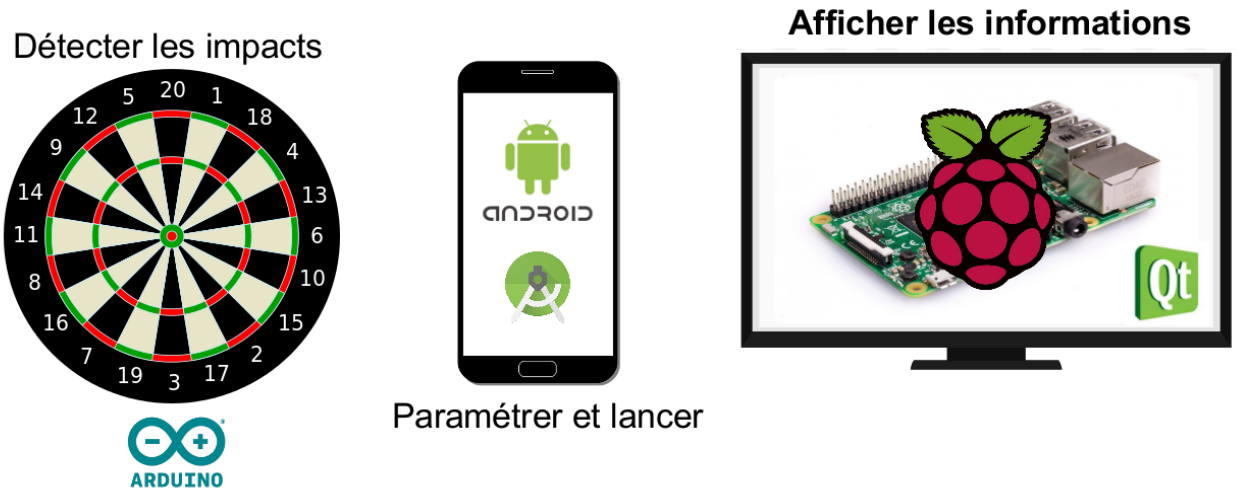
Dossier Technique

Projet DARTS 2021

Table des Matières

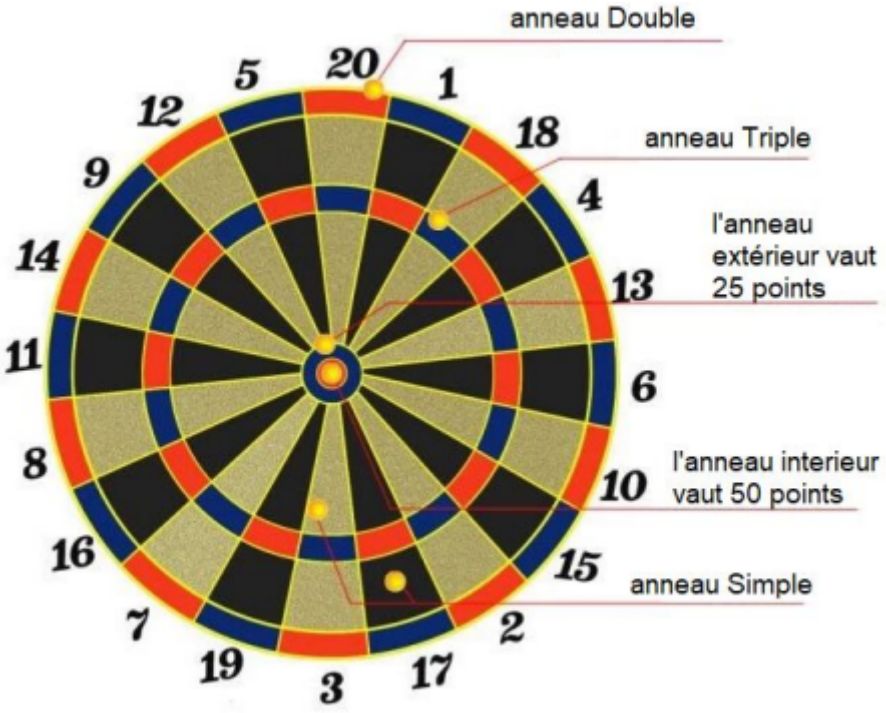
Présentation générale du projet

Le projet DARTS est un système numérique permettant de jouer à plusieurs jeux de fléchettes différents, ces jeux sont régis par les règles officielles qui sont adoptées dans tous les tournois.
Le système permet de jouer à plusieurs jeux différents : le 301, 301 double out, le 501 et le 501 double out.



- Zone d’impact de la cible

Les différentes “zone” de la cible sont :



Expression du Besoin

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 manches, point restant dans la manche, moyenne des volées, …) sur un écran de télévision.

Equipe du projet DARTS

Option IR :
Etudiant 1 : LEFORT Théodore
Etudiant 2 : HUGON Vincent

Option EC :
Etudiant 1 : COULLOMB Adrien

Répartition des Tâches (IR)

Etudiant 1 : LEFORT Théodore

Module de gestion de partie (Mobile-DARTS)

Sur le terminal mobile Android, l'application doit permettre de paramétrer et démarrer une partie.
Pour cela, les joueurs pourront :

- saisir leur nom
- paramétrer la partie :
    - le type de jeu : 501 double out, 301 double out, ...
    - le nombre de joueurs,
    - le nombre de manches gagnantes,
- lancer la partie
- gérer et visualiser le déroulement de la partie :
    - changer automatiquement de joueur
    - affichage du nombre de points
    - renseigner le module Cible d’une fléchette hors cible

Etudiant 2 : HUGON Vincent

Module de visualisation de partie (Écran-DARTS)

Ce module correspond à la partie “affichage” du système. Il a pour objectifs de réaliser la récupération d’informations envoyées par le terminal mobile, le calcul et l’affichage des statistiques pour la partie actuelle. Il communique en Bluetooth uniquement avec le terminal mobile Android.
L’ihm sera décomposé en trois écrans.

Un écran d'accueil.
Un écran de partie.
Un écran de fin.
Sur l'écran de partie, les joueurs pourront visualiser en continu :

- le nom des joueurs
- le type de jeu en cours et le score

Les données visualisées sont donc :

- Le type de jeu
- Le score de la partie en cours

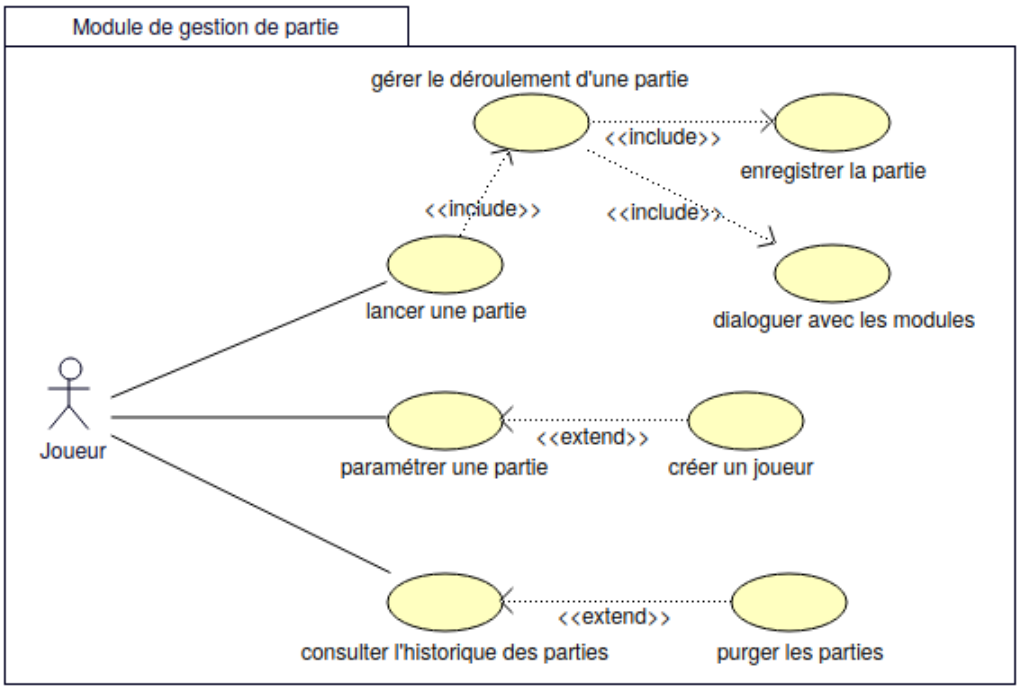
Partie LEFORT Théodore (IR)

● 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 gestion de partie

L'objectif est de pouvoir paramétrer et démarrer une partie, gérer les informations reçues par le module cible et les transférer au module écran.

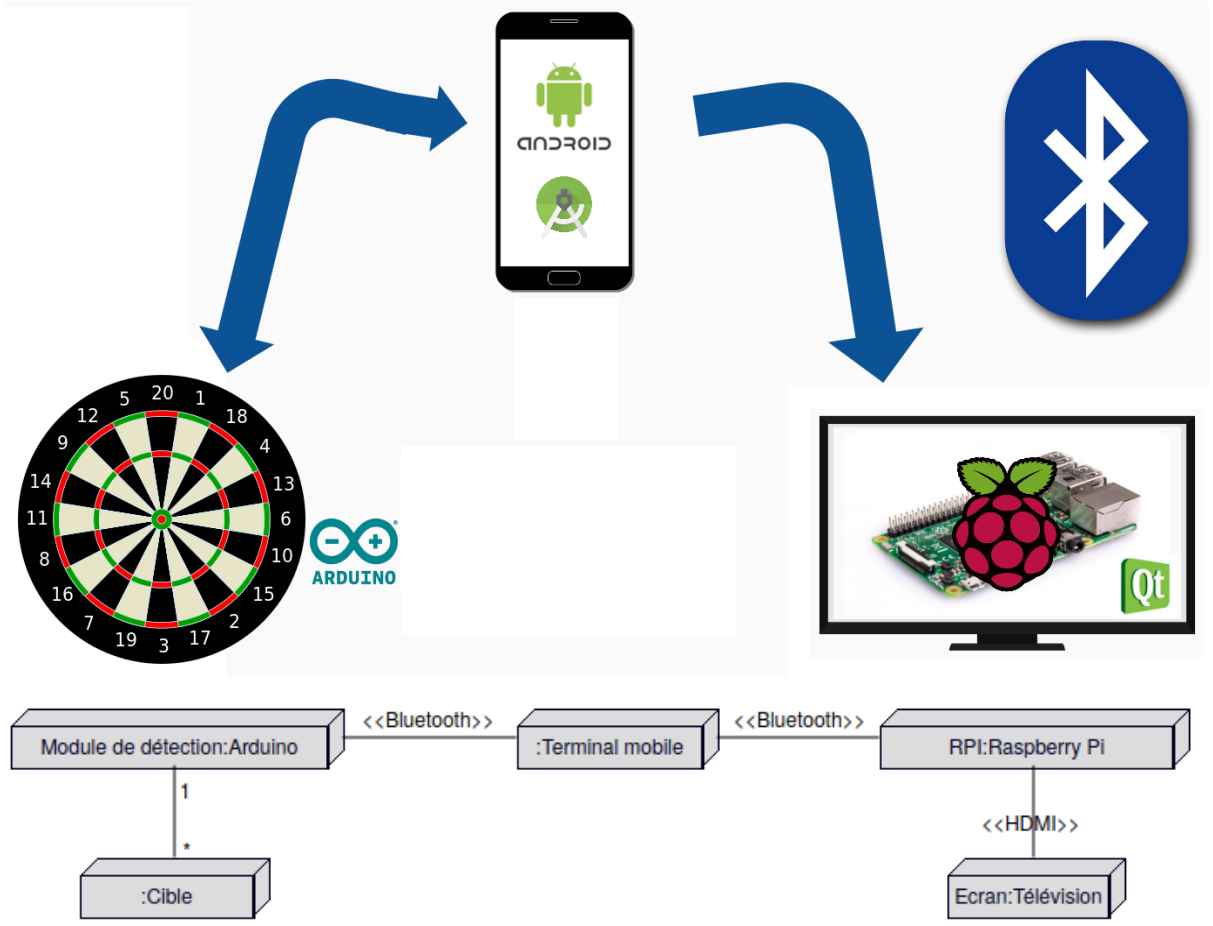


Le module de gestion de partie doit pouvoir :

- Lancer une partie de fléchettes, la gérer, dialoguer avec les modules, et l'enregistrer
- Paramétrer une partie avec possibilité de créer un joueur
- Consulter l'historique des parties avec la possibilité de purger celles-ci

● Description structurelle du système

Afin de pouvoir échanger, traiter et afficher toutes les informations essentielles pour une partie, il est nécessaire de communiquer en bluetooth entre les modules du système DARTS.



Le système DARTS est donc décomposé en trois modules :

● Le module de gestion de partie (Mobile-DARTS) : l'application sur un terminal mobile(sous Android) permet de configurer et lancer une partie

● Le module de détection des impacts (Cible-DARTS) : un système électronique permettant d'identifier la zone impactée par les fléchettes envoyées par les joueurs et d'envoyer ces informations au module de gestion de partie

● Le module de visualisation de partie (Écran-DARTS) : l'application réalisée en Qt sur une Raspberry Pi 4 reliée à un écran de télévision permettant de visualiser en temps réel le déroulement de la partie.

● Tâches à réaliser

- Installation de l'environnement de développement
- Mise en oeuvre de la tablette android et de la liaison sans fil
- Réalisation du dialogue avec les modules, création de l'IHM et développement du code source de l'application

● Organisation commune du projet

Afin d'avoir une meilleur organisation, communication et gestion des fichiers et code source, 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 et 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 permettant de toujours avoir une sauvegarde de notre code source en cas d'incident matériel et aussi d'avoir une historique complète du code source.

● Outils de développement

Désignation	Caractéristique	Version
Tablette android	Galaxy Tab S2	Android version 7.0
Planification	BeesBusy	-
Journal de bord	Google Drive	-
Diagrammes UML	Bouml	Version 7.11
Gestionnaire de Version	RiouxSVN (Subversion)	Version 1.13.0
Documentation du code	Doxygen	Version 1.8.17
Langage utilisé	Java	Java Version 11.0.11

● La communication Bluetooth

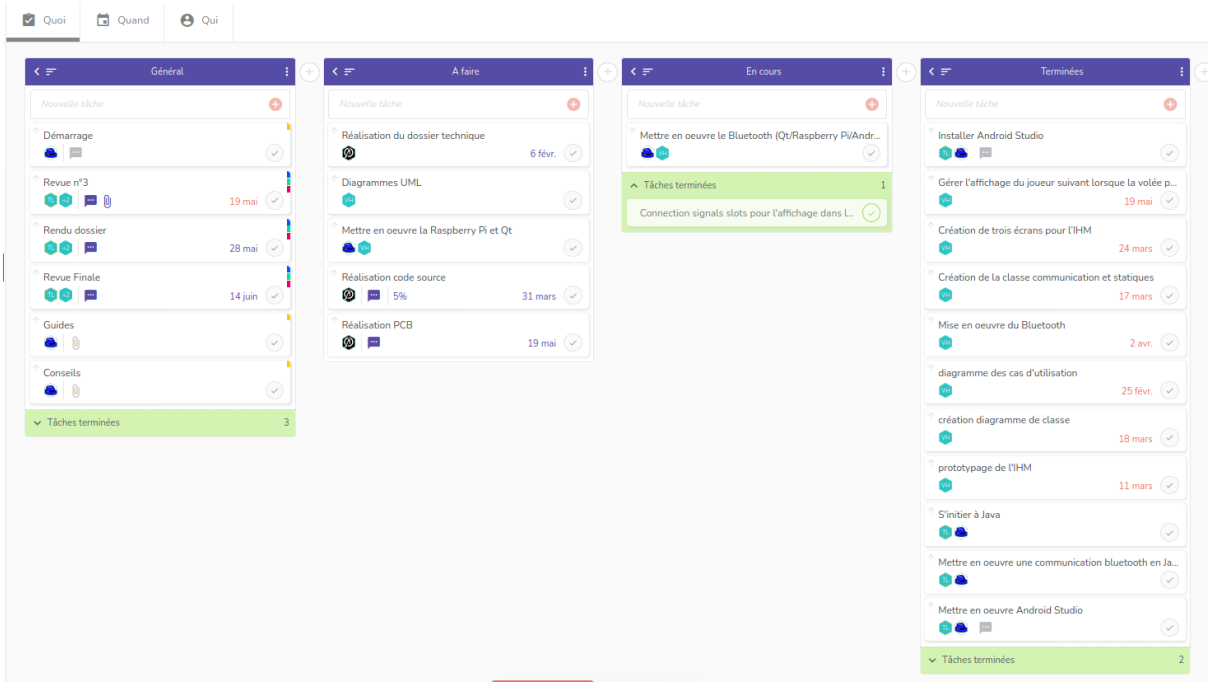
Le bluetooth est une norme de communications permettant l'échange bidirectionnel de données à courte distance en utilisant des ondes radio UHF sur une bande de fréquence de 2.4 GHz dont l'exploitation ne nécessite pas de licence vu la faible puissance d'émission. Le bluetooth établit donc une connexion sécurisée de proximité.
Il est utilisé pour simplifier les connexions entre les appareils électroniques (ici les modules 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 plusieurs composants de communiquer entre eux.
La tablette android est équipée de bluetooth 4.0, toutes les communication se feront alors dans cette version

Caractéristiques du bluetooth 4.0 :
- Débit de 3 Mbit/s
- Portée de 60 m
- Fréquence d'émission de la porteuse 2.4 GHz
- Modulation PSK

• Planification des Tâches

La planification des tâches à été effectuée sur BeesBusy.

Cette planification nous a permis de pouvoir suivre l'évolution de chacun des modules du projet DARTS.



● Les différentes trames échangées

Trame	Description
$DARTS;START	Cette trame permet d'envoyer les paramètres saisis sur le module de gestion de partie aux autres modules du système DARTS.
$DARTS;HIT	Cette trame permet de transférer les informations sur une fléchette qui a été touchée
$DARTS;OK	Cette trame permet d'envoyer un acquittement après une touche
$DARTS;NEXT	Cette trame permet d'indiquer au module de gestion de partie que l'on peut passer au joueur suivant
$DARTS;END	Cette trame permet de déclarer une fin de partie et d'envoyer le gagnant au module écran.

La trame START est envoyée au début d'une partie, lorsque le joueur a paramétré la partie et envoie alors ces informations aux autres modules

La trame HIT est envoyée par le module Cible vers le module gestion de partie, elle transmet les informations de touche d'une fléchette, cette information est ensuite transmise au module écran.

La trame OK est une trame d'acquittement qui est envoyée après le traitement d'une trame.

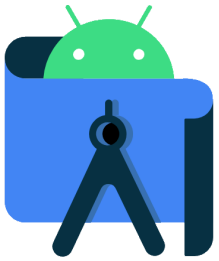
La trame NEXT indique qu'il faut passer au joueur suivant.

La trame END indique la fin d'une partie et transmet le nom du joueur gagnant.

- **Le développement de l’application**

L’application étant faite pour une tablette Android, le développement à été entièrement fait dans le logiciel Android Studio en Java.

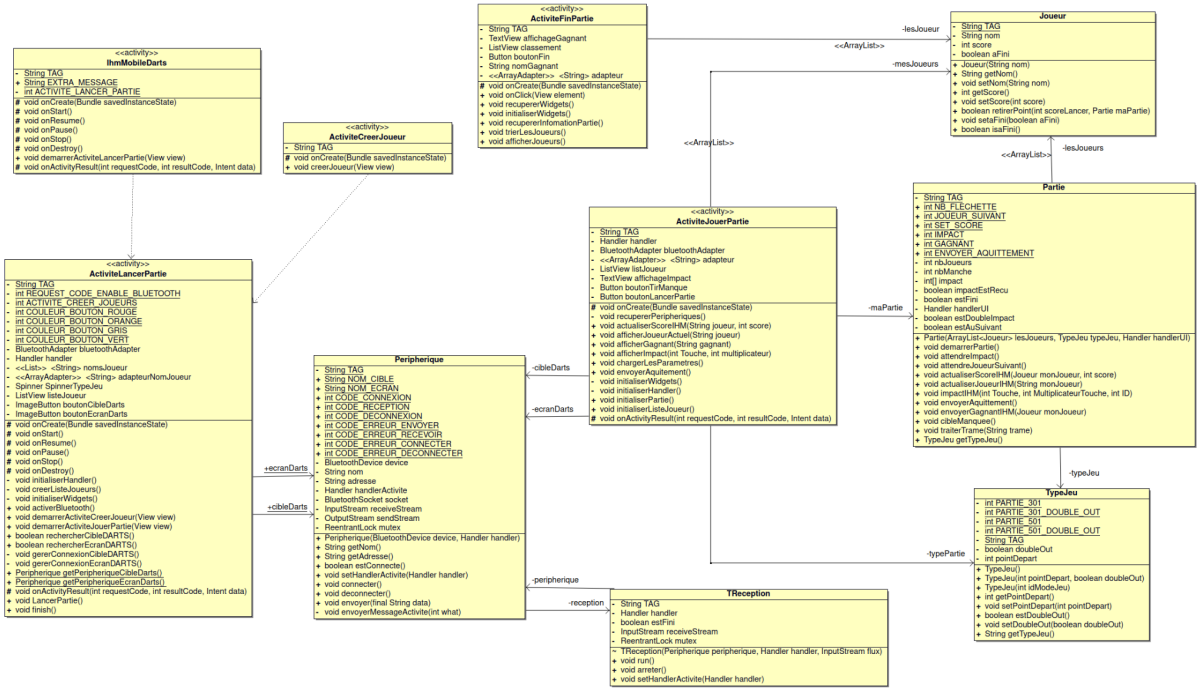
android studio

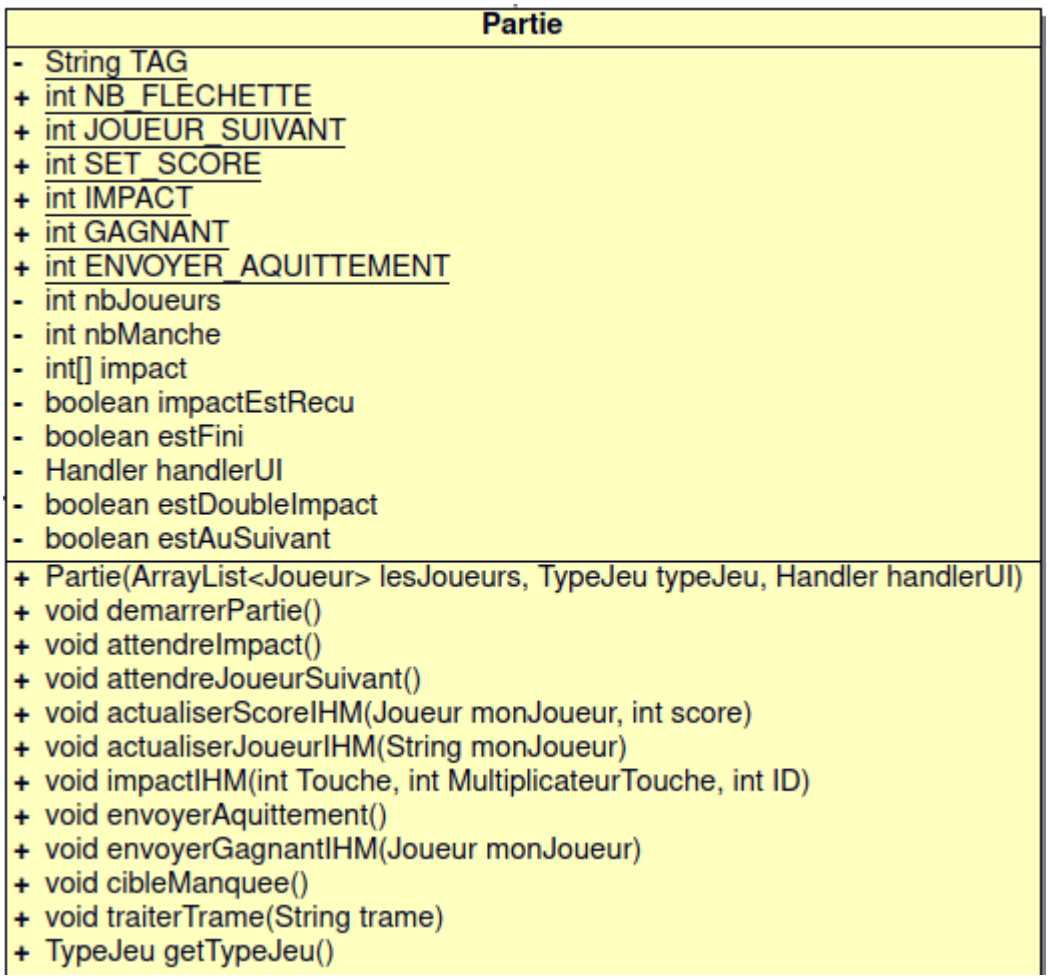


L’utilisation d' android studio nous permettra d’utiliser l’intégralité du SDK Android nous facilitant grandement la tâche pour créer notre application

Diagramme de classes de l’application module DARTS

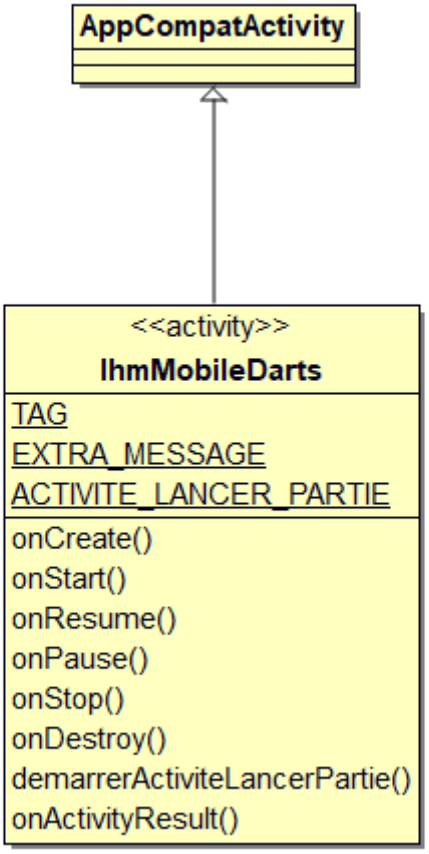
L’application contient deux types de classes différentes, des classes plus classiques que l’on peut retrouver dans la plupart des langages de programmation comme par exemple la classe Partie.





Elle contient des méthodes et des attributs qui permettent de traiter dans notre cas les parties qui sont jouées.

Contrairement à ces classes on retrouve les classes d'activitées qui elles sont unique au développement android, les activités. Une activité (activity) est la composante principale d'une application sous Android. Elle représente le code "métier" de l'application et possède une vue (View) graphique qui est visible par l'utilisateur. Un exemple de ces activités est notre Ihm d'accueil, IhmMobileDarts.



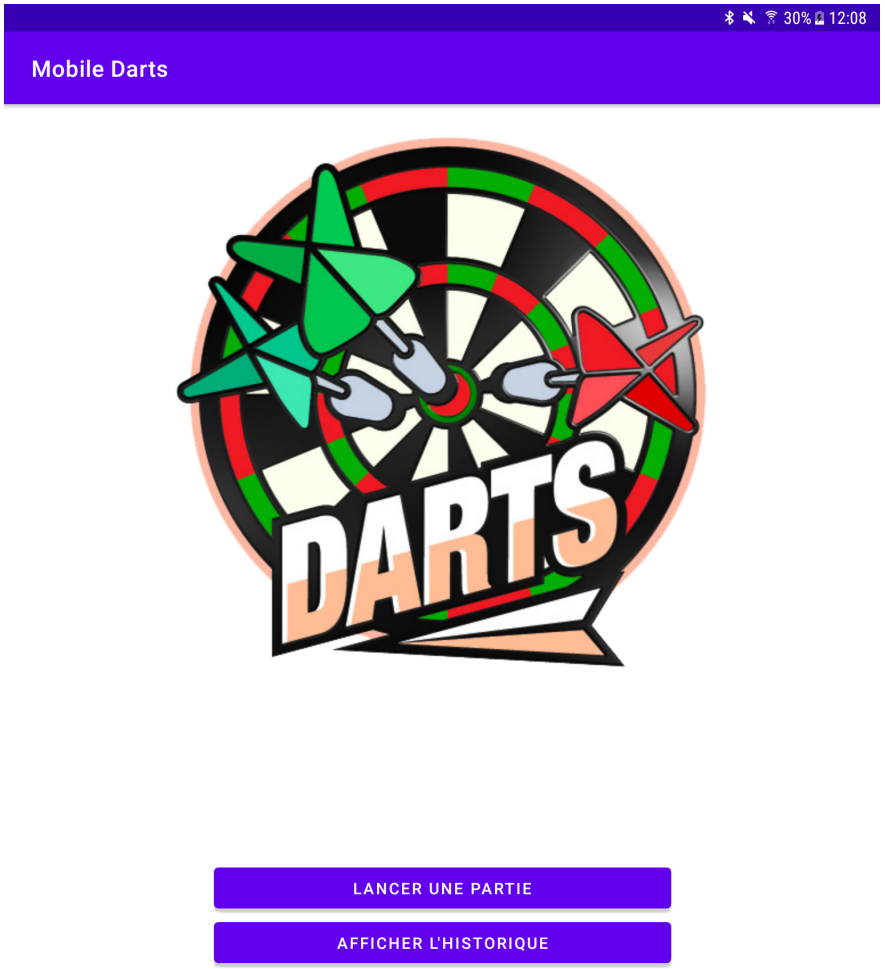
On peut voir que les activitées contiennent des méthodes qui leurs sont propres, celles ci sont actives en fonction de l'état de l'activité, elles ne peuvent avoir que quatre états :

- « Active » : l'activité est lancée par l'utilisateur, elle s'exécute au premier plan quand elle devient active la méthode onCreate() est appelée.
- « En Pause » : l'activité est lancée par l'utilisateur, elle s'exécute et est visible, mais elle n'est plus au premier plan. Une notification ou une autre activité lui a volé le focus et une partie du premier plan la méthode onPause() est alors appelée.
- « Stoppée » : l'activité a été lancée par l'utilisateur, mais n'est plus au premier plan et est invisible. L'activité ne peut interagir avec l'utilisateur qu'avec une notification, la méthode onStop() est alors appelée .
- « Morte » : l'activité n'est pas lancée.

- # Les activitées dans Android

Ces activités héritent toutes de la classe AppCompatActivity qui assure une compatibilité de ces activités au travers de l'écosystème Android.

Ces activités possèdent un fichier appelé layout permettant de leur construire une IHM qui leur est propre et qui est constructible graphiquement ou en programmation classique. Par exemple l'activité IhmMobileDarts ressemble à ceci :

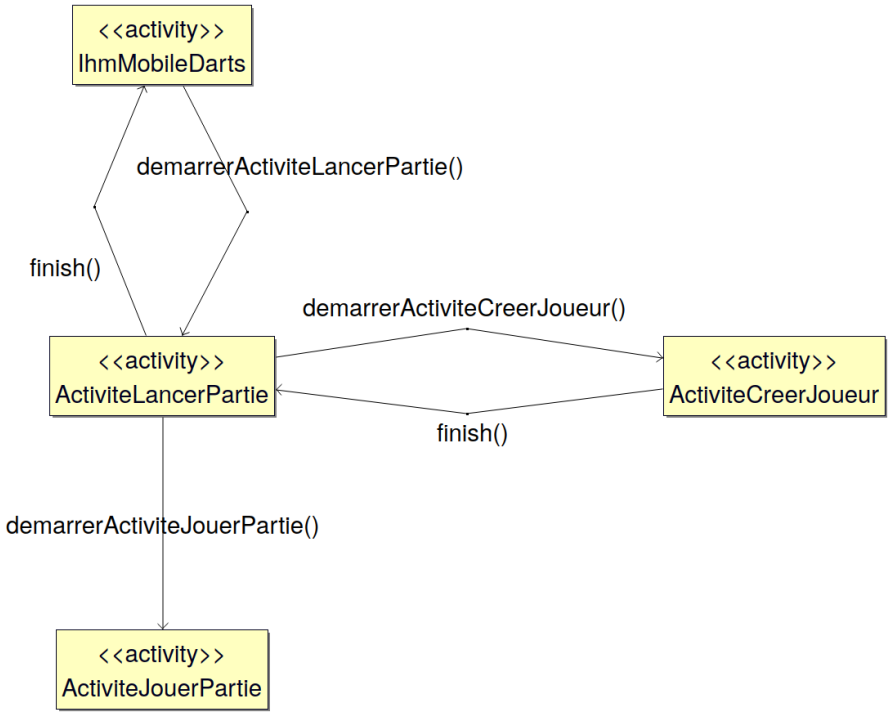


chacun des éléments de cette activité est enregistré dans un fichier layout appelé activity_main.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".IhmMobileDarts">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="401dp"
        android:layout_height="106dp"
        android:layout_marginStart="10dp"
        android:layout_marginLeft="10dp"
        android:layout_marginEnd="10dp"
        android:layout_marginRight="10dp"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.502"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/imageView">

        <Button
            android:id="@+id/boutonLancerPartie"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:onClick="demarrerActiviteLancerPartie"
            android:text="@string/lancer_une_partie" />

        <Button
            android:id="@+id/bouttonAfficherHistorique"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:onClick="demarrerActiviteAfficherHistorique"
            android:text="@string/afficher_l_historique" />

    </LinearLayout>
```

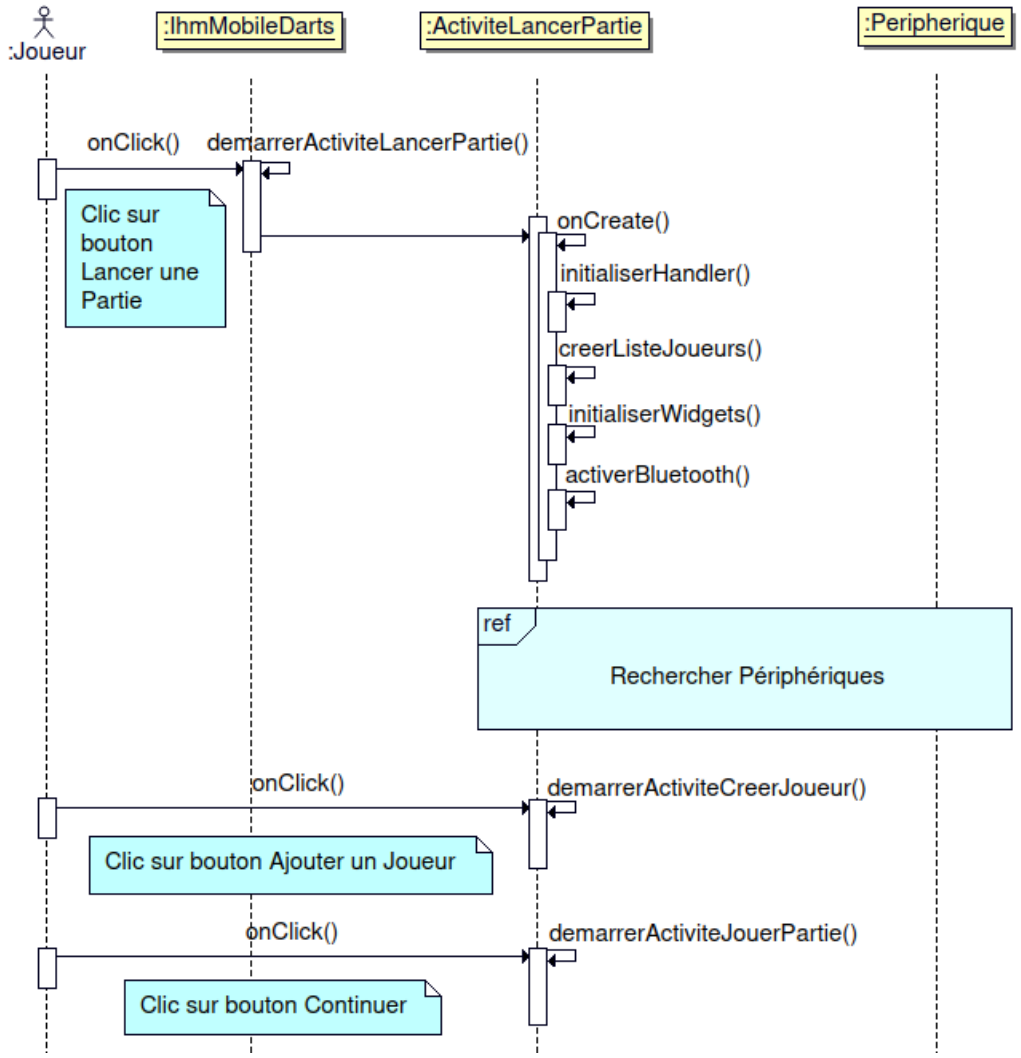
```
   <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

android:contentDescription="@string/sublime_logo_du_projet_darts"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/logo" />


</androidx.constraintlayout.widget.ConstraintLayout>
```

On retrouve notre image appelée ImageView et nos deux boutons.



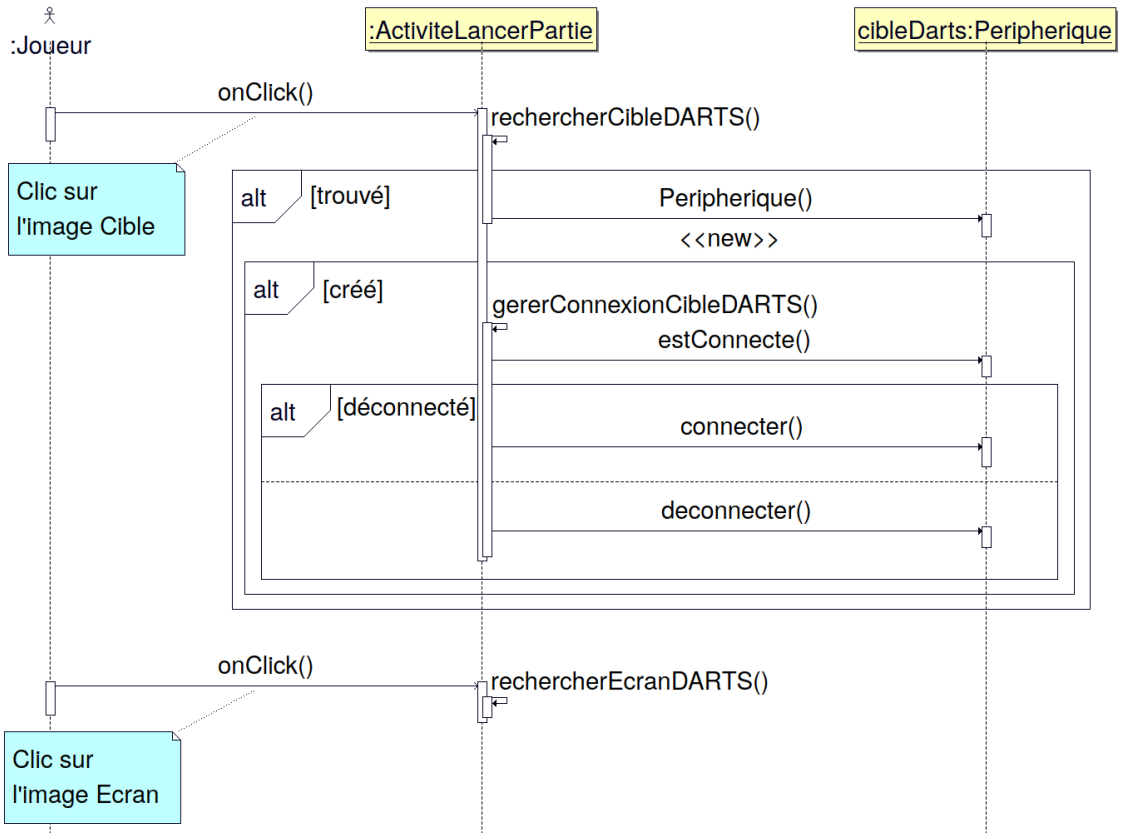
Dans ce diagramme d'état transitions on peut voir que les Activités s'enchaînent pendant l'utilisation et permettent alors l'affichage de pages différentes dans notre application.

- **Scénario d'un clic sur le bouton Lancer une partie**



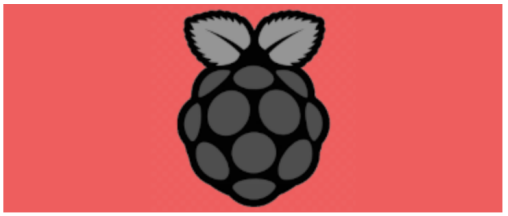
Au moment où le joueur appuie sur le bouton Lancer une partie, une méthode onClick() s'exécute et appelle une autre méthode appelée demarrerActiviteLancerPartie(), celle ci portant bien son nom lance l'activité Lancer Partie, c'est dans cette activité que le joueur va pouvoir configurer la partie, mais surtout se connecter aux modules connectés en bluetooth au lancement de cette activité la méthode OnCreate() est appelée.

```
protected void onCreate(Bundle savedInstanceState)
{
   Log.d(TAG,"onCreate()");
   super.onCreate(savedInstanceState);
   setContentView(R.layout.activity_activite_jouer_partie);

   initialiserHandler();
   initialiserWidgets();
   recupererPeripheriques();
   chargerLesParametres();
   initialiserListeJoueur();
   maPartie = new Partie(mesJoueurs, typePartie, handler);
}
```



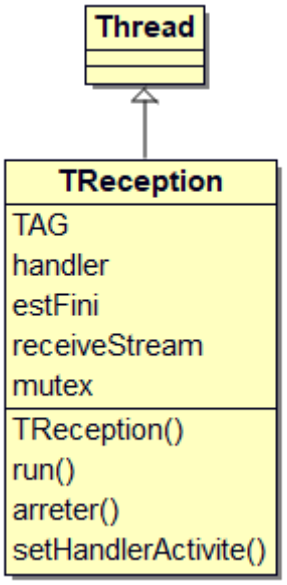
La connexion en bluetooth aux modules se fait par l'utilisation de deux méthodes quasiment identiques qui traitent chacune la connexion avec l'un des modules,

quand l’utilisateur clique sur l’un des boutons Cible ou écran une recherche s'effectue. Si la connexion réussit, la couleur du bouton passe au vert.



- **La réception des trames sous Android**

Android étant un système d'exploitation multitâche, la réception de trames se fait grâce à la classe TReception.



L'utilisation de threads pour la réception et l'envoi de trames permet de ne pas figer les activités visibles quand l'application attend une trame venant de la cible.

Traitement de la trame

La trame est récupérée puis traitée dans la classe Partie :

```
public void traiterTrame(String trame)
{
   Log.d(TAG, "traiterTrame() " + trame);
   trame = trame.replace("\r\n", "");
   String[] trameSectionee = trame.split(";", 5);
   for(int i=0; i < trameSectionee.length; ++i)
   {
       Log.d(TAG, "traiterTrame() trameSectionee[" + i + "] = " +
trameSectionee[i]);
   }

   switch (trameSectionee[1])
   {
       case("HIT"):
```

```
            Log.d(TAG, "traiterTrame : HIT");
            impact[0] = Integer.parseInt(trameSectionee[2]);
            impact[1] = Integer.parseInt(trameSectionee[3]);
            impact[2] = Integer.parseInt(trameSectionee[4]);
            envoyerAquittement();
            impactEstRecu = true;
            break;
        case("NEXT"):
            Log.d(TAG, "traiterTrame : NEXT");
            //estAuSuivant = true;
            break;
    }
}
```

Selon le type de trame, celle-ci est découpée puis les données utiles sont extraites pour l'utilisation dans une partie comme par exemple la trame HIT d'où on extrait les différentes valeurs de l'impact, la touche, le multiplicateur et l'ID de la fléchette dans la volée.

- **Déroulement d'une partie**

```
public void demarrerPartie() {
   Log.d(TAG, "demarrerPartie()");
   estAuSuivant = true;
   do
   {
       nbManche++;
       Iterator<Joueur> it = lesJoueurs.iterator();
       Log.d(TAG, "Manche numéro " + nbManche);
       while (it.hasNext())
       {
           Joueur monJoueur = it.next();
           Log.d(TAG, "C'est le tour de " + monJoueur.getNom());
           attendreJoueurSuivant();
           actualiserJoueurIHM(monJoueur.getNom());
           int pointVollee = 0;

           for(int i = 0; i < NB_FLECHETTE; i++)
           {
               if (monJoueur.isaFini())
               {
                   envoyerGagnantIHM(monJoueur);
                   i = NB_FLECHETTE;
                   estFini = true;
               }
               else
               {
                   impactEstRecu = false;
                   attendreImpact();
                   pointVollee =+ impact[0]*impact[1];
                   if(!monJoueur.retirerPoint(pointVollee, this))
                   {
                       i = NB_FLECHETTE;
                       pointVollee = 0;
                   }
                   else if (monJoueur.getScore() == 0 && !typeJeu.estDoubleOut())
                   {
                       envoyerGagnantIHM(monJoueur);
                       i = NB_FLECHETTE;
                       estFini = true;
                   }
                   else if (monJoueur.getScore() == 0 && typeJeu.estDoubleOut() && estDoubleImpact)
                   {
                       envoyerGagnantIHM(monJoueur);
                       i = NB_FLECHETTE;
                       estFini = true;
                   }
               }
           }
           actualiserScoreIHM(monJoueur, monJoueur.getScore());
       }
   }while (!estFini);
}
```

Cet algorithme contrôle tout le déroulement de la partie et appelle les méthodes qui permettent d'actualiser les données et de déclarer une fin de partie.

● Tests de validation

L’ihm est fonctionnelle	Oui
La création d’un joueur est possible	Oui
La liaison bluetooth avec la cible est opérationnelle	Oui
La liaison bluetooth avec l’écran est opérationnelle	Oui
L’envoi et la réception de trames est opérationnelle	Oui
Il est possible de jouer une partie	Oui
La partie se termine correctement	Oui

Partie HUGON Vincent (IR)

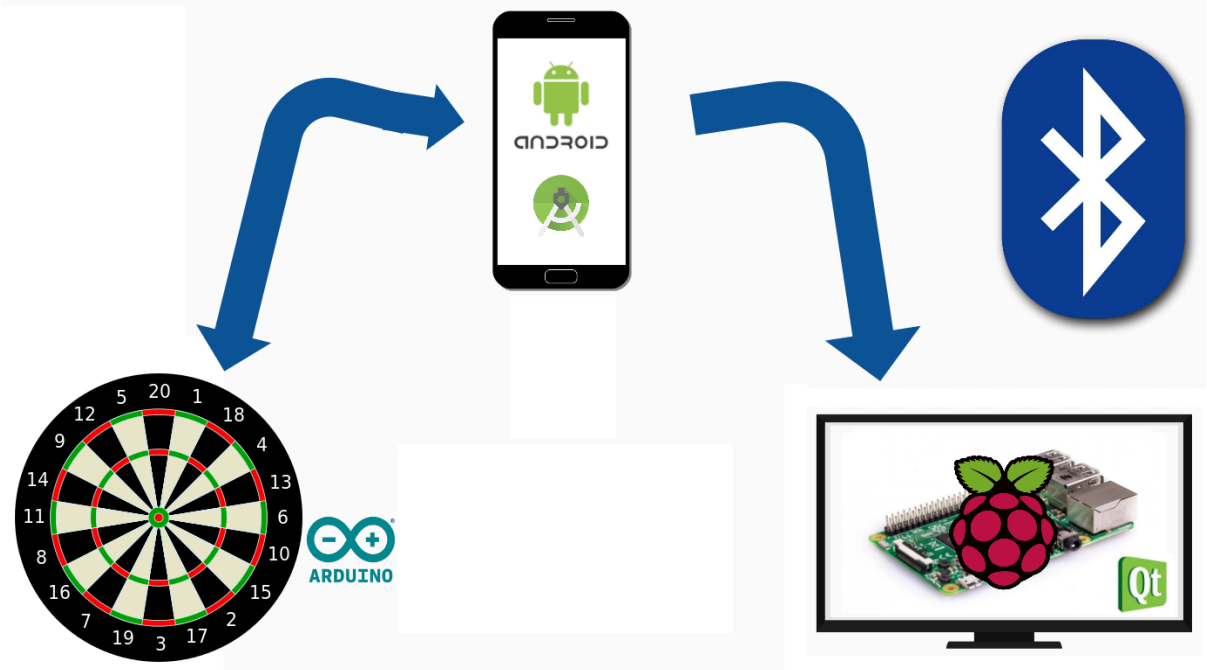
• Rappel du Besoin Initial

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 partie

L'objectif est d'afficher en temps réel le score, le type de jeu et le nom des joueurs sur l'écran.

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



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

● 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.

● Les outils de développement

Description	**Version**
Raspberry Pi 4	GNU/Linux Raspbian version 4.19
Planification	Beesbusy
Diagramme UML	BOUML 7.11
Gestionnaire de Version	RiouxSVN (subversion) 1.13.0
Documentation de code	Doxygen 1.8.17
Langage utilisé	C++ / avec le framework Qt 5.11.3
Journal de bord	Google drive

• Planification

La planification à été faite en trois itérations.
L'itération une à été de créer l'IHM et de mettre en œuvre le bluetooth et la Raspberry Pi.
La deuxième itération à été de communiquer avec le terminal mobile en Bluetooth et d'afficher les données d'une partie (score, type de jeu, nom des joueurs).
La troisième itération a été de gérer une partie, afficher les statistiques et configurer le mode Kiosque.

Pour la planification on utilise aussi beesbusy un site web pour la planification de projet.

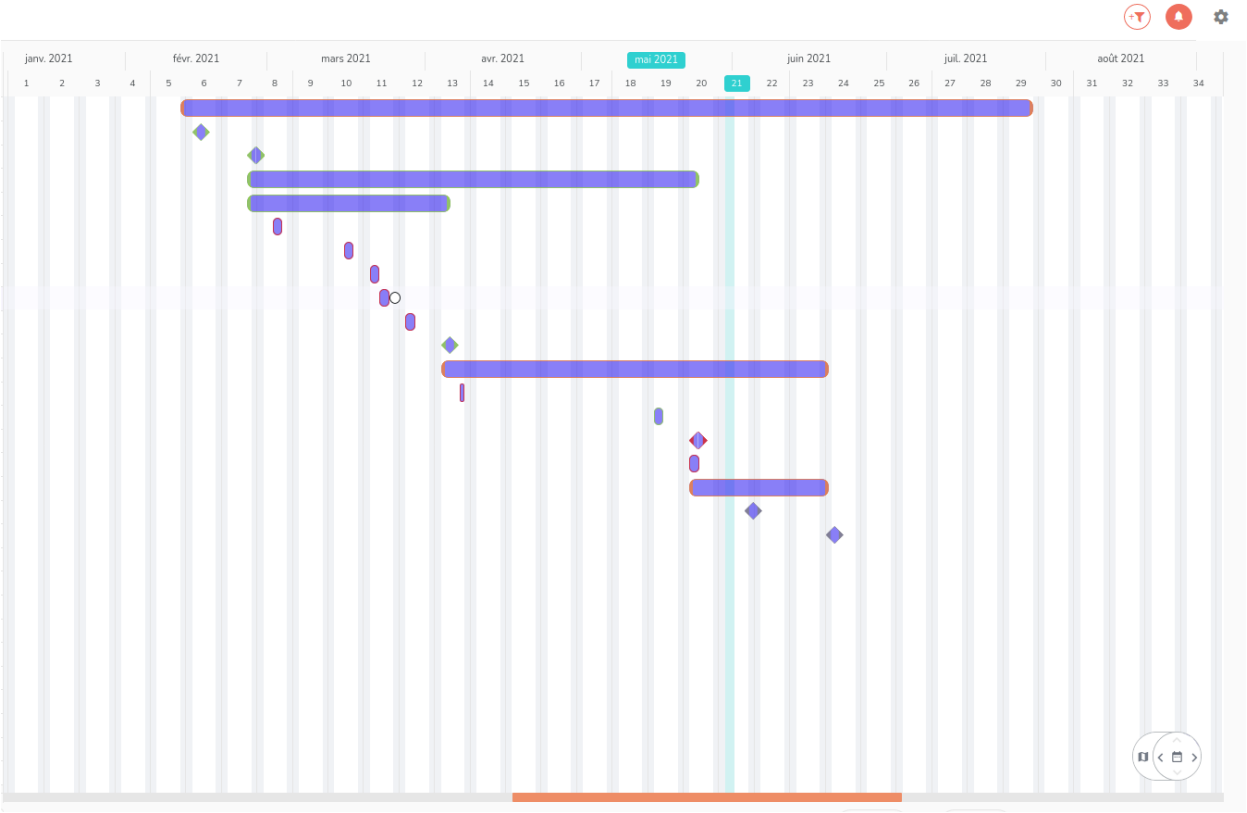


diagramme de gantt DARTS

● Transmission sans fil

Choix de la transmission sans fil : **Bluetooth**

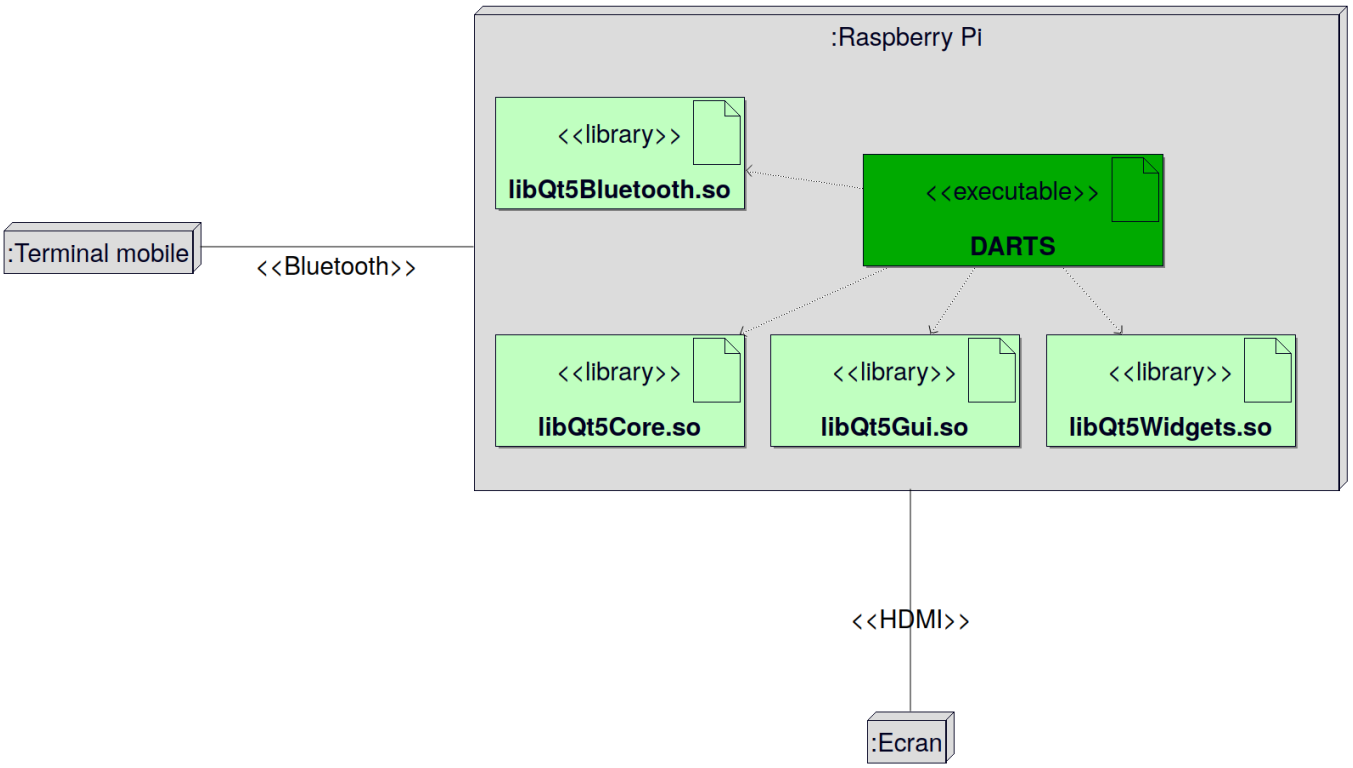
Le bluetooth est une norme de communications permettant l'échange bidirectionnel de données à courte distance en utilisant des ondes radio UHF sur une bande de fréquence de 2.4 GHz

Critères de choix :

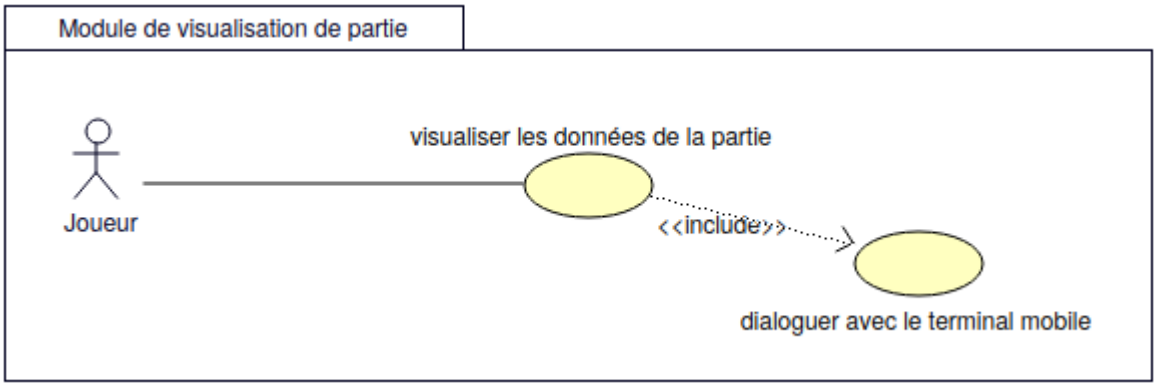
- ➢ Disponible sur la Raspberry Pi et sur le terminal mobile (Android)
- ➢ Permet l'échange bidirectionnel de données
- ➢ Portée suffisante de quelques mètres
- ➢ Débit (3 Mbit/s en 4.X) largement suffisant pour le projet DARTS
- ➢ API disponible sous Qt5

• Diagramme de déploiement

Le module écran darts doit être raccordé à une télévision en HDMI pour afficher les informations sur la partie en cours ainsi que d'être raccordé en bluetooth avec le terminal mobile pour configurer la partie.



● Diagramme des cas d'utilisation



Un ou plusieurs "Joueur" peuvent visualiser tout au long de la partie les différentes données (score, type de jeu et nom des joueurs).

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)
- Le score des différents joueurs
- Le type de jeu

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

Les différentes trames DART

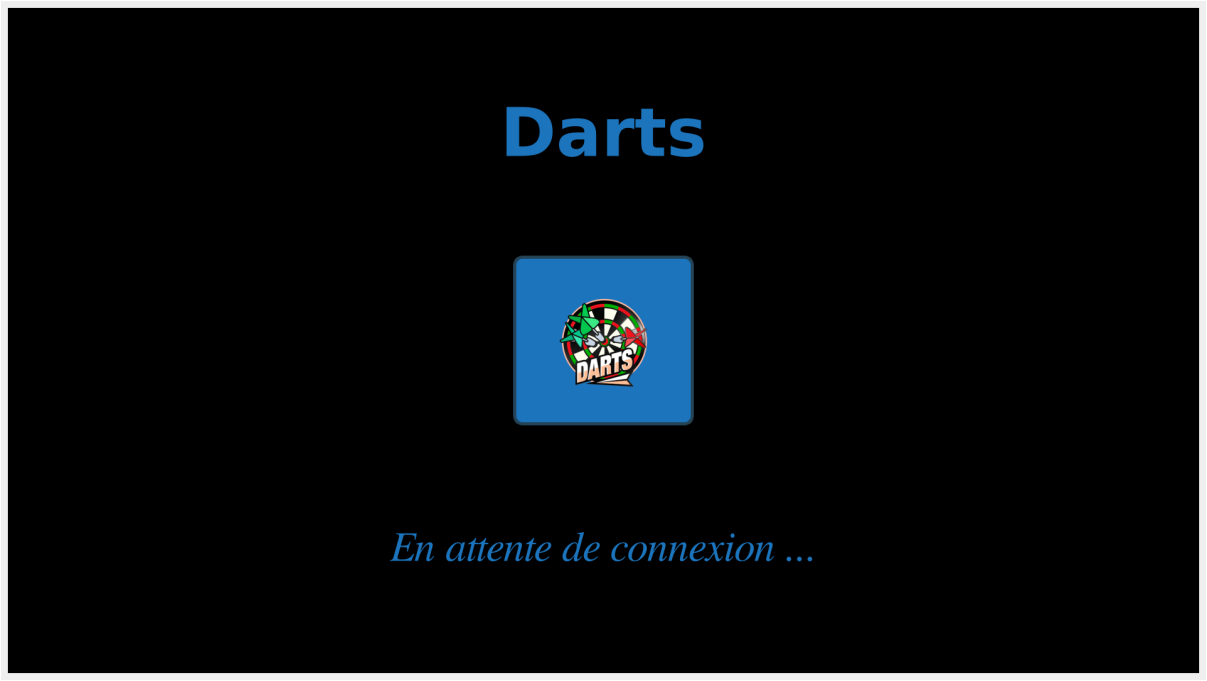
Trame	Description
`$DARTS;START`	Trame de début de partie. Cette trame contient le type de jeu, le nombre de joueurs est le nom des joueurs.
`$DARTS;HIT`	Trame de touche. Cette trame contient la valeur de la touche, le multiplicateur de touche et le numéro de touche.

$DARTS;OK	Trame d'acquittement. Cette trame sert d'acquittement quand une trame est reçue.
$DARTS;END	Trame de fin de partie. Cette trame sert à terminer la partie.

● L’interface homme machine (IHM)

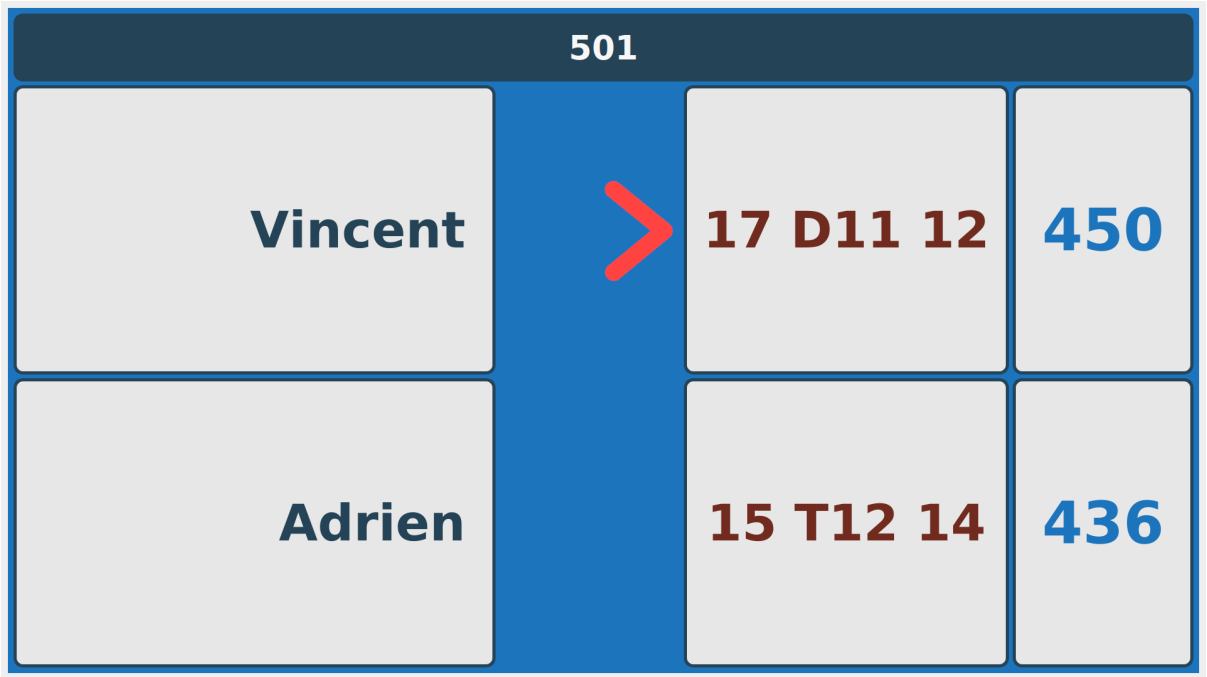
L’IHM a été réalisée sur QT, un framework C++.
Le programme est organisé en plusieurs fenêtres gérées à l’aide d'un **QStackedWidget**. Il stocke une pile de fenêtres où une seule est visible à la fois. Les différentes fenêtres sont réalisées sur la base d’un **QWidget**. L'affichage des textes utilise des **QLabel** et les images des **QPixmap**.
L’IHM est constitué de trois écrans, un écran d’accueil, un écran de partie et un écran de fin.

- **Écran d’accueil**



Sur l’écran d’accueil nous pouvons voir le nom de l’application et un message “En attente de connexion …” qui changent “En attente de début de partie” lorsqu'on connecte le terminal mobile en Bluetooth.

- **Ecran de partie**



L'écran de partie est affiché lorsqu'on reçoit une trame **START**.
Format d'une trame **START** :

- **`$DARTS;START;501;2;Billy;Robert\r\n`**

Cette trame contient le type de jeu, le nombre de joueurs et le nom des joueurs qui sont ensuite affichés dans des labels.

Le logiciel traite les trames de touche (trame **HIT**) et affiche la valeur de la volée et le score restant.
Format d'une trame **HIT** :

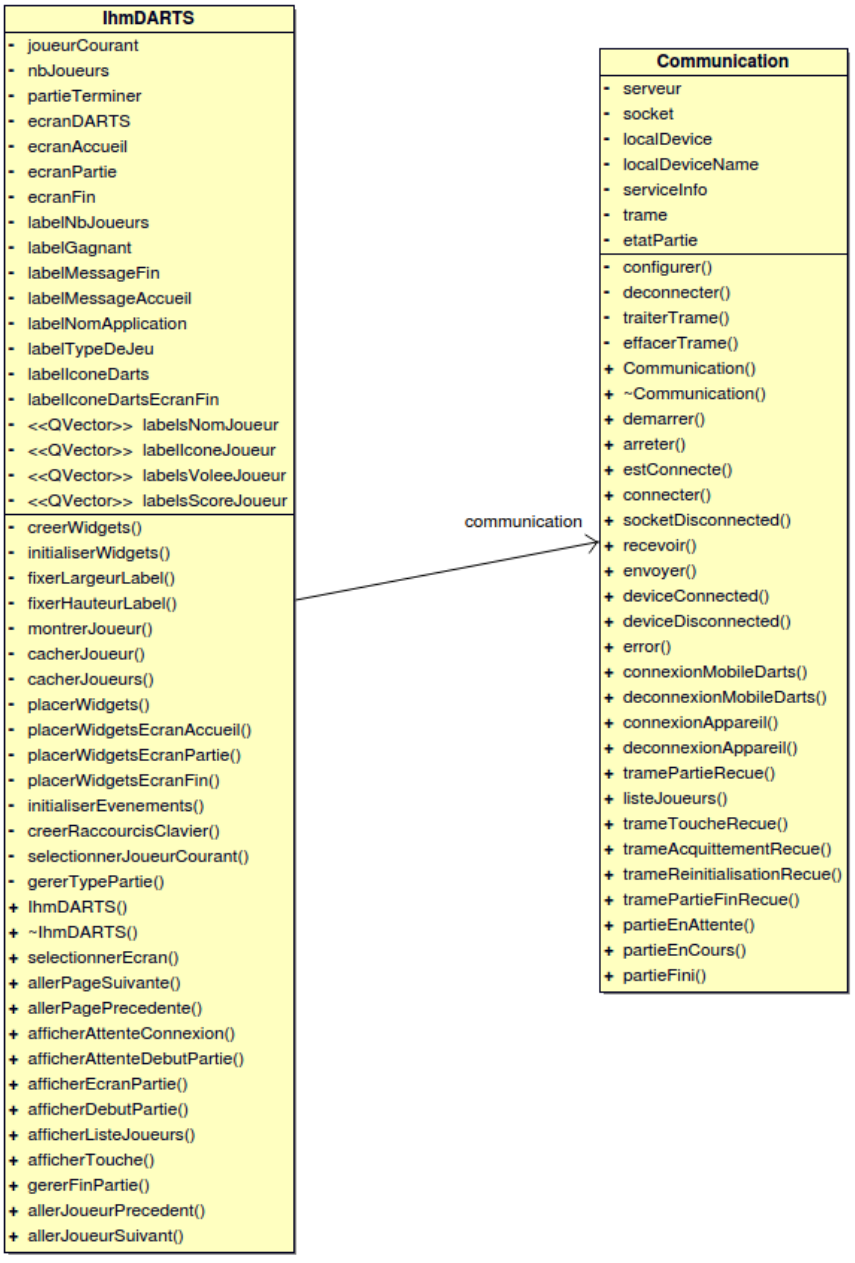
- **`$DART;HIT;25;2;0\r\n`**

- **Ecran de fin de partie**

L'écran de fin de partie est affiché quand le logiciel reçoit la trame **FIN** ou lorsque le score d'un joueur arrive à zéro.
Sur cet écran il est affiché le nom du joueur gagnant et un message de fin.
Format d'une trame **FIN** :

- **`$DARTS;FIN;LE_GAGNANT\r\n`**

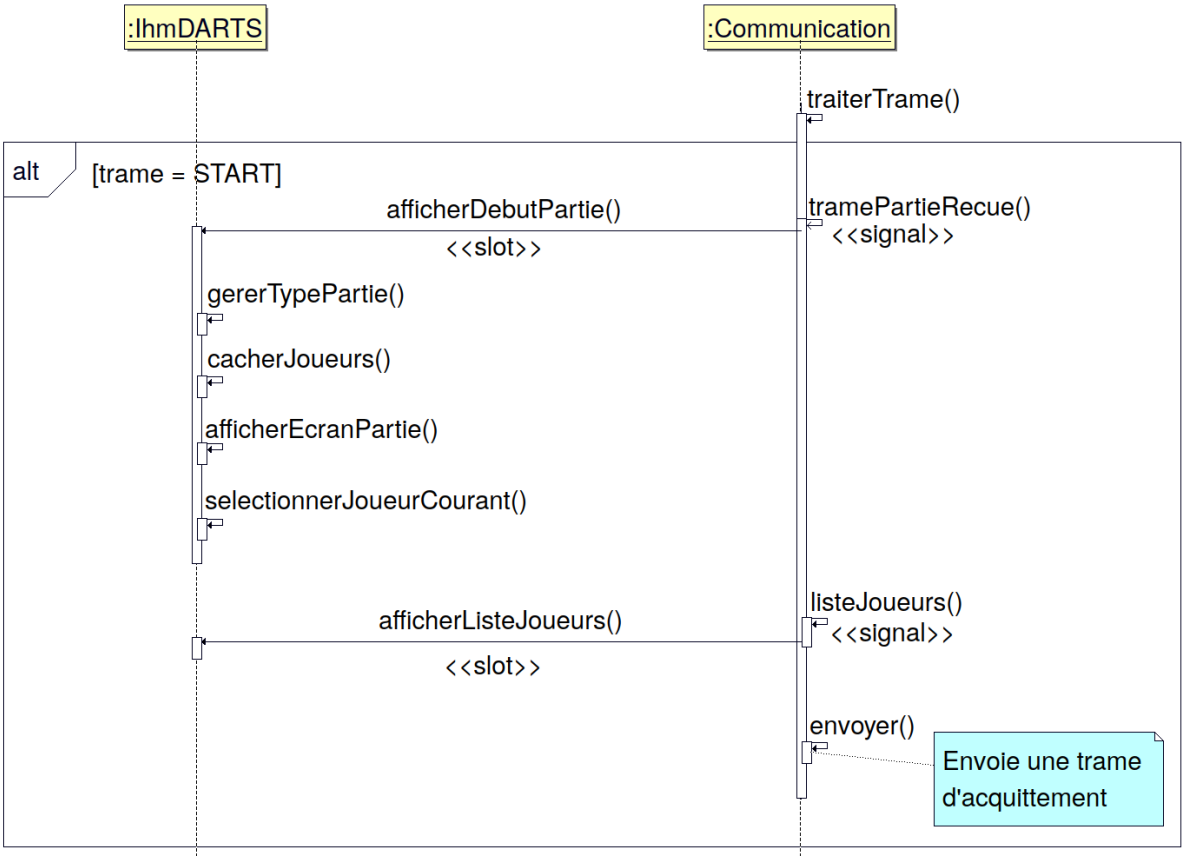
• Diagramme de classes



La classe ihmDARTS s'occupe de l’affichage des écrans et de la gestion des parties.

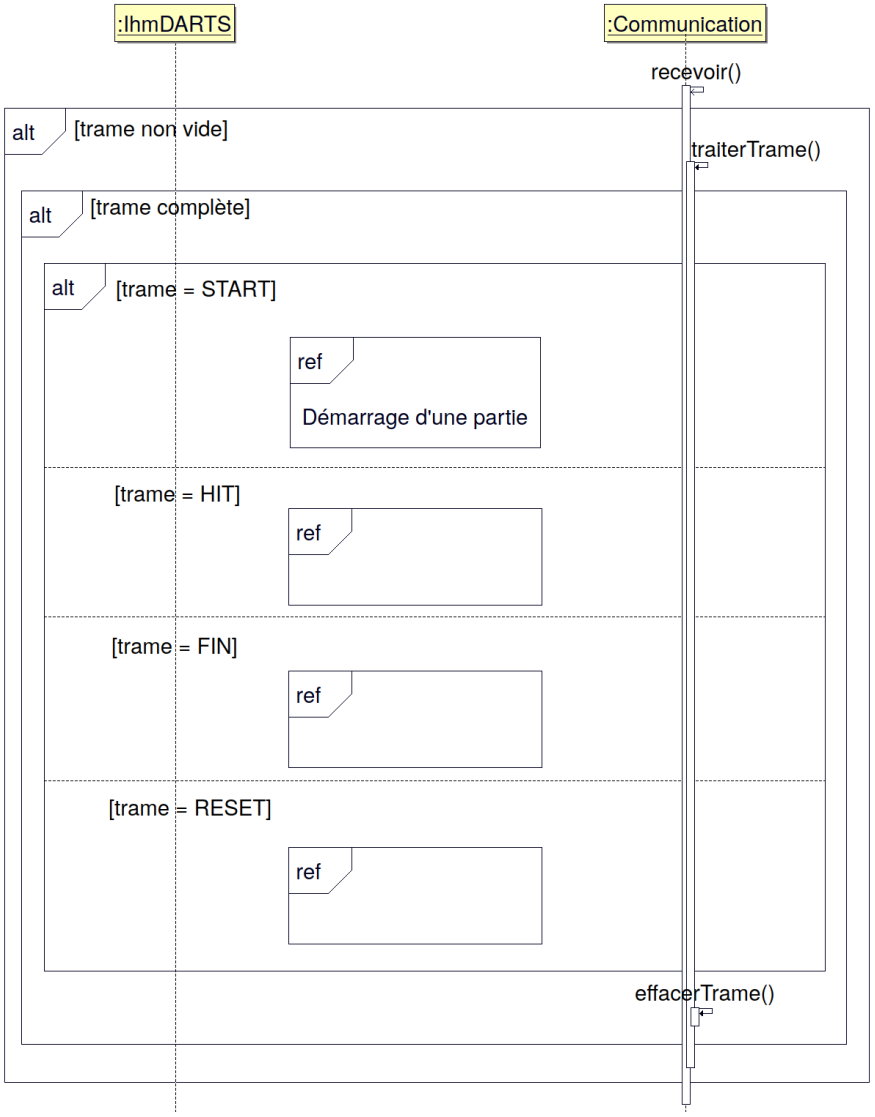
La classe Communication s'occupe de la liaison Bluetooth avec le terminal mobile et du traitement des différentes trames envoyées par le terminal mobile.

- **Scénario démarrage d'une partie**



On reçoit une trame START qui contient le type de jeu, le nombre de joueurs et le nom des joueurs. Ensuite on envoie un signal à l'IHM pour mettre à jour les informations de l'IHM. On envoie une trame d'acquittement une fois le traitement de trame terminé.

- **Dialoguer avec le terminal**



Quand on reçoit une trame on teste si la trame est vide, si elle n'est pas vide on teste si la trame est complète, ensuite on teste pour identifier le type de trames. Selon le type de trame on fait un traitement différent.

- **Décodage des trames**

```
bool Communication::traiterTrame()
{
   // Protocole
   const QString DEBUT_TRAME = "$";
   const QString ENTETE_TRAME = "DARTS";
   const QString DELIMITEUR_CHAMP = ";";
   const QString DELIMITEUR_FIN = "\r\n";
   const QString TRAME_PARTIE = "START";
   const QString TRAME_TOUCHE = "HIT";
   const QString TRAME_ACQUITTEMENT = "OK";
   const QString TRAME_REINITIALISATION = "RESET";
   const QString TRAME_FIN = "FIN";
   bool trameTraitee = false;

   // au moins une trame complète reçue ?
   if(trame.startsWith(DEBUT_TRAME + ENTETE_TRAME) &&
trame.endsWith(DELIMITEUR_FIN))
   {
       trame.remove(DELIMITEUR_FIN);

       if(trame.startsWith(DEBUT_TRAME + ENTETE_TRAME +
DELIMITEUR_CHAMP + TRAME_PARTIE))
       {
           etatPartie = ETAT_PARTIE_EN_COURS;
           emit partieEnCours(etatPartie);

           // Trame de début
          // Format : $DARTS;START;TYPE_PARTIE;NB_JOUEURS;NOMS\r\n
           QStringList champs = trame.split(";");
           int nbJoueurs = (champs.at(CHAMP_NB_JOUEURS)).toInt();
           emit tramePartieRecue(champs.at(CHAMP_TYPE_PARTIE),
nbJoueurs);
           QStringList nomsJoueur;
           for(int i = 0; i < nbJoueurs; ++i)
           {
               nomsJoueur << (champs.at(CHAMP_DEBUT_NOM_JOUEUR+i));
           }
```

```
            emit listeJoueurs(nomsJoueur);
            // Acquitte la trame
            envoyer(DEBUT_TRAME + ENTETE_TRAME + DELIMITEUR_CHAMP +
TRAME_ACQUITTEMENT + DELIMITEUR_FIN);
            trameTraitee = true;
        }
        else if(trame.startsWith(DEBUT_TRAME + ENTETE_TRAME +
DELIMITEUR_CHAMP + TRAME_TOUCHE))
        {
            // Trame de touche
            // Format : $DARTS;HIT;valeurTouche;multiplicateurTouche;ID\r\n
            QStringList champs = trame.split(";");
            int idTouche  = (champs.at(CHAMP_ID)).toInt();
            int valeurTouche  = (champs.at(CHAMP_TOUCHE)).toInt();
            int multiplicateurTouche =
(champs.at(CHAMP_MULTIPLICATEUR)).toInt();
            emit trameToucheRecue(idTouche, valeurTouche,
multiplicateurTouche);
            // Acquitte la trame
            envoyer(DEBUT_TRAME + ENTETE_TRAME + DELIMITEUR_CHAMP +
TRAME_ACQUITTEMENT + DELIMITEUR_FIN);
            trameTraitee = true;
        }
        else if(trame.startsWith(DEBUT_TRAME + ENTETE_TRAME +
DELIMITEUR_CHAMP + TRAME_REINITIALISATION))
        {
            // Trame de réinitialisation
            // Format : $DARTS;RESET\r\n
            emit trameReinitialisationRecue();
            // Acquitte la trame
            envoyer(DEBUT_TRAME + ENTETE_TRAME + DELIMITEUR_CHAMP +
TRAME_ACQUITTEMENT + DELIMITEUR_FIN);
            trameTraitee = true;
        }
        else if(trame.startsWith(DEBUT_TRAME + ENTETE_TRAME +
DELIMITEUR_CHAMP + TRAME_FIN))
        {
            etatPartie = ETAT_PARTIE_FIN;
            emit partieFini(etatPartie);
```

```
            // Trame de Fin
            // Format : $DARTS;FIN\r\n
            emit tramePartieFinRecue();
            // Acquitte la trame
            envoyer(DEBUT_TRAME + ENTETE_TRAME + DELIMITEUR_CHAMP +
TRAME_ACQUITTEMENT + DELIMITEUR_FIN);
            trameTraitee = true;
        }
        else
        {
            qDebug() << Q_FUNC_INFO << trame << "inconnue !";
            trameTraitee = false;
        }

        // Pour la prochaine trame
        effacerTrame();
    }
    else
    {
        qDebug() << Q_FUNC_INFO << trame << "incomplète !";
        trameTraitee = false;
        trame.clear();
    }

    return trameTraitee;
}
```

La méthode traiterTrame() teste si la trame reçue est valide pour cela elle vérifie si l'entête du protocole et le délimiteur de fin sont présents. Les trames valides seront ensuite testées pour savoir de quel type de trame il s'agit. Un signal contenant les champs sera émis.

● Tests de validation

Description	Réaliser
L'ihm "basique" est fonctionnelle.	Oui
La liaison Bluetooth avec le terminal mobile est fonctionnelle.	Oui
La réception de trame est opérationnelle.	Oui
L'affichage du nom et du score est opérationnel.	Oui
Changement automatique d'utilisateur après une volée.	Oui

Annexe

Protocole DARTS

1. Format général des trames

1.1. Type de contenu

- Caractères ASCII

1.2. Délimiteurs

- Début : **$DARTS**
- Fin : **\r\n**
- Champ de début : **;**

1.3. Liste des trames

- Trame de début : **START**
- Trame de touche : **HIT**
- Trame de joueur suivant : **NEXT**
- Trame de fin : **END**
- Trame d'acquittement : **OK**

1.4. Sens des trames

- émetteur : Application mobile Darts, Module cible Darts.
- Récepteur : ihm Darts, Application mobile Darts, Module cible Darts.

1.5. Liste des types de partie

- 301 : Partie en 301 Points ou pour gagner il faut atteindre 0 points.
- 301 Double Out : Partie en 301 Points ou pour gagner il faut atteindre 0 points et finir sur un coup Double.
- 501 : Partie en 501 Points ou pour gagner il faut atteindre 0 points.
- 501 Double Out : Partie en 501 Points ou pour gagner il faut atteindre 0 points et finir sur un coup Double.

2. Format détaillé des trames

Champs

- Trame début :

Format : **$DARTS;START;TYPE_PARTIE;NB_JOUEURS[;NOMS]\r\n**
Exemple : **$DARTS;START;501;2;Billy;Robert\r\n**

- Trame de touche :

Format : **$DARTS;HIT;valeurTouche;multiplicateurTouche;ID\r\n**

Les valeurs possibles pour valeurTouche sont :

Les secteurs 1 à 20.

Le centre de la cible (la bulle ou *Bull's Eye*).

Les valeurs possibles pour multiplicateurTouche sont : 1, 2 (double) et 3 (triple)

Les valeurs possibles pour ID sont : 0 à 2

Exemples :
$DART;HIT;25;2;0\r\n → Fléchette bullseye (50 points)
$DART;HIT;25;1;0\r\n → Fléchette demi-bulle (25points)

Les simples :
$DART;HIT;1;1;0\r\n → 1 pts
$DART;HIT;2;1;0\r\n → 2 pts

```
$DART;HIT;3;1;0\r\n → 3 pts
$DART;HIT;4;1;0\r\n → 4 pts
$DART;HIT;5;1;0\r\n → 5 pts
$DART;HIT;13;1;0\r\n → 13 pts
$DART;HIT;14;1;0\r\n → 14 pts
$DART;HIT;15;1;0\r\n → 15 pts
$DART;HIT;16;1;0\r\n → 16 pts
$DART;HIT;17;1;0\r\n → 17 pts
$DART;HIT;18;1;0\r\n → 18 pts
$DART;HIT;19;1;0\r\n → 19 pts
$DART;HIT;20;1;0\r\n → 20 pts

Les doubles :

$DART;HIT;1;2;0\r\n → 2 pts
$DART;HIT;2;2;0\r\n
$DART;HIT;3;2;0\r\n
$DART;HIT;4;2;0\r\n
$DART;HIT;5;2;0\r\n
$DART;HIT;6;2;0\r\n
$DART;HIT;7;2;0\r\n
$DART;HIT;8;2;0\r\n
$DART;HIT;9;2;0\r\n
$DART;HIT;20;2;0\r\n
$DART;HIT;22;2;0\r\n
$DART;HIT;22;2;0\r\n
$DART;HIT;23;2;0\r\n
$DART;HIT;24;2;0\r\n
$DART;HIT;25;2;0\r\n
$DART;HIT;26;2;0\r\n
$DART;HIT;27;2;0\r\n
$DART;HIT;28;2;0\r\n
$DART;HIT;29;2;0\r\n
$DART;HIT;20;2;0\r\n → 40 pts

Les triples :
```

```
$DART;HIT;1;3;0\r\n → 3 pts
$DART;HIT;2;3;0\r\n
$DART;HIT;3;3;0\r\n
$DART;HIT;4;3;0\r\n
$DART;HIT;5;3;0\r\n
$DART;HIT;6;3;0\r\n
$DART;HIT;7;3;0\r\n
$DART;HIT;8;3;0\r\n
$DART;HIT;9;3;0\r\n
$DART;HIT;30;3;0\r\n
$DART;HIT;33;3;0\r\n
$DART;HIT;33;3;0\r\n
$DART;HIT;33;3;0\r\n
$DART;HIT;34;3;0\r\n
$DART;HIT;35;3;0\r\n
$DART;HIT;36;3;0\r\n
$DART;HIT;37;3;0\r\n
$DART;HIT;38;3;0\r\n
$DART;HIT;39;3;0\r\n
$DART;HIT;30;3;0\r\n → 60 pts
```

- Trame d'acquittement :

Format : **$DARTS;OK\r\n**

- Trame de fin :

Format : **$DARTS;FIN;LE_GAGNANT\r\n**