Dossier Technique

Projet DARTS

Table des matières

Présentation générale 3

Expression du besoin 4

Présentation du projet 4

Synoptique du système DART 4

Le système DARTS est donc décomposé en trois modules : 4

Equipe de projet DARTS 5

Étudiants chargés du projet : 5

Répartition des tâches 5

Etudiant 1: Bounoir Fabien 5

Etudiant 2: Menella Erwan 6

Etudiant 3: Neyret Paul 6

Partie Bounoir Fabien (Étudiant IR) 7

Rappel du Besoin Initial 7

Objectifs du Module de visualisation de performance (ÉCRAN-DARTS) 7

Description structurelle du système 8

Zone d'impact de la cible 9

Tâches 9

Organisation commune au sein du projet 10

Outils de développement 10

Diagramme de déploiement 11

Spécification du Raspberry Pi 4 11

Le Bluetooth (partie Science Physique) 12

Evolution du bluetooth : 12

Caractéristique bluetooth 5.0: 13

Analyse 14

Planification 15

État d'une séance 16

Les différentes trames DART 17

L'interface Homme / Machine (IHM) 18

L'écran d'attente : 18

L'écran de règles : 19

L'écran de jeu : 20

L'écran de pause : 20

L'écran de fin : 22

- Mode Tournois 23

Lorsque le duel et fini un récapitulatif de la partie est affiché : 24

Pour lancer le duel suivant il suffit de renvoyer une trame Play: 24
Gestion de l'affichage de la cible 25
Diagramme de classes **26**
Scénario démarrage partie 27
Scénario d'impact fléchette sur la cible 28
Scénario trame STOP 29
Décodage des trames 30
Test de validation 32

Annexe **33**

Protocole Trame **33**
1 . Caractéristiques générales 33

Convention de Nommage **37**

Glossaire **37**

Terminologie **38**

Video **38**
Presentation mode de jeu 38
Description Ecran-DARTS 38

- Présentation générale

Le système **DARTS** permettra de jouer à plusieurs jeux de fléchettes différents. Cependant, il existe des règles officielles adoptées dans tous les tournois. En France, la pratique est régie par Fédération française de darts et dans le monde par la World Darts Federation. Il existe aussi une fédération dissidente (professionnelle) : la Professional Darts Corporation.

- Expression du besoin

La réalisation d'une application mobile connectée au différents appareils dans le but de gérer une partie de fléchettes à un ou plusieurs joueurs.

Le projet consiste à pouvoir paramétrer une partie de fléchettes à l'aide de l'application mobile qui communique avec la cible ainsi que l'écran de visualisation de la partie.

- Présentation du projet

Le système devra :

- Permettre de lancer une partie de fléchette
- Afficher un retour visuel aux différents joueurs
- Archiver les différents partie
- Afficher les statistiques de partie

Le système sera équipé de :

- 1 cible de fléchette électronique
- 1 ecran Raspberry Pi
- 1 tablette mobile

- Synoptique du système DART

Le système DARTS est donc décomposé en trois modules :

- **Module de gestion de partie (Mobile-DARTS) :** les joueurs paramètrent et lancent la partie à partir d'une application sur un terminal mobile (sous Android) ;

- **Module de détection des impacts (Cible-DARTS) :** la cible est équipée de capteurs permettant d'identifier la zone impactée par les fléchettes envoyées par les joueurs ;

- **Module de visualisation de partie (Écran-DARTS) :** les joueurs, les arbitres et le public peuvent visualiser en "temps réel" le déroulement de la partie (nombre de manche, point restant dans la manche, moyenne des volées, ...) sur un écran de télévision.

- Equipe de projet DARTS

Étudiants chargés du projet :

Option IR :

- ➢ Etudiant 1 : Bounoir Fabien
- ➢ Etudiant 2 : Mennella Erwan

Option EC :

- ➢ Etudiant 3 : Neyret PAUL

- Répartition des tâches

Etudiant 1: Bounoir Fabien

Module : Module de visualisation de performance (Écran-DARTS)
➔ Afficher un écran d'accueil
➔ Visualisation de l'impact de la fléchette sur la cible
➔ Visualisation des données de la partie en temps réel (nom joueur, score, manche, moyenne volées,la durée de la partie, solution pour finir la partie).
➔ ecran de fin de partie pour visualiser le gagnant , et les différente information de la partie

- 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

Etudiant 2: Menella Erwan

Module : Module de gestion de séance (Mobile-DARTS)
➔ enregistrer les joueurs
➔ Paramétrer une partie de fléchette
➔ Démarrer une partie de fléchette
➔ Enregistrer les données des séances
➔ Consulter l'historique des partie
➔ Dialoguer avec les modules

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

- Documentation :
    - dossier technique et les documents relatifs au module

Etudiant 3: Neyret Paul

Module : Module de détection des impacts (cible-DARTS)
➔ Détection des impacts sur la cible
➔ Identification de la zone toucher
➔ Dialoguer avec le terminal mobile

- Installation :
    - le module d'acquisition d'impact et de transmission de données

- Mise en oeuvre :
- La matrice de la cible
- la carte d'acquisition des impacts

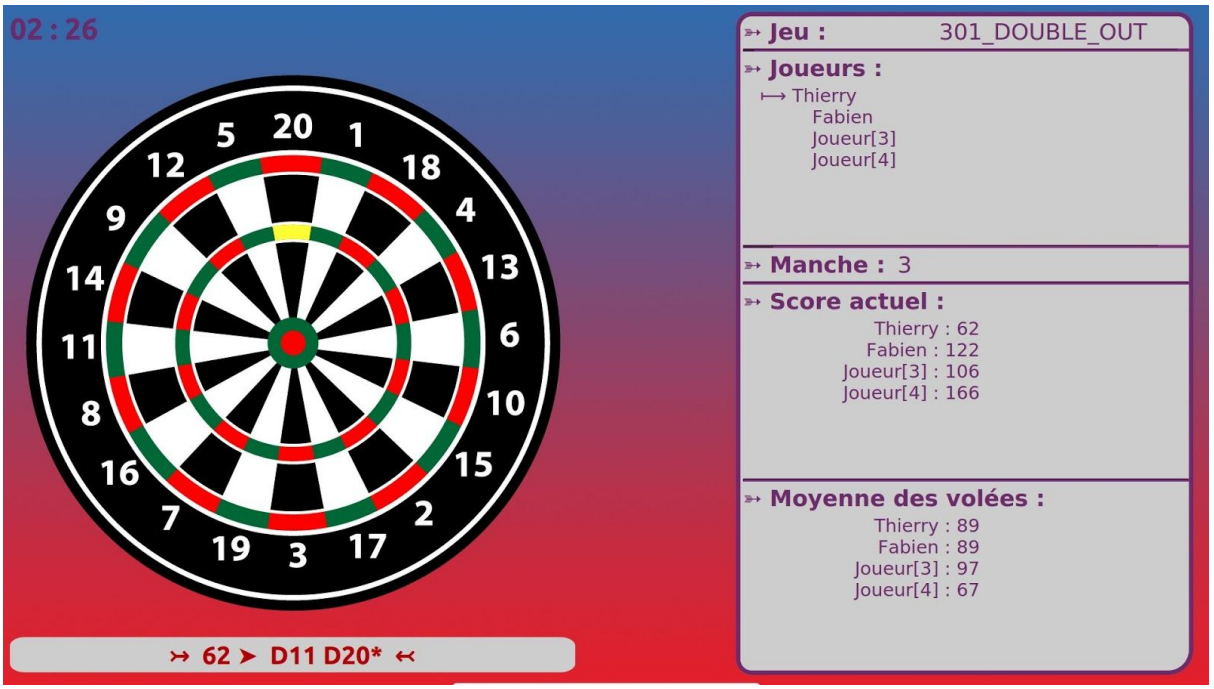
Partie Bounoir Fabien (Étudiant IR)

- Rappel du Besoin Initial

Le système DARTS est un système numérique permettant de jouer au jeu de fléchettes électroniques. Le joueur peut lancer une partie avec les différentes configurations qu'il souhaite.

- Objectifs du Module de visualisation de performance (ÉCRAN-DARTS)

Les joueurs peuvent visualiser en "temps réel" l'impact des fléchettes sur la cible, les scores, les moyennes des volées, les solutions pour finir la partie.



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

Le module de visualisation de partie doit permettre :

- d'afficher un écran d'accueil;
- d'afficher les noms des joueurs (si existant);
- d'afficher le type de jeu;
- d'afficher le numéro de la manche;
- d'afficher les scores des différents joueurs;

- de visualiser l'impact des flèches sur la cible en "temps réel";
- la plus haute et la moyenne des volées de chaque joueur;
- de dialoguer avec le terminal mobile.

- Description structurelle du système

Le système DARTS est donc décomposé en trois modules :

- Module de gestion de partie (Mobile-DARTS) : l'application sur un terminal mobile (sous Android) permet de configurer et lancer une partie ;
- Module de détection des impacts (Cible-DARTS) : un système électronique permet d'identifier la zone impactée par les fléchettes envoyées par les joueurs ;
- **Module de visualisation de partie (Écran-DARTS)** : l'application réalisée en Qt sur une Raspberry Pi 4 reliée à un écran de télévision permet de visualiser en "temps réel" le déroulement de la partie.

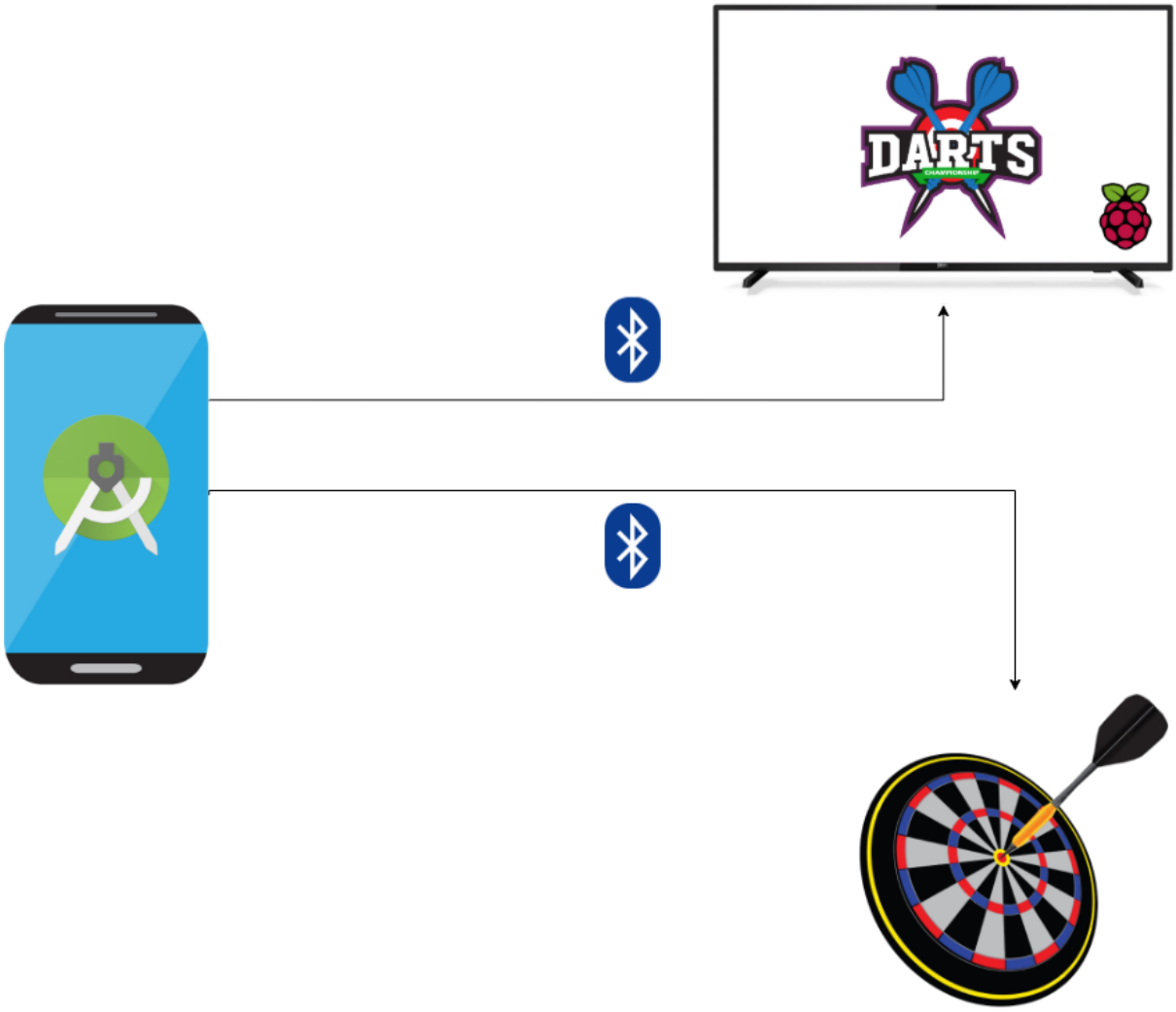


Figure 1: Description structurelle du système

L'ensemble des modules communique en Bluetooth en utilisant le protocole DART.

- Zone d'impact de la cible

Les différentes "zone" de la cible sont :

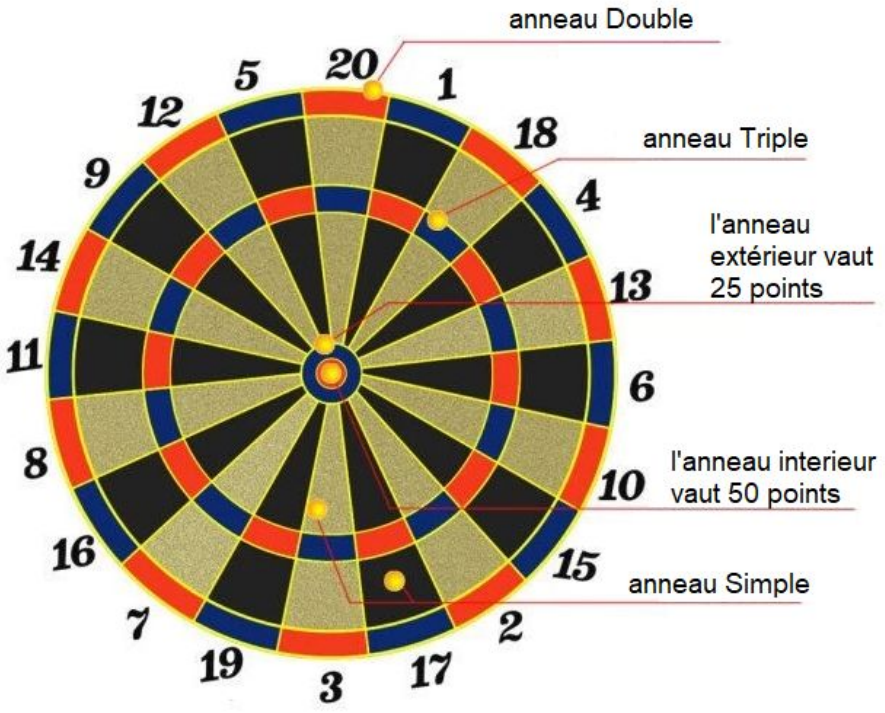


Figure 2: les différentes zones de la table

- Tâches

Installation :

L'environnement de développement

Mise en oeuvre :

La Raspberry Pi 4, et la liaison sans fil

Réalisation :

Dialoguer avec le terminal mobile,
L'ihm de l'application
Le code source de l'application

Configuration :

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

- Organisation commune au sein du projet

Afin d'avoir une meilleur organisation, communication et gestion des fichiers et codes sources, nous avons utilisé :

- Subversion 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

Les fichiers sources sont stockés sur un serveur que l'on nomme référentiel. il conserve la dernière version de chaque fichier, mais également toutes les versions précédentes.

- Outils de développement

Description	Version
Raspberry Pi 4	GNU/Linux Raspbian version 4.19
Planification	Trello + Trello gantt
Journal de bord	Roadbook (google drive)
Diagramme UML	BOUML 7.9.1
Gestion de versions	RiouxSVN (subversion) Version 1.9.3
Documentation de code	Doxygen Version 1.8.11
Langage utilisé	C++ / avec le framework Qt 5.11.3

Figure 3: tableau des différents outils de développement utilisés

- Diagramme de déploiement

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

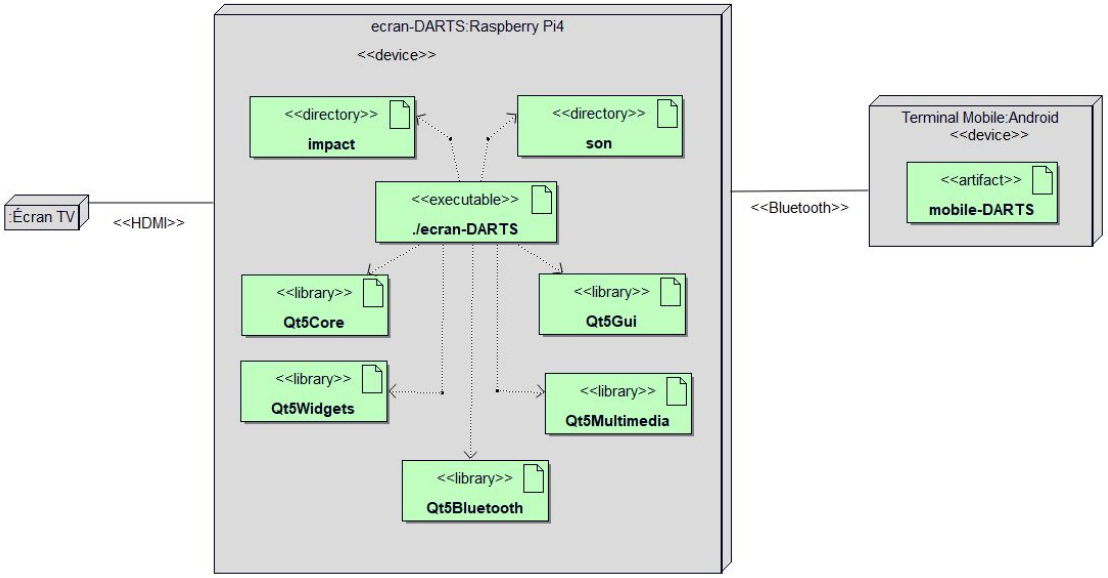


Figure 4: déploiement du système ecran-DARTS

- Spécification du Raspberry Pi 4

Raspberry PI 4	Caractéristiques
Système d'exploitation	Raspbian GNU/Linux
Processeur	Quad Core 1,5 GHz Broadcom BCM2711 ARMv8 CPU
RAM	4GB
Stockage	Micro SD 32 go
Sans-Fil	2.4 GHz et 5.0 GHz IEEE 802.11ac Wireless, Bluetooth 5.0 BLE
Connectiques	40-pin GPIO, 2 port USB 3.0, 2 port USB 2.0, Ethernet 10/100/1000 2 × micro-HDMI ports, 5V DC via USB-C

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

- Le Bluetooth (partie Science Physique)

Le bluetooth est une norme de communications permettant l'échange bidirectionnel de données à courte distance en utilisant des ondes radio UHF sur une bande de fréquence de **2.4 GHz** dont l'exploitation ne nécessite pas de licence vu la faible puissance d'émission. Le bluetooth établit donc une connexion sécurisée de proximité. il est utiliser pour simplifier les connexions entre les appareils électroniques(ici les appareils DARTS) en supprimant des liaisons filaires.

La communication bluetooth s'effectue via une relation de type maître / esclave.

La nouvelle version du protocole, Bluetooth LE (Low Energy), nécessite beaucoup moins d'énergie que le WiFi, de plus les développeurs travaillent sur la possibilité de faire un réseau maillé, ce qui permettrait à plusieur composant de communiquer entre eux.

→ la Raspberry pi est équipé du bluetooth 5.0 BLE tandis que la tablette est équipé du bluetooth 4.1

Evolution du bluetooth :

Version	Date	Améliorations
1.0	1999	
2.0	2004	Débit pratique supérieur porté à 2,1 Mbit/s. Rétrocompatibilité, réduction de la consommation des périphériques et optimisation des transferts
2.1	2007	
2.1+EDR	2007	Couplage plus simple et plus rapide. Sécurité renforcée et ajout du mode de connexion « NFC » (Near Field Communication) qui autorise des liaisons à très courte portée
3	2009	Débit théorique supérieur porté à 24 Mbit/s (Bluetooth v3.0 + HS)
4+LE	2010	Réduction de la consommation des périphériques(Low Energy). Haute vitesse basée sur le Wi-Fi. Logo "Bluetooth Smart Ready". Restitutions musicales stéréophoniques de qualité comparable au CD
4.1	2013	Connexion d'appareils multiples sur un seul accès pour la sortie du LTE.
4.2	2014	Réduction de la consommation des protocoles IP sécurisés pour les objets connectés

5	12/2016	Débit pratique supérieur en low energy (2 Mbit/s PHY) , portée de 40 m à 350 m et jusqu'à 500 mètres avec certains modules.

Caractéristique bluetooth 5.0:

- ➔ Portée de 40 m à 350 m et jusqu'à 500 mètres avec certains modules.
- ➔ Le système Bluetooth opère sur des bandes de fréquences comprise entre 2400 et 2483,5 MHz
  - ◆ Cette bande est divisée en canaux de 1 Mhz soit 79 canaux au total. Pour transmettre des datas, la technologie Bluetooth utilise le FHSS (Frequency Hopping Spread Spectrum).

- ➔ Le principe du FHSS est la commutation rapide entre plusieurs canaux de fréquence, utilisant un ordre pseudo aléatoire connu tant à l'émetteur qu'au récepteur pour la synchronisation.
  - ◆ cela permet : la synchronisation parfaite entre l'émetteur et le récepteur, d'émettre à plusieurs simultanément, de limiter les interférences (collisions).

On compare souvent Bluetooth à des soi-disant concurrents, comme WiFi et d'autres du même type. Bluetooth convient parfaitement aux applications qu'on lui confie tout comme WiFi a les siennes. Les réseaux WiFi sont relativement performants au niveau du débit, par contre ils consomment plus d'énergie. Contrairement à Bluetooth qui a un débit relativement faible (2Mbits/s, WiFi ~11-54Mbits/s) mais par contre consomme peu d'énergie. Chaque technologie doit être utilisée dans le but premier pour laquelle elle a été conçue.

- Analyse

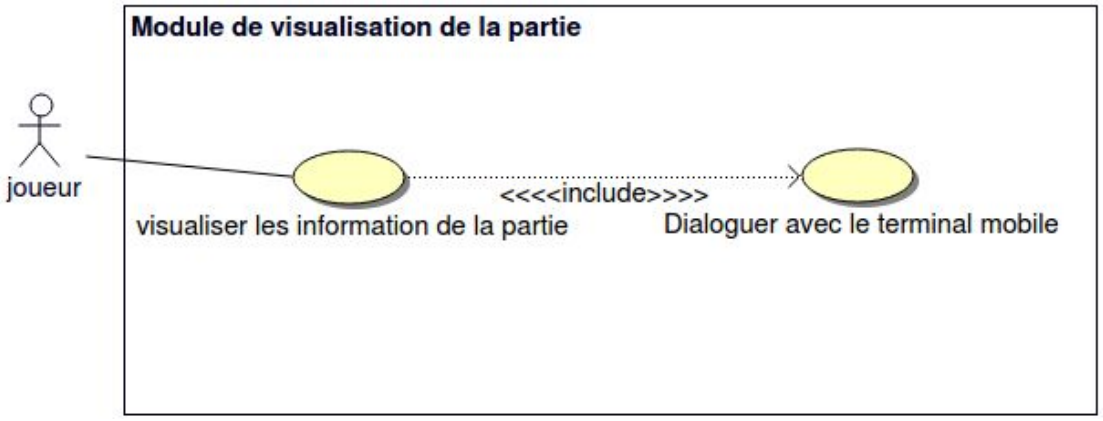


Figure 6: cas d'utilisation

Un ou plusieurs "Joueur" peuvent visualiser tout au long de la partie les différentes données (Score, Manche, Moyenne Volées, temps Partie, Impact fléchette).

Pour cela, le logiciel doit récupérer et traiter les trames émises par le terminal mobile.

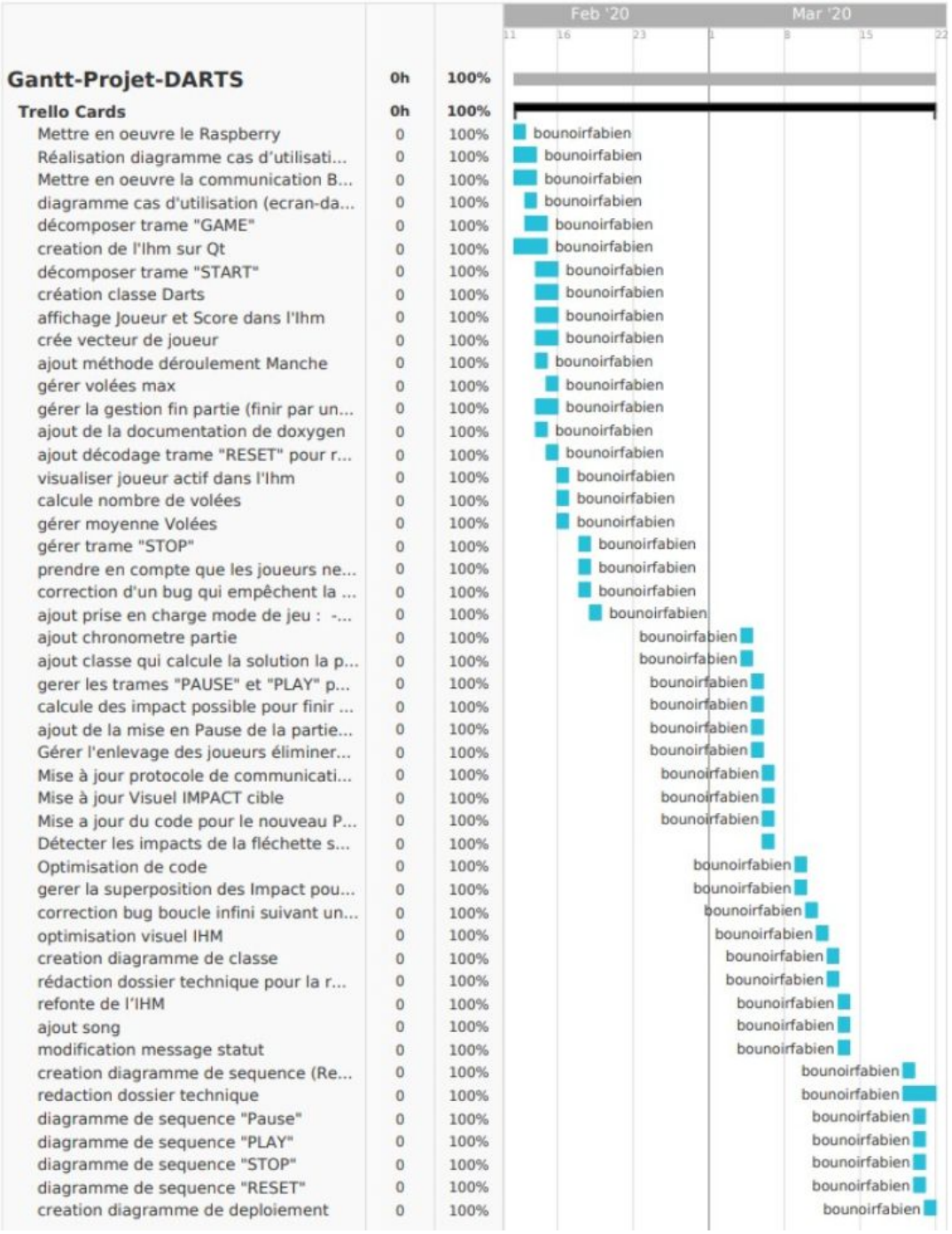
Les données visualisées sont :

- Le nom des différents joueurs (si définis)
- La durée de la partie
- Le score des différents joueurs
- La manche actuelle
- La moyenne des volées
- La volées maximale atteinte

La communication entre le logiciel et le terminal mobile est basée sur un protocole spécifique DART.(voir annexe)

- Planification

- Le développement de la partie communication Bluetooth avec la création du protocole DART étaient les plus logiques car ce sont les parties les plus importantes du projet.
- Ensuite, l'IHM a été développé en commençant par afficher les impacts des fléchettes.
- Le déroulement d'une partie a été implémenté.
- Pour finir, les différents modes de jeu ont été intégrés.



- État d'une séance

Une partie de fléchette 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.

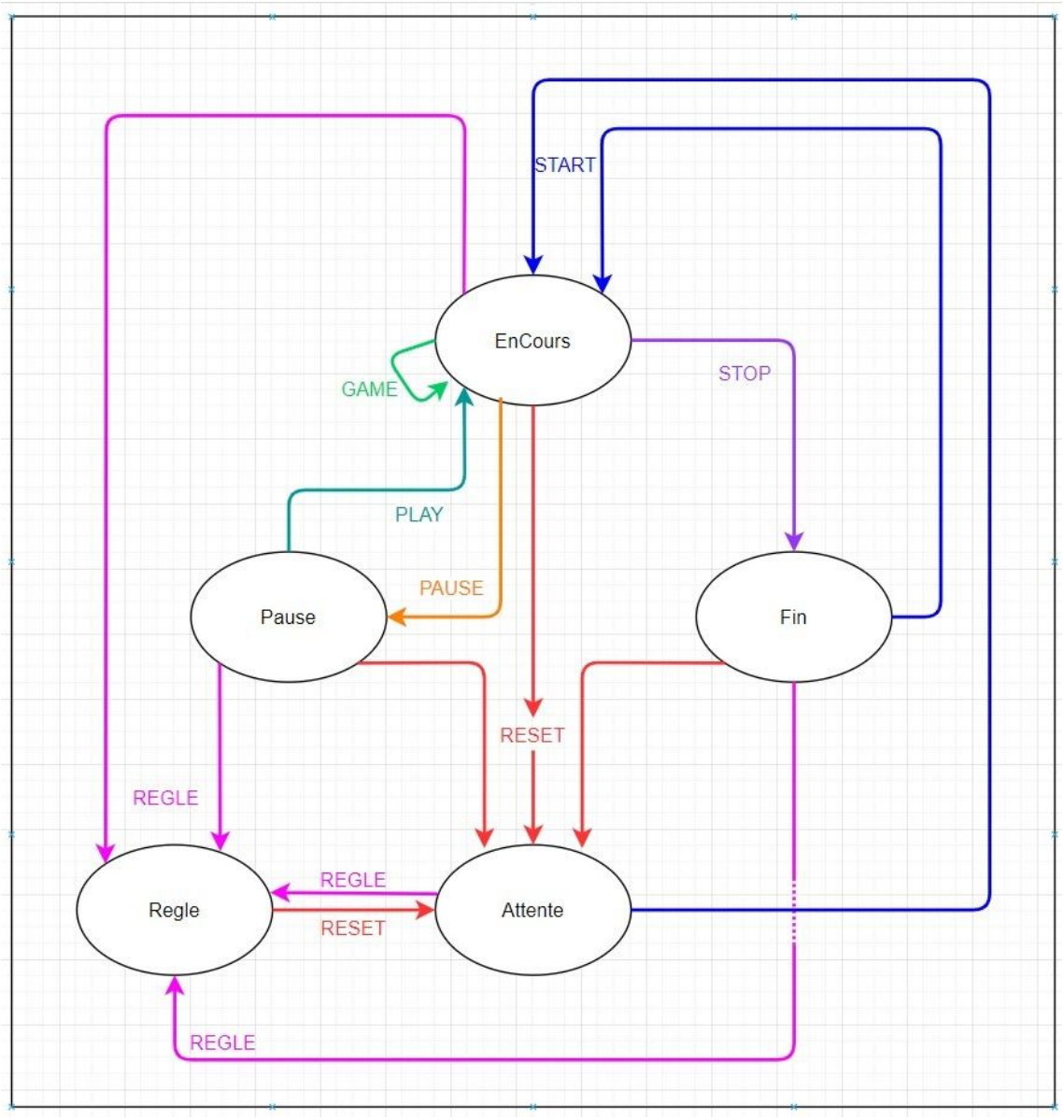


Figure 7: Les différents états d'une séance.

- Les différentes trames DART

Toutes les trames sont envoyées par le terminal mobile vers le module ÉCRAN-DARTS.

Trame	Description
$DART;START	Cette trame permet de choisir le type de partie, le nombre de joueurs et éventuellement leurs noms.
$DART;GAME	Cette trame indique l'impact d'une fléchette sur la cible.
$DART;PAUSE	Cette trame permet de mettre en pause la partie.
$DART;PLAY	Cette trame permet de reprendre la partie.
$DART;STOP	Cette trame permet de basculer vers l'écran de statistique et de finir la séance.
$DART;REGLE	Cette trame permet l'affichage des règles du mode de jeu choisie.
$DART;RESET	Cette trame permet de réinitialiser la partie qui était en cours et renvoie à l'écran d'attente.

Figure 8: Tableau des différentes trames

La trame **START** met le programme dans l'état **EnCours**, cet état peut se terminer avec 2 possibilités : la trame **RESET** ou la trame **STOP** .
En recevant une trame **PAUSE**, on va passer à l'état **Pause**. Il y a seulement deux possibilités pour quitter cet état : la trame **RESET** ou **PLAY**.
La trame **PLAY** fera passer de l'état **Pause** à l'état **EnCours.**
La trame **END** provoque le passage l'état **EnCours** à l'état **Fin**.
La trame **REGLE** fera passer l'application dans l'état **Regle** cet état s'apparente à un état d'attente, lorsque la vidéo d'explication des règles sera finie cela fera repasser l'application dans l'état où se trouvait.
La trame **RESET** fera passer à l'état **Attente** quelque soit l'état de la partie.

```
enum EtatPartie
{
      Attente = 0,
      EnCours = 1,
```

```
      Fin = 2,
      Pause = ,
     Regle
};
```

- L’interface Homme / Machine (IHM)

L’IHM a été réalisée avec Qt, un *framework* C++.

Le programme est organisé en plusieurs fenêtres gérées à l’aide d’un **`QStackedWidget`**. Il stocke une pile de fenêtres où une seule est visible à la fois. Les différentes fenêtres sont réalisées sur la base d’un **`QWidget`**.

L’affichage des textes utilise des **`QLabel`** et les images des **`QPixmap`**.

L’affichage “écran d’attente” est la première page, il s’agit de l'écran de connexion, elle est affichée au lancement du programme.

L’écran d’attente :

Figure 9: écran d’attente

Dans cet état, la trame attendue est :

- $DART;START;501;4;Fabien;Erwan;Thierry;Arthur\r\n

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

L’écran de règles :

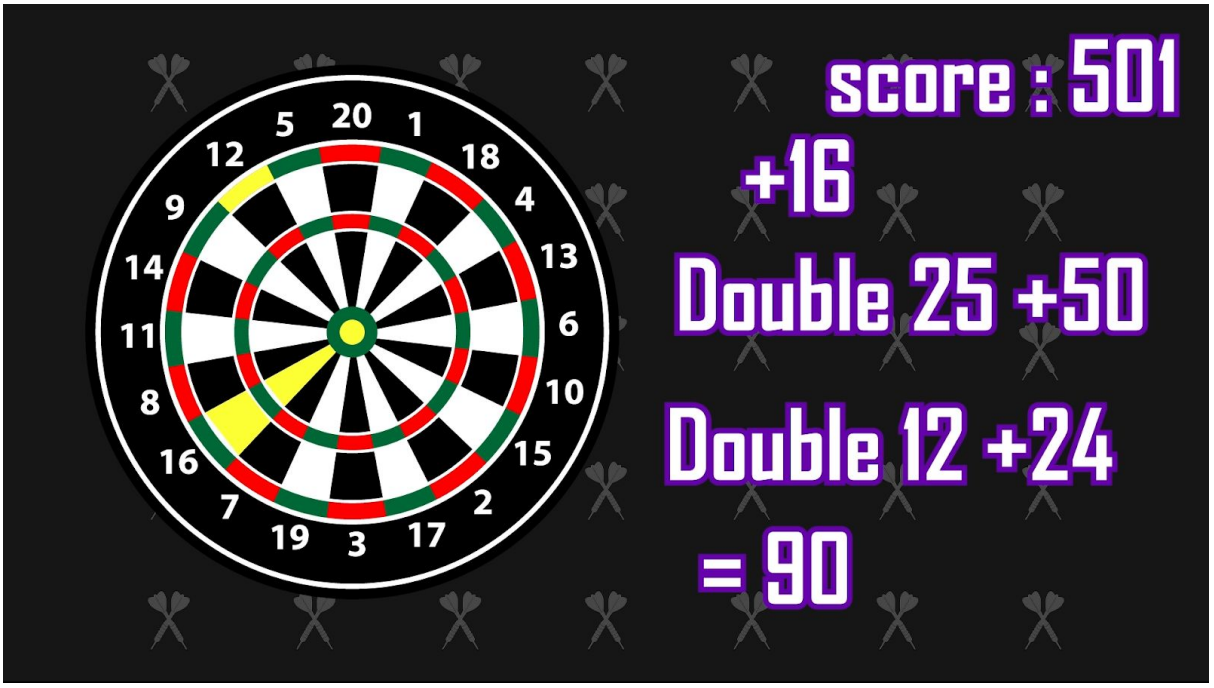


Figure 11: écran de règles

Cette écran affiche les règles du mode de jeu choisie.

Suivi du déroulement de la partie Configurée avec l’appareil mobile :

L'écran de jeu :

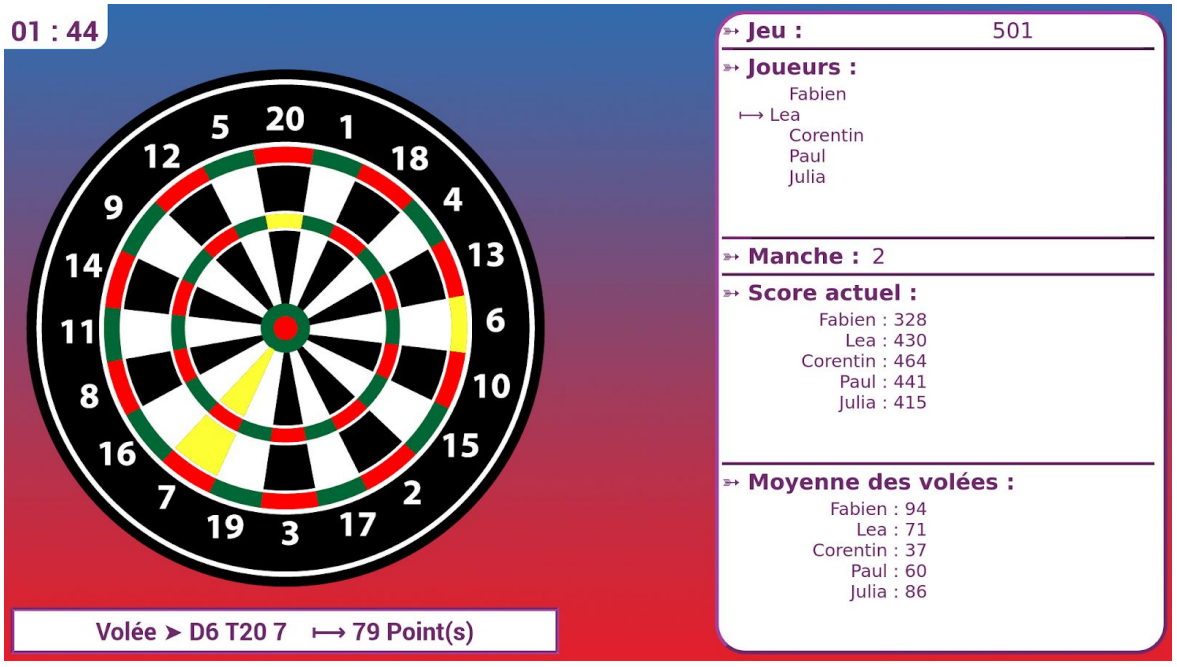


Figure 10: écran de jeu

Dans l'état **EnCours**, le logiciel traite les trames contenant un impact de fléchettes sur la cible :

- **`$DART;GAME;3;20\r\n`**

L'écran de pause :

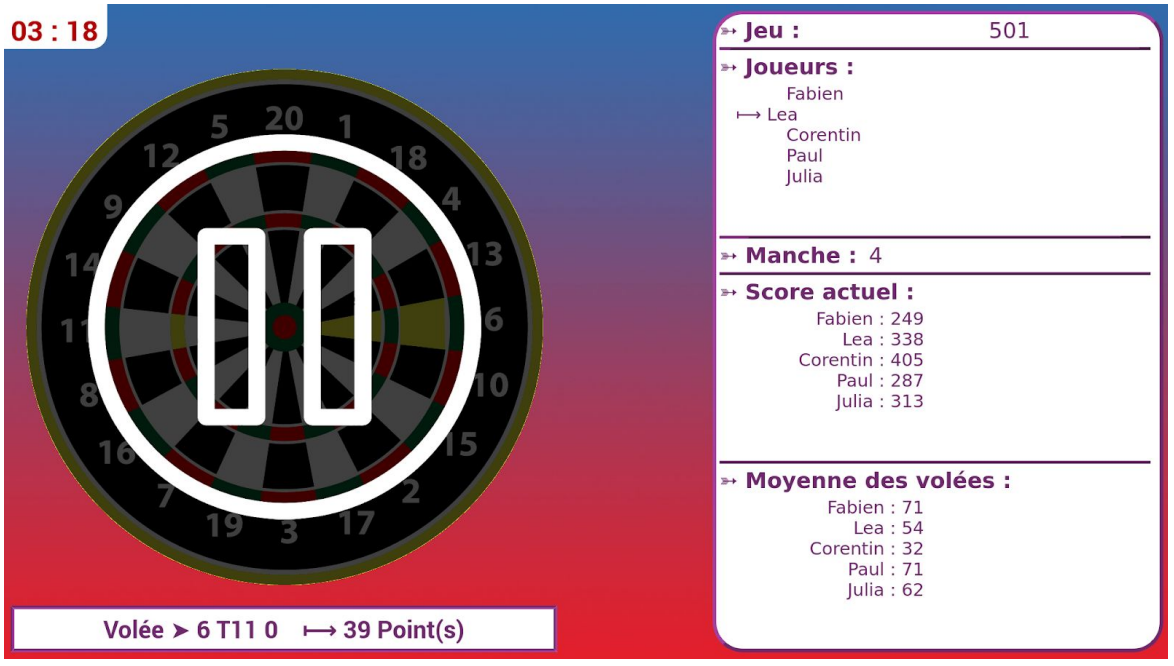


Figure 11: écran de pause

Il est possible de recevoir une trame qui place l'écran en "Pause" :

- **$DART;PAUSE\r\n**

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

La trame "Play" permet de reprendre le déroulement de la partie :

- **$DART;PLAY\r\n**

En utilisant la trame **PLAY**, on refait passer le l'application à l'état **EnCours**.

La réception d'une trame "Stop" termine la partie en cours :

- **$DART;STOP\r\n**

En utilisant la trame **STOP**, on passera de l'état **EnCours** à l'état **fin**.

La réception d'une trame "Regle" affichera et lancera la vidéo explicatif des règles :

- **$DART;Regle\r\n**

En utilisant la trame **REGLE**, On pourra passer de n'importe quel état à l'état **Regle**.

La trame "Reset" réinitialise le jeu :

- **$DART;RESET\r\n**

Avec la trame **RESET**, On pourra passer de n'importe quel état à l'état **Attente**.

L’écran de fin :

Quand un joueur arrive à zéro point la partie est finie, cela a pour effet de basculer l’affichage sur l'écran de fin, montrant le gagnant, les moyennes des volées, le nombre de volées, la volée maximale atteinte, et la durée de la partie.

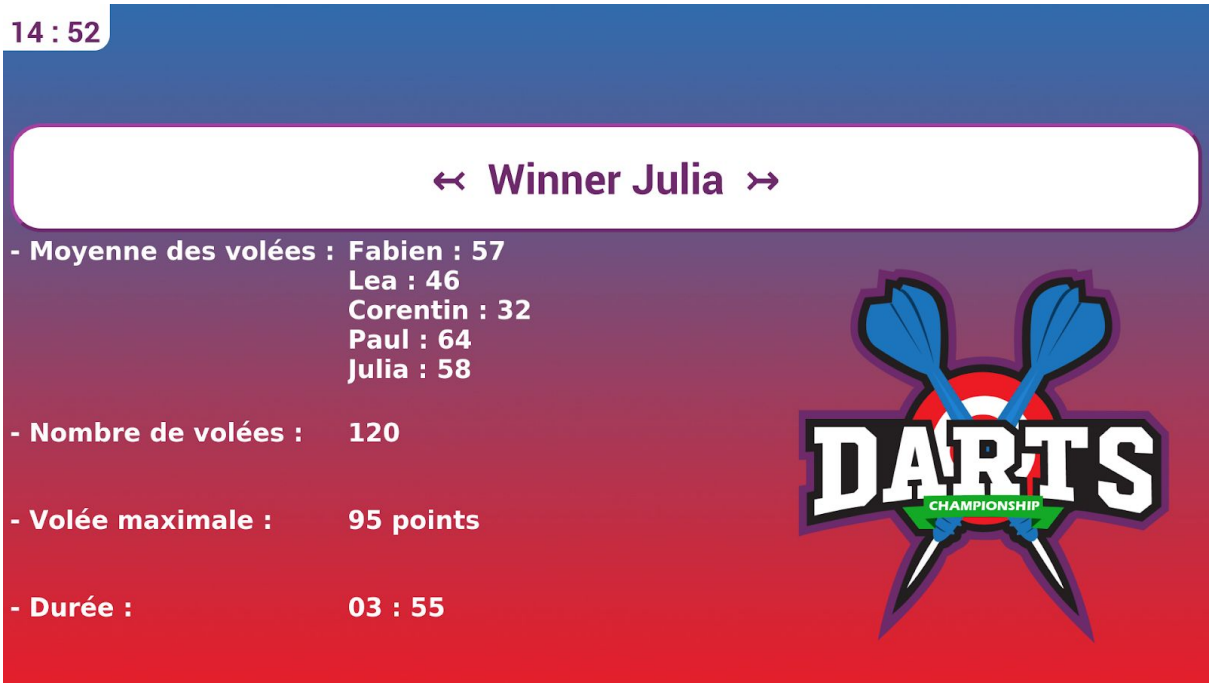


Figure 12: écran fin de partie

Depuis cet écran, il est possible de recevoir une trame **START** (ou **RESET**) pour démarrer une nouvelle partie.

- Mode Tournois

Pour initialiser le tournois il suffit d'envoyer une trame config :
$DART;TOURNOIS;CONFIG;501;Tournois Test;4;Fabien;Thierry;Erwan;Julia\r\n

Pour démarrer le tournois il faut pour cela envoyer une trame play :
$DART;TOURNOIS;PLAY\r\n

Lors de la réception d'une trame play l'affichage bascule sur l'écran de jeux :

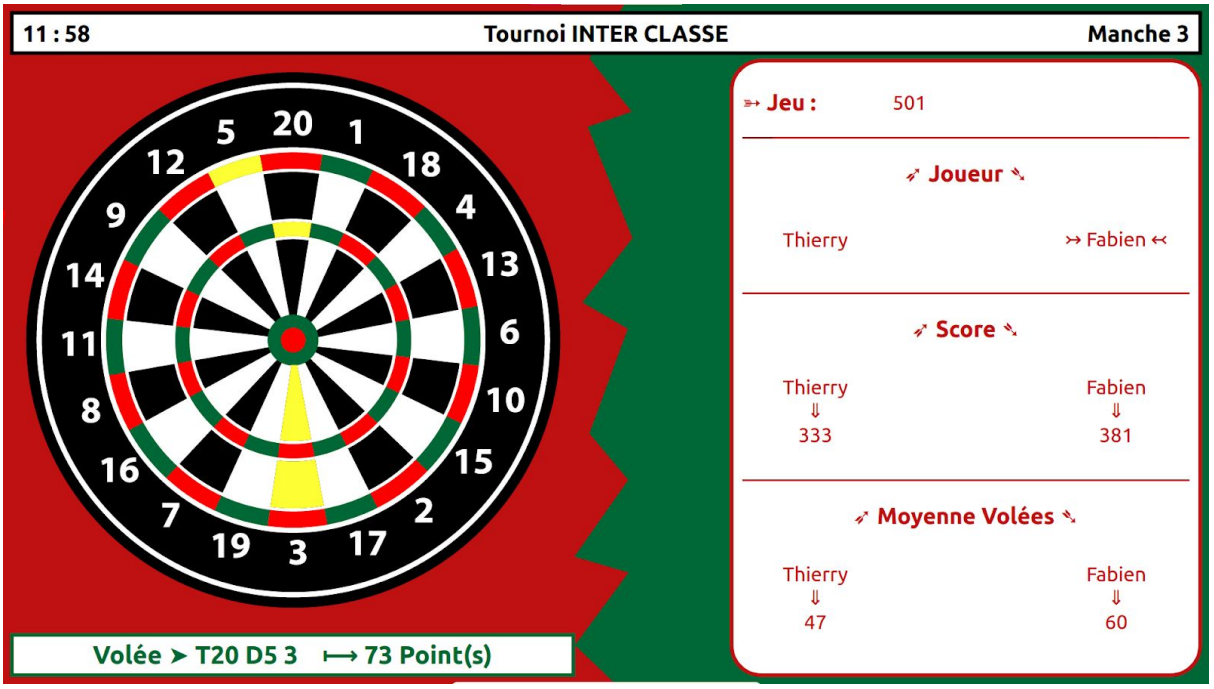


Figure 13: écran Tournois

Lorsque le duel et fini un récapitulatif de la partie est affiché :

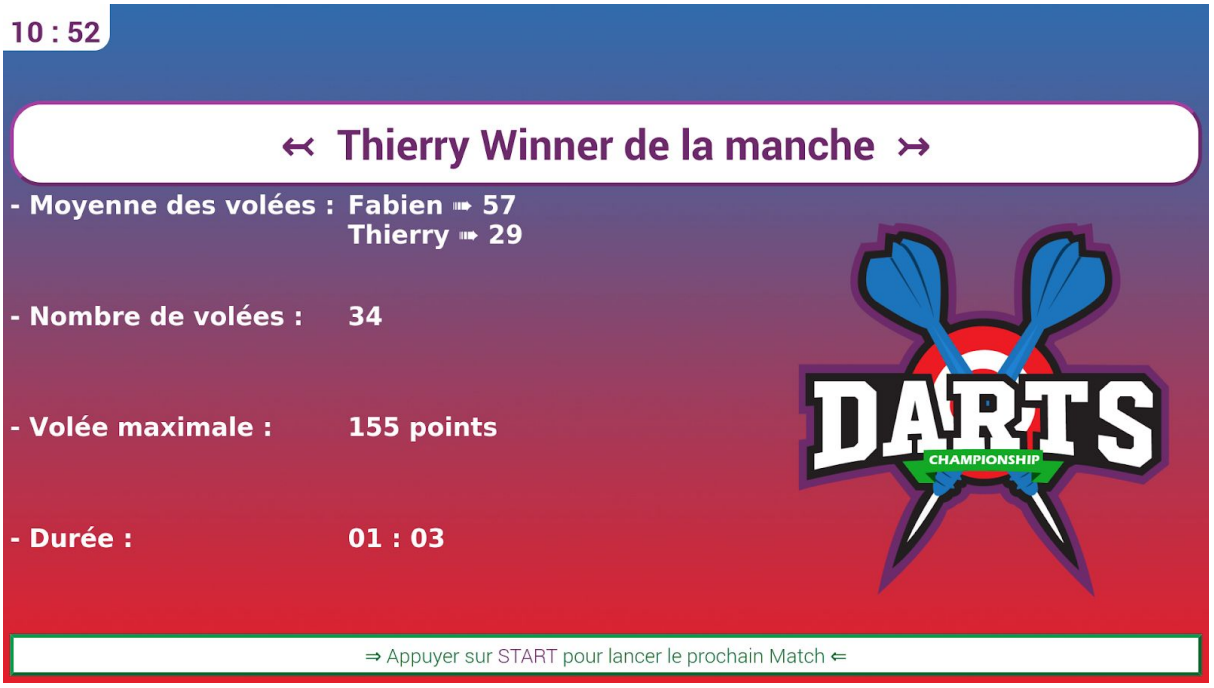
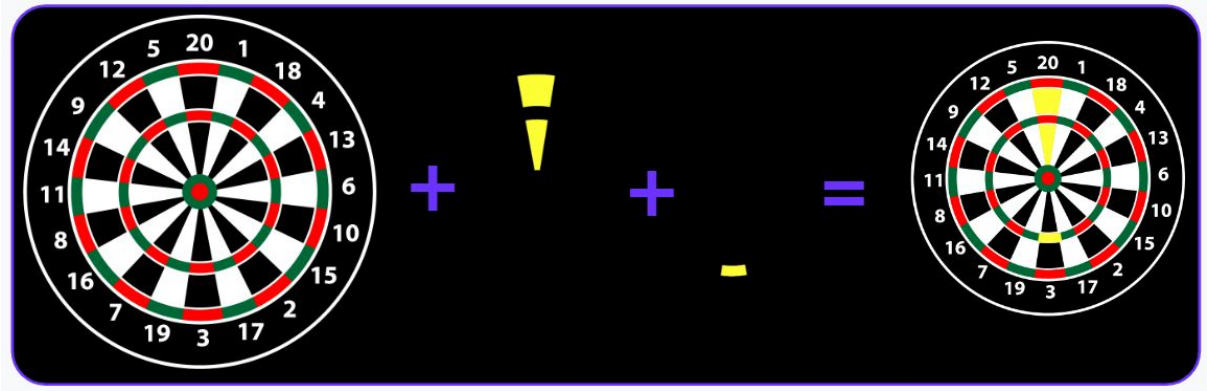


Figure 14: écran de fin

Pour lancer le duel suivant il suffit de renvoyer une trame Play:

`$DART;TOURNOIS;PLAY\r\n`

- Gestion de l'affichage de la cible



Pour l'affichage des impact j'ai opter pour un empilage d'image.
Grâce à la transparence des Image on peut utiliser un **QPainter** qui permet l'empilage des images.

```
if(QFileInfo(qApp->applicationDirPath() + "/impact/IMPACT_" + QString::number(typePoint)
+ "_" + QString::number(point) + ".png").exists())
    {
        QImage impact(qApp->applicationDirPath() + "/impact/IMPACT_" +
QString::number(typePoint) + "_" + QString::number(point) + ".png");

        QPixmap cibleImpacte = ui->labelVisualisationimpact->pixmap()->copy(); // on
récupère l'image précédente;

        QPainter p(&cibleImpacte);

        p.drawImage(QPoint(0, 0), impact);

        p.end();

        ui->labelVisualisationimpact->setPixmap(cibleImpacte);
    }
```

Tout d'abord on commence par vérifier que l'impact que l'on souhaite ajouter existe.
Pour pouvoir empiler les image on récupérer l'image déjà présente (ici la cible) puis grâce au **QPainter** on vient dessiner par-dessus l'impact reçu par la cible.

- Diagramme de classes

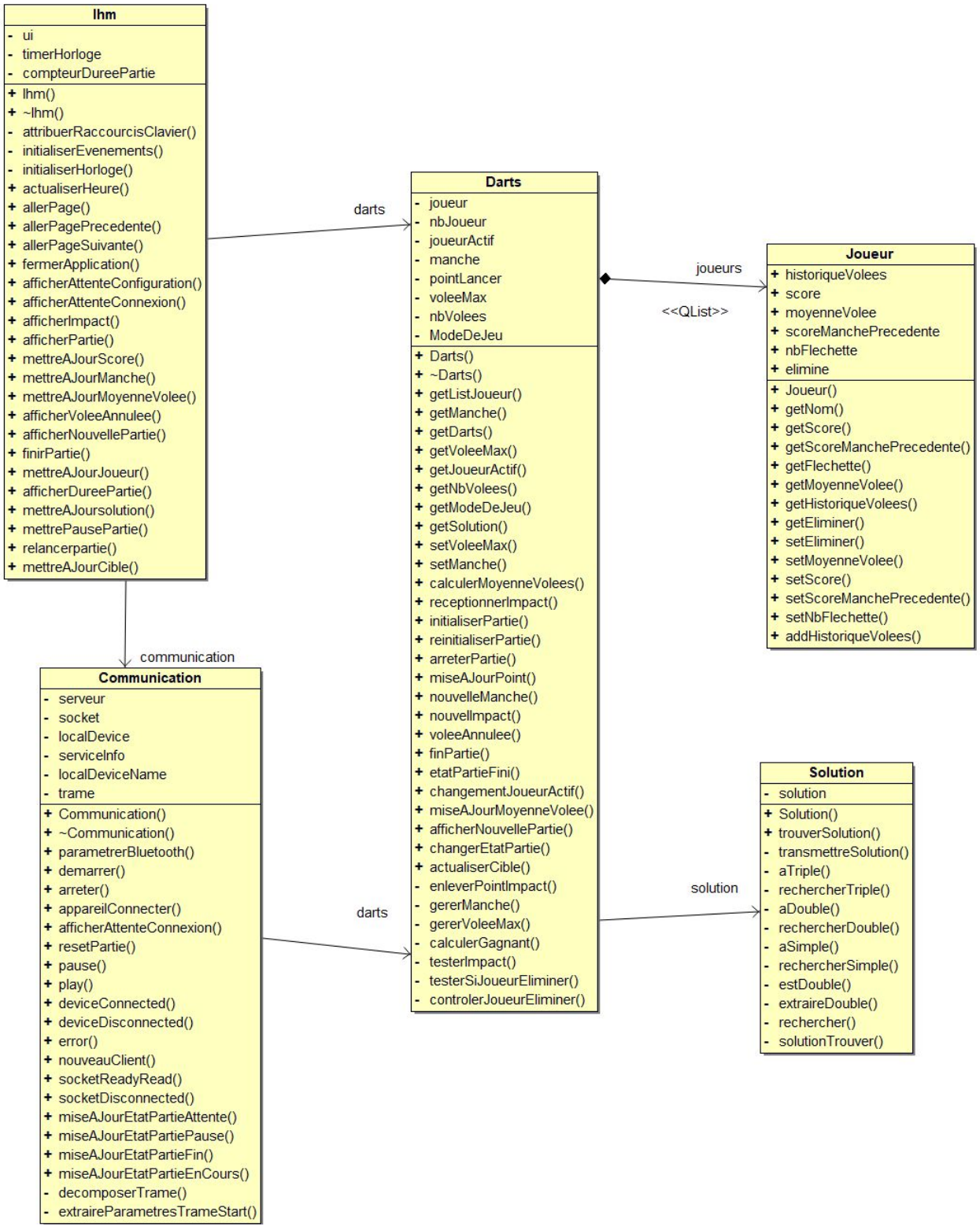


Figure 15: toutes les classes de l'application "écran-DARTS"

La classe principale est la classe **Darts**, elle s'occupe s'occupe de la gestion de la partie (lancer partie, gérer manche, gérer moyenne des joueurs, ...).
La classe **Ihm** est la classe qui s'occupe de la partie affichage de la partie. Les différents écran sont gérés par un **QStackedWidget** qui fournit une pile de QWidget.
La classe **Communication** gérer la communication Bluetooth avec le terminal mobile, elle assure la réception et le décodage des trames.
La classe **Solution** recherche les meilleurs solution pour finir la partie avec un double (DOUBLE OUT).

- Scénario démarrage partie

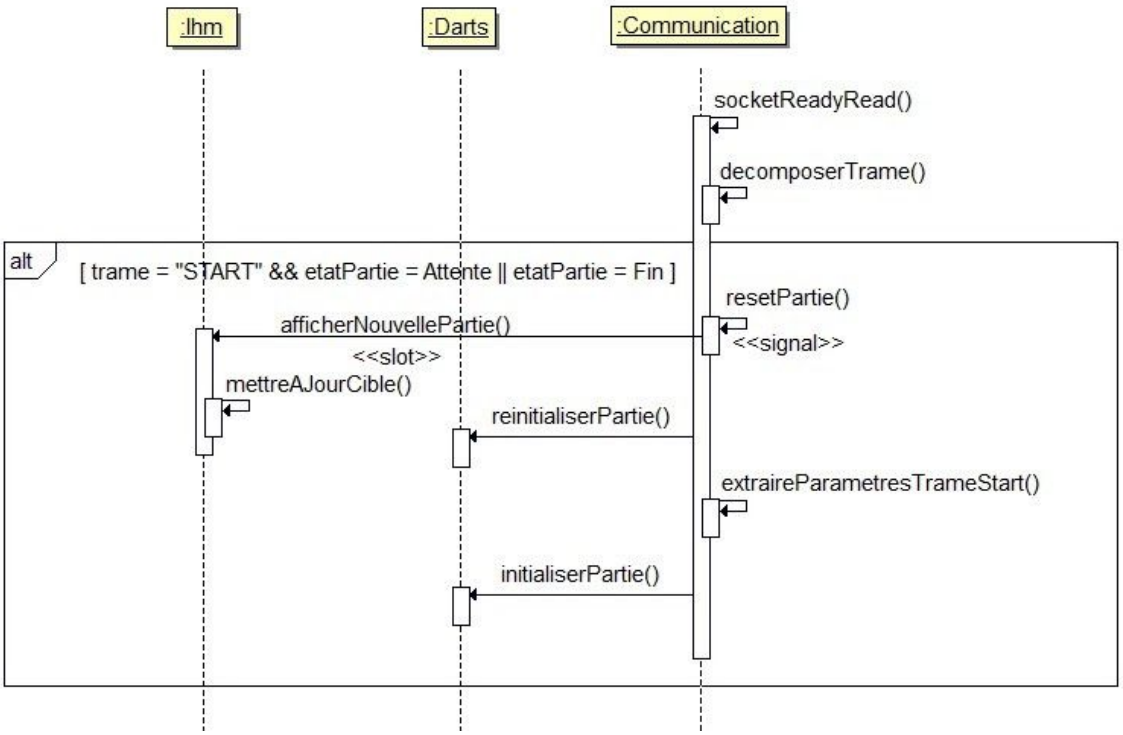


Figure 16: démarrage partie de fléchette

On reçoit une trame START qui contient le type de jeu et le nombre de joueur et les nom des joueurs (si définie). Cela permet ensuite d'envoyer un signal à l'ihm pour mettre à jour les informations de l'ihm, ensuite la classe dart réinitialise les information, pour par la suite initialiser la partie.

- Scénario d'impact fléchette sur la cible

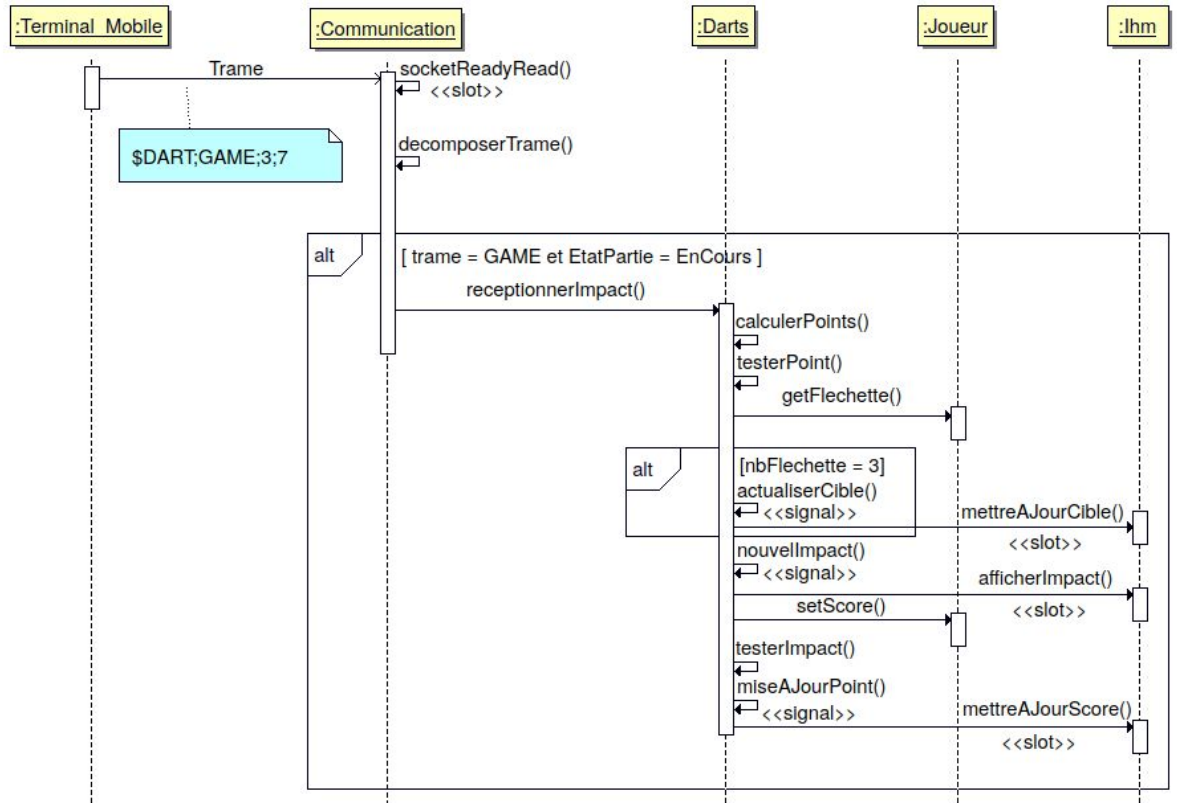


Figure 17: Impact fléchette sur la cible

On reçoit une trame GAME qui contient le type de point et les point de l'impact. Cela permet de calculer le score généré par la fléchette lancée. elle extrait les données et appelle la méthode **receptionnerImpact()** en lui passant le type de point et les point. celle-ci de calculer les points, puis met à jour le score du joueur si il ne tombe pas en dessous de zéro point, par la suite cela va provoquer l'actualisation des différentes informations dans l'IHM. L'IHM affiche les impacts sur une cible, les scores des différents joueurs, la manche, et la moyenne des volées.

- Scénario trame STOP

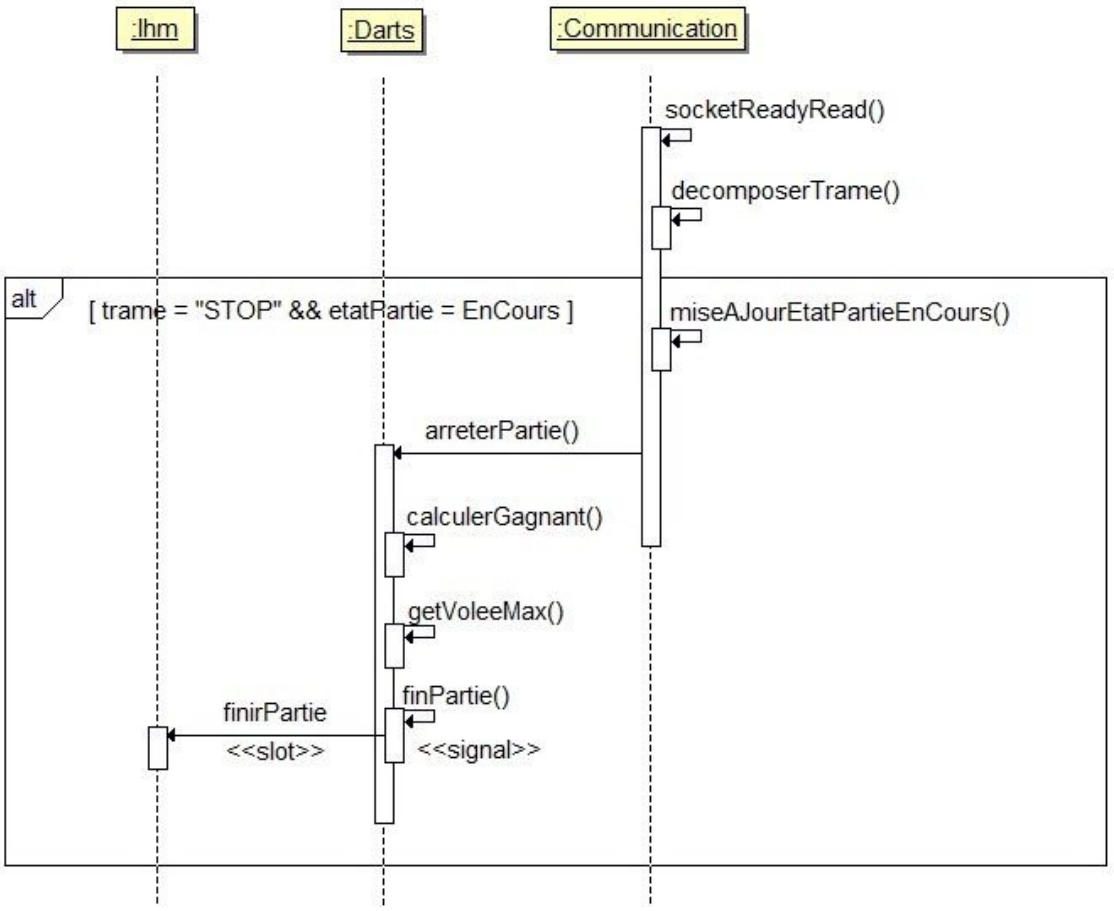


Figure 18: Reception trame STOP

On reçoit une trame STOP qui permet de stopper une partie en cours.
cela à pour effet d'arrêter la partie,puis de calculer la personne ayant le moin de point, de calculer la volées Max et d'émettre un signal à l'ihm pour mettre à jour l'affichage.

- Décodage des trames

```
/**
* @brief Méthode qui decompose la trame reçue et signale l'état en fonction du type de trame
*
* @fn Communication::decomposerTrame
*/
void Communication::decomposerTrame()
{
   if(estValide()) //test si la trame est valide
   {
       trame.remove(DELIMITEUR_FIN);
       if(trame.contains("START") && (etatPartie == EtatPartie::Attente || etatPartie ==
EtatPartie::Fin))      /** $DART;START;501;1;2;fabien;erwan */
       {
           emit resetPartie();
           darts->reinitialiserPartie();

           QString modeJeu;
           QStringList joueurs;

           extraireParametresTrameStart(joueurs, modeJeu);
       }
       else if(trame.contains("GAME") && etatPartie == EtatPartie::EnCours)      /**
$DART;GAME;3;7 */
       {
           darts->receptionnerImpact(trame.section(";",2,2).toInt(),
trame.section(";",3,3).toInt());
       }
       else if(trame.contains("GAME") && etatPartie == EtatPartie::Tournois)      /**
$DART;GAME;3;7 */
       {
           darts->receptionnerImpactTournois(trame.section(";",2,2).toInt(),
trame.section(";",3,3).toInt());
       }
       else if(trame.contains("REGLE")&& etatPartie != EtatPartie::Regle)   /** $DART;REGLE
*/
       {
           extraireParametresTrameRegle();
       }
       else if(trame.contains("PAUSE") && (etatPartie == EtatPartie::EnCours || etatPartie
== EtatPartie::Tournois))     /** $DART;PAUSE */
       {
           etatPrecedent = etatPartie;
           emit pause();
           miseAJourEtatPartiePause();
       }
```

```
        else if(trame.contains("PLAY") && etatPartie == EtatPartie::Pause)   /** $DART;PLAY
*/
        {
            emit play();
            relancerPartie();
        }
        else if(trame.contains("SON"))
        {
            emit jouerSon(trame.section(";",2,2));
        }
        else if(trame.contains("TOURNOIS"))
        {
            decomposerTrameTournois();
        }
        else if(trame.contains("RESET")) // quelque soit l'état de la partie    /**
$DART;RESET */
        {
            reamorcerPartie();
        }
        else if(trame.contains("STOP") && (etatPartie == EtatPartie::EnCours || etatPartie ==
EtatPartie::Pause))  /** $DART;STOP */
        {
            darts->arreterPartie();
        }
        else if(trame.contains("STOP") && (etatPartie == EtatPartie::Regle))  /** $DART;STOP
*/ //permet l'arret des regles en cours de lecture
        {
            emit stopperRegle();
        }
        else
        {
            qDebug() << Q_FUNC_INFO << "Trame non Traité: " << trame;
        }
    }
}
```

Cette méthode teste si la trame reçue et validée pour cela elle vérifie si l'entête du protocole et le délimiteur de fin. Les trames valides seront ensuite testées pour savoir à quelle action appartient la trame.

- Test de validation

Description	oui	non
la liaison bluetooth est opérationnel	✔	
la réception de trame est opérationnel	✔	
la visualisation des impact est identifiée et afficher en temps réel.	✔	
Les données de la partie(nombre de Volées, Volée Max, ...)	✔	
Statistique visible en fin de partie	✔	
Mode Tournois opérationnel	✔	

Annexe

Protocole Trame

1 . Caractéristiques générales

Protocole orienté ASCII
Délimiteur de début de trame : '**$**'
Nom du protocole : **DART**
Délimiteur de fin de trame : "**\r\n**"
Délimiteur des champs d'une trame : '**;**'
Format général d'une trame : **$DART;TYPE;DONNEES\r\n**
Types de trame :

- **START** : initialise une partie
- **GAME** : informe d'un impact sur la cible
- **PAUSE** : met en pause la partie
- **PLAY**: relance la partie qui était en pause
- **STOP** : arrête une partie
- **REGLE** : affiche la vidéo de règle en fonction du mode de jeu.
- **RESET** : réinitialise une partie

Les données dépendant de chaque type de trame.

2 . Mobile vers Écran

a . Trame Initialiser Partie : → démarrer une partie
Cette trame permet de choisir le type de partie, le nombre de joueurs et éventuellement leurs noms

Format : $DART;START;TYPE_PARTIE;AFFICHER_REGLE;NB_JOUEURS[;NOMS]\r\n

Les types de partie sont :

501_DOUBLE_OUT : Chaque joueur débute avec un capital de 501 points. Pour arriver à zéro on doit impérativement lancer sa dernière fléchette de la manche dans un secteur « double » ou dans le double centre (terme *double out* du nom de la règle).

301_DOUBLE_OUT : La règle du 301 est la même que celle du 501 à la différence que les joueurs débutent la partie avec un score de 301 au lieu de 501. Afin de terminer la partie, les joueurs doivent également lancer leur dernière flèche dans un double.

501 : Idem **501_DOUBLE_OUT** à la différence que les joueurs ne sont pas obligés de terminer par un double.

301 : Idem **301_DOUBLE_OUT** à la différence que les joueurs ne sont pas obligés de terminer par un double.

Exemples :

$DART;START;501;1;4;Fabien;Erwan;Thierry;Arthur\r\n → partie 501 avec l'affichage des règle 4 joueurs définis

$DART;START;501;0;4;Fabien;Erwan;Thierry;Arthur\r\n → partie 501 sans l'affichage des règle et 4 joueurs définis

$DART;START;501;1;2\r\n → partie 501 avec l'affichage des règle et 2 joueurs anonymes

$DART;START;501_DOUBLE_OUT;0;2;Fabien;Thierry\r\n → partie 501 double out sans l'affichage des règle et 2 joueurs définis

$DART;START;301;1;2;Fabien;Erwan\r\n → partie 301 avec affichage des règle et 2 joueurs définis

$DART;START;301_DOUBLE_OUT;1;2;Fabien;Thierry\r\n → partie 301 double out avec affichage des règle et 2 joueurs définis

b . Trame de jeu : → tous les impacts pris en charge

Cette trame indique l'impact d'une fléchette sur la cible.

Format : **$DART;GAME;TYPE_POINT;NUMERO_CIBLE\r\n**

TYPE_POINT:

- 1 → Simple point
- 2 → Double points
- 3 → Triple points
- 0 → Zéro point

NUMERO_CIBLE:

La cible est numérotée de 1 à 20, plus la demi-bulle à 25.

Exemples :
$DART;GAME;2;25\r\n → Fléchette *bullseye* (50 points)
$DART;GAME;1;25\r\n → Fléchette demi-bulle (25 points)

Les simples :
$DART;GAME;1;1\r\n → 1 point
$DART;GAME;1;2\r\n
$DART;GAME;1;3\r\n

$DART;GAME;1;4\r\n
$DART;GAME;1;5\r\n
$DART;GAME;1;6\r\n
$DART;GAME;1;7\r\n
$DART;GAME;1;8\r\n
$DART;GAME;1;9\r\n
$DART;GAME;1;10\r\n
$DART;GAME;1;11\r\n
$DART;GAME;1;12\r\n
$DART;GAME;1;13\r\n
$DART;GAME;1;14\r\n
$DART;GAME;1;15\r\n
$DART;GAME;1;16\r\n
$DART;GAME;1;17\r\n
$DART;GAME;1;18\r\n
$DART;GAME;1;19\r\n
$DART;GAME;1;20\r\n → 20 points

Les doubles :
$DART;GAME;2;1\r\n → 2 points
$DART;GAME;2;2\r\n
$DART;GAME;2;3\r\n
$DART;GAME;2;4\r\n
$DART;GAME;2;5\r\n
$DART;GAME;2;6\r\n
$DART;GAME;2;7\r\n
$DART;GAME;2;8\r\n
$DART;GAME;2;9\r\n
$DART;GAME;2;10\r\n
$DART;GAME;2;11\r\n
$DART;GAME;2;12\r\n
$DART;GAME;2;13\r\n
$DART;GAME;2;14\r\n
$DART;GAME;2;15\r\n
$DART;GAME;2;16\r\n
$DART;GAME;2;17\r\n
$DART;GAME;2;18\r\n
$DART;GAME;2;19\r\n
$DART;GAME;2 ;20\r\n → 40 points

Les triples :
$DART;GAME;3;1\r\n → 3 points
$DART;GAME;3;2\r\n
$DART;GAME;3;3\r\n
$DART;GAME;3;4\r\n
$DART;GAME;3;5\r\n
$DART;GAME;3;6\r\n

$DART;GAME;3;7\r\n
$DART;GAME;3;8\r\n
$DART;GAME;3;9\r\n
$DART;GAME;3;10\r\n
$DART;GAME;3;11\r\n
$DART;GAME;3;12\r\n
$DART;GAME;3;13\r\n
$DART;GAME;3;14\r\n
$DART;GAME;3;15\r\n
$DART;GAME;3;16\r\n
$DART;GAME;3;17\r\n
$DART;GAME;2;18\r\n
$DART;GAME;3;19\r\n
$DART;GAME;3;20\r\n → 60 points

$DART;GAME;0;0\r\n → Fléchette en dehors de la cible (zéro point)

c . Trame Reset :

$DART;RESET\r\n → remet à zéro

d . Trame Pause:

$DART;PAUSE\r\n → met en pause la partie

e . Trame Play:

$DART;PLAY\r\n → relance la partie qui était en pause

f . Trame Stop:

$DART;STOP\r\n → arrête la partie

g. Trame Regle:

$DART;REGLE;TYPE\r\n → trame qui affiche les regles du mode de jeu choisie
$DART;REGLE\r\n → trame qui affiche les regles du jeu en cours.

h . Trame Tournois

- **Initialisation**

$DART;TOURNOIS;CONFIG;501;Tournois Test;4;Fabien;Thierry;Erwan;Julia\r\n

- **Démarrer tournois**

$DART;TOURNOIS;PLAY\r\n

Les types de partie sont : **501_DOUBLE_OUT, 301_DOUBLE_OUT, 501, 301, …**

Convention de Nommage

La documentation avec **Doxygen** pour les attributs: //!< ….
Exemple: QString ModeDeJeu; //!< contient le mode de jeu en cours

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 retourne le mode de jeu
*
* @fn Darts::getModeDeJeu
* @return le mode de jeu
*/
QString Darts::getModeDeJeu()
{
    return ModeDeJeu;
}
```

Glossaire

DARTS : fléchettes

UML : unified Modeling Language

IHM : Interface Homme Machine

Terminologie

Raspberry Pi : c'est un nano-ordinateur monocarte à processeur ARM conçu par des professeurs du département informatique de l'université de Cambridge

Mode kiosque : Le mode kiosque consiste à dédier l'affichage graphique à l'usage exclusif d'une application, et de plus sans faire apparaître les habituels éléments du bureau graphique (panel, icones,...).
Parmi les exemples les plus couramment rencontrés : les distributeurs de billets, les affichages d'horaires dans les (aéro)gares,...

Video

Presentation mode de jeu

DOUBLE OUT

SANS DOUBLE OUT

Description Ecran-DARTS