جامعة سيدي محمد بن عبد الله بفاس
UNIVERSITÉ SIDI MOHAMED BEN ABDELLAH DE FES

كلية العلوم والتقنيات فاس
FACULTÉ DES SCIENCES ET TECHNIQUES DE FÈS

LICENCE SCIENCES ET TECHNIQUES
Génie Electrique

RAPPORT DE FIN D'ETUDES

Intitulé :

Etude & mise en œuvre d'un réseau de capteurs basé sur le bus CAN

Réalisé Par :

Elimane Cheikh WADE

Encadré par :

Pr Kamal ZARED

Soutenu le 05-07-2022 devant le jury

Pr	**Hicham GHENNIOUI**	**(FST FES)**
Pr	**Kamal ZARED**	**(FST FES)**

Remerciements

Au nom d'Allah, le Tout-Miséricordieux, le Très – Miséricordieux.

La louange est à Allah l'unique et la paix et le salut sur celui qui n'a point de messager après lui et sur sa famille, ses compagnons et tous ceux qui suivent son chemin jusqu'au jour de la résurrection. Nous rendons tout d'abord grâce à notre Seigneur, Le Tout Puissant, Le Très Haut qui nous a permis d'achever ce travail.

Il m'a été très difficile d'écrire cette page par souci de n'oublier les nombreuses personnes qu'il me faut citer pour leur aide, leur accueil, leur soutien ... ! Qu'elles soient toutes assurées de ma plus profonde reconnaissance même si leur nom n'y figure pas !

Je tiens, tout d'abord, à remercier et témoigner de ma profonde reconnaissance à mon encadrant le Professeur **Mr Kamal Zared,** pour son soutien, son orientation, ses remarques et recommandations pertinentes.

J'adresse également mes plus sincères remerciements aux **membres de jury** le **Pr Hicham GHENNIOUI** et le **Pr Kamal Zared**, et l'ensemble du **corps professoral de la FST Fès** pour leur soutien tout au long de ma formation.

Tout ceci ne serait rien sans **ma famille et mes amis**. Un énorme merci à **mes parents**, **mes frères et sœurs** pour m'avoir toujours soutenu et laissé libre de faire ce qui me plaisait et pour m'avoir encouragé dans mes choix qui ont beaucoup investi en moi.

Un grand merci également à un père **Babacar Dieng**, mon oncle **Cheikh Tidiane Gadio** et ma tante **Lala Mare** sans qui ces années d'étude n'auraient jamais été réussies, je vous souhaite à tous une longue vie avec beaucoup de bonheur et de réussite dans vos projets.

Résumé

Le marché de l'automobile est en perpétuelle évolution avec une concurrence massive obligeant l'innovation et la démarcation des principaux acteurs du marché. L'intégration de nouveaux systèmes embarqués plus puissants, plus performants et pouvant proposer tout type de fonctionnalités, permet de répondre aux besoins des consommateurs de plus en plus exigeants.

Les trois grands axes d'avenir sont la voiture électrique, la voiture connectée et la voiture automatisée. Le bus CAN est transversal à tout cela. Elle jouera un rôle dans des fonctions aussi diverses que la gestion de la recharge électrique, le contrôle moteur, la connectivité du véhicule à son environnement extérieur, ou encore les systèmes d'assistance à la conduite.

Notre projet s'est porté sur l'étude et la mise en œuvre du réseau de terrain CAN tout en essayant de satisfaire les exigences qu'il impose dans sa réalisation.

La mise en pratique consistait surtout à une simple communication entre deux microcontrôleurs Emetteur/Récepteur. Nous avons cherché à pousser la limite en mettant en œuvre un réseau de capteurs dans le protocole CAN rendu possible grâce au module CAN, le MCP2515, dans l'utilisation de l'Arduino.

Table des matières

Liste des figures

Liste des Tableaux

Liste des abréviations

CAN	Controller Area Network
CAN H	Controller Area Network High
CAN L	Controller Area Network Low
CRC	Cyclic Redundancy Code
DLC	Data Length Code DLL Data link layer
EOF	End Of Frame
ECU	Electronic Control Unit EOF End Of Frame
ISO	Organisme International de Normalisation
LCD	Liquid Crystal Display LED Light-Emitting Diode
LED	Light-Emitting Diode
NRZ	Non return to Zero
OSI	Open Systems Interconnection
PIC	Programmable Intelligent Computer
RTR	Remote Transmission Request
SAE	Society of Automotive Engineers SOF Start Of Frame
SPI	Serial pherephial interface
SCKL	Serial Clock
MOSI	Master Output, Slave Input
MISO	Master input, Slave Output
SS	Slave Select
IDE	Integrated development environment
IIC	Inter Integrated Circuits
DSP	Digital Signal Processor
SDA	Serial Data
SCL	Serial Clock
USB	Universal Serial Bus
MCU	Microcontrôleur
ICSP	In Circuit Serial Programming

Introduction générale

Le confort et la sécurité sont des objectifs que l'homme a voulu atteindre dans tous les aspects de sa vie. Les automobiles sont parmi ces aspects. Après leur invention pour rendre le transport plus facile et rapide. Le nombre des équipements ne cesse d'augmenter et les normes en matière de la pollution et de la consommation d'énergie sont mises en jeu. Puisque les services électroniques offrent plus de fonctionnalités, les pièces mécaniques des véhicules sont progressivement remplacées par des composants électroniques. **[1]**

Pour réduire la quantité des câbles, éliminer les anciennes connexions point-à-point et améliorer la communication entre les différents composants des véhicules, les constructeurs automobiles ont introduit des bus et des protocoles de communication (Ethernet, CAN, Flex Ray, LIN…). Le CAN (Control Area Network) normalisé avec le modèle ISO 11898 est le réseau le plus utilisé sur les nouvelles générations de véhicules et en particulier dans les véhicules électriques ou autonomes. Les propriétés de ce bus permettent une grande fiabilité dans la transmission et un faible coût de mise en œuvre. Pour ces raisons, le Bus CAN s'est imposé comme le réseau de terrain le plus utilisé dans l'industrie de l'automobile.

Dans le cadre de ce projet nous allons étudier, concevoir et réaliser un réseau de terrain bus CAN. L'objectif du travail consiste à savoir comment s'est développé le bus CAN ? Quel est son principe de fonctionnement ? Quelles sont ses caractéristiques de communication sur un très faible taux d'erreur ? Comment le mettre en œuvre ?

Ce travail comporte trois chapitres :

Dans le premier chapitre, nous présenterons notre laboratoire de recherches et d'innovations : Systèmes, Signaux et Composants où nous avons passé notre stage d'une durée de deux mois. Nous présenterons le problématique et la solution apportée par le bus CAN.

Dans le deuxième chapitre, nous allons commencer par faire une étude théorique en définissant les vocabulaires techniques, les normes qui peuvent être appliquées et les domaines d'application.

Finalement, au début du dernier chapitre, nous présenterons les outils et environnements de travail. En fin la réalisation d'un réseau de capteur bus CAN, nous allons établir une communication entre deux nœuds tel que chaque nœud sera composé d'un microcontrôleur (Arduino Uno), d'un contrôleur CAN MCP2515 intégrant un émetteur/récepteur TJA1050 et

d’un certain nombre de capteurs et/ou composants.

Nous finaliserons nos travaux avec une conclusion générale qui résumera l’ensemble du déroulement ainsi qu’une perspective.

Chapitre I : Présentation du lieu de stage

Dans ce chapitre, nous présenterons le lieu de stage, l'équipe qui le compose, le nombre de membres, les établissements en collaboration et en fin nous finirons avec les thématiques de recherche.

I. Lieu de stage

Notre stage a été effectué au sein du Laboratoire : Signaux, Systèmes et Composants d'acronyme LSSC. Ce dernier est un laboratoire de l'Université Sidi Mohammed Ben Abdellah (USMBA), sis à la faculté des Sciences et Techniques de Fès (FSTF). Il a pour directeur **Pr Farid ABDI** et son directeur adjoint **Pr Hicham GHENNIOUI**.

La mission étant de CONTRIBUER et RENFORCER la RECHERCHE, le DEVELOPPEMENT TECHNOLOGIQUE et l'INNOVATION au sein de l'USMBA.

II. Equipes de recherche

Le LSSC est constitué de cinq équipes de recherche :

① Signal, Télécommunication, CEM
② Intelligence Artificielle, Systèmes embarqués
③ Composants, Conception, Microélectronique
④ Capteurs, Réseaux
⑤ Synthèse, Matériaux

III. Effectif & Etablissement d'accueil

• Le laboratoire a un effectif de :

- 20 Enseignants chercheurs permanents
- 35 Doctorants

• Etablissement d'accueil : FST de Fès
• Etablissements membres : FST, ENS, FPT, FSDM

IV. Thématiques de recherche

Les différentes équipes du laboratoire LSSC travaillent dans différents domaines de recherche complémentaires et innovants. Les principales thématiques de recherche pour chaque équipe sont :

① Signal, Télécommunication, CEM

Cette thématique concerne la mise en place d'outils, de logiciels et de méthodes pour la mesure et la transmission de données pour des systèmes complexes et à usage intensif en utilisant les techniques et support de transmission (RF, optique), avec une attention particulière à l'intégrité, la robustesse, la fiabilité des architectures soft et hard mises en place.

② Intelligence Artificielle, Systèmes embarqués

Cette partie concerne les aspects d'aide à la décision à travers le développement d'outils d'aide faisant appel aux principes de l'intelligence artificielle incluent la modélisation holistique de système comportemental, informatique, physique et/ou aspects humains dont l'objectif est de faciliter la gestion et la commande du système (prévision).

③ Composants, Conception, Microélectronique

Parmi les objectifs de cette partie, la conception de composants électroniques analogiques (Am-Op régulateur LDO) mais aussi l'implémentation des systèmes plus complexes, numériques dédiés au traitement de l'information (vidéo, image ou parole-son).

④ Capteurs, Réseaux

L'objectif de cette partie est l'étude le dimensionnement est la mise en place d'un système de capteurs sans fils pour les mesures physiques et la transmissions des données. Ceci passe par la conception (ou l'utilisation) d'une plateforme pour la création et la gestion du réseau de capteurs.

⑤ Synthèse, Matériaux

Cette thématique fédère deux activités complémentaires : matériaux et composants, l'objectif général étant la synthèse de nouveaux matériaux fonctionnels, leurs caractérisation et l'optimisation de leurs propriétés (diélectriques, piézoélectriques ou ferroïques) et par la suite leur structuration pour l'insertion dans un dispositif (capteur).

V. Cahiers de charge

V.1. Problématique :

Depuis les années 1960 la longueur de câble utilisé dans l'automobile ne cesse de croître dépassant les 200 mètres en 1995. Le nombre de connexions atteint 1800 à cette même date. La sécurité et la fiabilité sont menacées (figure 1).

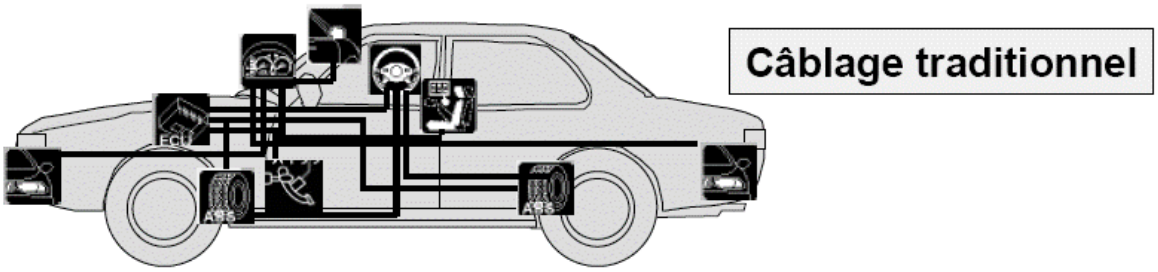


Figure 1 : Connexion point à point

Les normes en matière de pollution et de consommation d'énergie obligent les constructeurs à multiplier les capteurs et actionneurs intelligents dans leur véhicule accélérant ce processus de multiplication des câbles et connexions depuis plus d'une vingtaine d'années.

Le besoin de sécurité accrue (ABS, ESP, AIR-BAG…) et la demande confort (Mémorisation des réglages de conduite, climatisation, régulée par passager, système de navigation…) ne font que renforcer cette tendance.

V.2. Solution :

Enfin, BOSCH a développé une solution de multiplexage des informations circulant à bord de la voiture. Le bus CAN est un moyen de communication série qui supporte des systèmes embarqués temps réel avec un haut niveau de fiabilité. Ses domaines d'application s'étendent des réseaux moyens débits aux réseaux de multiplexages faibles coûts (figure 2).

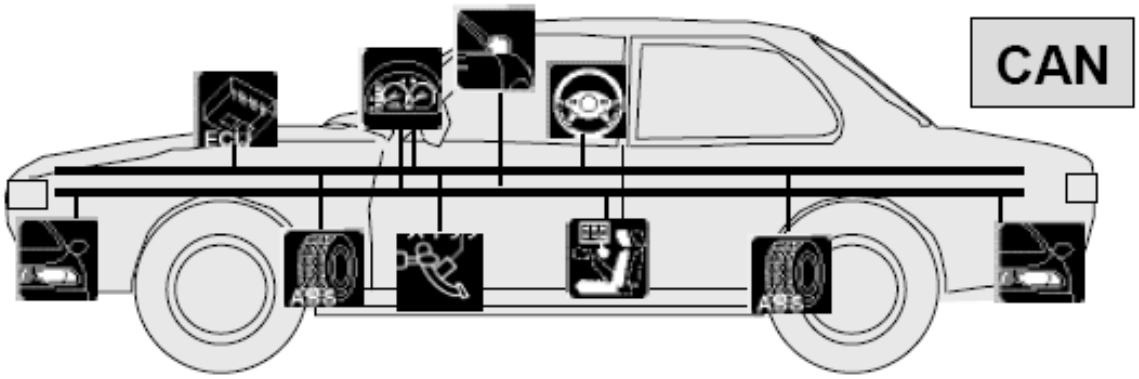


Figure 2 : Connexion bus can

La structure du bus CAN possède implicitement les principales propriétés suivantes :

- Hiérarchisation des messages,
- Garantie des temps de latence,
- Souplesse de configuration,
- Réception de multiples sources avec synchronisation temporelle,
- Fonctionnement multi-maître,
- Détections et signalisation d'erreurs,
- Retransmission automatique des messages altérés dès que le bus est de nouveau au repos,
- Distinction d'erreurs : d'ordre temporaire ou de non-fonctionnalité permanente au niveau d'un nœud, déconnexion automatique des nœuds défectueux.

En ce qui concerne notre projet, notre solution va se baser de celle proposé par la société BOSCH. Nous allons implémenter le bus CAN comme un réseau de capteurs. Ce dernier peut être formé de plusieurs nœuds. Chaque nœud sera composé d'un module MCP2515 relié à une carte Arduino et avec un ou plusieurs capteurs ou composants (figure 3).

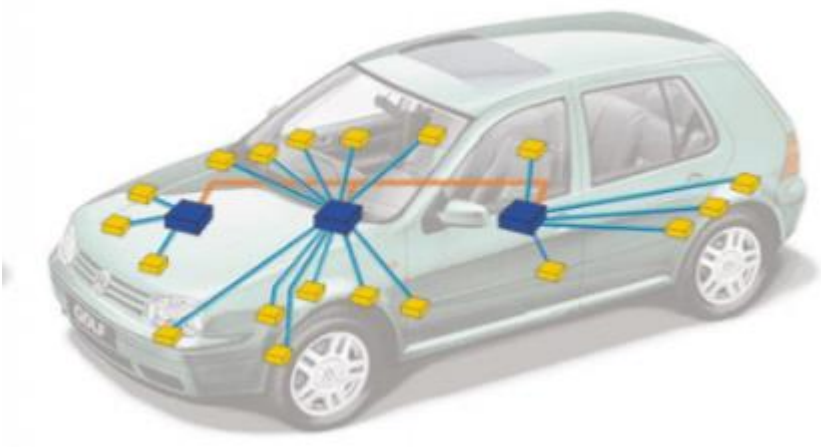


Figure 3 : Schéma de connexion à 3 nœuds

VI. Conclusion

Dans le présent chapitre, nous avons présenté le lieu de stage en faisant le point sur l'organigramme du laboratoire (LSSC) notamment les différentes équipes de recherche et les thématiques abordées.

Nous avons également vu le problème soulevé et la solution qui a été apportée par la société BOSCH et un schéma de connexion de nœuds de notre projet.

Chapitre II : Réseau de terrain bus CAN

La licence Génie Électrique consacre une partie de son enseignement au bus de terrain et notamment à ceux qui ont pour vocation à être intégrés au sein de systèmes embarqués. C'est le cas du réseau de terrain (en anglais CAN : Control Area Network). **[5]**

CAN est initialement créé par la compagnie allemande BOSCH et Intel pour des applications automobiles (Mercedes-Benz Classe S). CAN spécifie la partie physique du réseau, le contrôle de l'accès au support de transmission et la gestion des erreurs de transmission de communication.

Si au départ le bus CAN a été développé pour des systèmes embarqués (automobiles, machine agricoles, aéronautique…), il a été très rapidement adopté par le milieu des automatismes industriels.

I. Le bus CAN (Control Area Network):

I.2. Domaines d'applications

Le CAN est un réseau embarqué utilisé non seulement dans le marché automobile mais aussi dans des autres domaines (Figure 4) :

- Aéronautiques.
- Médicales, par exemple système PMS utilisant le CAN pour la communication entre l'unité de positionnement du patient et l'unité de production des rayons X.
- Production industrielle.
- Matériel agricole.
- Bateaux.

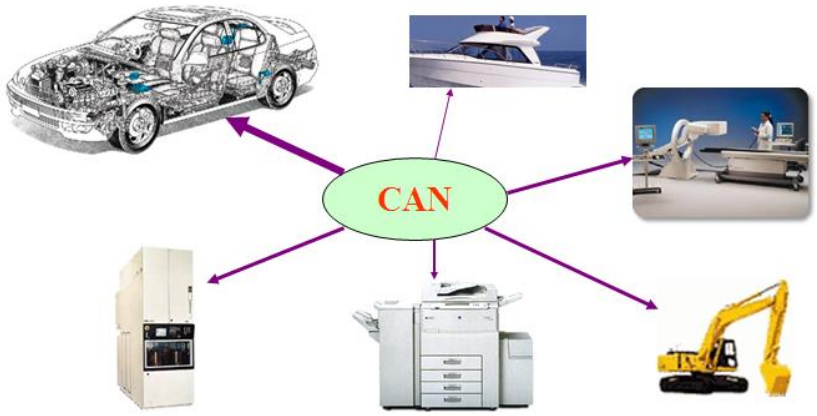


Figure 4 : Les domaines d'applications du bus CAN

I.4. Topologies d'un bus CAN

Le bus CAN est caractérisé par une communication :

- **Série** : les informations se transmettent les unes après les autres sur la même voie.
- **Bidirectionnel** : les informations se transmettent dans les deux sens.
- **Half duplex** : la transmission est bidirectionnelle mais d'une façon alternée (on ne peut pas transmettre et recevoir en même temps).

I.4.a. Le protocole CAN

Le concept de communication du bus CAN est celui de la diffusion d'information (broddcast) : chaque station du réseau écoute les trames par les stations émettrices. Ensuite chaque nœud décide quoi faire du message, s'il doit y répondre ou s'il doit agir ou non, etc.

Le protocole CAN autorise différentes stations à accéder simultanément au bus. C'est un procédé rapide et fiable d'arbitrage qui détermine la station qui émet en premier. L'accès au bus est donc aléatoire car une station peut émettre à n'importe quel moment.

Mais cet accès ; cette méthode est appelée CSMA CD/AMP (*Carrier Sense Multiple Acces With Collision Detection and Arbitration Message Priority*). Il est un protocole très robuste qui permet la détection et la signalisation d'erreurs, il est caractérisé par : **[6]**

- Hiérarchisation des messages (les messages sont classés selon leurs priorités Figure 11).
- Accès multiple (*Multiple Access* : MA) (La machine observe le média en cherchant à détecter une porteuse (*Carrier Sense* : CS). Si aucune trame n'est transmise, elle ne trouve pas de porteuse).
- Vérification du support de transmission (une station vérifie qu'aucune trame n'est émise sur le média. Si c'est le cas elle commence à émettre son paquet. Si ce n'est pas le cas, elle attend la fin de la transmission en cours).
- Détection de collision (lors de son émission une machine peut déceler un problème de contention, et s'arrêter avec l'intention de renvoyer son paquet ultérieurement quand elle aura de nouveau la parole. De façon à minimiser le risque de rencontrer une deuxième collision avec la même machine, chacune attend pendant un délai aléatoire avant de tenter une nouvelle émission)

Remarque :

De manière à ne pas saturer un réseau qui s'avérerait déjà très chargé, la machine

n'essaiera pas indéfiniment de retransmettre un paquet si à chaque tentative elle se trouve en conflit avec une autre ; après un certain nombre d'essais infructueux (le nombre maximum de reprises est de 16) le paquet est éliminé. On évite ainsi l'effondrement du réseau. Les couches supérieures sont averties que la transmission du message a échoué.

I.4.b. Protocole de communication

Le protocole de communication est une méthode permettant la communication entre des nœuds en respectant des règles et des procédures pour émettre et recevoir des informations sur un réseau.

Message : contient des données allant jusqu'à 64 bits.

Arbitrage : les nœuds émettent des messages dans le bus ; le message le plus prioritaire sera traité en premier.

Routage des informations : est la méthode qui détermine les chemins dans un réseau pour envoyer les données d'un expéditeur jusqu'à un ou plusieurs destinataires.

Trame : est une information structurée d'éléments numériques 0 et 1.

Acquittement : vérification du message reçu par les récepteurs, et si le message est correct, ils doivent acquitter en émettant un flag.

II. Caractéristiques physiques du bus CAN

II.1. Support de transmission

La transmission de données, en série asynchrone, s'effectue sur une paire filaire différentielle se terminant de part et d'autre par des résistances de terminaison. Ce support permet la liaison entre un ensemble de nœuds (ECU : Electronic Control Unit).

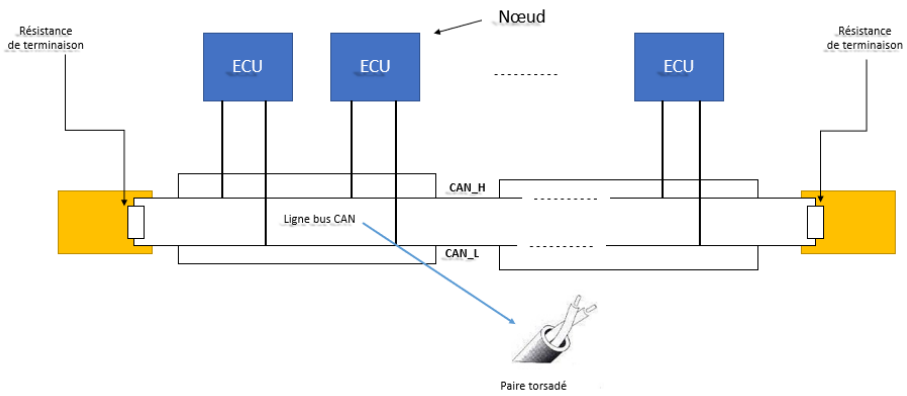


Figure 5 : Structure du bus CAN

L'utilisation de deux fils torsadés permet de supprimer les parasites qui peuvent être induit sur les lignes du bus. Il existe deux types principaux de transmission possible en bus CAN : CAN Low Speed et CAN High Speed.

Paramètres	CAN Low Speed	CAN High Speed
Débit	125Kb/s	125Kb/s à 1Mb/s
Nombre de nœuds sur le bus	2 à 20	2 à 30
Courant de sortie (mode émission)	1 mA sur 2,2 KΩ	25 à 50 mA sur 60Ω
Niveau dominant (0)	CANH = 4V CANL = 1V	CANH = 3,5V CANL = 1,5 V
Niveau récessif (1)	CAN H = 1,75V CANL = 3,25V	CAN H = 2,5V CANL = 2,5V
Caractéristique du câble	30 pF entre les câbles de ligne	2 X 120 Ω
Tensions d'alimentation	5V	5V

Tableau 1 : Différences entre les deux types de bus

II.2. Architecture des ECU

Les ECU ou calculateurs sont réalisés à l'aide d'unités de traitements (microcontrôleurs, DSP ou ordinateurs) reliés à des ensembles qui sont le CAN Controller et le transceiver qui est également appelé le module émetteur/récepteur. Le transceiver est un coupleur différentiel. Il permet l'adaptation des niveaux électriques de tensions circulant sur le bus. Le CAN Controller, quant à lui, a pour fonction de gérer les échanges de données sur le bus (priorité, erreur de données…).

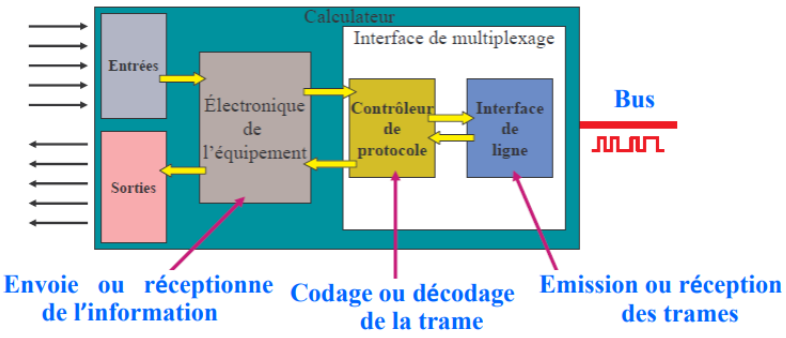


Figure 6 : Compositions des ECU dans le bus CAN

II.3. États logiques

Le CAN spécifie deux états logiques : récessif et dominant ; selon la norme ISO une tension différentielle est utilisée pour représenter les états logiques (0, 1) suivants.

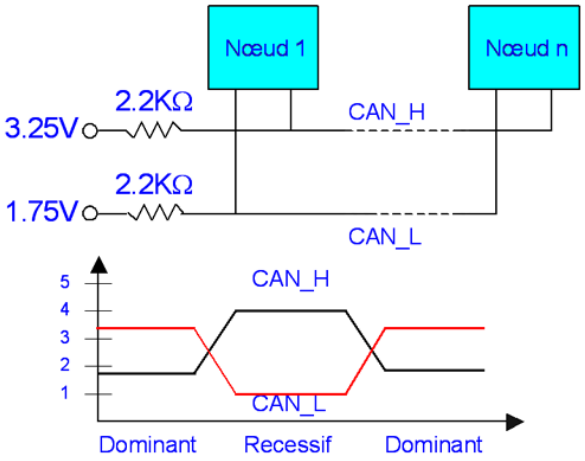


Figure 7 : ISO 11519 Low Speed CAN < 125Kbps

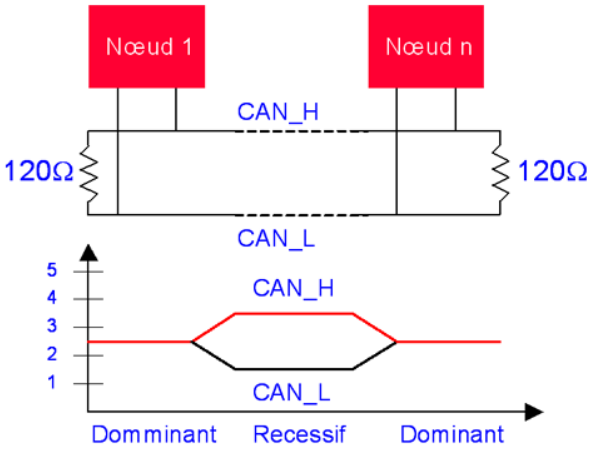


Figure 8 : ISO 11898 Speed CAN 125Kbps – 1Mbps

Etudions la version ISO11898 Speed CAN:

- En état récessif, la tension différentielle sur le CANH et le CANL est inférieure au seuil minimum (< 1.5 V).
- En état dominant, la tension différentielle sur le CANH et le CANL est supérieure au seuil minimal.

Si au moins un nœud émet un bit dominant, le statut du bus passe à l'état dominant indépendamment de la position des autres sorties de bits récessifs.

III. Le bus CAN dans le modèle OSI

Le modèle OSI (Open System Interconnexion) est un modèle de référence en ce qui concerne les réseaux proposés par l'ISO (International Standards Organisation) qui décrit les concepts et les démarches à suivre pour interconnecter des systèmes.

N° de la couche	Modèle ISO/OSI	Protocole CAN
7	Application	Spécifié par l'utilisateur
6	Présentation	Vide
5	Session	Vide
4	Transport	Vide
3	Réseau	Vide
2	Liaison de données	Protocole CAN
1	Physique	Protocole CAN

Tableau 2 : Différentes couches de la norme OSI

Le bus CAN doit répondre au modèle OSI. Pour le protocole CAN, le document de référence développé par la société BOSCH définie les deux premières couches (dites couches de basses). Ce sont :

- La couche physique
- La couche liaison de données.

❖ **La couche de liaison** est une couche qui fournit les moyens fonctionnels nécessaires à l'établissement, au maintien et à la libération des connexions entre les entités du réseau. Elle réalise les fonctions suivantes :

- La détection des erreurs.
- La signalisation des erreurs.
- La correction des erreurs.
- Le filtrage des messages.
- La mise en trame du message.
- L'arbitrage.
- L'acquittement.

- **La couche physique** assure la transmission des données et le transfert physique des bits entre chaque nœud, le bus CAN utilise le codage NRZ, aussi la communication des données sur un bus se fait par un canal de transmission constitué de deux fils torsadés CANH et CANL (Figure 9**)**. Le codage NRZ assure la fiabilité de transmission des données, en effet il permet de réduire les perturbations externes.

Figure 9 : Fils torsadés du bus CAN

- **La couche application** est la dernière couche du modèle OSI. Elle donne aux applications le moyen d'accéder aux couches inférieures. Cette couche n'est bien sûr pas vide pour le protocole CAN, mais sa spécification est laissée à l'utilisateur.

IV. Principe de fonctionnement du bus : Structure de trame

Généralement dans le standard CAN on peut distinguer plusieurs types de trame **[1]** :

Trame de données (*data frame*) **:** trame qui transporte des données.

Trame de requête (*remote frame*) : est émise par un nœud désirant recevoir une trame de données (l'identificateur est le même pour les deux trames dans ce cas).

Trame de surcharge (*overload frame*) **:** indique qu'une station est surchargée pendant un certain laps de temps. Utilisée par le contrôle de flux pour demander un délai supplément.

Trame d'erreur (*error frame*) **:** est transmise à chaque fois qu'un nœud du réseau détecte une erreur. Elle indique la nature des erreurs (de bit, de stuffing, de CRC, d'acquittement…) et permet la correction en demandant de renvoyer à nouveau la donnée.

Deux trames successives doivent respecter une pause c'est ce qu'on appelle une période d'**inter-trame** (*interframe*), cette pause est équivalente à la durée nécessaire pour transmettre 3 bits, pendant cette durée le bus est maintenu à l'état résistif équivalent à 1

logique.

IV.1. Trame de données

Les trames de données (*Data frame*) sont utilisées pour transmettre des informations entre un nœud source et un ou plusieurs nœuds récepteurs.

Il existe deux versions CAN :

CAN 2.0A : trame standard avec un identificateur de 11bits.

CAN 2.0B : trame étendue avec un identificateur de 29 bits.

La trame de données standard **CAN 2.0A** se décompose en **7** parties (Figure 10).

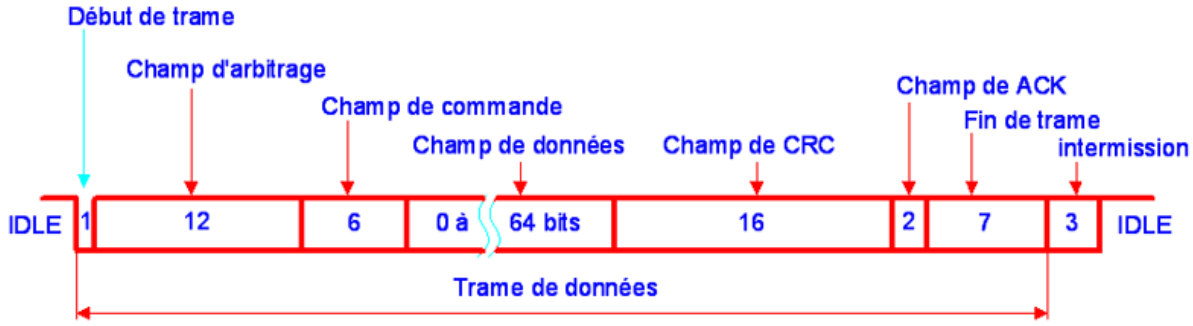


Figure 10 : Trame de données

Dès que le bus est libre **(**bus IDLE), n'importe quel nœud relié au réseau peut émettre un nouveau message.

IV.1.1. Début de trame SOF (Start Of Frame)

Permet de signaler à toutes les stations connectées le début d'un échange. Cet échange ne peut démarrer que si le bus était précédemment au repos, autrement dit, une transition de la valeur électrique de bus CAN de l'état récessif (état logique 1) vers l'état dominant (état logique 0).

IV.1.2. Champ d'arbitrage

Lorsque le bus est libre, aucune trame ne circule, les nœuds tentent de transmettre simultanément, c'est à dire les nœuds entrent dans une compétition puisque l'arbitre de cette compétition c'est l'identifiant de chaque nœud, celui qui a le plus faible identifiant, gagne le bus.

Les nœuds sont câblés sur le bus par le principe d'un « ET câblé ». En cas d'un conflit le bit 0 écrase le bit 1. Dans l'exemple ci-dessous trois stations entrant dans une compétition.

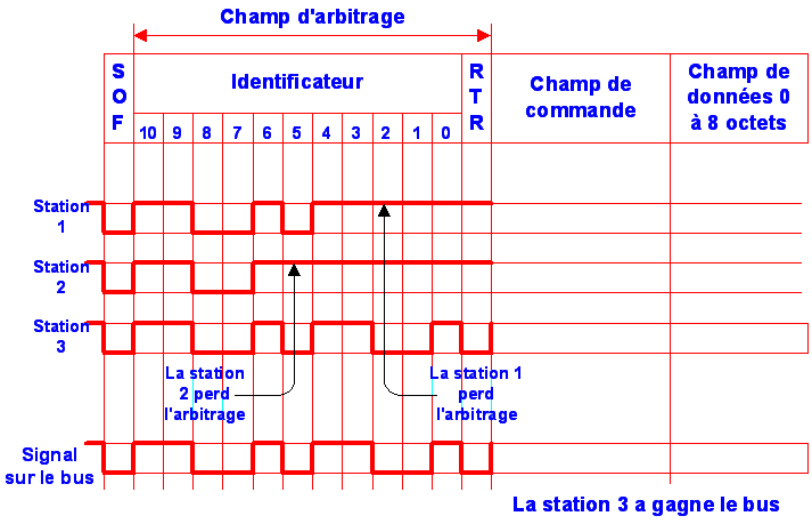


Figure 11 : Principe d'arbitrage dans le protocole CAN

- La station 2 perd la compétition suivie par la station 1.
- La station 3 gagne la compétition, donc gagne le bus.

Le champ d'arbitrage est composé de 11 bits d'identification pour la version standard, (Figure 12) et 29 bits pour la version étendue. Le champ d'identification est suivi par un bit dit RTR (*Remote Transmission Request*) qui détermine s'il s'agit d'une trame de données ou d'unetrame de demande de message, il est dominant dans le cas de trame de données.

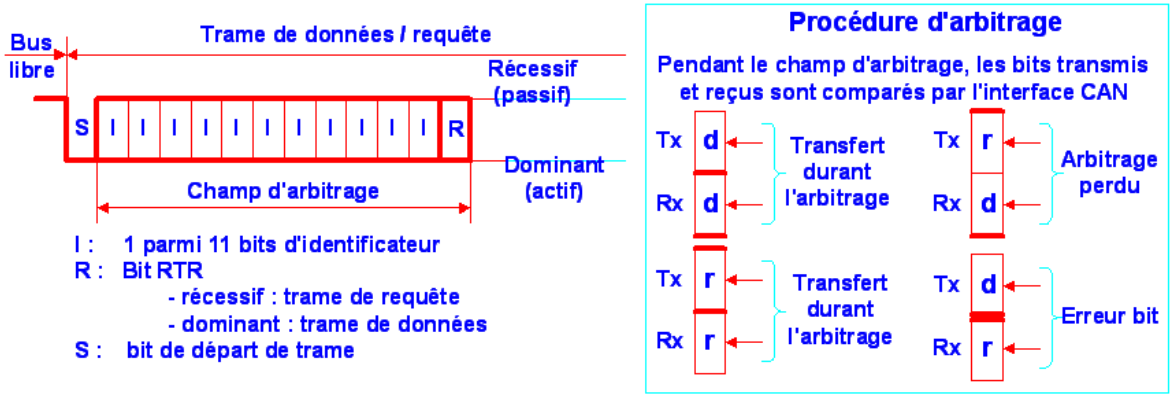


Figure 12 : Champ d'arbitrage

- **Le Rôle des bits dans le champ d'arbitrage**

Le bit SOF (début de trame de données)

C'est dominant il signale à toutes les stations le début d'un échange. Cet échange ne peut démarrer que si le bus était précédemment au repos.

Toutes les stations doivent se synchroniser sur le front avant la transition du bit de

départ.

Identificateur :

La longueur de l'identificateur est de 11 bits, les bits sont transmis dans l'ordre de ID_10 à ID_0 (le moins significatif est ID_0). Par ailleurs les 7 bits les plus significatifs (de ID_10 à ID_4) ne doivent pas être tous récessifs.

$ID = 1111111XXXX$ (X valeur indéterminée), c'est-à-dire un nombre maximal d'identificateurs de : $(2^{11} - 2^4) = 2048 - 16 = 2032$ combinaisons.

Le bit RTR :

Lors d'une trame de données (*data frame)*, le bit de RTR doit être dominant.

IV.1.3. Champ de commande

Le champ de commande est constitué de 6 bits (Figure 13**)** :

- ✓ Le bit R0 s'appelle IDE utilisé pour différencier le type entre le format standard et le format étendu. Dans le cas d'une trame standard ce bit doit être dominant et dans le cas d'une trame étendu, il est récessif. Le deuxième bit R1 est en réserve d'usages ultérieurs et permet d'assurer des compatibilités futures ascendantes (par exemple avec les trames étendues).

- ✓ Les quatre derniers bits du champ de commande (champ DLC – *Data Length Code*) indiquent le nombre d'octets de données contenus dans le champ de données pour une trame de données ou bien le nombre d'octets de données dont a besoin un nœud du réseau lors d'une trame de requête (Tableau 3). Le nombre d'octets de données ne peut pas excéder la valeur 8.

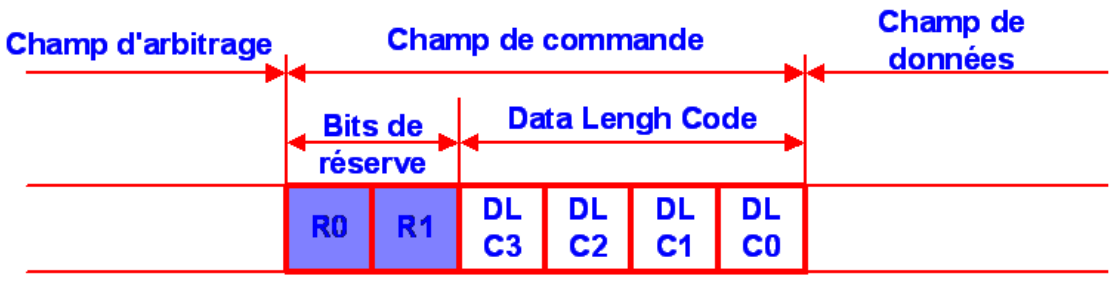


Figure 13 : Champ de commande

Nb d'octet	DLC 3	DLC 2	DL C1	DLC 0
0	D	D	D	D
1	D	D	D	R
2	D	D	R	D
3	D	D	R	R
4	D	R	D	D
5	D	R	D	R
6	D	R	R	D
7	D	R	R	R
8	R	D	D	D

D **:** Bit dominant et R **:** Bit récessif

Tableau 3 : Codage de champ DLC

IV.1.4. Champ de données

C'est l'endroit où se trouvent les données utiles transmises, Il a une longueur qui peut varier de 0 à 64 bits (0 à 8 octets) transmis avec le MSB (*Most Significant Bit* en tête) (Figure 14). Cette longueur a été déterminée lors de l'analyse du champ de commande.

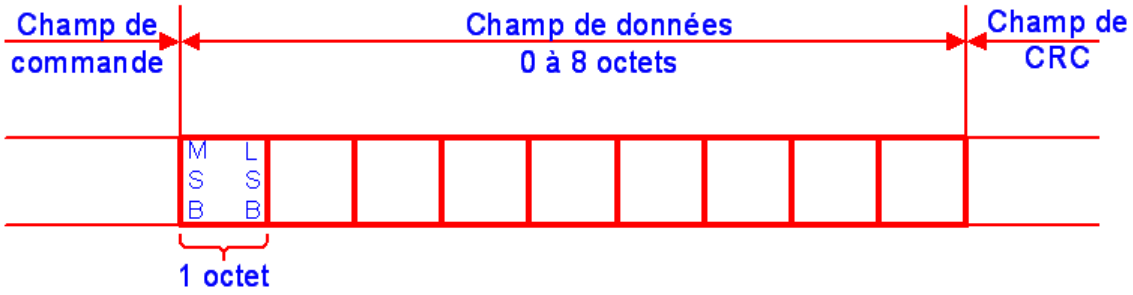


Figure 14 : Champ de données

Remarque :

De 0 à 8 bits inclus, cela fait neuf valeurs donc 4 bits du DLC pour définir le nombre de données contenues

IV.1.5. Champ de CRC

Le champ CRC est composé d'une séquence de 15 bits de CRC (*Cyclic Redundancy Check*) et d'un bit dit délimiteur (*CRC Delimiter*) toujours récessif (Figure 15). Ces bits sont calculés à l'émission et sont recalculés à la réception, s'il y a une différence, une erreur CRC est déclarée.

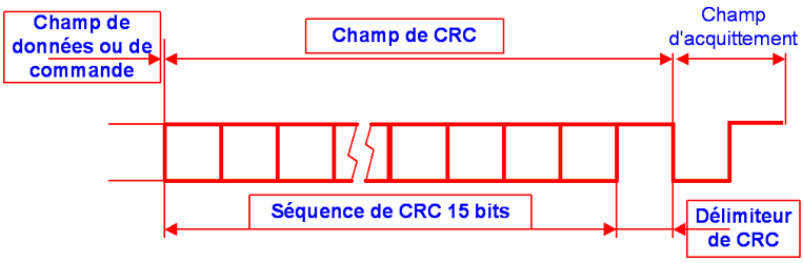


Figure 15 : Champ CRC

IV.1.6. Champ d'acquittement

Le champ ACK indique que la trame a été reçue normalement. Ce champ se compose de 2 bits ; un pour l'emplacement l'ACK Slot et un pour le délimiteur (bit récessif) (Figure 16).

- ✓ Un nœud en train de transmettre envoie un bit récessif pour l'ACK Slot.
- ✓ Un nœud ayant bien reçue correctement le message, le premier bit ACK Slot forcé par un bit dominant : il acquitte le message.

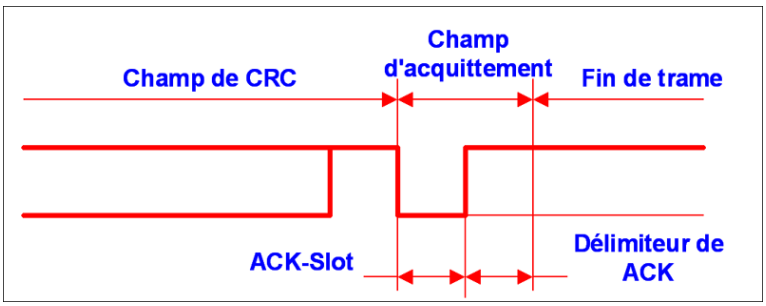


Figure 16 : Champ d'acquittement

IV.1.7. Fin de trame EOF

La trame de donnée se termine par un drapeau formé par une séquence de 7 bits récessifs, ce qui, dépasse de deux bits la largeur de la norme de *bit stuffing*. Ce champ a une structure fixe et les logiques de codage (à l'émission) et de décodage (aux réceptions) de *bit stuffing* sont désactivées pendant la séquence du champ de fin de trame.

IV.2. Trame de requête

Lorsqu'un nœud a besoin des données particulières, il va émettre une trame de requête dès que le bus sera libre en prenant soin d'indiquer dans le champ de contrôle le nombre d'octets de données dont il a besoin.

Une trame de requête est constituée de la même manière qu'une trame de données sauf

que le champ de données est vide, plus que ça il y'a une différence dans le champ d'arbitrage tel que le bit RTR est récessif. Pendant l'arbitrage la trame de données est prioritaire sur la trame de requête.

IV.3. Trame de surcharge

La trame de surcharge est utilisée lorsqu'un nœud demande un délai avec la réception d'une nouvelle trame et lorsque le nœud détecte un bit dominant pendant une période d'inter-trame (trois bits récessifs entre les trames).

Elle est formée de deux champs (Figure 17) :

- Le drapeau de surcharge constitué de six bits dominants,
- Le délimiteur de surcharge constitué de huit bits récessifs.

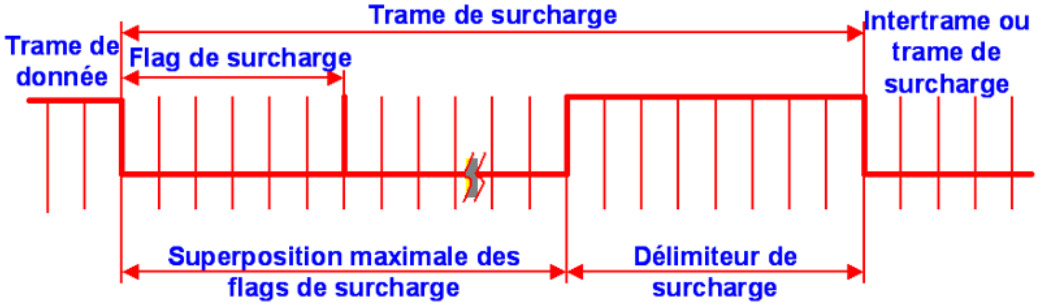


Figure 17 : Structure d'une trame de surcharge

IV.4. Trame d'erreur

Les trames d'erreur sont générées et transmises par le matériel CAN et sont utilisées pour indiquer l'existence de forte perturbation ou de pertes importantes pendant la transmission. Elles sont composées de deux trames d'erreurs qui peuvent être « active » de drapeau d'erreur de six bits et « passive » d'un délimiteur de champ (Figure 18).

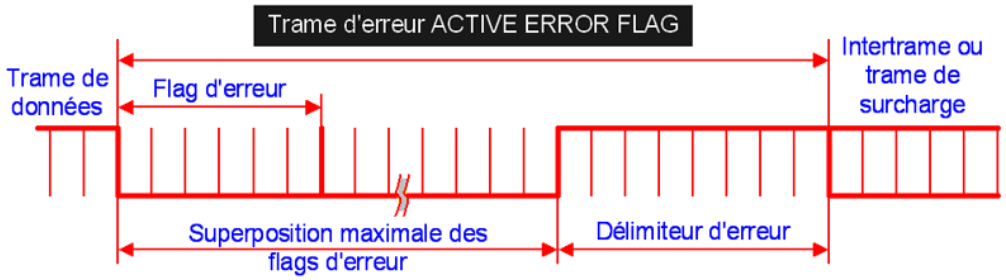


Figure 18 : Structure d'une trame d'erreur.

V. Différents types d'erreurs

Généralement la norme CAN implémente cinq mécanismes de détection des erreurs.

Erreur de bit : chaque fois qu'un nœud envoie un bit sur le bus, il regarde aussi en même temps le bit qu'il reçoit. Il est considéré comme erreur de bit lorsque le

bit envoyé est différent du bit reçu.

Erreur de Stuffing (*Stuff error***)** : le nœud détecte une erreur de stuffing lorsqu'il reçoit six bits consécutifs de mêmes valeurs dans une partie de message qui devrait être codé avec la méthode de stuffing. La trame a été altérée.

L'erreur de CRC (*CRC Error*) **:** est détecté lorsque le CRC calculé par le récepteur est complètement diffèrent de la valeur de CRC contenue dans la trame. La trame a été altérée.

Erreur de format : est détecté lorsqu'un bit qui devrait être à une certaine valeur a une valeur différente (*exemple* : d'un délimiteur)

Erreur d'ACK (*ACKnowledge error***) :** le transmetteur détecte une erreur d'acquittement lorsqu'il ne reçoit pas de bit dominant dans la phase d'acquittement ou pendant le ACK slot.

La Figure suivante résume les différents types d'erreurs et leur validité suivant l'endroit où l'on se trouve dans la trame.

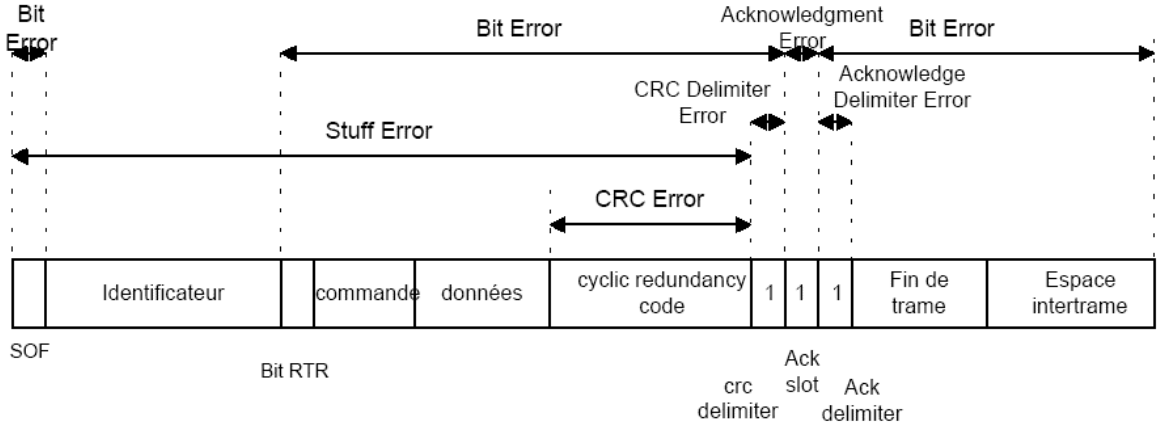


Figure 19 : Les sources d'erreur dans la trame CAN

VI. Conclusion

Dans ce chapitre nous avons introduit le bus de terrain CAN, son historique, son principe de fonctionnement aussi ses domaines d'applications puis les couches liées au bus CAN dans le modèle OSI. Il est adéquat aux applications qui exigent un grand nombre de messages courts avec une fiabilité élevée.

Le bus CAN est basé sur les messages et non sur les adresses ; il est particulièrement adapté lorsque des données sont sollicitées à plusieurs endroits.

Chapitre III : Réalisation d'un réseau bus CAN

Dans ce chapitre, nous allons mettre en œuvre un réseau de capteurs en se basant sur le protocole CAN. Tout d'abord nous allons présenter le microcontrôleur et l'ensemble des capteurs, composants et logiciels que nous avons utilisé dans la mise en pratique de notre réseau de communication : bus CAN. En fin de chapitre, nous réaliserons un montage bien détaillé d'un réseau de terrain avec le microcontrôleur Arduino, ainsi sera suivi de son algorithme en langage C.

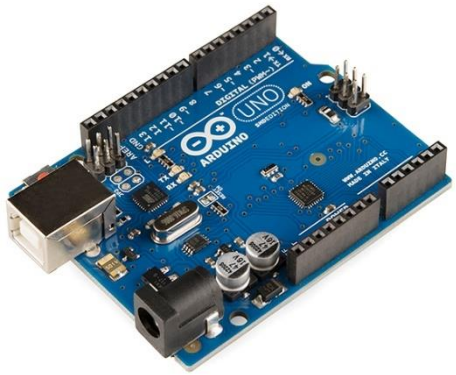
I. Outils et environnements de travail

I.1. Microcontrôleur : Arduino Uno

Un microcontrôleur (µC ou MCU) est un circuit qui rassemble l'ensemble des éléments essentiels d'un ordinateur : processeur, mémoire (mémoire morte et mémoire vive), unités périphériques et interfaces d'entrées-sorties. Les modèles le plus recherchés dans le milieu professionnel sont : le STM32, l'Uno, le PIC….

Construite autour de l'ATmega328, cette carte à MCU contient ce qui est nécessaire au fonctionnement du microcontrôleur. Arduino offre de nombreux avantages une multi fonctionnalité dans les systèmes d'exploitation multiplateforme et notamment dans la simplicité de son utilisation : logiciels et matériels extensible.

Il est basé sur un environnement de développement qui est variant du C et C++, allégé et restreint à l'utilisation de la carte.

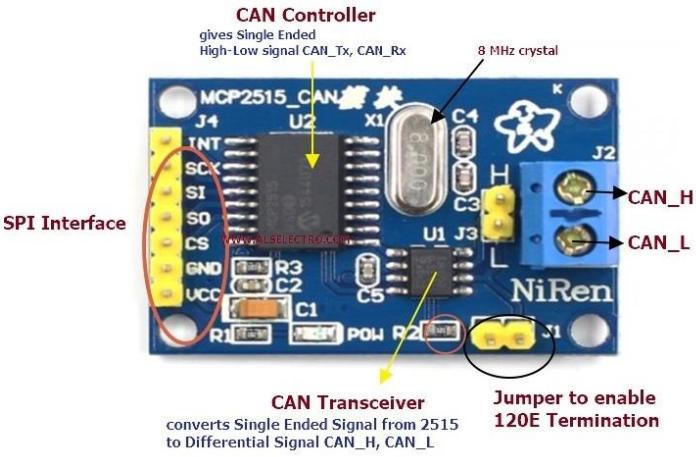


->6 entrées analogiques
->1 oscillateur à quartz de 16 MHz
->1 connecteur USB
->1 embase ICSP

Figure 20 : Carte programmable Arduino Uno R3

I.2 Module CAN MCP2515

Le MCP2515 de Microchip Technologies est un contrôleur CAN. Il est capable de transmettre et de recevoir des cadres distants et de données étendues et standards **[4]**. Le contrôleur de bus CAN MCP2515 est un module simple qui prend en charge le protocole CAN version 2.0B et peut être utilisé pour la communication à 1Mbps.



->Support CAN V2.0 B, taux de transfert jusqu'à 1Mb/S
->Alimentation électrique : DC 5V
->Contrôle du protocole SPI
->Consommation au travail : 5mA
->Courant de veille : 1uA
->Taille : 4x2,8 cm
->Température de fonctionnement : -40 C - +85 C

Figure 21 : Module CAN MCP2515

Ce module particulier est basé sur le contrôleur IC MCP2515 CAN et l'IC émetteur/récepteur CAN TJA1050 :

- Le MCP2515 IC est un contrôleur CAN autonome possède une interface SPI intégrée pour la communication avec les microcontrôleurs.
- Le TJA1050, il agit comme une interface entre le CI contrôleur CAN MCP2515 et le bus CAN physique.

➢ Présentation de l'interface SPI

L'interface SPI a été développée par Motorola vers 1985. Il s'agit d'une interface série synchrone conçue pour une communication à courte distance entre dispositifs. Depuis, elle est devenue une norme de facto utilisée par de nombreux fabricants de semi-conducteurs, surtout dans les microcontrôleurs et les microprocesseurs. **[2]**

La SPI utilise un maximum de quatre lignes de signaux (Figure 22). Le dispositif maitre généralement un processeur ou un contrôleur, fournit et contrôle les lignes d'horloge

(SCK) et la détection de puce (CS). Le fonctionnement en multiplex intégral est géré par les lignes de données MOSI (Master Out, Slave In) et MISO (Master In, Slave Out)

Figure 22 : Interface SPI

Il s'agit d'une partie de la description des ECU (figure 5), l'interface de multiplexage composée du contrôleur de protocole et de l'interface de ligne. C'est ce que constitut le MCP2515 d'un nœud.

Le MCP2515 est un contrôleur CAN qui communique avec l'Arduino via le bus SPI et une ligne d'interruption qui permet au MCP2515 de signaler à l'Arduino qu'un message vient d'arriver ou vient de partir, deux événements que ses deux microcontrôleurs doivent prendre en charge (Figure 23).

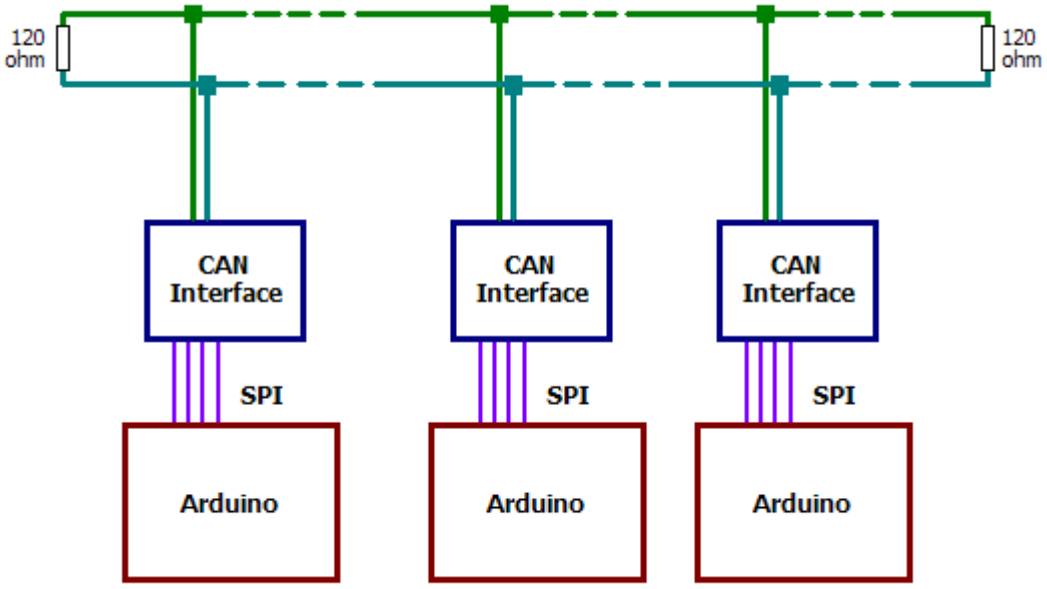


Figure 23 : Schème de connexion

I.3. Capteurs et composants

Un capteur est un dispositif transformant l'état d'une grandeur physique observée en une grandeur utilisable, telle qu'une tension électrique, une hauteur de mercure, une intensité ou la déviation d'une aiguille.

Les composants électroniques seront des valeurs ajoutées du projet, tel qu'un tableau de bord dans une voiture (afficheur LCD), le clackson (buzzer), la LED rouge (danger atteint).

I.3.a. Afficheur LCD

L'afficheur LCD est en particulier une interface visuelle entre un système et l'homme. Son rôle est de transmettre les informations utiles d'un système (capteur) à un utilisateur. Il affichera donc des données susceptibles d'être exploiter par l'utilisateur d'un système.

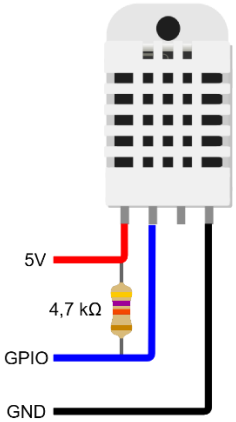
Figure 24 : Afficheur lcd+i2c

Pour simplifier le montage que nous aurons, nous allons utiliser l'afficheur LCD intégrant le I²C qui est un bus informatique. Il permet de relier facilement un microprocesseur et différents circuits tout en réduisant le nombre de lignes nécessaire à seulement deux lignes : SDA (Serial Data) et SCL (Serial Clock)

I.3.b. DHT22 : Capteur de température et d'humidité

Le DHT22 est un capteur numérique de base permettant de mesurer de manière efficace la température et l'humidité de l'air ambiant grâce à sa combinaison deux en un d'un capteur d'humidité capacitif et d'une thermistance.

De plus il se connecte facilement à votre microcontrôleur préféré grâce à un signal numérique sur un seul fil.



->Tension d'alimentation : 3 à 5V
->Plage de température : -40 à +80°C
->Humidité : de 0 à 100% RH
->Dimension : 15.1*25*7.7m
->broche 1) à gauche à la tension d'alimentation comprise entre 3 et 5V
->broche 2) à votre broche d'entrée de données.
->broche 3) non connecté
->broche 4) Masse.

Figure 25 : DHT22

I.3.c. Capteur ultrason

Les capteurs ultrason sont principalement utilisés dans l'industrie et servent à détecter et mesurer des distances entre divers types d'objets, quelle que soit leur forme (liquide, solide, granuleux, …). Ce sont des capteurs puissants et qui restent fiables même s'il y a présence de poussière ou si l'objet est brillant, transparent.

Figure 26 : Capteur ultrason

Un capteur à ultrasons émet à intervalles réguliers de courtes impulsions sonores à haute fréquence. Ces impulsions se propagent dans l'air à la vitesse du son. Lorsqu'elles rencontrent un objet, elles se réfléchissent et reviennent sous forme d'écho au capteur. Celui-ci calcule alors la distance le séparant de la cible sur la base du temps écoulé entre l'émission du signal et la réception de l'écho (*Figure 27*).

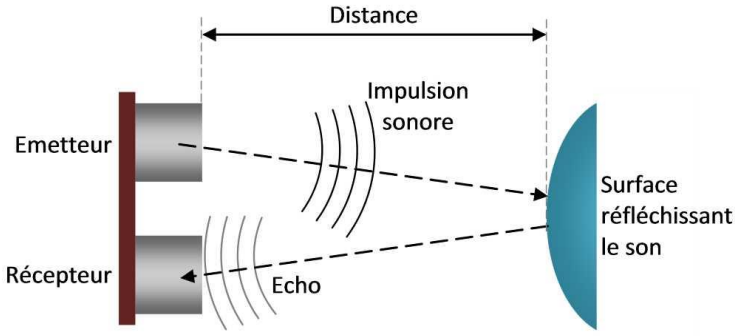


Figure 27 : Principe du capteur ultrason

I.3.d. Le buzzer Arduino

Un vibreur ou buzzer est un dispositif qui permet de tourner un signal électrique en onde sonore. Il contient une partie vibrante qui peut passer d'une position basse à une position haute. En fait la position haute ou basse du piezzo ou du buzzer est mécanique et fonctionne sur le principe de l'électro-aimant.

->Tension de fonctionnement : 3.3 - 6 V
->Signal de sortie : TTL (0 ou 1)
->Sensibilité réglable
->Dimensions : 34 x 16 x 15 mm

Figure 28 : Buzzer

I.2. Logiciels et éditeur de développement

Lors de notre réalisation, nous avons utilisés Arduino IDE et Fritzing. Le premier nous permet de programmer et configurer nos cartes Arduino pour le bon fonctionnement de nos capteurs et composants. Le deuxième nous permet de faire des circuits imprimés conçus de façon entièrement graphique et d'en imprimé le typon si nécessaire.

- Structure générale de l'environnement

L'IDE affiche une fenêtre graphique qui contient un éditeur de texte et tous les outils nécessaires à l'activité de programmation.

Figure 29 : Structure Arduino IDE

Vous pouvez donc saisir votre programme (langage proche du C), l'enregistrer, le compiler, le vérifier, le transférer sur une carte Arduino via une liaison USB de type B.

II. Application sur bus CAN

Grâce au microcontrôleur la carte Uno, nous allons réaliser deux nœuds. Le premier nœud contient deux capteurs : un ultrason et le DHT22. Le second se compose d'un afficheur LCD et d'un buzzer avec une LED de témoin.

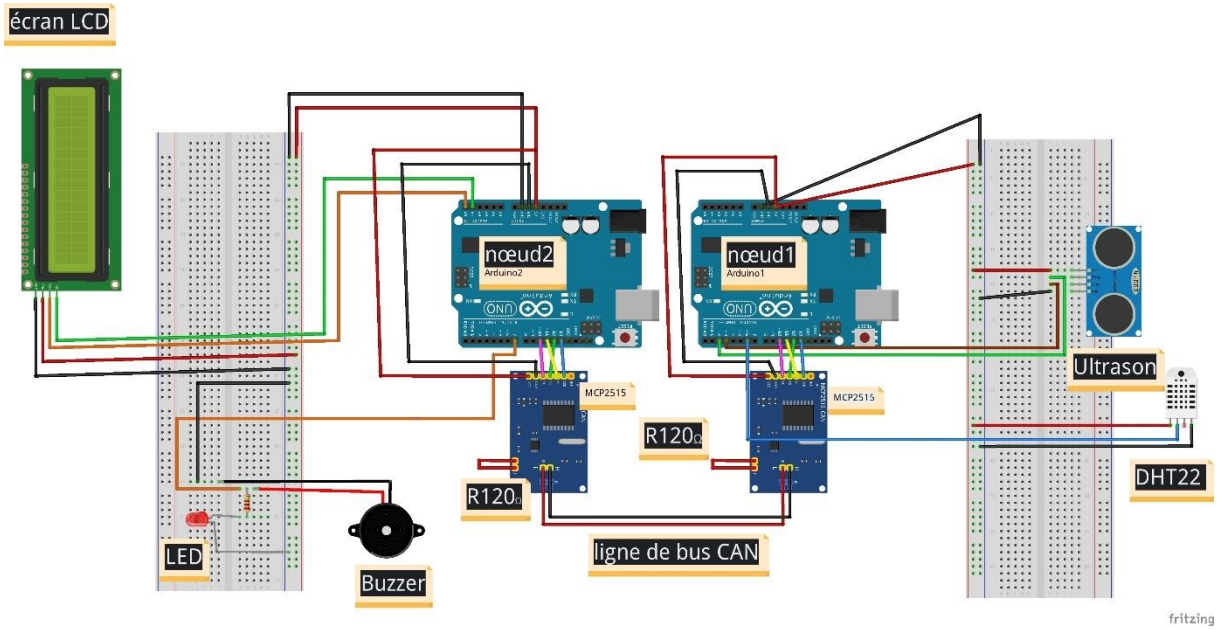


Figure 30 : Réseau de capteur bus CAN

Pin Arduino	Module CAN MCP2515
2	INT
10	CS
11	SI
12	SO
13	SCK

Tableau 4 : Connexion Arduino au MCP2515

II.1. Description du montage

Le capteur ultrason va calculer la distance d'un objet-obstacle qui se trouve dans sa devanture et nous signaler si l'obstacle en question à une distance donnée. La distance est transmise via le bus et afficher sur un écran LCD, mais aussi si l'objet est à moins de trois centimètres (distance critique) le buzzer se déclenche alertant l'utilisateur et la lampe rouge s'allume.

Le capteur DHT22 nous fournit la température et l'humidité d'un lieu exposé. Ces informations sont ensuite transmises dans le réseau et affichées dans l'écran. On peut souffler sur le capteur pour constater la variation rapide de l'humidité. A un certain moment on remarque également l'augmentation de la température mais brièvement et une fois qu'on arrête de ventiler le capteur, l'humidité revient plus ou moins à sa valeur et la température mets du temps pour se stabiliser.

II.2. Le programme Arduino (C)

Nous avons écrit les programmes d'émission et de réceptions suivants :

```
//Librairie de Communication
//SPI, CAN DHT22
#include <SPI.h>
#include <mcp2515.h>
#include <DHT.h>
#include <HCSR04.h>

#define dht22Pin 6
#define trigPin 7
#define echoPin 3
#define DHTYPE  DHT22
#define DELTA_TEMPERATURE 0.7

DHT dht(dht22Pin, DHTYPE);

UltraSonicDistanceSensor
distanceSensor(trigPin, echoPin);
```

```
//Librairie de Communication
//SPI, CAN DHT22
#include <SPI.h>
#include <mcp2515.h>
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2);

#define LEDBuzzerPin 7

struct can_frame canMsg1;
struct can_frame canMsg2;
struct can_frame canMsg3;

// SPI CS Pin 10
MCP2515 mcp2515(10);
```

```
float distance = 0.0;
float tempValue, humValue;

struct can_frame canMsg1;
struct can_frame canMsg2;
struct can_frame canMsg3;
MCP2515 mcp2515(10);

void setup(){

  //Commence la communication
  //série SPI à 9600 bauds
  Serial.begin(9600);
  SPI.begin();

  mcp2515.reset();
  //Définit CAN à la vitesse 500KBPS
  //et à l'horloge 8MHz
mcp2515.setBitrate(CAN_500KBPS,MCP_8MHZ);
  mcp2515.setNormalMode();

  //CAN_id comme 0xAA pour distance
  canMsg1.can_id  = 0xAA;
  //Longueur des données CAN comme 1
  canMsg1.can_dlc = 1;
  //CAN_id comme 0xBB pour la température
  canMsg2.can_id  = 0xBB;
  //Longueur des données CAN comme 1
  canMsg2.can_dlc = 1;
  //CAN_id comme 0xCC pour l'humidité
  canMsg3.can_id  = 0xCC;
  //Longueur des données CAN comme 2
  canMsg3.can_dlc = 2;

  dht.begin();
}

void loop(){

  Serial.print(canMsg1.can_id);
  distance =
distanceSensor.measureDistanceCm();
  tempValue = dht.readTemperature() -
DELTA_TEMPERATURE;
  humValue = dht.readHumidity();

  Serial.print("Temp value deg C : ");
  Serial.println(tempValue);
  Serial.print("Humidity value : ");
  Serial.println(humValue);
  Serial.print("Distance  : ");
  Serial.println(distance);
  Serial.print(" cm");

  canMsg1.data[0] = distance;
  canMsg1.data[1] = 0x00;
  //Envoie le message CAN
  mcp2515.sendMessage(&canMsg1);
```

```
void setup() {

  lcd.init();
  pinMode(LEDBuzzerPin, OUTPUT);
  delay(1000);

  //Commence la communication
  //série SPI à 9600 bauds
  SPI.begin();
  Serial.begin(9600);

  mcp2515.reset();
  //Définit CAN à la vitesse 500KBPS
  //et à l'horloge 8MHz
  mcp2515.setBitrate(CAN_500KBPS,
MCP_8MHZ);
  //Définit CAN en mode normal
  mcp2515.setNormalMode();

}

void loop(){

  lcd.backlight();
  // Pour recevoir des données
  if (mcp2515.readMessage(&canMsg1) ==
MCP2515::ERROR_OK) {

    if (canMsg1.can_id == 0xAA) {
      float x = canMsg1.data[0];
      Serial.println(x);
      lcd.setCursor(0, 0);
      lcd.print("D:");
      lcd.print(x);
      if(x < 3){
      // LED s'allume et Buzzer sonne
        digitalWrite(LEDBuzzerPin,
HIGH);
        digitalWrite(4, LOW);
      }else {
        // LED etein et Buzzer arrête
        digitalWrite(LEDBuzzerPin,
LOW);
      }
      delay(200);
    }
  }

  if (mcp2515.readMessage(&canMsg2) ==
MCP2515::ERROR_OK)  {

    if (canMsg2.can_id == 0xBB){

      int y = int(canMsg2.data[0]);
      Serial.print("Temperature :");
      Serial.println(y);
      lcd.setCursor(7, 0);
      lcd.print("T(deg):");
      lcd.print(y);
```

```
  delay(200);

  canMsg2.data[0] = tempValue;
  //Envoie le message CAN
  mcp2515.sendMessage(&canMsg2);
  delay(200);

  canMsg3.data[0]=0x00;
  canMsg3.data[1] = humValue;
  //Envoie le message CAN
  mcp2515.sendMessage(&canMsg3);
  delay(200);

  delay(100);

}
```

```
    delay(200);
   }
  }

 if (mcp2515.readMessage(&canMsg3) ==
MCP2515::ERROR_OK)  {

    if (canMsg3.can_id == 0xCC)  {

      int z = int(canMsg3.data[1]);
      Serial.print("Humidité :");
      Serial.println(z);
      lcd.setCursor(4, 1);
      lcd.print("H(%):");
      lcd.print(z);
      delay(200);

    }
  }
  delay(200);
}
```

Tableau 5 : Programme Emission/Réception

III. Conclusion

Ce chapitre détaille l'ensemble des éléments que nous allons utiliser pour la mise en œuvre de notre projet. La réalisation d'un bus CAN à partir des cartes Arduino, nous a permis de concevoir notre propre programme tout en partant des algorithmes de bases Emetteur/Récepteur de la librairie incluse *mcp2515.h.*

La mise en pratique d'un tel réseau de terrain nous a permis d'acquérir de nouvelles connaissances dans le domaine des systèmes embarqués, de la programmation qu'il exige dans les microcontrôleurs.

Conclusion et perspectives

Durant la réalisation de ce projet, nous nous sommes plus intéressés dans la réalisation d'un bus CAN qui pouvait également se faire à partir des PIC18F25K80 intégrant le transceiver, la démarche serait la même avec moins de composants. Ce qui fut l'objectif de départ.

À travers ce document, nous avons fait une présentation du Bus CAN dans sa généralité, ensuite connaitre les intérêts de ce bus de communication et de comment est-il composé dans le modèle OSI. Enfin nous avons décrit les caractéristiques d'un réseau BUS CAN et les informations qui le transmit ainsi que les différents types d'erreur et leur fonctionnement.

Le bus CAN reste le réseau de terrain le plus utilisé dans l'industrie automobile. Sur cette analyse, il est le plus imposant du fait de son faible coût de mise en œuvre d'un pair torsadé, de sa robustesse de fonctionnement et de son système de gestion de priorités.

La deuxième partie constituant en général la réalisation de notre projet. Les cartes et Arduino Uno nous ont permis de faire un grand exploit dans nos travaux. Par le module MCP2515 nous avons réalisé une liaison série SPI vers les microcontrôleurs. Ainsi, avec une liaison de type bus I²C nous avons relié dans l'afficheur LCD réduisant le nombre de fils. Le montage de notre réalisation est effectué grâce au logiciel libre Fritzing.

Toutefois, il serait judicieux d'inclure un système d'analyseur de trame afin de pourvoir stocker toute information circulant dans le bus sans disfonctionnement du réseau. Dans la mesure d'une évolution au détriment d'un certain nombre de réseau de terrain (I²C, Flex Ray, LIN), le bus CAN standard doit augmenter son débit de transmission (1Mbps) pour mieux se valoriser dans le domaine des systèmes embarqués.

Références et Bibliographies

[1] Mémoire de Projet de fin d'étude de Master : Etude et Réalisation d'un bus CAN, 2019

[2] **https://www.digikey.fr/fr/articles/why-how-to-use-serial-peripheral-interface-simplify-connections-between-multiple-devices**, Interface SPI, le 23/05/2022 à 21:04

[3] **https://www.technologuepro.com/cours-systemes-embarques/cours-systemes-embarques-Bus-CAN.htm** Protocole de communication, trame de données, le 29/04/2022 à 18:17

[4] **https://fr.farnell.com/microchip/mcp2515-i-so/controleur-can-spi-cms-soic18/dp/1292239**, Datasheet MCP2515, le 18/05/2022

[5] **https://hal.archives-ouvertes.fr/hal-00182292/document,** le 29/06/2022 à 17:19

[6] **https://fr.wikipedia.org/wiki/CSMA#:~:text=%C3%AAtre%20tr%C3%A8s%20long.-,CSMA%2FCD%20(Collision%20Detection),de%20la%20disponibilit%C3%A9%20du%20m%C3%A9dia.,** CSMA/CD (Collision Detection), 07/06/2022 à 19:43