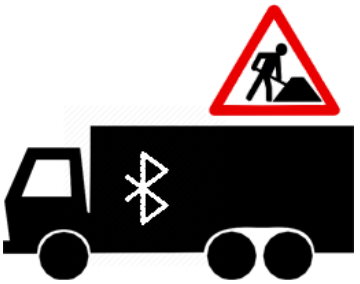
# Projet Io-Trucks

# Dossier Technique




## BERTRAND Émilien

**LaSalle Avignon Session 2021**

# Sommaire

# Présentation générale

Il s’agit de réaliser un système numérique intégré à un véhicule industriel qui permet d’interagir à courte portée sur les accessoires et de visualiser les informations associées sur une application mobile, tout en restant simple et ergonomique d’utilisation. Les accessoires en question, à savoir un hayon, un éclairage de confort, un triangle de signalisation dépliable muni de LEDs* en courant série et d’un gyrophare d’intervention, seront commandés par un microcontrôleur* (Espressif ESP32) qui lui même communiquera par Bluetooth* à l’application.

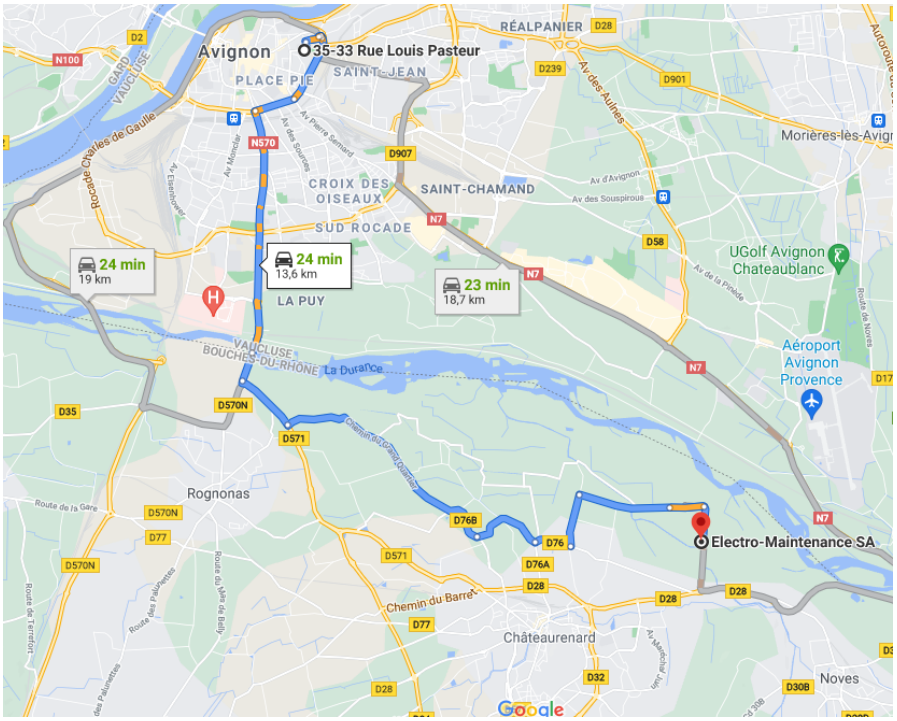
## Analyse de l’existant

Electro Maintenance est une entreprise spécialisée sur le marché des accessoires et équipements pour véhicules industriels.

Activités : Fabricant et distributeur de pièces détachées électriques, hydrauliques et accessoires destinés aux professionnels de la route.

Ses clients sont : Carrossiers, aménageurs, constructeurs, concessions PL, VUL et agricole, loueurs, TP, collectivités, etc...

Cette société ne cesse d'innover et d'étendre sa gamme, notamment dans le domaine des objets connectés avec son offre Atlas Connect.





Le système Atlas Connect permettra par exemple :

- de piloter l'éclairage
- de commander l'ouverture/fermeture de porte
- de monter/descendre le hayon

Electro Maintenance recherche des solutions pour les besoins suivants :

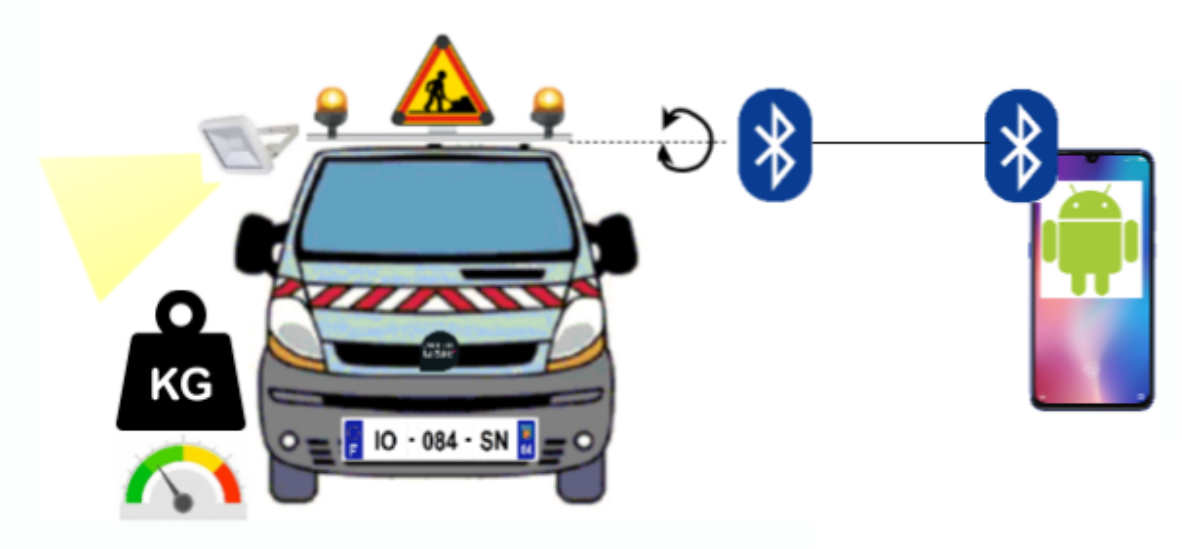
- d'un triangle rabattable
- d'un éclairage de confort en périphérie du camion
- d'une supervision de hayons
- d'un détecteur de surcharge

# Expression du besoin

Le système « Io-Trucks » devra remplir les missions suivantes :

- de déployer un triangle de signalisation fixé sur le toit du camion
- de signaler l'état d'une intervention (par feux de balisage et gyrophare)
- de piloter les éclairages périphériques (projecteur en périphérie du camion)
- d'acquérir les données de fonctionnement (état du triangle, état des éclairages, état du gyrophare et de surcharge du véhicule, ...) du camion
- d'assurer la transmission des données des états du camion via une communication sans fil (Bluetooth)
- d'afficher les données de fonctionnement reçues du camion sur l'écran du terminal mobile

# Description structurelle du système




Le camion « Io-Trucks » sera équipé :

- d'un système pilotant le triangle de signalisation
- d'un système de transmission de données sans fil Bluetooth
- d'un système mesurant le niveau de chargement du camion
- d'un système du supervision de hayon
- d'un système d'éclairage de confort (en option)

# Répartition des tâches

| | | |
|---|---|---|
| Étudiant 1<br>☒ EC ☐ IR | - Commander la montée/descente du triangle<br>- Informer de la position du triangle (haute/basse)<br>- Piloter les feux de signalisations du triangle<br>- Communiquer avec le terminal mobile<br>- Recevoir les ordres du terminal mobile<br>- Piloter les gyrophares (option)<br>- Piloter l'éclairage de confort (option) | Installation :<br>Le système embarqué, sa carte électronique et ses actionneurs<br><br>Mise en oeuvre :<br>La motorisation du triangle, les feux de signalisation, le gyrophares et la liaison sans fil<br><br>Configuration :<br>Les fins de courses du triangle, les actionneurs et la liaison sans fil<br><br>Réalisation :<br>Les diagrammes SysML, Le code source et les schémas du module et la carte électronique<br><br>Documentation :<br>Le dossier technique et les documents relatifs au module, un guide de mise en route et d'utilisation du module |
| Étudiant 2<br>☒ EC ☐ IR | - Évaluer le niveau de chargement du camion<br>- Avertir d'un dépassement du niveau de chargement (ponctuel et/ou répétitif)<br>- Communiquer avec le terminal mobile<br>- Commander la montée/descente du hayon | Installation :<br>Le système embarqué, sa carte électronique et ses capteurs<br><br>Mise en oeuvre :<br>La liaison sans fil, les capteurs, les actionneurs et la carte électronique<br><br>Configuration :<br>La mesure du niveau de chargement et le pilotage du hayon<br><br>Réalisation :<br>Les diagrammes SysML, Le code source et les schémas du module et la carte électronique<br><br>Documentation :<br>Le dossier technique et les documents relatifs au module, un guide de mise en route et d'utilisation du module |

| | | |
|---|---|---|
| Étudiant 3<br>☐ EC ☒ IR | - Appairer les accessoires<br>- Contrôler les accessoires (triangle, hayon et l'éclairage de confort)<br>- Visualiser les informations associées aux accessoires (déploiement du triangle, niveau de chargement et l'état de l'éclairage de confort) | Installation :<br>Le système embarqué, sa carte électronique et ses capteurs<br><br>Mise en oeuvre :<br>La liaison sans fil, les capteurs, les actionneurs et la carte électronique<br><br>Configuration :<br>La mesure du niveau de chargement et le pilotage du hayon<br><br>Réalisation :<br>Les diagrammes SysML, Le code source et les schémas du module et la carte électronique<br><br>Documentation :<br>Le dossier technique et les documents relatifs au module, un guide de mise en route et d'utilisation du module |

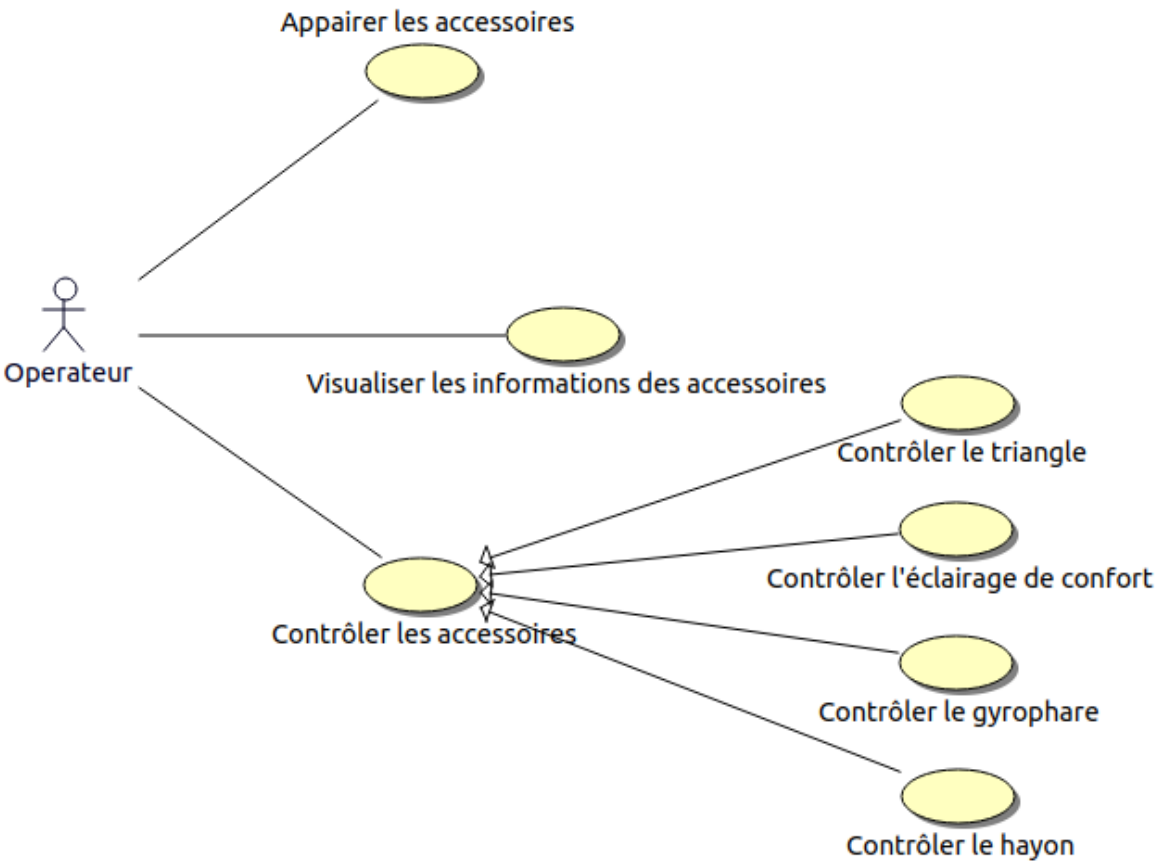
# Présentation personnelle

## Objectifs

Liste des fonctionnalités attendues spécifiquement en IR :

- Appairer les accessoires
- Contrôler les accessoires (triangle, hayon, éclairage de confort)
- Visualiser les informations associées aux accessoires (déploiement du triangle, niveau de chargement, état de l'éclairage de confort)

## Diagramme des cas d'utilisation




L'acteur Operateur pourra utiliser les accessoires du camion Io-Trucks en s'appairant via une communication sans fil, de visualiser les informations de celui-ci et de les contrôler directement à partir de l'application.

# Planification par priorités

| Haute | Moyenne | Basse |
|---|---|---|
| Visualiser les informations associées aux accessoires (déploiement du triangle, niveau de chargement, état de l'éclairage de confort) | Contrôler les accessoires (triangle, éclairage de confort, gyrophare, hayon) | Appairer les accessoires |
| Rendre l'interface la plus simple d'utilisation possible (ne doit pas prendre l'attention en conduisant) | | |

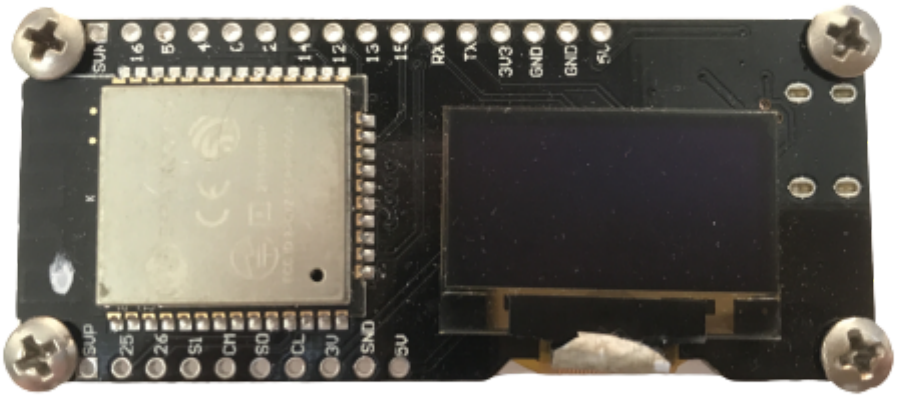
# Planification par itérations

| Itération 1 | Itération 2 | Itération 3 |
|---|---|---|
| Mettre en oeuvre Android Studio | Mettre en oeuvre le Bluetooth sous Android | Appairer les accessoires |
| Réaliser la maquette de l'IHM* | Spécifier le protocole de communication | Contrôler les accessoires (triangle, éclairage de confort, gyrophare, hayon) |
| | | Visualiser les informations associées aux accessoires (déploiement du triangle, niveau de chargement, état de l'éclairage de confort) |
| | | Finalisation de l'interface |

# Matériels

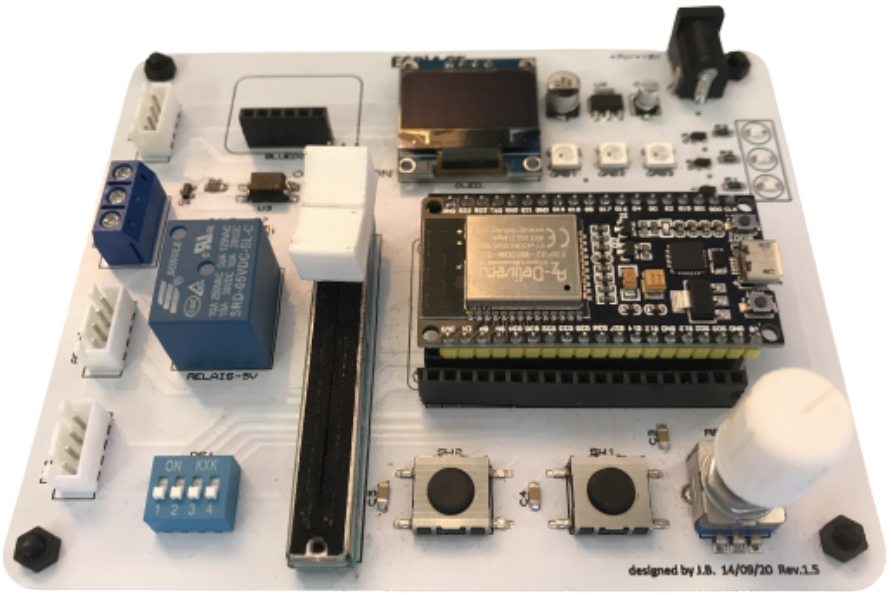
## Hardware

### Microcontrôleur ESP32 (Simulateur)




- Microcontrôleur : Dual core de 80 MHz à 240 MHz
- Mémoire flash* : 4 MB
- WiFi : 802.11 b/g/n
- Bluetooth : 4.0 LE et BR /EDR

Microcontrôleur Espressif ESP32




- Microcontrôleur : Dual core de 32 bit LX6
- Mémoire ROM* : 448 KB
- Mémoire SRAM* : 520 KB
- WiFi : 802.11 b/g/n
    - 802.11 n (2.4 GHz) 150 Mbps
- Bluetooth : 4.2 BR/EDR* LE

Tablette numérique (Samsung Galaxy Tab 4)

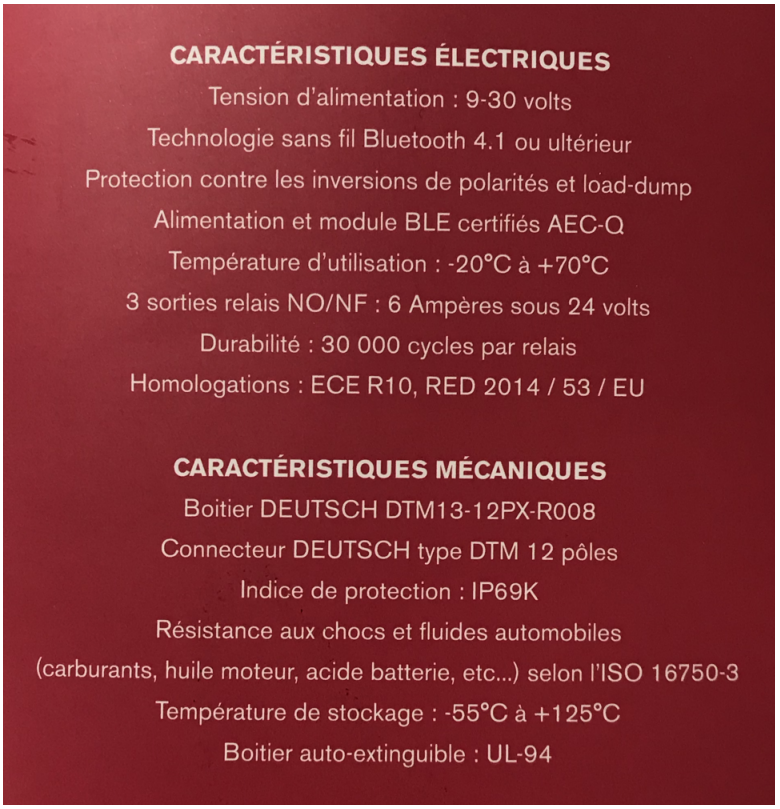

- Numéro du modèle : SM-T530
- Version Android : 5.0.2
- CPU* : 1.2GHz Quad-Core ARM Cortex-A7
- GPU* : Qualcomm Adreno 305
- Mémoire RAM* : 1,5Go
- Stockage Interne : 16Go (12Go disponible)

## Software

| Logo | Nom | Version |
|---|---|---|
|  | Android Studio | 4.1.2 |
|   | BOUML | 7.11 |
|  | Linux | 5.8.0-53-generic |
|   | SubVerSion | 1.13.0 |
|  | Ubuntu | 20.04 |

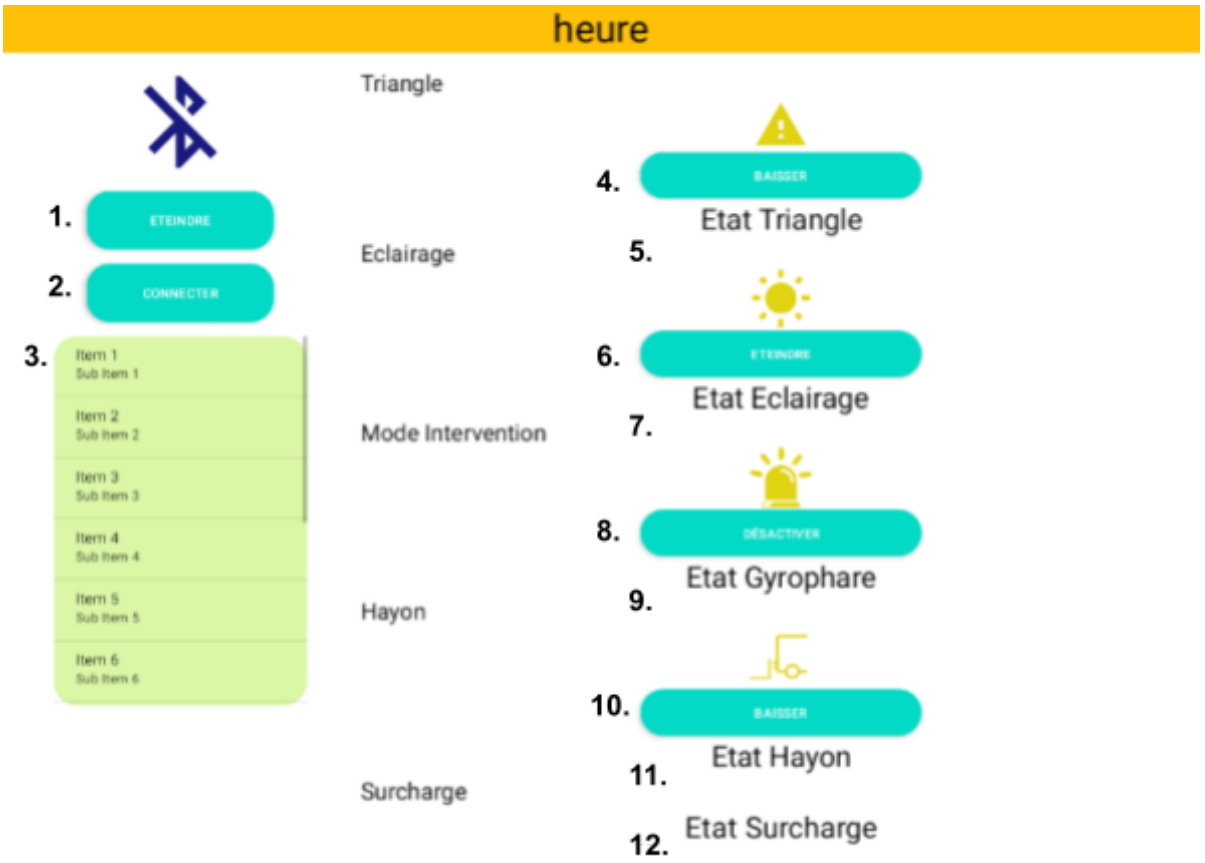
# Partie Physique

## Bluetooth



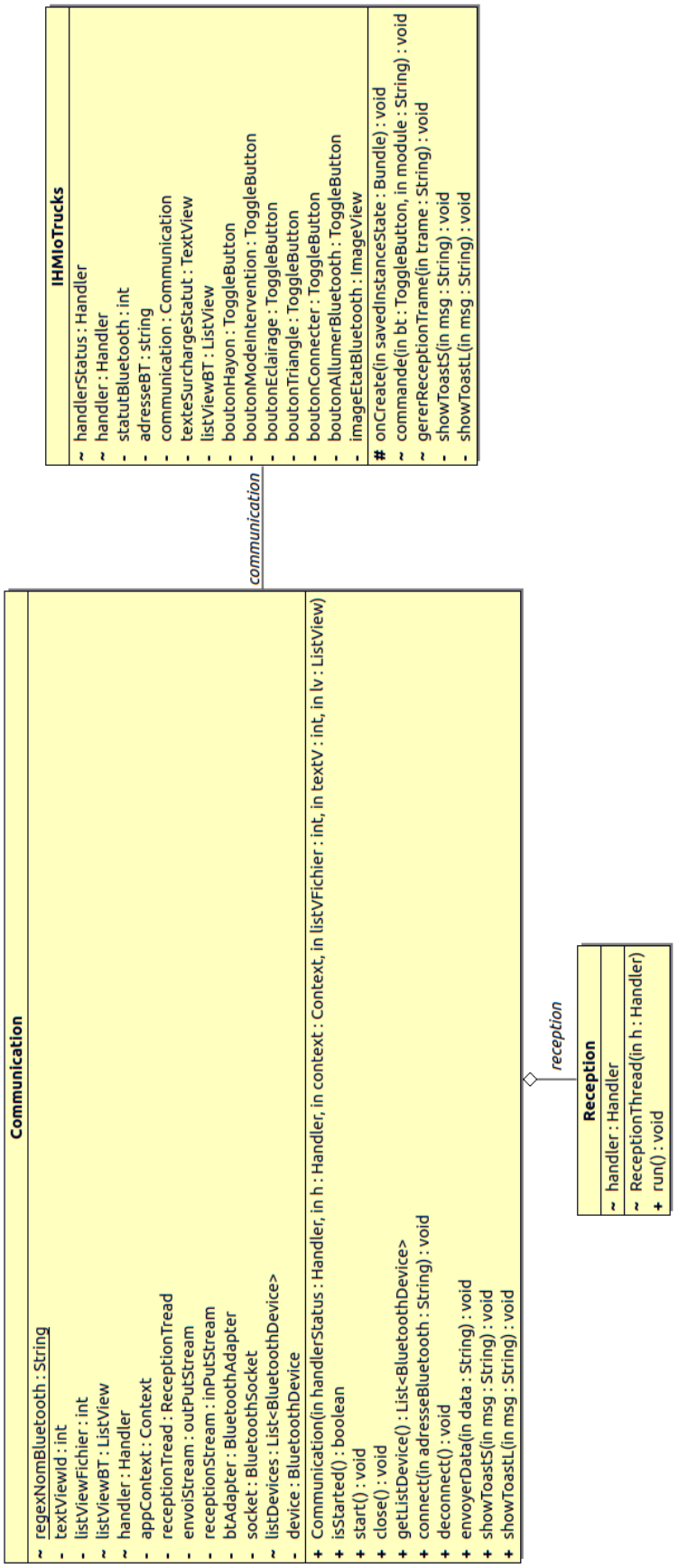

- Portée : ~ 10m
- Fréquence : 2,4 GHz
- Débit de données brutes : 1 Mbps
- Protocole : RF* Bluetooth LE*
- Version : 4.1
- Interopérabilité avec Bluetooth 4.0 et versions ultérieures
- Modulation : PSK*

# Prototypage et maquette de l'IHM

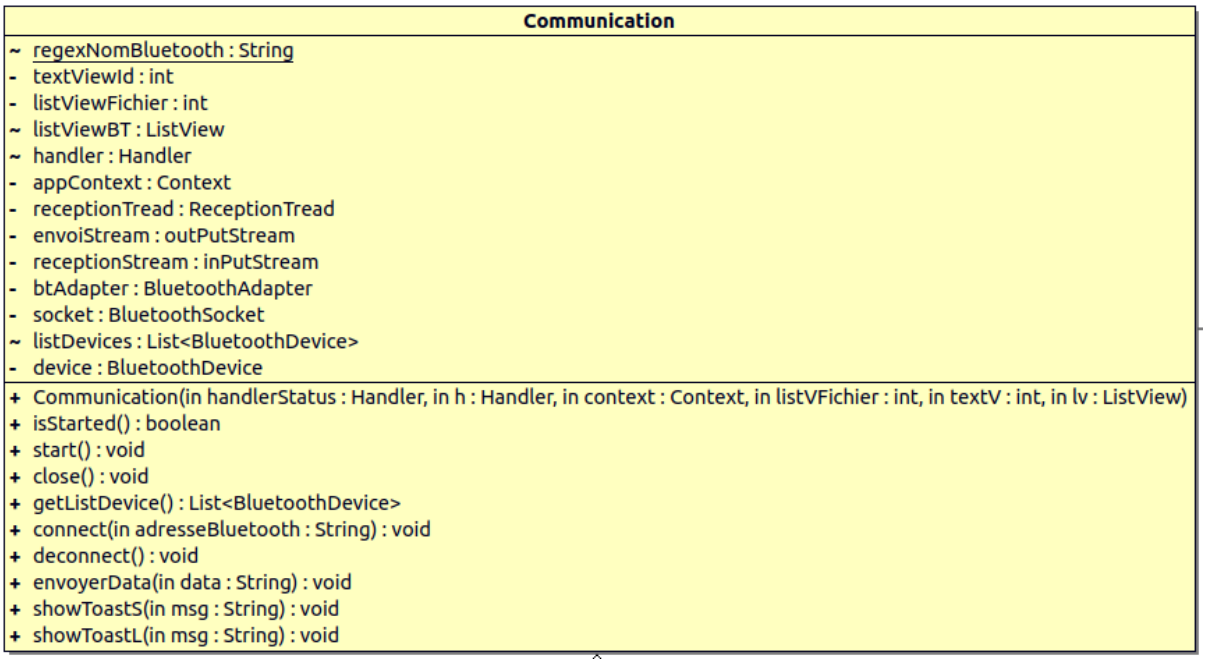



| | |
|---|---|
| 1. | Allumer / Éteindre le Bluetooth |
| 2. | Connecter / Déconnecter d'un Bluetooth (Si un Bluetooth est sélectionné) |
| 3. | Liste des périphériques |
| 4. | Bouton Lever / Baisser le Triangle |
| 5. | Etat du Triangle |
| 6. | Bouton Allumer / Éteindre l'éclairage de confort |
| 7. | Etat de l'éclairage de confort |
| 8. | Bouton Activer / Désactiver le Mode Intervention |
| 9. | Etat du Gyrophare |
| 10. | Bouton Lever / Baisser le Hayon |
| 11. | Etat du Hayon |
| 12. | Etat de la Surcharge (Normal / Attention / Surcharge) |

# Diagramme de classes

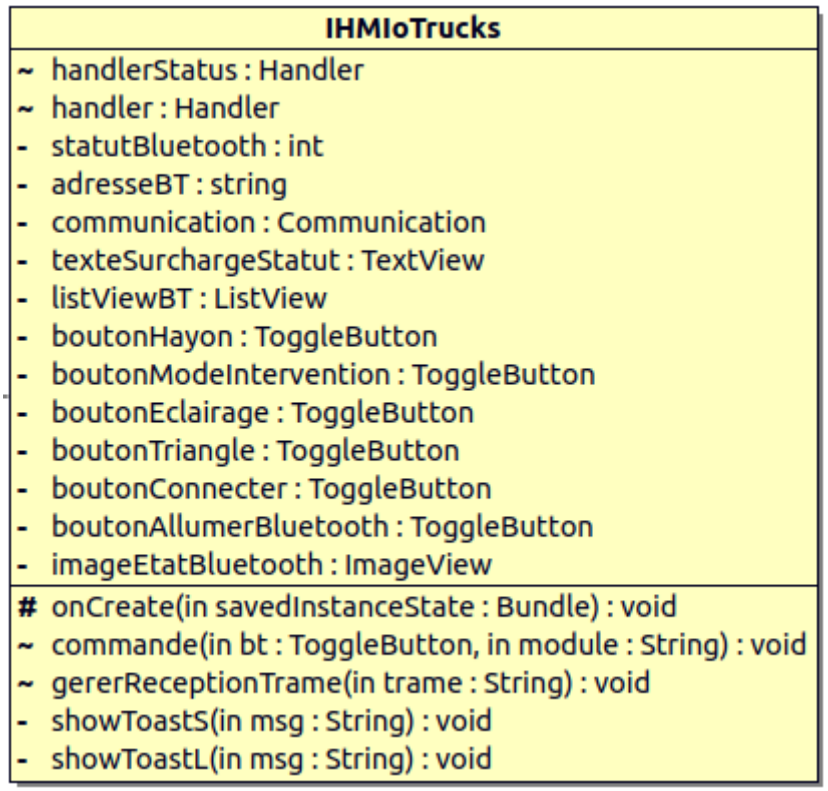


# Classe IHMIoTrucks



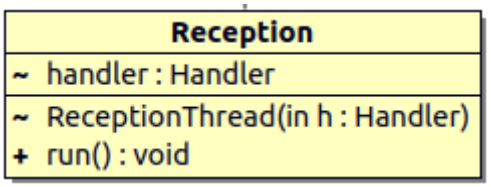

Cette classe sert à afficher : le Bluetooth, le triangle + LEDs, le mode d'intervention (le gyrophare) et l'éclairage de confort.

## Classe Communication




Son rôle est de fabriquer et de vérifier les trames de plus elle peut envoyer les trames et les recevoir.

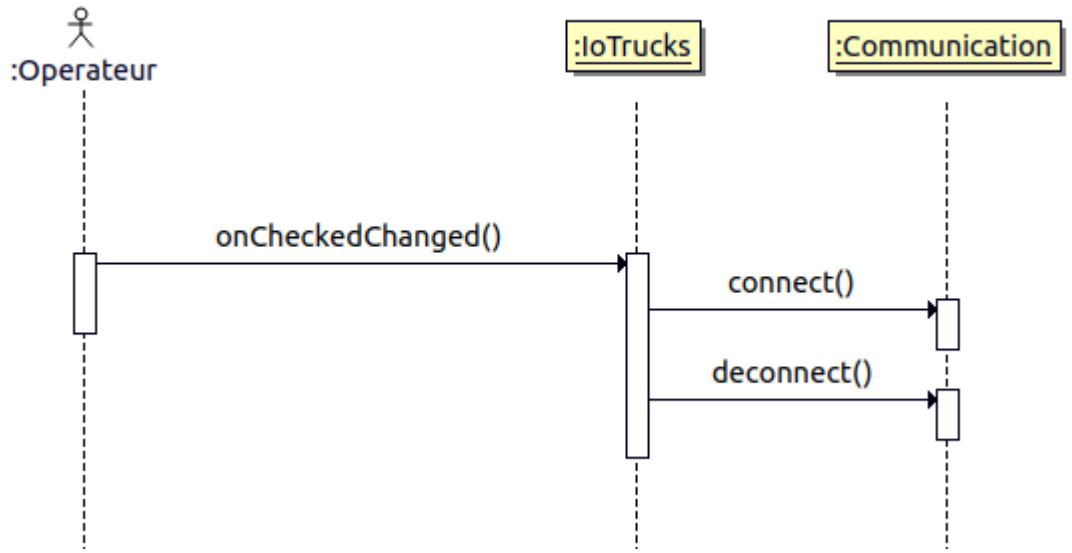
## Classe Reception




Son rôle est une classe interne dans la classe IHMIotrucks qui gère la réception des trames.

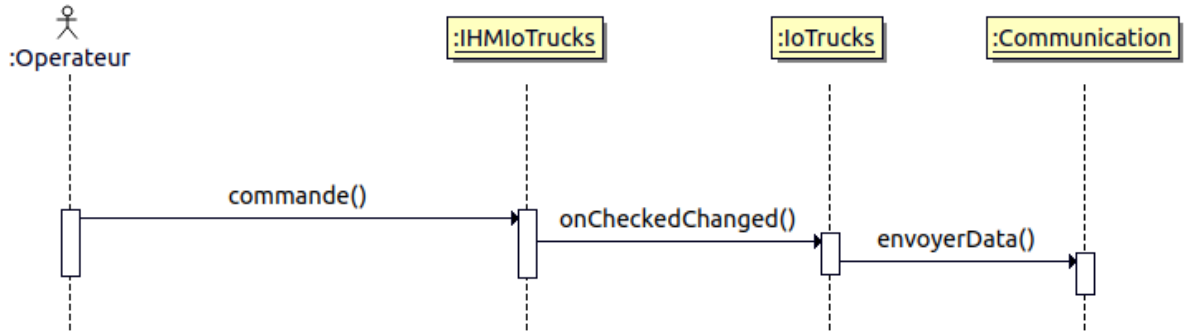
# Scénarios

## Scénario Appairer les accessoires



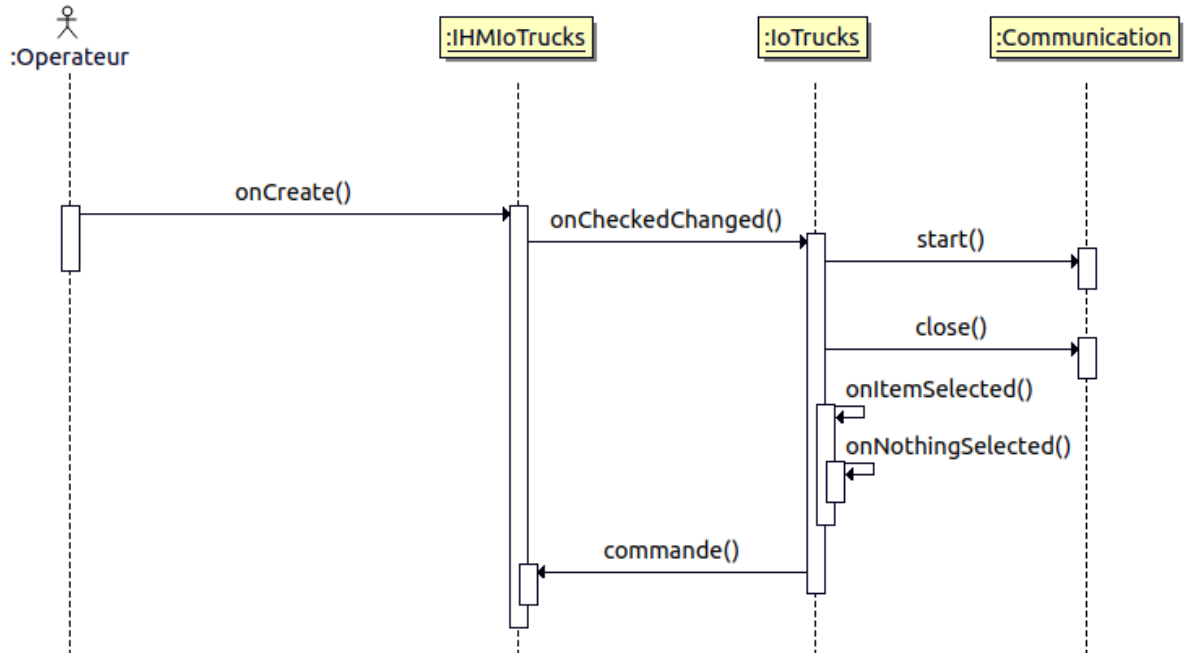

Lorsque l'on appaire les accessoires, la classe IHMIotrucks utilise la fonction gererBoutonsBluetooth(Communication communication) pour voir les états. Il l'utilise aussi au boutonTriangle pour l'état des LEDs et l'état du triangle, il l'utilise aussi au boutonModeIntervention pour l'état du gyrophare, il l'utilise aussi au boutonEclairage pour l'état de l'éclairage de confort et il l'utilise aussi au boutonHayon pour l'état du hayon.

## Scénario Contrôler les accessoires



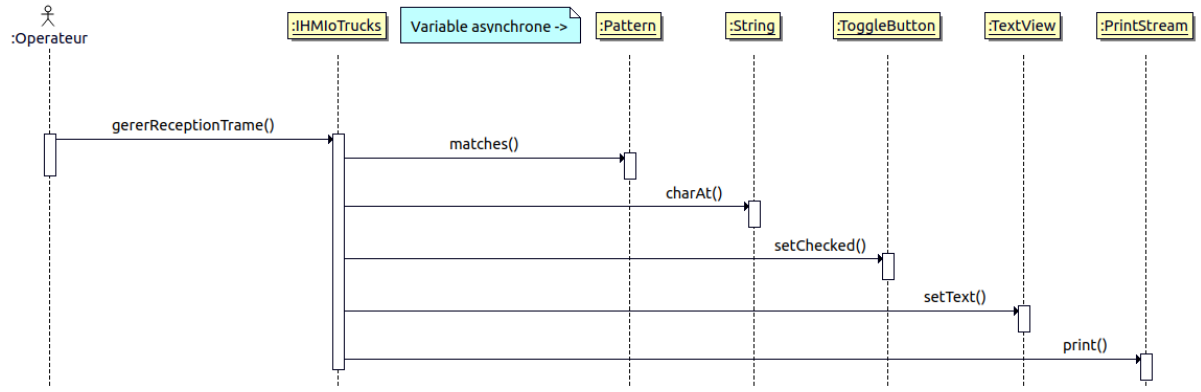

Lorsque l'on veut contrôler les accessoires, on a la possibilité de contrôler le triangle pour le lever ou le baisser, de contrôler l'éclairage pour allumer ou l'éteindre, de contrôler le mode d'intervention en l'activant et le désactivant selon le contexte de l'intervention et enfin de contrôler le hayon pour le lever et le baisser.

## Scénario Démarrage de l'application




Lorsque l'on démarre l'application, l'affichage de la tablette se lance. Si le Bluetooth est activé, la liste déroulante affiche automatiquement les Bluetooths Io-Trucks disponibles, on peut alors sélectionner un Bluetooth.

## Scénario Visualiser les informations des accessoires




Lorsque l'on veut visualiser les informations des accessoires, on reçoit la trame pour la vérifier, si elle est valide, on extrait les données de la trame pour pouvoir avoir une visue sur l'IHM avec le Bluetooth, le triangle, le mode d'intervention, l'éclairage et la surcharge.

# Protocole de communication

## Trame de commande

| | |
|---|---|
| T (triangle) | 1=Lever / 0=Baisser |
| G (gyrophares) | 1=Allumer / 0=Eteindre |
| E (eclairage de confort) | 1=Allumer / 0=Eteindre |
| H (hayon) | 1=Lever / 0=Baisser |

Si l'application envoie une trame pour indiquer au système que le matériel qu'elle veut piloter son état.

Exemples :

$iotruck;T;1\r\n → Lever triangle + allumer leds
$iotruck;T;0\r\n → Baisser triangle + éteindre leds
$iotruck;G;1\r\n → Allumer gyrophares
$iotruck;G;0\r\n → Eteindre gyrophares
$iotruck;E;1\r\n → Allumer éclairage de confort
$iotruck;E;0\r\n → Eteindre éclairage de confort
$iotruck;H;1\r\n → Monter le hayon
$iotruck;H;0\r\n → Descendre le hayon

## Trame de requêtes

Si l'application veut connaître l'état des matériels, elle doit envoyer une trame de requête.

**Partie triangle / gyrophares / éclairage de confort :**

Décomposition d'une trame : $iotruck;S;1\r\n
$iotruck;State1\r\n

**Partie hayon / niveau de charge :**

Décomposition d'une trame : $iotruck;S;2\r\n
$iotruck;State2\r\n

## Trame d'états (application vers système)

Les trames d'états sont des réponses aux trames de requêtes envoyées par l'application (système vers application).

**Partie triangle / gyrophares / éclairage de confort :**

Décomposition d'une trame :

$iotruck;S1;1;1;1\r\n
$iotruck;State1;état triangle;état gyrophares;état éclairage\r\n

Les différents états pour la trame S1 :

| | |
|---|---|
| état triangle | 1=Levé / 0=Baissé |
| état gyrophares | 1=Allumé / 0=Eteint |
| état éclairage | 1=Allumé / 0=Eteint |

Exemples :

$iotruck;S1;1;0;0\r\n
$iotruck;S1;0;1;0\r\n
$iotruck;S1;0;0;1\r\n
$iotruck;S1;1;1;0\r\n
$iotruck;S1;0;1;1\r\n
$iotruck;S1;1;0;1\r\n
$iotruck;S1;1;1;1\r\n

**Partie hayon / niveau de charge :**

Décomposition d'une trame : $iotruck;S2;x;x\r\n
$iotruck;State2 ;état hayon;état charge\r\n

Les différents états pour la trame S2 :

| | |
|---|---|
| état hayon | 1=Levé / 0=Baissé |
| état charge | 0=Normal / 1=Attention / 2=Surcharge |

Exemples :

$iotruck;S2;0;0\r\n
$iotruck;S2;1;0\r\n
$iotruck;S2;0;1\r\n
$iotruck;S2;1;1\r\n
$iotruck;S2;0;2\r\n
$iotruck;S2;1;2\r\n

## Trame de service (application vers système)

L'application envoie périodiquement (toutes les secondes) une trame ALIVE pour maintenir la connexion ouverte.

Décomposition d'une trame : $iotruck;A\r\n
$iotruck;Alive\r\n

Le système répondra par une trame d'acquittement.

Décomposition d'une trame : $iotruck;A\r\n
$iotruck;Ack\r\n

# Code Sources de démonstration

## Envoie de trame

```
/**
  * @brief Envoyer une trame
  * @param data
  */

public void envoyerData(String data)
{
     try
     {
          envoiStream.write(data.getBytes()); //!< On écrit les données dans le buffer
d'envoi

          envoiStream.flush(); //!< On s'assure qu'elles soient bien envoyées
          showToastS("Envoyé avec succès");
     }
     catch (IOException e)
     {
     showToastL("Echec de l'envoi\nerreur : "+e.getMessage());
     e.printStackTrace();
     }
}
```

Les fonction write() et flush() sont dans un try car elles ne sont pas garanties de fonctionner.
envoiStream est un canal d'émission auquel on fait écrire (write) une trame (data) que l'on aura convertie en binaire et que l'on envoie dans le flux (flush). Le catch permet de renvoyer un message d'erreur si ça ne fonctionne pas.

## Réception de trame

```
final Handler handler = new Handler() //!< Gestionnaire
{
     public void handleMessage(Message msg)
     {
          String data = msg.getData().getString("receivedData");

          long t = System.currentTimeMillis();
          //showToast(data);

          gererReceptionTrame(data);
     }
};
```

La classe Handler permet de réceptionner des messages / variables en dehors du onCreate() (c'est une tâche asynchrone).
handleMessage() réceptionne tous messages (une trame) possédant la key "receivedData", puis décortique la trame grâce à gererReceptionTrame().

Classe permettant la réception asynchrone des trames

```
/**
  * @class ReceptionThread
  * @brief Classe privé de reception de trame
  */
private class ReceptionThread extends Thread
{
     Handler handler;

     ReceptionThread(Handler h)
     {
          handler = h;
     }

     @Override public void run()
     {
          while(true)
          {
               try
               {
                    if(receptionStream.available() > 0) //!< On teste si des données sont disponibles

                    {
                         byte buffer[] = new byte[100
                         int k = receptionStream.read(buffer, 0, 100); //!< On lit les données, k représente
le nombre de bytes lu

                         if(k > 0)
                         {
                              byte rawdata[] = new byte[k];
                              for(int i=0;i<k;i++)
                              rawdata[i] = buffer[i];

                              String data = new String(rawdata);

                              Message msg = handler.obtainMessage(); //!< On envoie les données dans le
thread de l'UI pour les afficher
                              Bundle b = new Bundle();
                              b.putString("receivedData", data);
                              msg.setData(b);
                              handler.sendMessage(msg);
                         }
                    }
               }
               catch (IOException e)
               {
                    showToastL("Echec de reception\nErreur : "+e.getMessage());
                    e.printStackTrace();
               }
          }
     }
}
```

La classe ReceptionThread crée des tâches asynchrones.
Pendant la fonction run(), on vérifie que receptionStream (c'est un canal de réception) soit disponible puis on récupère les datas (read()) dans un tableau de byte.
Si la data (k) n'est pas nulle, on convertit le tableau en String, se qui forme un message puis on envoie les données (data) dans un nouveau handler avec la key "receivedData".

## Code .xml de l'IHM

### TextClock

```
<!-- Date et Heure -->

      <TextClock
            android:id="@+id/textClockDateHeure"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:autoSizeTextType="uniform"
            android:format12Hour="@null"
            android:format24Hour="dd/MM/yyyy HH:mm"
            android:text="heure"
            android:textAlignment="center"
            android:textColor="#000" />
```

Un TextClock permet d'afficher l'heure automatiquement sur le layout, il s'aligne avec l'heure du système. Il est possible de personnaliser un TextClock pour afficher les heures, les minutes, les secondes et la date sous différentes formes. Dans ce cas le TextClock affiche l'heure et les minutes au format 24h (HH:mm) et la date au format européen (dd/MM/yyyy).

### LinearLayout

```
<LinearLayout
      android:layout_width="match_parent"
      android:layout_height="0dp"
      android:layout_weight="94"
      android:weightSum="100"
      android:orientation="horizontal">
```

Un LinearLayout permet de mettre à la suite tous les éléments qu'il contient, il peut y avoir deux orientations, horizontales et verticales.

ImageView

```
<!-- Icon Bluetooth On/Off -->

            <ImageView
                  android:id="@+id/imageEtatBluetooth"
                  android:layout_width="174dp"
                  android:layout_height="0dp"
                  android:layout_marginTop="15dp"
                  android:layout_weight="15"
                  android:adjustViewBounds="true"
                  android:scaleType="fitCenter"
                  app:srcCompat="@drawable/ic_action_off"
                  tools:ignore="MissingConstraints" />
```

Un ImageView permet d'insérer une image dans un layout, il peut contenir des .xml, des .svg, et des images (.jpg, .png et .gif).

ToggleButton

```
<!-- Bouton Eteindre/Allumer -->

            <ToggleButton
                  android:id="@+id/boutonAllumerBluetooth"
                  android:layout_width="wrap_content"
                  android:layout_height="0dp"
                  android:layout_marginTop="15dp"
                  android:layout_weight="8"
                  android:background="@drawable/button_selector"
                  android:minWidth="200dp"
                  android:text="Allumage Bluetooth"
                  android:textColor="#fff"
                  android:textOff="Eteindre"
                  android:textOn="Allumer"
                  tools:ignore="MissingConstraints" />
```

Un ToggleButton permet de switcher en deux actions, le message à l'intérieur du bouton peut changer également.

ListView

```
<!-- Afficher la liste des appareils associés -->

            <ListView
                  android:id="@+id/listViewBluetooth"
                  android:layout_width="270dp"
                  android:layout_height="0dp"
                  android:layout_marginTop="15dp"
                  android:layout_weight="50"
                  android:background="@drawable/roundcorner"
                  android:backgroundTint="#DAF7A6" />
```

Un ListView permet de contenir plusieurs TextView pour générer une liste déroulante.

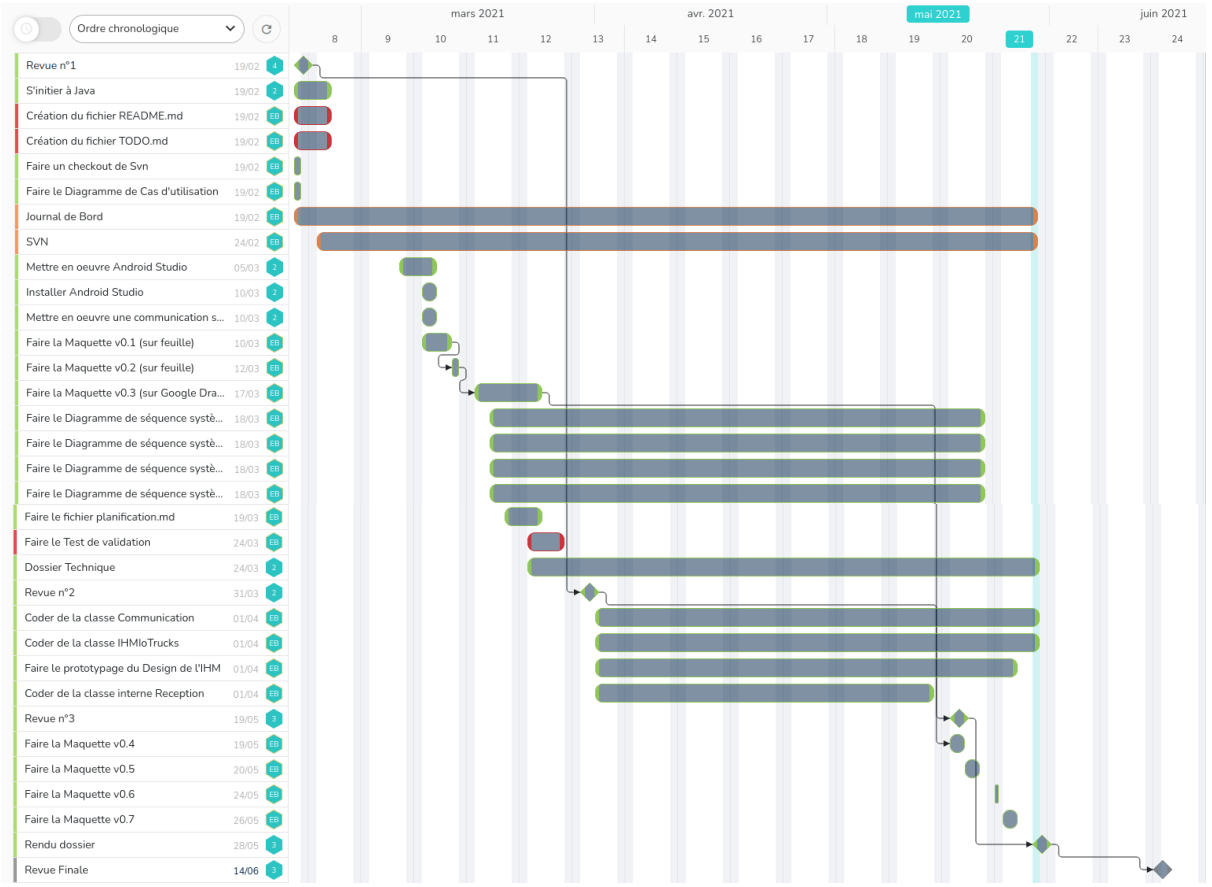
TextView

```
<!-- Triangle -->

            <TextView
                  android:id="@+id/textTriangle"
                  android:layout_marginTop="15dp"
                  android:layout_width="match_parent"
                  android:layout_height="0dp"
                  android:layout_weight="4"
                  android:text="Triangle"
                  android:textColor="#000"
                  app:autoSizeTextType="uniform" />
```

Un TextView permet d’insérer du texte dans le layout.

# Planification

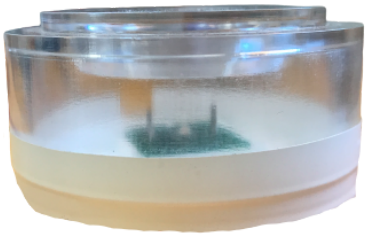


# Plan de test de validation

| Désignation | Démarche à suivre | Résultat attendu | Oui / Non | Remarques |
|---|---|---|---|---|
| Activation du Bluetooth | - Appuyer sur le bouton “Activer” le Bluetooth | Bluetooth est activé (L'icône du Bluetooth est affichée normalement) | Oui | Bluetooth est activé |
| Désactivation du Bluetooth | - Appuyer sur le bouton “Désactiver” le Bluetooth | Bluetooth est désactivé (L'icône du Bluetooth est affichée “barré”) | Oui | Bluetooth est désactivé |
| Déplacement du triangle vers le haut | - Appuyer sur le bouton “Lever” le triangle | Le triangle est levé | Non | Voir le tags 1.1 |
| Déplacement du triangle vers le bas | - Appuyer sur le bouton “Baisser” le triangle | Le triangle est baissé | Non | Voir le tags 1.1 |
| Activation du mode d’intervention | - Appuyer sur le bouton “Lever” le triangle<br>- Appuyer sur le bouton “Activer” du mode d’intervention | Le mode d’intervention est activé (le clignotement des LED du triangle son activé) | Non | Voir le tags 1.1 |
| Désactivation du mode d’intervention | - Appuyer sur le bouton “Désactiver” du mode d’intervention | Le mode d’intervention est désactivé | Non | Voir le tags 1.1 |
| Activation de l’éclairage | - Appuyer sur le bouton “Allumer” l’éclairage | L’éclairage est allumé | Non | Voir le tags 1.1 |
| Désactivation de l’éclairage | - Appuyer sur le bouton "Éteindre" l’éclairage | L’éclairage est éteint | Non | Voir le tags 1.1 |
| Déplacement du hayon vers le haut | - Appuyer sur le bouton “Lever” le hayon | Le hayon est levé | Non | Voir le tags 1.1 |
| Déplacement du hayon vers le bas | - Appuyer sur le bouton “Baisser” le hayon | Le hayon est baissé | Non | Voir le tags 1.1 |

# Annexes

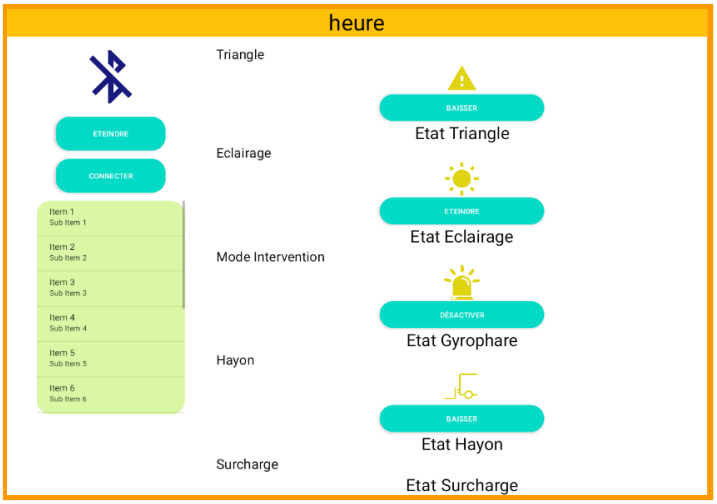
*LED :
Une diode électroluminescente est un dispositif opto-électronique capable d’émettre de la lumière lorsqu’il est parcouru par un courant électrique. Une diode électroluminescente ne laisse passer le courant électrique que dans un seul sens et produit un rayonnement monochromatique ou polychromatique incohérent par conversion d’énergie électrique lorsqu'un courant la traverse.



*Microcontrôleur :
Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (ROM et RAM), unités périphériques et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

*Bluetooth :
Un Bluetooth est une norme de communication permettant l'échange bidirectionnel de données à courte distance en utilisant des ondes radio UHF* sur la bande de fréquence de 2,4 GHz. Son but est de simplifier les connexions entre les appareils électroniques à proximité en supprimant des liaisons filaires. Elle peut remplacer par exemple les câbles entre ordinateurs, tablettes, haut-parleurs, téléphones mobiles entre eux ou avec des imprimantes, scanners, claviers, souris, manettes de jeu vidéo, téléphones portables, assistants personnels, systèmes avec mains libres pour microphones ou écouteurs, autoradios, appareils photo numériques, lecteurs de code-barres et bornes publicitaires interactives. Le Bluetooth 4.1 est sorti en décembre 2013, Bluetooth classique : peu ou pas de changement, Bluetooth LE : connexion de plusieurs appareils sur un seul accès maitre pour la sortie des smartphones LTE*.

*IHM :
L'interaction humain-machine appelé IHM, s'intéresse à la conception et au développement de systèmes interactifs en prenant en compte ses impacts sociétaux et éthiques. Les humains interagissent avec les ordinateurs qui les entourent et cette interaction nécessite des interfaces qui facilitent la communication entre l'humain et la machine. La facilitation de l'utilisation de dispositifs devient de plus en plus importante avec le nombre croissant d'interfaces numériques dans la vie quotidienne. L'IHM a pour but de trouver les moyens les plus efficaces, les plus accessibles et les plus intuitifs pour les utilisateurs de compléter une tâche le plus rapidement et le plus précisément possible.



*Microprocesseur :
Un microprocesseur est un processeur dont tous les composants ont été suffisamment miniaturisés pour être regroupés dans un unique boîtier. Fonctionnellement, le processeur est la partie d'un ordinateur qui exécute les instructions et traite les données des programmes.

*Mémoire flash :
La mémoire flash est une mémoire de masse à semi-conducteurs ré-inscriptible, c'est-à-dire une mémoire possédant les caractéristiques d'une mémoire vive mais dont les données ne disparaissent pas lors d'une mise hors tension. La mémoire flash stocke dans des cellules de mémoire les bits de données qui sont conservées lorsque l'alimentation électrique est coupée.

*Mémoire ROM :
Originellement, l'expression mémoire morte (Read-Only Memory : ROM) désignait une mémoire informatique non volatile et dont le contenu est fixé lors de sa programmation, qui pouvait être lue plusieurs fois par l'utilisateur, mais ne pouvait plus être modifiée. Ces mémoires sont les UVPROM, les PROM, les EPROM et les EEPROM.

*Mémoire SRAM :
La mémoire vive statique ou SRAM (Static Random Access Memory) est un type de mémoire vive utilisant des bascules pour mémoriser les données. Mais contrairement à la mémoire dynamique, elle n'a pas besoin de rafraîchir périodiquement son contenu. Comme la mémoire dynamique, elle est volatile : elle ne peut se passer d'alimentation sous peine de voir les informations effacées irrémédiablement.

*Bluetooth BR/EDR :
Le Bluetooth BR/EDR (Basic Rate/Enhanced Data Rate : taux de base/débit de données amélioré) permet une connexion sans fil continue et utilise une topologie de réseau point-à-point (P2P) pour établir des communications un à un (1:1) entre différents dispositifs. Le streaming audio BR/EDR est largement utilisé dans les enceintes et casques sans fil et les dispositifs mains libres des systèmes embarqués.

*CPU :
Un processeur ou CPU (Central Processing Unit) est un composant présent dans de nombreux dispositifs électroniques qui exécute les instructions machine des programmes informatiques. Avec la mémoire, c'est notamment l'une des fonctions qui existent depuis les premiers ordinateurs. Un processeur construit en un seul circuit intégré est un microprocesseur.

*GPU :
Un processeur graphique ou GPU (Graphics Processing Unit), également appelé coprocesseur graphique sur certains systèmes est une unité de calcul pouvant être présent sous forme de circuit intégré ou puce sur une carte graphique ou carte mère, ou encore intégré au même circuit intégré que le microprocesseur général (processeur graphique) et APU est d'assurer les fonctions de calcul d'image et afficher à l'écran ou à écrire sur mémoire de masse. Un processeur graphique a généralement une structure hautement parallèle qui le rend efficace pour une large palette de tâches graphiques comme le rendu 3D, en Direct3D ou en OpenGL, la gestion de la mémoire vidéo, le traitement du signal vidéo, la décompression Mpeg, etc.

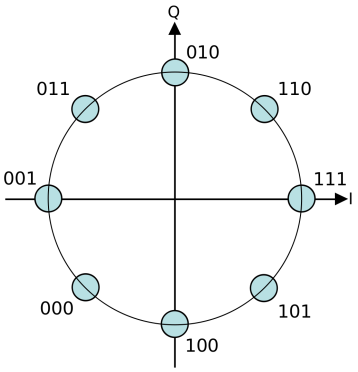
*Mémoire RAM :
La mémoire vive, parfois abrégée avec l'acronyme anglais RAM (Random Access Memory) est la mémoire informatique dans laquelle peuvent être enregistrées les informations traitées par un appareil informatique. On écrit mémoire vive par opposition à la mémoire morte.

*RF :
La radiofréquence (RF) est une technologie de transmission de données basée sur les ondes radio électromagnétiques. L’avantage de la technologie RF réside dans le fait que cette technologie offre une plage de signal plus large, pouvant aller jusqu’à 30 mètres. Les RF peuvent traverser les murs et il n’est pas nécessaire de diriger la télécommande vers l’appareil, car elle n’a pas besoin d’être en visibilité directe. Le Bluetooth qui est un type de RF est une norme de technologie sans fil ouverte pour la transmission de données sur de courtes distances. Il utilise des ondes radio sur une fréquence particulière pour la transmission de données d'un appareil à l'autre. Une variété d'appareils numériques utilisent la technologie Bluetooth, notamment les lecteurs MP3, les smartphones et périphériques, les ordinateurs portables, etc.

*Bluetooth LE :
Bluetooth à basse consommation ou Bluetooth à basse énergie anciennement connu sous le nom de Wibree, puis devenu la marque déposée Bluetooth Smart. C'est une technique de transmission sans fil créée par Nokia en 2006 sous la forme d'un standard ouvert basé sur le Bluetooth, qu'il complète mais sans le remplacer. Il est intégré aux normes Bluetooth depuis la version v4.0 publiée en juin 2010 par le Bluetooth SIG. Comparé au Bluetooth, le BLE permet un débit du même ordre de grandeur (1 Mbit/s) pour une consommation d'énergie 10 fois moindre. Cela permet d'intégrer cette technologie dans de nouveaux types d'équipements tels que montres, appareils de surveillance médicale ou capteurs pour sportifs. La technologie permet aux appareils de se connecter dans un rayon d'environ 10 mètres. Les modes BLE (bande passante plus limitée et très faible consommation) et Bluetooth standard (niveau d'émission plus élevé et portée plus grande) sont donc des technologies complémentaires.

Bluetooth SMART

*PSK :
Le PSK (Phase Shift Keying) désigne une famille de formes de modulations numériques qui ont toutes pour principe de véhiculer de l'information binaire via la phase d'un signal de référence dit porteuse et exclusivement par ce biais. Comme pour toute technique de modulation numérique, la phase en question ne peut prendre qu'un nombre fini de valeurs. Chacune de ces valeurs représente un unique nombre binaire dont la taille (la quantité d'information transmise) dépend du nombre de valeurs possibles pour la phase. Généralement, pour une modulation PSK donnée, les nombres binaires représentés sont tous de même taille.




*Onde radio UHF :
La bande des ultra hautes fréquences ou UHF (Ultra High Frequency) est la bande du spectre radioélectrique comprise entre 300 MHz et 3 000 MHz 1, soit les longueurs d'onde de 1 m à 0,1 m.

*Smartphones LTE :
Le LTE (Long Term Evolution) est une évolution des normes de téléphonie mobile GSM/EDGE, CDMA2000, TD-SCDMA et UMTS.