SCOUTBOT : PFE 2021 – 2022

Etat d’avancement et Notice d’utilisation

Table des matières

A. Etat du projet SCOUTBOT 2021-2022

1) Mécanique du robot

Ajouts :

- Elévation du robot avec l'ajout d'une roulette à l'avant
- Agrandissement des roues
- Supports en bois pour tenir la STM32MP1 à la verticale (obligé pour le BLE)
- Fixation des objets (moteurs, drivers moteurs …)
- Supports en bois pour fixer le lidar

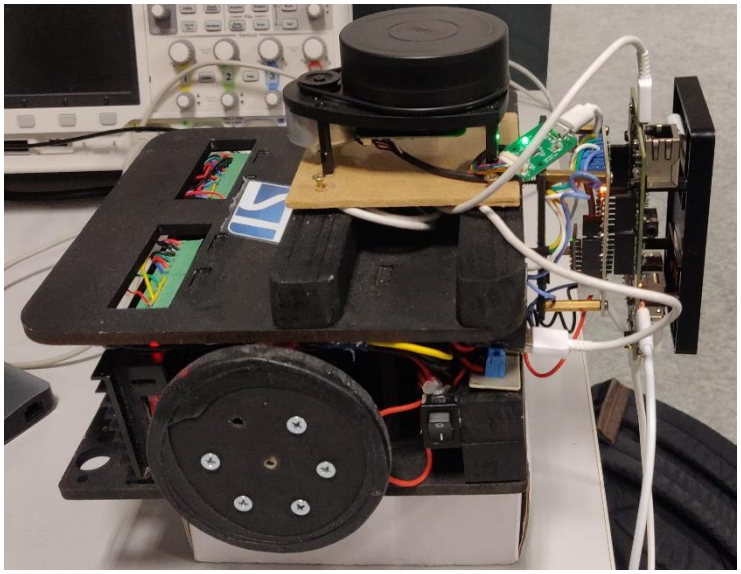


Figure 1: Mécanique du robot

Ce qu'il reste à faire :

- Acheter un RPLIDAR SLAMTEC A1 et le fixer (nous avons utiliser le notre en perso)
- Acheter une batterie (celle sur le robot est en fin de vie ...)
- Sécuriser le câble rouge qui sort de la batterie (prévu pour se connecter à la carte électronique pour la gestion de batterie) s'il n'est pas utilisé.
- Rajouter des capteurs de contacts

2) Electronique du robot

Ajouts :

- Création d'une nouvelle carte électronique se fixant sur les ports Arduino de la STM32MP1 à l'avant du robot.
- Carte électronique :
  → Sorties pour le driver moteur
  → Gestion de batterie
  → Ports pour ajouter des capteurs de contacts
  → Support pour utiliser un MPU6050 comme centrale inertielle

  → Schéma Electronique et PCB :

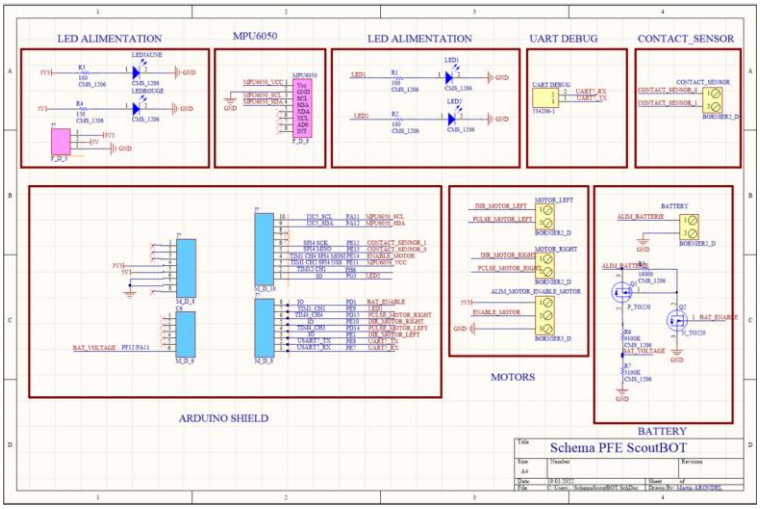


Figure 2 : Schéma Electronique

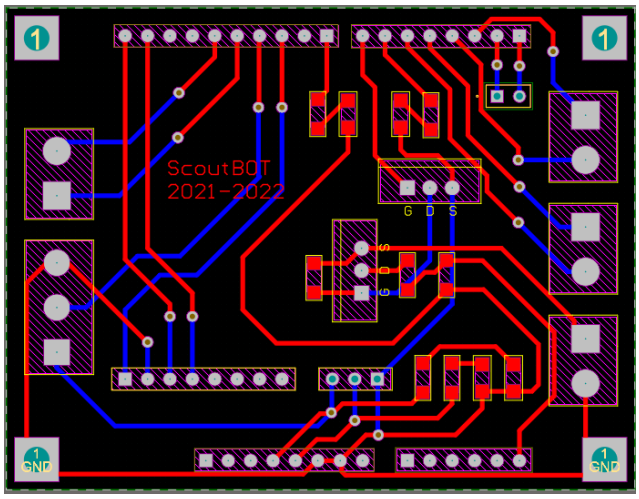


Figure 3 : PCB

Ce qu'il reste à faire :

- Souder les éléments à la gestion de batterie pour pouvoir lire la tension de sortie sur une broche analogique de la STM32MP1. (On n'a eu des problèmes software avec des conflits sur les broches en analogique lorsque le A7 fonctionne avec le M4).
- Ajouter des capteurs de contacts (ports prévus sur la carte)
- Acheter un chargeur de batterie (nous avons utilisé le notre en perso)

3) Software coté M4 RACE

Ajouts :

- Revue de la communication inter cœur (A7 et M4)
- Revue du code de gestion des moteurs (on n'utilise plus l'odométrie) juste une gestion classique en simulant une PWM sur les moteurs pas à pas.
- Ajout du code driver MPU6050
- Ajout d'un debug sur l'UART 7 ou l'UART 3 de la STM32MP1

Ce qu'il reste à faire :

- Finir de mettre en place la lecture en analogique de la gestion de batterie (code déjà écrit mais problème de comptabilité avec le cœur A7 du linux)
- Envoie des données de position du MPU6050 au cœur A7 afin de les coupler avec celles de l'indoor positionning (améliorer la position du robot dans la pièce).
- Ajouter le code pour la lecture des capteurs de contacts.

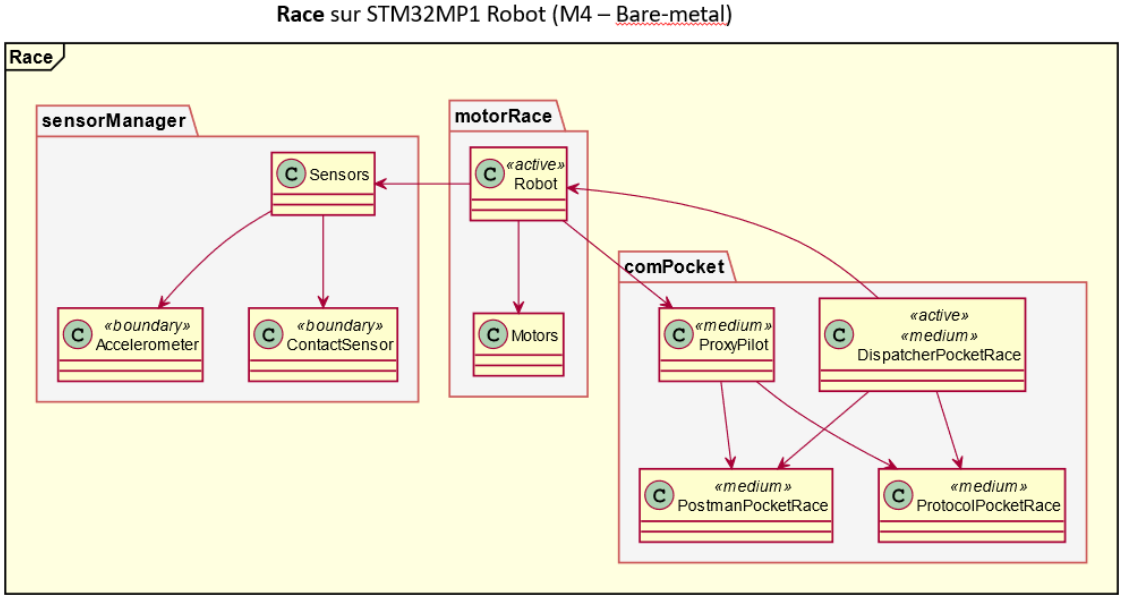


Figure 4 : Architecture Physique Race

4) Software coté A7 POCKET (robot)

Ajouts :

- Indoor Positionning
- Lecture des obstacles environnent avec le Lidar
- Communication avec le M4
- Communication Wifi avec JumpC dans les 2 sens
  → en envoie : données lidar et indoor positionning
  → en réception : commandes vocales
- Ordonnancement des taches : multi threading

Ce qu'il reste à faire :

- Améliorer la lecture des données lidar et de l'indoor positionning sans passer par la lecture d'un fichier texte (perte de temps dans l'affichage des données)
- Optimiser le code
- Ajouter la communication pour lire les données du MPU6050 sur le M4 et les coupler à l'indoor positionning.

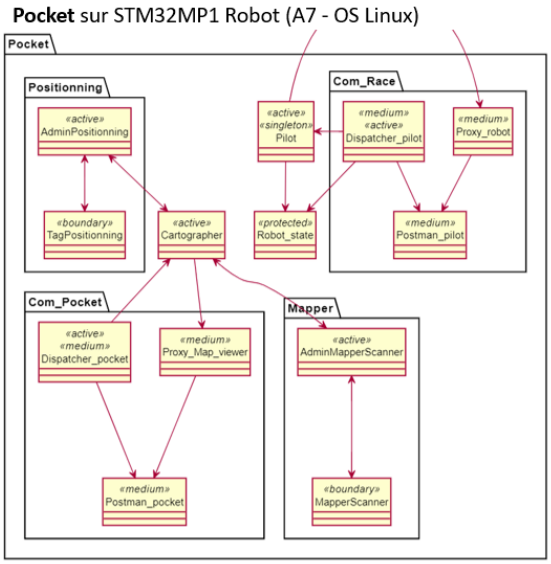


Figure 5: Architecture physique Pocket

5) Software coté A7 JUMPC (remote)

Ajouts : Création de l'application sur le STM32MP1 remote :

- Application réalisée avec GTK 3.0
- Mode d'affichage dynamique et statique
- Logiques de blocage
- Ordonnancement et multitâche
- Auto-gestion des demandes de données
- Connexion et envoie des données en Wifi
- Commande vocale
- Commande par boutons
- Zoom sur les obstacles en dynamique
- Adaptation de l'écran en fonction de la salle en mode statique
- Affichage de la salle

Ce qu'il reste à faire :

- Amélioration du design
- Reprendre le splash screen
- Améliorer la sécurité logicielle

- Poursuivre les tests unitaires
- Création d'un menu "Settings"
- Autres...

Problèmes rencontrés :

- CSS obsolète sur la carte, problème de compatibilité.

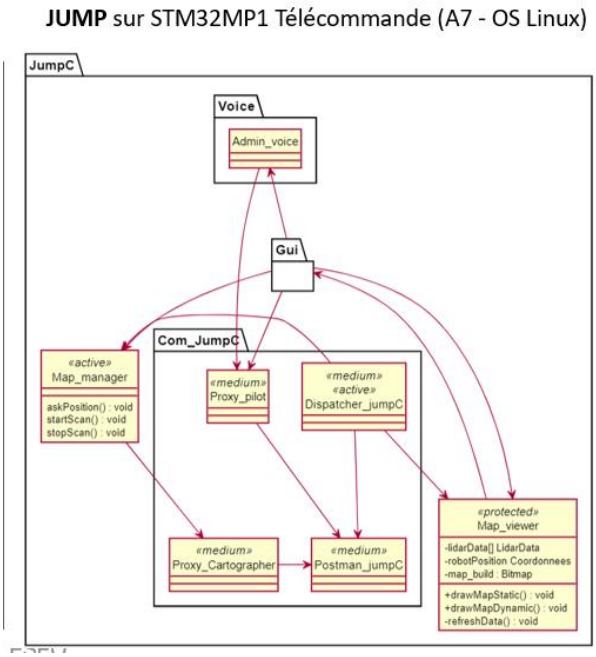


Figure 6 : Architecture physique Jump

6) Lancement du projet

Aujourd'hui nous devons lancer le projet à la main en exécutant les fichiers nous-mêmes.

Ce qu'il reste à faire :

Lancer l'exécutable *./pocket* directement au root de la carte. Pour que dès qu'on alimente la carte l'application se lance.

B. Notice d'utilisation SCOUTBOT

I. Coté ROBOT

1) Chargement Batterie

Au départ nous pensions que la batterie n'était pas équipée d'un BMS c'est pourquoi la batterie a été ouverte puis scotchée. Elle est donc protégée contre un chargement trop long. Cependant, nous avons aucune indication sur son taux de rechargement, à l'heure actuelle nous regardions au voltmètre. C'est pourquoi il y a le circuit de gestion de batterie sur la carte électronique. Nous n'avons pas eu le temps de le terminer.

Donc attention, charger la batterie avec un œil sur le voltmètre pour l'instant.

2) Positionnement de la carte sur le robot

La carte STM32MP1 est à la verticale sur le robot afin de garder un bon rayonnement de l'antenne.

Si elle est à l'horizontale elle ne pourra pas communiquer correctement avec les balises BLE autours d'elle.

3) Alimentation des modules électroniques

- La carte STM32MP1 s'alimente sur un des 2 ports USB fixé à l'avant du robot.

Ils fournissent du 5V pour alimenter la carte.

- Les moteurs sont alimentés directement à la sortie de la batterie en 12V.

4) Capteurs - Moteurs

- Le lidar se branche directement en USB sur la STM32MP1
- Le MPU6050 se branche sur la carte électronique
- Les câbles moteurs se branche sur la carte électronique

5) Drivers à installer si vous utilisé une nouvelle carte STM32MP1

- Driver CP210x (voir l'exploration lidar dans le projet pour l'installation)

6) Logiciel

Une fois le projet récupéré, il suffit de modifier le fichier *send_MP1.sh* dans le dossier package afin de choisir la destination du projet sur votre MP1.

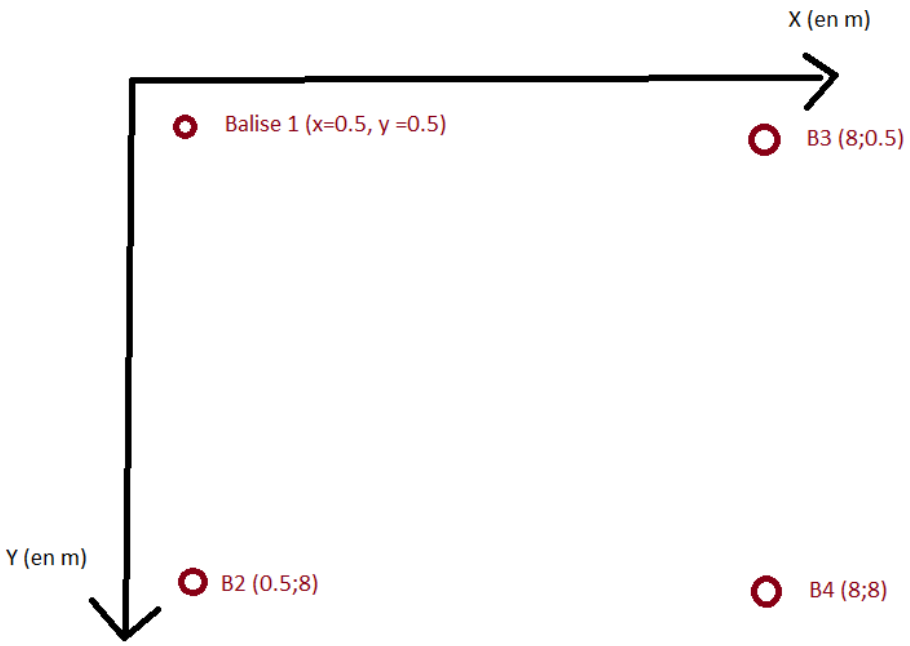
Ce fichier est conçu pour compiler et envoyer tous les fichiers nécessaires directement sur la carte en ssh.

Assuré vous que la connexion ssh fonctionne bien avant entre la carte et votre PC.

Une fois les fichiers envoyés sur la carte, exécuté le fichier*./pocket* et le programme va se lancer.

7) Positionnement des balises

Pour les balises, il faut les répartir dans les différentes pièces. Le minimum est d'avoir 3 balises BLE dans chaque pièce. Le plus optimisé est d'avoir 4 balises réparties aux 4 coins de la salle. Il faut un repère dans l'espace sur lequel on va placer nos différentes balises. Chaque balise aura sa propre position. Une balise ne peut pas être dans plusieurs pièces à la fois.



Il faut ensuite spécifier dans le code python les positions des différentes balises.

Sur le code python, balise 9,2,3,4 sont pour la pièce numéro 1 et 5,6,7,8 sont pour la pièce numéro 2.

Si vous ne trouvez plus les numéros sur les balises, vous pouvez les remplacer en scannant avec l'outil de scan BLE sur linux pour changer les adresses MAC dans le main du BLE.

Une fois cela fait vous pouvez lancer avec l'application JumpC le scan en statique.

II. Coté REMOTE

1) Alimentation de la télécommande

La télécommande est une STM32MP1. Elle s'alimente donc directement sur secteur avec son alimentation ou sur un PC.

2) Utilisation de la reconnaissance vocale

Lire le tutoriel sur le reconnaissance vocale, toutes les procédures à suivre y sont indiquées.

3) Logiciel

Il y a aussi un fichier *send_MP1.sh* qui envoie directement tous les fichiers compilés sur la carte. Il suffit juste de choisir la destination à l'intérieur de ce fichier bash.

Une fois les fichiers envoyées sur la carte, exécuté le fichier *./gtk_glade* et le programme va se lancer.

Attention : Il faut lancer Pocket avec Jump pour que la connexion se fasse entre les deux.