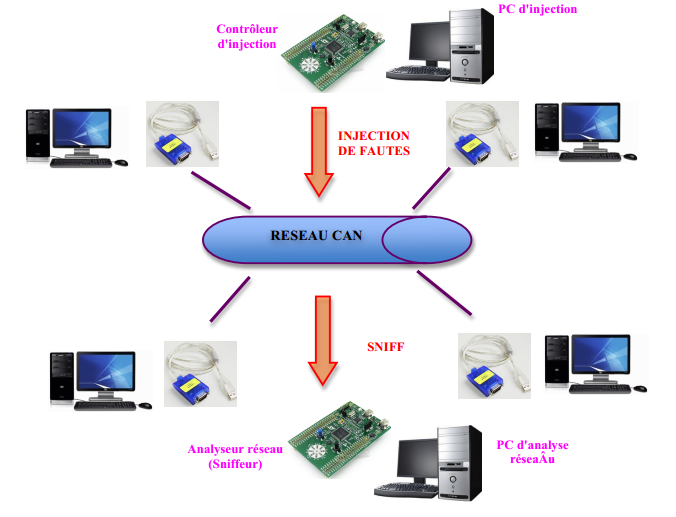
**Réseau et bus de terrain**

**PROJET AUTOMATIQUE ET RESEAU**

**ETUDE DU RESEAU CAN**

**Communication temps réel critique sur CAN**

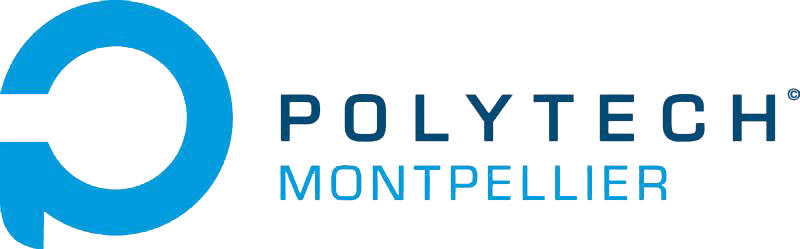
**Partie Network Analyzer (Sniffer)**



**Elie Faes**

**Adrien Peyrouty**

**Alexandre Lazareth**



**SE5 2014/2015**

**Table des matières**

[Introduction 3](#_Toc414190455)

[Analyse d’une trame 5](#_Toc414190456)

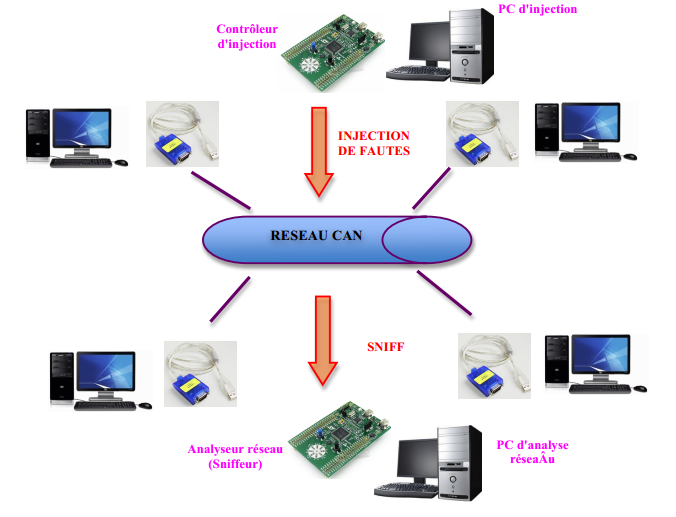
[Modélisation 6](#_Toc414190457)

[Implémentation 7](#_Toc414190458)

[Affichage des informations 8](#_Toc414190459)

# Introduction

Le but est d'étudier de manière approfondie les caractéristiques du réseau CAN dans le contexte d'une application embarquée, temps réel, critique et bien sûr communicante. Il s'agira donc d'étudier le comportement temporel du réseau sous contraintes réalistes par rapport au contexte, en comportement nominal mais aussi lors de l'occurrence de faute. Pour se faire, une partie du projet consistera en la conception d'un sniffer et d'un injecteur de fautes, afin de pouvoir étudier précisément des scénarios spécifiques. L'ensemble du projet doit être traité avec des outils propres à un projet : collaboration entre groupes et donc organisation et communication, méthodologie de conception (réfléchir avant d'agir..)



Le travail va être divisé en trois parties :

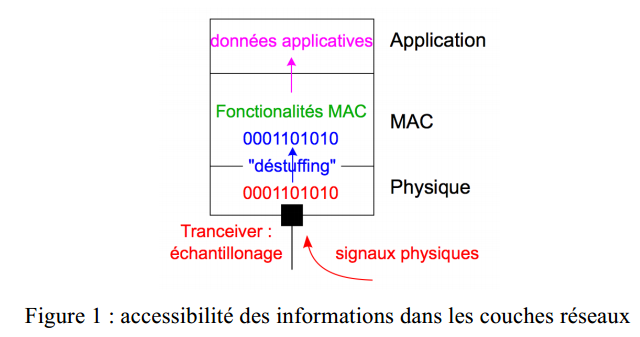
-Un groupe va travailler sur « l’analyseur réseau »

-Un autre groupe sur le « transceiver »

-Et le dernier sur l’injecteur de fautes

Ce rapport porte sur la partie « Analyseur réseau ».

L'idée d'implémenter nous même un analyseur réseau (sniffeur) CAN vient du fait que les contrôleurs MAC ne donnent en général accès qu'aux informations remontées aux couches supérieures, après les filtres des fonctionnalités MAC (en rose sur le figure 1). L'idée est pouvoir observer ce qui n'est pas observable avec les contrôleurs CAN traditionnels et donc d'observer et analyser les informations du très bas niveau, directement extraites de la couche physique (informations rouges et/ou bleues).



Par exemple dans le cas de trame normale, le sniffer devra pouvoir récupérer les bits de la trame transmise et reconstituer la trame reçue, comme un contrôleur CAN, mais il pourra aussi offrir la possibilité d'observer l'ensemble des bits transmis sur le réseau y compris les bits de bit stuffing. Mais il s'agit surtout de pouvoir observer des scénarios plus complexes impliquant des trames purement protocolaires, par exemple un scénario de détection et de signalisation d'erreur.

# Analyse d’une trame

**I Présentation du matériel**

CANUSB, avec le logiciel CAN Monitor Pro. Cette maquette permet aisément d’injecter et de recevoir des trames sur le réseau, et d’observer globalement ce qui se passe.

Vous avez également à votre disposition 2 éléments permettant de faire du "sniff" de trame CAN au  
niveau physique :

* Oscilloscope Agilent, à brancher sur le réseau pour l'observation du signal. Cet oscillo permet  
  la détection du début des trames CAN et la mise en forme du signal en signal carré.  
  Ces deux éléments ont dans votre cas la même utilité fonctionnelle.

**II Prise en main**

**A. Théorie**  
Il est nécessaire de connaître le fonctionnement du protocole CAN. Les cours de réseaux sont une  
première base, qui doit être complétée par les documents de spécifications officielles (site info indus).

**B. Pratique : premières communications avec les CANUSB**  
CAN Monitor Pro (CANUSB) permet de récupérer un certain nombre d’informations intéressantes.  
Tester quelques scénarios « normaux ».

**a) Etablir des scénarios de communications normales avec les CAN USB.**Etudier les différentes possibilités offertes.  
🡺Faire des captures des trames circulant sur le réseau pour les différents types de scénario mis en  
œuvre. (Les scénarios basiques n'ont pas besoins d'être mis dans le rapport.)  
*Attention : tous les nœuds communiquant sur un même réseau doivent partager les mêmes paramètres  
(fréquence, version, etc..)*

**III Tolérance aux fautes, théorie**

**A. Liste des fautes**  
CAN est un protocole réseau dédié à des systèmes nécessitant un certain degré de tolérance aux fautes.  
1) Faire la liste exhaustive de toutes les fautes pouvant perturber un réseau CAN.  
2) Pour chacune d'entre elle, imaginer comment les générer / simuler avec le matériel disponible.

**B. Gestion des fautes**  
3) Etudier les mécanismes permettant au contrôleur CAN de détecter et comptabiliser ces erreurs.  
Etudier pour chacune des fautes comment elles sont détectées, à quel moment du processus  
d'émission / réception.  
4) Etudier le comportement des compteurs d'erreurs.  
5) Etudier les différents états d’un contrôleur CAN.

*Utilisez comme documents le cours et les spécifs officielles. N’hésitez pas à demander des détails*

**IV Tolérance aux fautes, scénarios d'erreurs**  
Matériel nécessaire : CAN USB et CAN BUS Analyzer

**A. Scénario1 : priorité des messages**  
Etablir un scénario afin d’observer la concurrence entre des flux de messages prioritaires et d’autres  
non prioritaire.  
  
🡺Qu’observez vous et pourquoi ? Quel scénario avez vous mis en œuvre ? (rapport)

**B. Scénario2 : erreur de réception**  
Etablir un scénario afin d’observer des erreurs de réception sur l’analyseur de bus. Observer en  
particulier les valeurs des différents compteurs d’erreur et la valeur de l’état du contrôleur CAN.  
🡺Qu’observez vous et pourquoi ? Quel scénario avez vous mis en œuvre ? (rapport)

**C. Scénario3 : erreur d’émission**  
Idem mais pour l’observation d’erreur d’émission. Vous pouvez aussi observer le retour en état normal  
(décrémentation du compteur d’erreurs).  
🡺Qu’observez vous et pourquoi ? Quel scénario avez vous mis en œuvre ? (rapport)

**D. Scénario4 : BUS OFF**  
Essayez de mettre en Bus Off les CANUSB. Faites de même pour l’analyseur.  
🡺Comment avez vous procédez, qu’avez vous observez ? Pourquoi ? (rapport)  
Essayez, après être passé en Bus Off, de revenir en Error active.  
🡺Comment avez vous procédez, qu’avez vous observez ? Pourquoi ? (rapport)

*RMQ : Dans cette partie c’est l’évolution des compteurs et de l’état des contrôleurs qui vous  
donneront les indications suffisantes pour une compréhension globale des scénarios. Comparez avec  
la théorie pour comprendre quels scénarios mettre en œuvre et/ou comprendre les comportements que  
vous observez.*

**V Capture de trames au niveau physique**  
Matériel nécessaire : CAN USB (ou Bus analyzeur) et oscilloscope Agilent (ou analyseur logique IKAlogic).

**A. Théorie**  
*1) Capture bas niveau*  
Les différents « sniffer CAN » disponibles ne permettent malheureusement pas de montrer les  
informations très bas niveau, comme le bit stuffing, les retransmissions ou les trames d’erreurs.  
Pour avoir les détails précis du fonctionnement du protocole jusqu’au plus bas niveau, on va donc  
utiliser l'oscilloscope Agilent ou l'analyseur logique IKAlogic pour capturer directement le signal à la  
source, sans filtrage logiciel au niveau MAC.  
*2) Niveau physique*  
Les caractéristiques de la couche physique du protocole CAN sont normalisées par plusieurs  
normes, dont la plus utilisée est l'ISO 11898-2.  
Vous pourrez trouver un résumé fourni par microchip dans son Applicative Note AN228.

**B. Configuration de l’oscillo**

Utilisez tout d'abord l'oscillo en mode auto afin de bien le configurer sur un signal simple (par  
exemple une émission périodique entre les CAN USB)  
2) Puis configurer l'oscillo pour qu'il détecte le début des trames CAN :  
L'oscillo connaît un certain nb de motifs prédéfini permettant de se synchroniser sur le démarrage  
d'une séquence désirée. Ici, on va utiliser le SOF de CAN pour détecter le début d'une trame.  
Remettre le menu Edge en mode normal (attention, ne pas laisser en auto)

* Configurer le menu More sur CAN, puis Trigger sur SOF
* Config du trigger :
* Source : source 1, 2, 3 ou 4
* Baud : configurer le même débit que celui utilisé sur CAN
* Sample point : 75% (?)
* Signal type : CAN\_H ou CAN\_L (inversion sur la voie utilisée : i n v e r t)

**C. Etude d’une trame normale avec Bit Stuffing**Après avoir réglé l’oscillo et compris le processus de capture de trames, utilisez-le afin de comprendre  
et décoder le signal que vous observez, et mettre en évidence le mécanisme de bit stuffing.  
*🡺Capturez une trame et identifiez-en les différents champs, ainsi que les bits de bit stuffing  
(rapport)*

**VI Tolérance aux fautes, étude d'un scénario d'erreur (extension)**  
Vous allez ici essayez d'étudier un scénario d'erreur en essayant de suivre toutes les étapes de gestion  
d'erreur du protocole CAN. L'observation se fera obligatoirement au niveau physique, ces  
informations n'étant pas remontées par le contrôleur CAN.

**A. Mise en œuvre du scénario**  
3) Choisissez un scénario précis que vous pouvez simuler, et qui est censé générer des trames  
d'erreur.  
4) Expliquez son fonctionnement théorique, étape par étape.  
5) Décrivez les trames d'erreurs devant être générées en théorie.

**B. Capture bas niveau du scénario**  
6) Puis utilisez l'oscillo pour essayer de capturer les trames d'erreurs sur le réseau.  
Attention, réfléchissez à la bonne configuration de l'oscillo

# Modélisation

La première phase du projet consiste à réaliser l'étude amont des fonctionnalités du sniffer. Une phase de réflexion sera suivie de la modélisation de ces fonctionnalités sous la forme de machines à états. La modélisation permettra de fournir la structure du code à implémenter, et permet d'assurer que cette structure est fonctionnellement correcte grâce à des méthodes de vérification du modèle. (Décomposition fonctionnelle efficace pour l'implémentation, mais reflétant également le découpage en couches réseau (modèle OSI)).

Pour se faire, nous allons modéliser le fonctionnement du sniffer grâce aux réseaux de Petri et le logiciel TINA.

Dans un premier temps nous avons réalisé un modèle du fonctionnement normal (sans erreur) :

Ensuite nous avons modifié le modèle en intégrant les fautes injectées sur le réseau :

# Implémentation

La deuxième phase consiste en l'implémentation pure, c'est à dire le codage.

La cible matérielle est laissée libre. Il s'agira toutefois de s'accorder avec les autres groupes (en particulier celui gérant l'échantillonnage) afin de pouvoir être compatible. Il faudra également répartir les développements : les scénarios de base seront attribués au groupe gérant l'échantillonnage, tandis que le groupe sniffeur s'attaquera aux scénarios spécifiques.  
Il faudra ici mener une réflexion, en collaboration avec le groupe échantillonnage, de la répartition de  
la charge de stockage et de calcul de la cible matérielle. Des fonctionnalités peuvent être déportées sur  
le PC d'analyse réseau.

# Affichage des informations

La dernière étape du projet consiste en l'affichage interprété des informations du sniffer. Par exemple pour les trames normales, un prédécoupage peut être proposé, ou pour le bit stuffing les bits spéciaux peuvent être mis en valeurs.  
Cet affichage sera effectué sur le PC d'analyse réseau.