Compte rendu Audit Consultatif

AC sur le code C avec Frédéric JOUAULT / ProSE

Appel à l'ordre :

Date de la réunion : 12/05 **Heure début et fin :** 11h → 12h30 **Lieu :** Asie - B213

Réunion organisée par : Frédéric JOUAULT **Type de réunion :** Audit Consultatif Code C V1 **Rédacteur :** Thomas et Elisa **Point météo :** Très nuageux avec de la pluie	**Participants :** Elisa DECLERCK, Thomas ROCHER et Frédéric JOUAULT.

Objectif de la réunion :

Cet audit consultatif permet de faire un point poussé sur l'avancement du code C. Il nous permet d'avoir un retour complet et surtout des points d'améliorations pour progresser.

Sujets abordés et actions menées :

Elisa : Communication avec la Raspberry Pi	**Thomas** : Communication avec Android
• Retirer le module exemple du code de production. • Réorganiser le code du module Sender en renommant les fonctions et en définissant un type de trame. • Mettre à jour la conception générale du Sender en incluant le proxyGUI et en modifiant l'appel de sendFrame. • Effectuer des modifications dans le DriverCAN, le ProxyGUI, le ProxyLogger et le Postman pour améliorer la gestion des erreurs, la configuration et l'envoi de données.	• Améliorations dans le module "sender": renommer une fonction, modifier des paramètres et inclure des fichiers. • Révision de l'implémentation du module "sender" pour améliorer la gestion des threads, des erreurs et des communications. • Suggestions pour améliorer la conception détaillée, notamment en ce qui concerne les tailles de tableau et les identifiants de trame. • Améliorations proposées pour les modules "ProxyGUI" et "ProxyLogger", telles que la gestion

• Mettre en place la gestion de threads pour l'envoi et la réception, en s'assurant d'un appel adéquat du dispatcher.	des délimiteurs et l'utilisation de la fonction "snprintf". • Recommandations générales pour améliorer la qualité du code, notamment en ce qui concerne la documentation et l'utilisation d'assertions.

Commentaires :

Pastille orange/rouge, 70 % d'avancement

Prise de note complète de la réunion

Audit consultatif (CR) du 12 mai

Elisa : Communication avec la Raspberry Pi

Sender:
- Dans le fichier .h :
  - Renommer `sender_askSendingState` en `sender_AskSendingState`.
  - Préciser dans la conception générale que le paramètre `frame` devient `frames`.
  - Définir le type `Frame_t` dans un fichier `types.h`.
  - Expliquer dans la conception détaillée que le type `Frame_t` est incomplet car l'ID n'est pas nécessaire.

- Dans le fichier .c :
  - Inclure le fichier `proxyGUI.h`.
  - Mettre à jour la conception générale pour supprimer le lien entre Sender et Basket.
  - Renommer `perfomeAction` en `performAction`.
  - Noter que `StartSending` n'est pas réentrant en raison de l'utilisation de variables globales.
  - Modifier l'appel de `sendFrame` pour prendre en paramètre les trames à envoyer.
  - Remplacer la boucle infinie d'attente active par un sémaphore dans la fonction `send`.
  - Ajouter l'état `FORGET` dans l'enum pour ignorer explicitement les évènements qui ne doivent pas être traités.

- Implémenter soit comme un singleton, soit comme instanciable (pas de mélange des deux approches).
- Modifier les instructions `stop on error` pour retourner les codes d'erreur.
- Modifier la boucle infinie dans la fonction `run` pour recevoir un message de fin.
- Faire le changement d'état dans `performAction` plutôt que dans `run`.

DriverCAN:
- Dans le fichier .c :
- Déplacer la configuration dans la fonction `new`.

ProxyGUI:
- Dans le fichier .h :
- Déclarer `encodeMessage` comme méthode privée.
- Mettre les énumérations dans le fichier `types.h` pour partager les types entre la Pi et l'Android.

- Dans le fichier .c :
- Noter qu'il serait plus pertinent d'utiliser `snprintf` plutôt que `sprintf`.

ProxyLogger:
- Dans le fichier .h :
- Modifier `SetFrame` pour passer un pointeur.

- Dans le fichier .c :
- Noter qu'il serait plus pertinent d'utiliser `snprintf` plutôt que `sprintf`.

Postman:
- Dans le fichier .c :
- Supprimer la file d'attente de réception (`MQ`).
- Noter que l'appel à `accept` dans la fonction `start` est bloquant, ce qui peut poser problème (à résoudre en utilisant un thread).
- Souligner qu'il est possible d'écrire sur la socket avant qu'elle ne soit acceptée.
- Ajouter une boucle pour envoyer les octets manquants dans la fonction `send`.

Dispatcher:
- Dans le fichier .h :
- Déplacer le type dans le fichier `types.h`.
- Déclarer la méthode `decole` comme privée.

- Dans le fichier .c :
- Noter que la fonction `start` est bloquante et qu'il faut utiliser un thread.

Général:
- Noter qu'il est nécessaire d'avoir deux threads, un pour l'envoi et un pour la réception. On peut choisir entre un thread pour le postman et un pour le dispatcher, ou bien deux threads pour le postman et appeler le dispatcher.

Thomas : Communication avec Android

Enlever le module exemple de la branche develop.

Sender:
- Majuscule `askSendingState()` -> `AskSendingState()`.
- Dans la fonction `startSending`, le paramètre `frame` devrait être `frames`.
- Le fichier `sender.c` doit inclure `proxyGUI.h`.
- `performe` -> `perform`.
- Soit on utilise une structure `MsQ_u` et du multi-instance, soit on utilise un singleton. Utiliser une variable globale pour passer le message dans `sendFrame`. `sendFrames` devrait prendre `this` en multi-instance.
- Dans `sender_StartSending`, il faudrait mettre le pointeur dans la file d'attente plutôt que de passer les paramètres via des variables globales. Rendre cela réentrant.
- `void *send`, `exit(FAILURE)` = top level d'un `write`.
- Changer `while(1)` (`sv_keep_sending`) en sémaphore.
- Remplacer l'attente active par un sémaphore.
- Mettre `S_forget` dans la machine à états.
- S'il n'y a pas de `S_FORGER`, faire un `unlink` vers l.124 ou l.74.
- Il y a des stops en erreurs.
- Retourner un code d'erreur (pas de gestion globale).
- Implémentation de la machine à états, changement d'état dans `performeActions` et non dans...

Dans la conception détaillée :
- Quand on passe une taille en paramètre, elle doit obligatoirement faire référence à la taille du tableau.
- Indiquer que c'est normal si notre frame n'a pas d'ID car c'est pour CANdroid et non CANgateway.

Dans la conception générale :
- Mise à jour car plus d'appel de `sender` vers `Basket`.
- Il faut rajouter les d'Elisa.

ProxyGUI:

- `encode` en privée, donc pas besoin de le mettre dans le `.h` (le faire partout).
- Il manque un `types.h` pour la liaison, les types de données.
- Utiliser un délimiteur pour terminer.
- `setFrame()` doit passer un pointeur en paramètre et pas juste une frame.

ProxyLogger:
- Utiliser `snprintf` et dans le dernier `snprintf`, tu peux...

Qualité :
- La documentation devrait plutôt être dans le point c que dans le point h.

Postman:
- Faire un thread pour le dispatcher et enlever le thread pour le `receive` dans le postman.
- Tant qu'on n'a pas initialisé `socket_donnee`, on ne peut pas commencer le thread car si un proxy fournit déjà une information mais que la socket n'est pas encore acceptée, tu l'envoies dans le vide.
- Le `start` démarre un thread qui attend et dès qu'il y a la connexion.
- Le postman fait le `start` du dispatcher quand il y a la connexion et s'arrête s'il n'y a plus de connexion, etc.
- Dans le `stop`, faire les joins avant les `close(socket_donnee)` et `close(ecoute)`.
- Le dispatcher appelle directement le `read` qui appelle directement le `read` de la socket.
- Il manque une boucle pour l'envoi des octets non encore envoyés par le `send`.
- Pour le `receive`, tu peux utiliser `waitAll`.

Dispatcher:
- Encode et décode.
- Envoyer `send` et le message `stop`.
- Faire un thread dans le `start`.
- Conflit de point h pour dire que c'est 50.
- Quand tu `read`, tu dois savoir combien d'octets tu lis.
- Quand on fait un `read`, on peut recevoir plusieurs messages en même temps ou la moitié d'un message.
- `recv` indique combien il a lu, et pourtant tu ne le vérifies même pas dans `readMessage()`.
- Il n'est pas possible d'utiliser `waitAll`, donc ne pas le prendre en compte.
- Avoir un caractère de fin pour indiquer la fin du message (un saut de ligne plutôt). Les sauts de ligne séparent les messages.
- Pour les données dans le décodage, si on a un octet nul, il faut arrêter la lecture.
- Faire toute la gestion d'erreur pour le décodage. Les appels à `strchr` ne sont pas nécessaires lorsqu'il y a un appel à `strtol`.
- Ajouter des `asserts`.