Étudiants IR
Hammouma Youssef
Botella-broc Yohann

BTS SN IR 2018-2019

Dossier Technique

Projet TTPA

(Table Tennis Performance Analyser)

campus La Salle

Table des matières

Présentation générale

Présentation générale du système supportant le projet

Le tennis de table, comme tout sport de raquette, est un sport complet tant sur le plan physique que mental. Il requiert un entraînement rigoureux et régulier afin de pouvoir se perfectionner. Les joueurs comme les entraîneurs ont besoin de connaître les performances en jeux et lors des phases d'entraînement. De nombreux entraîneurs ont constaté que le taux de réussite et la précision du coup sont les clefs de la réussite. Un système, qui mettrait en évidence les résultats d'un joueur, lui permettra d'optimiser au mieux ses performances.

Analyse de l'existant

La « Waldner » est une table de ping-pong de nouvelle génération qui a été conçue pour intégrer un système informatique de pointe.

La table dispose d'une surface sensible au contact humain et à celui des balles de ping-pong. La table est également d'un noyau informatique de type Mac Pro 12 qui observe le jeu. Sous la surface, on trouve deux Quad-Core Intel Xeon cadencés à 2.4GHz, une carte graphique ATI Radeon HD 5870, une connexion Wi-Fi. Les dimensions de la table sont de 2,74 m de long, 1,52 m de large, 76 cm de haut et 5 cm d'épaisseur.
La surface, qui est un écran, est faite d'acrylique très lisse et à faible coefficient de frottement. L'écran a une résolution de 2880 x 1800 pixels. Après et pendant le match, de nombreux éléments statistiques peuvent être obtenus sur la table. Les joueurs peuvent par exemple comparer les zones d'impact de balles (où chaque impact est matérialisé par un cercle et faisant mention d'informations comme la vitesse et la trajectoire. Toutes les données sont stockées et peuvent être visualisées en en temps réel sur des périphériques comme l'iPad et l'iPhone.

Expression du besoin

La réalisation d'une application mobile connectée à différents appareils dans le but de réaliser une séance d'entraînement de tennis de table.

Le projet consiste à pouvoir paramétrer sa séance d'entraînement comme bon nous semble à l'aide de l'application mobile qui communique avec un robot lanceur de balle ainsi qu'une table équipée de capteurs d'impacts. Les statistiques en temps réel sont affichées sur un écran.

Ce projet est mené en partenariat avec le club de tennis de table PPC Sorgues, après certains essais le constat est qu'un joueur de type "pro" n'utilise pas l'écran car il n'en a aucune utilité précise néanmoins il s'est avéré que l'écran est un outil ludique pour les jeunes joueurs qui peuvent se défier et observer les statistiques de leurs camarades lorsqu'ils attendent leur tour de jeu.

Présentation du projet

Le système devra :

- Permettre de lancer une séance d'entraînement
- Régler le robot lanceur de balles
- Afficher un retour visuel au joueur
- Archiver les séances d'entraînement
- Afficher les statistiques de la séance

Le système sera équipé de :

- 10 capteur de vibration piézoélectrique
- 1 robot lanceur de balles
- 1 ecran Raspberry Pi
- 1 tablette mobile

Exigences

Le système TTPA devra permettre :

- La saisie du nom du joueur
- Le paramétrage du robot lanceur de balles
    - la fréquence d'envoi des balles
    - le nombre de balles à jouer pour la séance
    - les effets et la puissance des balles
    - la rotation du robot lanceur
- De choisir la position du robot ainsi que la zone à atteindre sur la table

Pour démarrer une séance d'entraînement, il est nécessaire d'être connecté en Bluetooth avec au moins un des deux modules : module de distribution de balles et/ou module de détection des impacts. On pourra visualiser sur l'interface l'état de connexion de chaque module. Le joueur ou l'entraîneur pourra mettre en pause une séance, la reprendre ou l'arrêter.

Synoptique du système TTPA

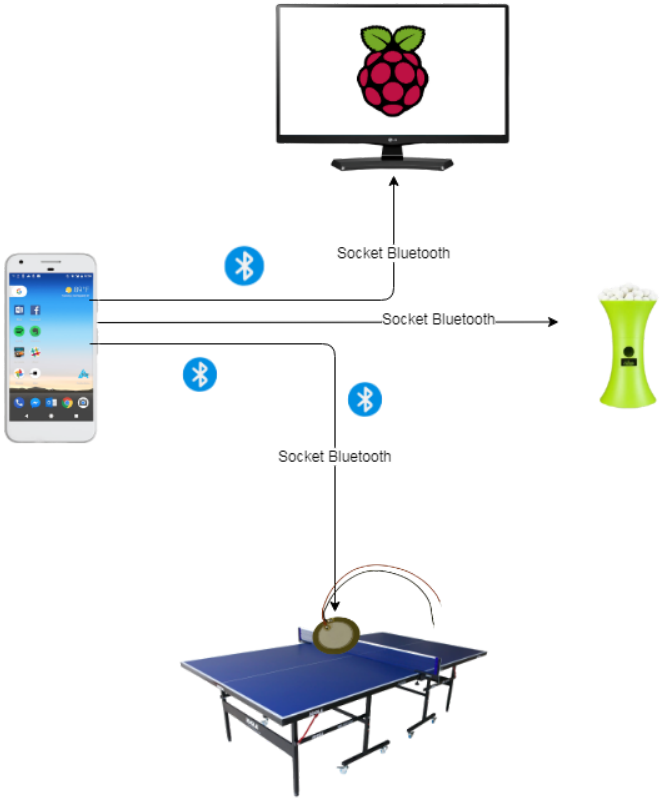
Le système TTPA est donc décomposé en quatre modules :

- **Module de gestion de séance (Mobile-TTPA)** : le joueur et/ou l'entraîneur paramètre et lance la séance d'entraînement à partir d'une application sur un terminal mobile (sous Android)

- **Module de distribution de balles (Robot-TTPA)** : le robot lanceur envoie des balles, en suivant un rythme fixé par le joueur, avec des effets différents et réalistes.

- **Module de détection des impacts (Table-TTPA)** : la table est équipée de capteurs permettant d'identifier la zone impactée par la balle renvoyée par le joueur ;

- **Module de visualisation de performance (Écran-TTPA)** : l'entraîneur peut visualiser en "temps réel" le rythme de jeu, la précision, le pourcentage de réussite.



Le schéma ci- dessus explique le fonctionnement du système, le terminal mobile communique avec les différents périphériques à l'aide du bluetooth. Chaque périphérique (terminal mobile y compris) possède une socket, elles communiquent toutes entre elles.

Equipe de projet TTPA

Étudiants chargés du projet

Option EC :

- ❏ Etudiant 1 : Cannon Ludwing
- ❏ Etudiant 2 : Monteiro Goeffrey

Option IR :

- ❏ Etudiant 3 : Botella-Broc Yohann
- ❏ Etudiant 4 : Hammouma Youssef

Répartition des tâches

Etudiant 1: Cannon Ludwing

Module : Module de détection des impacts (Table-TTPA)

- ➔ Détection des impacts de la balle sur la table côté « joueur »
- ➔ Identification de la zone frappée côté «distributeur »
- ➔ Lancer une séquence d'acquisition
- ➔ Dialoguer avec le terminal mobile

- Installation :
  - Les capteurs de détection d'impacts
  - le module d'acquisition d'impact et de transmission de données

- Mise en oeuvre :
  - Les différents capteurs
  - la carte d'acquisition d'impacts
  - étalonner et valider la carte d'acquisition
  - mettre en forme les mesures validant l'acquisition

- Configuration : Des entrées/sorties du µc en fonction de la carte d'acquisition
- Réalisation :
    - carte électronique assurant l'acquisition des impacts
    - programme de détection d'impacts côté "Joueur" et "Distributeur"
    - diagrammes SysML

- Documentation :
    - dossier technique et les documents relatifs au module
    - Un guide de mise en route et d'utilisation du module

Etudiant 2: Monteiro Goeffrey

Module : Module de distribution de balles (Robot-TTPA)

➔ Envoi des balles sur toute la table avec suivant un rythme fixée par le joueur
➔ Imprimer des effets réalistes et différents à la balle (coupés/liftés, sans effet)
➔ Détection d'absence et/ou de bourrage de balles
➔ Signaler au joueur l'instant d'envoie d'une balle
➔ Afficher les données d'une séance
➔ Communiquer via une liaison sans fil

- Installation :
    - Le système de distribution de balle

- Mise en oeuvre :
    - L'envoie de balles cadencées, la rotation imprimée aux balles
    - l'oscillation de la tête
    - la communication sans fil avec le la balle (coupés/liftés, sans effet) système de gestion et d'affichage de la séance

- Configuration : Des entrées/sorties du µc en fonction des modules utilisés, la balle transmission sans fil
- Réalisation :
    - programme assurant le rythme et les effets à imprimer aux balles
    - diagrammes SysML

- Documentation :
    - dossier technique et les documents relatifs au module
    - Un guide de mise en route et d'utilisation du module

Etudiant 3: Botella-Broc Yohann

Module : Module de visualisation de performance (Écran-TTPA)

- ➔ Afficher un écran d'accueil
- ➔ Visualisation de l'impact de balle dans les zones de la table
- ➔ Visualisation des données de la séance en temps réel (le pourcentage de balles par zones et nombre de pourcentages de balles bonnes)
- ➔ Visualisation des statistiques en fin de séance

- Installation :
  - Environnement de développement

- Mise en oeuvre :
  - La liaison sans fil
  - Raspberry Pi

- Configuration :
  - La liaison sans fil
  - Écran en mode "kiosque"
- Réalisation :
  - diagrammes UML
  - IHM du module
  - Code source de l'application

- Documentation :
  - dossier technique et les documents relatifs au module
  - Un guide de mise en route et d'utilisation du module

Etudiant 4: Hammouma Youssef

Module : Module de gestion de séance (Mobile-TTPA)

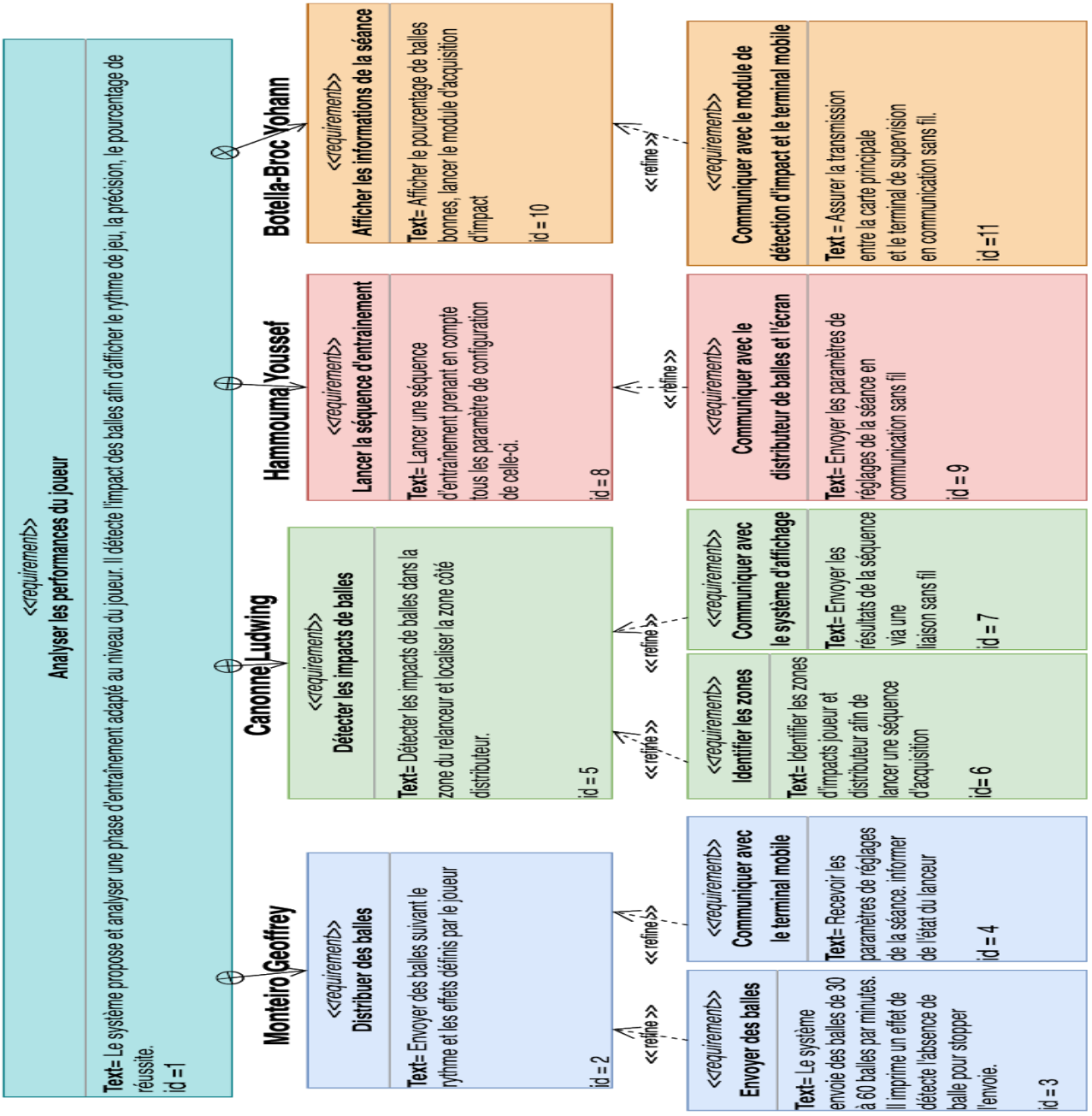
- ➔ Créer un profil de joueur
- ➔ Paramétrer une séance d'entraînement
- ➔ Gérer une séance d'entraînement
- ➔ Enregistrer les données des séances
- ➔ Consulter l'historique des séances d'un joueur
- ➔ Purger les séances
- ➔ Dialoguer avec les modules

- Réalisation :
  - diagrammes UML
  - IHM du module
  - Code source de l'application

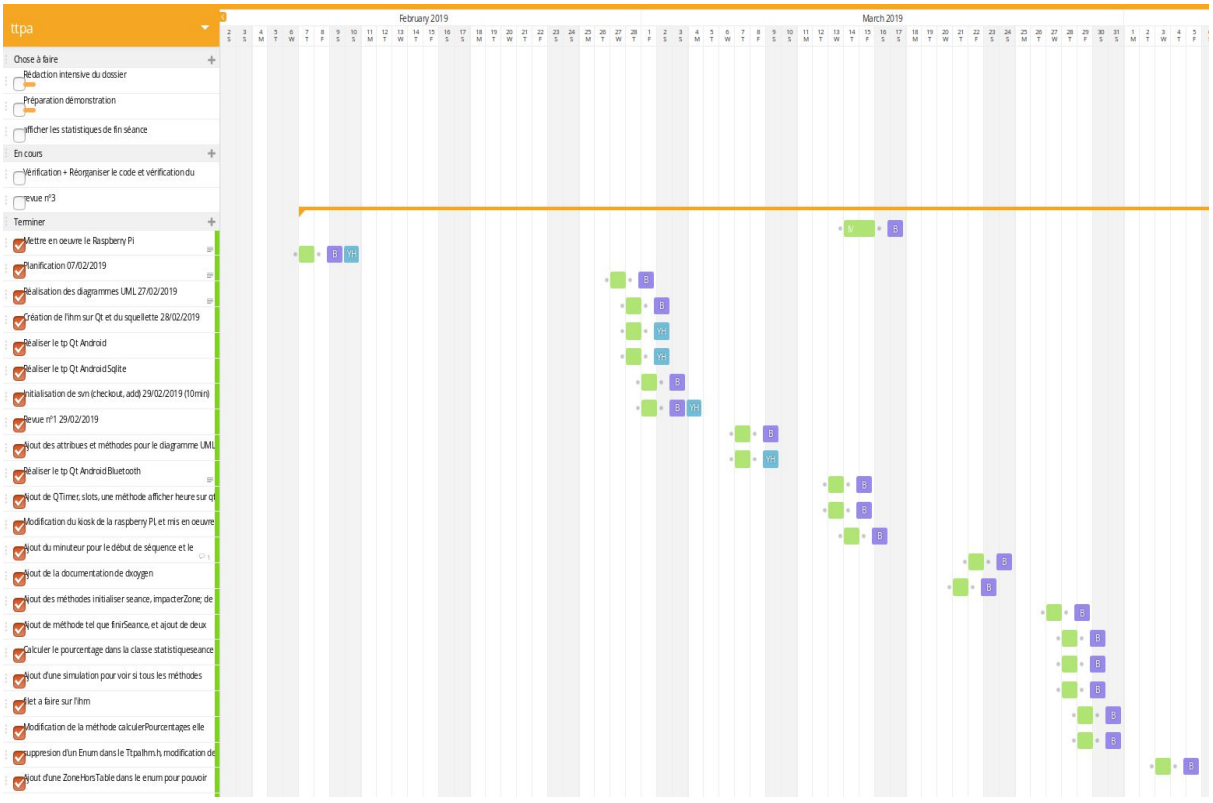
- Documentation :
  - dossier technique et les documents relatifs au module
  - Un guide de mise en route et d'utilisation du module

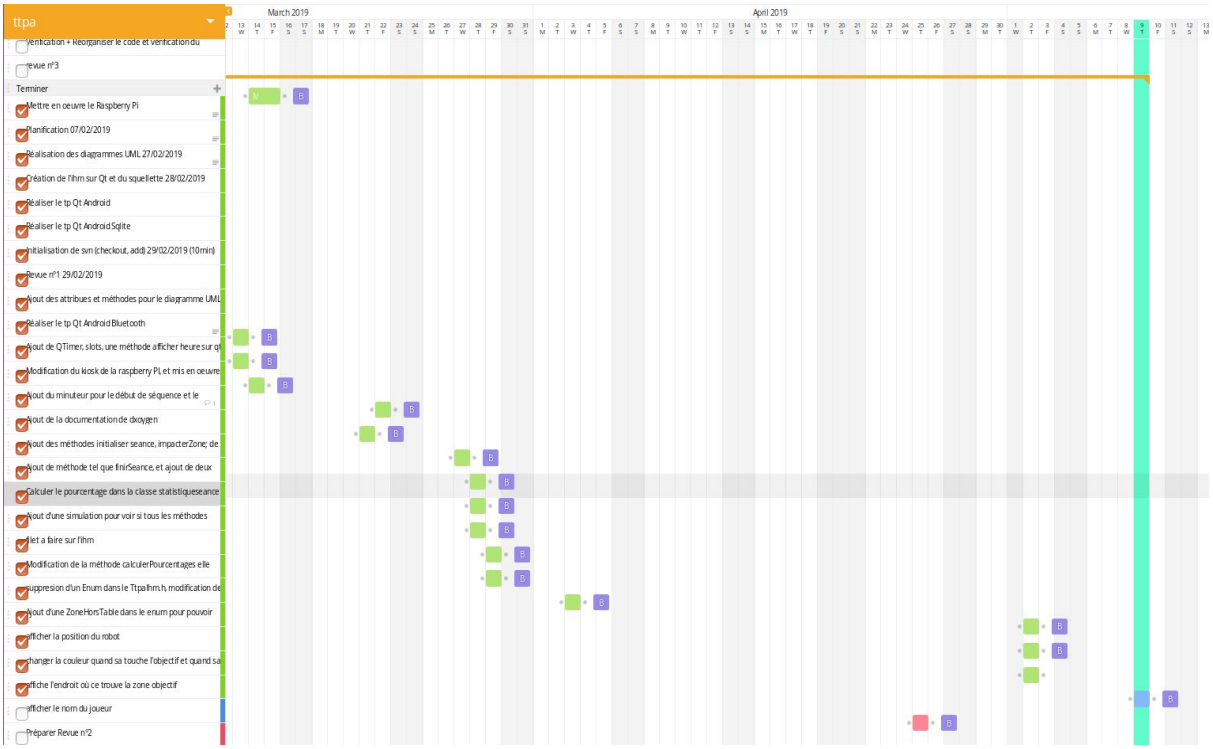
Etude du système

Diagramme des d'exigences

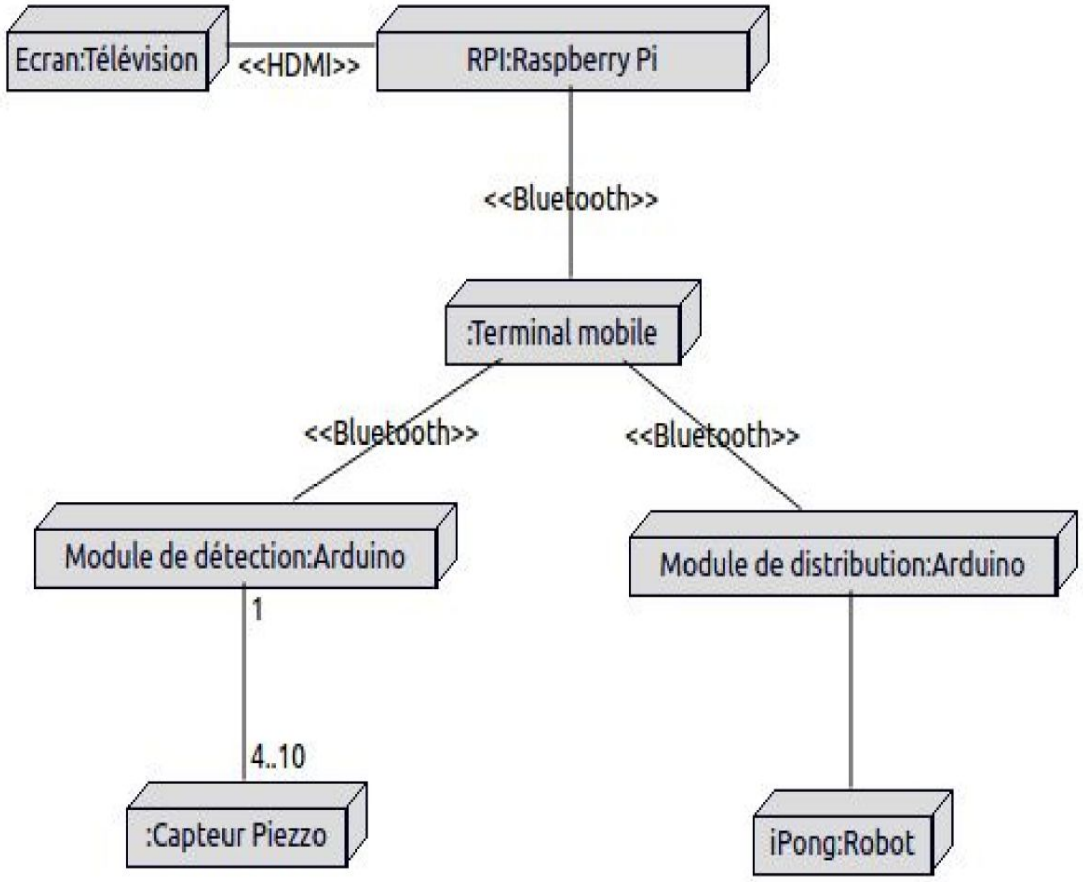


Planification des tâches





Déploiement du système



Contraintes matérielles et logicielles

Matériels

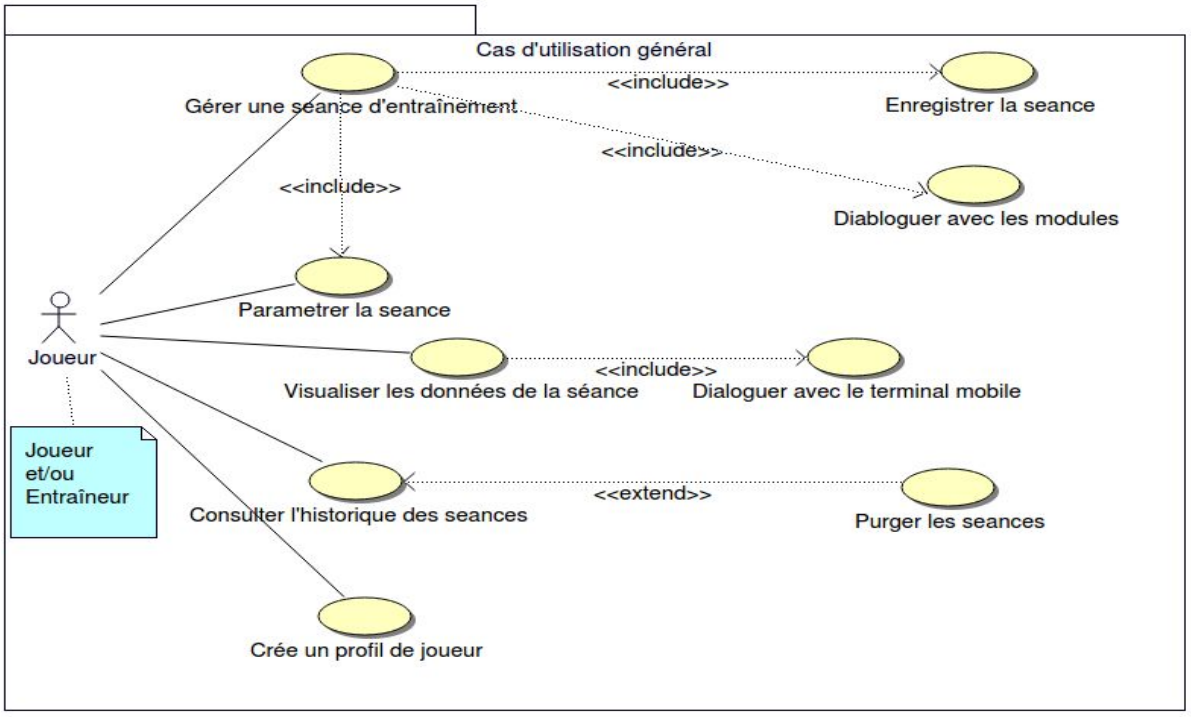
- Capteurs piézoélectriques
- Robot

Logiciels

- Qt/Qml 5.10.1
- Subversion 1.9.3
- Bouml 7.8 + Draw.io
- Doxygen 1.8.11
- Android 7.0
- Gestion de projet : Trello + TrelloGantt

Diagramme des cas d'utilisation

Le diagramme de cas d'utilisation du Terminal Mobile présent ci-dessous présente la vision du joueur ou d'un entraîneur par rapport au système :



Le **joueur** (ou l'**entraîneur**) aura la possibilité de Gérer une séance d'entraînement (Créer et démarrer une séance d'entraînement) en l'ayant préalablement paramétrée ou non (paramètres par défaut). Chaque séance sera enregistrée localement dans le terminal mobile. Ensuite, après avoir joué sa première séance par exemple, il pourra Consulter l'historique des séances et enfin Purger l'historique des séances.

Pour effectuer une séance, on aura besoin de dialoguer avec les modules en Bluetooth.

L'entraîneur peut visualiser en continu les données (pourcentage de balles par zones et le nombre de balles dans la zone à atteindre) d'une séance d'entraînement. Pour cela, le sous-système a besoin de récupérer les trames émises par le terminal mobile. Les données visualisées sont :

- Le pourcentage de balles dans chaque zone en comptant les balles hors table
- Le nombre et pourcentage de balles hors table (une balle n'ayant pas touché une zone)
- Le nombre de balles jouées par le robot sur le nombre de balles pour la séance
- L'heure ainsi que le temps de la séance depuis son lancement
- Le nom du joueur

Une communication entre le sous-système ECRAN-TTPA et le terminal mobile basée sur un protocole spécifique est nécessaire.

Partie Botella-Broc Yohann (Étudiant IR)

Table des matières

- Rappel du Besoin Initial

Ce système doit fournir une analyse de performance d'un joueur de tennis de table face à un robot lanceur pendant une séance d'entraînement. Le joueur a pour objectif de relancer la balle dans une "zone" précise de la table. La table est découpée en 7 "zones", cette précision est donc limitée à ces cases. La performance du joueur pendant la séance est basée sur le nombre de balles renvoyées dans la zone à atteindre. Le robot lanceur est paramétrable afin d'obtenir une variété de balles réalistes.

Ce projet est en partenariat avec : Ping Pong Club Sorgues, basé sur un besoin réel du club pour leurs entraînements.

- Objectifs

Module de visualisation de performance (ECRAN-TTPA) : l'entraîneur peut visualiser en "temps réel" le rythme de jeu, la précision, le pourcentage de réussite. Sur le terminal mobile, l'application doit permettre de créer et démarrer une séance d'entraînement.

Afin de pouvoir afficher toutes les informations essentielles pour une séance d'entraînement, il est nécessaire de communiquer avec le terminal mobile.

La partie du projet dont je me suis occupé concerne le Module de visualisation de performance. Il doit permettre :

- d'afficher un écran d'accueil
- d'afficher le nom du joueur (si existant), la durée écoulée de la séance
- de visualiser l'emplacement du robot et de l'objectif de la séance
- de visualiser les statistiques pour chaque zone
- de visualiser l'impact de balles dans les zones de la table en temps réel (en vert pour un impact dans la zone objectif sinon en rouge)
- de visualiser le nombre de balles jouées et le nombre total de coups dans la zone à atteindre
- de visualiser les statistiques finales de la séance
- de dialoguer avec le terminal mobile

- Description structurelle du système

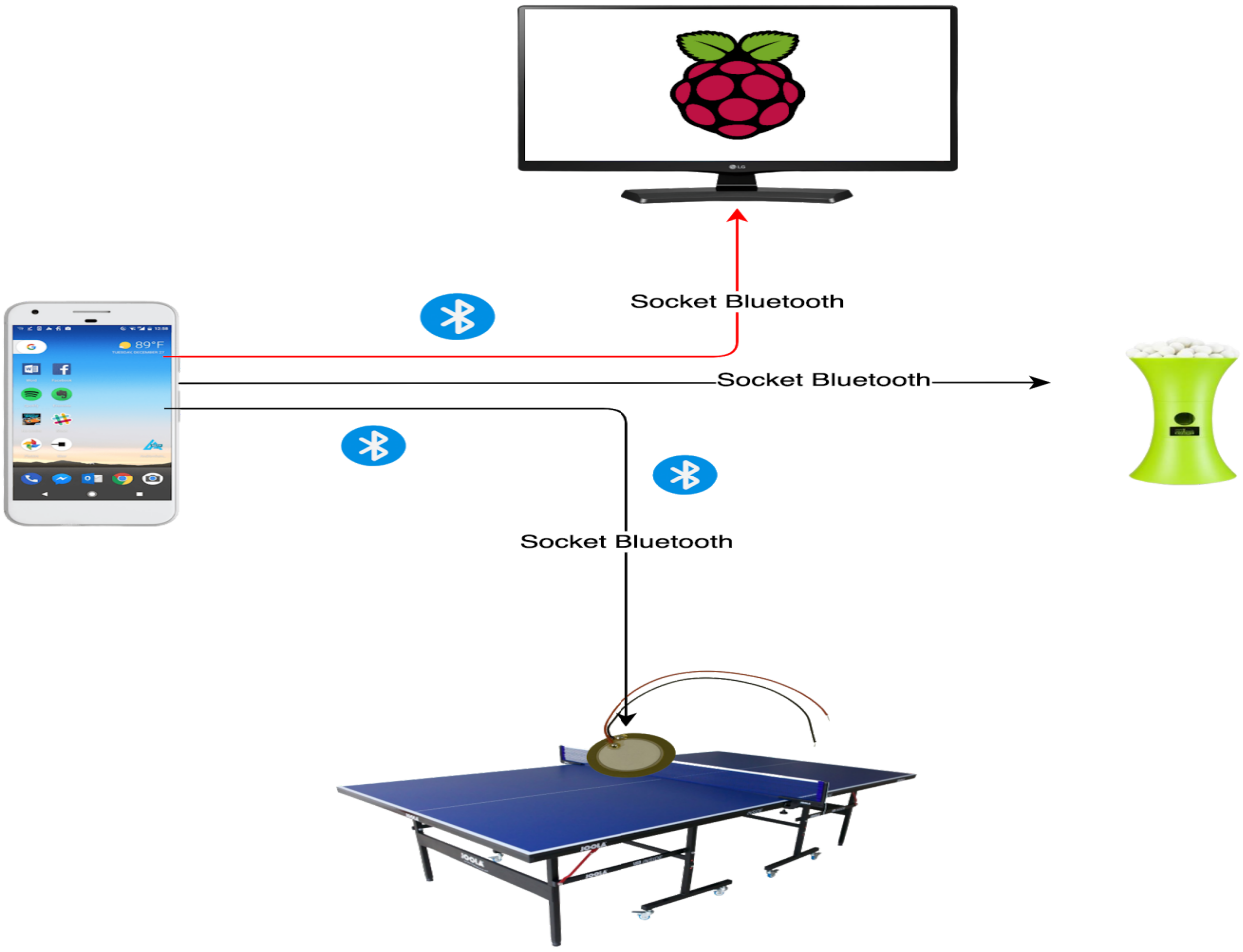


Figure 1: Description structurelle du système

Le terminal mobile communique en Bluetooth avec la Raspberry Pi 3, la table et le robot. Toutes les informations passent par un protocole TTPA.

Remarque : un(e) socket est une interface logicielle avec les services du système d'exploitation, grâce à laquelle un développeur exploite de manière uniforme les services d'un protocole réseau.

- Les Zones de la table TTPA

Les différentes "zones" de la table sont :

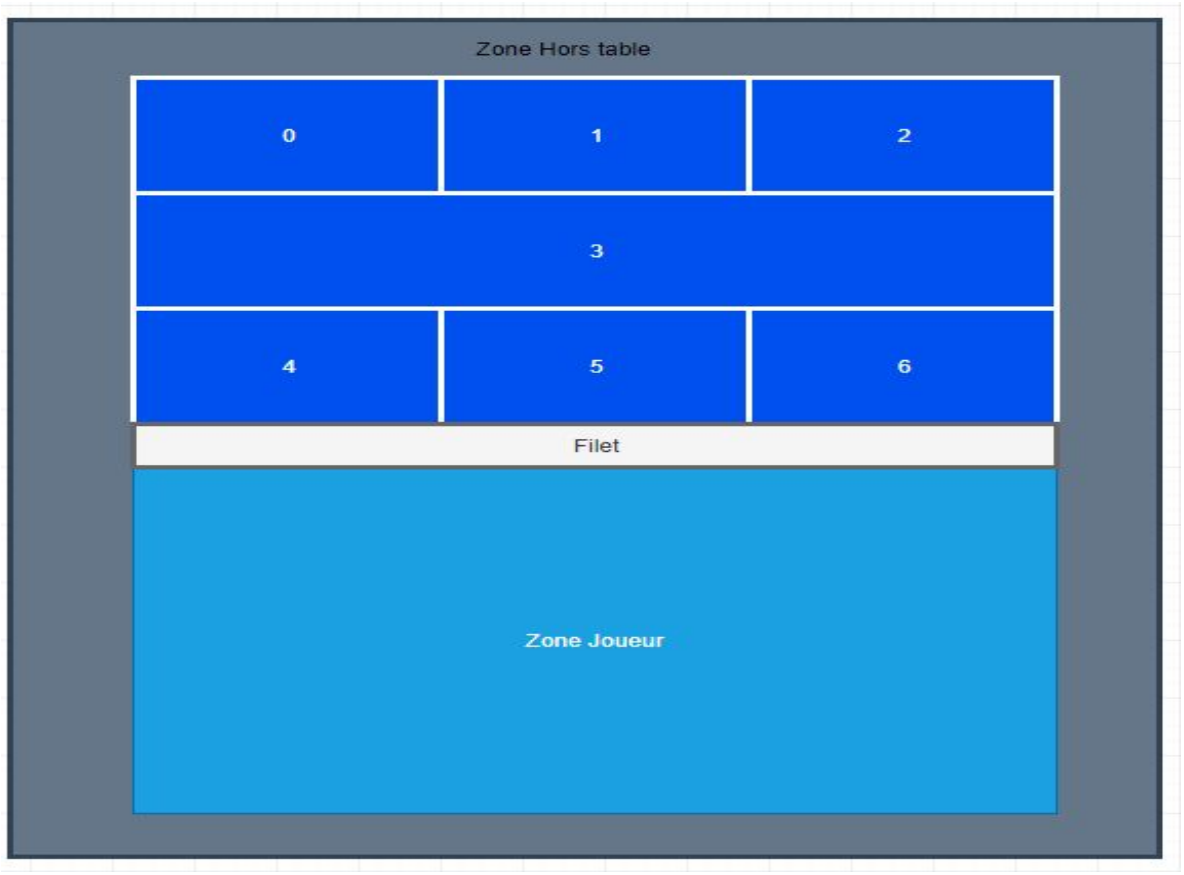


Figure 2: Les différentes zones de la table

- **Le haut (zone 0 à 6)** est la partie de la table comptabilisant les balles dans les zones pour les statistiques, il s'agit du côté où la **zone objectif** et le **robot lanceur** sont placés (celui-ci peut aussi être désactivé et donc absent).

- **Le milieu (zone 3)** est une partie de la table qui regroupe 3 capteurs piézoélectriques (zone peu intéressante pour des joueurs expérimentés)

- **Filet** est la partie central de la table, c'est un obstacle entre les deux côtés

- **Le bas la zone joueur** est le côté du **joueur**, ou seul l'impact de balle est détecté à des fins de vérification et de synchronisation. Cela permet de vérifier si la balle envoyée par le robot peut être jouée.

- **La zone hors table**

- Tâches

Installation :

L'environnement de développement

Réalisation:

Dialoguer avec le terminal mobile
Les diagrammes UML, L'IHM du module, Le code source de application

Documentation :

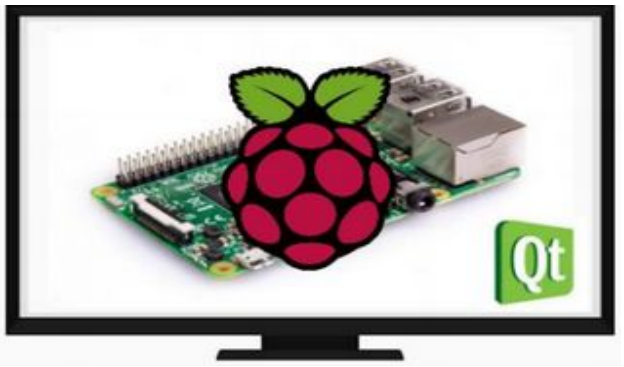
Le dossier technique et les documents relatifs au module,
Un guide de mise en route et d'utilisation du module

Mise en oeuvre :

La liaison sans fil, la Raspberry Pi 3

Configuration :

La liaison sans fil, l'écran en mode "kiosque"



- **Outils de développement**

Description	*Version*
Système d'exploitation du poste de développement	Linux 4.8.0 (Ubuntu 16.04)
Planification	Trello gantt + Trello
Journal de bord	RoadBook (Google Drive)
Diagrammes UML	BOUML 7.8
Gestion de versions	RiouxSvn (Subversion) 1.9.3 (r1718519)
Documentation de code	Doxygen 1.8.11
Langage utilisé	C++ / avec le framework Qt 5.11.2
Raspberry Pi 3	Raspbian Linux version 4.14.84-v7+

Figure 3: tableau des outils de développement

-

- Diagrammes de déploiement

Grâce à la vue d'implémentation statique du système, le module ecran-ttpa nécessite d'être raccordé à une télévision en HDMI pour afficher les informations ainsi que de communiquer en Bluetooth avec le terminal mobile.

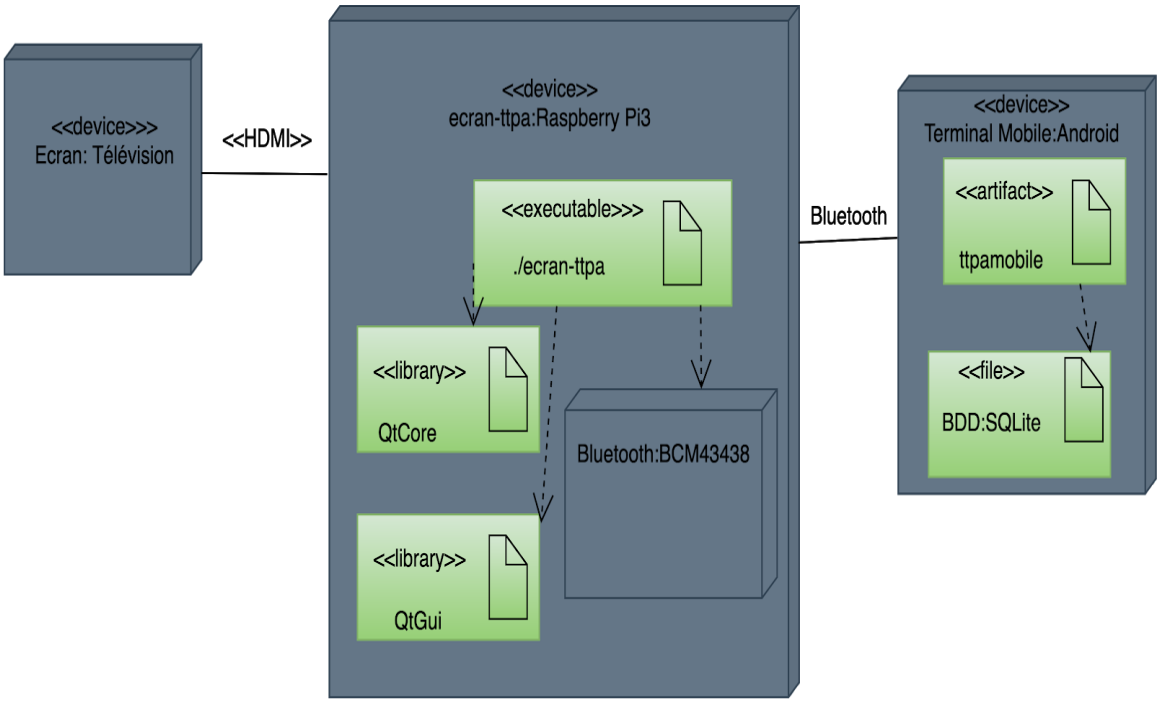


Figure 4: déploiement du système ecran-ttpa

- Spécification du Raspberry Pi3

Raspberry PI 3B	*Caractéristiques*
Système d'exploitation	Raspbian GNU/Linux 9.1 (stretch)
Processeur	Quad Core 1.2GHz Broadcom BCM2837 ARMv8 CPURAM1GB
Stockage	Micro SD
Sans-Fil	BCM43438 LAN et "Bluetooth Low Energy" (BLE)
Connectiques	40-pin GPIO, 4 ports USB 2, HDMI, CSI, DSI

Figure 5: tableau des caractéristiques de la raspberry pi 3.

- Le Bluetooth (partie Sciences Physiques)

Le Bluetooth : le réseau de proximité

En reliant les objets entre eux (ordinateur et casque sans fil, smartphone et voiture, ..) le Bluetooth établi une connexion sécurisée de proximité. Bien que cela puisse apparaître comme une des limites de ce protocole : il faut que les appareils soient proches les uns des autres, c'est en réalité son principal argument en regard de la sécurité: l'interception du signal ne peut pas se faire à distance.
Le système Bluetooth opère dans les bandes de **fréquences** ISM* (Industrial, Scientific and Medical) 2,4 GHz dont l'exploitation ne nécessite pas de licence vu la faible puissance d'émission et le risque faible d'interférences. Cette bande de fréquences est comprise entre 2 400 et 2 483,5 MHz. Un transceiver à **sauts de fréquence** est utilisé pour limiter les interférences et l'atténuation. Par ailleurs, lorsque plusieurs dispositifs sont connectés, ils le sont sous la forme d'un pico réseau, et toujours via une relation de type maître / esclave. Ces derniers sont au maximum de 7 et ne peuvent pas communiquer entre eux.
La nouvelle version du protocole, **Bluetooth LE** (pour Low Energie), nécessite beaucoup moins d'énergie que le WiFi, et les développeurs travaillent sur la possibilité de faire un réseau maillé, ce qui permettrait à plusieurs composants de communiquer entre eux.

Tableau de correspondance pour le numéro de pièce entre Broadcom et Cypress

Numéro de pièce Broadcom	Numéro de pièce Cypress
BCM43438	CYW43488

Caractéristiques :

- ➢ simple bande 2,4 GHz.
- ➢ Prise en charge des débits de données Broadcom TurboQAM® à 2,4 GHz (256-QAM) et de la bande passante du canal de 20 MHz.
  La **modulation d'amplitude en quadrature**(QAM)
- ➢ Prise en charge du contrôle de parité à faible densité (LDPC) de Tx et Rx pour une portée et une efficacité énergétique améliorées.

➢ Conforme à la spécification Bluetooth Core version 4.1 avec des dispositions pour la prise en charge des spécifications futures.
(version 4.1 créé en 2013 connexion d'appareils multiples sur un seul accès pour la sortie du **LTE**(*Long Term Evolution)*.)

➢ Récepteur FM: bandes FM de 65 MHz à 108 MHz; prend en charge les normes RDS (European Data Data System) et les normes RBDS (North Broadcasting Data System) d'Amérique du Nord.

➢ Débit théorique supérieur porté à 25 Mbit/s pour la version 4.1, porté de 60 m

Sécurité :

➢ Prise en charge de WPA et WPA2 (Personnel) pour un cryptage et une authentification puissants.

➢ AES dans le matériel WLAN pour un cryptage des données plus rapide et une compatibilité IEEE 802.11i(norme Wi-Fi).

- Schéma fonctionnel du CYW43438

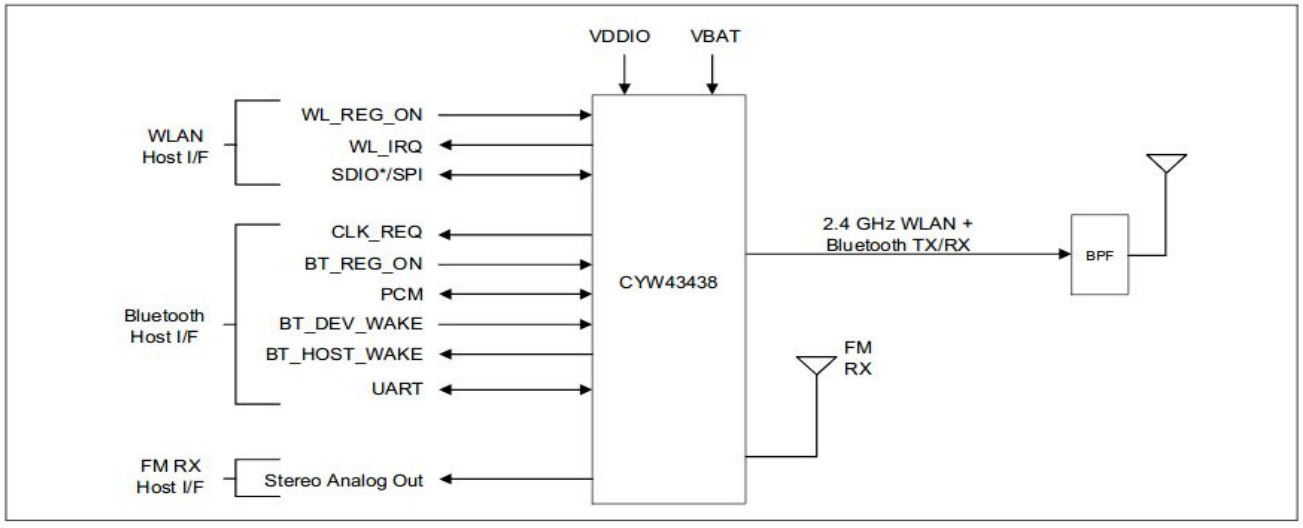


Figure 6 : CYW43438

8 entrées
7 sorties, 2 antennes
communication bluetooth 2.4 GHz WLAN + Bluetooth TX/RX
TX c'est la transmission RX la réception
UART la communication en liaison série
FM RX récepteur audio
CLK_REQ fréquence d'horloge
VBAT la tension de la batterie

- Diagramme des cas d'utilisation

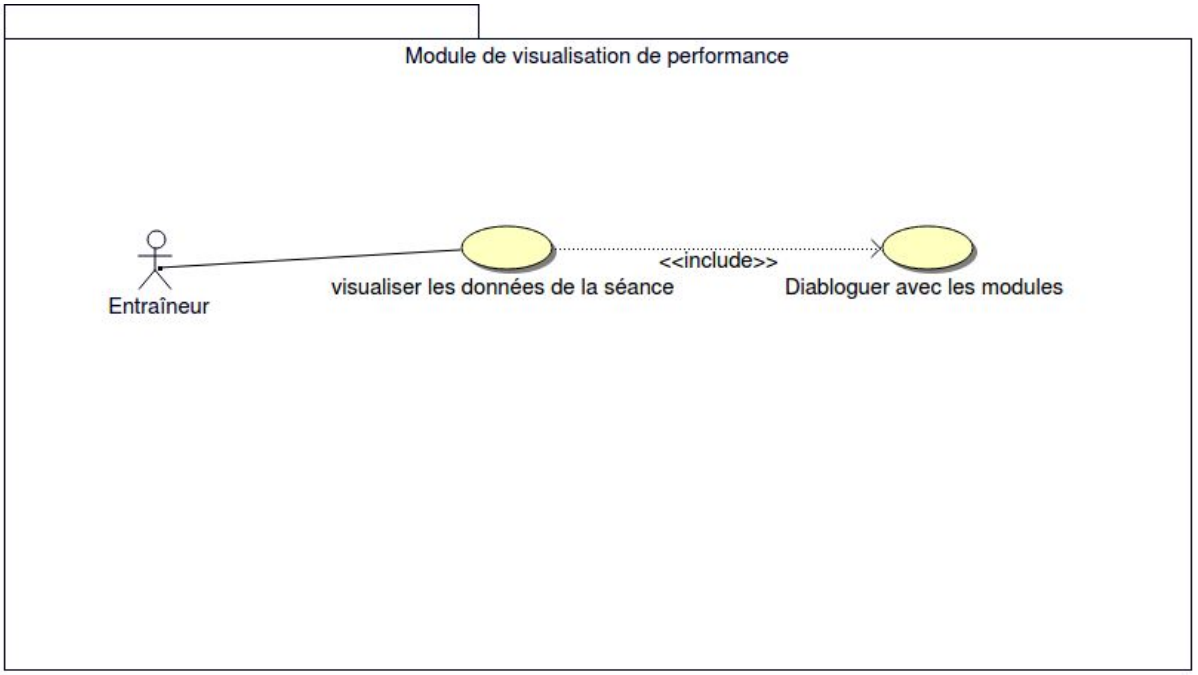


Figure 7 : cas d'utilisation

L'entraîneur peut visualiser en continu les données (pourcentage de balles par zones et le nombre de balles dans la zone à atteindre) d'une séance d'entraînement. Pour cela, le sous-système a besoin de récupérer les trames émises par le terminal mobile.

Les données visualisées sont :

- le nom du joueur (si existant), la durée écoulée de la séance;
- Le pourcentage de balles dans chaque zone en comptant les balles hors table;
- Le nombre et pourcentage de balles hors table (une balle n'ayant pas touché une zone);
- Le nombre de balles jouées par le robot sur le nombre de balles pour la séance;
- L'heure ainsi que le temps de la séance depuis son lancement;
- Le nom du joueur

Une communication entre le sous-système ECRAN-TTPA et le terminal mobile basée sur un protocole spécifique est nécessaire.

- Planification

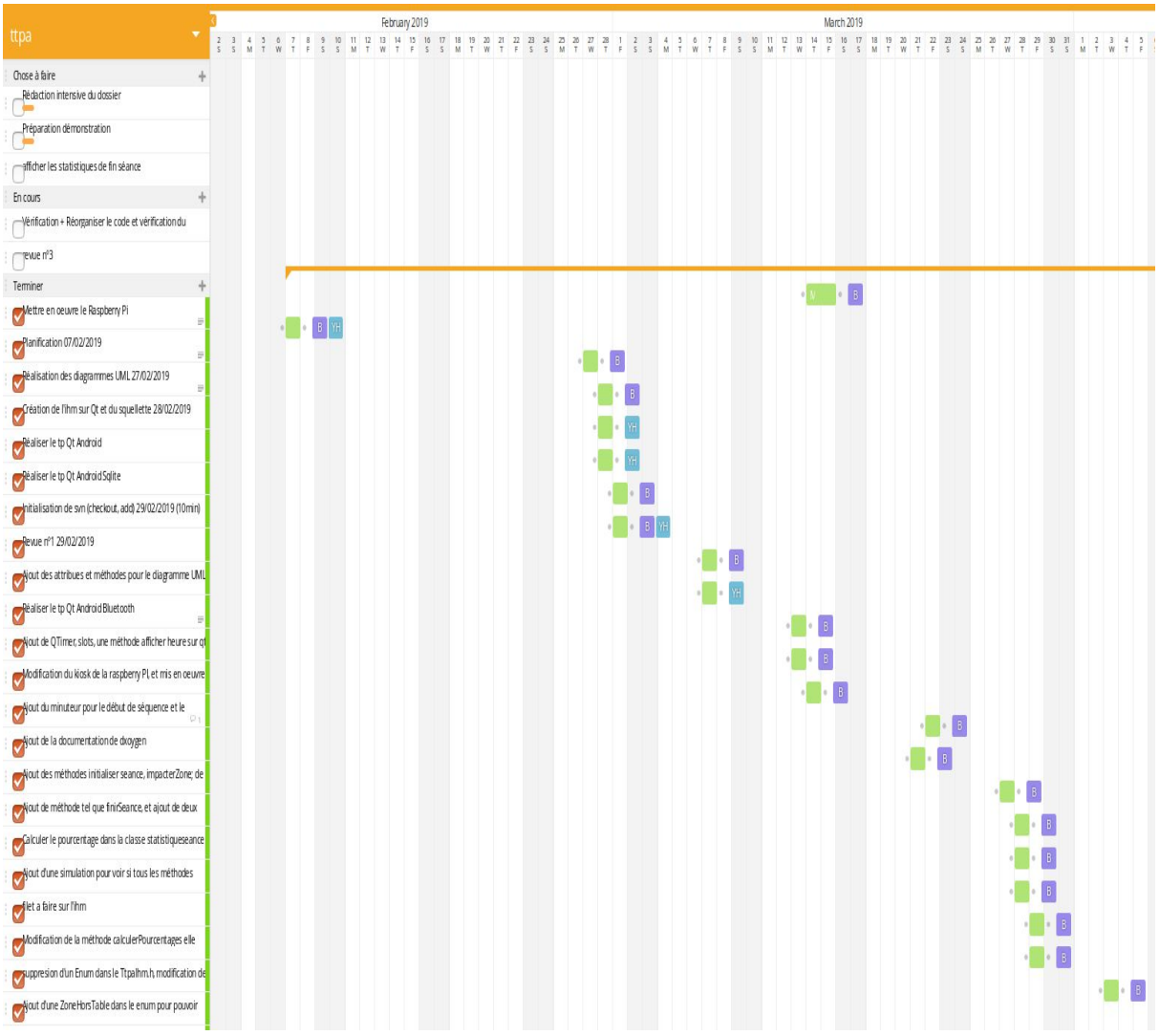


Figure 8: Planification avec trello gantt

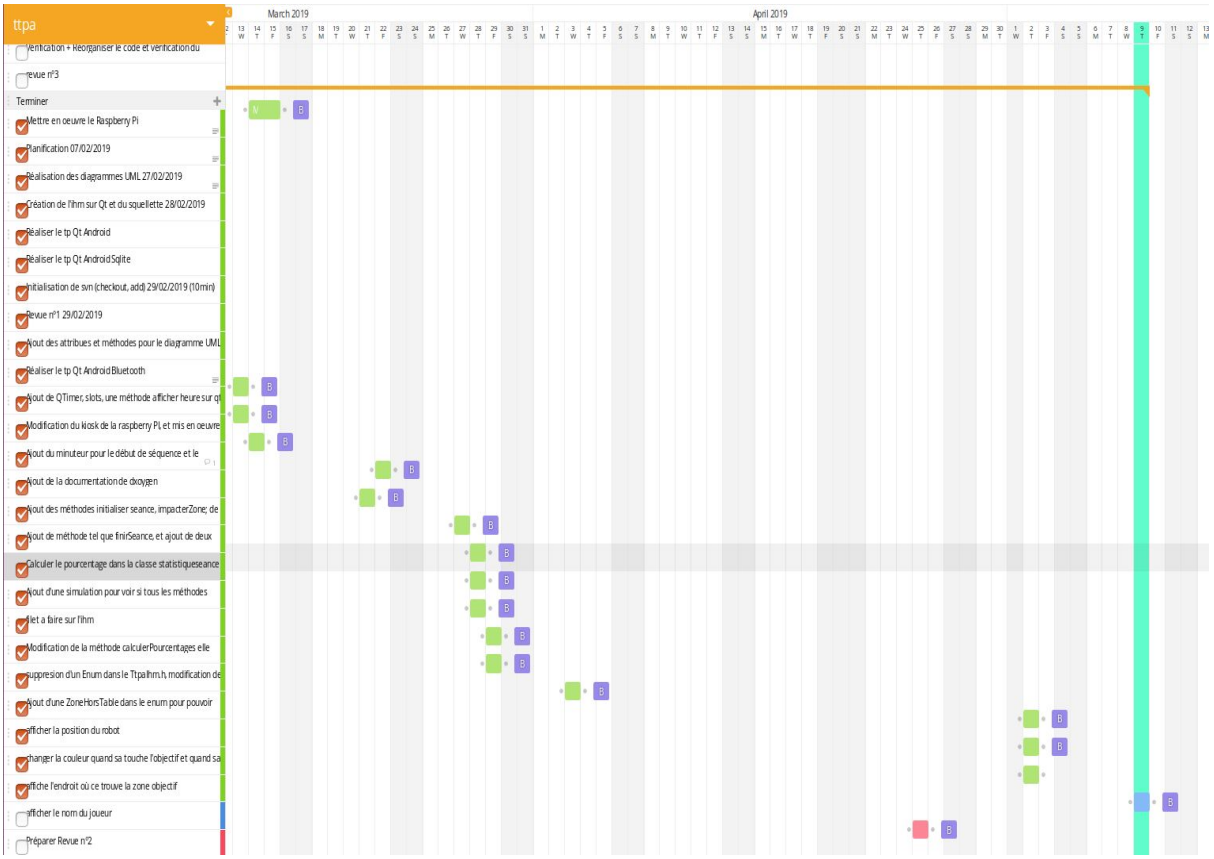


Figure 9: Planification avec trello gantt

- Le développement de l'interface de la table était le plus logique à faire en premier. Il s'agit de l'élément le plus important et le plus visible.
- Le développement a d'abord fonctionné en mode simulation.
- Par la suite les pourcentages par zones ont été ajoutés ainsi que la prise en compte des zones robot et objectif.
- L'ajout du Bluetooth, création du protocole TTPA et l'ajout des fonctions pour la communication entre le terminal mobile est venu après.
- Pour finir, l'ajout de la fenêtre de fin de séance a été réalisé vers la fin du projet.

- Organisation commune au sein du projet

Afin de permettre une meilleur organisation, communication et gestion des fichiers et codes sources, nous avons utilisé :

- **Subversion** qui est un logiciel libre de gestion de versions hébergé sur le site RiouxSVN pour l'ensemble du code source du projet
- un espace de stockage commun (**Google Drive**) pour tous les documents ressources

Enfin, pour la communication matériel entre les différents membres du projet, nous avons mis au point un **protocole de communication**, dont je détaillerai les caractéristiques dans le développement de la partie écran raspberry PI 3B.
Les fichiers sources sont stockés sur un serveur que l'on nomme le référentiel. Il conserve la dernière version de chaque fichier (appelée HEAD), mais également toutes les versions précédentes.

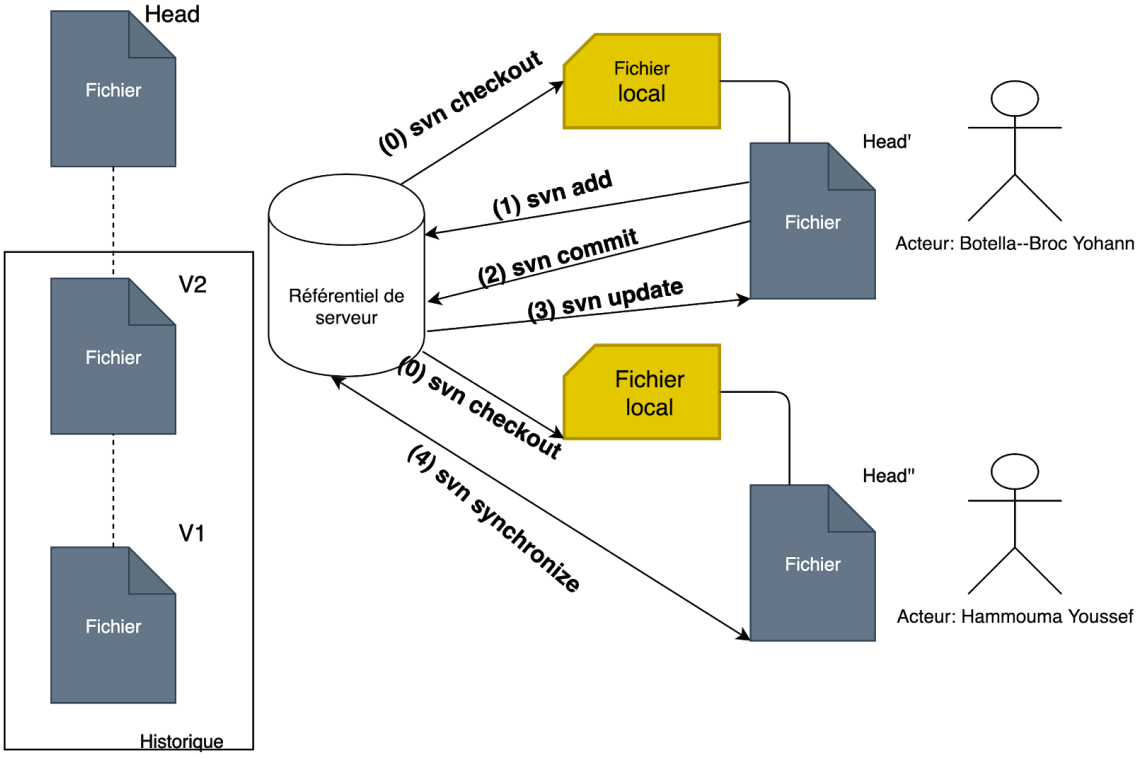


Figure 10: Schémas svn (subversion)

- État d'une séance

Une séance d'entraînement est gérée par le programme en fonction de son état. Le changement d'état est réalisé par la réception de trames envoyées par le terminal mobile.

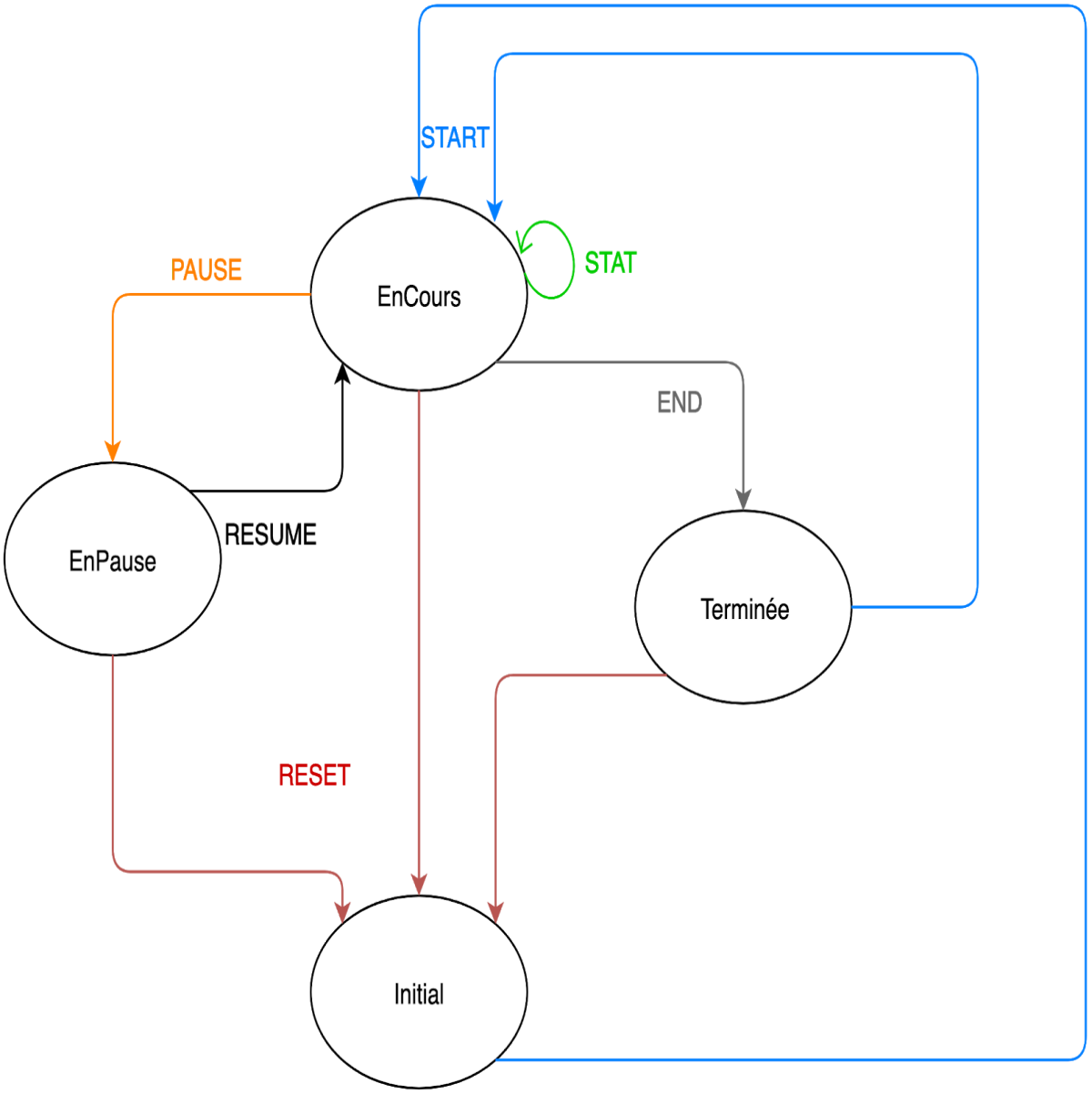


Figure 11: Les différents états d'une séance.

- Les différentes trames TTPA

Toutes les trames sont envoyées par le terminal mobile.

Trame	Description
$ecran-ttpa-2:START	Permet de décoder le nom du joueur, la zone objectif, la zone robot et déclenche le timer de début séance et le basculement vers l'écran séance.
$ecran-ttpa-2:STAT	permet de décoder le numéro de l'impact d'une balle.
$ecran-ttpa-2:PAUSE	permet de mettre la séance en pause.
$ecran-ttpa-2:RESUME	permet de reprendre la séance.
$ecran-ttpa-2:END	permet de basculer vers l'écran de résultats et de finir la séance.
$ecran-ttpa-2:RESET	permet de réinitialiser la séance qui était en cours et renvoie à l'écran d'accueil.

Figure 12: Tableau des différentes trames

La trame **START** déclenche le passage à l'état EnCours qui peut se terminer avec 2 possibilités : la trame **RESET** ou la trame **END**.
En utilisant la trame **PAUSE**, on va passer à l'état EnPause. Il y a seulement deux possibilités pour quitter cet état : la trame **RESET** ou **RESUME**.
La trame **RESUME** de l'état EnPause on passera à l'état EnCours.
Avec la trame **END** on passera de l'état EnCours à l'état Terminée.
Pour finir, la trame **RESET** fera passer à l'état Initial quelque soit l'état de la séance.

Voir l'ensemble du protocole TTPA en Annexe.

- L'interface homme machine (IHM)

Le programme est composé de plusieurs fenêtre crée à l'aide d'un QStackedWidget.

L'IHM utilise des QLabel et QWidget avec des couleurs définies en CSS en fonction de leur état.

L'affichage d'écran d'attente est la première page, il s'agit de l'écran de connexion, elle est affichée au lancement du programme et à la déconnexion du terminal mobile.

L'écran d'attente :

Figure 13: écran d'attente

Dans cet état, les trames attendues sont :

- $ecran-ttpa-2:START;Simon Gauzy;5;6*XX

Cette trame va déclencher le passage à l'état EnCours.

Suivi du déroulement de la séance configurée pour le joueur:

L'écran principal :

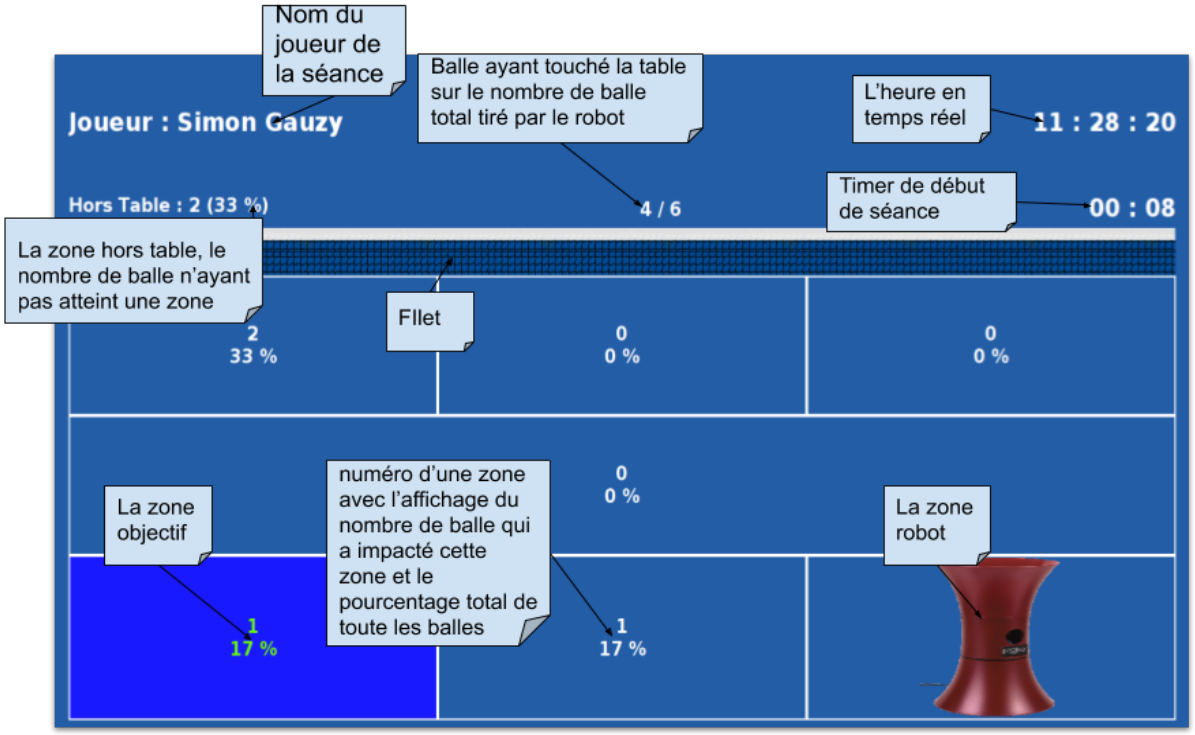


Figure 14: ecran de séance

Dans l'état **EnCours**, on traite les trames :

- $ecran-ttpa-2:STAT;5*XX

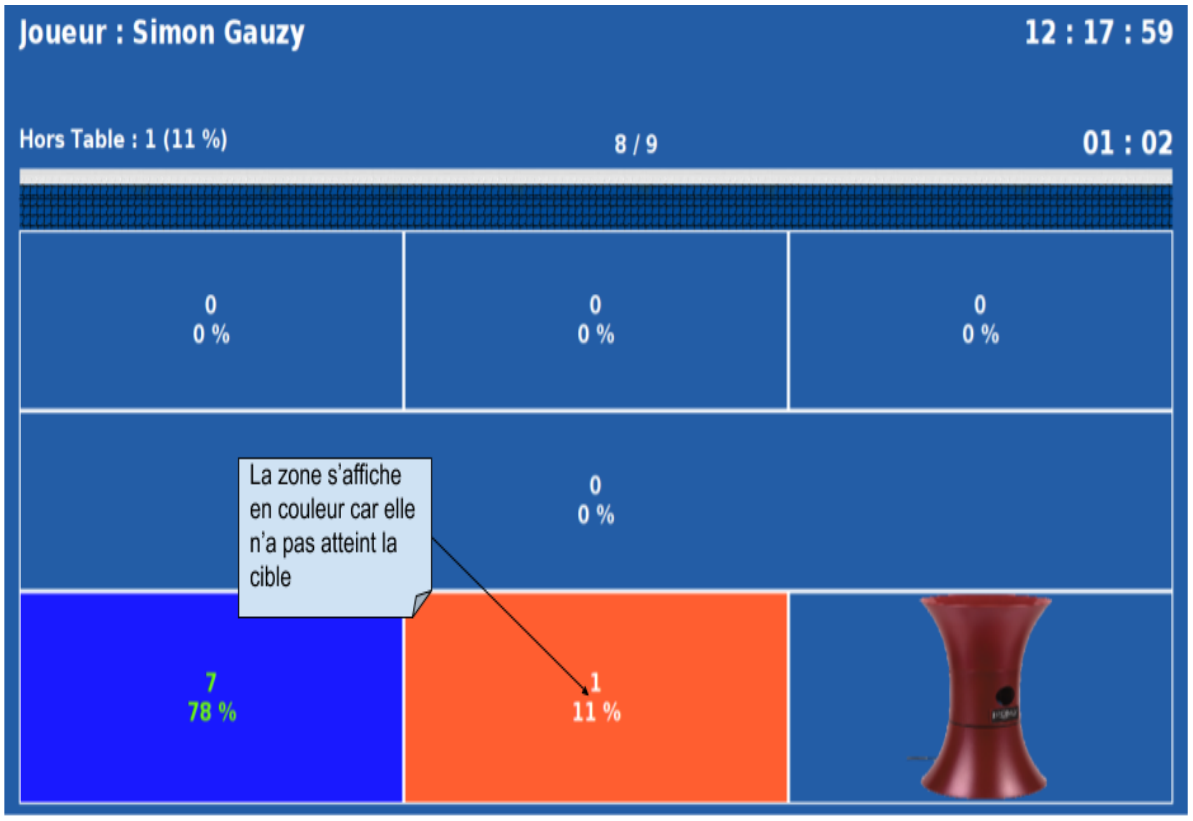


Figure 15: La zone impactée

➢ $ecran-ttpa-2:STAT;4*XX

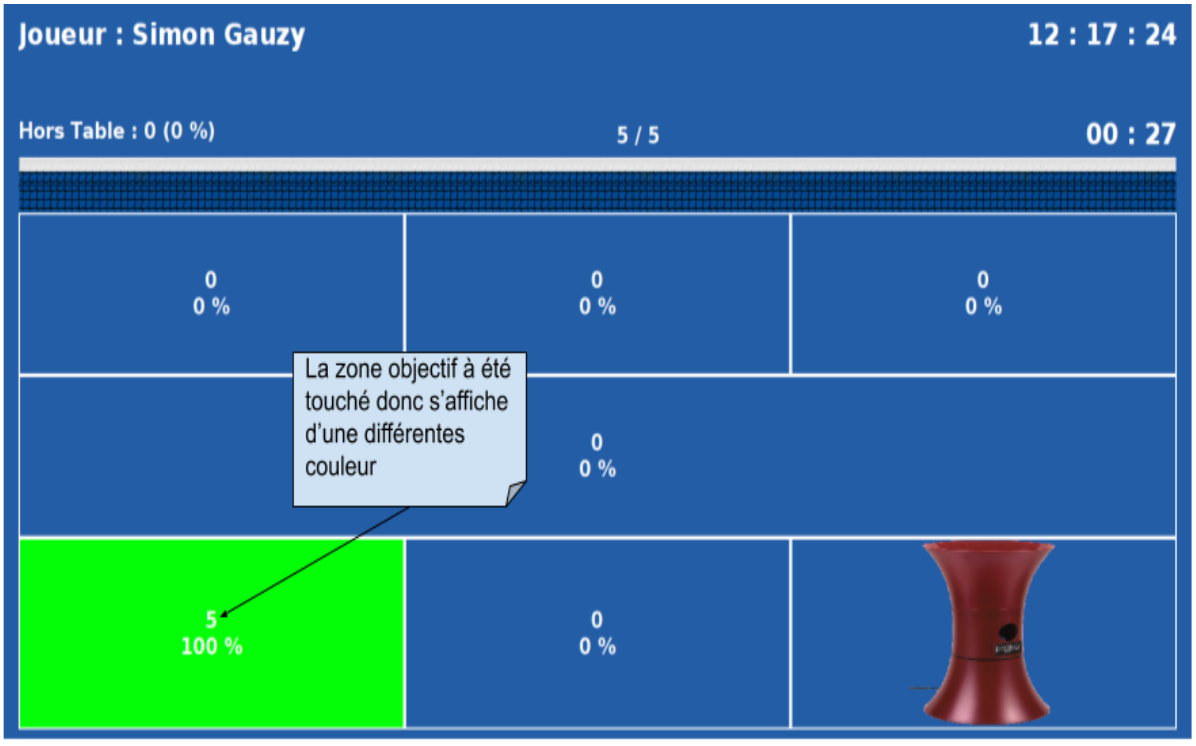


Figure 16: La zone Objectif

➢ $ecran-ttpa-2:PAUSE

En utilisant la trame **PAUSE**, on va passer à l'état EnPause.

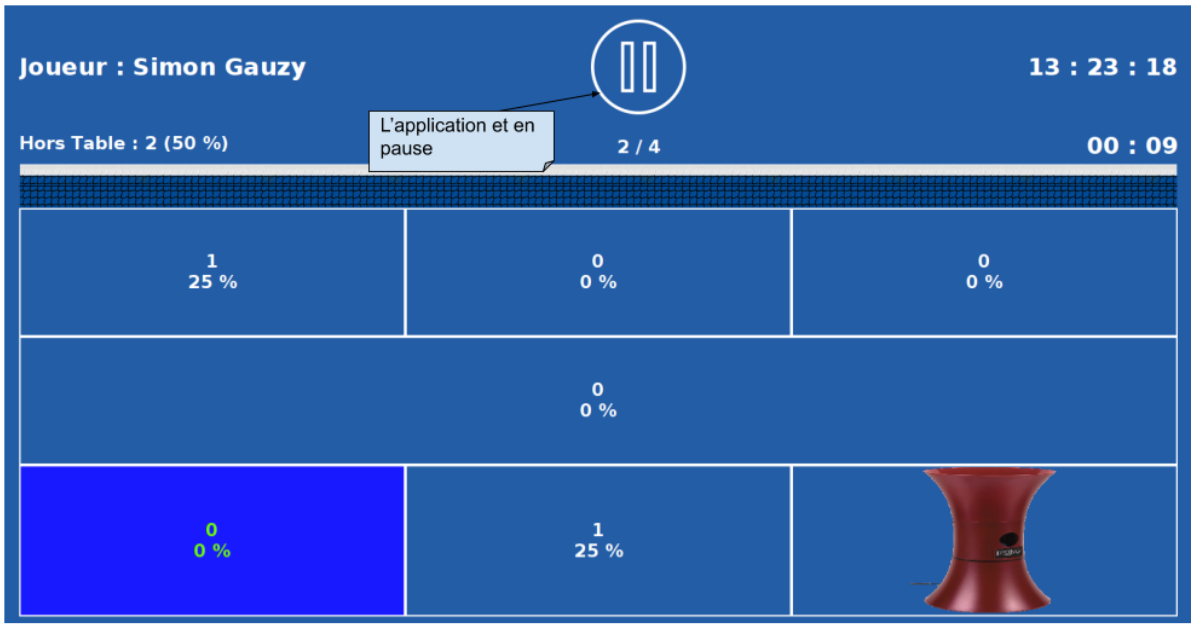


Figure 17 : l'écran en mode pause

➢ $ecran-ttpa-2:RESUME

La trame **RESUME** refait passer à l'état EnCours.

➢ $ecran-ttpa-2:END

Avec la trame **END** on passera de l'état EnCours à l'état Terminée.

➢ $ecran-ttpa-2:RESET

Pour finir, la trame **RESET** de n'importe qu'elle état fera passer à l'état Initial.

L’écran résultats :

L’affichage de l’écran résultats est ensuite affiché, il résume la séance, montrant des statistiques suivantes :

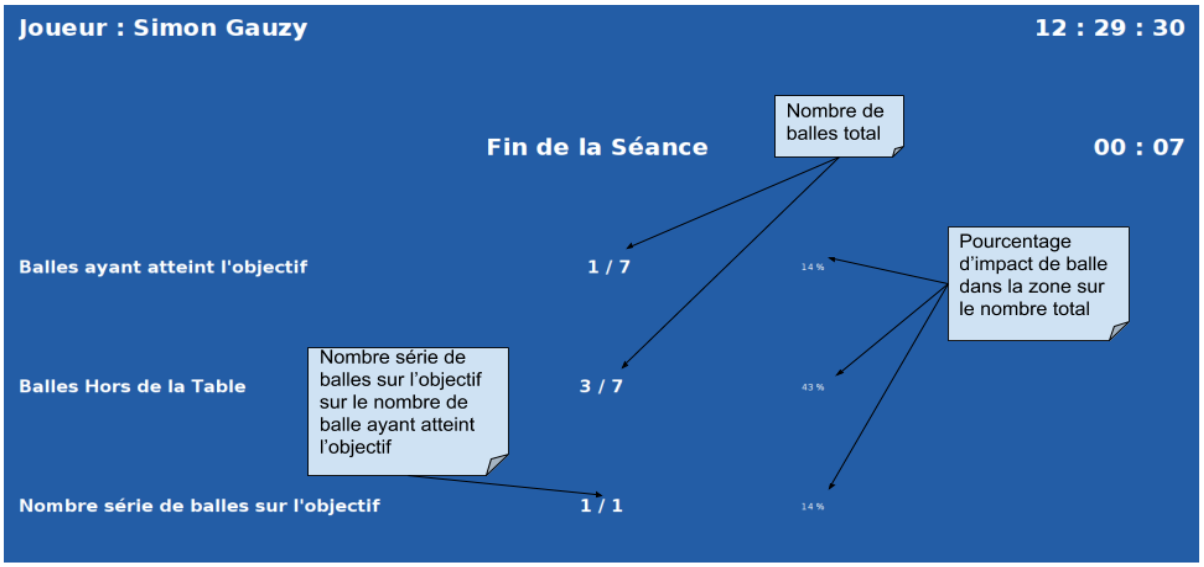


Figure 18: L’écran résultat

Ici il est possible de recevoir une trame **START** pour démarrer une nouvelle séance.

- Diagramme de classes

Figure 19: toute les classe de l'écran TTPA

La classe principale est **Ttpalhm**. Elle a en charge l'affichage des données à l'écran. Les trois écrans sont gérés par un QStackedWidget qui fournit une pile de Widgets. Elle est associée à la classe **StatistiquesSeance** qui gère les statistiques de la séance.
La classe **StatistiquesSeance** assure les calculs liés aux pourcentages de balles dans les zones, balles maximum de la séance et les balles envoyées par le robot. La méthode impacterZone(int numeroZone) déclenche la mise à jour des statistiques au cours d'une séance.
La classe **CommunicationBluetooth** gère la communication entre l'écran et le terminal mobile à l'aide du protocole TTPA.

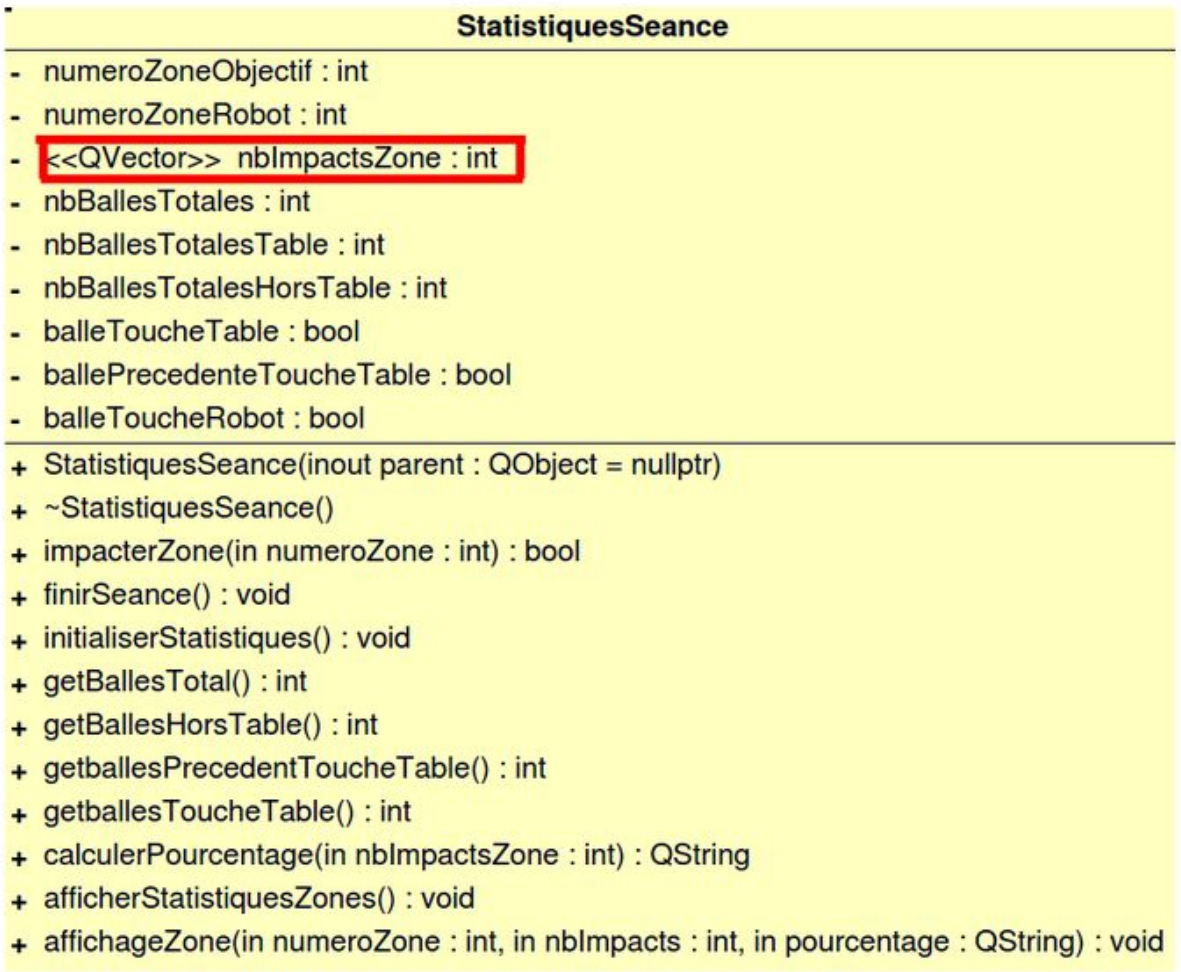


Figure 20: Impact d'une zone à l'aide d'un vecteur

QVector nbImpactZone : int

Le type vector est un tableau dynamique. QVector est un type spécifique à Qt. Ici, il permet de stocker l'ensemble des impacts de balle pour chaque zone.

La classe StatistiqueSeance gère une table composée de 9 zones. Pour faciliter l'utilisation de ces zones, on va les déclarer sous la forme d'un enum.

```
    /** les zones numérotées de 1 à 7 de la table plus une zone non définie et le nombre de zones total*/
    enum Zone
    {
        ZoneNonDefinie = -1,
        Zone1 = 0,
        Zone2,
        Zone3,
        Zone4,
        Zone5,
        Zone6,
        Zone7,
        NbZonesImpact, // 7
        ZoneHorsTable, // 8
        NbZonesTotales, // 9
    };
```

Chaque zone touchée incrémentera sa valeur dans le QVector.

Exemple :

Zone	Valeur	Calcul du pourcentage
Zone1	3	30%
Zone2	2	20%
Zone3	5	50%
Zone4	0	0%
Zone5	0	0%
Zone6	0	0%
Zone7	0	0%
NbZonesImpact	10	--

ZoneHorsTable	0	0%

Figure 21 : impact d'une zone

La classe CommunicationBluetooth assure la réception et le décodage des trames.

```
void CommunicationBluetooth::decoderTrame(QString trame)
{
	QStringList trames;
	trame.remove("\r\n");
	qDebug() << Q_FUNC_INFO << "trame" << trame;

	const QString typeTrame = "ecran-ttpa";
	QString donnees;
	QString nomJoueur;
	QString zoneRobot;
	QString zoneObjectif;
	QString numeroZone;

	if(trame.startsWith("$" + typeTrame))
	{
	/**
	* Démarrage
	* $ecran-ttpa-1:START;JULIEN;2;1*XX
	*                        |   | |
	*                        |   | |
	*                        |   | +- Zone Robot
	*                        |   +--- Zone Objectif
	*                        +------- Nom Joueur
	*
	* $ecran-ttpa-1:STAT;2*XX
	*                  |
	*                  |
	*                  +----------- Numéro Zone Impact
	*/
	donnees = trame.section(':', 1, 1); // START;JULIEN;2;1*XX
	if(trame.contains("START") && (etatSeance == EtatSeance::Initial
|| etatSeance == EtatSeance::Terminee))
	{
		etatSeance = EtatSeance::EnCours;
		// START;JULIEN;2;1*XX
		nomJoueur = donnees.section(';', 1, 1);
		zoneObjectif = donnees.section(';', 2, 2);
		zoneRobot = donnees.section(';', 3, 3).at(0);
		qDebug() << Q_FUNC_INFO << "nomJoueur" << nomJoueur <<
"zoneObjectif" << zoneObjectif << "zoneRobot" << zoneRobot;
```

```
            emit departSeance(nomJoueur, zoneObjectif, zoneRobot);
      }
      else if(trame.contains("STAT") && etatSeance ==
EtatSeance::EnCours)
      {
            // STAT;2*XX
            numeroZone = donnees.section(';', 1, 1).at(0);
            qDebug() << Q_FUNC_INFO << "numeroZone" << numeroZone;
            emit impactZone(numeroZone);
      }
      else if(trame.contains("PAUSE") && etatSeance ==
EtatSeance::EnCours)
      {
            etatSeance = EtatSeance::EnPause;
            //PAUSE*XX
            emit mettreEnPauseSeance();
      }
      else if(trame.contains("RESUME") && etatSeance ==
EtatSeance::EnPause)
      {
            etatSeance = EtatSeance::EnCours;
            //RESUME*XX
            emit repriseSeance();
      }
      else if(trame.contains("END") && etatSeance ==
EtatSeance::EnCours)
      {
            etatSeance = EtatSeance::Terminee;
            //END*XX
            emit finSeance();
      }
      else if(trame.contains("RESET") && (etatSeance ==
EtatSeance::EnCours || etatSeance == EtatSeance::EnPause || etatSeance
== EtatSeance::Terminee))
      {
            etatSeance = EtatSeance::Initial;
            //RESET*XX
            emit resetSeance();
      }
      }
}
```

Ici on découpe les trames reçues à l'aide de la méthodes **section(QString, int)** de la classe **QString** pour récupérer chaque trame individuellement dans le cas de concaténation de plusieurs trames durant la réception. Chaque trame sera traitée

par la suite. L'entête du protocole est vérifié, et les trames sont ensuite gérées à l'aide de signaux en utilisant **emit.**

- Scénario du démarrage d'une séance

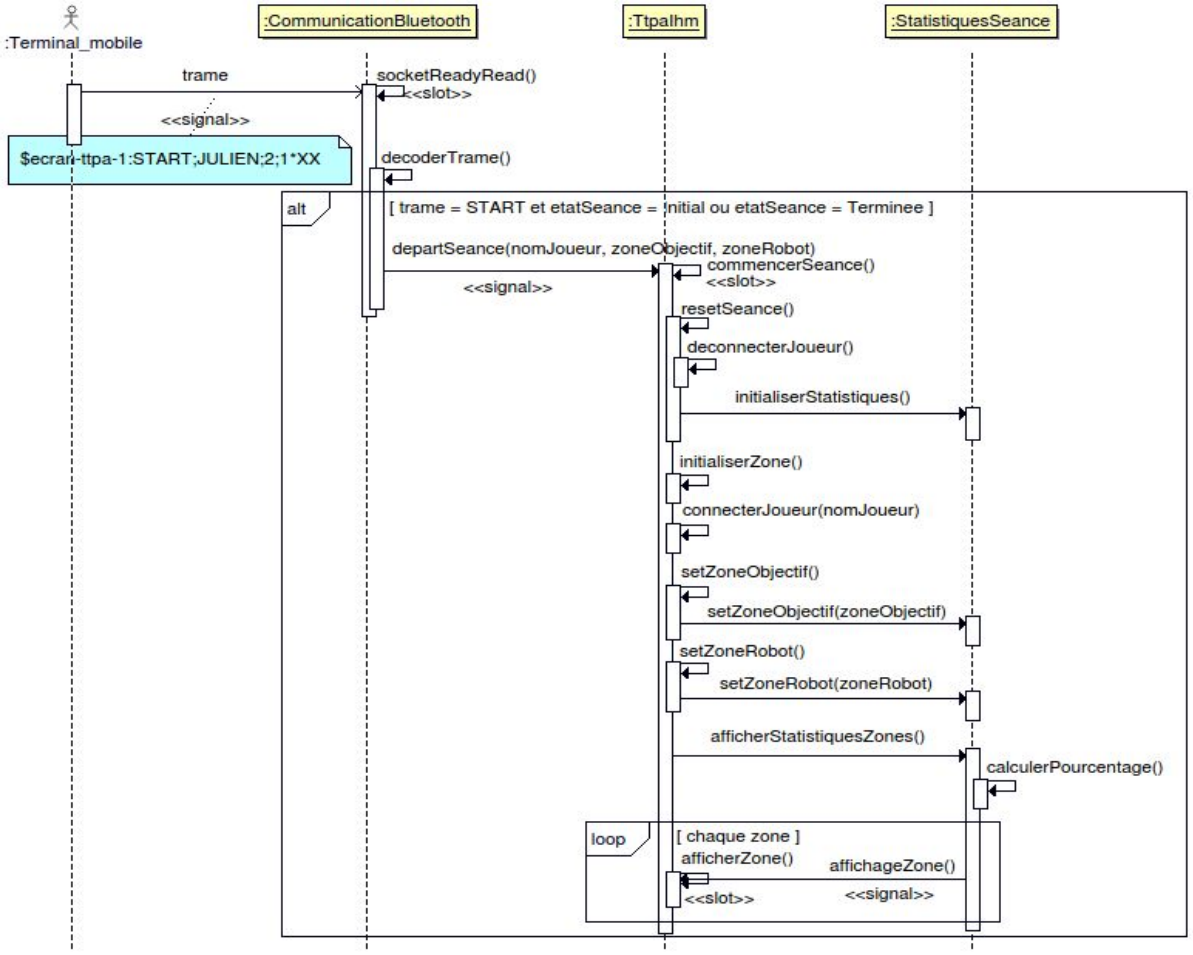


Figure 22: démarrage d'une séance

Le terminal mobile envoie une trame **START**. La méthode decoderTrame() détecte le type de trame **START**. Si l'état de la séance est soit initial ou Termniée, elle extrait les données et émet un signal departSeance avec les paramètres de la nouvelle séance. Ce signal déclenche le slot commencerSeance(). Celle-ci a pour rôle de réinitialiser la séance (affichage et statistiques).

- Scénario d'impact de balle dans une zone

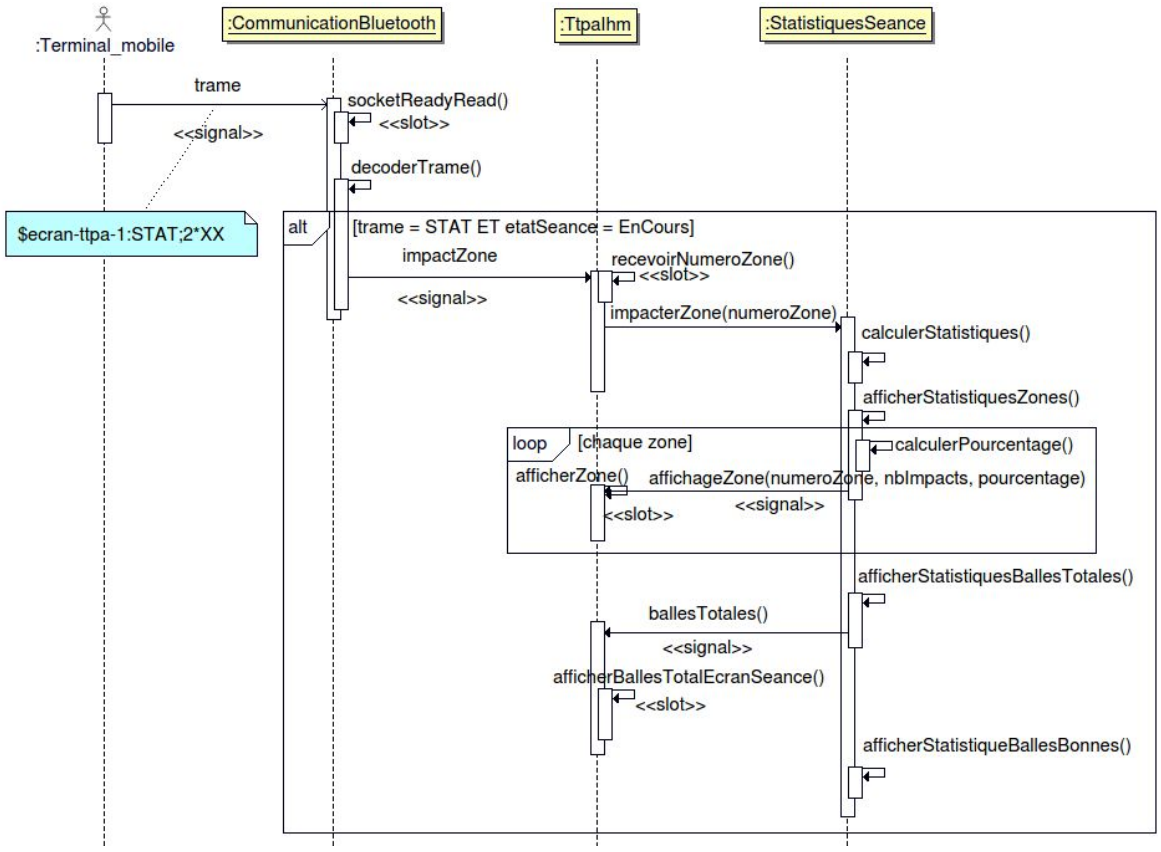


Figure 23: impact de balle dans une zone

On reçoit une trame STAT contenant le numéro de zone impactée. Cela va permettre de (re)calculer et d'afficher les statistiques pour chaque zone.elle extrait les données et émet un signal impactZone avec les paramètre de la séance. Ce signal délenche le slot recevoirNumeroZone(). Celle-ci a pour rôle calculerStatistiques() décrite ci dessous. Provoque l'affichage de ces informations dans l'IHM. L'IHM affiche les données statistiques dans la zone à l'exception de la zone Robot.

```
void StatistiquesSeance::calculerStatistiques(uint8_t numeroZone)//1
{
      if(numeroZone == getZoneRobot())//2
         nbImpactsZone[ZoneHorsTable] += 1;
      else
         nbImpactsZone[numeroZone] += 1;//3
      if(numeroZone == getZoneObjectif())
      {
         nbBallesBonnes++;
      }
      qDebug() << Q_FUNC_INFO << "nbBalles zone = " <<
nbImpactsZone[numeroZone];//4

      ballePrecedenteToucheTable = balleToucheTable;
      balleToucheTable = false;
      ballePrecedenteToucheTable = balleToucheTable;
      balleToucheTable = numeroZone;

      nbBallesTotales++;
      qDebug() << Q_FUNC_INFO << "nbBallesTotales = " <<
nbBallesTotales;

      nbBallesTotalesTable = 0;
      for(int numeroZone = Zone1; numeroZone < NbZonesTotales;
++numeroZone)
      {
         if(numeroZone == NbZonesImpact)
            continue;
         if(numeroZone < NbZonesImpact)
            nbBallesTotalesTable += nbImpactsZone[numeroZone];
      }
      qDebug() << Q_FUNC_INFO << "nbBallesTotalesTable = " <<
nbBallesTotalesTable << "nbBallesBonnes= " << nbBallesBonnes;

      if (numeroZone != ZoneNonDefinie)
      {
            if (numeroZone == numeroZoneObjectif || numeroZoneObjectif
== ZoneNonDefinie)
            {
                  nbBallesEnchainees++;//5
```

```
            }
            else
            {
                  nbBallesEnchainees = 0;
            }
      }
      else
            nbBallesEnchainees = 0;

      if (nbBallesEnchainees > nbBallesEnchaineesMax)
            nbBallesEnchaineesMax = nbBallesEnchainees;//5

}
```

.1)La méthode calclerStatistiques() reçoit le numeroZone en paramètre

.2)Si le numéro de la zone est égal au numéro de la zoneRobot, alors on incrémente le nombre d'impacts pour la zone hors table (une balle dans la zone du robot est considéré comme une balle hors table)

.3)Si le numéro zone est égal à la zone Objectif alors incrémente le nombre de balle ayant atteint la zone objectif

.4)Le nombre de balles dans chaque zone.

.5)Si le numéro zone est égal au numéro objectif ou le numéro zone est égal à la zone non définie incrémente le nombre de balles enchaînées sinon retourne a 0.

- Scénario de fin d'une séance

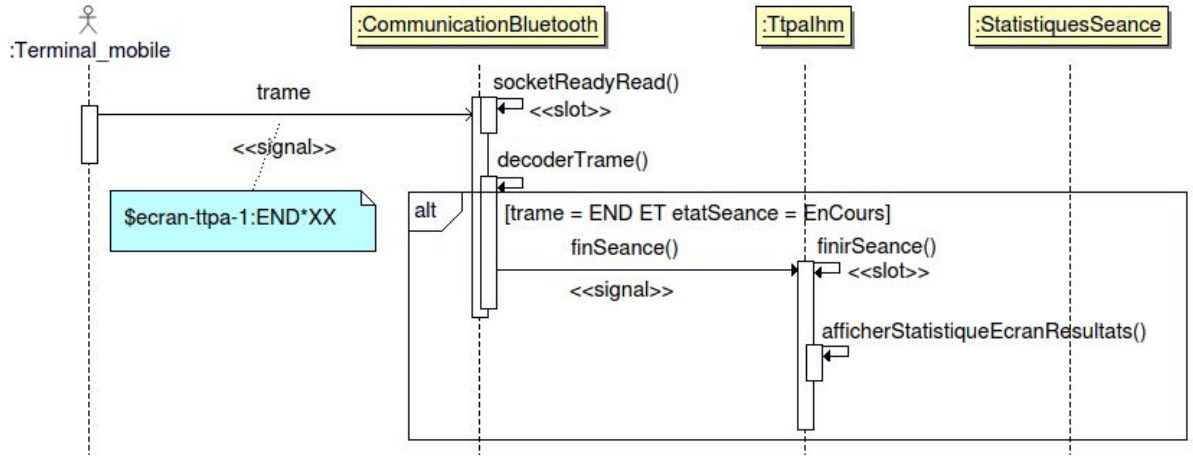


Figure 24 : fin d'une séance

Le terminal mobile envoie avec une trame **END.** La méthode decoderTrame() agis si la trame = END et etatSeance = EnCours, un signal finSeance est envoyé à la classe TtpaIhm. Le slot finirSeance() sera exécuté pour afficher toutes les statistiques de l'écran résultats.

- Tests de validation

Description	OUI	NON
Le système d'exploitation est installé et fonctionnel		
L'écran est configuré en mode "kiosque"		
La zone d'impact est identifiée et affichée en temps réel		
Les données de la séance (le pourcentage de balles par zones et le nombre de pourcentage ayant été sur la bonne zone) sont affichées en temps réel		
Les liaisons sans fil sont opérationnelles (Dialoguer avec le terminal mobile)		
Les statistiques sont affichées en fin de séquence		

Partie Hammouma Youssef (Étudiant IR)

Table des matières

Objectifs

Le travail de l'étudiant est de développer une application mobile Android permettant le lancement d'une séance de tennis de table ainsi que son paramétrage. L'application doit permettre à l'utilisateur d'utiliser des profils et de visualiser les statistiques de jeu en fonction du profil choisi.

Le terminal mobile communiquera avec les différents périphériques via une liaison sans fil *Bluetooth*.

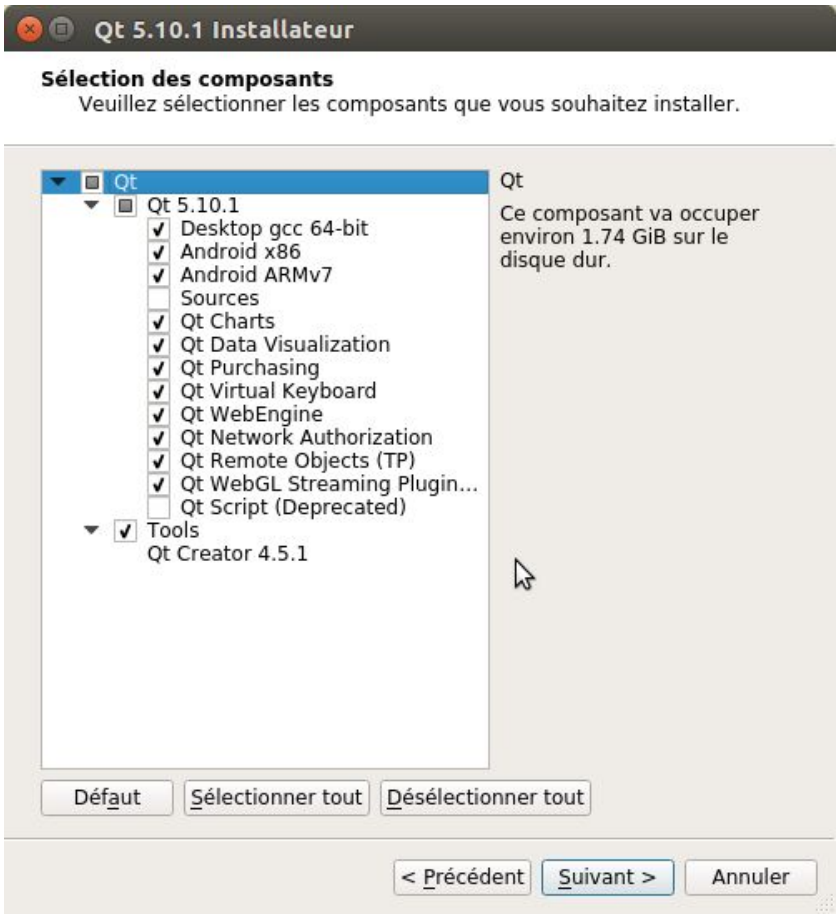
Pré-requis

Qt 5

Pour développer l'application mobile, il faut un environnement de développement : Qt Creator. La version installée est la 5.10 car elle possède tous les modules nécessaires au développement de notre application notamment QBluetooth.

Installation :

```
$ wget
https://download.qt.io/archive/qt/5.10/5.10.1/qt-opensource-linux-x64-5.
10.1.run
$ chmod +x qt-opensource-linux-x64-5.10.1.run

// Mono-utilisateur dans $HOME/Qt5.10.1/
$ ./qt-opensource-linux-x64-5.10.1.run

Ou

// Multi-utilisateur dans /opt/Qt5.10.1/
$ gksudo ./qt-opensource-linux-x64-5.10.1.run
```



Une fois les commandes exécutées l'assistant d'installation se lance, il ne reste plus qu'à installer Qt.

L'installation est validée, Qt est sous la bonne version.

Java SE Development Kit

JRE (Java Runtime Environement) est le kit destiné au client pour pouvoir exécuter un programme Java. Il se compose essentiellement d'une machine virtuelle Java (JVM) capable d'exécuter le bytecode et les bibliothèques standard de Java.

Le développement Android nécessite le SDK Java, le JRE seul n'est pas suffisant. **On a besoin de la version 1.8 de Java SDK**.

Installation :

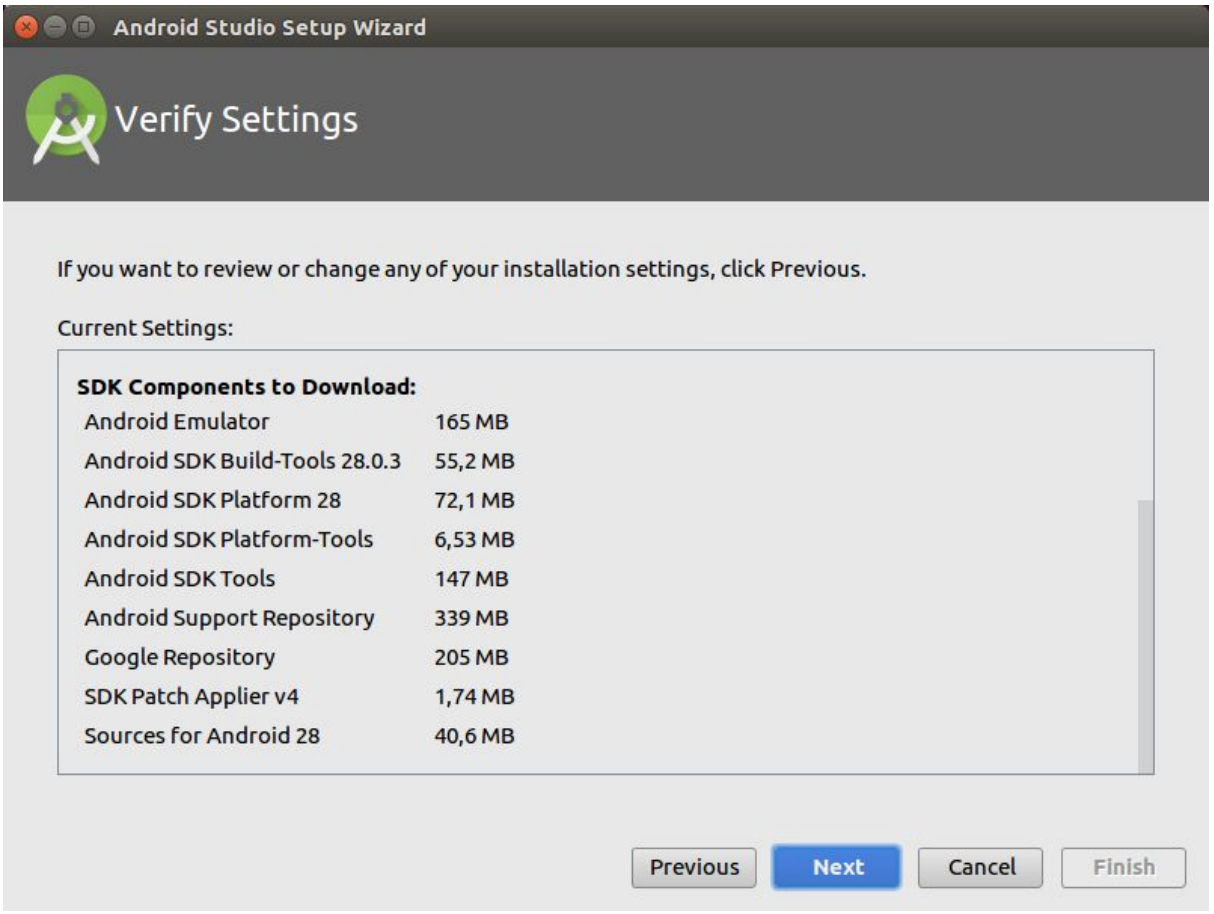
```
$ sudo cp jdk-8u102-linux-x64.tar.gz /usr/local

$ cd /usr/local/

$ sudo tar zxvf jdk-8u102-linux-x64.tar.gz

$ sudo rm jdk-8u102-linux-x64.tar.gz

$ vim $HOME/.bashrc

export PATH=/usr/local/jdk1.8.0_102/bin:$PATH

$ source $HOME/.bashrc
```

SDK Android (Android Studio)

Le SDK d'Android est fourni avec Android Studio.
Installation :

```
$ sudo mv android-studio-ide-171.4443003-linux.zip /usr/local/

$ cd /usr/local/

$ sudo unzip android-studio-ide-171.4443003-linux.zip

$ sudo rm android-studio-ide-171.4443003-linux.zip
$ vim $HOME/.bashrc

export
PATH=$PATH:/usr/local/android-studio/bin:$HOME/Android/Sdk/platform-tool
s:$HOME/Android/Sdk/tools:$HOME/Android/Sdk/tools/bin
```

```
$ source $HOME/.bashrc

$ studio.sh
```



Une fois l'installation terminée le SDK sera installé, l'Android SDK est un ensemble complet d'outils de développement incluant un débogueur, des bibliothèques logicielles, un émulateur basé sur QEMU, de la documentation, des exemples de code et des tutoriaux.

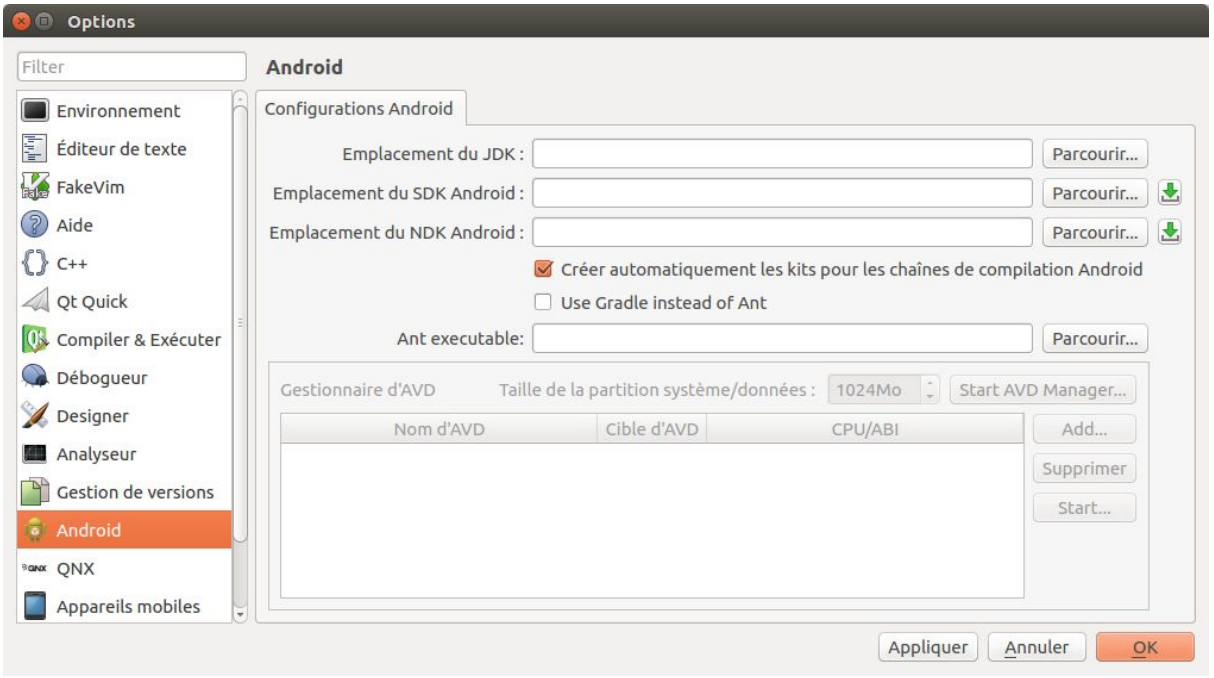
Android NDK

Android NDK (Android Native Development Kit) est une API du système d'exploitation Android permettant de développer directement dans le langage C/C++ du matériel cible, par opposition au Android SDK qui est une abstraction en bytecode Java, indépendante du matériel.

Installation :

```
$ cd $HOME/Android/Sdk
$ wget
https://dl.google.com/android/repository/android-ndk-r17c-linux-x86_64.z
ip
$ unzip android-ndk-r17c-linux-x86_64.zip
```

Une fois tous les modules installés il ne reste plus qu'à les configurés dans les réglages de Qt.



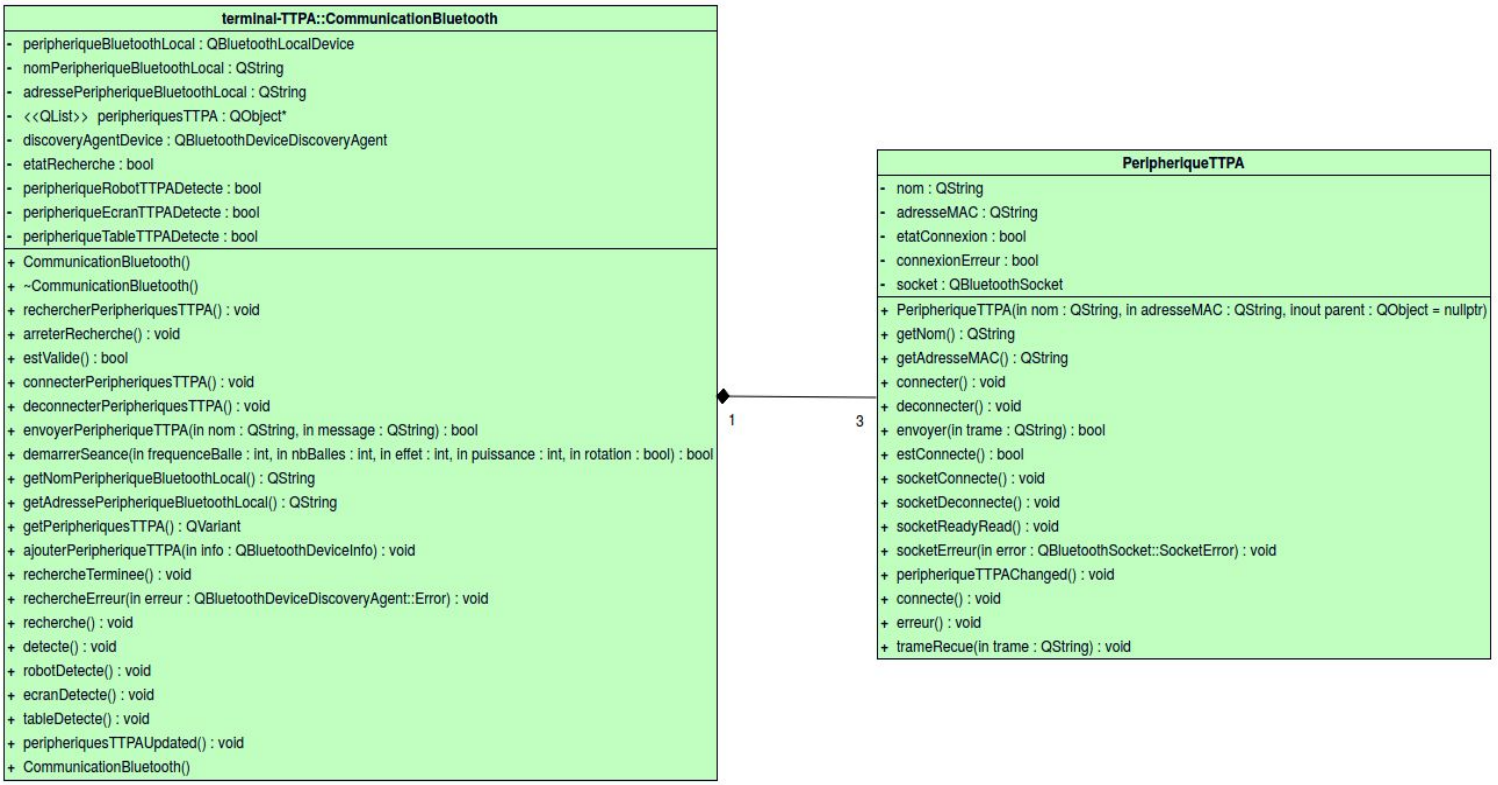
Ressources logicielles

Fonctionnalité	Elément utilisé	Version
Système d'exploitation du poste de développement	GNU/LINUX	4.8.0
Logiciel de planification	Trello + Trello Gantt	
Génération des diagrammes	BOUML	7.8
Gestion de versions	Subversion	1.9.3
Environnement de développement	Qt/Qml	5.10.1
Génération de la documentation	Doxygen	1.8.11
Tablette test	Tablette Samsung	Android 7.0

Caractéristiques et spécifications (Tablette)

Modèle	Processeur Graphique	Mémoire vive (RAM)	Batterie	Système d'exploitation	Bluetooth
Galaxy Tab S2 (SM-T813)	Qualcomm Adreno 510, 550 MHz	3 Go, 933 MHz	5870 mAh	Android 7.0	4.1

Diagramme de classes



La classe CommunicationBluetooth assure la détection et la connexion des périphériques TTPA, elle permet l'envoie des informations nécessaires au déroulement d'une séance (voir Protocole Trame en *Annexe*) tels que :

- Une trame indiquant le démarrage d'une séance à tous les périphériques TTPA
- La transmissions des données de la page "Paramètres" au robot
- La Zone du robot ainsi que la zone "objectif" choisie par le joueur
- Le nom du joueur
- Lorsque la table détecte un impact de balle, la classe le transmet à l'écran TTPA

La classe PeripheriqueTTPA permet la communication avec les appareils Bluetooth TTPA elle permet de créer les 3 sockets Clients qui communiqueront avec les sockets Serveur des différents appareils TTPA.

Sockets

Effectivement, la classe périphérique créé 3 sockets (une pour chaque périphérique), chaque périphérique, possède une socket et la tablette possède aussi une socket. Il y a donc 7 sockets qui communiquent lors du déroulement d'une séance.

Un socket est un point de terminaison d'une communication *bidirectionnelle*, c'est-à-dire entre un client et un serveur en cours d'exécution.

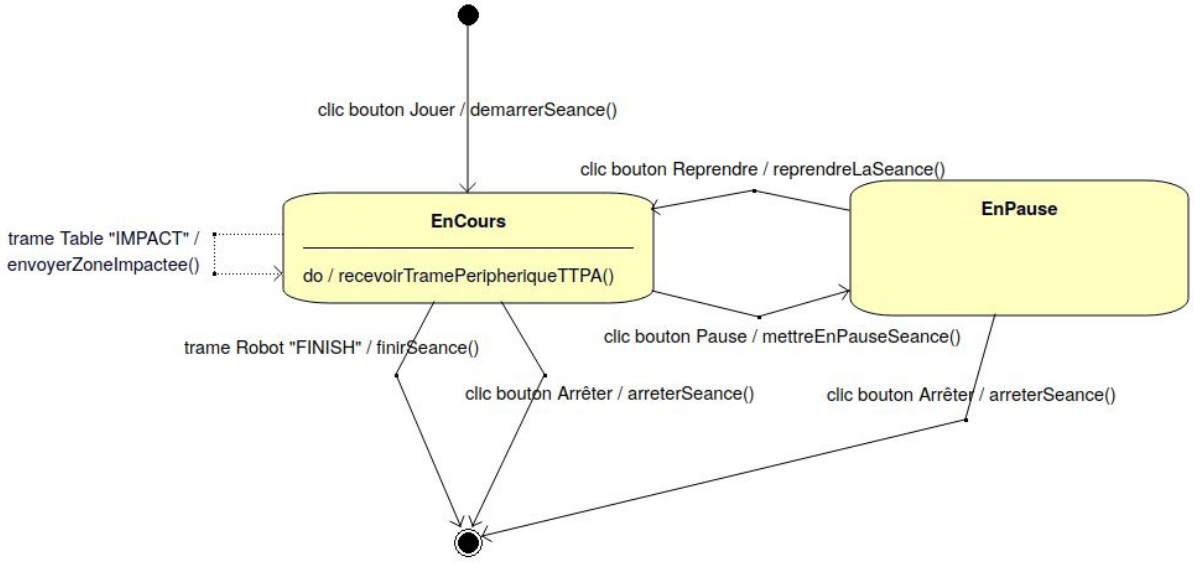
Qt permet d'utiliser des sockets à l'aide du module bluetooth, il suffit de l'indiquer dans le fichier ***.pro*** du projet :

```
QT += qml quick quickcontrols2 bluetooth sql
```

La méthode connecter() permet d'instancier une socket si il n'y en pas déjà une et connecte les différents signaux aux slots, puis génère un UUID qui permet d'avoir une unicité, et connecte la socket à l'adresse MAC ainsi que l'UUID.

```
void PeripheriqueTTPA::connecter()
{
      if (!socket)
      {
      socket = new
QBluetoothSocket(QBluetoothServiceInfo::RfcommProtocol);
      connect(socket, SIGNAL(connected()), this,
SLOT(socketConnecte()));
      connect(socket, SIGNAL(disconnected()), this,
SLOT(socketDeconnecte()));
      connect(socket, SIGNAL(readyRead()), this,
SLOT(socketReadyRead()));
      connect(socket, SIGNAL(error(QBluetoothSocket::SocketError)),
this, SLOT(socketErreur(QBluetoothSocket::SocketError)));
      }
      else if (socket->isOpen())
      {
      socket->close();
      }
      QBluetoothUuid uuid = QBluetoothUuid(QBluetoothUuid::SerialPort);
      socket->connectToService(QBluetoothAddress(adresseMAC), uuid);
      socket->open(QIODevice::ReadWrite);
}
```

Diagramme états-transitions



Une séance possède 4 états :

- Initiale (Tous les appareils sont prêts)
- En cours (La séance est lancée, les différents périphériques communiquent à l'aide des trames)
- En pause (Le déroulement de la séance est mis en pause)
- Arrêtée

Chaque état correspond à une trame qui indique aux différents modules l'état dans lequel se trouve la séance (voir Annexe).

Dans le code source, les différents états ont été définis à l'aide d'un Enum :

```
enum EtatDeLaSeance
      {
      Initial = 0,
      EnCours = 1,
      EnPause = 2,
      Arretee
```

Au lancement de l'application l'état de la séance est "**Initial**" puis change en fonction des manipulations de l'utilisateur sur l'application.

Déploiement de l'APK

L'application TTPA possédant une base de donnée interne, le déploiement de celle-ci s'effectue de manière spécifique.

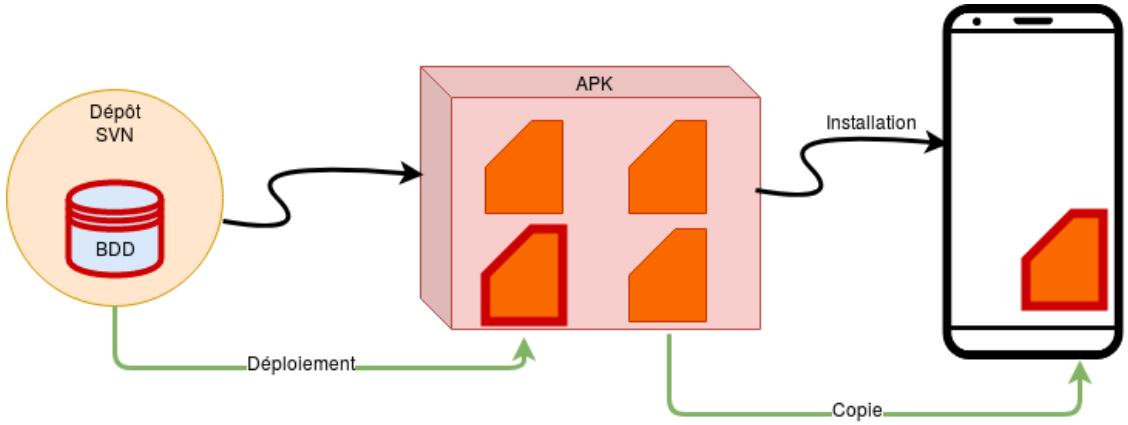
La base de donnée se situe dans le dépôt SVN avec le code source, lors de la compilation l'APK se crée, il faut déployer la base de donnée dans l'arborescence de l'APK. Ceci s'effectue dans le ***.pro***, pour le projet nous avons déployer la base de donnée dans le dossier ***/assets/db***.

```
unix:!macx:
{
      android:
      {
      # déploie la base de données avec l'apk
      deployment.files += ttpa.sqlite
      deployment.path = /assets/db
      INSTALLS += deployment
      }
      !android:
      {
      # copie la base de données dans le dossier build
      CONFIG += file_copies
      COPIES += bd
      bd.files = ttpa.sqlite
      bd.path = $$OUT_PWD/
      bd.base = $$PWD/
      }
}
```

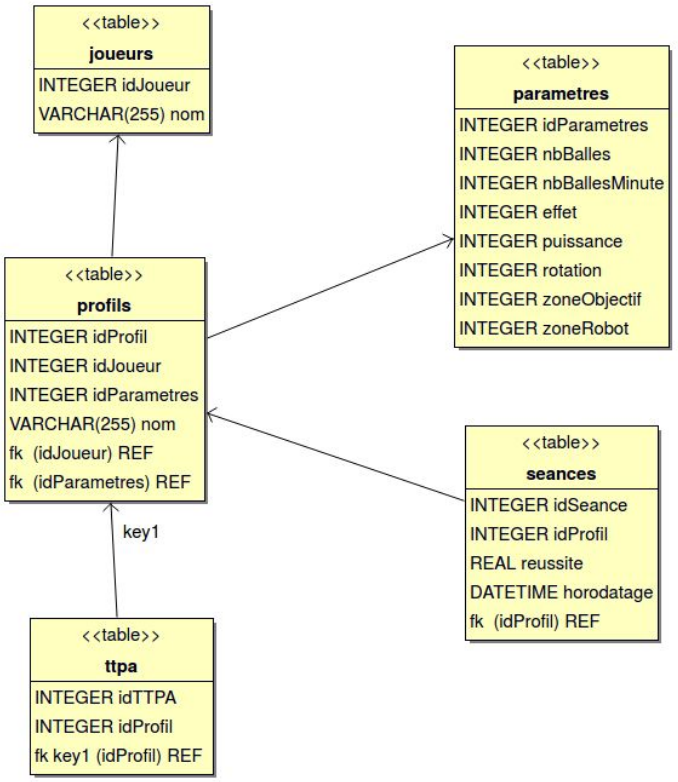
Une fois la base de donnée déployée dans l'APK il faut maintenant la copier dans le terminal mobile lors de l'installation de l'application. Dans le projet la base de donnée est installée à la racine de l'application à l'aide de la méthode ouvrir() de la classe BaseDeDonnees.

```
if(!getNomBDD().isOpen())
      {
      QFile sfile(QString("assets:/db") + QString("/" + fichierBase));
      QFile dfile(QString("./" + fichierBase));

      // supprime le fichier destination
      /*if (sfile.exists())
      {
            if (dfile.exists())
            {
            dfile.remove();
            }
      }*/

      // copie
      if (sfile.exists())
      {
            sfile.copy(dfile.fileName());
            bool retour =
dfile.setPermissions(QFileDevice::ReadUser|QFileDevice::WriteUser);
            qDebug() << Q_FUNC_INFO << retour << dfile.permissions();
      }

      getNomBDD().setDatabaseName(QString("./") + fichierBase);
```

Voici un schéma expliquant le déroulement du déploiement de la base de donnée du dépôt Subversion au terminal mobile.



Base de données

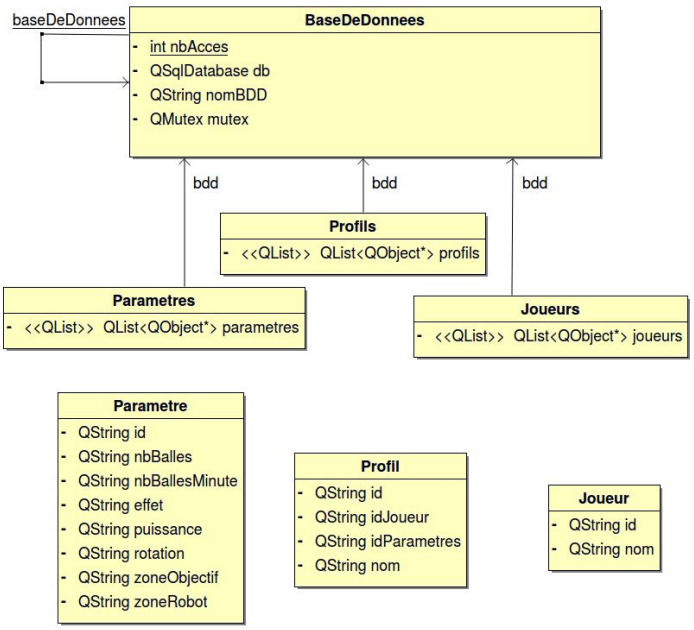


La base de donnée est composée de 5 tables :

- ❏ **joueurs** : Contient le nom du joueur qui à lancé une séance
- ❏ **parametres** : Contient les différents paramètres réglés lors du lancement de la séance
- ❏ **profils** : Contient deux clés étrangères (**idJoueur**, **idParametres**) permettant d'associer un joueur et ses paramètres au profil.
- ❏ **seances** : contenant la clé étrangère (**idProfil**) ainsi que la réussite de la séance et sa date.

La base de donnée est réalisée à l'aide de SQLite qui est un système de gestion de base de données, l'un des points forts de SQLite est le fait qu'il soit comme vu plus tôt intégré à l'APK ce qui signifie qu'il est trés rapide, étant donné qu'il s'exécute sur la même machine, pas besoin de passer par un réseau lorsqu'il est question d'exécuter des requêtes ou de lire des résultats.

Diagramme de classes (Base de données)



Les classes ***Profils***, ***Parametres*** et ***Joueurs*** sont associées à la classe *BaseDeDonnees* à l'aide de l'objet bdd situé dans chacune des différentes classes, c'est elles qui permettent d'effectuer les différentes requêtes SQL propres à leurs classes.

QML

Qt Markup Langage est un langage déclaratif d'interface utilisateur. Il est compatible avec le framework Qt et s'interface avec son API et peut communiquer avec notre code C++.

Association d'un objet C++ au document principal QML

Pour associer un objet C++ à un fichier QML il faut tout d'abord inclure dans le main.cpp la classe de l'objet désiré pour accéder à la déclaration de celle-ci.

```
#include <QGuiApplication>
#include <QQmlApplicationEngine>
#include <QQuickStyle>
#include <QIcon>
#include <QQmlContext>
#include <QtSql/QtSql>
#include "CommunicationBluetooth.h"
```

Par exemple ci-dessus la classe "CommunicationBluetooth".

Il faut ensuite instancier un objet et ajouter l'objet C++ au contexte QML, une fois l'objet instancié il faut charger le document principal QML dans l'application.

```
//Instaciation de l'objet CommunicationBluetooth
CommunicationBluetooth *communicationBluetooth = new
CommunicationBluetooth;
//Ajout de l'objet au contexte QML
engine.rootContext()->setContextProperty("communicationBluetooth",
communicationBluetooth);
//Charge le document principal QML dans l'application
engine.load(QUrl(QStringLiteral("qrc:/FenetreTTPA.qml")));
```

Appel d'une méthode C++ à partir de QML

Pour appeler une méthode C++ depuis QML il faut dans un premier temps que celle-ci soit déclarée avec **Q_INVOKABLE** dans une classe qui hérite de **QObject.**

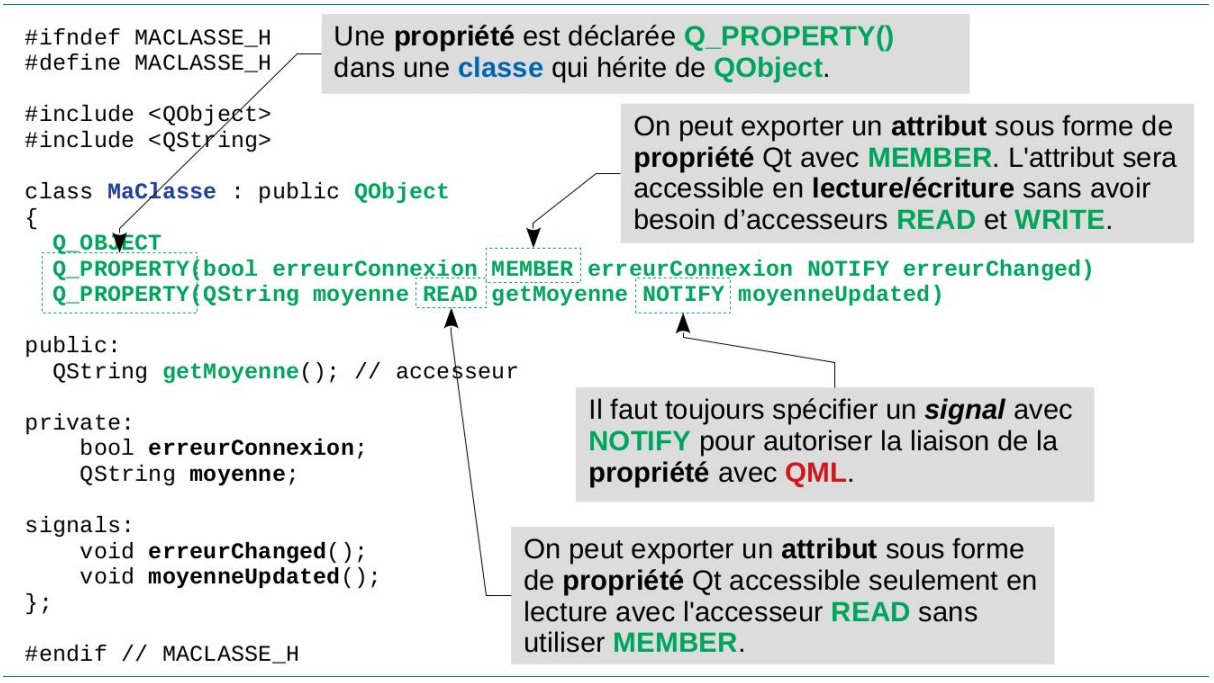
```
class CommunicationBluetooth : public QObject
{
Q_INVOKABLE bool arreterSeance();
...
...
};
```

CommunicationBluetooth.h

Il est maintenant possible d'appeler la méthode de l'objet communicationBluetooth à partir de QML/JS (Il est aussi possible d'appeler un slot).

```
onClicked: {
            boutonJouer.text = "Jouer";
            boutonArreter.enabled = false;
            communicationBluetooth.arreterSeance();
            nomJoueur.enabled = true;
            messageErreurRobot.visible = false;
      }
```

Déclarer une propriété C++ pour QML



Une des utilisations de Q_PROPERTY dans l'application TTPA est notamment la détection des différents périphériques qui permettent à l'application d'afficher une LED verte lorsqu'un périphérique est détecté.

```
class CommunicationBluetooth : public QObject
{
      Q_OBJECT
      Q_PROPERTY(bool peripheriqueEcranTTPADetecte MEMBER
peripheriqueEcranTTPADetecte NOTIFY detecte)
      Q_PROPERTY(bool peripheriqueTableTTPADetecte MEMBER
peripheriqueTableTTPADetecte NOTIFY detecte)
      Q_PROPERTY(bool peripheriqueRobotTTPADetecte MEMBER
peripheriqueRobotTTPADetecte NOTIFY detecte)

...
...
};
```

CommunicationBluetooth.h

```
Item {
      property bool recherche: communicationBluetooth.etatRecherche
      property bool robotDetecte:
communicationBluetooth.peripheriqueRobotTTPADetecte
      property bool ecranDetecte:
communicationBluetooth.peripheriqueEcranTTPADetecte
      property bool tableDetecte:
communicationBluetooth.peripheriqueTableTTPADetecte
```

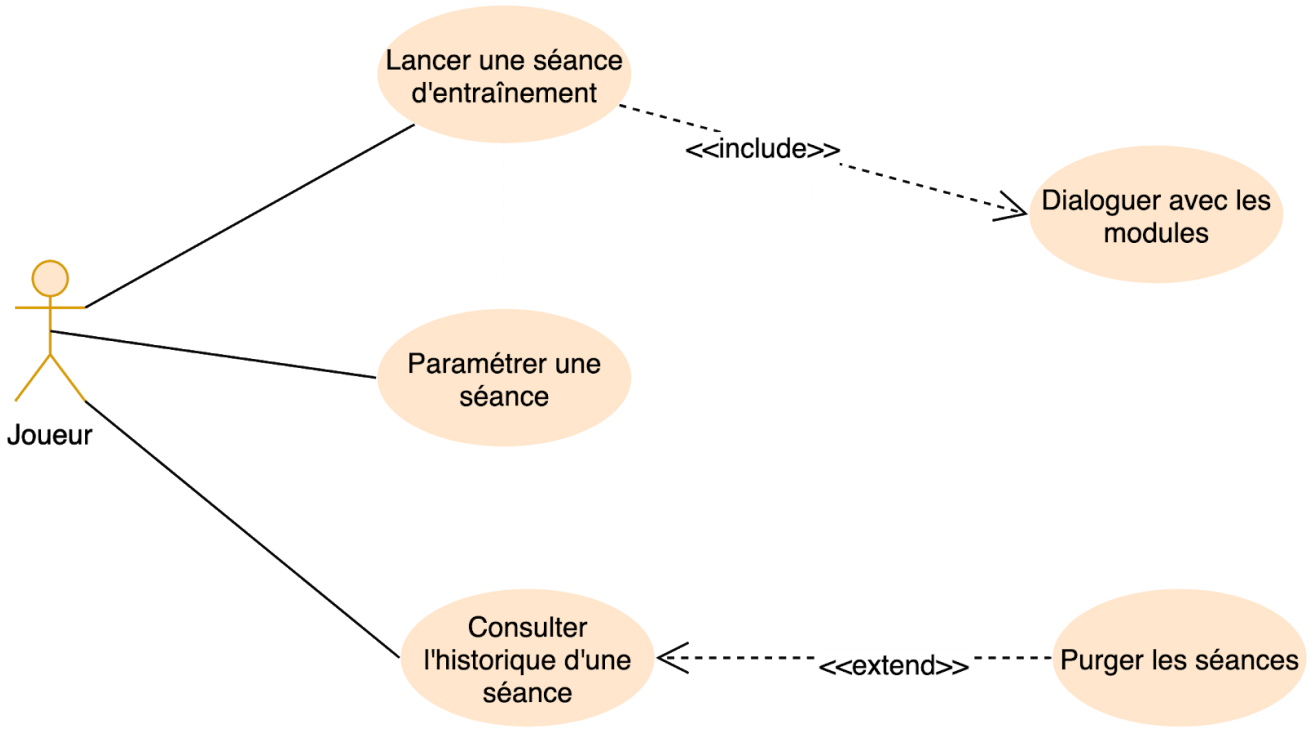
FenetreAccueil.qml

Une fois le signal reçu, nous pouvons effectuer les actions voulues.

```
onRobotDetecteChanged: {
      if(communicationBluetooth.peripheriqueRobotTTPADetecte)
      {
            console.log("ROBOT DETECTE")
            etatRobot.color = "#41CD52"
            communicationBluetooth.connecterPeripheriquesTTPA();
            boutonJouer.enabled = true;
      }
      }
```

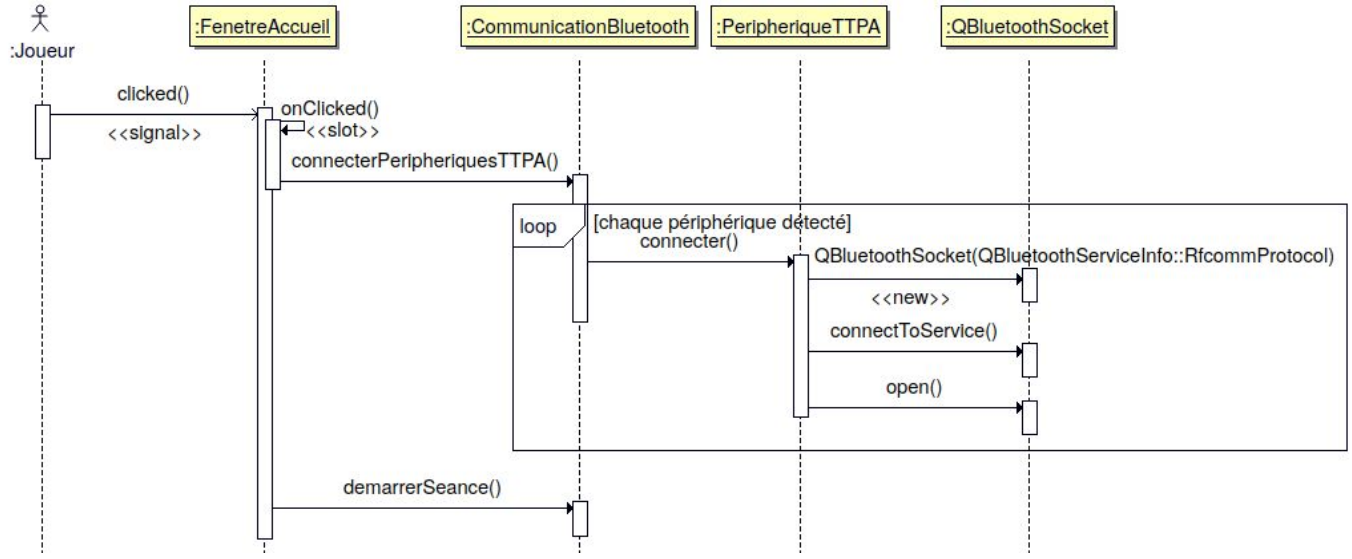
Conception logicielle de l'application

Diagramme des cas d'utilisations



Ici le joueur peut lancer une séance, qui inclut le fait qu'il doit communiquer avec les différents modules. Il a aussi la possibilité de paramétrer une séance et consulter l'historique d'une séance qui lui offre la possibilité s'il le souhaite de purger la séance.

Diagramme de séquence “Démarrer la séance”



Code source de la méthode demarrerSéance() :

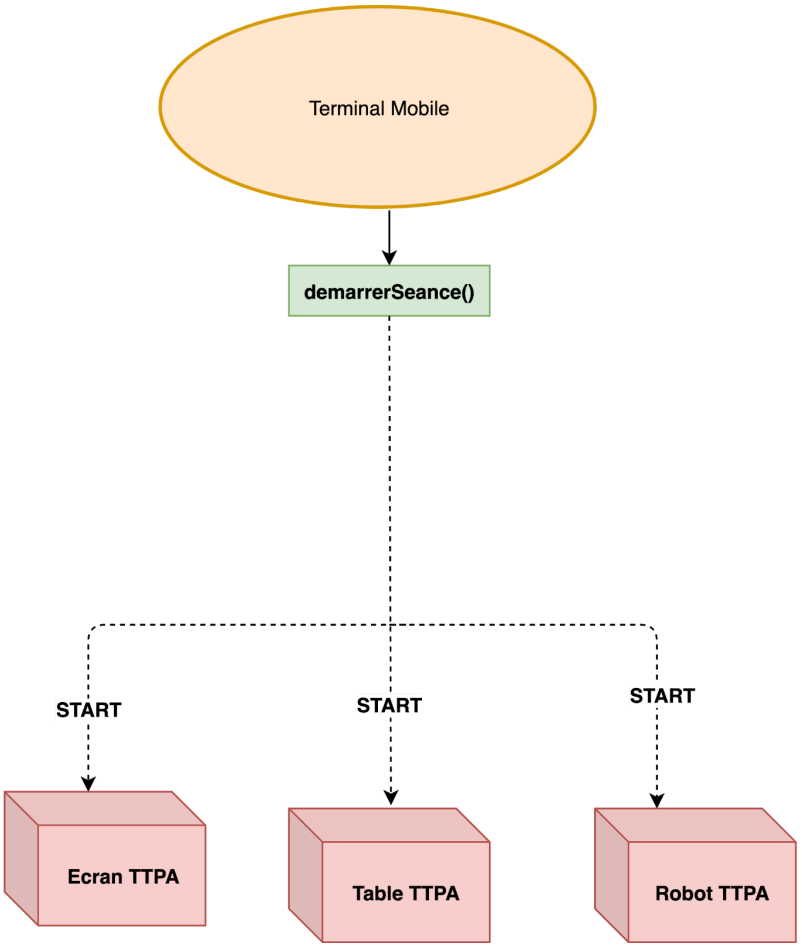
```
bool CommunicationBluetooth::demarrerSeance(QString nomJoueur, QString
zoneObjectif, QString zoneRobot, int frequenceBalle, int nbBalles, int
effet, int puissance, bool rotation)
{
      if(etatSeance == Initial)
      {
      demarrerSeanceRobot(frequenceBalle, nbBalles, effet, puissance,
rotation);
      demarrerSeanceEcran(nomJoueur, zoneObjectif, zoneRobot);
      demarrerSeanceTable(nbBalles);
      etatSeance = EtatDeLaSeance::EnCours;
      }
}
```

La méthode permet de vérifier si la séance est bien dans son état “**Initial**” puis si elle l’est, appelle les différentes méthodes permettant de démarrer la séance de chaque appareil et place la séance dans l'état “**En cours**”.

Envoi de trames

Le terminal mobile envoi des trames à tous les appareils du système TTPA, voici le schéma d'envoi des trames.

Démarrage de la séance :

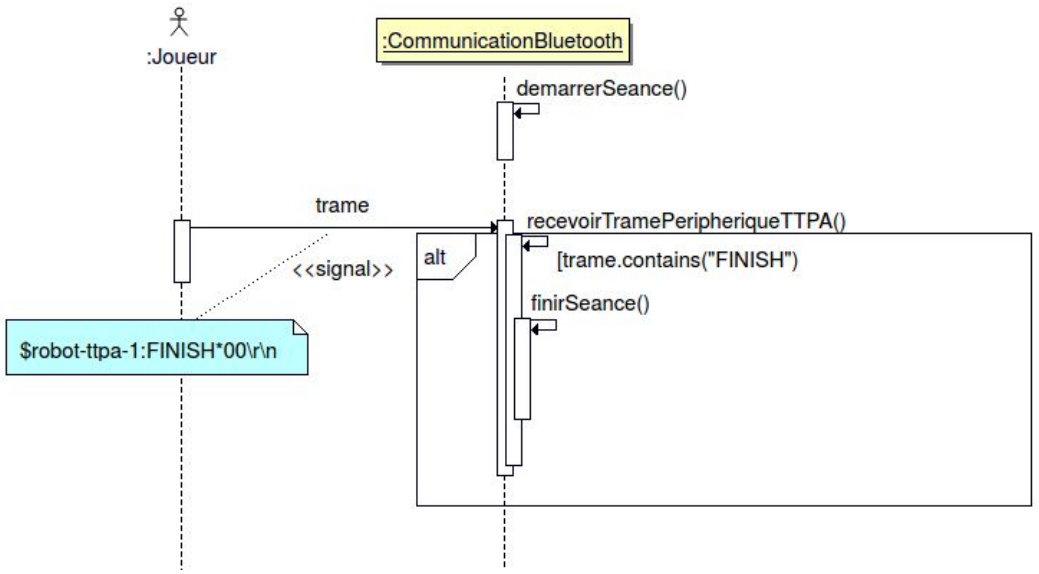


Au démarrage une trame "**START**" est envoyée aux appareils TTPA contenant chacune les paramètres requis à chaque appareil (voir Annexe Trame), le schéma ci-dessus est similaire pour chaque transition d'état du système (le terminal mobile envoie une trame à chaque appareil indiquant l'état dans lequel se situe la séance.

Fin “naturelle” de la séance

Lorsque le robot TTPA à envoyer sa dernière balle (l’utilisateur ayant paramétré le nombre de balles avant le démarrage de la séance), il envoi une trame “FINISH” au terminal mobile qui indiquera à tous les appareils, robot y compris que la séance est terminée.

Diagramme de séquence “Fin de séance”



Code source de la méthode finirSeance() :

```
bool CommunicationBluetooth::finirSeance()
{
      qDebug() << Q_FUNC_INFO;

      QString trameFin = ":END*00\r\n";

      if(etatSeance == EnCours)
      {
      envoyerPeripheriqueTTPA("robot-ttpa-11", trameFin);
      envoyerPeripheriqueTTPA("ecran-ttpa-2", trameFin);
      envoyerPeripheriqueTTPA("table-ttpa-11", trameFin);
      emit fini();
      etatSeance = EtatDeLaSeance::Arretee;
      return true;
      }
      return false;
}
```

Bluetooth

Bluetooth est une norme de communications permettant l'échange bidirectionnel de données à très courte distance en utilisant des ondes radio UHF sur une bande de fréquence de **2,4 GHz**. Son objectif est de simplifier les connexions entre les appareils électroniques (ici les appareils TTPA) en supprimant des liaisons filaires.

Effectivement l'**écran ttpa** ainsi que la tablette utilisée possède la version 4.1 du Bluetooth qui en outre des avantages de Bluetooth **4.0** (portée de connexion augmentée jusqu'à 100 mètres), Bluetooth **4.1** comprend plusieurs améliorations supplémentaires. Celles-ci concernent principalement la qualité de connexion notamment le taux d'erreur de transmission qui a pu ainsi être réduit d'environ ⅔.

Le **robot ttpa** utilise la version 4.2 du bluetooth qui permet une vitesse de transmission beaucoup plus rapide et donc une hausse de l'autonomie de l'appareil, la version 4.2 apporte aussi un nouveau chiffrement lors de l'appairage pour plus de sécurité.

La table ttpa elle, utilise la version 3.0 du bluetooth qui ne possède donc pas certaines fonctionnalités du à sa version antérieure.

Evolution du Bluetooth :

Année	Version	Fonctionnalité	Débits de données	Portée	Fréquence d'émission (porteuse)
2009	3.0	Haute vitesse à l'aide de la norme IEEE 802.11	2,1 Mbit/s	10 m	2,4 GHz
2010	4.0	Protocole ***Low-Energy***	3 Mbit/s	60 m	2,4 GHz
2013	4.1	Connexion de périphérique indirecte	3 Mbit/s	60 m	2,4 GHz
2014	4.2	Protocole IPv6 pour les connexions directes	3 Mbit/s	60 m	2,4 GHz

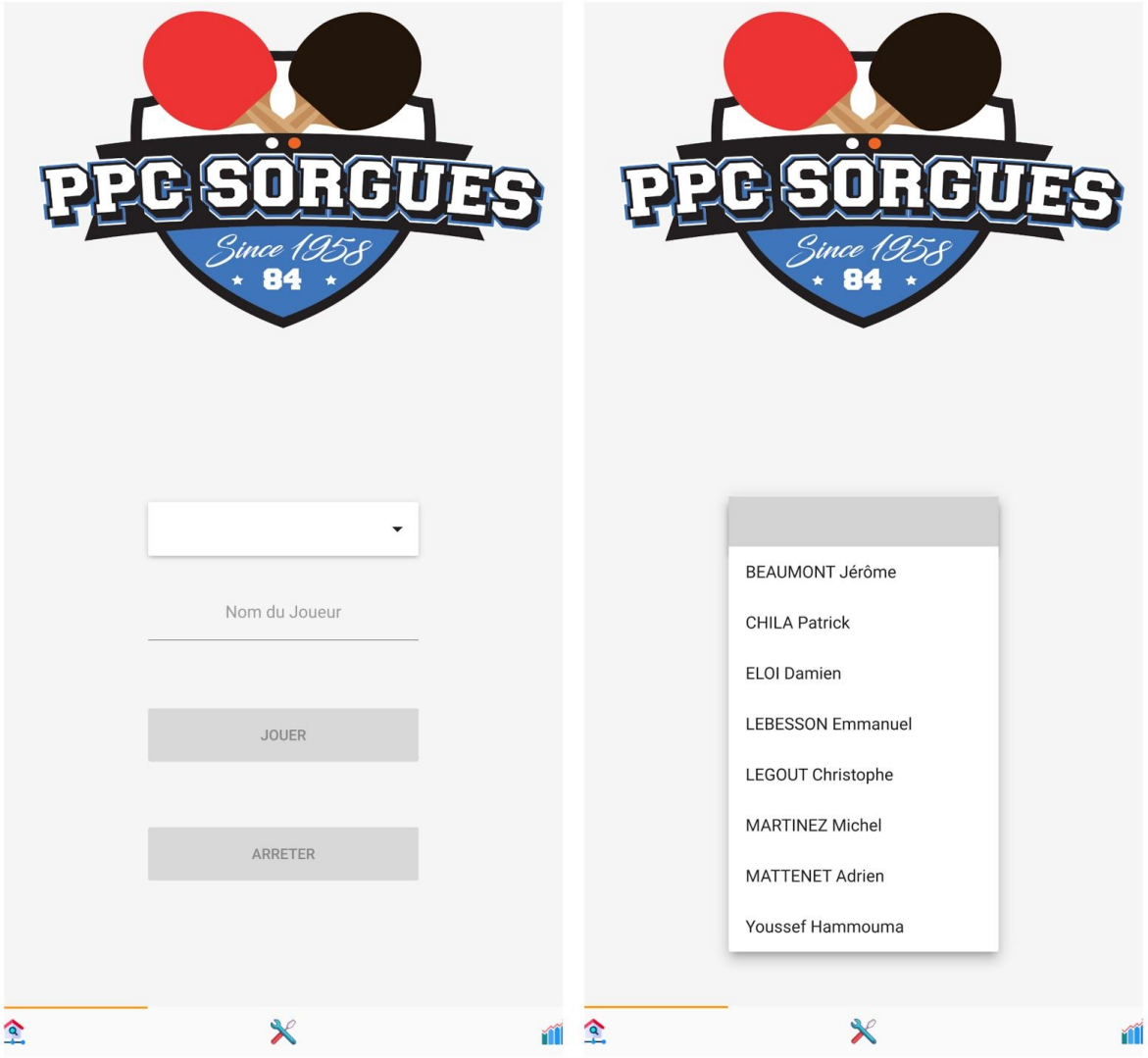
Application TTPA

L'application TTPA se décompose en 4 parties :

- L'accueil
- La connexion aux appareils TTPA (situé dans la page d'accueil)
- Les paramètres
- Les statistiques

Accueil

Permet le choix ou la création d'un joueur, ainsi que la connexions au différents appareils TTPA.



Dans la ComboBox, ce situe les joueurs de la table “**joueurs**” de la base de donnée.

Chaque Joueur à la possibilité de créer une séance qui lui sera ensuite associée. Par exemple ci-dessous le joueur “**BEAUMONT Jérôme**” à été sélectionné, sa séance associée est donc disponible dans la ComboBox située dans la partie “Réglages”.

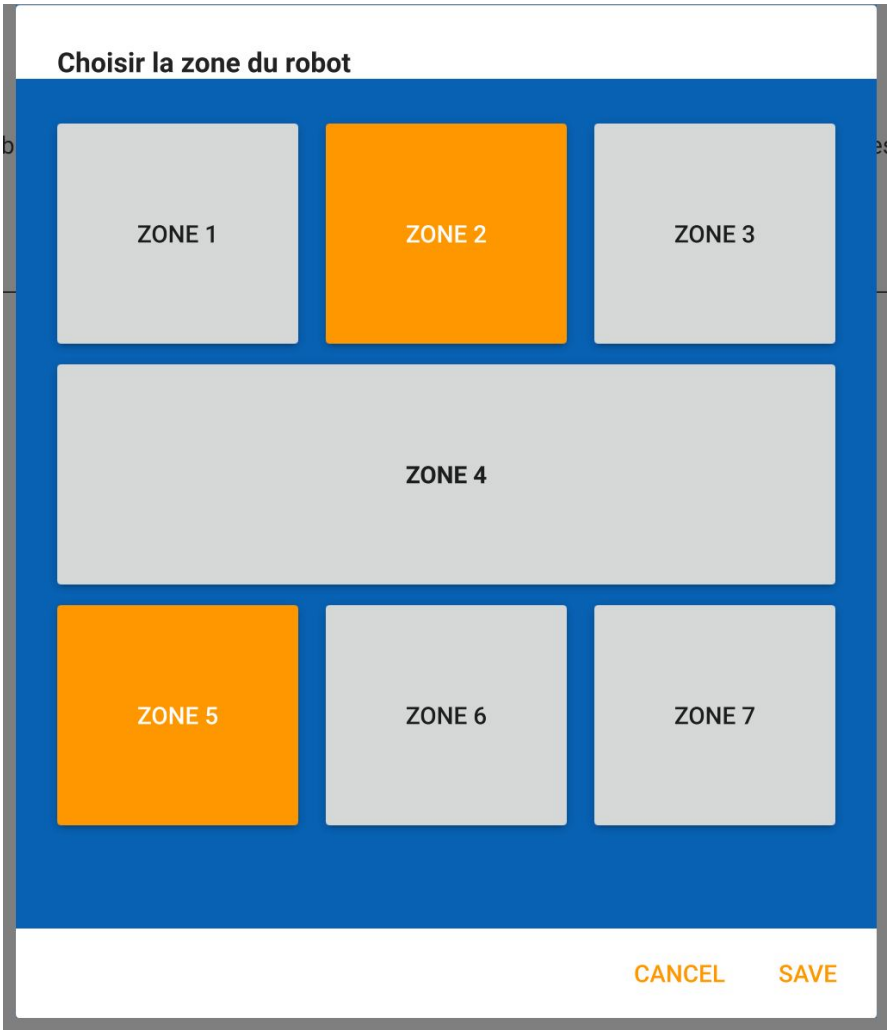
Paramètres

Permet au joueur de choisir les différents réglages du **robot ttpa** :

- Nombre de balles / minutes
- Nombre de balles
- Effet
- Puissance
- Rotation

La page “paramètres” permet aussi de sélectionner la zone objectif du joueur ainsi que la zone où sera placé le robot.

Choix des zones

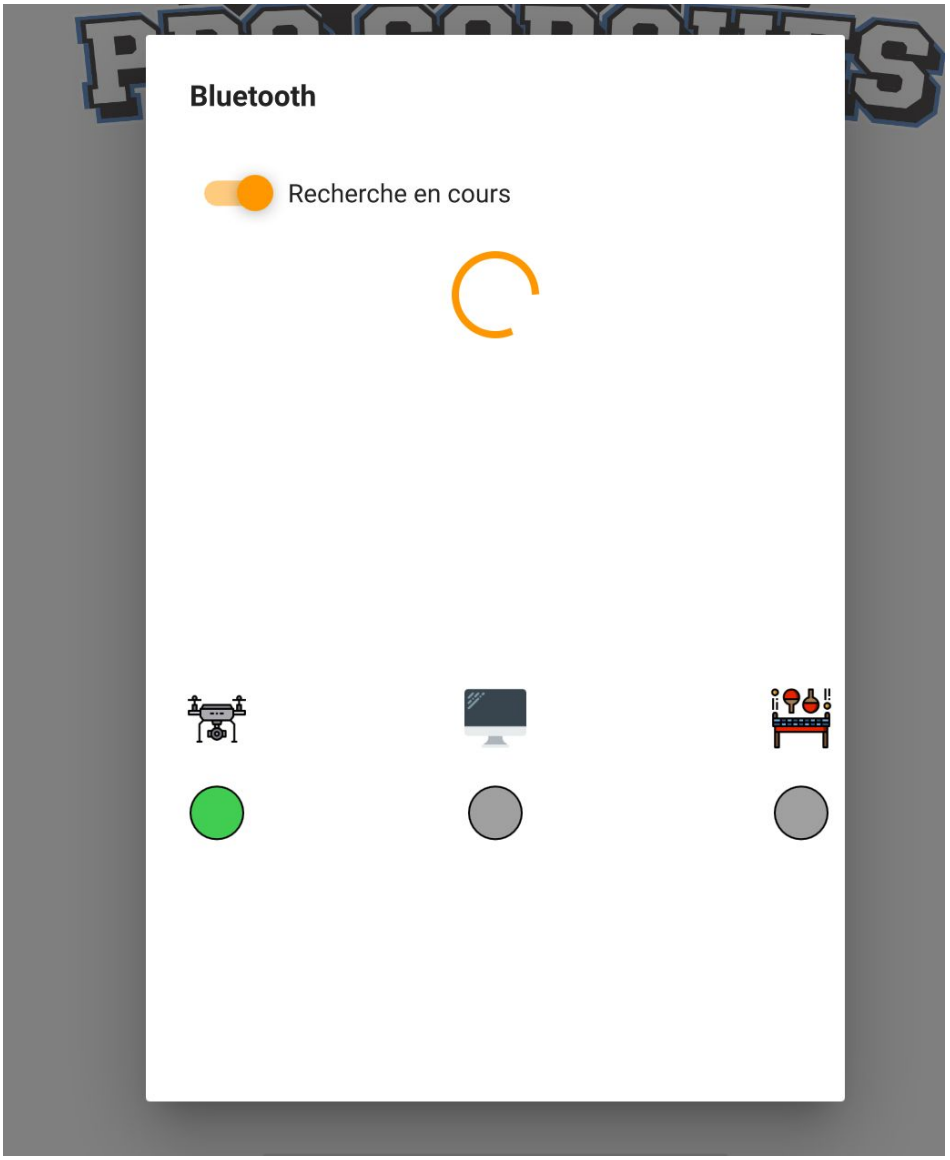


Connexion aux appareils TTPA

La connexion aux appareils s'effectue dans une "boîte de dialogue", lorsque la recherche est lancée par le joueur, l'application lance la recherche à l'aide de la méthode rechercherPeripheriquesTTPA(), voici son code source :

```
void CommunicationBluetooth::rechercherPeripheriquesTTPA()
{
	qDebug() << Q_FUNC_INFO;

	qDeleteAll(peripheriquesTTPA);
	peripheriquesTTPA.clear();
	peripheriqueTableTTPADetecte = false;
	peripheriqueRobotTTPADetecte = false;
	peripheriqueEcranTTPADetecte = false;
	emit detecte();
	emit peripheriquesTTPAUpdated();

	if(discoveryAgentDevice != NULL)
	{
	qDebug() << Q_FUNC_INFO << discoveryAgentDevice->isActive();
	discoveryAgentDevice->start();
	if (discoveryAgentDevice->isActive())
	{
		etatRecherche = true;
		emit recherche();
	}
	}
}
```

Une fois les appareils détectés, les voyants s'allument en vert puis le terminal mobile se connecte aux différents périphériques détectés.



Tests de validation

Description	OUI	NON
La base de donnée est complète	X	
La liaison Bluetooth est opérationnelle	X	
L'application mobile est fonctionnelle	X	

ANNEXE

Protocole Trame

Introduction

Le protocole est orienté **caractères ASCII**.

Format des trames

$ENTETE:TYPE;DATAS*XX\r\n

Les différents champs d'une trame :

- Début de trame : **$**
- Champ type : chaîne de caractères indiquant le type de trame
- Données composées de champs de longueur variable
- Séparateur 1 (identifiant dans l'entête) : "tiret" **-**
- Séparateur 2 (fin entête) : "deux points" **:**
- Séparateur 3 (données) : "point virgule" **;**
- Fin des données suivi du *checksum* : *XX (XX : valeur du *checksum*)
- Fin de trame : "\r\n"

Les différents types de trame :

- "START" : Démarrage de la séance)
- "PAUSE" : Mise en pause de la séance)
- "RESUME" : Reprise de la séance)
- "END" : Fin de la séance)
- "RESET" : Réinitialise la séance)
- "ERROR" : Indique une erreur
- "STAT" : Pour les statistiques (écran)
- "IMPACT" : Indique un impact dans une zone de la table
- "BILAN" : Nombre de balle en fin de séance

Format de l'entête

$NOM-PROTOCOLE-NUMERO:...

Les différents périphériques sont identifiés par les noms suivants :

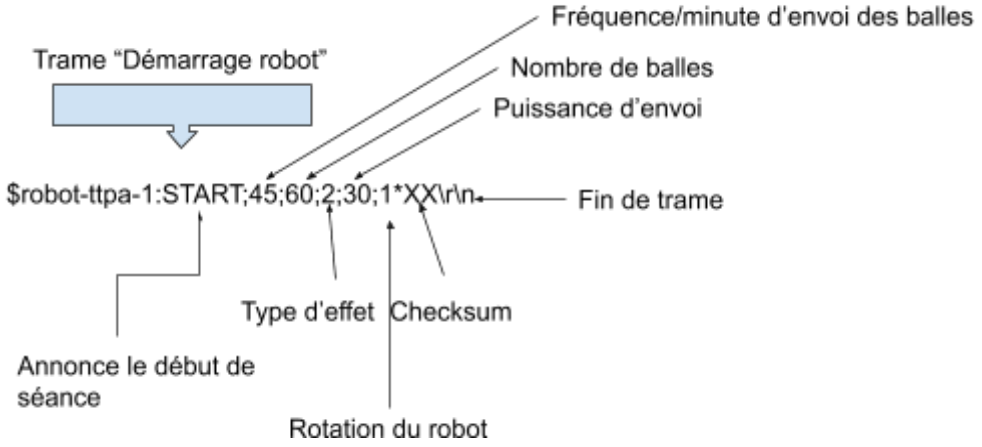
- **robot** : robot lanceur de balles
- **ecran** : écran de visualisation de statistiques
- **table** : module de détection des impacts de balles

Le numéro permettra d'identifier plusieurs périphériques de même type et d'associer les trois modules. Par exemple pour la table n°1, on aura les périphériques suivants : robot-ttpa-1, table-ttpa-1 et ecran-ttpa-1.

Robot

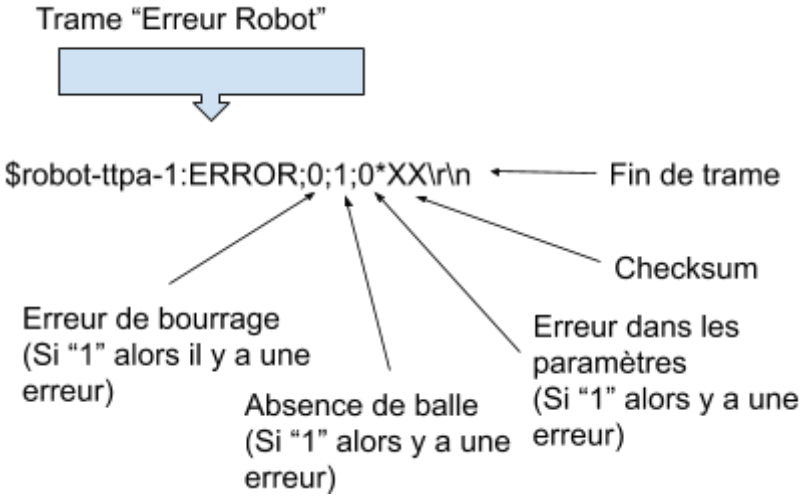
Trame "Démarrage Robot"

Sens : Tablette → Robot
Permet de paramétrer et démarrer une séance



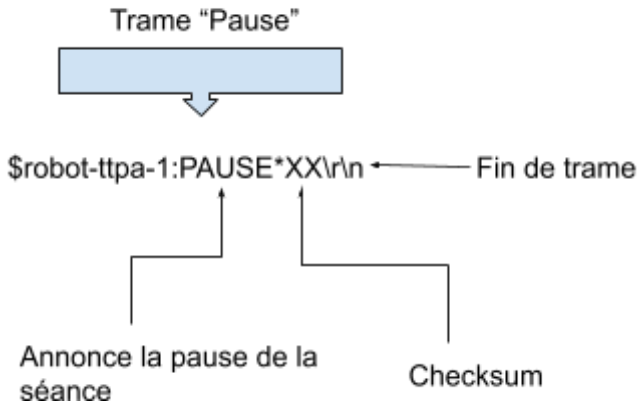
Trame "Erreur Robot"

Sens : Robot → Tablette
Permet de signaler une erreur liée au robot, l'erreur peut être un bourrage, un manque de balle, voire une erreur liée au paramétrage.



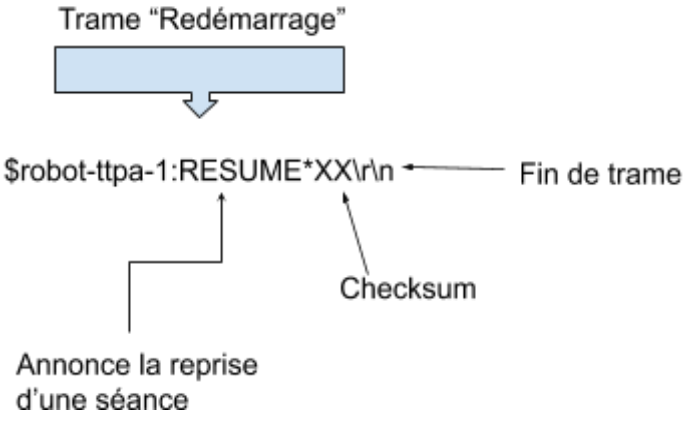
Trame “Pause”

Sens : Tablette → Périphériques TTPA



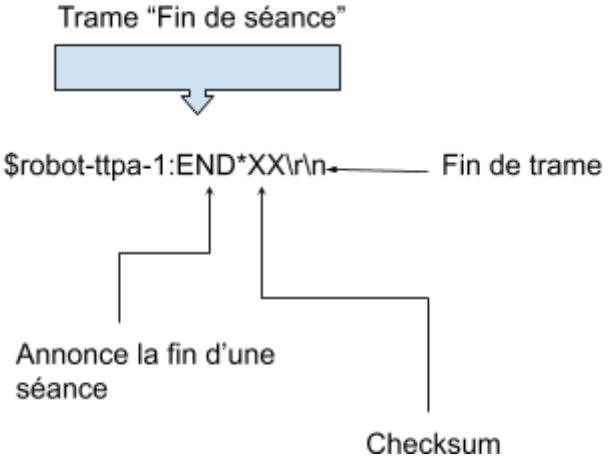
Trame “Redémarrage séance”

Sens : Tablette → Périphériques TTPA



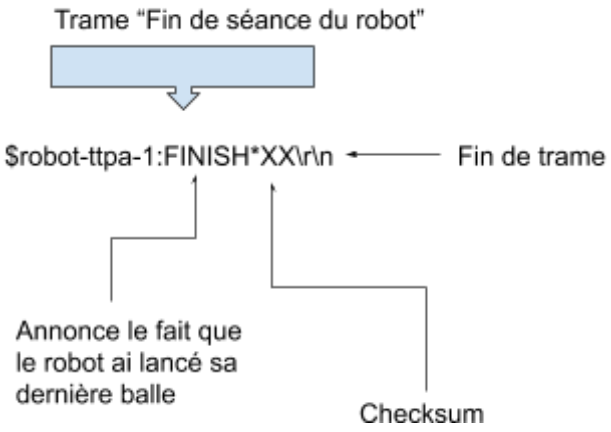
Trame “Fin de séance”

Sens : Tablette → Périphériques TTPA



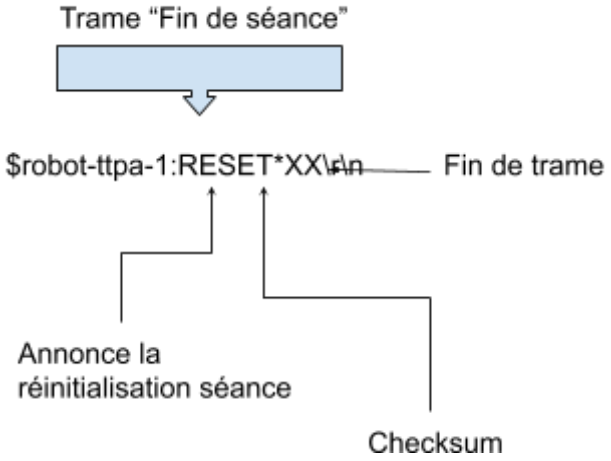
Trame “Fin de séance du robot”

Sens : Robot → Tablette TTPA



Trame “Réinitialisation de la séance”

Sens : Tablette → Périphériques TTPA

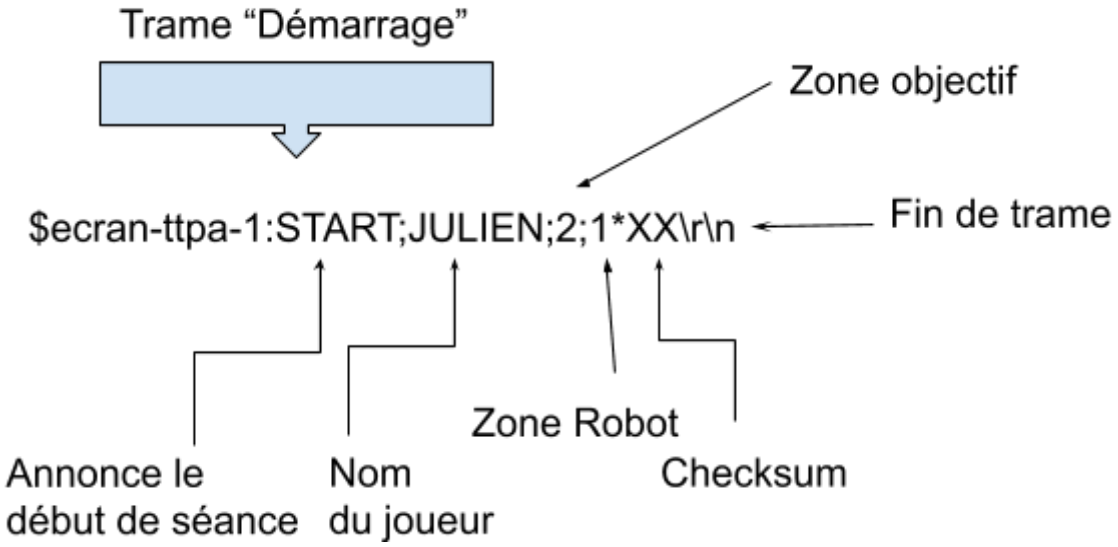


Modification de démarrage séance

Ecran

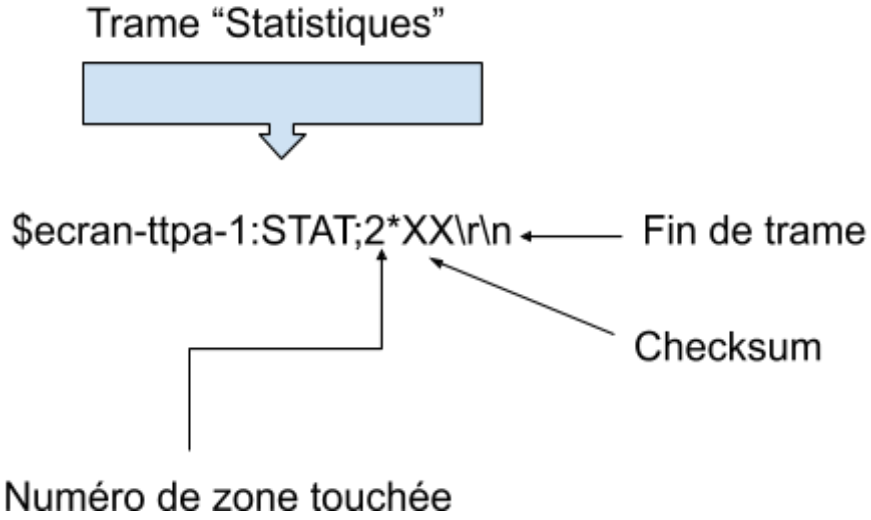
Trame “Démarrage séance”

Sens : Tablette → Écran



Trame “Statistiques”

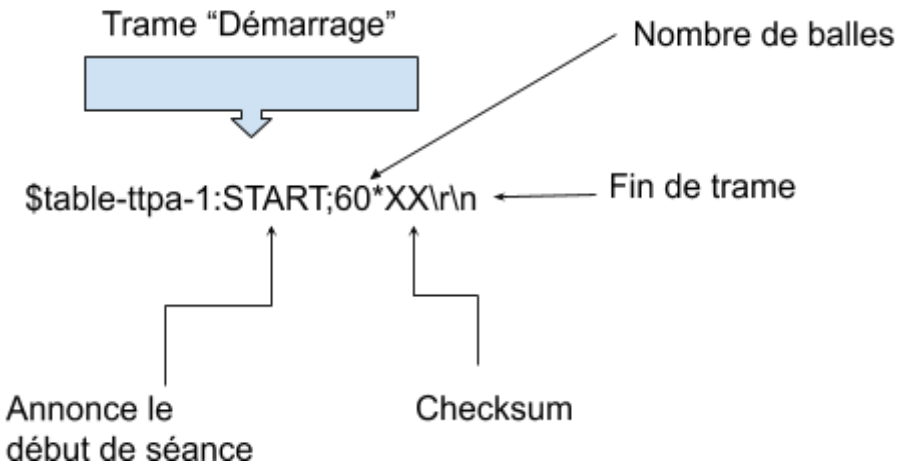
Sens : Tablette → Écran



Table

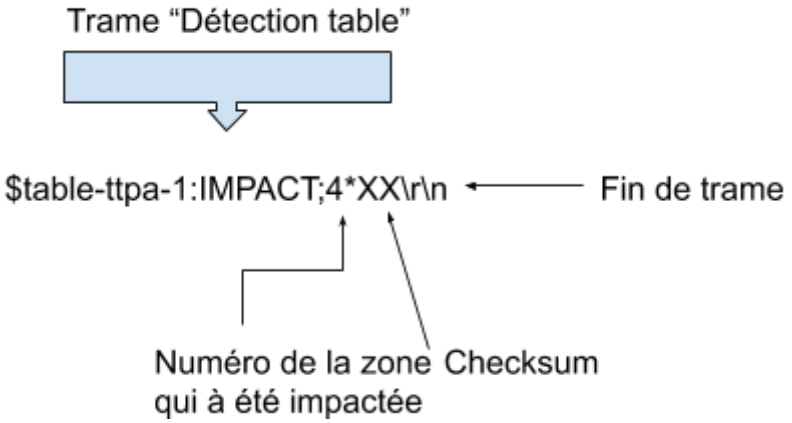
Trame “Démarrage séance”

Sens : Tablette → Table



Trame “Détection Table”

Sens : Table → Tablette
Permet de signaler un impact dans une zone de la table



Annexe Trames

Les données transmises :

- ❏ Fréquence d'envoi des balles par minute (20 à 70 balles/minute)
- ❏ Nombre de balles (+100)
- ❏ Effet appliqué à la balle type
- ❏ Puissance de frappe
- ❏ La rotation du robot (booléen)
- ❏ Nombre de balles totales (int)
- ❏ Nombre de balles touchées (qui ont été renvoyées sur la table) (int)
- ❏ Zone du robot (int comprise entre 0 et 6)
- ❏ Zone d'objectif (int comprise entre 0 et 6)
- ❏ Nom du joueur (chaîne de caractères)

Convention de Nommage

La documentation avec ***Doxygen*** pour les attributs: //!< ....
Exemple: int numeroZoneObjectif; //!< Placement de la zone de l'objectif

Nom de classe: NomDeClasse
Nom de variable: nomDeVariable
Nom de fichier: nomdefichier
La documentation avec Doxygen pour tous sauf les attributs : /** @.... */

L'indentation se fait avec quatre espace
Exemple:

```
/**
 * @brief récupère les donnés des balles récupère les donnés des balles
ayant touché la table
 *
 * @fn StatistiquesSeance::getballesToucheTable
 * @return
 */
int StatistiquesSeance::getballesToucheTable()
{
      return balleToucheTable;
}
```

Glossaire

TTPA : Table de Tennis Performance Analyser

UML : Unified Modeling Language

IHM : Interface Homme Machine

Terminologie:

Système d'exploitation : Un système d'exploitation est l'ensemble de programmes central d'un équipement informatique qui sert d'interface entre le matériel et les logiciels applicatifs.

Distribution Linux : Une distribution GNU/Linux est un ensemble de logiciels libres, assemblés autour du noyau Linux et de paquets GNU. Elles comprennent le plus souvent un logiciel d'installation et des outils de configuration.

GUI : C'est une interface graphique (GUI pour Graphical User Interface) est un dispositif de dialogue homme-machine (IHM), dans lequel les objets à manipuler sont dessinés sous forme de pictogrammes à l'écran.

Raspberry Pi: (nano-ordinateur monocarte à processeur ARM)

Mode kiosque : Le mode kiosque (ou kiosk) a été conçu à l'origine pour les bornes internet (Internet kiosks) et il est utilisé ici sur un PA. Il met en place un environnement logiciel restreint aux fonctions essentielles : empêcher les utilisateurs d'utiliser le système, dédier l'affichage graphique exclusivement à l'application et enlever les composants graphiques habituels (curseur souris, barre de titre, tableau de bord, icones, etc ...).

SSH : SSH (Secure Shell) est à la fois un programme informatique et un protocole de communication sécurisé. Le protocole de connexion impose un échange de clés de chiffrement en début de connexion. Le protocole SSH a été conçu avec l'objectif de remplacer les différents programmes rlogin, telnet, rcp, ftp et rsh.