**Sureté de fonctionnement**

**PROJET AUTOMATIQUE ET RESEAU**

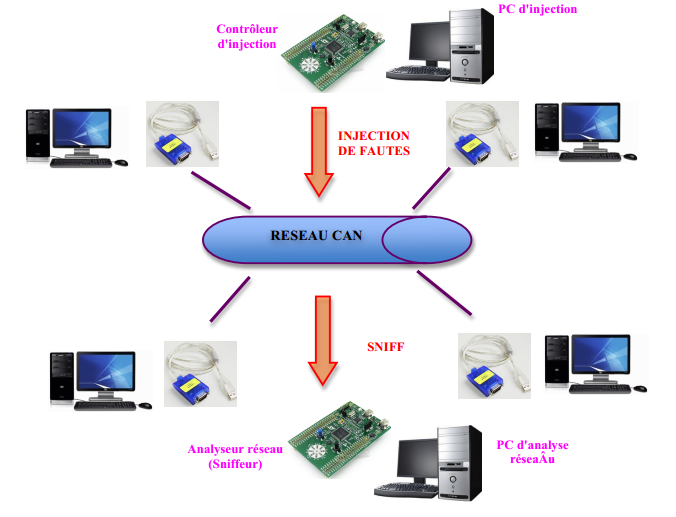
**&**

**SURETE DE FONCTIONNEMENT**

**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

Ce document a pour but de …

# Fonctionnalités du sniffer

Le sniffer à pour but d’écouter ce qu’il se passe sur le réseau CAN, d’en extraire les différentes trames, de manière à les analyser, pour afficher leur contenu sur le PC de l’utilisateur. Il devra aussi être capable de détecter d’éventuelles erreurs et de les signaler à l’utilisateur.

## Outils utilisés

-Utilisation d’une STM32F4 discovery

-Utilisation de Coocox CoIDE

-Utilisation de cppcheck (application) pour la vérification du code

-Utilisation d’une liaison avec le transceiver

## Communication avec le transceiver (bufferring, protocole ?)

-Liaison série UART à 10,5Mbits/s

-Protocole

-Buffer de réception circulaire

## Décodage (destuffing, découpage de la trame, détection d’erreur…)

-Destuffing

-Découpage de la trame (SOF, ID, RTR, r0, r1, DLC, Data, CRC, ACK, EOF, IFS)

-Détection d’erreur (trame d’erreur, CRC, ACK)

## Envoie série vers le PC

-Serveur UART sur la carte (dernière trame ou continue + commande stop)

-Mise en forme avec la STM32 (trame complète, BS, détails)

-115200 bauds, no parity, 1 bit stop

## Interface graphique QT (si on a le temps)

-Si implémenté, utilisation des mêmes commandes série

# Détail des fonctionnalités

## Communication avec le transeiver

La communication sera unidirectionnelle (le sniffer sera le destinataire) et se fera via une communication série à 10,5 Mbits/s, en suivant le protocole que voici :

* octet 1 : 0x04 = octet commande
* octet 2 : nombre d'octets reçus (0-255)
* octets suivants : bits reçus (transmis octet par octet SOF en tête)

On enregistrera chaque trame les unes après les autres dans un tableau de trames et chaque trame sera un tableau d’octet. Ce tableau sera accessible à la tâche de décodage des trames. Son empreinte mémoire sera importante. Si nous nous contentons de stocker au maximum 16 trames de 24 octets (169 bits max), nous aurons besoin de 384 octets. Nous disposons de 192KBytes de RAM donc ça passe LLLLLAAARGE !!