.
- PAN 4 : Intégration du protocole au projet.

Elèves concernés

- RICHART Félix
- ROMERO Julien
- CASTELLANOS Arturo

Module administrateur

IHM basique qui gère vestiaire, commandes au bar (paiement), jeu (événement lancé par l'admin propagé au portable)
admin superuser, doit avoir le privilège de changer les couleurs au dessus des utilisateurs dans les jeux, il doit vérifier que les participants ont complété le jeu ou non (checké tous les autres)
pas dans le prototype allégé

pan 2: messages de l'admin et vers l'admin pris en compte dans le protocole

pan 3: simuler un demande de changement de couleur de l'admin qui a priorité sur les changements manuels de la personne sur le bracelet

pan 4: interface simple, jeu

module sécurité

(optionnel)

S♦curiser les paiements par cryptage de messages

Pour cela, contacter expert s♦curité♦ sur technologies simples, en mode consultation

Ensuite choix motiv♦ de la techno et impl♦mentation

Annexe 3 : Liste des modifications apportées au document (après PAN1):

Tableau des modifications de fond apportées au projet avec validation des experts et encadrant informatique

<u>Libellé / date</u>	<u>Description brève</u>	<u>Validé par :</u>
Business Model enrichi (01/14)	Développement des offres Réseau de valeurs Sondage Pistes de partenariats Parcours d'usage	R. MANIAK
Changement Bump en IR (01/14)	Le bump se fera désormais par IR courte portée	T. GRABA
Changement Architecture Informatique (01/14)	Les schémas ont été mis à jour pour plus de clarté. Administrateur mis sur un ordinateur	M. CHOLLET
Modification diagramme de Gant (12/13)	Le diagramme de Gant a été très légèrement revu. Les phases d'intégration pré PAN 3&4 sont nettement allongées. Définition de responsables sur les tâches (surtout hors module)	Jury PAN 1