MASTER OF SCIENCE IN ENGINEERING

Hes·so

Projet DeseNET

Département : EIE
Unité d'enseignement : MA_DeSEm

Auteur Sébastien Deriaz
Professeurs Medard Rieder
Assistants Thomas Sterren

Date 23 décembre 2021

Table des matières

1 Introduction

L'objectif de ce travail est de mettre en œuvre le protocol DeseNET sur une carte STM32 Nucleo. Le protocole a été rédigé par M. Clausen et nous a été fourni pour ce travail.
Un projet de base nous a été fournit et il conviendra de le remplir pour implémenter les différentes fonctions manquantes.
Le projet sera séparé en deux parties

1. Simulation : Test du programme dans un environnement simulé
2. Implémentation sur la carte : Test du programme sur la carte Nucleo équipée d'un module de communication sans-fil

2 Fichiers crées

1. `joystickapplication.h` et `joystickapplication.cpp` : Machine d'état du joystick et polling de l'état
2. `mpdu.h` et `mpdu.cpp` : Classe héritée de `frame` qui implémente les méthodes pour créer une frame MPDU (stockage des événements / sample values et headers spécifiques).

3 Modifications apportées aux fichiers

1. `platform_config.h` (`ide-stm32cubeide\platform\nucleo-stm32l476rg\platform-config.h`)
   (a) Changement du `DESENET_SLOT_NUMBER`
2. `factory.h` (`src\app`)
   (a) Ajout du joystick (plus include pour `joystickapplication.h`)
3. `factory.cpp` (`src\app`)
   (a) Include de `joystick.h`
   (b) Initialisation du joystick (+ `setObserver`)
   (c) Start Joystick / joystickApplication
   (d) méthodes `joystickApplication()` et `joystick()`
4. `abstractapplication.cpp` (`src\common\mdw\desenet\sensor`)
   (a) Ajout de `subscribeToSvGroup()` dans `svPublishRequest`
   (b) Ajout de `eventReceived()` dans `evPublishRequest`
5. `abstractapplication.h` (`src\common\mdw\desenet\sensor`)
   (a) Ajout `const` a l'event id
6. `net.cpp` (`src\common\mdw\desenet\sensor`)
   (a) Ajout de `slotNumber` à l'initialisation de `networkentity`
7. `joystick.cpp` (`src\common\platform\nucleo-stm32l476rg\board`)
   (a) Ajout du namespace `board`
8. `joystick.h` (`src\common\platform\nucleo-stm32l476rg\board`)
   (a) Ajout du namespace `board`
9. `joystick.cpp` (`src\common\platform\qt-meshsim\board`)
   (a) Ajout du namespace `board`
10. `joystick.h` (`src\common\platform\qt-meshsim\board`)
    (a) Ajout du namespace `board`
11. `networkentity.h` (`src\common\mdw\desenet\sensor`)
    (a) Ajout de l'include pour `mpdu.h`
    (b) Héritage de `ITimeSlotManager` + Ajout de la méthode virtuelle `onTimeSlotSignal`
    (c) Ajout de paramètre `slotNumber` à initalize
    (d) Ajout des méthodes `subscribeToSvGroup` et `unsubscribe`
    (e) Ajout de la méthode `eventReceived`
    (f) Liste des applications et struct `AppBind`
12. `networkentity.cpp` (`src\common\mdw\desenet\sensor`)
    (a) Ajout de `slotNumber` dans l'initialisation
    (b) Initialisation des relations de `pTimeSlotManager`
    (c) Complétion de la méthode `onReceive`
    (d) Ajout de la méthode virtuelle `onTimeSlotSignal`
    (e) Ajout des méthodes `subscribeToSvGroup` et `unsubscribe`
    (f) Ajout de la méthode `eventReceived`

4 Fonctionnement

La station de base (que l'on programme) doit être capable, lorsqu'elle reçoit une trame spéciale (beacon), de renvoyer les données qu'elle a collecté. Ces données sont :

1. Événements (par exemple un mouvement sur le joystick)
2. État mesuré (par exemple l'accéléromètre embarqué)

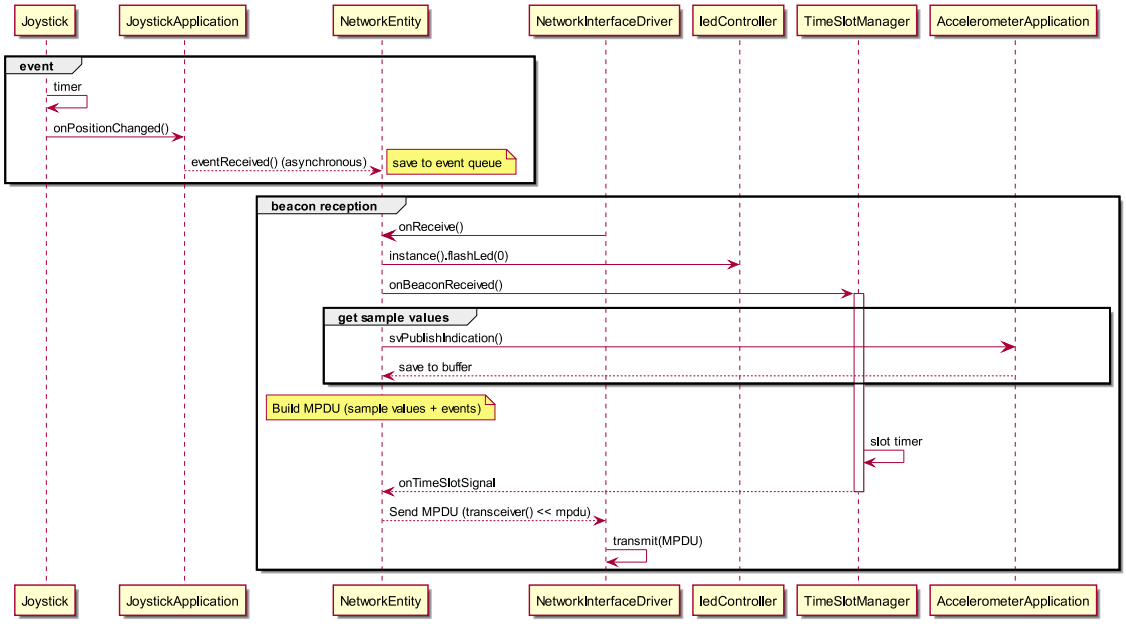

FIGURE 1 – Diagramme de séquence

4.1 MPDU

La création d'un MPDU se fait de la manière suivante :

1. Reset
2. Ajout des sample values (accéléromètre)
3. Ajout des événements (jusqu'à ce qu'il n'y en ait plus ou que le buffer soit plein). Effectuer un "commit" après chaque événement
4. Finalisation

Une fois qu'un MPDU est prêt, il est envoyé au moment où le temps du slot est atteint (slot timer).

5 Tests

Les tests seront réalisés en deux parties

1. Test avec le simulateur
2. Test sur la carte

Le simulateur permet de trouver rapidement les premières erreurs et de mettre en place le fonctionnement du récepteur ainsi que l'envoi des trames.
Les tests sur la carte permettent de finaliser le système et de détecter les erreurs liées au système embarqué (au lieu du système simulé).
Les tableaux suivants montrent l'état du système dans sa version finale.

5.1 Simulation

Description	État	Remarque
Beacon		
Réception du beacon (envoi d'un message dans le port série lorsque le beacon est reçu)	✓	
Led allumée (clignotement de la led lorsque le beacon est reçu. Ce test permet de valider le précédent sans utiliser le port série)	✓	
Joystick		
Test de flanc (mettre le testbench en mode "continuous" et appuyer sur le bouton sans le relâcher). On devrait observer le bouton correspondant s'allumer puis s'éteindre au beacon suivant le relâchement)	✓	
Test de flanc sur tous les boutons (un seul à la fois)	✓	
Test d'appui puis relâchement dans un seul cycle (le bouton ne devrait pas s'allumer dans le testbench)	✓	
Valeurs correctes lors du remplissage de la queue d'événements (appuis à répétition sur le joystick lors d'un cycle)	✗	Comme la queue d'événements est effacée à chaque envoi des données, il est normal que des données soient perdues si trop d'événements ont étés enregistrés.
Accéléromètre		
Test en déplaçant la fenêtre en haut à droite et en haut à gauche (les valeurs de l'accéléromètre devraient changer et leurs valeurs doivent être répétables en fonction de la position de la fenêtre)	✓	
svPDU		
Formation correcte du header (vérifier si il n'y a pas d'erreurs dans le testbench)	✓	
Données correctes (affichage des données dans le testbench)	✓	
evPDU		
Formation correcte du header (vérifier si il n'y a pas d'erreurs dans le testbench)	✓	
Données correctes (affichage des données dans le testbench)	✓	
Transmission		
Aucune erreur lors de l'envoi d'un svPDU	✓	
Aucune erreur lors de l'envoi de svPDU et d'un ou plusieurs evPDU	✓	
Envoi des données au bon moment (timeslot)	✗	Erreurs occasionnelles due à la transmission entre les applications et au système d'exploitation

5.2 Carte

Description	État	Remarque
Beacon		
Réception du beacon (envoi de données dans le port série lorsque le beacon est reçu) [1]	✓	
Joystick		
Test de flanc (mettre le testbench en mode "continuous" et appuyer sur le bouton sans le relâcher). On devrait observer le bouton correspondant s'allumer puis s'éteindre au beacon suivant le relâchement)	✓	
Test de flanc sur tous les boutons (un seul à la fois)	✓	
Test d'appui puis relâchement dans un seul cycle (le bouton ne devrait pas s'allumer dans le testbench)	✓	
Valeurs correctes lors du remplissage de la queue d'événements (appuis à répétition sur le joystick lors d'un cycle)	✗	voir commentaire dans **??**
Accéléromètre		
Positionnement de la carte sur son côté / à l'envers (les trois mesures doivent s'échanger les valeurs en fonction de l'orientation de la carte à condition qu'elle soit posée "droite")	✓	
Vérifier que les valeurs de l'accéléromètre sont cohérentes avec les autres cartes des collègues (toutes les mesures doivent être quasi-identiques à condition que les cartes soient orientées de la même manière)	✓	
svPDU		
Formation correcte du header	✓	Voir **??**
Données correctes	✓	
evPDU		
Formation correcte du header	✓	
Données correctes	✓	
Transmission		
Aucune erreur lors de l'envoi d'un svPDU	✓	
Aucune erreur lors de l'envoi de svPDU et d'un ou plusieurs evPDU	✓	
Envoi des données au bon moment (timeslot)	✗	Erreurs occasionnelles dues à des pertes (perturbations du WiFi) de paquets et/ou des erreurs de timing dans la station de base et/ou les stations subordonnées.

1. Vérifié avec un envoi sur le port série