**SURETE DE FONCTIONNEMENT :** mettre en place des méthodes (de travail, d’organisation, de développement, de test) permettant de minimiser les risques qui pourraient nuire à la finalité du projet.

**Règles de codage**

1. Variables
   1. Pas de variables globales, utilisation de structures et pointeurs
   2. Définition des structures dans les headers (.h)
   3. Définition des variables et paramètres en début de fonction
   4. Syntaxe variable : nomVariable
2. Fonctions
   1. Un fichier par "module" (par exemple un main.c, un can.c, … )
   2. Prototypes dans les headers (.h)
   3. Syntaxe fonction : Fichier\_NomFonction(paramètre1, paramètre2)
3. Defines
   1. A placer dans les headers (.h)
   2. Syntaxe define : NOM\_DEFINE

**Travail collaboratif**

* Définition des différentes tâches et attribution de celle-ci à chaque binôme.
* Travail par 2 sur chaque grosse partie.
* Point d’avancement à chaque début de séance (30min), l’objectif est de mettre en commun et d’évoquer les éventuels problèmes rencontrés pour mettre à profit l’expérience de chacun et minimiser la perte de temps.
* A chaque fin de séance : remplir un document sur l’état d’avancement : ce qui est fait, ce qui reste à faire, ce qui pose problème.
* Mise en commun des documents sur une plateforme GitHub en plus des sauvegardes sur les différents PCs.

**Validation du système**

Pour valider le système, nous procèderons de la manière suivante :

1. Test unitaire : tester chaque fonction avant de tester l’ensemble, aide à identifier rapidement les éventuelles pannes (voir tableau Excel Liste\_tests\_unitaires.xlsx)
2. Test de l’ensemble
3. Test des différents scénarios :
4. Trame correcte : scénario simple, être capable d’envoyer (injecteur) une trame correcte et vérifier qu’elle est bien reçue par le CAN bleu ou noir. De même, être capable de recevoir (sniffer) une trame correcte envoyée depuis le CAN bleu ou noir.
5. Trame erronée : le bon nombre de bit mais pas la valeur attendue (modification de bit) / trame fausse, pas la bonne taille (nombre de bits faux) cf AMDE. (Faire une liste des erreurs possibles. Cette liste permet une réflexion sur les erreurs réalisables mais nous ne sommes pas obligés de toutes les mettre en œuvre (la partie Injecteur choisit ce qu’elle décide de réaliser). Réflexion : par ex une erreur sur l’identifiant est une erreur applicative, l’identifiant reste une suite de 0 et de 1 donc physiquement la trame est correcte : réflexion à avoir sur les types d’erreurs.
6. Surcharge du réseau : plusieurs nœuds (STM) en même temps, comment se comporte le système ?
7. Mélange de scénarios : surcharge + trame erronée par ex. Idem, comment se comporte le système ?

Analyse des modes de défaillance (AMDE) pour les trames erronées et fausses

2 types de fautes peuvent apparaitre sur un réseau CAN, les fautes permanentes et les fautes transitoires. Les premières sont dues à des défaillances physiques sur le réseau, elles sont simples à identifier car répétables. Les secondes sont induites par des phénomènes tel que des perturbations électromagnétiques, elles sont plus dur à identifier du fait de leur caractère aléatoire.

Ces fautes peuvent avoir différents effets sur le système : De l'information perdue Information erronée (correctness) Information Retardee (Temporally invalid information) Surcharge du trafic

Source : <https://hal.archives-ouvertes.fr/hal-00321438/document>