Io-Trucks

version 0.2

BTS SNIR LaSalle Avignon 2020

Table des matières

1 Le projet

Le projet **Io-Trucks** est un système numérique intégré à un véhicule industriel qui permet d'interagir avec ses accessoires et de visualiser les informations associées sur une application mobile.

1.1 Table des matières

1.2 Informations

Auteur

Arthur Mathieu mathieu.arthur.pro@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/io-trucks/

2 Changelog

r1 | www-data | 2020-02-01 15 :03 :29 +0100 (sam. 01 févr. 2020) | 1 ligne

Creating initial repository structure

3 README

3.1 Projet

3.1.1 Présentation

Le projet **Io-Trucks** est un système numérique intégré à un véhicule industriel qui permet :

— d'interagir avec ses accessoires et,
— de visualiser les informations associées sur une application mobile

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

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

3.1.2 Informations

Auteur

Arthur Mathieu mathieu.arthur.pro@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/io-trucks/

4 A propos

Auteur

Arthur Mathieu mathieu.arthur.pro@gmail.com

Date

2020

Version

0.2

Voir également

https://svn.riouxsvn.com/io-trucks/

5 Licence GPL

6 Liste des choses à faire

Membre com.lasalle.io_trucks.MainActivity.traiterTrame (String[] trame)

Traiter les autres types de trames

7 Documentation des espaces de nommage

7.1 Paquetage com

Paquetages

— package lasalle

7.2 Paquetage com.lasalle

Paquetages

— package io_trucks

7.3 Paquetage com.lasalle.io_trucks

Classes

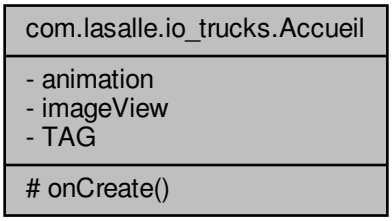
— class Accueil
Classe de l'activité Accueil La classe Acceuil est l'activité de démarrage de l'application.
— class Communication
Classe de Communication et de connexion bluetooth.
— class MainActivity
Classe IHM principale.
— class Peripherique
Classe permettant de gérer les périphériques.
— class Protocole

8 Documentation des classes

8.1 Référence de la classe com.lasalle.io_trucks.Accueil

Classe de l'activité Accueil La classe Acceuil est l'activité de démarrage de l'application.

Graphe de collaboration de com.lasalle.io_trucks.Accueil :



Fonctions membres protégées

— void onCreate (Bundle savedInstanceState)
Méthode appelée à la création de l'activité Accueil.

Attributs privés

— Animation animation
— ImageView imageView

Attributs privés statiques

— static final String TAG = "IHMAccueil"

8.1.1 Description détaillée

Classe de l'activité Accueil La classe Acceuil est l'activité de démarrage de l'application.

Définition à la ligne 26 du fichier Accueil.java.

8.1.2 Documentation des fonctions membres

8.1.2.1 onCreate()

```
void com.lasalle.io_trucks.Accueil.onCreate (
            Bundle savedInstanceState )  [protected]
```

Méthode appelée à la création de l'activité Accueil.

Paramètres

savedInstanceState	

Méthode perméttant la gestion du démarrage de l'animation Cette méthode définit ce qui se passe au démarrage de l'animation

Paramètres

animation	Animation vers l'activité suivante

Méthode perméttant la gestion de la fin de l'aniamtion Cette méthode définit ce qui se passe à la fin de l'animation Ici, elle démarre l'activité : MainActivity

Paramètres

animation	Animation vers l'activité suivante

Définition à la ligne 43 du fichier Accueil.java.

```
00044     {
00045         super.onCreate(savedInstanceState);
00046         setContentView(R.layout.activity_accueil);
00047         Log.i(TAG,"onCreate()");
00048 
00049         imageView = (ImageView)findViewById(R.id.imageView);
00050         animation = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.fade_in);
00051         animation.setAnimationListener(new Animation.AnimationListener()
00052         {
00058             @Override
00059             public void onAnimationStart(Animation animation)
00060             {
00061             }
00062 
00069             @Override
00070             public void onAnimationEnd(Animation animation)
00071             {
00072                 // A la fin de l'animation, on lance l'activité principale
00073                 Intent intent = new Intent(Accueil.this, MainActivity.class);
00074                 startActivity(intent);
00075             }
00076 
00077             @Override
00078             public void onAnimationRepeat(Animation animation)
00079             {
00080             }
00081         });
00082         imageView.startAnimation(animation);
00083     }
```

8.1.3 Documentation des données membres

8.1.3.1 animation

```
Animation com.lasalle.io_trucks.Accueil.animation  [private]
```

Attributs

Définition à la ligne 35 du fichier Accueil.java.

8.1.3.2 imageView

```
ImageView com.lasalle.io_trucks.Accueil.imageView  [private]
```

Définition à la ligne 36 du fichier Accueil.java.

8.1.3.3 TAG

```
final String com.lasalle.io_trucks.Accueil.TAG = "IHMAccueil"  [static], [private]
```

Constantes

Définition à la ligne 31 du fichier Accueil.java.

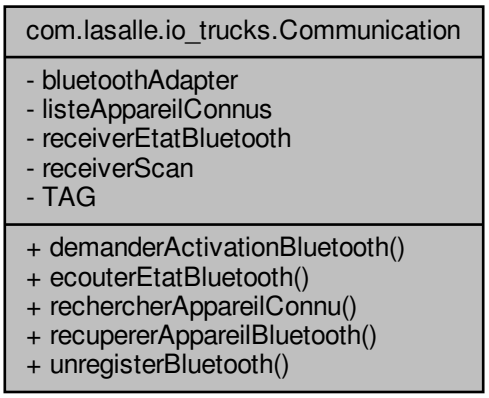
La documentation de cette classe a été générée à partir du fichier suivant :

— Accueil.java

8.2 Référence de la classe com.lasalle.io_trucks.Communication

Classe de Communication et de connexion bluetooth.

Graphe de collaboration de com.lasalle.io_trucks.Communication :



Fonctions membres publiques

- void demanderActivationBluetooth (Context contextAcceuil)
  Vérifie si le bluetooth est disponible et activé, sinon demande l'autorisation de l'activé
- BroadcastReceiver ecouterEtatBluetooth ()
  Vérifie les modification d'état du bluetooth.
- void rechercherAppareilConnu (Context contextAcceuil)
  Méthode de recherche des appareils qui ont déjà était appairer.
- BluetoothDevice recupererAppareilBluetooth (String nomAppareil)
  Méthode qui retourne l'appareil Bluetooth io-trucks.
- void unregisterBluetooth (Context contextAcceuil)
  Méthode pour unregister les receiver à la destruction de l'application.

Attributs privés

- BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()
- Set< BluetoothDevice > listeAppareilConnus
- BroadcastReceiver receiverEtatBluetooth
- BroadcastReceiver receiverScan

Attributs privés statiques

- static final String TAG = "Communication"

8.2.1 Description détaillée

Classe de Communication et de connexion bluetooth.

Définition à la ligne 25 du fichier Communication.java.

8.2.2 Documentation des fonctions membres

8.2.2.1 demanderActivationBluetooth()

```
void com.lasalle.io_trucks.Communication.demanderActivationBluetooth (
            Context contextAcceuil )
```

Vérifie si le bluetooth est disponible et activé, sinon demande l'autorisation de l'activé

Paramètres

contextAcceuil	contexte de l'acceuil

Définition à la ligne 43 du fichier Communication.java.

Référencé par com.lasalle.io_trucks.MainActivity.onCreate().

```
00044     {
00045         if (bluetoothAdapter == null)
00046         {
00047             Log.i(TAG,"demanderActivationBluetooth() bluetoothAdapter = null");
00048             Toast.makeText(contextAcceuil, R.string.str_bluetoot_inexistant, Toast.LENGTH_SHORT).show();
00049         }
00050         else if (!bluetoothAdapter.isEnabled())
00051         {
00052             Log.i(TAG,"demanderActivationBluetooth() bluetooth désactivé");
00053             Toast.makeText(contextAcceuil, R.string.str_bluetooth_eteint, Toast.LENGTH_SHORT).show();
00054             Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
00055             ((Activity) contextAcceuil).startActivityForResult(enableBtIntent,1);
00056         }
00057         else
00058         {
00059             Log.i(TAG,"demanderActivationBluetooth() bluetooth activé");
00060             Toast.makeText(contextAcceuil, R.string.str_bluetooth_allumer, Toast.LENGTH_SHORT).show();
00061         }
00062     }
```

8.2.2.2 ecouterEtatBluetooth()

```
BroadcastReceiver com.lasalle.io_trucks.Communication.ecouterEtatBluetooth ( )
```

Vérifie les modification d'état du bluetooth.

Renvoie

renvoie le Receiver de l'état du bluetooth

Définition à la ligne 68 du fichier Communication.java.

Références com.lasalle.io_trucks.Communication.receiverEtatBluetooth.

Référencé par com.lasalle.io_trucks.MainActivity.creerLiasonReceiverEtatBluetooth().

```
00069     {
00070         receiverEtatBluetooth = new BroadcastReceiver()
00071         {
00072             @Override
00073             public void onReceive(Context context, Intent intent)
00074             {
00075                 final String action = intent.getAction();
00076                 Log.i(TAG,"onReceive() " + action);
00077                 if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED))
00078                 {
00079                     final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, BluetoothAdapter.
      ERROR);
00080                     switch (state)
00081                     {
00082                         case BluetoothAdapter.STATE_OFF:
00083                             Log.i(TAG,"onReceive() Bluetooth désactivé !");
00084                             Toast.makeText(context, R.string.str_bluetooth_eteint, Toast.LENGTH_SHORT).show
      ();
00085                             break;
00086                         case BluetoothAdapter.STATE_TURNING_OFF:
00087                             Toast.makeText(context, R.string.str_bluetooth_eteint_en_cours, Toast.
      LENGTH_LONG).show();
00088                             break;
00089                         case BluetoothAdapter.STATE_ON:
00090                             Log.i(TAG,"onReceive() Bluetooth activé !");
00091                             Toast.makeText(context, R.string.str_bluetooth_allumer, Toast.LENGTH_SHORT).
      show();
00092                             break;
00093                         case BluetoothAdapter.STATE_TURNING_ON:
00094                             Log.i(TAG,"onReceive() Bluetooth s'active !");
00095                             Toast.makeText(context, R.string.str_bluetooth_allumer_en_cours, Toast.
      LENGTH_LONG).show();
00096                             break;
00097                     }
00098                 }
00099             }
00100         };
00101
00102         return receiverEtatBluetooth;
00103     }
```

8.2.2.3 rechercherAppareilConnu()

```
void com.lasalle.io_trucks.Communication.rechercherAppareilConnu (
             Context contextAcceuil )
```

Méthode de recherche des appareils qui ont déjà était appairer.

Paramètres

contextAcceuil	Contient le contexte de l'activité Acceuil

Définition à la ligne 109 du fichier Communication.java.

Référencé par com.lasalle.io_trucks.MainActivity.onClick().

```
00110     {
00111         listeAppareilConnus = bluetoothAdapter.getBondedDevices();
00112         for(BluetoothDevice blueDevice : listeAppareilConnus)
00113         {
00114             Toast.makeText(contextAcceuil, "Appareil " + blueDevice.getName(), Toast.LENGTH_SHORT).show();
00115         }
00116     }
```

8.2.2.4 recupererAppareilBluetooth()

```
BluetoothDevice com.lasalle.io_trucks.Communication.recupererAppareilBluetooth (
             String nomAppareil )
```

Méthode qui retourne l'appareil Bluetooth io-trucks.

Paramètres

nomAppareil	le nom de l'appareil Bluetooth io-trucks

Renvoie

Renvoie le device utilisé pour la communication

Définition à la ligne 123 du fichier Communication.java.

Référencé par com.lasalle.io_trucks.MainActivity.onClick().

```
00124     {
00125         listeAppareilConnus = bluetoothAdapter.getBondedDevices();
00126         for(BluetoothDevice blueDevice : listeAppareilConnus)
00127         {
00128             if(blueDevice.getName().equals(nomAppareil))
00129             {
00130                 Log.d(TAG, "recupererAppareilBluetooth() io-trucks trouvé : " + blueDevice.getName() + "
       (" + blueDevice.getAddress() + ")");
00131                 return blueDevice;
00132             }
00133         }
00134         return null;
00135     }
```

8.2.2.5 unregisterBluetooth()

```
void com.lasalle.io_trucks.Communication.unregisterBluetooth (
             Context contextAcceuil )
```

Méthode pour unregister les receiver à la destruction de l'application.

Paramètres

contextAcceuil	contexte de l'acceuil

Définition à la ligne 141 du fichier Communication.java.

Référencé par com.lasalle.io_trucks.MainActivity.onPause().

```
00142     {
00143         if (bluetoothAdapter != null)
00144         {
00145             bluetoothAdapter.cancelDiscovery();
00146             contextAcceuil.unregisterReceiver(receiverEtatBluetooth);
00147         }
00148     }
```

8.2.3 Documentation des données membres

8.2.3.1 bluetoothAdapter

```
BluetoothAdapter com.lasalle.io_trucks.Communication.bluetoothAdapter = BluetoothAdapter.getDefaultAdapter()
[private]
```

Définition à la ligne 36 du fichier Communication.java.

8.2.3.2 listeAppareilConnus

```
Set<BluetoothDevice> com.lasalle.io_trucks.Communication.listeAppareilConnus  [private]
```

Définition à la ligne 37 du fichier Communication.java.

8.2.3.3 receiverEtatBluetooth

```
BroadcastReceiver com.lasalle.io_trucks.Communication.receiverEtatBluetooth  [private]
```

Attributs

Définition à la ligne 34 du fichier Communication.java.

Référencé par com.lasalle.io_trucks.Communication.ecouterEtatBluetooth().

8.2.3.4 receiverScan

```
BroadcastReceiver com.lasalle.io_trucks.Communication.receiverScan  [private]
```

Définition à la ligne 35 du fichier Communication.java.

8.2.3.5 TAG

```
final String com.lasalle.io_trucks.Communication.TAG = "Communication"  [static], [private]
```

Constantes

Définition à la ligne 30 du fichier Communication.java.

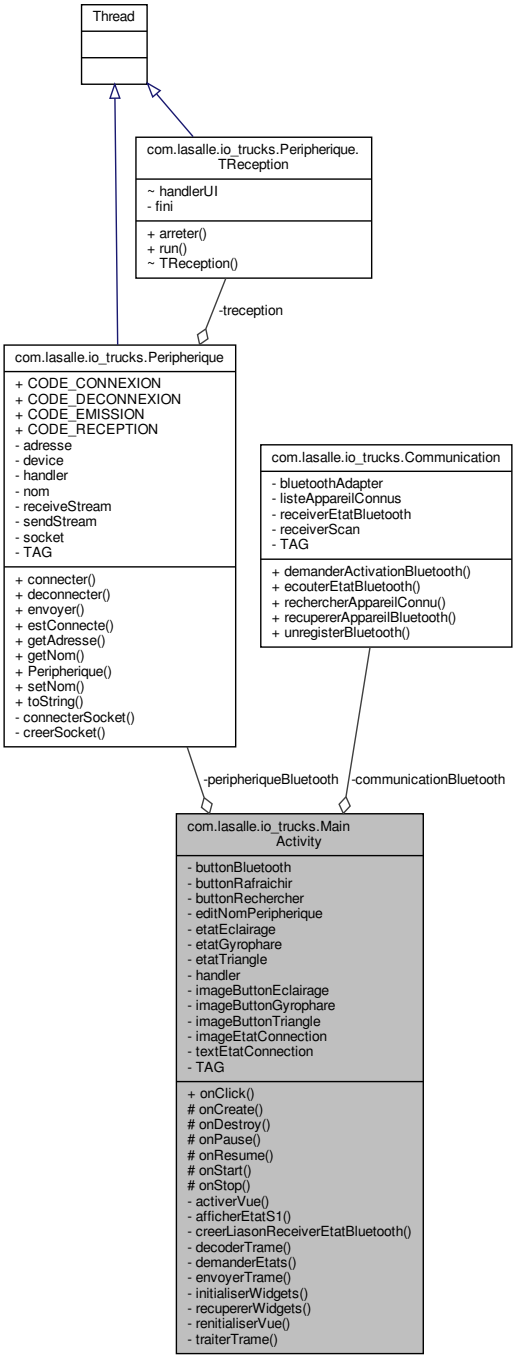
La documentation de cette classe a été générée à partir du fichier suivant :

— Communication.java

8.3 Référence de la classe com.lasalle.io_trucks.MainActivity

Classe IHM principale.

Graphe de collaboration de com.lasalle.io_trucks.MainActivity :



Fonctions membres publiques

— void onClick (View element)
Méthode de gestion des clics Ceci est la méthode qui gère l'écoute des clics sur les différents widgets de l'interface.

Fonctions membres protégées

- void onCreate (Bundle savedInstanceState)
  Méthode appelée à la création de l'activité MainActivity.
- void onDestroy ()
  Méthode appelée à la destruction de l'application (après onStop() et détruite par le système Android)
- void onPause ()
  Méthode appelée après qu'une boîte de dialogue s'est affichée (on reprend sur un onResume()) ou avant onStop() (activité plus visible)
- void onResume ()
  Méthode appelée après onStart() ou après onPause()
- void onStart ()
  Méthode appelée au démarrage après le onCreate() ou un restart après un onStop()
- void onStop ()
  Méthode appelée lorsque l'activité n'est plus visible.

Fonctions membres privées

- void activerVue ()
- void afficherEtatS1 (String[ ] trame)
- void creerLiasonReceiverEtatBluetooth ()
  Méthode pour créer les Registers de l'état du bluetooth et donc le lien avec l'état du bluetooth.
- void decoderTrame (String trame)
  Méthode permettant de décoder les trames reçues.
- void demanderEtats ()
- void envoyerTrame (String trame)
  Méthode qui envoie une trame au périphérique Bluetooth.
- void initialiserWidgets ()
  Méthode pour initialiser les Widgets.
- void recupererWidgets ()
  Méthode pour associer la vue à l'objet des Widgets.
- void renitialiserVue ()
  Méthode permettant de rénitialiser la vue de l'activitée.
- void traiterTrame (String[ ] trame)
  Méthode permettant de traiter les trames en fonctions de leurs contennue.

Attributs privés

- Button buttonBluetooth
- Button buttonRafraichir
- Button buttonRechercher
- Communication communicationBluetooth = new Communication()
- EditText editNomPeripherique
- Boolean etatEclairage = false
- Boolean etatGyrophare = false
- Boolean etatTriangle = false
- final Handler handler
  Handler de l'application et des périphériques bluetooth Cette handler permet de gérer le thread de communication de l'application.
- ImageButton imageButtonEclairage
- ImageButton imageButtonGyrophare
- ImageButton imageButtonTriangle
- ImageView imageEtatConnection
- Peripherique peripheriqueBluetooth = null
- TextView textEtatConnection

Attributs privés statiques

- static final String TAG = "IHMMainActivity"

8.3.1 Description détaillée

Classe IHM principale.

Définition à la ligne 32 du fichier MainActivity.java.

8.3.2 Documentation des fonctions membres

8.3.2.1 activerVue()

```
void com.lasalle.io_trucks.MainActivity.activerVue ( )  [private]
```

Définition à la ligne 296 du fichier MainActivity.java.

```
00297     {
00298         buttonBluetooth.setText("Déconnecter");
00299         buttonRafraichir.setEnabled(true);
00300         imageButtonTriangle.setEnabled(true);
00301         imageButtonGyrophare.setEnabled(true);
00302         imageButtonEclairage.setEnabled(true);
00303         imageEtatConnection.setImageResource(R.drawable.green_cricle);
00304         textEtatConnection.setText("Connecté");
00305     }
```

8.3.2.2 afficherEtatS1()

```
void com.lasalle.io_trucks.MainActivity.afficherEtatS1 (
             String [] trame )  [private]
```

Définition à la ligne 363 du fichier MainActivity.java.

Références com.lasalle.io_trucks.Protocole.LEVE, et com.lasalle.io_trucks.Protocole.ON.

Référencé par com.lasalle.io_trucks.MainActivity.traiterTrame().

```
00364     {
00365         Log.v(TAG, "traiterTrame() trame[2] = " + trame[2] + " (triangle)");
00366         if(trame[2].equals(Protocole.LEVE))
00367         {
00368             imageButtonTriangle.setImageResource(R.drawable.triangle);
00369             etatTriangle = true;
00370         }
00371         else
00372         {
00373             imageButtonTriangle.setImageResource(R.drawable.triangle_b_w);
00374             etatTriangle = false;
00375         }
00376 
00377         Log.v(TAG, "traiterTrame() trame[3] = " + trame[3] + " (gyrophare)");
00378         if(trame[3].equals(Protocole.ON))
00379         {
00380             imageButtonGyrophare.setImageResource(R.drawable.flash);
00381             etatGyrophare = true;
00382         }
00383         else
00384         {
00385             imageButtonGyrophare.setImageResource(R.drawable.flash_b_w);
00386             etatGyrophare = false;
00387         }
00388 
00389         Log.v(TAG, "traiterTrame() trame[4] = " + trame[4] + " (éclairage)");
00390         if(trame[4].equals(Protocole.ON))
00391         {
00392             imageButtonEclairage.setImageResource(R.drawable.spotlight);
00393             etatEclairage = true;
00394         }
00395         else
00396         {
00397             imageButtonEclairage.setImageResource(R.drawable.spotlight_b_w);
00398             etatEclairage = false;
00399         }
00400     }
```

8.3.2.3 creerLiasonReceiverEtatBluetooth()

```
void com.lasalle.io_trucks.MainActivity.creerLiasonReceiverEtatBluetooth ( )  [private]
```

Méthode pour créer les Registers de l'état du bluetooth et donc le lien avec l'état du bluetooth.

Définition à la ligne 290 du fichier MainActivity.java.

Références com.lasalle.io_trucks.Communication.ecouterEtatBluetooth().

Référencé par com.lasalle.io_trucks.MainActivity.onResume().

```
00291     {
00292         IntentFilter filter = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
00293         registerReceiver(communicationBluetooth.
      ecouterEtatBluetooth(), filter);
00294     }
```

8.3.2.4 decoderTrame()

```
void com.lasalle.io_trucks.MainActivity.decoderTrame (
             String trame )  [private]
```

Méthode perméttant de décoder les trames reçues.

Paramètres

trame	Contient la trame reçue

Définition à la ligne 328 du fichier MainActivity.java.

Références com.lasalle.io_trucks.Protocole.DELIMITEUR_CHAMP, com.lasalle.io_trucks.Protocole.DELIMITEUR_FIN, com.lasalle.io_trucks.Protocole.EN_TETE, et com.lasalle.io_trucks.MainActivity.traiterTrame().

```
00329     {
00330         String nouvelleTrame = "";
00331         // Exemple : trame = "$iotruck;S1;0;0;0\r\n"
00332         nouvelleTrame = trame.replace(Protocole.EN_TETE,""); // enlever aussi le ; ?
00333         // Exemple : nouvelleTrame = ";S1;0;0;0\r\n"
00334         nouvelleTrame.replace(Protocole.DELIMITEUR_FIN,"");
00335         // Exemple : nouvelleTrame = ";S1;0;0;0"
00336         String[] trameCouper = nouvelleTrame.split(Protocole.DELIMITEUR_CHAMP);
00337         // Exemple : trameCouper = [0];[1];[2];[3];[4]
00338         Log.v(TAG, "decoderTrame() découpage de la trame");
00339         // le premier champ est vide
00340         for(int i = 1; i < trameCouper.length; i++)
00341         {
00342             Log.v(TAG, "decoderTrame() champ " + i + " = " + trameCouper[i]);
00343         }
00344         traiterTrame(trameCouper);
00345     }
```

8.3.2.5 demanderEtats()

```
void com.lasalle.io_trucks.MainActivity.demanderEtats ( )  [private]
```

Méthode qui envoie les trames de demande d'états S1 et S2

Définition à la ligne 441 du fichier MainActivity.java.

Références com.lasalle.io_trucks.MainActivity.envoyerTrame().

Référencé par com.lasalle.io_trucks.MainActivity.onClick().

```
00442     {
00443         envoyerTrame("$iotruck;S1\r\n");
00444         envoyerTrame("$iotruck;S2\r\n");
00445     }
```

8.3.2.6 envoyerTrame()

```
void com.lasalle.io_trucks.MainActivity.envoyerTrame (
             String trame )  [private]
```

Méthode qui envoie une trame au périphérique Bluetooth.

Définition à la ligne 244 du fichier MainActivity.java.

Références com.lasalle.io_trucks.Peripherique.envoyer(), et com.lasalle.io_trucks.Peripherique.estConnecte().

Référencé par com.lasalle.io_trucks.MainActivity.demanderEtats(), et com.lasalle.io_trucks.MainActivity.onClick().

```
00245     {
00246         if(peripheriqueBluetooth != null)
00247         {
00248             if (peripheriqueBluetooth.estConnecte())
00249             {
00250                 Log.i(TAG, "envoyerTrame() trame : " + trame);
00251                 peripheriqueBluetooth.envoyer(trame);
00252             }
00253         }
00254     }
```

8.3.2.7 initialiserWidgets()

```
void com.lasalle.io_trucks.MainActivity.initialiserWidgets ( )  [private]
```

Méthode pour initialiser les Widgets.

Définition à la ligne 275 du fichier MainActivity.java.

Références com.lasalle.io_trucks.MainActivity.renitialiserVue().

Référencé par com.lasalle.io_trucks.MainActivity.onCreate().

```
00276     {
00277         renitialiserVue();
00278
00279         buttonBluetooth.setOnClickListener(this);
00280         buttonRechercher.setOnClickListener(this);
00281         buttonRafraichir.setOnClickListener(this);
00282         imageButtonTriangle.setOnClickListener(this);
00283         imageButtonGyrophare.setOnClickListener(this);
00284         imageButtonEclairage.setOnClickListener(this);
00285     }
```

8.3.2.8 onClick()

```
void com.lasalle.io_trucks.MainActivity.onClick (
             View element )
```

Méthode de gestion des clics Ceci est la méthode qui gère l'écoute des clics sur les différents widgets de l'interface.

Paramètres

element	element définis le widget sur lequel le clic est recenssé

Définition à la ligne 133 du fichier MainActivity.java.

Références com.lasalle.io_trucks.Peripherique.connecter(), com.lasalle.io_trucks.Peripherique.deconnecter(), com.lasalle.io_trucks.MainActivity.demanderEtats(), com.lasalle.io_trucks.MainActivity.envoyerTrame(), com.lasalle.io_trucks.Peripherique.estConnecte(), com.lasalle.io_trucks.MainActivity.handler, com.lasalle.io_trucks.Communication.rechercherAppareilConnu(), et com.lasalle.io_trucks.Communication.recupererAppareilBluetooth().

```
00134     {
00135         if(element == buttonBluetooth)
00136         {
00137             if(buttonBluetooth.getText().equals("Connecter"))
00138             {
00139                 BluetoothDevice blueDevice = communicationBluetooth.
      recupererAppareilBluetooth(editNomPeripherique.getText().
      toString());
00140                 if(blueDevice == null)
00141                 {
00142                     AlertDialog.Builder boiteAvertissementNonTrouver = new AlertDialog.Builder(this);
00143                     boiteAvertissementNonTrouver.setMessage("L'appareil io-trucks n'as pas été trouvé.
       Vérifiez si celui a été appairé correctement.");
00144                     boiteAvertissementNonTrouver.setPositiveButton("Continuer", new DialogInterface.
      OnClickListener() {
00145                         @Override
00146                         public void onClick(DialogInterface dialog, int which) {
00147                         }
00148                     });
00149                     boiteAvertissementNonTrouver.show();
00150                     Log.i(TAG, "Appareil io-trucks non trouvé !");
00151                     return;
00152                 }
00153                 else
00154                 {
00155                     if(peripheriqueBluetooth == null)
00156                     {
00157                         Log.i(TAG, "Instancie peripheriqueBluetooth");
00158                         peripheriqueBluetooth = new Peripherique(blueDevice,
      handler);
00159                     }
00160                     if(!peripheriqueBluetooth.estConnecte())
00161                     {
00162                         Log.i(TAG, "Connexion peripheriqueBluetooth");
00163                         peripheriqueBluetooth.connecter();
00164                     }
00165                     else // déjà connecté !
00166                     {
00167                     }
00168                 }
00169             }
00170             else if(buttonBluetooth.getText().equals("Déconnecter"))
00171             {
00172                 if (peripheriqueBluetooth.estConnecte())
00173                 {
00174                     peripheriqueBluetooth.deconnecter();
00175                 }
00176             }
00177         }
00178         else if(element == buttonRechercher)
00179         {
00180             communicationBluetooth.rechercherAppareilConnu(
      this);
00181         }
00182         else if(element == imageButtonTriangle)
00183         {
00184             Log.i(TAG,"button Triangle");
00185             if(!etatTriangle)
00186             {
00187                 envoyerTrame("$iotruck;T;1\r\n");
00188                 imageButtonTriangle.setImageResource(R.drawable.triangle);
00189                 etatTriangle = true;
00190             }
00191             else
00192             {
00193                 envoyerTrame("$iotruck;T;0\r\n");
00194                 imageButtonTriangle.setImageResource(R.drawable.triangle_b_w);
00195                 etatTriangle = false;
00196             }
00197         }
00198         else if(element == imageButtonGyrophare)
00199         {
```

```
00200             Log.i(TAG,"button Gyrophare");
00201             if(!etatGyrophare)
00202             {
00203                 envoyerTrame("$iotruck;G;1\r\n");
00204                 imageButtonGyrophare.setImageResource(R.drawable.flash);
00205                 etatGyrophare = true;
00206             }
00207             else
00208             {
00209                 envoyerTrame("$iotruck;G;0\r\n");
00210                 imageButtonGyrophare.setImageResource(R.drawable.flash_b_w);
00211                 etatGyrophare = false;
00212             }
00213         }
00214         else if(element == imageButtonEclairage)
00215         {
00216             Log.i(TAG,"button Eclairage");
00217             if(!etatEclairage)
00218             {
00219                 envoyerTrame("$iotruck;E;1\r\n");
00220                 imageButtonEclairage.setImageResource(R.drawable.spotlight);
00221                 etatEclairage = true;
00222             }
00223             else
00224             {
00225                 envoyerTrame("$iotruck;E;0\r\n");
00226                 imageButtonEclairage.setImageResource(R.drawable.spotlight_b_w);
00227                 etatEclairage = false;
00228             }
00229         }
00230         else if(element == buttonRafraichir)
00231         {
00232             Log.i(TAG,"button Rafraichir");
00233             demanderEtats();
00234         }
00235         else
00236         {
00237             Log.i(TAG,"button Inconnu : " + element.getId());
00238         }
00239     }
```

8.3.2.9 onCreate()

```
void com.lasalle.io_trucks.MainActivity.onCreate (
             Bundle savedInstanceState )  [protected]
```

Méthode appelée à la création de l'activité MainActivity.

Paramètres

savedInstanceState	

Définition à la ligne 61 du fichier MainActivity.java.

Références com.lasalle.io_trucks.Communication.demanderActivationBluetooth(), com.lasalle.io_trucks.MainActivity.initialiserWidgets(), et com.lasalle.io_trucks.MainActivity.recupererWidgets().

```
00062     {
00063         super.onCreate(savedInstanceState);
00064         setContentView(R.layout.activity_main);
00065         Log.i(TAG,"onCreate()");
00066 
00067         recupererWidgets();
00068         initialiserWidgets();
00069 
00070         communicationBluetooth.demanderActivationBluetooth
      (this);
00071     }
```

8.3.2.10 onDestroy()

```
void com.lasalle.io_trucks.MainActivity.onDestroy ( )  [protected]
```

Méthode appelée à la destruction de l'application (après onStop() et détruite par le système Android)

Définition à la ligne 120 du fichier MainActivity.java.

```
00121     {
00122         super.onDestroy();
00123         Log.i(TAG,"onDestroy()");
00124 
00125     }
```

8.3.2.11 onPause()

```
void com.lasalle.io_trucks.MainActivity.onPause ( )  [protected]
```

Méthode appelée après qu'une boîte de dialogue s'est affichée (on reprend sur un onResume()) ou avant onStop() (activité plus visible)

Définition à la ligne 98 du fichier MainActivity.java.

Références com.lasalle.io_trucks.Communication.unregisterBluetooth().

```
00099     {
00100         super.onPause();
00101         Log.i(TAG,"onPause()");
00102         communicationBluetooth.unregisterBluetooth(this);
00103     }
```

8.3.2.12 onResume()

```
void com.lasalle.io_trucks.MainActivity.onResume ( )  [protected]
```

Méthode appelée après onStart() ou après onPause()

Définition à la ligne 87 du fichier MainActivity.java.

Références com.lasalle.io_trucks.MainActivity.creerLiasonReceiverEtatBluetooth().

```
00088     {
00089         super.onResume();
00090         Log.i(TAG,"onResume()");
00091         creerLiasonReceiverEtatBluetooth();
00092     }
```

8.3.2.13 onStart()

```
void com.lasalle.io_trucks.MainActivity.onStart ( )  [protected]
```

Méthode appelée au démarrage après le onCreate() ou un restart après un onStop()

Définition à la ligne 77 du fichier MainActivity.java.

```
00078     {
00079         super.onStart();
00080         Log.i(TAG,"onStart()");
00081     }
```

8.3.2.14 onStop()

```
void com.lasalle.io_trucks.MainActivity.onStop ( )  [protected]
```

Méthode appelée lorsque l'activité n'est plus visible.

Définition à la ligne 109 du fichier MainActivity.java.

```
00110     {
00111         super.onStop();
00112         Log.i(TAG,"onStop()");
00113 
00114     }
```

8.3.2.15 recupererWidgets()

```
void com.lasalle.io_trucks.MainActivity.recupererWidgets ( )  [private]
```

Méthode pour associer la vue à l'objet des Widgets.

Définition à la ligne 259 du fichier MainActivity.java.

Référencé par com.lasalle.io_trucks.MainActivity.onCreate().

```
00260     {
00261         buttonBluetooth = findViewById(R.id.buttonConnecter);
00262         buttonRechercher = findViewById(R.id.buttonBounded);
00263         buttonRafraichir = findViewById(R.id.buttonRafraichir);
00264         imageButtonTriangle = findViewById(R.id.imageButtonTriangle);
00265         imageButtonGyrophare = findViewById(R.id.imageButtonGyrophares);
00266         imageButtonEclairage = findViewById(R.id.imageButtonEclairage);
00267         imageEtatConnection = findViewById(R.id.imageViewEtatConnection);
00268         textEtatConnection = findViewById(R.id.textViewEtatConnection);
00269         editNomPeripherique = findViewById(R.id.editNomPeripherique);;
00270     }
```

8.3.2.16 renitialiserVue()

```
void com.lasalle.io_trucks.MainActivity.renitialiserVue ( )  [private]
```

Méthode permettant de rénitialiser la vue de l'activitée.

Définition à la ligne 310 du fichier MainActivity.java.

Référencé par com.lasalle.io_trucks.MainActivity.initialiserWidgets().

```
00311     {
00312         buttonBluetooth.setText("Connecter");
00313         buttonRafraichir.setEnabled(false);
00314         imageButtonTriangle.setEnabled(false);
00315         imageButtonGyrophare.setEnabled(false);
00316         imageButtonEclairage.setEnabled(false);
00317         imageButtonTriangle.setImageResource(R.drawable.triangle_b_w);
00318         imageButtonEclairage.setImageResource(R.drawable.spotlight_b_w);
00319         imageButtonGyrophare.setImageResource(R.drawable.flash_b_w);
00320         imageEtatConnection.setImageResource(R.drawable.red_circle);
00321         textEtatConnection.setText("Déconnecté");
00322     }
```

8.3.2.17 traiterTrame()

```
void com.lasalle.io_trucks.MainActivity.traiterTrame (
             String [] trame )  [private]
```

Méthode permettant de traiter les trames en fonctions de leurs contennue.

Paramètres

trame	contient la trame reçu après avoir était décoder et découper

A faire Traiter les autres types de trames

Définition à la ligne 351 du fichier MainActivity.java.

Références com.lasalle.io_trucks.MainActivity.afficherEtatS1(), et com.lasalle.io_trucks.Protocole.TRAME_REQUETE_STATE1.

Référencé par com.lasalle.io_trucks.MainActivity.decoderTrame().

```
00352     {
00353         Log.v(TAG, "traiterTrame() trame[1] = " + trame[1] + " (type)");
00354         if(trame[1].equals(Protocole.TRAME_REQUETE_STATE1))
00355         {
00356             afficherEtatS1(trame);
00357         }
00361     }
```

8.3.3 Documentation des données membres

8.3.3.1 buttonBluetooth

```
Button com.lasalle.io_trucks.MainActivity.buttonBluetooth  [private]
```

Définition à la ligne 44 du fichier MainActivity.java.

8.3.3.2 buttonRafraichir

```
Button com.lasalle.io_trucks.MainActivity.buttonRafraichir  [private]
```

Définition à la ligne 46 du fichier MainActivity.java.

8.3.3.3 buttonRechercher

```
Button com.lasalle.io_trucks.MainActivity.buttonRechercher  [private]
```

Définition à la ligne 45 du fichier MainActivity.java.

8.3.3.4 communicationBluetooth

```
Communication com.lasalle.io_trucks.MainActivity.communicationBluetooth = new Communication()  [private]
```

Définition à la ligne 50 du fichier MainActivity.java.

8.3.3.5 editNomPeripherique

```
EditText com.lasalle.io_trucks.MainActivity.editNomPeripherique  [private]
```

Définition à la ligne 54 du fichier MainActivity.java.

8.3.3.6 etatEclairage

```
Boolean com.lasalle.io_trucks.MainActivity.etatEclairage = false  [private]
```

Définition à la ligne 43 du fichier MainActivity.java.

8.3.3.7 etatGyrophare

```
Boolean com.lasalle.io_trucks.MainActivity.etatGyrophare = false  [private]
```

Définition à la ligne 42 du fichier MainActivity.java.

8.3.3.8 etatTriangle

```
Boolean com.lasalle.io_trucks.MainActivity.etatTriangle = false  [private]
```

Attributs

Définition à la ligne 41 du fichier MainActivity.java.

8.3.3.9 handler

```
final Handler com.lasalle.io_trucks.MainActivity.handler  [private]
```

Valeur initiale :

```
= new Handler()
    {
        @Override
        public void handleMessage(Message msg)
        {
            super.handleMessage(msg);

            switch(msg.what)
            {
                case Peripherique.CODE_CONNEXION:
                    Log.v(TAG, "handleMessage() io-trucks connecté");
                    activerVue();
                    demanderEtats();
                    break;
                case Peripherique.CODE_RECEPTION:
                    Log.v(TAG, "handleMessage() io-trucks réception : " + (String)msg.obj);
                    decoderTrame((String)msg.obj);
                    break;
                case Peripherique.CODE_EMISSION:
                    Log.v(TAG, "handleMessage() io-trucks émission : " + (String)msg.obj);

                    break;
                case Peripherique.CODE_DECONNEXION:
                    Log.v(TAG, "handleMessage() io-trucks déconnecté");
                    renitialiserVue();
                    break;
            }
        }
    }
```

Handler de l'application et des périphériques bluetooth Cette handler permet de gérer le thread de communication de l'application.

Définition à la ligne 406 du fichier MainActivity.java.

Référencé par com.lasalle.io_trucks.MainActivity.onClick().

8.3.3.10 imageButtonEclairage

```
ImageButton com.lasalle.io_trucks.MainActivity.imageButtonEclairage  [private]
```

Définition à la ligne 49 du fichier MainActivity.java.

8.3.3.11 imageButtonGyrophare

```
ImageButton com.lasalle.io_trucks.MainActivity.imageButtonGyrophare  [private]
```

Définition à la ligne 48 du fichier MainActivity.java.

8.3.3.12 imageButtonTriangle

```
ImageButton com.lasalle.io_trucks.MainActivity.imageButtonTriangle  [private]
```

Définition à la ligne 47 du fichier MainActivity.java.

8.3.3.13 imageEtatConnection

```
ImageView com.lasalle.io_trucks.MainActivity.imageEtatConnection  [private]
```

Définition à la ligne 52 du fichier MainActivity.java.

8.3.3.14 peripheriqueBluetooth

```
Peripherique com.lasalle.io_trucks.MainActivity.peripheriqueBluetooth = null  [private]
```

Définition à la ligne 51 du fichier MainActivity.java.

8.3.3.15 TAG

```
final String com.lasalle.io_trucks.MainActivity.TAG = "IHMMainActivity"  [static], [private]
```

Constantes

Définition à la ligne 37 du fichier MainActivity.java.

8.3.3.16 textEtatConnection

```
TextView com.lasalle.io_trucks.MainActivity.textEtatConnection  [private]
```

Définition à la ligne 53 du fichier MainActivity.java.

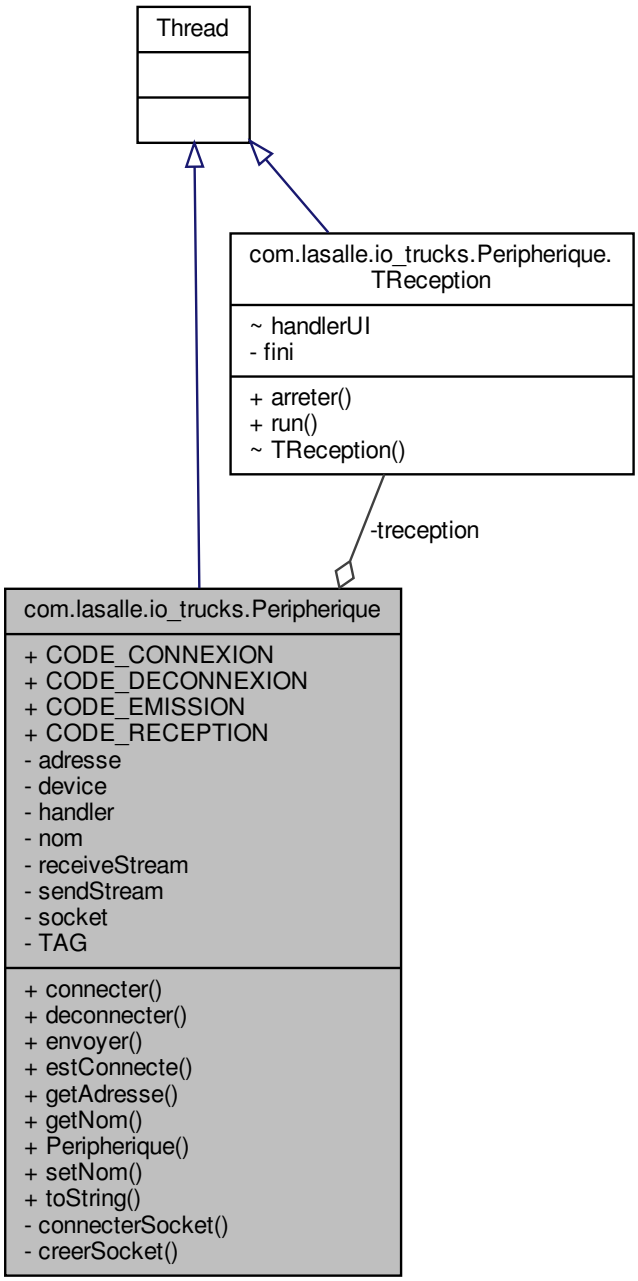
La documentation de cette classe a été générée à partir du fichier suivant :

— MainActivity.java

8.4 Référence de la classe com.lasalle.io_trucks.Peripherique

Classe permettant de gérer les périphériques.

Graphe de collaboration de com.lasalle.io_trucks.Peripherique :



Classes

— class TReception
Déclaration de la classe TReception.

Fonctions membres publiques

- void connecter ()
  Méthode permettant de se connecter à un périphérique.
- boolean deconnecter ()
  Méthode permettant de se déconnecter du périphérique.
- void envoyer (final String data)
  Méthode permettant d'envoyer une trame à l'aide du Thread.
- boolean estConnecte ()
  Méthode permettant de savoir si on est connecter.
- String getAdresse ()
  Méthode pour obtenir l'adresse du périphérique.
- String getNom ()
  Méthode pour obtenir le nom du périphérique.
- Peripherique (BluetoothDevice device, Handler handler)
  Constructeur de la classe Périphérique.
- void setNom (String nom)
  Méthode permettant de set le nom du périphérique.
- String toString ()
  Méthode permettant de renvoyer le périphérique en un String.

Attributs publics statiques

- static final int CODE_CONNEXION = 0
- static final int CODE_DECONNEXION = 3
- static final int CODE_EMISSION = 2
- static final int CODE_RECEPTION = 1

Fonctions membres privées

- boolean connecterSocket ()
  Méthode permettant de connecter le socket.
- void creerSocket ()
  Méthode de création du socket bluetooth.

Attributs privés

- String adresse
- BluetoothDevice device = null
- Handler handler = null
- String nom
- InputStream receiveStream = null
- OutputStream sendStream = null
- BluetoothSocket socket = null
- TReception treception = null

Attributs privés statiques

- static final String TAG = "Peripherique"

8.4.1 Description détaillée

Classe permettant de gérer les périphériques.

Définition à la ligne 27 du fichier Peripherique.java.

8.4.2 Documentation des constructeurs et destructeur

8.4.2.1 Peripherique()

```
com.lasalle.io_trucks.Peripherique.Peripherique (
            BluetoothDevice device,
            Handler handler )
```

Constructeur de la classe Périphérique.

Paramètres

device	Définis l'appareil associé
handler	Définis la gestion du thread

Définition à la ligne 54 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.creerSocket(), com.lasalle.io_trucks.Peripherique.device, et com.lasalle.io_trucks.Peripherique.handler.

```
00055     {
00056         if (device != null)
00057         {
00058             this.device = device;
00059             this.nom = device.getName();
00060             this.adresse = device.getAddress();
00061             this.handler = handler;
00062         }
00063         else
00064         {
00065             this.device = device;
00066             this.nom = "Aucun";
00067             this.adresse = "";
00068             this.handler = handler;
00069         }
00070 
00071         creerSocket();
00072     }
```

8.4.3 Documentation des fonctions membres

8.4.3.1 connecter()

```
void com.lasalle.io_trucks.Peripherique.connecter ( )
```

Méthode perméttant de se connecter à un périphérique.

Définition à la ligne 183 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.connecterSocket(), et com.lasalle.io_trucks.Peripherique.creerSocket().

Référencé par com.lasalle.io_trucks.MainActivity.onClick().

```
00184     {
00185         new Thread()
00186         {
00187             @Override
00188             public void run()
00189             {
00190                 if (connecterSocket())
00191                 {
00192                     return;
00193                 }
00194                 // sinon reconnexion
00195                 creerSocket();
00196                 connecterSocket();
00197 
00198             }
00199         }.start();
00200     }
```

8.4.3.2 connecterSocket()

```
boolean com.lasalle.io_trucks.Peripherique.connecterSocket ( )  [private]
```

Méthode perméttant de connecter le socket.

Renvoie

Renvoie un booléen afin de savoir si le socket est connecter

Définition à la ligne 206 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.CODE_CONNEXION.

Référencé par com.lasalle.io_trucks.Peripherique.connecter().

```
00207     {
00208         try
00209         {
00210             if(socket == null)
00211                 return false;
00212             if (!socket.isConnected())
00213             {
00214                 socket.connect();
00215                 Message msg = Message.obtain();
00216                 msg.what = CODE_CONNEXION;
00217                 handler.sendMessage(msg);
00218 
00219                 treception.start();
00220                 return true;
00221             }
00222         }
00223         catch (IOException e)
00224         {
00225             Log.d(TAG, "Erreur connect()");
00226             e.printStackTrace();
00227         }
00228         return false;
00229     }
```

8.4.3.3 creerSocket()

```
void com.lasalle.io_trucks.Peripherique.creerSocket ( )  [private]
```

Méthode de création du socket bluetooth.

Définition à la ligne 77 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.connecter(), et com.lasalle.io_trucks.Peripherique.Peripherique().

```
00078     {
00079         try
00080         {
00081             if(device != null)
00082             {
00083                 socket = device.createRfcommSocketToServiceRecord(UUID.fromString("
      00001101-0000-1000-8000-00805F9B34FB"));
00084                 receiveStream = socket.getInputStream();
00085                 sendStream = socket.getOutputStream();
00086             }
00087         }
00088         catch (IOException e)
00089         {
00090             Log.d(TAG, "Erreur createRfcommSocketToServiceRecord()");
00091             e.printStackTrace();
00092             socket = null;
00093         }
00094         if (socket != null)
00095         {
00096             treception = new TReception(handler);
00097         }
00098     }
```

8.4.3.4 deconnecter()

```
boolean com.lasalle.io_trucks.Peripherique.deconnecter ( )
```

Méthode perméttant de se déconnecter du périphérique.

Renvoie

Renvoie un booléen afin de savoir si on est bien déconnecter

Définition à la ligne 235 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.TReception.arreter(), et com.lasalle.io_trucks.Peripherique.CODE_DECONNEXION.

Référencé par com.lasalle.io_trucks.MainActivity.onClick().

```
00236     {
00237         try
00238         {
00239             treception.arreter();
00240 
00241             socket.close();
00242             Message msg = Message.obtain();
00243             msg.what = CODE_DECONNEXION;
00244             handler.sendMessage(msg);
00245 
00246             return true;
00247         }
00248         catch (IOException e)
00249         {
00250             Log.d(TAG, "Erreur close()");
00251             e.printStackTrace();
00252             return false;
00253         }
00254     }
```

8.4.3.5 envoyer()

```
void com.lasalle.io_trucks.Peripherique.envoyer (
             final String data )
```

Méthode perméttant d'envoyer une trame à l'aide du Thread.

Paramètres

data	Représente les données à envoyer

Définition à la ligne 149 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.CODE_EMISSION.

Référencé par com.lasalle.io_trucks.MainActivity.envoyerTrame().

```
00150     {
00151         if (socket == null)
00152             return;
00153 
00154         new Thread()
00155         {
00156             @Override
00157             public void run()
00158             {
```

```
00159                 try
00160                 {
00161                     if (socket.isConnected())
00162                     {
00163                         sendStream.write(data.getBytes());
00164                         sendStream.flush();
00165                         Message msg = Message.obtain();
00166                         msg.what = CODE_EMISSION;
00167                         msg.obj = data;
00168                         handler.sendMessage(msg);
00169                     }
00170                 }
00171                 catch (IOException e)
00172                 {
00173                     Log.d(TAG, "Erreur write()");
00174                     e.printStackTrace();
00175                 }
00176             }
00177         }.start();
00178     }
```

8.4.3.6 estConnecte()

```
boolean com.lasalle.io_trucks.Peripherique.estConnecte ( )
```

Méthode perméttant de savoir si on est connecter.

Renvoie

Renvoie un booléen définissant l'état

Définition à la ligne 122 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.MainActivity.envoyerTrame(), et com.lasalle.io_trucks.MainActivity.onClick().

```
00123     {
00124         return socket.isConnected();
00125     }
```

8.4.3.7 getAdresse()

```
String com.lasalle.io_trucks.Peripherique.getAdresse ( )
```

Méthode pour obtenir l'adresse du périphérique.

Renvoie

Renvoie l'adresse du périphérique

Définition à la ligne 113 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.adresse.

```
00114     {
00115         return adresse;
00116     }
```

8.4.3.8 getNom()

```
String com.lasalle.io_trucks.Peripherique.getNom ( )
```

Méthode pour obtenir le nom du périphérique.

Renvoie

Renvoie le nom du périphérique

Définition à la ligne 104 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.nom.

```
00105     {
00106         return nom;
00107     }
```

8.4.3.9 setNom()

```
void com.lasalle.io_trucks.Peripherique.setNom (
             String nom )
```

Méthode perméttant de set le nom du périphérique.

Paramètres

nom	définis le nom du périphérique

Définition à la ligne 131 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.nom.

```
00132     {
00133         this.nom = nom;
00134     }
```

8.4.3.10 toString()

```
String com.lasalle.io_trucks.Peripherique.toString ( )
```

Méthode perméttant de renvoyer le périphérique en un String.

Renvoie

Renvoie un String contennant le nom et l'adresse du périphérique

Définition à la ligne 140 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.adresse.

```
00141     {
00142         return "\nNom : " + nom + "\nAdresse : " + adresse;
00143     }
```

8.4.4 Documentation des données membres

8.4.4.1 adresse

```
String com.lasalle.io_trucks.Peripherique.adresse  [private]
```

Définition à la ligne 41 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.getAdresse(), et com.lasalle.io_trucks.Peripherique.toString().

8.4.4.2 CODE_CONNEXION

```
final int com.lasalle.io_trucks.Peripherique.CODE_CONNEXION = 0  [static]
```

Définition à la ligne 33 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.connecterSocket().

8.4.4.3 CODE_DECONNEXION

```
final int com.lasalle.io_trucks.Peripherique.CODE_DECONNEXION = 3  [static]
```

Définition à la ligne 36 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.deconnecter().

8.4.4.4 CODE_EMISSION

```
final int com.lasalle.io_trucks.Peripherique.CODE_EMISSION = 2  [static]
```

Définition à la ligne 35 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.envoyer().

8.4.4.5 CODE_RECEPTION

```
final int com.lasalle.io_trucks.Peripherique.CODE_RECEPTION = 1  [static]
```

Définition à la ligne 34 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.TReception.run().

8.4.4.6 device

```
BluetoothDevice com.lasalle.io_trucks.Peripherique.device = null  [private]
```

Définition à la ligne 43 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.Peripherique().

8.4.4.7 handler

```
Handler com.lasalle.io_trucks.Peripherique.handler = null  [private]
```

Définition à la ligne 42 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.Peripherique().

8.4.4.8 nom

```
String com.lasalle.io_trucks.Peripherique.nom  [private]
```

Attributs

Définition à la ligne 40 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.getNom(), et com.lasalle.io_trucks.Peripherique.setNom().

8.4.4.9 receiveStream

```
InputStream com.lasalle.io_trucks.Peripherique.receiveStream = null  [private]
```

Définition à la ligne 45 du fichier Peripherique.java.

8.4.4.10 sendStream

```
OutputStream com.lasalle.io_trucks.Peripherique.sendStream = null  [private]
```

Définition à la ligne 46 du fichier Peripherique.java.

8.4.4.11 socket

```
BluetoothSocket com.lasalle.io_trucks.Peripherique.socket = null  [private]
```

Définition à la ligne 44 du fichier Peripherique.java.

8.4.4.12 TAG

```
final String com.lasalle.io_trucks.Peripherique.TAG = "Peripherique"  [static], [private]
```

Constantes

Définition à la ligne 32 du fichier Peripherique.java.

8.4.4.13 treception

```
TReception com.lasalle.io_trucks.Peripherique.treception = null  [private]
```

Définition à la ligne 47 du fichier Peripherique.java.

La documentation de cette classe a été générée à partir du fichier suivant :

— Peripherique.java

8.5 Référence de la classe com.lasalle.io_trucks.Protocole

Graphe de collaboration de com.lasalle.io_trucks.Protocole :

Attributs publics statiques

— static final String BAISSE = "0"
— static final String CHARGE_ATTENTION = "1"
— static final String CHARGE_NORMAL = "0"
— static final String CHARGE_SURCHARGE = "2"
— static final String DELIMITEUR_CHAMP = ";"
— static final String DELIMITEUR_FIN = "\r\n"
— static final String EN_TETE = "$iotruck"
— static final String LEVE = "1"
— static final int NB_PARAMETRES_CMD = 2
— static final int NB_PARAMETRES_REQUETE = 1
— static final int NB_PARAMETRES_SERVICE = 1
— static final String OFF = "0"
— static final String ON = "1"
— static final String TRAME_CMD_CHAMP1_ECLAIRAGE = "E"
— static final String TRAME_CMD_CHAMP1_GYRO = "G"
— static final String TRAME_CMD_CHAMP1_HAYON = "H"
— static final String TRAME_CMD_CHAMP1_TRIANGLE = "T"
— static final String TRAME_REQUETE_STATE1 = "S1"
— static final String TRAME_REQUETE_STATE2 = "S2"
— static final String TRAME_SERVICE = "A"

8.5.1 Description détaillée

Définition à la ligne 3 du fichier Protocole.java.

8.5.2 Documentation des données membres

8.5.2.1 BAISSE

```
final String com.lasalle.io_trucks.Protocole.BAISSE = "0"  [static]
```

Définition à la ligne 28 du fichier Protocole.java.

8.5.2.2 CHARGE_ATTENTION

```
final String com.lasalle.io_trucks.Protocole.CHARGE_ATTENTION = "1"  [static]
```

Définition à la ligne 33 du fichier Protocole.java.

8.5.2.3 CHARGE_NORMAL

```
final String com.lasalle.io_trucks.Protocole.CHARGE_NORMAL = "0"  [static]
```

Etat de la charge

Définition à la ligne 32 du fichier Protocole.java.

8.5.2.4 CHARGE_SURCHARGE

```
final String com.lasalle.io_trucks.Protocole.CHARGE_SURCHARGE = "2"  [static]
```

Définition à la ligne 34 du fichier Protocole.java.

8.5.2.5 DELIMITEUR_CHAMP

```
final String com.lasalle.io_trucks.Protocole.DELIMITEUR_CHAMP = ";"  [static]
```

Définition à la ligne 9 du fichier Protocole.java.

Référencé par com.lasalle.io_trucks.MainActivity.decoderTrame().

8.5.2.6 DELIMITEUR_FIN

```
final String com.lasalle.io_trucks.Protocole.DELIMITEUR_FIN = "\r\n"  [static]
```

Définition à la ligne 10 du fichier Protocole.java.

Référencé par com.lasalle.io_trucks.MainActivity.decoderTrame().

8.5.2.7 EN_TETE

```
final String com.lasalle.io_trucks.Protocole.EN_TETE = "$iotruck"  [static]
```

Général

Définition à la ligne 8 du fichier Protocole.java.

Référencé par com.lasalle.io_trucks.MainActivity.decoderTrame().

8.5.2.8 LEVE

```
final String com.lasalle.io_trucks.Protocole.LEVE = "1"  [static]
```

Définition à la ligne 27 du fichier Protocole.java.

Référencé par com.lasalle.io_trucks.MainActivity.afficherEtatS1().

8.5.2.9 NB_PARAMETRES_CMD

```
final int com.lasalle.io_trucks.Protocole.NB_PARAMETRES_CMD = 2  [static]
```

Trame de commande CMD

Définition à la ligne 14 du fichier Protocole.java.

8.5.2.10 NB_PARAMETRES_REQUETE

```
final int com.lasalle.io_trucks.Protocole.NB_PARAMETRES_REQUETE = 1  [static]
```

Trame de requête

Définition à la ligne 24 du fichier Protocole.java.

8.5.2.11 NB_PARAMETRES_SERVICE

```
final int com.lasalle.io_trucks.Protocole.NB_PARAMETRES_SERVICE = 1  [static]
```

Définition à la ligne 39 du fichier Protocole.java.

8.5.2.12 OFF

```
final String com.lasalle.io_trucks.Protocole.OFF = "0"  [static]
```

Définition à la ligne 20 du fichier Protocole.java.

8.5.2.13 ON

```
final String com.lasalle.io_trucks.Protocole.ON = "1"  [static]
```

Définition à la ligne 19 du fichier Protocole.java.

Référencé par com.lasalle.io_trucks.MainActivity.afficherEtatS1().

8.5.2.14 TRAME_CMD_CHAMP1_ECLAIRAGE

```
final String com.lasalle.io_trucks.Protocole.TRAME_CMD_CHAMP1_ECLAIRAGE = "E"  [static]
```

Définition à la ligne 17 du fichier Protocole.java.

8.5.2.15 TRAME_CMD_CHAMP1_GYRO

```
final String com.lasalle.io_trucks.Protocole.TRAME_CMD_CHAMP1_GYRO = "G"  [static]
```

Définition à la ligne 16 du fichier Protocole.java.

8.5.2.16 TRAME_CMD_CHAMP1_HAYON

```
final String com.lasalle.io_trucks.Protocole.TRAME_CMD_CHAMP1_HAYON = "H"  [static]
```

Définition à la ligne 18 du fichier Protocole.java.

8.5.2.17 TRAME_CMD_CHAMP1_TRIANGLE

```
final String com.lasalle.io_trucks.Protocole.TRAME_CMD_CHAMP1_TRIANGLE = "T"  [static]
```

Définition à la ligne 15 du fichier Protocole.java.

8.5.2.18 TRAME_REQUETE_STATE1

```
final String com.lasalle.io_trucks.Protocole.TRAME_REQUETE_STATE1 = "S1"  [static]
```

Définition à la ligne 25 du fichier Protocole.java.

Référencé par com.lasalle.io_trucks.MainActivity.traiterTrame().

8.5.2.19 TRAME_REQUETE_STATE2

```
final String com.lasalle.io_trucks.Protocole.TRAME_REQUETE_STATE2 = "S2"  [static]
```

Définition à la ligne 26 du fichier Protocole.java.

8.5.2.20 TRAME_SERVICE

```
final String com.lasalle.io_trucks.Protocole.TRAME_SERVICE = "A"  [static]
```

Trame de service

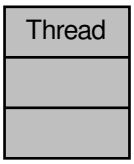
Définition à la ligne 38 du fichier Protocole.java.

La documentation de cette classe a été générée à partir du fichier suivant :

— Protocole.java

8.6 Référence de la classe Thread

Graphe de collaboration de Thread :



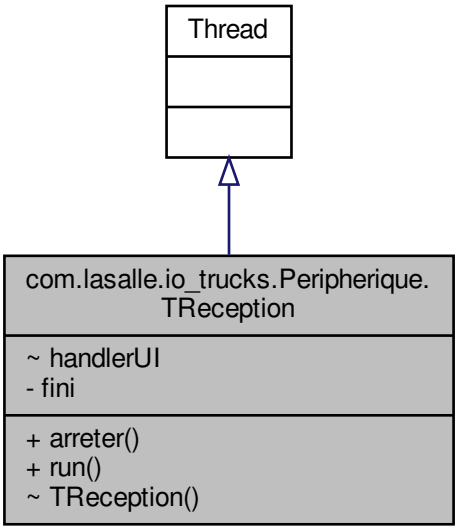
La documentation de cette classe a été générée à partir du fichier suivant :

— Peripherique.java

8.7 Référence de la classe com.lasalle.io_trucks.Peripherique.TReception

Déclaration de la classe TReception.

Graphe de collaboration de com.lasalle.io_trucks.Peripherique.TReception :



Fonctions membres publiques

— void arreter ()
— void run ()

Attributs privés

— boolean fini

8.7.1 Description détaillée

Déclaration de la classe TReception.

Définition à la ligne 260 du fichier Peripherique.java.

8.7.2 Documentation des fonctions membres

8.7.2.1 arreter()

```
void com.lasalle.io_trucks.Peripherique.TReception.arreter ( )
```

Définition à la ligne 312 du fichier Peripherique.java.

Référencé par com.lasalle.io_trucks.Peripherique.deconnecter().

```
00313         {
00314             if (fini == false)
00315             {
00316                 fini = true;
00317             }
00318             try
00319             {
00320                 Thread.sleep(250);
00321             }
00322             catch (InterruptedException e)
00323             {
00324                 e.printStackTrace();
00325             }
00326         }
```

8.7.2.2 run()

```
void com.lasalle.io_trucks.Peripherique.TReception.run ( )
```

Définition à la ligne 272 du fichier Peripherique.java.

Références com.lasalle.io_trucks.Peripherique.CODE_RECEPTION.

```
00273         {
00274             Log.d(TAG, "TReception run() start");
00275             BufferedReader reception = new BufferedReader(new InputStreamReader(
    receiveStream));
00276             while (!fini)
00277             {
00278                 try
00279                 {
00280                     String trame = "";
00281                     if (reception.ready())
00282                     {
00283                         trame = reception.readLine();
00284                     }
00285                     if (trame.length() > 0)
00286                     {
00287                         Log.d(TAG, "run() trame : " + trame);
00288                         Message msg = Message.obtain();
00289                         msg.what = Peripherique.CODE_RECEPTION;
00290                         msg.obj = trame;
00291                         handlerUI.sendMessage(msg);
00292                     }
00293                 }
00294                 catch (IOException e)
00295                 {
00296                     Log.d(TAG, "Erreur read()");
00297                     e.printStackTrace();
00298                 }
00299
00300                 try
00301                 {
00302                     Thread.sleep(250);
00303                 }
00304                 catch (InterruptedException e)
00305                 {
00306                     e.printStackTrace();
00307                 }
00308             }
00309             Log.d(TAG, "TReception run() stop");
00310         }
```

8.7.3 Documentation des données membres

8.7.3.1 fini

```
boolean com.lasalle.io_trucks.Peripherique.TReception.fini  [private]
```

Définition à la ligne 263 du fichier Peripherique.java.

La documentation de cette classe a été générée à partir du fichier suivant :

— Peripherique.java

9 Documentation des fichiers

9.1 Référence du fichier Accueil.java

Déclaration de la classe Accueil.

Classes

— class com.lasalle.io_trucks.Accueil
Classe de l'activité Accueil La classe Acceuil est l'activité de démarrage de l'application.

Paquetages

— package com.lasalle.io_trucks

9.1.1 Description détaillée

Déclaration de la classe Accueil.

Auteur

Mathieu Arthur

Version

0.1

Définition dans le fichier Accueil.java.

9.2 Accueil.java

```
00001 package com.lasalle.io_trucks;
00002
00003 import androidx.appcompat.app.AppCompatActivity;
00004
00005 import android.content.Intent;
00006 import android.os.Bundle;
00007 import android.util.Log;
00008 import android.view.animation.Animation;
00009 import android.view.animation.AnimationUtils;
00010 import android.widget.ImageView;
00011
00012 import java.io.Serializable;
00013
00026 public class Accueil extends AppCompatActivity
00027 {
00031     private static final String TAG = "IHMAccueil";
00035     private Animation animation;
00036     private ImageView imageView;
00037
00042     @Override
00043     protected void onCreate(Bundle savedInstanceState)
00044     {
00045         super.onCreate(savedInstanceState);
00046         setContentView(R.layout.activity_accueil);
00047         Log.i(TAG,"onCreate()");
00048
00049         imageView = (ImageView)findViewById(R.id.imageView);
00050         animation = AnimationUtils.loadAnimation(getApplicationContext(), R.anim.fade_in);
00051         animation.setAnimationListener(new Animation.AnimationListener()
00052         {
00058             @Override
00059             public void onAnimationStart(Animation animation)
00060             {
00061             }
00062
00069             @Override
00070             public void onAnimationEnd(Animation animation)
00071             {
00072                 // A la fin de l'animation, on lance l'activité principale
00073                 Intent intent = new Intent(Accueil.this, MainActivity.class);
00074                 startActivity(intent);
00075             }
00076
00077             @Override
00078             public void onAnimationRepeat(Animation animation)
00079             {
00080             }
00081         });
00082         imageView.startAnimation(animation);
00083     }
00084 }
```

9.3 Référence du fichier Changelog.md

9.4 Changelog.md

```
00001 \page page_changelog Changelog
00002
00003 r1 | www-data | 2020-02-01 15:03:29 +0100 (sam. 01 févr. 2020) | 1 ligne
00004
00005 Creating initial repository structure
```

9.5 Référence du fichier Communication.java

Déclaration de la classe Communication.

Classes

— class com.lasalle.io_trucks.Communication
Classe de Communication et de connexion bluetooth.

Paquetages

— package com.lasalle.io_trucks

9.5.1 Description détaillée

Déclaration de la classe Communication.

Auteur

Mathieu Arthur

Version

0.1

Définition dans le fichier Communication.java.

9.6 Communication.java

```
00001 package com.lasalle.io_trucks;
00002
00003 import android.app.Activity;
00004 import android.bluetooth.BluetoothAdapter;
00005 import android.bluetooth.BluetoothDevice;
00006 import android.content.BroadcastReceiver;
00007 import android.content.Context;
00008 import android.content.Intent;
00009 import android.util.Log;
00010 import android.widget.Toast;
00011
00012 import java.util.Set;
00013
00025 public class Communication
00026 {
00030     private static final String TAG = "Communication";
00034     private BroadcastReceiver receiverEtatBluetooth;
00035     private BroadcastReceiver receiverScan;
00036     private BluetoothAdapter bluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
00037     private Set<BluetoothDevice> listeAppareilConnus;
00038
00043     public void demanderActivationBluetooth(Context contextAcceuil)
00044     {
00045         if (bluetoothAdapter == null)
00046         {
00047             Log.i(TAG,"demanderActivationBluetooth() bluetoothAdapter = null");
00048             Toast.makeText(contextAcceuil, R.string.str_bluetoot_inexistant, Toast.LENGTH_SHORT).show();
00049         }
00050         else if (!bluetoothAdapter.isEnabled())
00051         {
00052             Log.i(TAG,"demanderActivationBluetooth() bluetooth désactivé");
00053             Toast.makeText(contextAcceuil, R.string.str_bluetooth_eteint, Toast.LENGTH_SHORT).show();
00054             Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
00055             ((Activity) contextAcceuil).startActivityForResult(enableBtIntent,1);
00056         }
00057         else
00058         {
00059             Log.i(TAG,"demanderActivationBluetooth() bluetooth activé");
00060             Toast.makeText(contextAcceuil, R.string.str_bluetooth_allumer, Toast.LENGTH_SHORT).show();
00061         }
00062     }
00063
00068     public BroadcastReceiver ecouterEtatBluetooth()
00069     {
00070         receiverEtatBluetooth = new BroadcastReceiver()
00071         {
00072             @Override
00073             public void onReceive(Context context, Intent intent)
00074             {
00075                 final String action = intent.getAction();
00076                 Log.i(TAG,"onReceive() " + action);
00077                 if (action.equals(BluetoothAdapter.ACTION_STATE_CHANGED))
00078                 {
00079                     final int state = intent.getIntExtra(BluetoothAdapter.EXTRA_STATE, BluetoothAdapter.
      ERROR);
00080                     switch (state)
```

```
00081                     {
00082                         case BluetoothAdapter.STATE_OFF:
00083                             Log.i(TAG,"onReceive() Bluetooth désactivé !");
00084                             Toast.makeText(context, R.string.str_bluetooth_eteint, Toast.LENGTH_SHORT).show
      ();
00085                             break;
00086                         case BluetoothAdapter.STATE_TURNING_OFF:
00087                             Toast.makeText(context, R.string.str_bluetooth_eteint_en_cours, Toast.
      LENGTH_LONG).show();
00088                             break;
00089                         case BluetoothAdapter.STATE_ON:
00090                             Log.i(TAG,"onReceive() Bluetooth activé !");
00091                             Toast.makeText(context, R.string.str_bluetooth_allumer, Toast.LENGTH_SHORT).
      show();
00092                             break;
00093                         case BluetoothAdapter.STATE_TURNING_ON:
00094                             Log.i(TAG,"onReceive() Bluetooth s'active !");
00095                             Toast.makeText(context, R.string.str_bluetooth_allumer_en_cours, Toast.
      LENGTH_LONG).show();
00096                             break;
00097                     }
00098                 }
00099             }
00100         };
00101 
00102         return receiverEtatBluetooth;
00103     }
00104 
00109     public void rechercherAppareilConnu(Context contextAcceuil)
00110     {
00111         listeAppareilConnus = bluetoothAdapter.getBondedDevices();
00112         for(BluetoothDevice blueDevice : listeAppareilConnus)
00113         {
00114             Toast.makeText(contextAcceuil, "Appareil " + blueDevice.getName(), Toast.LENGTH_SHORT).show();
00115         }
00116     }
00117 
00123     public BluetoothDevice recupererAppareilBluetooth(String nomAppareil)
00124     {
00125         listeAppareilConnus = bluetoothAdapter.getBondedDevices();
00126         for(BluetoothDevice blueDevice : listeAppareilConnus)
00127         {
00128             if(blueDevice.getName().equals(nomAppareil))
00129             {
00130                 Log.d(TAG, "recupererAppareilBluetooth() io-trucks trouvé : " + blueDevice.getName() + " ("
       + blueDevice.getAddress() + ")");
00131                 return blueDevice;
00132             }
00133         }
00134         return null;
00135     }
00136 
00141     public void unregisterBluetooth(Context contextAcceuil)
00142     {
00143         if (bluetoothAdapter != null)
00144         {
00145             bluetoothAdapter.cancelDiscovery();
00146             contextAcceuil.unregisterReceiver(receiverEtatBluetooth);
00147         }
00148     }
00149 }
```

9.7 Référence du fichier MainActivity.java

Déclaration de la classe MainActivity.

Classes

— class com.lasalle.io_trucks.MainActivity
Classe IHM principale.

Paquetages

— package com.lasalle.io_trucks

9.7.1 Description détaillée

Déclaration de la classe MainActivity.

Auteur

Mathieu Arthur

Version

0.1

Définition dans le fichier MainActivity.java.

9.8 MainActivity.java

```
00001 package com.lasalle.io_trucks;
00002
00003 import androidx.appcompat.app.AlertDialog;
00004 import androidx.appcompat.app.AppCompatActivity;
00005
00006 import android.bluetooth.BluetoothAdapter;
00007 import android.bluetooth.BluetoothDevice;
00008 import android.content.DialogInterface;
00009 import android.content.IntentFilter;
00010 import android.os.Bundle;
00011 import android.os.Handler;
00012 import android.os.Message;
00013 import android.util.Log;
00014 import android.view.View;
00015 import android.widget.Button;
00016 import android.widget.EditText;
00017 import android.widget.ImageButton;
00018 import android.widget.ImageView;
00019 import android.widget.TextView;
00020
00032 public class MainActivity extends AppCompatActivity implements View.OnClickListener
00033 {
00037     private static final String TAG = "IHMMainActivity";
00041     private Boolean etatTriangle = false;
00042     private Boolean etatGyrophare = false;
00043     private Boolean etatEclairage = false;
00044     private Button buttonBluetooth;
00045     private Button buttonRechercher;
00046     private Button buttonRafraichir;
00047     private ImageButton imageButtonTriangle;
00048     private ImageButton imageButtonGyrophare;
00049     private ImageButton imageButtonEclairage;
00050     private Communication communicationBluetooth = new
      Communication();
00051     private Peripherique peripheriqueBluetooth = null;
00052     private ImageView imageEtatConnection;
00053     private TextView textEtatConnection;
00054     private EditText editNomPeripherique;
00055
00060     @Override
00061     protected void onCreate(Bundle savedInstanceState)
00062     {
00063         super.onCreate(savedInstanceState);
00064         setContentView(R.layout.activity_main);
00065         Log.i(TAG,"onCreate()");
00066
00067         recupererWidgets();
00068         initialiserWidgets();
00069
00070         communicationBluetooth.demanderActivationBluetooth(this);
00071     }
00072
00076     @Override
00077     protected void onStart()
00078     {
00079         super.onStart();
00080         Log.i(TAG,"onStart()");
00081     }
00082
00086     @Override
00087     protected void onResume()
00088     {
00089         super.onResume();
```

```
00090         Log.i(TAG,"onResume()");
00091         creerLiasonReceiverEtatBluetooth();
00092     }
00093
00097     @Override
00098     protected void onPause()
00099     {
00100         super.onPause();
00101         Log.i(TAG,"onPause()");
00102         communicationBluetooth.unregisterBluetooth(this);
00103     }
00104
00108     @Override
00109     protected void onStop()
00110     {
00111         super.onStop();
00112         Log.i(TAG,"onStop()");
00113
00114     }
00115
00119     @Override
00120     protected void onDestroy()
00121     {
00122         super.onDestroy();
00123         Log.i(TAG,"onDestroy()");
00124
00125     }
00126
00132     @Override
00133     public void onClick(View element)
00134     {
00135         if(element == buttonBluetooth)
00136         {
00137             if(buttonBluetooth.getText().equals("Connecter"))
00138             {
00139                 BluetoothDevice blueDevice = communicationBluetooth.
      recupererAppareilBluetooth(editNomPeripherique.getText().toString());
00140                 if(blueDevice == null)
00141                 {
00142                     AlertDialog.Builder boiteAvertissementNonTrouver = new AlertDialog.Builder(this);
00143                     boiteAvertissementNonTrouver.setMessage("L'appareil io-trucks n'as pas été trouvé.
       Vérifiez si celui a été appairé correctement.");
00144                     boiteAvertissementNonTrouver.setPositiveButton("Continuer", new DialogInterface.
      OnClickListener() {
00145                         @Override
00146                         public void onClick(DialogInterface dialog, int which) {
00147                         }
00148                     });
00149                     boiteAvertissementNonTrouver.show();
00150                     Log.i(TAG, "Appareil io-trucks non trouvé !");
00151                     return;
00152                 }
00153                 else
00154                 {
00155                     if(peripheriqueBluetooth == null)
00156                     {
00157                         Log.i(TAG, "Instancie peripheriqueBluetooth");
00158                         peripheriqueBluetooth = new Peripherique(blueDevice,
      handler);
00159                     }
00160                     if(!peripheriqueBluetooth.estConnecte())
00161                     {
00162                         Log.i(TAG, "Connexion peripheriqueBluetooth");
00163                         peripheriqueBluetooth.connecter();
00164                     }
00165                     else // déjà connecté !
00166                     {
00167                     }
00168                 }
00169             }
00170             else if(buttonBluetooth.getText().equals("Déconnecter"))
00171             {
00172                 if (peripheriqueBluetooth.estConnecte())
00173                 {
00174                     peripheriqueBluetooth.deconnecter();
00175                 }
00176             }
00177         }
00178         else if(element == buttonRechercher)
00179         {
00180             communicationBluetooth.rechercherAppareilConnu(this);
00181         }
00182         else if(element == imageButtonTriangle)
00183         {
00184             Log.i(TAG,"button Triangle");
00185             if(!etatTriangle)
00186             {
00187                 envoyerTrame("$iotruck;T;1\r\n");
00188                 imageButtonTriangle.setImageResource(R.drawable.triangle);
00189                 etatTriangle = true;
00190             }
```

```
00191             else
00192             {
00193                 envoyerTrame("$iotruck;T;0\r\n");
00194                 imageButtonTriangle.setImageResource(R.drawable.triangle_b_w);
00195                 etatTriangle = false;
00196             }
00197         }
00198         else if(element == imageButtonGyrophare)
00199         {
00200             Log.i(TAG,"button Gyrophare");
00201             if(!etatGyrophare)
00202             {
00203                 envoyerTrame("$iotruck;G;1\r\n");
00204                 imageButtonGyrophare.setImageResource(R.drawable.flash);
00205                 etatGyrophare = true;
00206             }
00207             else
00208             {
00209                 envoyerTrame("$iotruck;G;0\r\n");
00210                 imageButtonGyrophare.setImageResource(R.drawable.flash_b_w);
00211                 etatGyrophare = false;
00212             }
00213         }
00214         else if(element == imageButtonEclairage)
00215         {
00216             Log.i(TAG,"button Eclairage");
00217             if(!etatEclairage)
00218             {
00219                 envoyerTrame("$iotruck;E;1\r\n");
00220                 imageButtonEclairage.setImageResource(R.drawable.spotlight);
00221                 etatEclairage = true;
00222             }
00223             else
00224             {
00225                 envoyerTrame("$iotruck;E;0\r\n");
00226                 imageButtonEclairage.setImageResource(R.drawable.spotlight_b_w);
00227                 etatEclairage = false;
00228             }
00229         }
00230         else if(element == buttonRafraichir)
00231         {
00232             Log.i(TAG,"button Rafraichir");
00233             demanderEtats();
00234         }
00235         else
00236         {
00237             Log.i(TAG,"button Inconnu : " + element.getId());
00238         }
00239     }
00240
00244         private void envoyerTrame(String trame)
00245     {
00246         if(peripheriqueBluetooth != null)
00247         {
00248             if (peripheriqueBluetooth.estConnecte())
00249             {
00250                 Log.i(TAG, "envoyerTrame() trame : " + trame);
00251                 peripheriqueBluetooth.envoyer(trame);
00252             }
00253         }
00254     }
00255
00259     private void recupererWidgets()
00260     {
00261         buttonBluetooth = findViewById(R.id.buttonConnecter);
00262         buttonRechercher = findViewById(R.id.buttonBounded);
00263         buttonRafraichir = findViewById(R.id.buttonRafraichir);
00264         imageButtonTriangle = findViewById(R.id.imageButtonTriangle);
00265         imageButtonGyrophare = findViewById(R.id.imageButtonGyrophares);
00266         imageButtonEclairage = findViewById(R.id.imageButtonEclairage);
00267         imageEtatConnection = findViewById(R.id.imageViewEtatConnection);
00268         textEtatConnection = findViewById(R.id.textViewEtatConnection);
00269         editNomPeripherique = findViewById(R.id.editNomPeripherique);;
00270     }
00271
00275     private void initialiserWidgets()
00276     {
00277         renitialiserVue();
00278
00279         buttonBluetooth.setOnClickListener(this);
00280         buttonRechercher.setOnClickListener(this);
00281         buttonRafraichir.setOnClickListener(this);
00282         imageButtonTriangle.setOnClickListener(this);
00283         imageButtonGyrophare.setOnClickListener(this);
00284         imageButtonEclairage.setOnClickListener(this);
00285     }
00286
00290     private void creerLiasonReceiverEtatBluetooth()
00291     {
00292         IntentFilter filter = new IntentFilter(BluetoothAdapter.ACTION_STATE_CHANGED);
00293         registerReceiver(communicationBluetooth.ecouterEtatBluetooth(), filter);
```

```
00294     }
00295
00296     private void activerVue()
00297     {
00298         buttonBluetooth.setText("Déconnecter");
00299         buttonRafraichir.setEnabled(true);
00300         imageButtonTriangle.setEnabled(true);
00301         imageButtonGyrophare.setEnabled(true);
00302         imageButtonEclairage.setEnabled(true);
00303         imageEtatConnection.setImageResource(R.drawable.green_cricle);
00304         textEtatConnection.setText("Connecté");
00305     }
00306
00310     private void renitialiserVue()
00311     {
00312         buttonBluetooth.setText("Connecter");
00313         buttonRafraichir.setEnabled(false);
00314         imageButtonTriangle.setEnabled(false);
00315         imageButtonGyrophare.setEnabled(false);
00316         imageButtonEclairage.setEnabled(false);
00317         imageButtonTriangle.setImageResource(R.drawable.triangle_b_w);
00318         imageButtonEclairage.setImageResource(R.drawable.spotlight_b_w);
00319         imageButtonGyrophare.setImageResource(R.drawable.flash_b_w);
00320         imageEtatConnection.setImageResource(R.drawable.red_circle);
00321         textEtatConnection.setText("Déconnecté");
00322     }
00323
00328     private void decoderTrame(String trame)
00329     {
00330         String nouvelleTrame = "";
00331         // Exemple : trame = "$iotruck;S1;0;0;0\r\n"
00332         nouvelleTrame = trame.replace(Protocole.EN_TETE,""); // enlever aussi le ; ?
00333         // Exemple : nouvelleTrame = ";S1;0;0;0\r\n"
00334         nouvelleTrame.replace(Protocole.DELIMITEUR_FIN,"");
00335         // Exemple : nouvelleTrame = ";S1;0;0;0"
00336         String[] trameCouper = nouvelleTrame.split(Protocole.
      DELIMITEUR_CHAMP);
00337         // Exemple : trameCouper = [0];[1];[2];[3];[4]
00338         Log.v(TAG, "decoderTrame() découpage de la trame");
00339         // le premier champ est vide
00340         for(int i = 1; i < trameCouper.length; i++)
00341         {
00342             Log.v(TAG, "decoderTrame() champ " + i + " = " + trameCouper[i]);
00343         }
00344         traiterTrame(trameCouper);
00345     }
00346
00351     private void traiterTrame(String[] trame)
00352     {
00353         Log.v(TAG, "traiterTrame() trame[1] = " + trame[1] + " (type)");
00354         if(trame[1].equals(Protocole.TRAME_REQUETE_STATE1))
00355         {
00356             afficherEtatS1(trame);
00357         }
00361     }
00362
00363     private void afficherEtatS1(String[] trame)
00364     {
00365         Log.v(TAG, "traiterTrame() trame[2] = " + trame[2] + " (triangle)");
00366         if(trame[2].equals(Protocole.LEVE))
00367         {
00368             imageButtonTriangle.setImageResource(R.drawable.triangle);
00369             etatTriangle = true;
00370         }
00371         else
00372         {
00373             imageButtonTriangle.setImageResource(R.drawable.triangle_b_w);
00374             etatTriangle = false;
00375         }
00376
00377         Log.v(TAG, "traiterTrame() trame[3] = " + trame[3] + " (gyrophare)");
00378         if(trame[3].equals(Protocole.ON))
00379         {
00380             imageButtonGyrophare.setImageResource(R.drawable.flash);
00381             etatGyrophare = true;
00382         }
00383         else
00384         {
00385             imageButtonGyrophare.setImageResource(R.drawable.flash_b_w);
00386             etatGyrophare = false;
00387         }
00388
00389         Log.v(TAG, "traiterTrame() trame[4] = " + trame[4] + " (éclairage)");
00390         if(trame[4].equals(Protocole.ON))
00391         {
00392             imageButtonEclairage.setImageResource(R.drawable.spotlight);
00393             etatEclairage = true;
00394         }
00395         else
00396         {
00397             imageButtonEclairage.setImageResource(R.drawable.spotlight_b_w);
```

```
00398             etatEclairage = false;
00399         }
00400     }
00401
00406     final private Handler handler = new Handler()
00407     {
00408         @Override
00409         public void handleMessage(Message msg)
00410         {
00411             super.handleMessage(msg);
00412
00413             switch(msg.what)
00414             {
00415                 case Peripherique.CODE_CONNEXION:
00416                     Log.v(TAG, "handleMessage() io-trucks connecté");
00417                     activerVue();
00418                     demanderEtats();
00419                     break;
00420                 case Peripherique.CODE_RECEPTION:
00421                     Log.v(TAG, "handleMessage() io-trucks réception : " + (String)msg.obj);
00422                     decoderTrame((String)msg.obj);
00423                     break;
00424                 case Peripherique.CODE_EMISSION:
00425                     Log.v(TAG, "handleMessage() io-trucks émission : " + (String)msg.obj);
00429                     break;
00430                 case Peripherique.CODE_DECONNEXION:
00431                     Log.v(TAG, "handleMessage() io-trucks déconnecté");
00432                     renitialiserVue();
00433                     break;
00434             }
00435         }
00436     };
00437
00441     private void demanderEtats()
00442     {
00443         envoyerTrame("$iotruck;S1\r\n");
00444         envoyerTrame("$iotruck;S2\r\n");
00445     }
00446 }
```

9.9 Référence du fichier Peripherique.java

Déclaration de la classe Peripherique.

Classes

— class com.lasalle.io_trucks.Peripherique
Classe permettant de gérer les périphériques.
— class com.lasalle.io_trucks.Peripherique.TReception
Déclaration de la classe TReception.

Paquetages

— package com.lasalle.io_trucks

9.9.1 Description détaillée

Déclaration de la classe Peripherique.

Auteur

Mathieu Arthur

Version

0.1

Définition dans le fichier Peripherique.java.

9.10 Peripherique.java

```
00001 package com.lasalle.io_trucks;
00002
00003 import android.bluetooth.BluetoothDevice;
00004 import android.bluetooth.BluetoothSocket;
00005 import android.os.Handler;
00006 import android.os.Message;
00007 import android.util.Log;
00008
00009 import java.io.BufferedReader;
00010 import java.io.IOException;
00011 import java.io.InputStream;
00012 import java.io.InputStreamReader;
00013 import java.io.OutputStream;
00014 import java.util.UUID;
00015
00027 public class Peripherique extends Thread
00028 {
00032     private static final String TAG = "Peripherique";
00033     public final static int CODE_CONNEXION = 0;
00034     public final static int CODE_RECEPTION = 1;
00035     public final static int CODE_EMISSION = 2;
00036     public final static int CODE_DECONNEXION = 3;
00040     private String nom;
00041     private String adresse;
00042     private Handler handler = null;
00043     private BluetoothDevice device = null;
00044     private BluetoothSocket socket = null;
00045     private InputStream receiveStream = null;
00046     private OutputStream sendStream = null;
00047     private TReception treception = null;
00048
00054     public Peripherique(BluetoothDevice device, Handler handler)
00055     {
00056         if (device != null)
00057         {
00058             this.device = device;
00059             this.nom = device.getName();
00060             this.adresse = device.getAddress();
00061             this.handler = handler;
00062         }
00063         else
00064         {
00065             this.device = device;
00066             this.nom = "Aucun";
00067             this.adresse = "";
00068             this.handler = handler;
00069         }
00070
00071         creerSocket();
00072     }
00073
00077     private void creerSocket()
00078     {
00079         try
00080         {
00081             if(device != null)
00082             {
00083                 socket = device.createRfcommSocketToServiceRecord(UUID.fromString("
      00001101-0000-1000-8000-00805F9B34FB"));
00084                 receiveStream = socket.getInputStream();
00085                 sendStream = socket.getOutputStream();
00086             }
00087         }
00088         catch (IOException e)
00089         {
00090             Log.d(TAG, "Erreur createRfcommSocketToServiceRecord()");
00091             e.printStackTrace();
00092             socket = null;
00093         }
00094         if (socket != null)
00095         {
00096             treception = new TReception(handler);
00097         }
00098     }
00099
00104     public String getNom()
00105     {
00106         return nom;
00107     }
00108
00113     public String getAdresse()
00114     {
00115         return adresse;
00116     }
00117
00122     public boolean estConnecte()
00123     {
00124         return socket.isConnected();
```

```
00125     }
00126
00131     public void setNom(String nom)
00132     {
00133         this.nom = nom;
00134     }
00135
00140     public String toString()
00141     {
00142         return "\nNom : " + nom + "\nAdresse : " + adresse;
00143     }
00144
00149     public void envoyer(final String data)
00150     {
00151         if (socket == null)
00152             return;
00153
00154         new Thread()
00155         {
00156             @Override
00157             public void run()
00158             {
00159                 try
00160                 {
00161                     if (socket.isConnected())
00162                     {
00163                         sendStream.write(data.getBytes());
00164                         sendStream.flush();
00165                         Message msg = Message.obtain();
00166                         msg.what = CODE_EMISSION;
00167                         msg.obj = data;
00168                         handler.sendMessage(msg);
00169                     }
00170                 }
00171                 catch (IOException e)
00172                 {
00173                     Log.d(TAG, "Erreur write()");
00174                     e.printStackTrace();
00175                 }
00176             }
00177         }.start();
00178     }
00179
00183     public void connecter()
00184     {
00185         new Thread()
00186         {
00187             @Override
00188             public void run()
00189             {
00190                 if (connecterSocket())
00191                 {
00192                     return;
00193                 }
00194                 // sinon reconnexion
00195                 creerSocket();
00196                 connecterSocket();
00197
00198             }
00199         }.start();
00200     }
00201
00206     private boolean connecterSocket()
00207     {
00208         try
00209         {
00210             if(socket == null)
00211                 return false;
00212             if (!socket.isConnected())
00213             {
00214                 socket.connect();
00215                 Message msg = Message.obtain();
00216                 msg.what = CODE_CONNEXION;
00217                 handler.sendMessage(msg);
00218
00219                 treception.start();
00220                 return true;
00221             }
00222         }
00223         catch (IOException e)
00224         {
00225             Log.d(TAG, "Erreur connect()");
00226             e.printStackTrace();
00227         }
00228         return false;
00229     }
00230
00235     public boolean deconnecter()
00236     {
00237         try
00238         {
```

```
00239             treception.arreter();
00240
00241             socket.close();
00242             Message msg = Message.obtain();
00243             msg.what = CODE_DECONNEXION;
00244             handler.sendMessage(msg);
00245
00246             return true;
00247         }
00248         catch (IOException e)
00249         {
00250             Log.d(TAG, "Erreur close()");
00251             e.printStackTrace();
00252             return false;
00253         }
00254     }
00255
00260     private class TReception extends Thread
00261     {
00262         Handler handlerUI;
00263         private boolean fini;
00264
00265         TReception(Handler h)
00266         {
00267             handlerUI = h;
00268             fini = false;
00269         }
00270
00271         @Override
00272         public void run()
00273         {
00274             Log.d(TAG, "TReception run() start");
00275             BufferedReader reception = new BufferedReader(new InputStreamReader(receiveStream));
00276             while (!fini)
00277             {
00278                 try
00279                 {
00280                     String trame = "";
00281                     if (reception.ready())
00282                     {
00283                         trame = reception.readLine();
00284                     }
00285                     if (trame.length() > 0)
00286                     {
00287                         Log.d(TAG, "run() trame : " + trame);
00288                         Message msg = Message.obtain();
00289                         msg.what = Peripherique.CODE_RECEPTION;
00290                         msg.obj = trame;
00291                         handlerUI.sendMessage(msg);
00292                     }
00293                 }
00294                 catch (IOException e)
00295                 {
00296                     Log.d(TAG, "Erreur read()");
00297                     e.printStackTrace();
00298                 }
00299
00300                 try
00301                 {
00302                     Thread.sleep(250);
00303                 }
00304                 catch (InterruptedException e)
00305                 {
00306                     e.printStackTrace();
00307                 }
00308             }
00309             Log.d(TAG, "TReception run() stop");
00310         }
00311
00312         public void arreter()
00313         {
00314             if (fini == false)
00315             {
00316                 fini = true;
00317             }
00318             try
00319             {
00320                 Thread.sleep(250);
00321             }
00322             catch (InterruptedException e)
00323             {
00324                 e.printStackTrace();
00325             }
00326         }
00327     }
00328 }
```

9.11 Référence du fichier Protocole.java

Classes

— class com.lasalle.io_trucks.Protocole

Paquetages

— package com.lasalle.io_trucks

9.12 Protocole.java

```
00001 package com.lasalle.io_trucks;
00002
00003 public class Protocole
00004 {
00008     public static final String EN_TETE = "$iotruck";
00009     public static final String DELIMITEUR_CHAMP = ";";
00010     public static final String DELIMITEUR_FIN = "\r\n";
00014     public static final int NB_PARAMETRES_CMD = 2;
00015     public static final String TRAME_CMD_CHAMP1_TRIANGLE = "T";
00016     public static final String TRAME_CMD_CHAMP1_GYRO = "G";
00017     public static final String TRAME_CMD_CHAMP1_ECLAIRAGE = "E";
00018     public static final String TRAME_CMD_CHAMP1_HAYON = "H";
00019     public static final String ON  = "1";
00020     public static final String OFF = "0";
00024     public static final int NB_PARAMETRES_REQUETE = 1;
00025     public static final String TRAME_REQUETE_STATE1 = "S1"; // triangle / gyrophares /
      éclairage de confort
00026     public static final String TRAME_REQUETE_STATE2 = "S2"; // hayon /  niveau de
     charge
00027     public static final String LEVE  = "1";
00028     public static final String BAISSE = "0";
00032     public static final String CHARGE_NORMAL = "0";
00033     public static final String CHARGE_ATTENTION = "1";
00034     public static final String CHARGE_SURCHARGE = "2";
00038     public static final String TRAME_SERVICE = "A"; // Alive / Acquittement
00039     public static final int NB_PARAMETRES_SERVICE = 1;
00040 }
```

9.13 Référence du fichier README.md

9.14 README.md

```
00001 \mainpage Le projet
00002
00003 \tableofcontents
00004
00005 Le projet **Io-Trucks** est un système numérique intégré à un véhicule industriel qui permet
      d'interagir avec ses accessoires et de visualiser les informations associées sur une application mobile.
00006
00007 \section section_tdm Table des matières
00008 - \ref page_README
00009 - \ref page_changelog
00010 - \ref page_about
00011 - \ref page_licence
00012
00013 \section section_infos Informations
00014
00015 \author Arthur Mathieu <mathieu.arthur.pro@gmail.com>
00016 \date 2020
00017 \version 0.2
00018 \see https://svn.riouxsvn.com/io-trucks/
00019
00020
00021 \page page_README README
00022
00023 [TOC]
00024
00025 # Projet {#projet}
00026
00027 ## Présentation {#presentation}
00028
```

```
00029 Le projet **Io-Trucks** est un système numérique intégré à un véhicule industriel qui permet :
00030
00031 * d'interagir avec ses accessoires et,
00032 * de visualiser les informations associées sur une application mobile
00033
00034 Le système « Io-Trucks » devra remplir les missions suivantes :
00035
00036 * déployer un triangle de signalisation fixé sur le toit du camion
00037 * signaler l'état d'une intervention (par feux de balisage et gyrophare)
00038 * piloter les éclairages périphériques (projecteur en périphérie du camion)
00039 * acquérir les données de fonctionnement (état du triangle, des éclairages, surcharge du véhicule,
       ...) du camion ;
00040 * assurer la transmission des données des états du camion via une communication sans fil ;
00041 * afficher les données de fonctionnement reçues du camion sur l'écran du terminal mobile ;
00042
00043 ## Informations {#informations}
00044
00045 \author Arthur Mathieu <mathieu.arthur.pro@gmail.com>
00046 \date 2020
00047 \version 0.2
00048 \see https://svn.riouxsvn.com/io-trucks/
00049
00050
00051 \page page_about A propos
00052
00053 \author Arthur Mathieu <mathieu.arthur.pro@gmail.com>
00054 \date 2020
00055 \version 0.2
00056 \see https://svn.riouxsvn.com/io-trucks/
00057
00058
00059 \page page_licence Licence GPL
00060
00061 This program is free software; you can redistribute it and/or modify
00062 it under the terms of the GNU General Public License as published by
00063 the Free Software Foundation; either version 2 of the License, or
00064 (at your option) any later version.
00065
00066 This program is distributed in the hope that it will be useful,
00067 but WITHOUT ANY WARRANTY; without even the implied warranty of
00068 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
00069 GNU General Public License for more details.
00070
00071 You should have received a copy of the GNU General Public License
00072 along with this program; if not, write to the Free Software
00073 Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
```

Index