Robot - TP5 - Embarqué - Télémètre

Installation

Mergez la branche `tp4` dans la branche `main` et créez une branche `tp5`.

Commun

Comme pour le TP4 :

- Vous devrez définir le plus possible de valeur en tant que constante (`constexpr`) juste en dessous des `includes`.
- Vous pouvez créer autant de fichiers sources et/ou classes que vous jugerez utile.
- Sur le projet `robot-command`, il y a une branche pour contrôler le robot au clavier et visualiser les données renvoyées par les capteurs en utilisant la branche `motorKeyboard`. Voir dans dans le `README.md` les instructions pour le compiler.

```
git clone git@gitlab.cri.epita.fr:jeremie.graulle/ssie-robot-command.git
git checkout motorKeyboard
```

Etape 1 : Ajout du télémètre ultrason

En vous inspirant de l'exemple `esp-idf-cxx/examples/hc_sr04_cxx` vous devrez ajouter le télémètre à ultrason dans votre projet.

Vous devrez créer un thread dédié à ce capteur effectuant une mesure régulière et dans un premier temps, la loggant sur le port série (pour la voir sur la sortie `monitor`).

Vous devrez modifier la classe HcSr04 pour :

- Avoir un fichier d'entête et un fichier source.
- Éviter l'utilisation de types flottants dans la fonction `receive` et faire la conversion en utilisant uniquement des entiers de 32 bits pour diminuer la charge CPU et du coup retourner une valeur en mm.
- Ajouter en paramètre du constructeur le `groupIp` pour le membre `_capTimer(mcpwmGroupId)`.
- Modifier la queue `_rspQueue` pour ne considérer que la valeur la plus récente.
- Ajouter la Doxygen sur le header de la classe.

Ensuite, dans le thread du télémètre ultrason, vous devrez remplacer le log sur le port série par l'envoi de la notification `ultrasoundDistanceDetected` vers le client en utilisant la fonction `sendNotification()` du serveur Json RCP avec les paramètres :

- `index` : valeur entière du numéro du capteur en commençant par 0 (donc ici toujours 0)
- `value` : valeur entière mesurée par le capteur (en mm)
- `changedCount` : valeur entière auto-incrémentée, afin que le client puisse vérifier si des messages ont été perdus

Attentions:

- Pour les tests le plafond n'est pas une bonne surface réfléchissante pour les ultrasons on a des valeurs incohérentes, il vaut mieux le tester sur les murs ou le sol.
- `Us` dans le nom de la variable veut dire micro-seconde il faut donc changer le nom de la variable si vous changez l'unité.

Etape 2 : Ajout des interrupteurs

En vous inspirant de l'étape 5 du TP4, vous devrez envoyer le message `switchIsDetected` en Json avec les valeurs suivantes :

- `index` : l'index du capteur, ici 0 pour droite et 1 pour gauche.
- `value` : un booléen à vrai si l'interrupteur est enfoncé et faux sinon.
- `changedCount` : valeur entière auto incrémentée à chaque message.

Attention : Il faudra ajouter un anti-rebond logiciel (debouncing) pour envoyer un seul message à chaque changement d'état.

Etape 3 : Ajout du télémètre IR

Sur le robot, il faut déconnecter le capteur de suivi de ligne et brancher à la place le télémètre IR. Vous devrez envoyer le message `irProximityDistanceDetected` en Json avec les valeurs suivantes :

- `index` : l'index du capteur, ici 0
- `value` : valeur entière mesurée par le capteur (en mm)
- `changedCount` : valeur entière auto incrémentée à chaque message.

Il faudra créer une fonction de conversion de la lecture analogique brute vers la distance en mm, en établissant une table de correspondance (Lookup Table) en mesurant au moins une dizaine de points, puis, à partir de cette table faire une interpolation linéaire pour retourner une estimation de la distance mesurée en mm. De plus comme toutes les données sont connues à la compilation cette fonction devra être `constexpr`.