RAPPORT
FREESTYLE SYSTEM ASSIST

Département Ingénierie des systèmes

AUBRY Alexandre , COURIVAUD Raphaël, DUPONCHEL Nicolas, NAMANE Sélim

Table des matières

Table des illustrations

INTRODUCTION

I. Préface

Nous sommes un groupe de 4 étudiants venant tous de CPGE. Après deux années d'études théoriques pures, nous avions la volonté commune de créer quelque chose de concret, mettant en application nos connaissances passées mais également celles que nous allions acquérir durant l'année. Nous cherchions donc des sujets de projets pouvant nous intéresser. La première idée nous ayant plu fut le suivi automatique d'un sportif par drone équipé d'une caméra.

Durant ces prémisses de réflexions, l'entreprise Parrot est venue à l'ESIEE afin de faire la promotion d'un concours (Parrot Awards) dont ils étaient les instigateurs. Cela nous a tout de suite intéressés et nous avons donc discuté avec des élèves diplômés de l'ESIEE qui y étaient salariés. Le but de ce concours était de créer un objet ou une application sur le thème des jeux et jouets connectés. Nous nous sommes donc bien évidemment lancés dans ce concours. Ce dernier avait l'avantage de nous offrir l'opportunité d'être aidés par un parrain appartenant à Parrot. C'est monsieur Guillaume Béranger, ingénieur chez Parrot et diplômé de l'ESIEE en 2012, qui fut notre parrain. Nous lui avons donc exposé notre projet. Ce dernier lui avait plu, il émit juste quelques craintes en ce qui concerne le respect du sujet.

Il est important de noter également que nous étions dans le devoir de réaliser un projet en fin de 3ème année dont l'importance était de taille puisqu'il nous permettrait ou non de passer en 4ème année. L'aide d'un professeur nous était donc également indispensable. Monsieur Rostom Kachouri (enseignant chercheur à ESIEE Paris, département Informatique et Télécommunications) eu un grand intérêt pour notre idée mais surtout pour notre volonté de nous investir dans une telle aventure. Il nous a conseillé au début puis a dû se retirer car il était investi dans beaucoup d'autres travaux et n'aurait donc pas eu le temps de s'occuper à plein temps de notre projet. Monsieur Kachouri nous a fait des remarques en ce qui concernait la complexité de ce projet de suivi automatique. Il avait également peur que cela n'appartienne pas vraiment au sujet du concours Parrot.

Nous avons donc réfléchi à d'autres sujets. Une idée sur laquelle nous nous sommes penchés était une tourelle automatique utilisée au paintball et à l'Airsoft (jeux de guerre utilisant des munitions à billes et lanceurs à air comprimée).

L'idée du skate connecté nous avait également traversé l'esprit, le souci était la localisation précise du skate dans l'espace et la détection de figures. Monsieur Kachouri nous alors rapproché de Mr Grandpierre (enseignant chercheur à ESIEE Paris, département Informatique et Télécommunications) qui nous alors montré une carte Freescale FRDM-KL25Z couplée à une FRDM-FXS-MULTI-B. Celle ci possédant tous les capteurs nécessaires : accéléromètres,

gyroscopes et magnétomètres formant une centrale inertielle. Grâce à ce kit nous avons pu nous lancer réellement dans la réalisation de notre projet : Freestyle System Assist.

II. Introduction

Le but de ce projet est de repérer les figures effectuées par un skateur et de leur affecter un certain nombre de points. Le nombre de points varierait en fonction de la figure effectuée mais également en fonction de la réussite de la figure (puissance et force du saut que l'on appelle "pop" dans le monde du skateboard).

Nous avions également pour idée de faire une restitution en trois dimensions de la planche afin que l'on puisse voir ses mouvements en temps réel mais également de manière à restituer les mouvements lorsqu'une certaine figure est effectuée. Pour reconnaitre les figures, nous nous sommes donc équipés d'un kit de développement Freescale et avons développé une application Android. Le kit possède une centrale inertielle capable de comprendre les mouvements de la planche à tout instant et de les envoyer au téléphone via un émetteur Bluetooth.

L'application nous donnerait, grâce aux données et aux algorithmes implémentés, le nom des figures effectuées ainsi que le nombre de points qu'elles méritent. Le module hardware serait quant à lui fixé solidement sous la planche de skate. Nous n'autoriserions que les figures sur un sol plat (en flat) et par conséquent interdirions toute figure sur module. Cela nous permet de limiter les problèmes de casse sachant qu'il ne s'agirait que d'un premier prototype. Notre produit serait donc composé d'un hardware, le module, et d'un software, l'application Android.

Nous avons décidé d'appeler le produit "Xtrick"

CARTE FREESCALE

III. Carte Freescale

A. Description

1. Caractéristiques techniques

Nous disposons, pour notre projet de deux cartes de développement Freescale :

- Une carte FRDM-KL25Z : elle possède un processeur ARM Cortex MO+, des accéléromètres, des magnétomètres, une LED et un système tactile.
- la carte FRDM-FXS MULTI-B : elle dispose d'un système Bluetooth, d'un port de carte SD, de gyroscopes, d'accéléromètres et de magnétomètres.

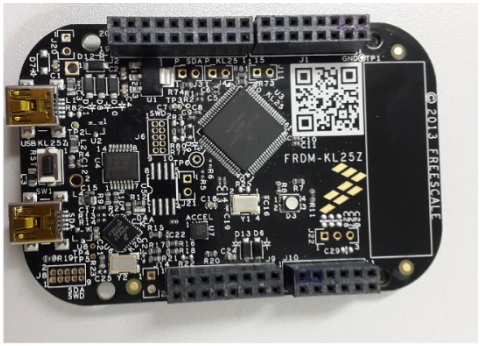


Figure 2 Carte FRDM-KL25Z

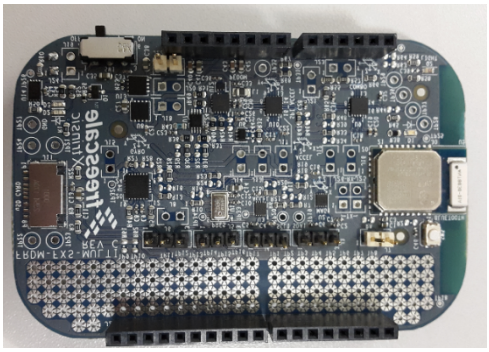


Figure 1 Carte FRDM-FXS-MULTI-B

Le choix de ces cartes a été établi afin de répondre à nos attentes. En effet, nous devons disposer d'un système contenant tous les capteurs nécessaires à nos prises de mesures et un système Bluetooth permettant de lier la carte à notre application Android.

Nous nous sommes renseignés sur différentes cartes : Raspberry Pi, Arduino. Cependant il fallait de plus incorporer des capteurs supplémentaires, un système Bluetooth et une batterie.

Nous optons alors pour le kit Freescale.

2. Logiciel Freescale

Notre choix a également été simplifié par la présence d'un logiciel Freescale : "Freescale Sensor Fusion Toolbox". Par liaison USB, nous avons la capacité de récupérer les données des capteurs de la carte FRDM-FXS-MULTI-B. Tous les mouvements réalisés par la carte sont retransmis simultanément à l'application. Cette application est également capable d'afficher les graphiques des valeurs des capteurs. Il s'agit de la première étape dans l'utilisation de la carte.

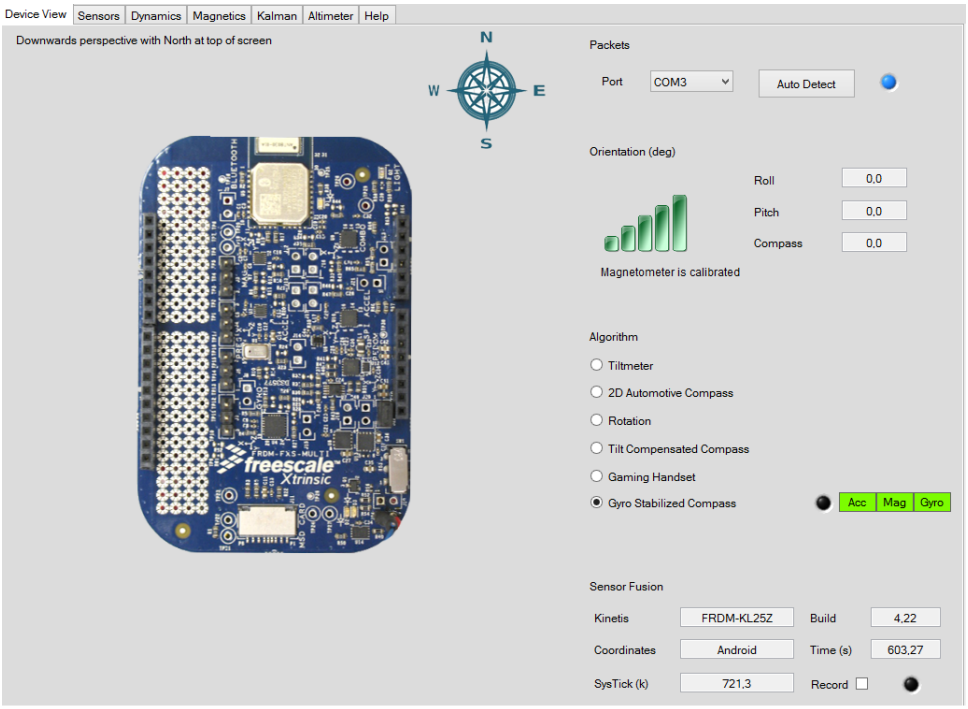


Figure 3 Menu de l'application Freescale

Nous y observons les mouvements en trois dimensions ainsi que les capteurs utilisés.

Ce logiciel permet également de tracer les courbes issues des différents capteurs. Ces premières données donnent lieu à nos interprétations primaires, c'est-à-dire déterminer le rôle des différents capteurs lors du mouvement de la carte.

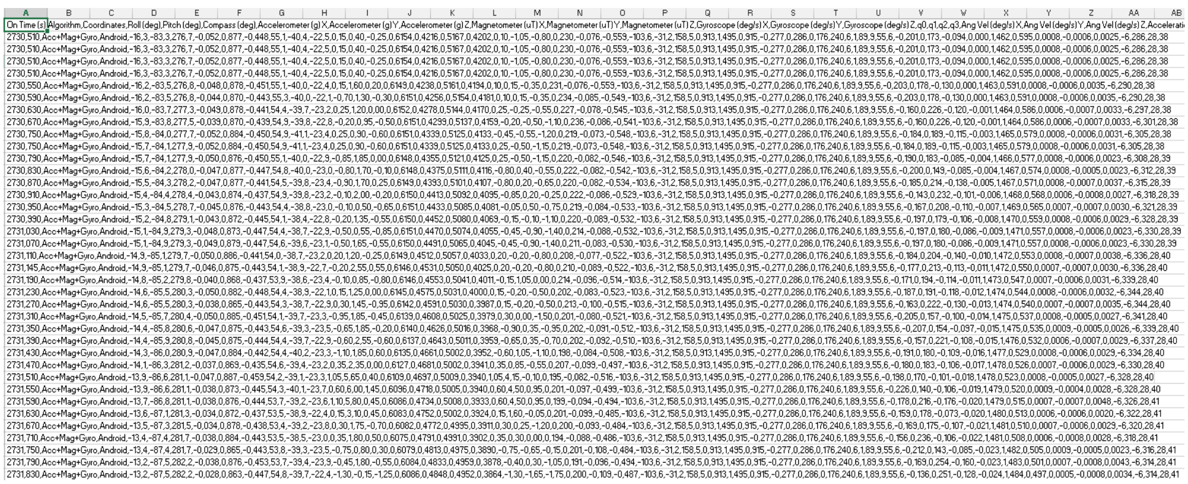
Figure 4 Courbes issues des valeurs des capteurs

Nous voyons ici les courbes de valeurs des capteurs selon les trois axes x, y et z, issues des Accéléromètres, Gyroscopes et magnétomètres.

B. Programmation

3. TeraTerm

Nous pouvons récupérer un fichier en format .csv à partir de la page de menu, s'ouvrant avec Excel par exemple, dans lequel la totalité des données des capteurs ainsi que l'altitude sont affichées. Dans un premier temps, nous avons fait de nombreux tests afin d'interpréter le changement de direction de la carte. Lorsqu'on déplace la carte vers le haut ou vers la droite, en fonction des valeurs des capteurs récupérées, nous devrions être capables de reconnaître son déplacement.



```
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z AA AB
On Time (s),Algorithm,Coordinates,Roll (deg),Pitch (deg),Compass (deg),Accelerometer (g) X,Accelerometer (g) Y,Accelerometer (g) Z,Magnetometer (uT) X,Magnetometer (uT) Y,Magnetometer (uT) Z,Gyroscope (deg/s) X,Gyroscope (deg/s) Y,Gyroscope (deg/s) Z,q0,q1,q2,q3,Ang Vel (deg/s) X,Ang Vel (deg/s) Y,Ang Vel (deg/s) Z,Accelerati
2730,510,Acc+Mag+Gyro,Android,-16,3,-83,3,276,7,-0,052,0,877,-0,448,55,1,-40,4,-22,5,0,15,0,40,-0,25,0,6154,0,4216,0,5167,0,4202,0,10,-1,05,-0,80,0,230,-0,076,-0,559,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,201,0,173,-0,094,0,000,1,462,0,595,0,0008,-0,0006,0,0025,-6,286,28,38
2730,510,Acc+Mag+Gyro,Android,-16,3,-83,3,276,7,-0,052,0,877,-0,448,55,1,-40,4,-22,5,0,15,0,40,-0,25,0,6154,0,4216,0,5167,0,4202,0,10,-1,05,-0,80,0,230,-0,076,-0,559,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,201,0,173,-0,094,0,000,1,462,0,595,0,0008,-0,0006,0,0025,-6,286,28,38
2730,510,Acc+Mag+Gyro,Android,-16,3,-83,3,276,7,-0,052,0,877,-0,448,55,1,-40,4,-22,5,0,15,0,40,-0,25,0,6154,0,4216,0,5167,0,4202,0,10,-1,05,-0,80,0,230,-0,076,-0,559,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,201,0,173,-0,094,0,000,1,462,0,595,0,0008,-0,0006,0,0025,-6,286,28,38
2730,510,Acc+Mag+Gyro,Android,-16,3,-83,3,276,7,-0,052,0,877,-0,448,55,1,-40,4,-22,5,0,15,0,40,-0,25,0,6154,0,4216,0,5167,0,4202,0,10,-1,05,-0,80,0,230,-0,076,-0,559,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,201,0,173,-0,094,0,000,1,462,0,595,0,0008,-0,0006,0,0025,-6,286,28,38
2730,550,Acc+Mag+Gyro,Android,-16,2,-83,5,276,8,-0,048,0,878,-0,451,55,1,-40,0,-22,4,0,15,1,60,0,20,0,6149,0,4238,0,5161,0,4194,0,10,0,15,-0,35,0,231,-0,076,-0,559,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,203,0,178,-0,130,0,000,1,463,0,591,0,0008,-0,0006,0,0035,-6,290,28,38
2730,590,Acc+Mag+Gyro,Android,-16,2,-83,5,276,8,-0,044,0,870,-0,443,55,3,-40,0,-22,1,-0,70,1,30,-0,30,0,6151,0,4256,0,5154,0,4181,0,10,0,15,-0,35,0,234,-0,085,-0,549,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,203,0,178,-0,130,0,000,1,463,0,591,0,0008,-0,0006,0,0035,-6,290,28,38
2730,630,Acc+Mag+Gyro,Android,-16,0,-83,7,277,3,-0,049,0,878,-0,441,54,4,-39,7,-23,2,0,25,1,20,0,00,0,6152,0,4278,0,5144,0,4170,0,25,-0,25,-0,55,0,227,-0,078,-0,545,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,160,0,226,-0,120,-0,001,1,464,0,586,0,0006,-0,0007,0,0033,-6,297,28,38
2730,670,Acc+Mag+Gyro,Android,-15,9,-83,8,277,5,-0,039,0,870,-0,439,54,9,-39,8,-22,8,-0,20,0,95,-0,50,0,6151,0,4299,0,5137,0,4159,-0,20,-0,50,-1,10,0,236,-0,086,-0,541,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,160,0,226,-0,120,-0,001,1,464,0,586,0,0006,-0,0007,0,0033,-6,301,28,38
2730,750,Acc+Mag+Gyro,Android,-15,8,-84,0,277,7,-0,052,0,884,-0,450,54,9,-41,1,-23,4,0,25,0,90,-0,60,0,6151,0,4339,0,5125,0,4133,-0,45,-0,55,-1,20,0,219,-0,073,-0,548,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,184,0,189,-0,115,-0,003,1,465,0,579,0,0008,-0,0006,0,0031,-6,305,28,38
2730,750,Acc+Mag+Gyro,Android,-15,7,-84,1,277,9,-0,052,0,884,-0,450,54,9,-41,1,-23,4,0,25,0,90,-0,60,0,6151,0,4339,0,5125,0,4133,0,25,-0,50,-1,15,0,219,-0,073,-0,548,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,184,0,189,-0,115,-0,003,1,465,0,579,0,0008,-0,0006,0,0031,-6,305,28,38
2730,790,Acc+Mag+Gyro,Android,-15,7,-84,1,277,9,-0,050,0,876,-0,450,55,1,-40,0,-22,9,-0,85,1,85,0,00,0,6148,0,4355,0,5121,0,4125,0,25,-0,50,-1,15,0,220,-0,082,-0,546,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,190,0,183,-0,085,-0,004,1,466,0,577,0,0008,-0,0006,0,0023,-6,308,28,39
2730,830,Acc+Mag+Gyro,Android,-15,6,-84,2,278,0,-0,047,0,877,-0,447,54,8,-40,0,-23,0,-0,80,1,70,-0,10,0,6148,0,4375,0,5111,0,4116,-0,80,0,40,-0,55,0,222,-0,082,-0,542,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,200,0,149,-0,085,-0,004,1,467,0,574,0,0008,-0,0005,0,0023,-6,312,28,39
2730,870,Acc+Mag+Gyro,Android,-15,5,-84,3,278,2,-0,047,0,877,-0,441,54,5,-39,8,-23,4,-0,90,1,70,0,25,0,6149,0,4393,0,5101,0,4107,-0,80,0,20,-0,65,0,220,-0,082,-0,534,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,185,0,214,-0,138,-0,005,1,467,0,571,0,0008,-0,0007,0,0037,-6,315,28,39
2730,910,Acc+Mag+Gyro,Android,-15,4,-84,4,278,4,-0,043,0,874,-0,437,54,9,-39,8,-23,2,-0,10,2,00,-0,20,0,6150,0,4413,0,5092,0,4095,-0,85,0,20,-0,25,0,222,-0,086,-0,529,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,143,0,232,-0,101,-0,006,1,468,0,568,0,0006,-0,0008,0,0027,-6,318,28,39
2730,950,Acc+Mag+Gyro,Android,-15,3,-84,5,278,7,-0,045,0,876,-0,443,54,4,-38,8,-23,0,-0,10,0,50,-0,65,0,6151,0,4433,0,5085,0,4081,-0,05,0,50,-0,75,0,219,-0,084,-0,533,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,167,0,208,-0,110,-0,007,1,469,0,565,0,0007,-0,0007,0,0030,-6,321,28,39
2730,990,Acc+Mag+Gyro,Android,-15,2,-84,8,279,1,-0,043,0,872,-0,445,54,1,-38,4,-22,8,-0,20,1,35,-0,55,0,6150,0,4452,0,5080,0,4069,-0,15,-0,10,-1,10,0,220,-0,089,-0,532,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,197,0,179,-0,106,-0,008,1,470,0,559,0,0008,-0,0006,0,0029,-6,328,28,39
2731,030,Acc+Mag+Gyro,Android,-15,1,-84,9,279,3,-0,048,0,873,-0,447,54,4,-38,7,-22,9,-0,50,0,55,-0,85,0,6151,0,4470,0,5074,0,4055,-0,45,-0,90,-1,40,0,214,-0,088,-0,532,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,197,0,180,-0,086,-0,009,1,471,0,557,0,0008,-0,0006,0,0023,-6,330,28,39
2731,070,Acc+Mag+Gyro,Android,-15,1,-84,9,279,3,-0,049,0,879,-0,447,54,6,-39,6,-23,1,-0,50,1,65,-0,55,0,6150,0,4491,0,5065,0,4045,-0,45,-0,90,-1,40,0,211,-0,083,-0,530,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,197,0,180,-0,086,-0,009,1,471,0,557,0,0008,-0,0006,0,0023,-6,330,28,39
2731,110,Acc+Mag+Gyro,Android,-14,9,-85,1,279,7,-0,050,0,886,-0,441,54,0,-38,7,-23,2,0,20,1,20,-0,25,0,6149,0,4512,0,5057,0,4033,0,20,-0,20,-0,80,0,208,-0,077,-0,522,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,184,0,204,-0,140,-0,010,1,472,0,553,0,0008,-0,0007,0,0038,-6,336,28,40
2731,145,Acc+Mag+Gyro,Android,-14,9,-85,1,279,7,-0,046,0,875,-0,443,54,1,-38,9,-22,7,-0,20,2,55,0,55,0,6146,0,4531,0,5050,0,4025,0,20,-0,20,-0,80,0,210,-0,089,-0,522,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,177,0,213,-0,113,-0,011,1,472,0,550,0,0007,-0,0007,0,0030,-6,336,28,40
2731,190,Acc+Mag+Gyro,Android,-14,8,-85,2,279,8,-0,040,0,868,-0,437,53,9,-38,6,-23,4,-0,10,0,85,-0,80,0,6146,0,4553,0,5041,0,4011,-0,15,1,05,0,00,0,214,-0,096,-0,514,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,171,0,194,-0,114,-0,011,1,473,0,547,0,0007,-0,0006,0,0031,-6,339,28,40
2731,230,Acc+Mag+Gyro,Android,-14,6,-85,5,280,3,-0,050,0,882,-0,448,54,4,-38,9,-22,1,0,15,1,25,0,00,0,6145,0,4575,0,5031,0,4000,0,15,-0,20,-0,50,0,202,-0,083,-0,523,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,187,0,191,-0,118,-0,012,1,474,0,544,0,0008,-0,0006,0,0032,-6,344,28,40
2731,270,Acc+Mag+Gyro,Android,-14,6,-85,5,280,3,-0,038,0,865,-0,443,54,3,-38,7,-22,9,0,30,1,45,-0,95,0,6142,0,4591,0,5030,0,3987,0,15,-0,20,-0,50,0,213,-0,100,-0,515,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,163,0,222,-0,130,-0,013,1,474,0,540,0,0007,-0,0007,0,0035,-6,344,28,40
2731,310,Acc+Mag+Gyro,Android,-14,5,-85,7,280,4,-0,050,0,885,-0,451,54,1,-39,7,-23,3,-0,95,1,85,-0,45,0,6139,0,4608,0,5025,0,3979,0,30,0,00,-1,50,0,201,-0,080,-0,521,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,205,0,157,-0,100,-0,014,1,475,0,537,0,0008,-0,0005,0,0027,-6,341,28,40
2731,350,Acc+Mag+Gyro,Android,-14,4,-85,8,280,6,-0,047,0,875,-0,443,54,6,-39,3,-23,5,-0,65,1,85,-0,20,0,6140,0,4626,0,5016,0,3968,-0,90,0,35,-0,95,0,202,-0,091,-0,512,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,207,0,154,-0,097,-0,015,1,475,0,535,0,0009,-0,0005,0,0026,-6,339,28,40
2731,390,Acc+Mag+Gyro,Android,-14,4,-85,9,280,8,-0,045,0,875,-0,444,54,4,-39,7,-22,9,-0,60,2,55,-0,60,0,6137,0,4643,0,5011,0,3959,-0,65,0,35,-0,70,0,202,-0,092,-0,510,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,157,0,221,-0,108,-0,015,1,476,0,532,0,0006,-0,0007,0,0029,-6,337,28,40
2731,430,Acc+Mag+Gyro,Android,-14,3,-86,0,280,9,-0,047,0,884,-0,442,54,4,-40,2,-23,3,-1,10,1,85,0,60,0,6135,0,4661,0,5002,0,3952,-0,60,1,05,-1,10,0,198,-0,084,-0,508,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,191,0,180,-0,109,-0,016,1,477,0,529,0,0008,-0,0006,0,0029,-6,334,28,40
2731,470,Acc+Mag+Gyro,Android,-14,1,-86,3,281,2,-0,037,0,869,-0,435,54,6,-39,4,-23,2,0,35,2,35,0,00,0,6127,0,4681,0,5002,0,3941,0,35,0,85,-0,55,0,207,-0,099,-0,497,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,180,0,183,-0,106,-0,017,1,478,0,526,0,0007,-0,0006,0,0029,-6,330,28,40
2731,510,Acc+Mag+Gyro,Android,-13,9,-86,6,281,1,-0,047,0,887,-0,459,54,2,-39,1,-23,3,1,05,5,65,0,40,0,6109,0,4697,0,5009,0,3940,1,05,4,15,-0,10,0,195,-0,082,-0,516,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,198,0,170,-0,101,-0,018,1,478,0,523,0,0008,-0,0005,0,0027,-6,328,28,40
2731,550,Acc+Mag+Gyro,Android,-13,9,-86,6,281,1,-0,038,0,873,-0,445,54,3,-40,1,-23,7,0,60,6,00,1,45,0,6096,0,4718,0,5005,0,3940,0,60,4,50,0,95,0,201,-0,097,-0,499,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,226,0,140,-0,106,-0,019,1,479,0,520,0,0009,-0,0004,0,0028,-6,328,28,40
2731,590,Acc+Mag+Gyro,Android,-13,7,-86,8,281,1,-0,038,0,876,-0,444,53,7,-39,2,-23,6,1,10,5,80,0,45,0,6086,0,4734,0,5008,0,3933,0,60,4,50,0,95,0,199,-0,094,-0,494,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,178,0,216,-0,176,-0,020,1,479,0,515,0,0007,-0,0007,0,0048,-6,326,28,41
2731,630,Acc+Mag+Gyro,Android,-13,6,-87,1,281,3,-0,034,0,872,-0,437,53,5,-38,9,-22,4,0,15,3,10,0,45,0,6083,0,4752,0,5002,0,3924,0,15,1,60,-0,05,0,201,-0,099,-0,485,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,159,0,178,-0,073,-0,020,1,480,0,513,0,0006,-0,0006,0,0020,-6,322,28,41
2731,670,Acc+Mag+Gyro,Android,-13,5,-87,3,281,5,-0,034,0,878,-0,438,53,4,-39,2,-23,8,0,30,1,75,-0,70,0,6082,0,4772,0,4995,0,3911,0,30,0,25,-1,20,0,200,-0,093,-0,484,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,169,0,175,-0,107,-0,021,1,481,0,510,0,0007,-0,0006,0,0029,-6,320,28,41
2731,710,Acc+Mag+Gyro,Android,-13,4,-87,4,281,7,-0,038,0,884,-0,443,53,5,-38,5,-23,0,0,35,1,80,0,50,0,6075,0,4791,0,4991,0,3902,0,35,0,30,0,00,0,194,-0,088,-0,486,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,156,0,236,-0,106,-0,022,1,481,0,508,0,0006,-0,0008,0,0028,-6,318,28,41
2731,750,Acc+Mag+Gyro,Android,-13,4,-87,4,281,7,-0,029,0,865,-0,443,53,8,-39,3,-23,5,-0,75,0,80,0,30,0,6079,0,4813,0,4975,0,3890,-0,75,-0,65,-0,15,0,201,-0,108,-0,484,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,212,0,143,-0,085,-0,023,1,482,0,505,0,0009,-0,0005,0,0023,-6,316,28,41
2731,790,Acc+Mag+Gyro,Android,-13,2,-87,5,282,2,-0,038,0,876,-0,453,53,7,-39,4,-23,9,-0,45,1,80,-0,55,0,6084,0,4833,0,4959,0,3878,-0,40,0,30,-1,05,0,191,-0,096,-0,494,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,169,0,254,-0,160,-0,023,1,483,0,501,0,0007,-0,0008,0,0043,-6,314,28,41
2731,830,Acc+Mag+Gyro,Android,-13,2,-87,5,282,2,-0,028,0,863,-0,447,54,8,-39,7,-22,4,-1,30,-0,15,-1,25,0,6086,0,4848,0,4952,0,3864,-1,30,-1,65,-1,75,0,200,-0,109,-0,487,-103,6,-31,2,158,5,0,913,1,495,0,915,-0,277,0,286,0,176,240,6,1,89,9,55,6,-0,136,0,251,-0,128,-0,024,1,484,0,497,0,0005,-0,0008,0,0034,-6,314,28,41
```

Figure 5 fichier.csv de l'ensemble des données récupérées

Nous nous sommes rapidement rendu compte que le fichier fourni est illisible. De plus, nous récupérons les données seulement en fin d'acquisition. Décrypter ce fichier s'avère être très compliqué, nous devons donc trouver un autre moyen pour pouvoir analyser les données. Si possible nous voulions les récupérer simultanément avec les déplacements de la carte. Cela nous permettrait de voir les variations des différentes valeurs et donc de déterminer plus facilement lesquelles influent sur les déplacements de la carte.

Nous avons donc pris l'initiative d'utiliser le logiciel TeraTerm, il s'agit d'un émulateur de terminal. Ce dernier permet d'obtenir les données à partir d'un port COM en connexion USB. Ce sont les valeurs de tous les capteurs pour chaque période d'échantillonnage.

Voici le principe d'utilisation : il faut tout d'abord choisir d'utiliser le port Série puis effectuer les paramétrages de la vitesse d'échantillonnage des données et le nombre de bits relevés.

Cette étape marque une avancée importante, nous avons donc à présent un moyen de déterminer les valeurs données par l'accéléromètre, le gyroscope ainsi que le magnétomètre.

4. CodeWarrior

Après cette première étape, l'objectif est de transmettre les données des capteurs, non plus à l'aide du port USB, mais par le biais du système Bluetooth.

Pour cela nous nous sommes longuement documentés sur CodeWarrior, ce qui fut une tâche difficile à accomplir. Il permet de coder sur le processeur de notre carte FRDM-KL25Z. Le langage de programmation utilisé est le C et le C++.

Afin de prendre en main ce nouvel outil, nous avons une application permettant de modifier la couleur de la LED. Cette première application est codée en langage temps réel, utilisant des wait() entre chaque allumage de LED.

Voici le code permettant le clignotement de la LED. Celle-ci devient successivement rouge, puis verte et enfin bleue :

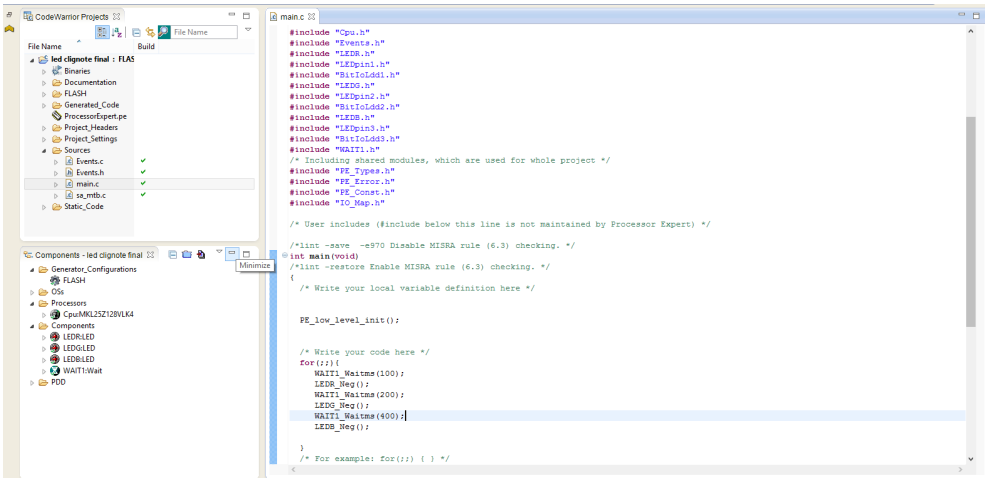


Figure 6 Aperçu de l'interface CodeWarrior

Nous avons ensuite développé un code permettant de modifier la couleur de la LED suivant le placement du doigt sur le support tactile. Là encore cela n'a pas été une mince affaire car il a fallu définir toutes les pins associée à la LED et au pavé tactile.

Le langage C++ nous étant inconnu, coder le système Bluetooth et le transfert de données nous semblaient extrêmement difficiles sur CodeWarrior. En effet le logiciel est très fourni, de nombreuses fonctions internes sont utilisées et la documentation associée est assez pauvre. En outre, nous devons également faire face à un nouveau problème : la puissance du

processeur. Nous devons traiter un grand nombre de calculs et doutions de la capacité du processeur : ARM 32 Bit Cortex- M0, mémoire Flash de 128 KB, mémoire SRAM de 16 KB.

Nous devons donc trouver une alternative afin d'effectuer les calculs et leur traitement. L'application peut être développée sur le processeur de la carte ou sur celui du téléphone. Aujourd'hui les smartphones disposent de quadri cœurs, voire même d'octo cœurs. Ces nouveaux appareils très puissants seraient capables d'effectuer le traitement des données de manière optimale.

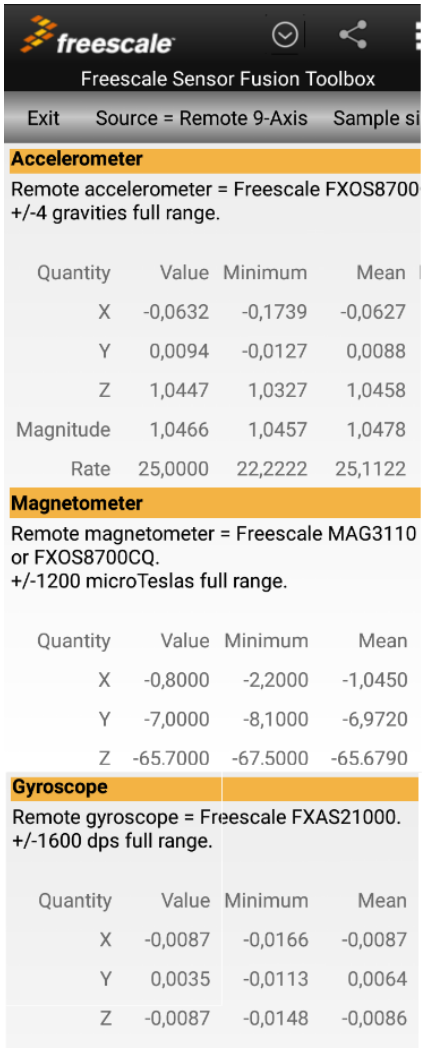
Nous avons donc commencé notre recherche sur la manière dont nous allons implémenter l'envoi des données en Bluetooth sur le logiciel CodeWarrior. Nous y avons passé beaucoup de temps, fait beaucoup de test, nous avons lu beaucoup de documentation, mais ces recherches ne nous ont pas permis de réussir ce que nous comptions faire. Après plusieurs jours sans grand succès, nous nous sommes aperçu que Freescale livrait directement un code source permettant d'activer et d'envoyer des données par le système Bluetooth. Il nous était donc impossible d'avancer par nos propres moyens sans cette découverte.

À l'origine nous avions dans l'idée de définir avec quelles pins était relié le système Bluetooth. Cela n'a rien donné.

En prenant comme base le support de développement fourni par Freescale, nous en avons fini avec CodeWarrior pour notre projet.

5. Freescale sur Android

Nous avons trouvé sur le site Web Freescale une application Android permettant de recevoir et d'afficher les données transmises par Bluetooth. Cette application met en scène une représentation en trois dimensions de la carte supérieure contenant tous les capteurs et le support Bluetooth. Cela est possible à partir d'une étude centrée autour des valeurs des capteurs que sont le magnétomètre, l'accéléromètre et le gyroscope. Tous les mouvements effectués par la carte, sont représentés en trois dimensions.



Cette application nous permet de choisir d'utiliser les capteurs internes du téléphone, de choisir quels capteurs seront actifs, et même d'établir une connections Wifi avec les cartes qui sont capables de transmettre leur donnée en Wifi.

A l'instar de TeraTerm, cette application affiche les données des capteurs simultanément ainsi que la valeur des quaternions. Cet objet mathématique nous étant jusqu'alors inconnu sera développé dans la partie Ma de notre projet.

PAIRAGE BLUETOOTH

IV. Pairage bluetooth

Notre application nécessite un moyen de communication avec le module placé sur la planche. Pour cela nous choisissons le Bluetooth. Il faut alors implémenter du code dans notre application pour que celle-ci puisse détecter notre module automatiquement, démarrer la communication et récupérer les données dont nous avons besoin.

Permission Android :

Afin de pouvoir gérer les fonctions liées au Bluetooth sur une application Android, il faut préciser certaines choses à l'Android Manifest. C'est ce document qui permet d'avoir toutes les informations concernant l'application : son nom, son logo, ses différentes activités, et les permissions. Ces permissions permettent de définir à quelles parties du téléphone l'application a le droit d'accéder. Nous y ajoutons alors les lignes de code suivantes :

```
<USES-PERMISSION ANDROID:NAME="ANDROID.PERMISSION.BLUETOOTH_ADMIN" />

<USES-PERMISSION ANDROID:NAME="ANDROID.PERMISSION.BLUETOOTH" />
```

Activation du module Bluetooth du téléphone :

Pour pouvoir utiliser le Bluetooth dans notre application, il nous faut pouvoir utiliser le module Bluetooth attribué par défaut au téléphone. On récupère ce dernier avec l'appel de la méthode *BluetoothAdapter.getDefaultAdapter()*.

Pairage automatique :

Afin de détecter notre carte automatiquement, nous allons en premier lieu parcourir la liste des appareils déjà appairés au téléphone (dans les réglages Bluetooth internes au téléphone), et vérifier si cette liste contient le nom de notre module. C'est le rôle de cette boucle for :

```
for (BluetoothDevice device : pairedDevices) {
    name = device.getName();
    String address = device.getAddress();
    dev = device;
    uuid = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
    sts = name.startsWith("blueradio BR");
    if (sts) {
        setBtSts(BtStatus.PAIRED, "Found IMU.");
        return;
    }
}
```

Figure 7 Code permettant le pairage automatique de la carte

Création du BluetoothSocket :

Nous créons alors un objet appelé BluetoothSocket qui permettra de gérer les canaux d'émission et de réception en nous servant des informations récupérées précédemment :

```
myServerSocket = dev.createInsecureRfcommSocketToServiceRecord(uuid);
myBluetooth.cancelDiscovery();
```

Stopper la détection :

Une fois notre BluetoothSocket instancié, il faut interrompre la recherche de nouveaux périphériques. Il suffit simplement de faire appel à la méthode cancelDiscovery() sur l'objet de type BluetoothAdapter.

Connexion:

Il ne nous reste plus qu'à créer une connexion. C'est ce que permet le code suivant :

```
myServerSocket.connect();
```

Figure 8 Code permettant la connexion Bluetooth du module

Envoi de données :

Dans notre application, il n'est en aucun cas nécessaire d'envoyer des informations à notre module. En effet, la carte est codée de façon à ce que celle-ci envoie continuellement les informations relatives à ses capteurs. Nous ne ferons qu'écouter la carte lors de la discussion via Bluetooth.

Réception de données :

C'est bien sûr cette partie qui devient la plus délicate. Afin de recevoir les données envoyées par notre module, il nous faut utiliser un canal de réception appelé InputStream. Après avoir connecté notre Socket, nous pouvons alors instancier notre canal de sortie à partir de celui-ci :

```
myBluetoothInputThread = new BluetoothInputThread(myServerSocket, handler);
myBluetoothInputThread.setPriority(Thread.MAX_PRIORITY);
myBluetoothInputThread.start();
```

Figure 9 Création du Thread de réception

Instanciation du canal de réception

Comme vu précédemment, nous créons la classe BluetoothInputThread permettant de temporiser et d'interpréter les données reçues. Le processus consiste à décoder les informations reçues sous forme de bytes, et de les transmettre à notre application à l'aide d'un objet appelé Handler.

Reconnexion

Afin de prévenir l'utilisateur lors d'une déconnexion inattendue, nous procédons de la façon suivante : si jamais nous recevons plus de quatre échantillons de données dont les valeurs des capteurs sont nulles, alors nous affichons un message et relançons une tentative de connexion comme précédemment.

MATHÉMATIQUES

V. Mathématique

A. Utilisation des quaternions

1. Un peu d'historie

Les quaternions ont été découverts par Sir William Rowan Hamilton en Octobre 1843. Il a décrit cette découverte dans ses mémoires selon les mots suivants :

> *"Je me promenais avec Lady Hamilton sur le pont Brougham [Dublin] et cette dernière me parlait de pensées personnelles d'une très profonde signification [...]. Quand soudainement le circuit galvanique de la pensée se ferma et les équations principales des*
>
> *Quaternions se présentèrent `a moi exactement comme je les utilise encore maintenant...".*

La solution lui vint à l'esprit sous la forme des relations : $i^2 = j^2 = k^2 = ijk = -1$ Cet érudit du XIXème siècle irlandais était un astronome de renom, diplômé de l'Académie Royale de Dublin à l'âge de 22 ans seulement. Ses travaux sur l'optique géométrique lui ont permis d'être connu auprès du grand public et ceux en mécanique analytique ont été la base de la mécanique quantique. Hamilton savait que les nombres complexes pouvaient être représentés dans le plan à deux dimensions, et il chercha longtemps une opération dans l'espace à trois dimensions qui généraliserait la multiplication complexe. Frobenius montrera en 1877 que cette recherche était vaine, il fallait introduire une dimension supplémentaire.

Cette découverte découle de l'étude d'une extension des corps $\mathbb{C}$ des complexes. Les quaternions nous permettent de représenter de manière analytique les applications de $\mathbb{R}^2$, comme les homothéties, les rotations, ou bien les translations. Hamilton souhaitait étendre ces notions à l'espace $\mathbb{R}^3$. Après des tentatives infructueuses sur d'éventuels "ternions", c'est-à-dire des triplets (a, b, c) (en considérant les complexes comme des couples (a, b) pour représenter a+ib), Hamilton adapta ses "ternions" aux contraintes opératoires et inventa des quadruplets (a, b, c, d) qu'il nomma "quaternions".

L'étude des quaternions a donc pour finalité une étude des actions de la géométrie dans un espace à 3 dimensions. Les quaternions ont en réalité une application bien plus vaste, notamment en physique pour exprimer les lois de la mécanique ou celle de l'électromagnétisme

de façon globale. Ces derniers possèdent également une application en chimie pour d'écrire le spin de l'électron ainsi qu'en informatique (très utile pour notre projet) pour la création des algorithmes 3D dans les jeux vidéo, ou en mécanique spatiale pour le mouvement des satellites.

2. Quelques mots sur l'espace des rotations

Pour étudier les quaternions, il faut se placer dans l'espace des rotations. Il s'agit d'un espace de 4 dimensions : un scalaire et 3 vecteurs formant une base orthonormée. On pourrait faire l'analogie avec l'espace des complexes qui est un espace à deux dimensions avec une rotation.

L'espace des rotations est représenté par des quaternions unitaires (de norme égale 1 : $||q||^2 = x^2+y^2+z^2$). Lorsque l'angle de la rotation est nul, le quaternion existe. Il est assimilable à un point dans l'espace des rotations et correspond à la rotation identité. Lorsque l'angle est petit mais différent de la rotation nulle, l'ensemble des solutions est assimilable à une sphère entourant la rotation identité ou chaque point de la sphère correspond à une direction donnée. Lorsque l'angle augmente jusqu'à atteindre 180°, on peut assimiler cela à des sphères de rayon croissant. Mais lorsque la valeur de l'angle dépasse cette limite, les rotations suivant les axes ne cessent de diverger jusqu'à un certain point, puis elles convergent jusqu'à atteindre 360° ou elles sont toutes identiques.

3. Définition des quaternions

L'ensemble des quaternions est appelé $\mathbb{H}$. C'est un algèbre possédant les lois + et × associatif mais non commutatif (un peu comme l'ensemble des Matrices M). Un quaternions de $\mathbb{H}$ est définie par un scalaire q_0 et un vecteur à trois dimensions $q_1\mathbf{i}+ q_2\mathbf{j}+q_3\mathbf{k}$, où $(\mathbf{i},\mathbf{j},\mathbf{k})$ sont des vecteurs unitaires d'une base orthonormée :

$q= q_0+ q_1\mathbf{i}+ q_2\mathbf{j}+q_3\mathbf{k}$ où $(q_0,q_1,q_2,q_3) \in R^4$.

Les quaternions ont plusieurs notations:

- $q= q_0+ q_1\mathbf{i}+ q_2\mathbf{j}+q_3\mathbf{k}$
- $q= w+ x\mathbf{i}+ y\mathbf{j}+z\mathbf{k}$
- $q= a+ b\mathbf{i}+ c\mathbf{j}+d\mathbf{k}$

Pour des raisons pratiques nous utiliserons les 3 notations dans ce rapport.

$$i^2 = j^2 = k^2 = -e \quad e^2 = e$$
$$e.i = i.e = i \quad e.k = k.e = k \quad e.j = j.e = j$$
$$i.j = -j.i = k$$
$$j.k = -k.j = i$$
$$k.i = -i.k = j$$

Le nombre réel positif $\|q\|$ défini par $\|q\|^2 = a^2 + b^2 + c^2 + d^2 = q\,\overline{q}$ est appelé **norme** de q. C'est la norme euclidienne associée au produit scalaire usuel sur $\mathbb{R}^4$. Tout quaternion non nul q admet un inverse (unique) donné par $q^{-1} = \frac{1}{\|q\|^2}\overline{q}$.

On peut faire le lien entre quaternions et matrices. Si nous multiplions $\mathbb{H}$ par la gauche, nous faisons alors un morphisme injectif $\mathbb{H} \hookrightarrow \mathrm{End}_{\mathbb{R}}(\mathbb{H}) \approx \mathcal{M}_4(\mathbb{R})$. , $\mathbb{R}$-linéaire nous autorisant donc à représenter un quaternion sous forme matricielle. La matrice associée au quaternion $q = a + bi + cj + dk$ est la matrice suivante :

$$\begin{pmatrix} a & -b & -c & -d \\ b & a & -d & c \\ c & d & a & -b \\ d & -c & b & a \end{pmatrix}$$

Il est possible d'introduire une forme polaire du quaternion (comme la forme polaire du nombre complexe). Cela permet d'utiliser l'angle de rotation et non plus des constantes. On peut donc faire le lien entre les constantes scalaires et l'angle de la rotation.

$$q = w + x\mathbf{i} + y\mathbf{j} + z\mathbf{k} = w + \vec{u}\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \cos(\alpha/2) + \vec{u}\sin(\alpha/2)$$

Où $\vec{u}$ est un vecteur unitaire. On peut retrouver l'angle de la rotation avec le cos et avoir le vecteur $\vec{u}$, vecteur autour duquel s'effectue la rotation.

A. Angles d'Euler

Les angles d'Euler sont un triplet d'angles de rotation (X, Y, Z). Ils représentent respectivement les rotations autour des axes X, Y et Z. Par exemple le triplet (0, 0, 0) est synonyme de la matrice identité, il n'y a aucune rotation. Le triplet (15, 0, 80) quant à lui représente successivement une rotation de 15° autour de l 'axe X et une rotation de 80° autour de l 'axe Z.

Les angles d'Euler ont un défaut : ils utilisent les matrices, la commutativité pose donc problème. Il peut arriver que l'une des rotations autour d'un axe soit confondue avec un autre axe. Le problème se nomme le gimbal lock.

Cependant il existe un moyen d'éviter ce problème, il faut modifier les axes de de rotations :

$$R_{OX} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

Matrice de rotation autour de X d' angle θ

$$R_{OY} = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix}$$

Matrice de rotation autour de Y d' angle φ

$$R_{OZ} = \begin{pmatrix} \cos\rho & -\sin\rho & 0 \\ \sin\rho & \cos\rho & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Matrice de rotation autour de Z d' angle ρ

Nous avons ici les trois angles θ, φ et ρ distincts.

Si l'on veut combiner une rotation selon plusieurs axes, il suffit à présent de multiplier leur matrice de rotation. Généralement, le moyen le plus simple de combiner trois rotations est de faire X*Y*Z, avec X, Y Z respectivement la matrice de rotation associée à l'axe du même nom. Nous récupérons donc ensuite les nouvelles coordonnées de chaque axe en fonction de celle du précèdent triplet.

Cette méthode de résolution par les matrices est convenable mais il existe une autre méthode plus approprié faisant appel aux quaternions. Cette dernière permet d'utiliser l'interpolation, c'est à dire de mettre en place de nouvelles valeurs correspondant à un à un état intermédiaire. Par exemple si l'on souhaite effectuer la rotation entre deux points A et B de manière continue, c'est à dire de mettre en place un chemin entre ces deux points, nous devons créer un ensemble de points intermédiaires. Les quaternions favorisent des rotations plus souples et plus réalistes.

B. Matrices rotation

Comme cela a été dit précédemment l'ordre de rotation peu avoir un rôle important lors de l'utilisation des angles d'Euler. Les quaternions ont comme caractéristique d'éviter ce problème.

En effet, au lieu de faire tourner des points selon une succession de rotation, les quaternions permettent de faire tourner ces points selon un axe prédéfini et d'un angle quelconque.

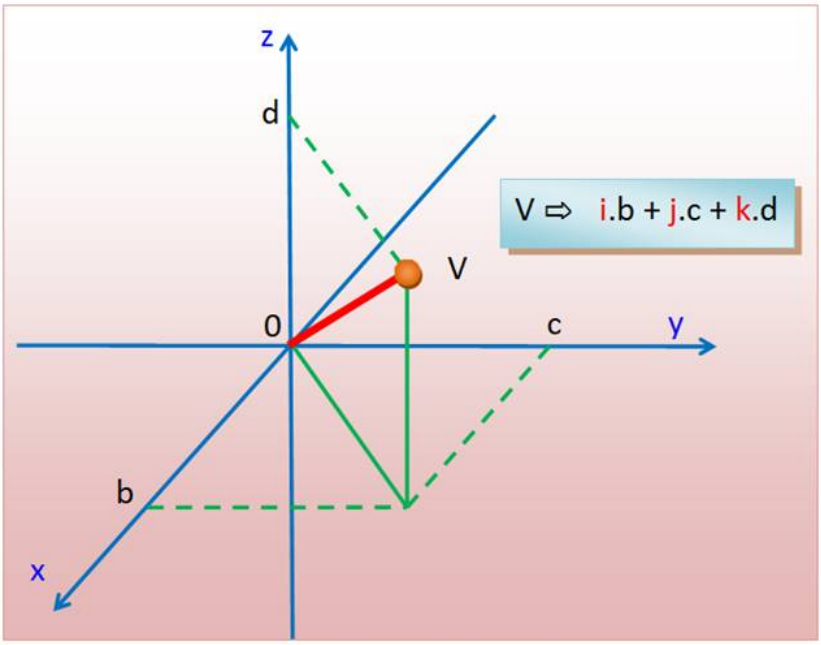
Nous utilisons toujours des matrices de rotations mais au lieu de multiplier les différentes matrices de rotation entre elles, nous multiplions les quaternions les représentant.

Les quaternions seront ici représentés par un quadruplet (a,b,c,d). La formule fondamentales des quaternions est Q = a+ib+jc+kd, avec X un scalaire, (i, j, k) un triplet de complexes, et (b,c,d) l'axe de rotation sous la forme d'un vecteur.

Ils étendent la notion de rotation à trois dimensions à celle en quatre dimensions. Il est également courant de définir les quaternions de la manière suivante:

$$Q=\cos(\text{theta}/2) +\sin(\text{theta}/2)*(ib+jc+kd)$$

Si l'on souhaite au contraire trouver la valeur de l'angle de rotation, cela peut être fait très simplement en utilisant l'identification : a = cos(theta/2), avec étant la seul inconnue, on obtient donc theta = arccos(a)*2 .



Représentation des quaternions dans une repère en trois dimensions

Les quaternions sont représentés dans une matrice de rotation carrée de taille trois.

$$\begin{bmatrix} 1-2b^2-2c^2 & 2ab-2cd & 2ac+2bd \\ 2ab+2cd & 1-2a^2-2c^2 & 2bc-2ad \\ 2ac-2bd & 2bc+2ad & 1-2a^2-2b^2 \end{bmatrix}$$

Dans notre projet, nous récupérons puis plaçons dans un tableau l'ensemble des valeurs des quaternions émise au cours de l'acquisition. Nous nous sommes donc servi de la composante scalaire des quaternions, ici "a" afin de déterminer l'angle de rotation effectué dans l'axe défini en fonctions des trois composantes vectorielles.

Dans nos calculs, nous réalisons tous t'abord une rotation selon X, puis Y et enfin Z:

Nous but est de récupérer les trois angles respectivement des rotations autour de X, Y et Z :

$$R_{xyz}(\theta, \varphi, \rho) = R_{0x}(\theta) * R_{0y}(\varphi) * R_{0z}(\rho) =$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} * \begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{bmatrix} * \begin{bmatrix} cos(\rho) & -sin(\rho) & 0 \\ sin(\rho) & cos(\rho) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

=

$$\begin{bmatrix} cos(\phi)cos(\rho) & cos(\phi)sin(\rho) & -sin(\phi) \\ sin(\theta)sin(\phi)cos(\rho) - cos(\theta)sin(\rho) & sin(\theta)sin(\phi)sin(\rho) + cos(\theta)cos(\rho) & cos(\phi)sin(\theta) \\ cos(\theta)sin(\phi)cos(\rho) + sin(\theta)sin(\rho) & cos(\theta)sin(\phi)sin(\rho) - sin(\theta)cos(\rho) & cos(\phi)cos(\theta) \end{bmatrix}$$

Pour cela nous utilisons la matrice suivante

$$\begin{bmatrix} \theta \\ \phi \\ \rho \end{bmatrix} = \begin{bmatrix} atan2(2cd + 2ab, d^2 - c^2 - b^2 + a^2) \\ -asin(2bd - 2ac) \\ atan2(2bc - 2ac, a^2 + b^2 - c^2 - d^2) \end{bmatrix}$$

Nous pouvons donc définir la valeur de la rotation effectuée et à partir de là en déduire le nom de la figure réalisée.

MODELISATION 3D

VI. Modélisation 3D

Nous avons dans le cadre de notre projet, voulu modéliser une planche de skate en 3D pour que le skateur puisse revoir ses sessions à posteriori. Nous n'avons pas implémenté la capture de vidéo, mais nous avons cependant réussi à modéliser en temps réel le skate sur notre application mobile. Pour réaliser cette modélisation nous avons utilisé OpenGL 2.0 (https://www.opengl.org/).

A. OpenGL

OpenGL est une librairie multi-plateformes permettant l'affichage de formes géométriques 2D et 3D très facilement. Elle permet de définir des formes à partir de points et de vecteurs que nous avons dû définir pour notre skate.

1. Définition du la forme

Nous avons tout d'abord défini la forme de notre skate. Cette librairie permet uniquement de définir des vecteurs et des triangles. C'est sûrement la partie la plus dure de cet affichage. Nous avons donc dû définir des vecteurs construisant plusieurs triangles dans un ordre précis.

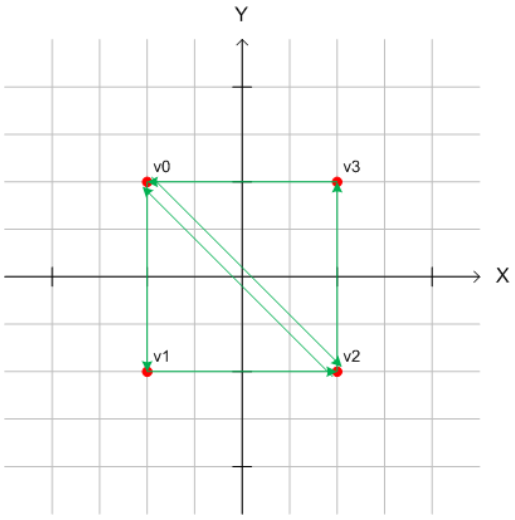


Figure 10 Exemple de création d'un carré avec OpenGL

```
private float[] vertices = { // Vertices for top and bottom
   -1.0f, -1.0f, 0.0f,  // 0. left-bottom-front
    1.0f, -1.0f, 0.0f,  // 1. right-bottom-front
   -1.0f,  1.0f, 0.0f,  // 2. left-top-front
    1.0f,  1.0f, 0.0f,  // 3. right-top-front
   -1.0f, -1.0f, 0.0f,  // 0. left-bottom-front
    1.0f, -1.0f, 0.0f,  // 1. right-bottom-front
   -1.0f,  1.0f, 0.0f,  // 2. left-top-front
    1.0f,  1.0f, 0.0f,  // 3. right-top-front
   -1.0f, -1.0f, 0.0f,  // 0. left-bottom-front
    1.0f, -1.0f, 0.0f,  // 1. right-bottom-front
   -1.0f,  1.0f, 0.0f,  // 2. left-top-front
    1.0f,  1.0f, 0.0f   // 3. right-top-front
};

float[] textureCoordinates = { // Texture coords for the above face
   0.0f, 1.0f,  // A. left-bottom (NEW)
   1.0f, 1.0f,  // B. right-bottom (NEW)
   0.0f, 0.0f,  // C. left-top (NEW)
   1.0f, 0.0f,  // D. right-top (NEW)
   0.0f, 1.0f,  // A. left-bottom (NEW)
   1.0f, 1.0f,  // B. right-bottom (NEW)
   0.0f, 0.0f,  // C. left-top (NEW)
   1.0f, 0.0f,  // D. right-top (NEW)
   0.0f, 1.0f,  // A. left-bottom (NEW)
   1.0f, 1.0f,  // B. right-bottom (NEW)
   0.0f, 0.0f,  // C. left-top (NEW)
   1.0f, 0.0f   // D. right-top (NEW)
};
```

Figure 11 Code définissant les vecteurs de notre skate

Nous avons ensuite trouvé les images de skate adéquates, pour avoir un affichage plus propre et plus attractif pour notre application.

Figure 12 Dessus de la planche

Figure 13 Dessous de la planche

2. Affichage de la forme

Nous avons maintenant notre forme de base qui est définie avec nos images appliquées à celle-ci. Nous devons maintenant l'afficher.

Nous avons défini deux autres classes JAVA. Une Activité qui hérite de GLSurfaceView et une autre qui définit le Renderer de cette dernière. L'une définit la surface sur laquelle nous allons afficher notre skate et l'autre le rendu de la surfaceView. Soit son aspect à sa création, lorsque la surface change etc…

```
public class MyGLSurfaceView extends GLSurfaceView
public class MyGLRenderer implements GLSurfaceView.Renderer
```

Dans un premier temps nous devons définir la SurfaceView. C'est la surface noire derrière notre skate. Elle correspond à un quelconque Layout Android. On peut ajouter cette surfaceView dans n'importe quelle Activité et afficher ce que l'on veut (ou presque) sur cette dernière.

```
GLSurfaceView pcbGlview = (GLSurfaceView) findViewById(R.id.pcb_glview);
```

Elle répond à différents évènements, lorsqu'un utilisateur appuie sur l'écran par exemple.

Nous paramétrons le Renderer de cette surfaceView avec le Renderer que nous avons créé.

```
public MyGLSurfaceView(Context context){
    super(context);

    // Create an OpenGL ES 2.0 context.
    setEGLContextClientVersion(2);

    // Set the Renderer for drawing on the GLSurfaceView
    mRenderer = new MyGLRenderer();
    setRenderer(mRenderer);

    // Render the view only when there is a change in the drawing data
    setRenderMode(GLSurfaceView.RENDERMODE_WHEN_DIRTY);
}
```

Nous devons alors, dans un deuxième temps, définir le Renderer de notre classe. Cette interface définit toutes les méthodes dont Android a besoin pour dessiner notre figure. Elle implémente de plus l'interface GLSurfaceViewRenderer. Cette implémentation nous oblige à redéfinir plusieurs méthodes comme :

```
public void onDrawFrame(GL10 gl)
public void onSurfaceCreated(GL10 gl, javax.microedition.khronos.egl.EGLConfig
config)
```

```
@Override
public void onSurfaceCreated(GL10 gl, javax.microedition.khronos.egl.EGLConfig config) {
    GLES20.glClearColor(0.0f, 0.0f, 0.0f, 1.0f);

    this.aSquare = new Square();

}
```

Figure 14 Code appelé à la création de la surfaceView

On définit grâce à ces deux méthodes l'aperçu que l'on aura de la surfaceView lorsqu'elle sera créée. Ici nous appliquons juste une couleur noire en arrière-plan et créons une nouvelle forme.

La méthode onDrawFrame() permet de définir l'affichage de notre skate et surtout la mise à jour de sa position. C'est ici que nous définissons la position de notre forme.

B. Application des matrices de rotation

Nous avons besoin dans le cadre de notre projet de faire bouger le skate affiché sur l'application en temps réel. Pour cela nous devons relier les informations envoyées par la carte à notre application mobile. Lorsque nous avons défini ces deux classes et que nous avons notre forme prête à l'emploi nous avons besoin de la faire bouger dans l'espace.

Pour cela la librairie OpenGL propose de nombreuses méthodes permettant de faire bouger nos formes très facilement. En effet nous avons commencé par faire tourner notre skate à vitesse constante autour d'un axe. Pour cela nous avons à notre disposition une méthode qui créée une matrice de rotation qui correspond à la rotation de l'angle que nous avons choisi. Nous avons ensuite à entrer cette matrice de rotation dans la fonction adéquat pour avoir notre figure qui commence à tourner.

Dans un second temps nous voulons que notre application réagisse aux appuis sur l'écran de l'utilisateur. Nous avons alors besoin dans notre surfaceView de redéfinir la méthode onTouchEvent(). Elle nous permet de définit ce que fais l'application lorsque l'utilisateur bouge son doigt sur l'écran de son téléphone.

```
@Override
public boolean onTouchEvent(MotionEvent e) {
    // MotionEvent reports input details from the touch screen
    // and other input controls. In this case, you are only
    // interested in events where the touch position changed.

    float x = e.getX();
    float y = e.getY();
    float z = e.getY();

    switch (e.getAction()) {
        case MotionEvent.ACTION_MOVE:

            float dx = x - mPreviousX;
            float dy = y - mPreviousY;
            float dz = z - mPreviousZ;

            // reverse direction of rotation above the mid-line
            if (y > getHeight() / 2) {
                dx = dx * -1 ;
            }

            // reverse direction of rotation to left of the mid-line
            if (x < getWidth() / 2) {
                dy = dy * -1 ;
            }

            mRenderer.setAngle(
                    mRenderer.getAngle() +
                            ((dx + dy + dz) * TOUCH_SCALE_FACTOR));
            requestRender();
    }

    mPreviousX = x;
    mPreviousY = y;
    mPreviousZ = z;
/       float[] color = { 0.0f, 0.0f, 1.0f,0.0f };
/       mRenderer.setSquareColor(color);
    requestRender();
```

Figure 15 Code définissant les réactions au touché de l'écran

Dans ce cas la, nous pouvons faire bouger notre figure avec notre doigt.

Nous voulons maintenant récupérer les informations de la carte et les utiliser pour pouvoir afficher la rotation de notre skate en temps réel. En fait il y a une certaine latence qui est due au temps de récupération et de calcul des quaternions et de la matrice de rotation. Ici nous prennons un vecteur de rotation pour pouvoir afficher les mouvements.

Nous avons juste à appliquer une rotation selon les 3 axes pour pouvoir retrouver la position de notre skate. Il existe une méthode proposée par la librairie OpenGL et plus précisément sur son SurfaceViewRenderer qui permet de donner directement les rotations selon les 3 axes.

```
// Call back to draw the current frame.
@Override
public void onDrawFrame(GL10 gl) {
    gl.glClear(GL10.GL_COLOR_BUFFER_BIT | GL10.GL_DEPTH_BUFFER_BIT);  // clear buffers to preset values
    gl.glMatrixMode(GL10.GL_MODELVIEW);
    gl.glLoadIdentity();  // replace the current matrix with the identity matrix

    demo.dataSelector.getData(rv, this.screenRotation);  // screenRotation only affects fixed rotations

    if (this.lastDisplayedCube<0) {
        A_FSL_Sensor_Demo.write(null, "problem found in onDrawFrame\n");
    }

    gl.glTranslatef(0.0f, 0.0f, 2*cubes[this.lastDisplayedCube].offset);   // Translate into the screen
    gl.glRotatef(rotationDegrees[this.screenRotation], 0, 0, 1);  // portrait/landscape rotation
    gl.glRotatef(rv.a, rv.x, rv.y, rv.z); // parameters are angle and axis of rotation
```

Figure 16 Code permettant de mettre à jour la position du skate

ANALYSE DES FIGURES

VII. Analyse des figures

A. Récupération des données

Pour récupérer ces données nous créons un thread de traitement permettant leur gestion en parallèle à l'application.

Afin de ne pas compliquer notre code en insérant des contraintes liées au temps (intervalle de temps entre chaque échantillon par exemple), nous préférons laisser le choix au processeur du téléphone. En effet, si nous laissons notre thread travailler sans contraintes, celui-ci nous livre une quantité de l'ordre de la centaine de millions d'échantillons de valeurs en seulement 5 secondes d'acquisition. Il nous suffit alors de faire un simple compteur, et d'imposer au thread de nous livrer un échantillon chaque fois que son compteur est congru à 100 000. Ce qui nous permet d'obtenir environs 200 échantillons en 5 secondes, un nombre suffisant de valeurs pour rechercher une figure.

Nous décidons dans un premier temps de récupérer les données brutes dans une ArrayList<ArrayList<Double>> à chaque période d'échantillonnage nous créons une ArrayList de double avec 4 éléments. Les 4 coordonnées de notre quaternion. Nous nous rendons compte très vite que ce n'est pas la bonne solution.

Nous décidons alors de changer de méthode. Nous calculons à chaque période d'échantillonnage une matrice 1x3 comportant les angles d'Euler sur les 3 axes. Nous plaçons ces 3 résultats dans une ArrayList<Double> à laquelle nous ajoutons la valeur de l'accélération sur l'axe Z (nous verrons pourquoi dans la partie Tricks). Nous gardons à chaque fois ces ArrayList de 4 Double dans une autre ArrayList.

Nous avons alors une ArrayList<ArrayList<Double>> contenant à chaque période d'échantillonnage la valeur absolue des angles sur les 3 axes du repère (et une accélération). Cette valeur n'est pas très intéressante non plus car elle ne permet pas de définir un mouvement et encore moins une rotation mais uniquement de trouver la position de notre carte dans l'espace. Celle-ci nous suffira tout de même pour la détection de nos figures.

```
Double beta =  Math.atan2(2 * q2 * q3 + 2 * q0 * q1, q3 * q3 - q2 * q2 - q1 * q1 + q0 * q0) * 180.000/ Math.PI;
Double alpha = Math.asin(2 * q1 * q3 - 2 * q0 * q2) * 180.000/ Math.PI;
Double phi =  Math.atan2(2 * q1 * q2 + 2 * q0 * q3, q1 * q1 + q0 * q0 - q3 * q3 - q2 * q2) * 180.000/ Math.PI;
```

B. Analyse des données

Vient maintenant la partie qui permet de définir les figures en fonction des données que nous avons récupérées lors des différentes acquisitions. Le principe de détection est simple, nous disposons d'un tableau de valeurs, nous avons simplement à le parcourir et tester les conditions nécessaires selon les figures que l'on veut détecter. Afin d'optimiser la détection de figure, nous testons en premier lieu les figures plus complexes en terme de rotation. En effet une fois un 'flip' détecté (rotation autour de l'axe Y de la planche) il est inutile de chercher à détecter un 'ollie' (simple saut avec la planche).

1. Première approche :

Nos premiers pas sur la détection de figure visaient à identifier ou non une figure appelée 180 qui consiste bien sûr à faire tourner la planche d'un demi-tour. Pour cela nous avons simplement comparé les positions de départ et de fin de l'acquisition. Avec une marge d'une dizaine de degrés, nous sommes capable d'affirmer ou non si le skateur réussis cette figure.

Le problème étant majeur : le skateur doit réussir parfaitement la figure, et ne plus bouger sa planche jusqu'à la fin de l'acquisition, ce qui n'est pas acceptable aux vues de nos attentes. Mais cette méthode reste tout de même notre point de départ pour l'étude de figures.

2. Définition des différentes figures

n x 180 :

Nous reprenons la méthode précédente mais pour un 360 cette fois ci (un tour complet de notre planche). En suivant le même principe, si la planche revient à sa position initiale, alors la rotation finale autour de Z est proche de 0. Il suffit alors d'identifier un tour. Pour cela, il suffit de trouver un saut de 360 degrés entre 2 valeurs successives dans notre tableau d'acquisitions.

Afin de détecter cette variation brusque de 360 degrés, nous implémentons la méthode findNumberRotation() qui nous donne le nombre de saut obtenus sur un axe. Ces 2 conditions nous permettent donc de déterminer toutes les figures de type n x 180.

Un 180 est composé d'une valeur finale de 180 degrés et de zéro rotation.

Un 360 est composé d'une valeur finale de 0 degré et d'une rotation.

Un 540 (un tour et demi) est composé d'une valeur finale de 180 degrés et d'une rotation.

Un 720 (2 tours) est composé d'une valeur finale de 0 degrés et de 2 rotations.

Ainsi de suite pour un nombre infini de tours.

Éviter l'utilisation de la valeur de fin :

Comme nous l'avons précisé précédemment, nous devons remédier au problème de la valeur finale. Il n'est pas acceptable que le skateur ne puisse pas bouger sa planche une fois sa figure réussie. Nous allons alors procéder différemment au début : nous commençons d'abord par parcourir le tableau, et lorsqu'une valeur proche de 0 (ou 180) degrés est détectée, nous vérifions si les 10 valeurs suivantes du tableau sont elles aussi proches de 0 (ou 180) degrés. Ce qui correspondrait au bref moment d'immobilité de la planche une fois la figure terminée. Si c'est le cas, là alors nous revenons au point précédent.

Back Flip / Front Flip :

Le principe est simplement le même que précédemment pour les rotations autour de Z, nous avons procédé de la même manière pour les rotations autour de X.

Kick Flip / HeelFlip :

Ces figures sont assez courantes en freestyle en skateboard. Il est vraiment important de pouvoir les détecter. Elles consistent à sauter et faire tourner la planche de 360 degrés selon l'axe Y de la planche.

Pour la rotation autour de Y nous avons fait face à un problème très contraignant : Les priorités de rotations. En effet en récupérant les valeurs des quaternions, nous sommes capables de trouver la position de notre planche à tout instant si nous connaissons les valeurs initiales. Mais notre méthode ne prend pas en compte l'ordre des 3 rotations effectuées avant d'arriver à la position finale.

Par exemple, lorsqu'un objet effectue une rotation de 90 degrés selon X puis de 90 degrés selon Z, il sera formellement dans la même position finale que s'il tourne de 90 degrés sur son axe Y. C'est exactement le problème que nous rencontrons pour détecter ce type de figures : lors d'un demi-tour selon l'axe Y de la planche, nous obtenons le tableau final suivant [180, 0, 180], or nous souhaiterions obtenir [0, 180, 0].

Pour régler ce problème nous décidons alors d'implémenter la méthode findNumberFlipStatement (voir le code en annexe, classe Traitement) qui nous permet, tout

comme findNumberRotation de trouver le nombre de fois où nous avons atteint une situation du type [180, 0, 180].

Celle-ci ne fonctionne que dans les situations optimales, à savoir une rotation parfaite autour de Y, ce qui est improbable dans le monde de notre étude. Nous améliorons alors la méthode et pensons à cette astuce :

```
Double test = (x+y)/2;
```

Si la valeur de 'test' est suffisamment proche de 180, cela signifie que nous avons fait une rotation selon Y et donc un flip. Ce qui règle le problème de la rotation parfaite.

Manual / Nose Manual :

Le Manual consiste a rouler sur 2 roues et non sur 4 pendant le plus longtemps possible. Le principe est simple, dès qu'une valeur de rotation selon X est supérieure à 10 degrés, cela suffit à dire que le skateur est en position de Manual. Il suffit alors de compter le nombre de valeurs supérieures à 10 à partir de la première relevée pour avoir une estimation de la durée du Manual.

Viens alors une complication mineure : si jamais le skateur venait à toucher le sol avec l'arrière de sa planche et réussissait à continuer dans cette position de Manual, la figure serait tout de même validée. Pour fixer ce problème nous imposons alors une valeur maximale de 30 degrés. Les Manuals sont donc opérationnels pour notre application.

Ollie :

Afin de détecter la première figure apprise par tout skateur, il suffit d'identifier une variation brusque de la valeur de rotation selon l'axe X. Cette simple méthode suffit, mais nous verrons dans la partie Pop comment nous la performons.

3. Hiérarchie de figures :

Maintenant que nous savons détecter chacune de nos figures, il faut pouvoir optimiser le parcours du tableau d'acquisition et la restitution de figure. Pour cela nous imposons un ordre de test selon la priorité de la figure.

Par exemple lors d'un Kick Flip, le skateur doit réaliser un Ollie suivi d'une rotation. Si nous tentons de détecter le Ollie avant le Kick Flip dans nos tests, alors dans notre cas, seul un Ollie sera détecté et non le Flip. Voici alors l'ordre de priorité que nous avons attribué aux figures de notre application :

1. Kick Flip
2. N x 360
3. N x 180
4. Backflip
5. Ollie
6. Manual.

Après détection nous créons un nouvel objet Tricks et arrêtons l'analyse du tableau avec un 'return'.

Goofy ou Regular ?

Lors du lancement de notre application, le skateur peut choisir son orientation (pied droit ou gauche à l'avant de la planche). En fonction de ce choix, une précision supplémentaire est accordée aux figures. Et afin de détecter s'il s'agit d'un Nose Manual ou d'un simple Manual (sur les roues avants ou arrières), nous observons le signe des valeurs de variation d'angle.

'Pop' :

Afin d'attribuer un score spécifique à la réussite des figures, nous allons utiliser la valeur de l'accélération récupérée dans le tableau d'acquisitions comme précisé précédemment. Nous parcourons l'ensemble des valeurs d'accélération et gardons la plus forte, pour finalement la multiplier par un coefficient de réussite. Notre application est finalement capable de détecter 2 Ollies et de les distinguer.

C. Tricks

Une fois notre figure détectée, il nous faut gérer son affichage à l'utilisateur, et le nombre de points que cette figure lui rapportera. Nous créons la classe Tricks (voir le code en annexe) qui se chargera de tout ceci. En réalité cette classe est l'intermédiaire entre l'algorithme de calcul de figure, et la partie affichage de l'application. C'est elle qui impose quel nom et quel score attribuer à la figure retournée au joueur, en fonction des informations reçues de l'algorithme de reconnaissance.

Afin de pouvoir afficher toutes ces données au joueur, nous rééditons la méthode toString() de Java à l'aide du @Override. Cette pratique consiste à modifier une méthode préexistante dans la Java Doc. Nous faisons alors en sorte que cette nouvelle méthode affiche nom, score, et valeur du 'pop' ou longueur de la figure en fonction de la figure réalisée. Nous n'avons plus qu'à appeler la méthode toString() sur notre nouvel objet Tricks, puis afficher le texte obtenu dans une ListView. Une ListView est une liste d'affichage permettant d'y placer plusieurs lignes de texte, idéale ici pour notre application.

```
@Override
public String toString(){
    if(this.TricksName == "Connection card lost"){
        return (this.TricksName+ "- Please try to restart");
    }
    else if(this.TricksName == "Nothing"){
        return (this.TricksName + " " + this.TricksValue);
    }
    else if(this.TricksName.contains("Manual")){
        return (this.TricksName + " " + this.TricksValue + "  -  Long " + this.PopValue);
    }
    else {
        return (this.TricksName + " " + this.TricksValue + "  -  Pop "+this.PopValue);
    }
}
```

AFFICHAGE DES STATISTIQUES

VIII. Affichage statistiques

Nous avons aussi pensé qu'il serait bien de garder en mémoire un certain historique des différentes parties de l'utilisateur.

A. Délimitation en Parties

Nous avions commencé par implémenter des simples acquisitions de 5 secondes, que l'on pouvait faire sans limite. Afin de rendre notre application plus ludique, nous avons instauré un système de parties. Ceci permet de limiter le nombre d'acquisitions à 5 et de retenir le score total obtenu sur la partie, pour pouvoir ensuite l'enregistrer dans la base de statistiques de l'application.

1. Classe Player

Nous avons tout d'abord réalisé plusieurs classes permettant de structurer le code des parties. Avant nous créions à chaque nouvelle partie une seule acquisition. Et actualisions le score s'il était plus grand que le meilleur score actuel. Nous avons créé une classe Player avec plusieurs attributs permettant de garder un nom, une orientation de pied, et une liste de partie. Nous avons redéfinis la méthode toString() pour faciliter l'affichage des informations des joueurs. A chaque fois qu'une partie se termine nous appelons la méthode addGame(Game pG) qui permet d'ajouter la partie au joueur courant.

```
/**
 * Constructor for objects of class Player
 */
public Player(A_FSL_Sensor_Demo pF)
{
    this.fsl = pF;
    SharedPreferences settings = this.fsl.getSharedPreferences("SharedSettings", Context.MODE_PRIVATE);
    String vPlayer = settings.getString("PlayerName", "false");
    String vFoot = settings.getString("FootOrientation", "false");
    this.aName = vPlayer;
    this.aOrientation = vFoot;
    this.aGames = new ArrayList<Game>();
}
```

Figure 17 Constructeur de la classe Player

```
@Override
public String toString(){
    String vReturnString = this.aName;
    for(int i=0;i<this.aGames.size();i++){
        vReturnString += this.aGames.get(i).toString();
     }
    return vReturnString;
}
```

Figure 18 Redéfinition de la méthode toString()

2. Classe Game

```
public class Game implements Comparable<Game>{
```

Figure 19 Exemple implémentation d'une interface en JAVA

Nous avons aussi défini une nouvelle classe Game qui sert à définir nos différentes parties. Elle cache en fait uniquement une liste de Tricks (Classe expliquée plus haut). Nous avons ici aussi définis plusieurs méthodes importantes. Nous avons définis un méthode ComputeScore() qui permet de calculer le score actuel de la partie en prenant en compte tous les tricks réalisés durant cette dernière. Mais aussi la méthode toString().

Nous avons implémenté l'interface Comparable sur cette classe pour pouvoir redéfinir la méthode compareTo() et pouvoir trier très facilement un tableau de Parties. Cette méthode compareTo() renvoit un entier négatif si la partie passée en paramètre est plus « grande » que la partie sur laquelle nous appelons la méthode. Nous devions alors définir ce que voulait dire 'plus grande' pour une Partie. Ici il s'avère évident que nous nous baserons sur le score de celle-ci. Nous avons donc juste à soustraire les scores des deux parties pour écrire cette fonction.

```
public void computeScore(){
    Log.v("GAME", "begin Compute Score");
    this.aScore =0;
    for(Tricks t: this.aTricks){
        this.aScore += t.setTotalScore();
    }
}
```

Figure 20 Méthode ComputeScore() de la classe Game

```
@Override
public String toString(){
    String vReturnString = "" ;
    for(int i=0; i<this.aTricks.size();i++){
        Tricks vT = this.aTricks.get(i);
        vReturnString +=  vT.toString() + "\n" ;
    }
    return vReturnString + "\n" ;
}
```

Figure 21 Redéfinition de la méthode toString() pour la classe Game

```
@Override
public int compareTo(Game pG){
    return this.getScore() - pG.getScore();
}
```

Figure 22 Redéfinition de la méthode compareTo() due à l'implémentation de Comparable<Game>

3. Classe BestGames

Nous avons aussi voulu garder les cinq parties les plus réussies par les différents utilisateurs. Nous avons alors créé une classe BestGames qui permet de stocker dans une liste les cinq meilleurs scores. Nous devons créer plusieurs méthodes afin que cette liste reste toujours triée à chaque fois que l'on lui ajoute un élément.

Nous décidons dans un premier temps d'utiliser un tableau de cinq éléments. Mais il s'avère qu'en JAVA les ArrayList<Game> sont bien plus faciles à utiliser que les tableaux simples. Nous prenons alors une ArrayList comme point de départ et créons une méthode d'ajout à la bonne position pour chaque partie réalisée par l'utilisateur.

B. Stockage des différentes parties

Nous voulions à la base garder en mémoire toutes les parties réalisées par un joueur en fonction de la date et effectuer une page de statistiques détaillées en fonction du temps. Mais nous n'avons pas réussi à trouver un moyen assez simple pour stocker en local ces informations. En effet, nous avons tout d'abord créé une base de données SQLite. La création de cette base de données s'est révélée très compliquée. En effet il faut créer environ 3 classes JAVA pour l'utilisation de d'une Base SQLite et autant de classe pour chaque objet différent que nous voulons stocker dans cette dernière.

Après que nous ayons créé cette interface permettant d'accéder à un environnement SQL nous avions besoin de manipuler celle-ci mais cette manœuvre s'est avérée plus que compliquée. Surtout si nous voulions créer une base de données relationnelle (ie : Plusieurs joueurs pouvant avoir plusieurs parties pouvant contenir plusieurs Tricks).

Nous avons alors abandonné cette idée. Nous avons décidé de ne garder que les cinq meilleurs scores réalisés sur l'application. Pour cela nous utilisons les SharedPreferences proposées par Android. Celles-ci permettent de stocker les préférences des utilisateurs pour une application. Nous pouvons alors stocker des Strings, des Booléens, des entiers. Nous pouvons stocker et récupérer ces informations grâce à un ensemble de clé-valeurs. Chaque clé correspond à une valeur unique. Nous allons un peu détourner leur utilisation pour stocker une String contenant toutes les informations dont nous avons besoin.

Nous créons alors une String avec des séparateurs bien précis qui nous permettent de délimiter les différentes informations que nous voulons stocker. Chaque partie entre des « / », le nom du joueur et son score son eux séparés par des tirets. Nous pouvons stocker cette information sur la mémoire local du téléphone en utilisant les SharedPreferences. Nous utilisons cette façon de faire dans la fonction save() de la classe BestGames.

```
public void save(){
    Log.v("BEST GAMES", "save()");
    SharedPreferences settings = fsl.getSharedPreferences(PREFS_NAME, 0);
    SharedPreferences.Editor editor = settings.edit();
    String content = turnArrayInSaveString(this.aGames);
    editor.putString("BestGames", content);
    editor.commit();
}
```

Figure 23 Fonction permettant de sauver les meilleurs parties en local

Nous avons aussi besoin de récupérer ces données au démarrage de l'application. Lorsqu'on créer un nouvel objet bestGames nous récupérons les informations stockées à partir des SharedPreferences. Dans ce cas ce n'est plus un PUT mais un GET avec la clé correspondante. Dans le constructeur on appelle la méthode readFromFile() qui va directement récupérer les informations dont nous avons besoin et nous retourne une String.

```
/**
 * Constructor for objects of class BestGames
 */
public BestGames(A_FSL_Sensor_Demo pF)
{
    this.fsl=pF;
    String data = readFromFile();
    ArrayList<Game> result = turnStringIntoArray(data);
    this.aGames = result;
    if(this.aGames.size()<1){
        Log.v("BestGames", "Taille nulle");
        this.aGames.add(new Game("", 0));
    }

}
```

Figure 24 Constructeur de la classe BestGames permettant de récupérer les meilleures parties

```
public String readFromFile(){
    SharedPreferences settings = fsl.getSharedPreferences(PREFS_NAME, 0);
    String content = settings.getString("BestGames", "false");
    return content;
}
```

Figure 25 Méthode permettant d'accéder aux SharedPreference de l'appareil

Nous devons bien entendu transformer notre ArrayList<Games> en String pour pouvoir les stocker à la fermeture de l'application mais aussi de transformer la String que nous récupérons au démarrage pour pouvoir recréer notre ArrayList.

```
public ArrayList<Game> turnStringIntoArray(String vS){
    Log.v("BEST GAME", "turnStringIntoArray()");
    String[] vGames = vS.split("/");
    ArrayList<Game> vReturnArray = new ArrayList<Game>();
    if(vGames.length>1) {
        for (int i = 0; i < vGames.length; i++) {
            String[] infos = vGames[i].split("-");
            String vPlayerName = infos[0];
            Integer vScore = Integer.parseInt(infos[1]);
            Game vGame = new Game(vPlayerName, vScore);
            vReturnArray.add(vGame);
        }
    }
    return vReturnArray;
}
```

Figure 26 Fonction permettant de transformer la chaine de caractères sauvée en ArrayList

```
public String turnArrayInSaveString(ArrayList<Game> pA){
    Log.v("BEST GAME", "turnArrayInSAveString()");
    String vReturnString = pA.get(0).getPlayerName() + "-" + pA.get(0).getScore();
    for(int i=1; i<pA.size();i++){
        Game vG = pA.get(i);
        vReturnString += "/" + pA.get(i).getPlayerName() + "-" + pA.get(i).getScore();
    }
    return vReturnString;
}
```

Figure 27 Fonction permettant de transformer une ArrayList en String pour être sauvé en local

C. Affichage des différentes parties

Nous souhaitons faire une page de statistiques dans notre application. Nous ajoutons un Layout que l'on rendra visible lors de l'appel de la page de statistiques. Un Layout, linéaire ou relatif, et une sorte de conteneur, dans lequel nous pouvons placer toute sorte d'objet graphique. Dans notre cas nous y ajoutons chaque partie jouée sous forme de texte. Chacune d'entre elle contiendra le nom du joueur, ses orientations (Goofy ou Regular), et ses 5 tentatives de figures.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/skate_4"
    android:id="@+id/statisticsView"
    android:orientation="vertical"
    android:visibility="gone">


    <ScrollView
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:id="@+id/ScrollViewStats"
        android:layout_gravity="center_horizontal"
        android:layout_margin="25dp"
        android:background="@drawable/list_background"


        >

        <LinearLayout
            android:orientation="vertical"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/linearStat">
        </LinearLayout>


    </ScrollView>
</LinearLayout>
```

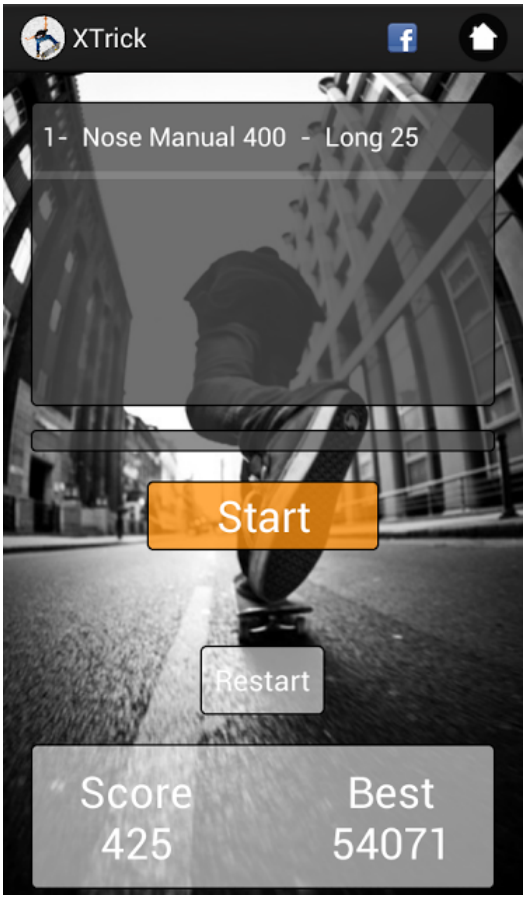
Figure 28 Layout Statistiques

PARTAGE SOCIAL

IX. Partage social

A. Facebook

Une facette de notre application est également le partage via les réseaux sociaux l'ensemble de ses exploits, et en tête de file nous retrouvons évidemment Facebook.



Il regroupe aujourd'hui près d'un milliard et demi d'adhérents. Ces outils informatiques très tendance rassemble un très vaste public. Notre application centrée autour du sport peut permettre une émulation entre différents utilisateurs et mettre en place une certaine compétition. Il parait donc essentiel d'implémenter cette fonctionnalité dans notre application.

Facebook est conscient de son attrait et a mis à disposition des développeurs un SDK. Il s'agit d'un moyen simple et efficace d'ajouter cette fonctionnalité dans son application sous la forme d'un module à incorporer à son code Android. Nous récupérons de cette manière une bibliothèque Facebook dans notre application contenant toutes les méthodes nécessaires.

Facebook a mis en ligne un service spécialement destiné aux développeurs d'applications Android, dans lequel figures toutes les indications nécessaires à son utilisation:

https://developers.facebook.com/docs/android/getting-started

Nous avons donc suivi ces indications, tout d'abord il faut créer un identifiant à son application dans la base de données de Facebook puis importer le module à son application.

Nous avons également permis à notre application d'autoriser la liaison Internet afin de permettre l'échange de données.

La mise en place du SDK Facebook permet de partager soit un lien internet, soit une photo, soit une vidéo sur le réseau social.

Dans notre cas, nous partageons une image, il s'agit en fait d'une capture d'écran de l'application dans laquelle figure le nom des figures réalisées, ainsi que le score atteint par le skateur.

Un commentaire par rapport à sa performance peut également être publié simultanément à l'envoi de l'image. Cela permet de rajouter un côté ludique à notre application.

B. Autres :

Nous avons également la possibilité de partager nos performances sans utiliser le SDK Facebook. Nos résultats peuvent donc être transmis par sms, par mail, Google + et sur tous les autres réseaux sociaux.

Si l'on choisit de ne pas utiliser le SDK Facebook mais que l'on souhaite tout de même publier des informations sur Facebook, il n'est alors possible que de mettre en ligne un message seul. Ce n'est pas ce que nous recherchons car il n'y aura aucun moyen de savoir que le message est été envoyé à partir de notre application. La possibilité d'envoyer un message prédéfini dans lequel nous pourrions dire que ce message est été envoyé à partir de l'application Xtrick serait impossible. Cette fonctionnalité est tout de même possible sur tous les autres réseaux sociaux.

PACKAGING

X. Packaging

A. Matériaux

Afin de protéger la carte des chocs qu'elle pourrait subir, il fallait trouver des matériaux solides. Un caisson en métal n'était pas envisageable. En effet avec le Bluetooth, des parasites auraient pu venir interférer avec le signal émis et donc fausser nos données reçues. Il fallait alors un matériau plutôt de type plastique. Nous avons en premier lieu pensé à du plexiglas. Le problème était le prix mais également le fait qu'il soit transparent; le fait de voir les cartes n'était pas très attrayant. Sachant que nous sommes tous étudiants et que nous ne gagnons pas notre vie, nous devions trouver une solution qui de surcroit ne nous couterait pas trop chère si possible. Nous avons donc eu l'idée de nous procurer des boites en plastique auprès de notre école. Malheureusement, les boîtes étaient soit trop petites, soit trop grandes ou encore trop fragiles. Nous avons donc décidé d'aller voir s'il était possible de trouver des chutes de matériaux correspondant à nos attentes dans une déchetterie municipale. L'avantage était que les éléments sont en grande partie triés ce qui nous permis de cibler exactement ce qui nous intéressait. Nous avons trouvé une vieille valise de qualité en plastique presque intacte.

Ayant accès à de l'outillage de précision, nous étions capables de découper des morceaux correspondant aux cotes de la carte afin que cette dernière puisse y rentrer parfaitement. L'avantage de ce type de plastique est qu'il est très solide (il devait résister aux multiples voyages d'une personne) mais aussi souple ce qui permettait un certain degré de liberté en terme de manipulation. Nous avons donc pris les différents morceaux et les avons ajustés avec du scotch dans le but de former une boite solide. Lorsque l'on mit la carte, cette dernière rentrait parfaitement bien mais des légers mouvements subsistaient. Nous nous sommes

procuré de la mousse compacte et solide, non conductrice afin de bloquer la carte complètement. Ainsi, lorsque la planche bougerait, la carte ferait exactement les mêmes mouvements que la planche et nous aurions donc une bonne précision en ce qui concerne la détection de figures.

Désormais, nous avions une boite aux dimensions exactes de la carte et nous fallait trouver un moyen de la fixer sur le skate par la suite.

B. Fixation

En ce qui concerne la fixation du module, notre but était de trouver un système qui nous permettait de mettre et retirer la carte sans trop de difficultés. Il fallait de plus que la boite soit solidaire de la planche pour des raisons de précisions une fois encore. Nous avions pour première idée de fixer quatre équerres sur la planche et d'y fixer la boite avec des vis. Celle-ci ne bougerait donc pas mais la limite de ce système était sa modularité. En effet il aurait fallu visser puis dévisser pour chaque utilisation, cela aurait été un véritable calvaire. Nous avons donc trouvé des petites attaches en formes de U. Chacune venait se loger entre l'équerre et la boite d'une part, et venait au-dessus de la boite d'autre part. Nous joignons alors les attaches dans le sens de la longueur aves des joints toriques. Grâce à la solidité du joint torique et à sa grande résistance, nous avions espoir que les attaches restent en place solidement.

Malheureusement lors d'une session de test au Cosanostra, à cause de toutes les vibrations subies, un des joints toriques s'est défait emportant avec lui une des attaches. De plus, un des sportifs présent sur place à cassé une des équerres lors d'une petite session. Le problème était qu'en partant avec les vis, l'équerre avait abimé le bois de la planche à tel point qu'il n'était plus possible de mettre des nouvelles vis. Nous avons du reboucher les trous avec de la colle araldite puis lisser après séchage pour qu'il nous soit possible de mettre des vis de nouveau et que cela reste bien solide.

Il nous fallait donc trouver un autre moyen de faire tenir cette boite sur cette planche. Nous avons eu l'idée de mettre des tiges filetées traversant la boite de part en part. Nous placions

d'un coté des écrous indesserables de sorte que ça bloque complètement et des écrous papillons avec des rondelles de l'autre. Il nous serait donc possible de mettre deux tiges traversant la boite dans sa largeur, ce qui nous seraient facile d'installer et d'enlever. Il y avait une interrogation cependant, lorsque nous allions mettre les tiges il ne fallait pas que ces dernières touches la carte. En effet, le cas échéant elle pourrait les détériorer. De plus, vu que les tiges sont en acier, nous avions peur qu'elles jouent le rôle d'antenne et donc qu'elle parasite le signal émis par le module Bluetooth. Nous avons réussi à mettre les tiges de sorte qu'elles ne touchent pas la carte. Nous avons également tropicalisé les tiges. Cela signifie que nous l'avons enduite d'un produit formant une fine pellicule rendant les tiges non conductrices. Ainsi nous n'avons eu aucun des problèmes auparavant redoutés.

Le système installé est donc solidaire de la planche, n'abime pas les composants de la carte, et n'empêche aucunement la bonne communication avec l'application.

CONCLUSION

XI. Conclusion

Ce projet est l'aboutissement de plusieurs mois de travail. Nous sommes arrivés à un produit qui satisfait notre groupe ainsi que nos parrains et suiveurs.

Les objectifs que nous nous étions fixés ont tous été réalisés mise à part la restitution en trois dimensions d'une figure donnée à n'importe quel moment, et ceux indépendamment du module.

Mais nous avons ajouté une dimension à ce projet : l'aspect social. Les réseaux sociaux sont désormais synonymes de vie quotidienne, il était donc important que l'on puisse partager nos expériences de ride avec des amis ou même d'autres sportifs dans le monde entier. Nous avons donc ajouté un bouton de partage sur notre application et une interface de statistiques permettant d'afficher les sessions actuelles du rider et les 5 meilleurs score effectués à partir de l'application.

On pourrait donc imaginer des compétitions entre amis ou inconnus à l'échelle mondiale ou à l'échelle de son jardin.

Ce projet est intéressant également pour sa modularité. Finalement la reconnaissance de figures est faite par l'algorithme. Vu que nous avons écrit ce dernier il est possible de modifier les noms et de comprendre d'autres mouvements du module pour que notre produit puisse détecter des figures en ski, en surf, snowboard et même les mouvements d'un gymnaste par exemple.

Il pourrait apporter une nouvelle dimension aux compétitions de sports extrêmes. En effet les juges pourrait avoir une certaine aide afin de pourvoir visualiser les figures plus facilement directement sur l'application et avoir une reconnaissance des figures déjà établie.

Xtrick pourrait donc servir tant sur un plan **récréatif** que sur un plan **professionnel** et ceux dans beaucoup de domaines. C'est ce qui fait la richesse de ce projet, et son avenir possible.

ANNEXES

XII. Annexes

A. Management de projet

Comme pour tout projet bien mené, l'organisation et la planification sont deux points essentiels. Cela nous a permis de nous fixer des objectifs à court, à moyen, et à long terme.

Nous avons tout d'abord mis en place un Gantt dans lequel nous fixons des objectifs sur le long terme. Nous y planifions l'ensemble des points importants.

En début de projet, il s'avérait être difficile de définir l'ensemble des tâches que nous aurons à effectuer, cependant nous avons réussi à définir des tâches ainsi que leur durée. Le Gantt nous a également permis de ne pas prendre de retard. En effet chaque tâche possède une date limite, et si celle-ci était dépassée, nous mettions deux ou trois membres du groupe sur la même tâche afin de ne pas accumuler de retard.

Le fait de planifier notre projet, et d'avoir réussi à terminer toutes les tâches que nous nous étions fixés nous sera aussi bénéfique dans le futur, dans notre vie professionnelle.

Voici le Gantt que nous avons suivi et au maximum respecté :

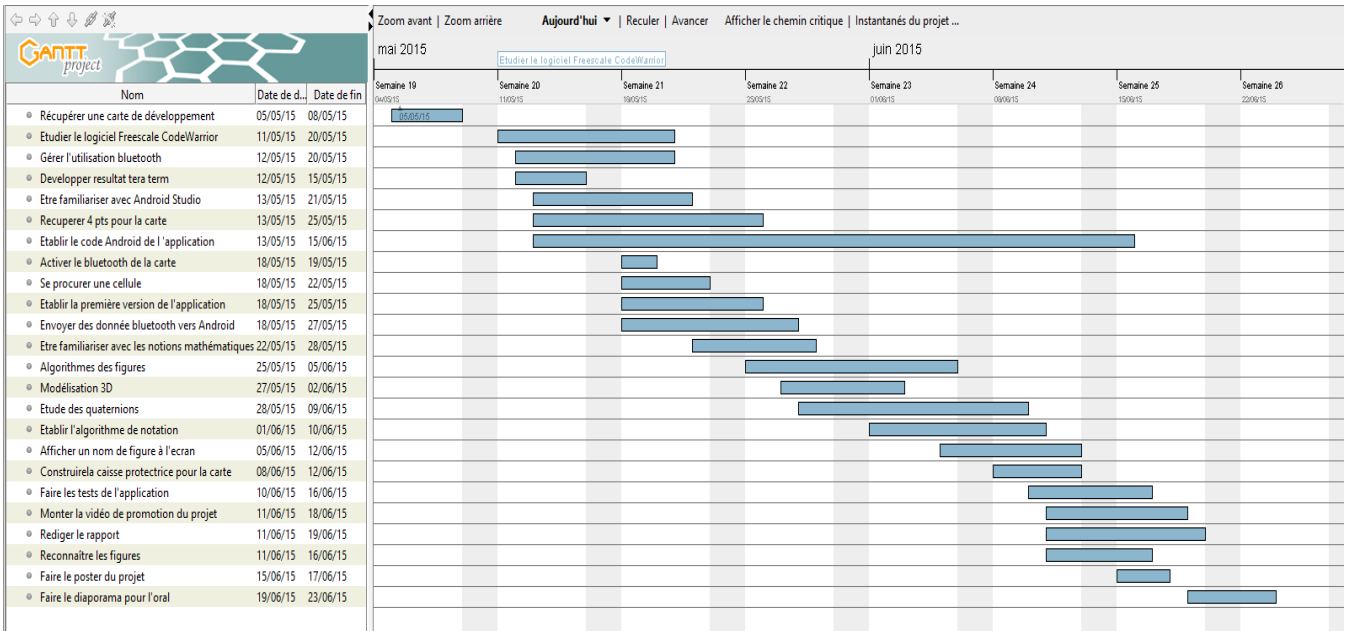


Figure 29 Diagramme de GANTT

De manière hebdomadaire, nous planifions les objectifs à court terme.

Chaque début de semaine, nous nous fixions des objectifs devant être finis pour le vendredi soir, si cela n'était pas fait, nous repoussions cette limite au dimanche soir afin de pouvoir démarrer de nouvelles tâches la semaine suivante. Cette manière de travailler a été efficace, nous avons réussi à ne pas prendre de retard et à terminer notre projet dans les temps.

Chaque fin de semaine nous établissions également des Highlights dans lequelles nous y inscrivons les points difficiles rencontrés durant la semaine, les différentes tâches accomplies, ainsi que les tâches à venir.

Highlights semaine du 11 au 17 mai 2015

Points durs	Actions accomplies	Tâches à venir
• Choisir la carte à utiliser • Utiliser le logiciel CodeWarrior • Prise en main du logiciel Android Studio • Mettre en place les algorithmes pour l'application • Récupérer les données d'accélération auprès du logiciel TeraTerm • Envoyer les données dues aux différents capteurs • Alimenter la carte par batterie afin d'établir la connexion Bluetooth	• Se renseigner auprès de nos professeurs et de notre contact Parrot afin de choisir ce qui convient le mieux • Rechercher et traiter les informations • Mise en place de plusieurs applications afin de se familiariser avec ce logiciel • Réussite dans l'allumage de la LED de la carte, première étape dans l'utilisation de la carte et du logiciel • Prise de renseignement et mise en place des premières applications, phase de découverte du logiciel • Mise en place de texte et de boutons sur une application Android • Récupération de la documentation	• Après réflexion, la carte Freescale en notre possession restera notre outil de travail pour ce projet. Continuer le travail effectué avec cette carte. • Continuer à se renseigner sur le logiciel jusqu'à être à l'aise • Réussir à gérer le composant Bluetooth de la carte, faire en sorte qu'il envoie les données • Continuer à se documenter et à développer nos compétences Android • Mettre en place l'application telle que nous l'imaginons actuellement, créer les différentes activités et boutons nécessaires • Traiter les informations et se lancer dans la mise en place de l'algorithme, nécessaire pour l'application Android • Traiter ces informations et

	• Visionnage des résultats sur l'interface du logiciel, les résultats semble correspondre car changement lorsque l'on bouge la carte • Utilisation du logiciel associé à la carte Freescale • Besoin d'une connections USB pour visionner les données en trois dimensions • Utilisation de jumper sur notre carte afin d'utiliser la batterie et le système Bluetooth	en déterminer la position (intégrer deux fois) • Récupérer ces données non plus en série mais par Bluetooth et les envoyer vers l'application Android

Highlights semaine du 18 au 24 mai

Highlights :

- Gérer le Bluetooth
- Gérer les API : CodeWarrior et Android Studio
- Créer l'application Android
- Créer et manipuler un élément 3D
- Avoir un rendez-vous avec notre contact Parrot
- Décider d'implémenter l'algorithme sur le téléphone et non sur la carte

Points Durs	Actions Accomplies	Tâches a venir
• La gestion du bluetooth a été très compliquée. La documentation sur Internet est mal faite et très éparpillée rendant le travail de recherche assez compliqué (connexion entre le CPU et le module bluetooth par exemple) • CodeWarrior est assez difficile à maîtriser. Il y' a beaucoup de configurations à faire. En fait dès que l'on souhaite ajouter un élément de code ou un hardware, il faut le configurer et trouver la bonne configuration. Cela rajoute un gros travail de recherche en plus de celui du code (Problème que l'on a pas	• Faire les branchements entre le module bluetooth et le CPU • Activer le bluetooth sur la carte • Reconnaître la carte sur un ordinateur portable et sur un téléphone • Envoyer des données via le bluetooth • Manipuler Android Studio • Manipuler CodeWarrior et être capable de coder sur le processeur • Coder une application permettant de calculer le score, de paramétrer le bluetooth, de reconnaître la carte et d'afficher le meilleur score • Modéliser un	• Continuer à améliorer l'application • Se filmer pour les vidéos de présentation (Journée des Projet et Parrot) • Lire le code source de l'application mobile Freescale afin de comprendre comment sont envoyées les données par la carte et comment l'application s'occupe des ces données • Faire notre propre algorithme pour le nombre de points par figure et le type de figure. Utile également pour la restitution en 3D de la figure

avec Android, point qui nous a permis de faire le choix d'implémentation de l'algorithme • Modélisation en trois dimensions. Code très compliqué. Forme faites à partir de triangle, cela implique qu'il est impossible pour l'instant de modéliser un skate	objet 3D et pouvoir le manipuler	

Highlights semaine du 25 au 31 mai

Highlights :

- Gérer le Bluetooth
- Développer notre code sur Android Studio
- Analyser le code de l'application et en tirer les informations importantes

Points Durs	Actions Accomplies	Tâches à venir
• Le code de l'application est très fourni, il y' a beaucoup d'informations, de données. Il faut déterminer lesquelles sont utiles dans notre projet. • Choisir si nous devrions plutôt incorporer notre code personnelle dans l'application Freescale, ou au contraire incorporer le code Freescale dans notre code personnel. • Comprendre de nouvelles notions mathématiques jusqu'ici inconnues, à l'exemple des quaternions, elles seront essentiels dans l'interprétation de la figure réalisée par le rider.	• Adapter le code de l'application Freescale à notre utilisation (1ère version). Seul les boutons et fonctionnalités utiles doivent être dorénavant accessible depuis l'application. • Mise en place de l'image d'un skate au lieu d'une carte Freescale, plus attractif, plus "visuel" . • Meilleur compréhension du code de la part de l'équipe.	• Continuer à développer notre application. • Continuer à trier les données de l'application Freescale afin de rendre notre application la plus simple et complète possible. • Établir notre propre algorithme afin de calculer le nombres de points et afin de définir la figure réalisée.

Highlights semaine du 1 au 7 juin

Highlights :

- Établir l'algorithme permettant la reconnaissance des figures
- Compléter le code de l'application Android en récupérant les données des capteurs

Points Durs	Actions accomplies	Tâches à venir
• Établir un algorithme permettant à l'aide des valeurs des quaternions de déterminer l'axe de rotation et la valeur de l'angle Incorporer cet algorithme dans notre application afin de pouvoir définir la figure qui a été réalisée • Choisir entre le fait d'utiliser les quaternions ou les coordonnées X,Y,Z pour définir les figures, choix finalement porté sur quaternions car plus fidèle au mouvement effectué, plus réaliste	• Nous avons trouver où sont les données dans l'application et nous avons réussi à afficher les valeurs des quaternions. • Compréhension de la méthode permettant de définir l 'axe de rotation et la figure à l'aide des quatre composantes des quaternions. • Garder un nombre d'échantillons nécessaires pour le traitement de l'algorithme, une centaine par seconde, soit 1 échantillons tous les 10 millisecondes. • Remplissage d'un tableau contenant toutes les valeurs des quaternions	• Mettre en place l'algorithme dans notre application • Gérer les différentes figures dans notre application • Mettre en place le widget facebook dans notre application • Récupérer un boîtier pour notre carte afin de la placer sous le skate • Améliorer notre application en la rendant la plus attractive possible.

Highlights semaine du 8 au 14 juin

Highlights:

- Établir la reconnaissance des figures par l'application
- Mettre en place l'algorithme de notation des figures
- Développer et intégrer l'outil Facebook dans notre application
- Mise en place et teste du boîtier de protection de la carte sur la carte

Points Durs	Actions accomplies	Tâches à venir
• L'algorithme de reconnaissance des figures a été difficile à définir. La reconnaissance des figures est effectuée en fonction du nombre de rotation autour des 3 axes X,Y,Z. Différents algorithmes ont été testés, des essais ont été effectués et la dernière version de l'algorithme semble la plus stable même s'il arrive que l'algorithme se trompe dans le nom de la figure réalisée • L'algorithme de notation des figures à également été mis en place. Chaque figure doit avoir une note différente en fonction de la difficulté, mais également en fonction du "pop" • Incorporer le Sdk Facebook dans notre application permettant de partager le score ainsi que le nom des figures effectuées.	• Mise en place des algorithmes de reconnaissance des figures et de notation dans notre application • Réalisation d'un nouveau design de l'application • Réalisation d'un boîtier afin de protéger la carte des chocs sur le skate. • Garder dans notre code Android que les classes et méthodes qui nous sont utiles	• Finir l'intégration du composant Facebook dans l'application • Essayer d'améliorer la stabilité de reconnaissance des figures • Réaliser un poster pour la journée des projets • Compléter le rapport à rendre lors de la remise du projet • Monter la vidéo de promotion de notre projet • Préparer la présentation oral du projet

B. Code partiel

C. Bibliographie

1. Document Freescale

- **https://developer.mbed.org/teams/Freescale/wiki/frdm-kl25z-pinnames**
- **https://ece.uwaterloo.ca/~ece222/Freescale/FRDM-KL25Z-Pinouts-Rev1.pdf**
- **https://developer.mbed.org/platforms/KL25Z/**
- **http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-KL25Z**
- **http://cache.freescale.com/files/sensors/doc/data_sheet/XSFLK_DS.pdf?fasp=1&WT_TYPE=Data%20Sheets&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=pdf&WT_ASSET=Documentation&fileExt=.pdf**
- **http://cache.freescale.com/files/sensors/doc/support_info/FRDM-FXS-MULTI-B.pdf?fasp=1&WT_TYPE=Supporting%20Information&WT_VENDOR=FREESCALE&WT_FILE_FORMAT=pdf&WT_ASSET=Documentation&fileExt=.pdf**
- **http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=FRDM-FXS-MULTI-B**
- **http://cache.freescale.com/files/microcontrollers/doc/fact_sheet/FREEDEVPLTSNSRFS.pdf**

2. Facebook

- **https://developers.facebook.com/**
- **https://developers.facebook.com/docs/android**
- **https://developers.facebook.com/quickstarts/?platform=android**
- **https://developers.facebook.com/docs/sharing/android#message**

D. Remerciements

Dans ce projet nous avons été soutenus par des personnes venant de tous secteurs (ingénierie, corps professoral, secteur privé). Nous tenons donc à remercier :

- Mr Guillaume Beranger pour son aide précieuse et son investissement dans notre projet.
- Mr Rostom Kachouri dont les conseils nous ont permis d'aboutir à un tel projet.
- Mr Thierry Grandpierre pour le kit Freescale.
- Mr Sylvain Dupont-Legendre qui a été notre suiveur et chef de projet. Il nous a permis de structurer notre avancement et motivé chaque semaine.
- Mme Corinne Berland pour sa grande disponibilité et son aide précieuse durant toute la durée de notre projet.
- Mr Michel Aubry qui s'est investi dans la conception d'une protection solide de la carte.
- Le Cosanostra Skate Park (8 Rue du Tir, 77500 Chelles) de nous avoir laissé filmer librement, et tester notre produit dans ses locaux.