Aide au fonctionnement du logiciel JumpC v1.0

Edited by DUCLOS François, ARONDEL Martin, HINCELIN Arnaud, BRIENT Nathan et LE ROUX Adrien

Table des matières

1. Introduction

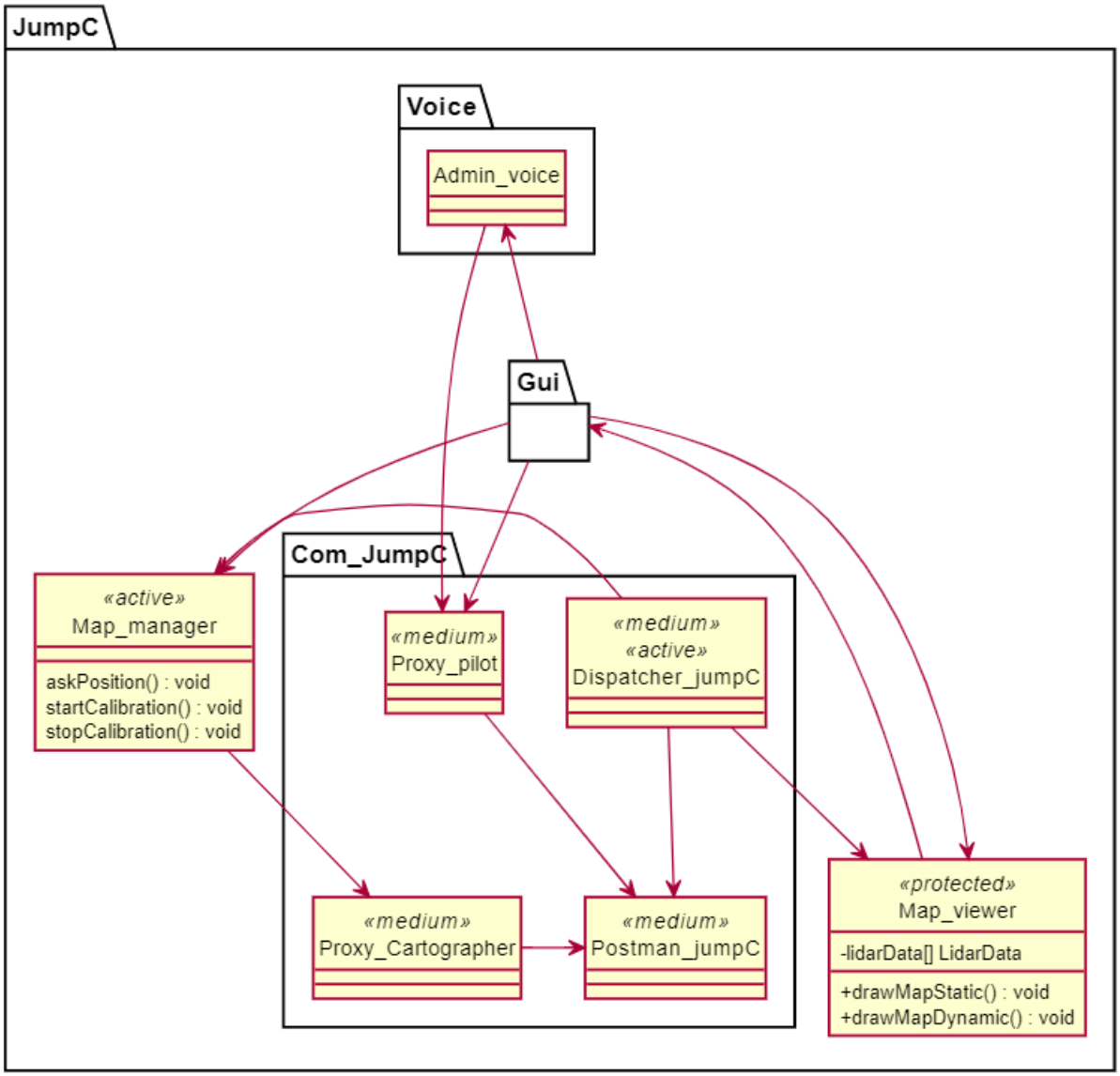
JumpC est une application Linux embarqué destiné à être exécuté sur carte STM32MP1 sur son cortex A7. Le but de cette application est d'interagir en temps réel avec Pocket, donc avec le robot en lui envoyant des directions par des appuis ou vocalement, ou bien d'avoir un affichage dynamique et statique des obstacles ainsi que la position en temps réel du robot. Le frontend dît GUI est designé à partir de la libgtk-3.0 qui est préinstallé sur l'OS Open ST Linux de chez STMicroelectronics. Cependant il est à noté que durant notre PFE 2021-2022, nous avons rencontré un problème de compatibilité de version entre la version de la licence Open ST et celle sur Mint/Debian, notamment au niveau des feuilles de styles CSS.

2.1 Prérequis

Afin d'utiliser JumpC dans ses fonctionnalités optimales, vous devez au préalable :

- Avoir installé un OS Linux sur la cible embarquée.
- Avoir installé le wifi sur votre carte STM32MP1 sur laquelle vous voulez exécuter le logiciel JumpC. (Voir Cf :)
- Programmer la carte afin de la connecter automatiquement lors de son démarrage à la deuxième carte STM32MP1 sur laquelle le logiciel Pocket est installé.
  Ce qu'il faut comprendre, c'est que sur la carte où est installé Pocket, l'appareil STM32MP1 joue le rôle de modem WIFI.
- Avoir Pocket en attente de connexion.
- Avoir préinstallé les librairies permettant d'utiliser la commande vocale (Cf :)
- Avoir exporté les dossiers JumpCSS, Images et les fichiers projetUI.glade du dossier JumpC racine et

2.2 Architecture logique



A noté que dans les noms donnés aux classes dans cette architecture sont différents de ceux donnés dans le code. En effet :

- **Gui** correspond au package **App**
- **Map_viewer** est dans le package **Mapping**

Gui contient 5 classes dont :

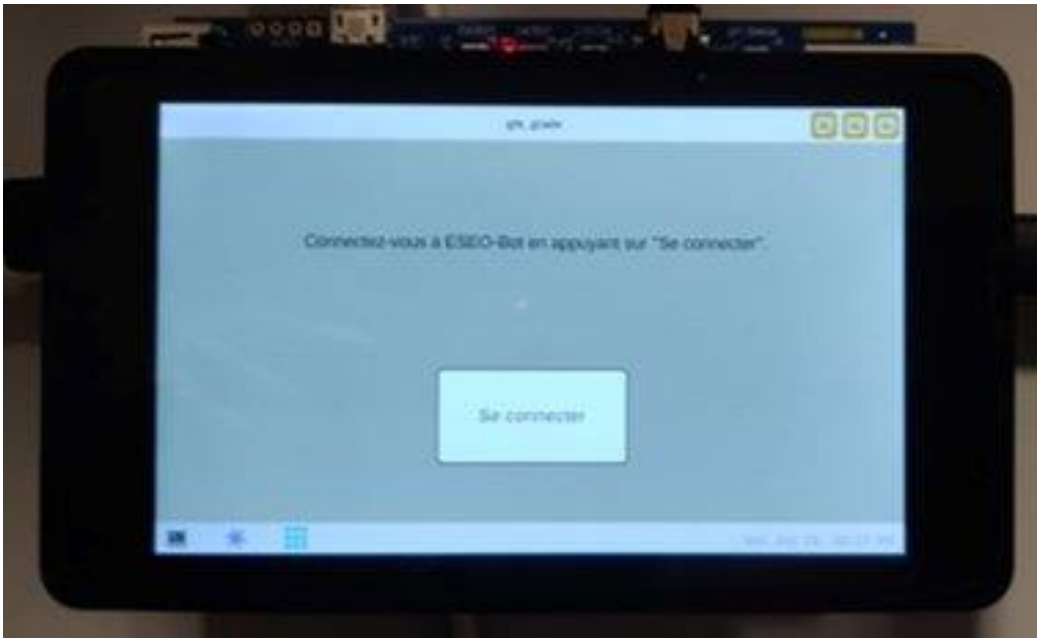
- **Splash_screen** : correspond à l'écran de démarrage avec le logo ST. Sur l'application, on a connu un problème avec le watchdog. Le but étant de montrer l'écran de présentation ST avec les noms des éditeurs au projet pendant 2 secondes s'est avéré être plus compliquer dû à un crash soudain de l'application. A corriger et à améliorer. On pensait sinon un tap sur l'écran pour passer à l'écran suivant.

- **Connection_screen** : correspond à l'écran de connexion qui interagit avec Pocket
- **Calibration_screen** : correspond à l'écran de calibration qui interagit avec Pocket
- **Main_screen** : correspond à l'écran principal qui comporte l'affiche d'obstacles lidar et de l'indoor positionning BLE ainsi que les commandes à envoyer aux robots.
- **Popup_screen** : représente les différentes fenêtres d'affichage popup.

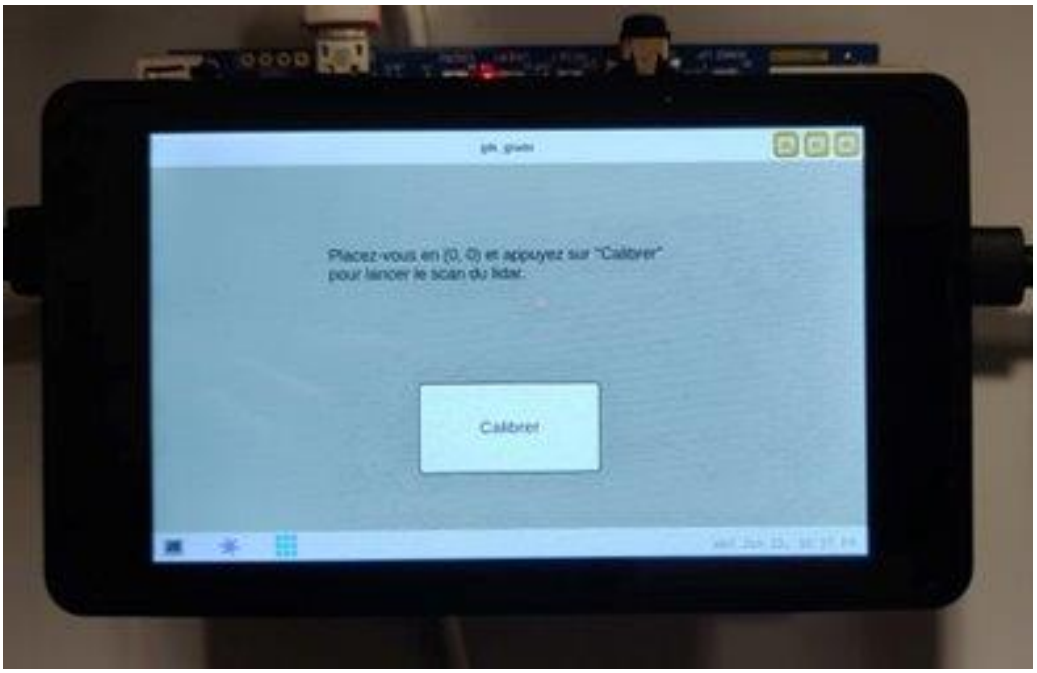
2.3 Frontend et design

Afin de pouvoir designer plus efficacement, nous avons utilisé un framework appelé **Glade** disponible sur OS Linux qui permet de traiter plus facilement notre design. Par la suite, le fichier est édité en format ressemblant au XML **dit .glade**. Ensuite, pour accéder lui appliquer des actions, il suffit d'appeler le fichier dans le code et de le stocker dans un **GBuilder**.

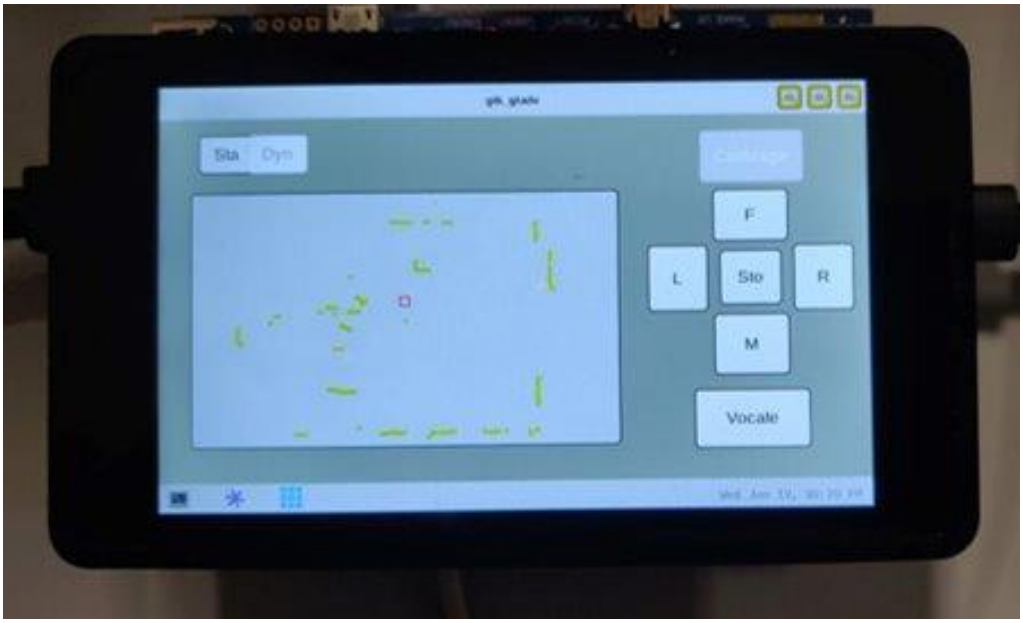
Connection Screen



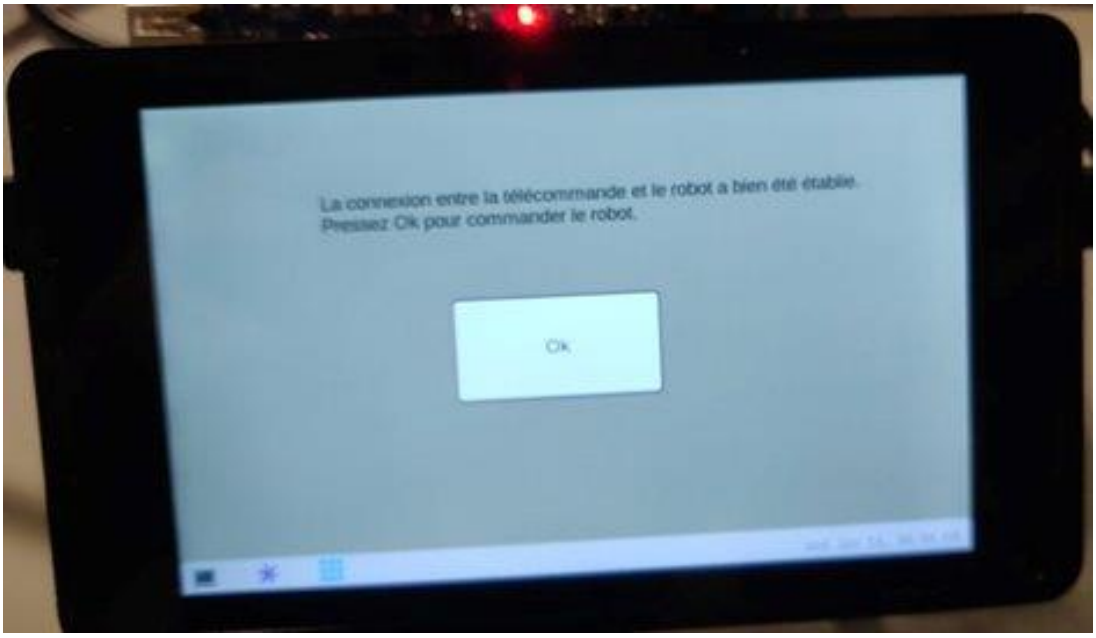
Calibration Screen



Main Screen



Popup Screen



2.4 Utilisation

Pour utiliser l'application, il faut tout d'abord avoir validé les prérequis du 2.1 et ensuite lancer l'exécutable qui s'appelle gtk_glade.

PS : Pour comprendre ce qu'il est possible de faire, allez consulter la **spécification** du PFE 2021-2022 qui regroupe le déroulement du scénario principal.

Après être lancé, le premier screen qui s'affiche est le « Connection Screen » que vous pouvez apercevoir ci-dessus. Après avoir appuyé sur connecter, un pop-up « Connection successful » apparait à l'écran. Il suffit de demander ok afin d'afficher l'écran « Main Screen ».

Arrivé sur cet écran, rien n'est calibré mais vous avez déjà la possibilité de commander le robot avec la télécommande par voix ou par boutons. Pour afficher les données sur l'écran, il suffit d'appuyer sur « Calibrage » à droite de l'écran qui vous emmène à l'écran « Calibration Screen ».

Calibration

Il suffit de placer le robot au (0, 0) du point de la pièce c'est-à-dire au beacon qui représente le point (0, 0) de la pièce.

Une fois fait, appuyer sur « Calibrer ». A ce moment-là, la télécommande envoie un signal à Pocket pour lui demander de démarrer et commencer son scan. Si la calibration a été réussite, alors une pop-up « Calibration successful » doit s'afficher à l'écran.

A noté que une fois sur 8, une erreur de librairie de type « Erreur Pango » s'affiche. Nous pensons que c'est un problème interne à la librairie mais à approfondir.

Cliquer sur ok et vous verrez apparaitre le « Main Screen » à nouveau.

Au bout de quelques secondes (< 7 secondes) les premières données lidar s'affiche à l'écran en statique (Mode mis par défaut au démarrage).

Après 10 secondes, la position du robot s'affiche à l'écran. A noté que l'écran est adapté en fonction de la pièce. (Existe ROOM1 et ROOM2)

L'utilisateur peut switcher sur le mode dynamique qui affiche la position centrale du robot au milieu de l'écran ainsi que les obstacles lidar avec un temps de rafraichissement de 1 seconde.

Le zoom est établi sur une technique tri. On tri les éléments reçus avec une technique appelé heapysSort qui est de time complexity nlog(n) (binary tree sort), on prend les 2 extrema en valeur absolu, et en fonction de la valeur la plus grande, on réalise un changement de repère.

Ce qui zoom les éléments qui sont affichés sur l'écran.

Changer de pièce

Lors du changement de pièce, il faut absolument appuyer sur le « toggle button » « Calibrage » afin de désactiver la calibration.

Ensuite replacer revenir à l'étape **Calibration.**

Au-dessus à droite de l'écran d'affichage devrait s'afficher le nom de la nouvelle salle.