Université de Bretagne Occidentale

Master 2 Informatique

Département Informatique

2020/2021

Mobiles et Objets Connectés

La Matrice Connecté

Auteur :
Maureen PERON

Auteur :
Timothé LANNUZEL

Auteur :
William PENSEC

25 janvier 2021

FACULTÉ
DES SCIENCES
& TECHNIQUES

Sommaire

I Introduction

Ce rapport décrit le travail effectué pour le projet avec en détail les architectures choisies.

Le projet consistait à la réalisation d'un jeu constitué d'un point se trouvant dans une grille et qui peut être déplacé via téléphones ou des commandes à distances. Le code réalisé peut etre retouvé sur Github : https ://github.com/WilliamPsc/MatriceConnectee_IoT

Vous trouverez dans le dossier racine du projet, le dossier :

- **ArduinoRPi** : comporte les deux fichiers de code Arduino nécessaire dans le projet.
- **ClientPC** : comporte un fichier nodeJS permettant de simuler une application Android sur l'activité de déplacement de la LED.
- **Episode1** : contient le code Java donné pour l'épisode 1 du projet permettant de commander la matrice sur windows avec des terminaux.
- **Episode2** : contient le code JavaScript (Node) de tout l'épisode 3 du projet.
- **MatriceConnectee** : contient le code de l'application Android (pour Android Version $>$ 6.0 normalement)
- **Rapport de Projet** : contient ce rapport ainsi que les images de base qui ont servi à le faire ainsi que le code LaTeX

II Travail effectué

II.1 Épisode 1

Partie 1 : La matrice (Java) – Local	Le travail a été effectué Fichiers Java: - MyMatrix.java pour la matrice - Main.java pour le contrôle Utilisation: Lancé les fichiers MyMatrix.java puis Main.java sur un ordinateur avec 2 terminaux
Partie 2 : La matrice (Java) – Réseau (Socket)	Le travail a été effectué Fichiers Java: - Main_Serveur.java pour la matrice serveur - Main_Client.java pour le contrôle client Utilisation: Lancer les fichiers Main_Serveur.java puis Main_Client.java sur un ordinateur avec 2 terminaux.
Partie 3 : La commande Android – Réseau (Socket)	Le travail a été effectué Fichiers Android: - MenuActivity.java pour le menu de l'application - DeplacementActivity.java pour le contrôle du point sur la grille sur Android Utilisation: Lancer le fichier Main_Serveur.java sur un ordinateur puis utiliser son téléphone pour contrôler.

II.2 Épisode 2

Partie 1 : Architecture Noeud (Java) – Local	Le travail a été effectué Fichier Java : - Noeud.java pour le noeud serveur - Main_Client.java pour les commandes - MainClientMatrice.java pour la matrice Utilisation: Lancer les fichiers Noeud.java puis Main_Client.java et enfin MainClientMatrice.java sur un ordinateur avec 3 terminaux.
Partie 2 : Commande Android / Matrice Android (Noeud Java)	Le travail a été effectué Fichier Android: - MyMatrix.java - VueMatrice.java Utilisation: Lancer le fichier Noeud.java sur ordinateur puis utiliser 2 téléphones avec un qui contrôle et l'autre qui regarde la matrice.
Partie 3 : Commande Android / Matrice Android (MQTT)	Le travail n'a pas été fait car il n'était plus question des MQTT dans la suite du sujet nous avons donc décidé de passer cette question au lieu de rester bloqué et de prendre du temps.
Partie 4 : Commande Android / Matrice Android (NodeJS)	Le travail n'a pas été fait car nous n'avions pas de serveur permettant d'envoyer les broadcasts
Partie 5 : Commande Android / Matrice JS (NodeJS)	Le travail n'a pas été fait pour la même raison que la partie précédente.

II.3 Épisode 3

Partie 1 : Commande d'une LED par le système (Raspberry PI)	Le travail a été effectué Raspberry Pas de fichier utilisé nous avons juste utilisé des lignes de commandes Utilisation: - "gpio mode 8 out" pour déclarer le pin - "gpio write 8 1" pour allumer la led - "gpio write 8 0" pour éteindre la led

Partie 2 : Commande d'une LED via le port série (Raspberry PI/Arduino)	Le travail a été effectué Raspberry Utilisation de minicom sur Raspberry sur le port "/dev/ttyAMA0" Fichier Arduino: - initArduino.ino qui permet d'allumer une LED sur le port 13 selon le caractère lu. Utilisation: Lancer minicom sur le Raspberry avec l'Arduino connecté au port "/dev/ttyAMA0", et entré sur minicom soit 1 pour allumer soit 0 pour éteindre la LED.
Partie 3 : Commande d'une LED par le port série/NodeJS (Raspberry PI/Arduino)	Le travail a été effectué Fichier Raspberry: - communication_RaspberryArduino.js code permettant d'envoyer 0 à l'Arduino - communication_RaspberryArduino2.js code permettant d'envoyer 1 à l'Arduino - communication_RaspberryArduino3.js code permettant d'envoyer 1 ou 0 à l'Arduino toutes les 1 secondes. Utilisation: Lancer un des programmes .js présenté ci-dessus sur Raspberry avec l'Arduino connecté au port "/dev/ttyAMA0"
Partie 4 : Commande d'une LED par le port série/Serveur NodeJS (Raspberry PI/Arduino)	Le travail a été effectué Fichier Raspberry: - commandeLES_partie4.js code permettant de gérer les communications entre l'Arduino et un téléphone Android. Fichier Android: - SwitchLED.java code permettant d'envoyer soit 1 ou 0 au serveur Raspberry. Utilisation: Lancer le programme commandeLES_partie4.js sur Raspberry avec l'Arduino connecté au port "/dev/ttyAMA0", et utiliser son téléphone pour allumer ou éteindre la led.

Partie 5 : Commande d'une LED par le port série/Clients/Serveur NodeJS (Raspberry PI/Arduino) Partie 6 : Commande d'une LED par le port série/Client/Serveur NodeJS (Raspberry PI/Arduino) et Client Android	Le travail a été effectué Fichier Raspberry: - commandeLES_partie5.js code permettant de gérer les communications entre l'Arduino et les messages reçus via le package net de NodeJS Fichier PC: Nous avons malheureusement perdu le fichier clientPC. Cependant pour vérifier les résultats de cette partie nous pouvons utiliser le code réalisé dans la partie 6 qui permet de remplacer ClientPC par l'application Android. Fichier Android: - SwitchLed.java code permettant d'envoyer soit 1 ou 0 au serveur. Le bouton Android a été désactivé pour la suite du projet mais il est possible de le réactiver sur l'activité activity_menu.xml Utilisation: Lancer le programme commandeLES_partie5.js sur Raspberry avec l'Arduino connecter au port "/dev/ttyAMA0", et utiliser son téléphone pour allumer ou éteindre la led.
Partie 7 : Commande d'une Matrice à LEDs par le port série/Client/Serveur NodeJS (Raspberry PI/Arduino) et Client Android	Le travail a été effectué Fichier Arduino: - matriceLED.ino code permettant d'afficher le point de la grille sur une matrice à LED Arduino fichier Raspberry - matrice.js code permet de gérer les communications entre l'Arduino et un téléphone Android. fichier Android: - DeplacementActivity.java pour le contrôle du point sur la grille sur Android Utilisation: Lancer matrice.js sur Raspberry avec l'Arduino connecté au port "/dev/ttyAMA0", et utiliser son téléphone soit pour afficher la matrice sur son téléphone soit pour commander la position du point.
Partie 8 : La sans-fil !	Le travail n'a pas été fait car lors de la configuration d'un second Xbee pour assigner le canal et l'ID. Le logiciel XCTU tournait en boucle sans s'arrêter.

II.4 Épisode 4

Le JEU	Cette partie n'a pas été commencé Nous n'avons pas eu le temps de faire cette partie mais l'idée de base est déjà créée avec les parties précédentes, il ne faudrait que modifier un peu tout pour prendre en compte deux joueurs et ajouter les parties de gestion de ces deux joueurs ainsi que les modifications d'interface graphique afin de prendre en compte les points et autres idées que l'on pourrait avoir.

II.5 Autres implémentations faites

Nous avons ajouté certaines fonctionnalités supplémentaires qui ont surtout servi à une meilleure ergonomie de l'application.

Nous pouvons cité par exemple la traduction complète de l'application en Anglais (US) et Français avec une possibilité d'ajout d'autres langues si nécessaire.

Nous avons également ajouté des messages s'affichant en bas de l'écran en cas de connexion bien réalisé ou mal réalisée et pareil pour la déconnexion.

Nous avons ajouté une activité de paramètres qui devait servir au départ à inscrire dans une box l'IP du serveur et les ports à utiliser mais nous n'avons pas eu le temps de finir cette partie qui est pourtant très proche de la fin !

Enfin, afin de mieux nous organiser dans notre travail, nous avons utilisé GitHub afin de travailler à plusieurs et à distance et également pour versionner le projet pour éviter des problèmes qui aurait pu survenir en cas d'écrasement d'un fichier fonctionnel et par la perte de son contenu. Pour utiliser GitHub nous avons utilisé le logiciel sur Windows GitKraken (https ://www.gitkraken.com/).

III Conclusion

Nous n'avons pas réussi à tout implémenter notamment dû aux conditions sanitaires actuelles et également à certains problèmes tel le XBEE qui ne se connectait

pas. Malgré cela, une très grosse partie du projet a été réalisé avec le détail des codes dans les différents sous-dossiers présents dans le dossier principal. Nous avons travaillé le côté fonctionnel mais aussi un peu le côté ergonomique en ajoutant des petites fonctionnalités bonus tel la traduction complète de l'application ou les ajouts des messages si la connexion s'est bien faite ou si la déconnexion s'est bien passée. Vous pouvez voir le résultat de l'application dans le dossier ou alors sinon dans la section Annexes qui suit il y a quelques screenshots de l'application sur un téléphone Samsung A40.

IV Annexes

Je met des screenshots de l'application Android avec quelques fonctionnalités actives au cas où elles ne fonctionneraient pas sur un autre appareil.

FIGURE 1 – Menu principal de l'application

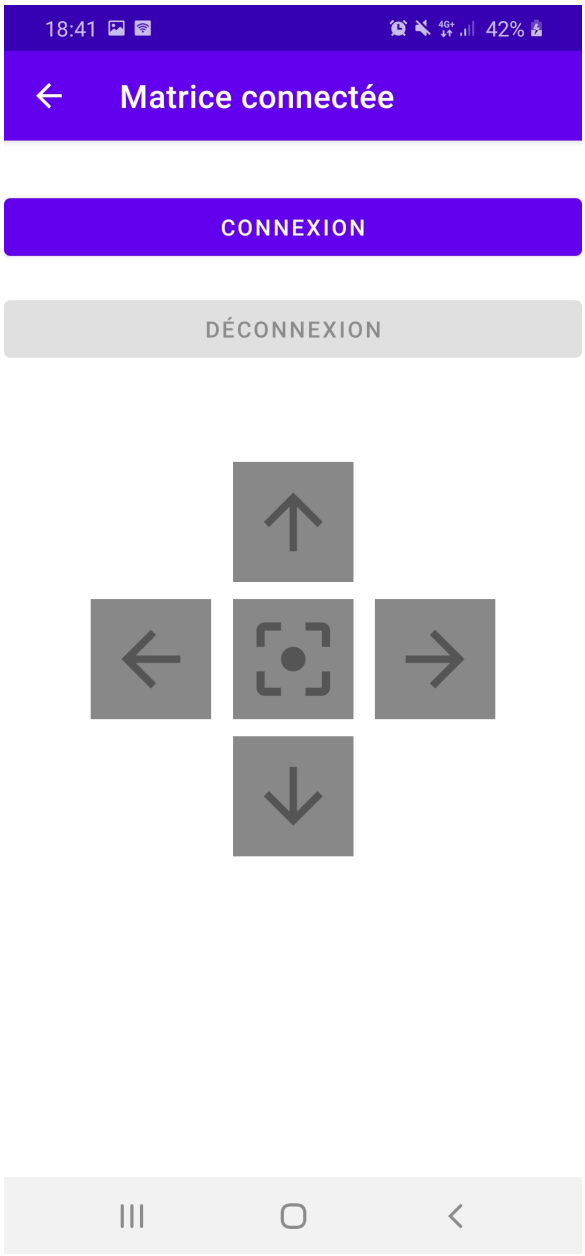

FIGURE 2 – Activité déplacement du point en non connecté

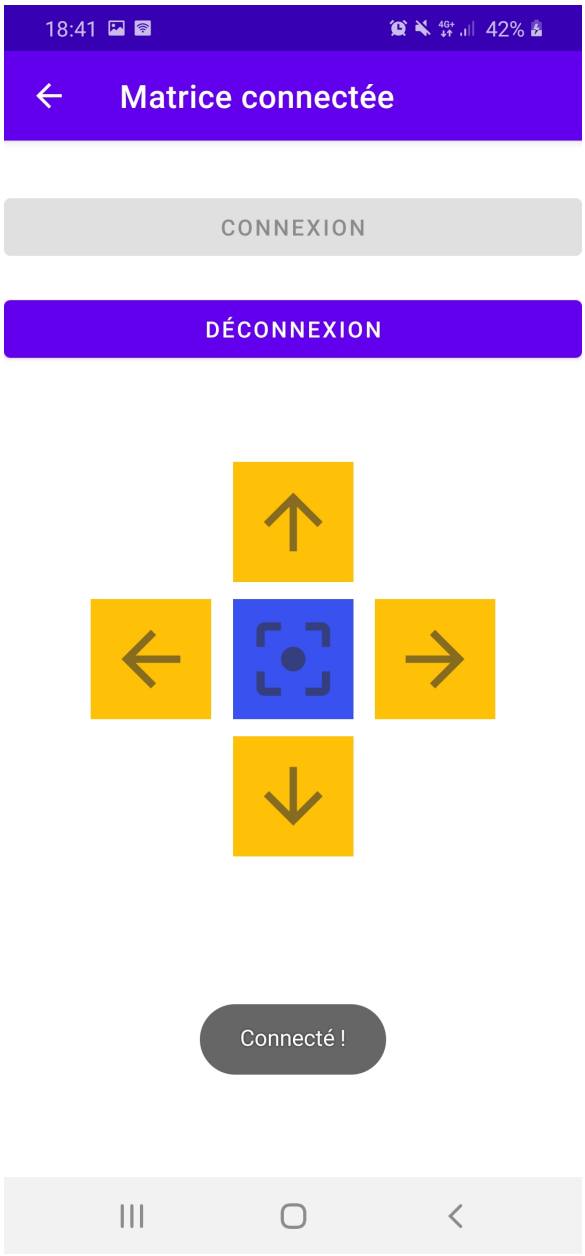

FIGURE 3 – Activité déplacement du point en connecté

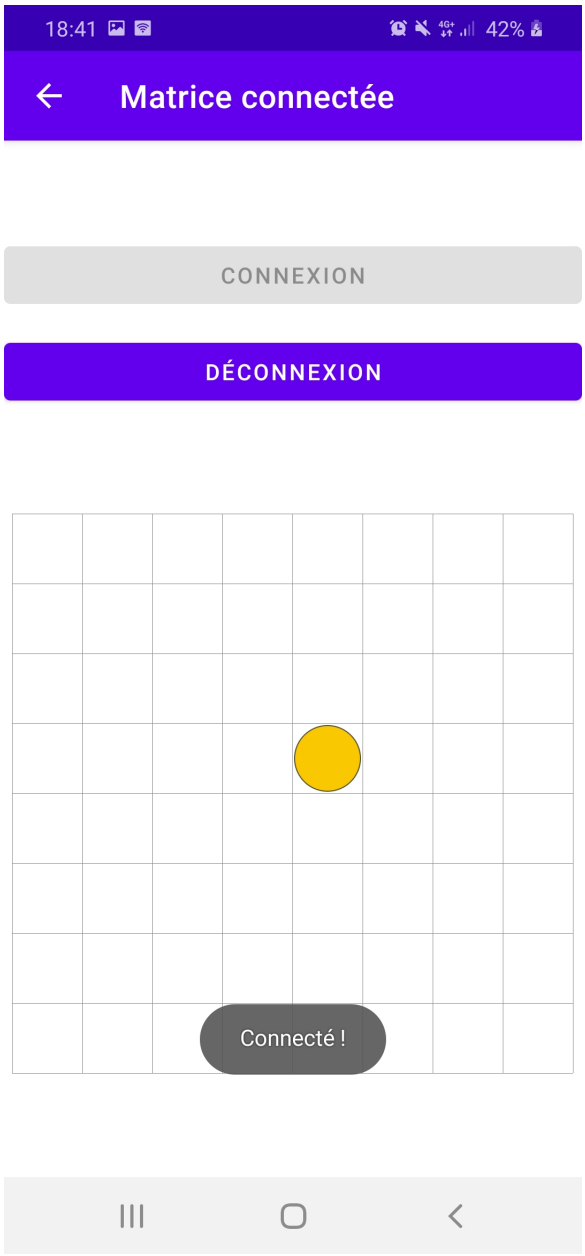

FIGURE 4 – Activité affichage de la matrice en connecté

FIGURE 5 – Activité affichage de la matrice en non connecté

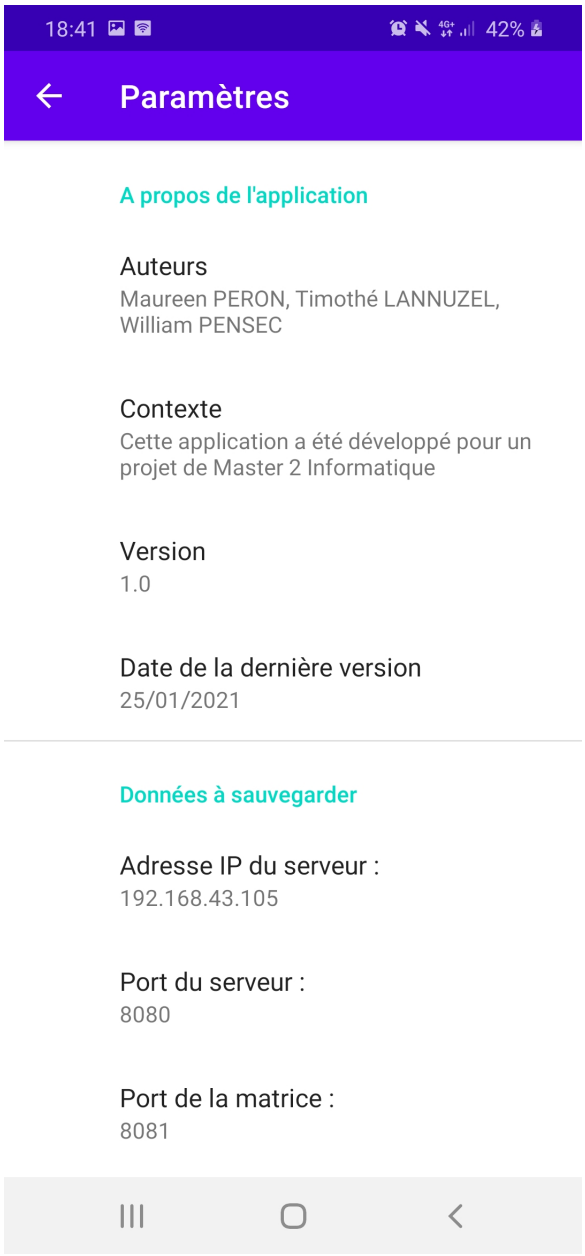

FIGURE 6 – Activité paramètres