



PROJETS TUTORÉS

Serveur Web embarqué pour réseau CAN

WALTER Thomas – CHEVROULET Ludovic

Proposé par : LOMBARD Christophe



Remerciement :

Avant de débiter ce rapport, nous tenons à adresser nos remerciements à tous ceux qui nous ont aidés, de près ou de loin, à réaliser ce projet.

Nous tenons aussi à remercier tout particulièrement Mr. LOMBARD Christophe de nous avoir encadrés tout au long de ce projet. Il a été toujours disponible ainsi qu'à l'écoute de nos questions.

Nous remercions également Mr. GUSTIN Frédéric qui a su se rendre disponible pour faire avancer ce projet, ainsi que Mr. HUBERT pour avoir imprimé les cartes électroniques.

TABLE DES MATIÈRES

1.	INTRODUCTION	3
1.1	OBJECTIFS	3
1.2	DESCRIPTIF DU PROJET	3
1.3	SCHÉMAS DE PRINCIPE	3
2.	ÉTUDE THÉORIQUE.....	4
2.1	DÉCOUPAGE DES TACHES.....	4
2.2	SYNOPTIQUE.....	5
2.2.1	Carte embarquée.....	5
2.2.2	Carte serveur	5
2.3	COMMUNICATION.....	6
2.3.1	Bus SPI	6
2.3.2	Liaison RF	6
2.3.3	Liaison Ethernet.....	6
2.3.4	Bus Can	7
2.4	CHOIX DES COMPOSANTS	8
3.	RÉALISATION	11
3.1	SCHÉMAS ÉLECTRIQUE DE LA PREMIÈRE ÉTAPE (SANS LIAISON RF).....	11
3.1.1	Photo de la carte	11
3.1.2	Problèmes rencontrés	12
3.2	CARTE ÉLECTRIQUE DE LA SECONDE ÉTAPE (AVEC LIAISON RF)	13
3.2.1	Photo de la carte	13
3.2.2	Problèmes rencontrés	13
3.3	ORGANIGRAMMES	14
3.3.1	Carte embarquée.....	14
3.3.2	Carte serveur	15
3.4	SITE WEB.....	16
3.4.1	Carte SD	16
3.4.2	Page WEB.....	16
3.4.3	Fichier « HISTORIQUE »	17
4.	CONCLUSION	18
5.	BIBLIOGRAPHIE.....	19



1. INTRODUCTION

1.1 OBJECTIFS

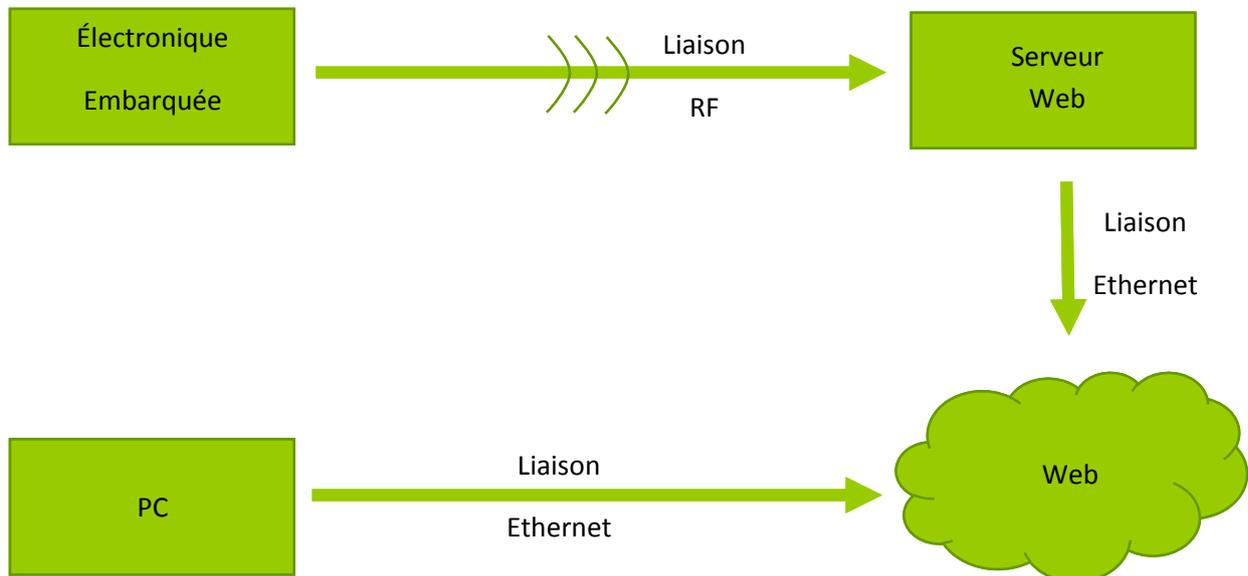
Mettre en œuvre un serveur Web embarqué à base d'Arduino et récupérer des informations véhicule (vitesse ou régime moteur ou autres) à partir d'un navigateur.

1.2 DESCRIPTIF DU PROJET

Il faut fournir une organisation matérielle détaillée à base de microcontrôleur Atmega328 (ATMEL) dont le but est de décoder et transmettre les informations d'une trame de véhicule CAN2A (envoyée par un script développé en CAPL à partir de Canalyzer). Un shield Ethernet (fourni) permet l'envoi des informations sur le réseau pour affichage sur un navigateur (prévoir HTML5 et CSS).

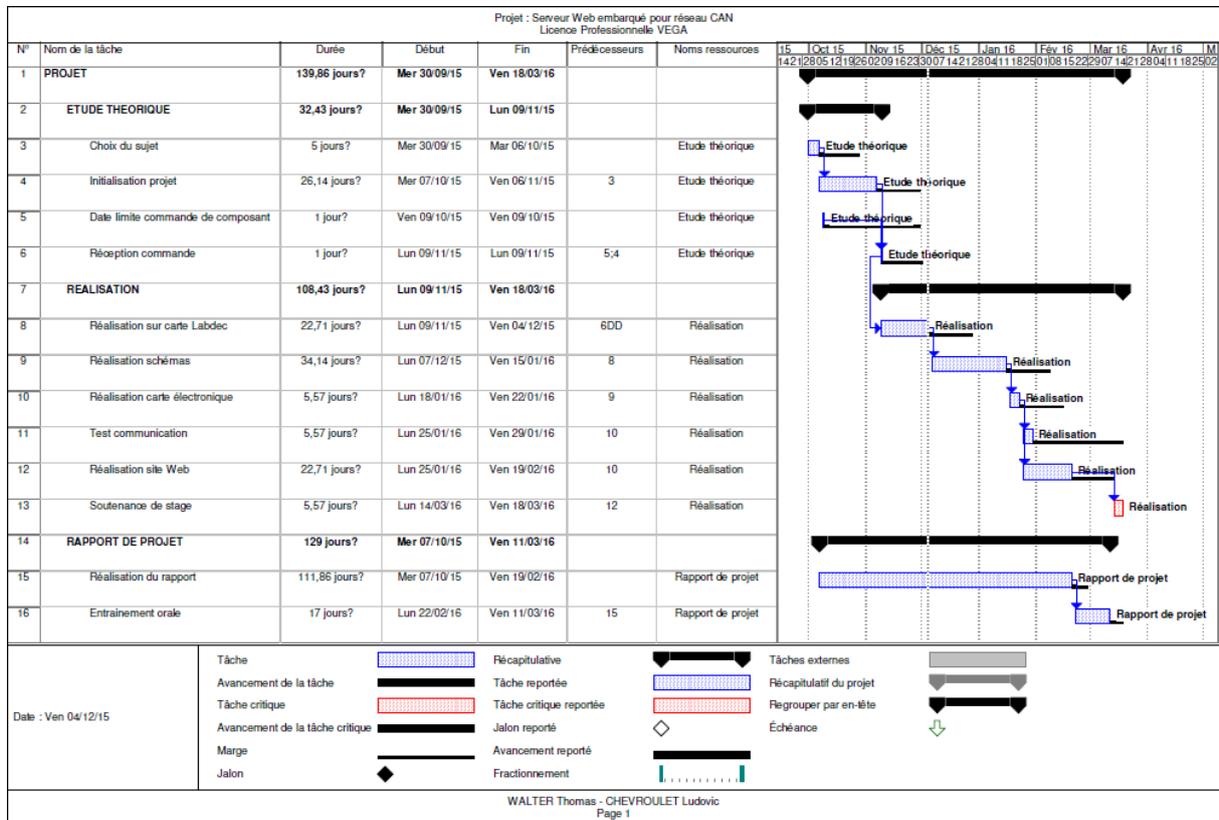
Ensuite, après validation, il sera envisagé une communication RF entre l'électronique embarquée et l'interprétation des données (PC).

1.3 SCHÉMAS DE PRINCIPE



2. ÉTUDE THÉORIQUE

2.1 DÉCOUPAGE DES TACHES

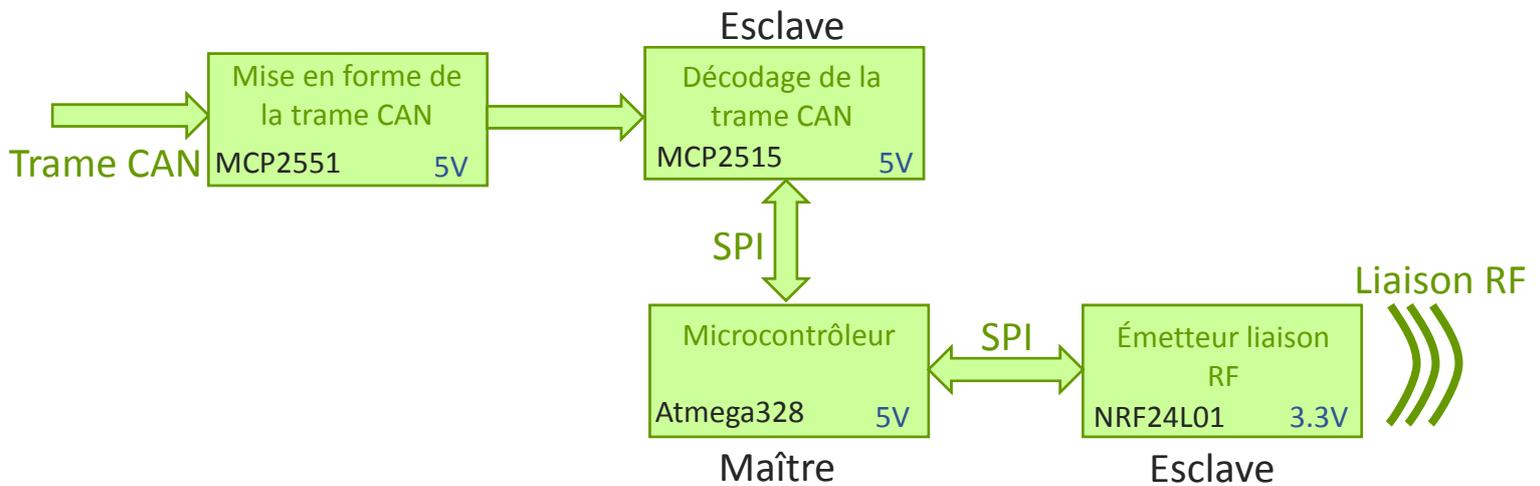


Pour réaliser ce projet, nous avons établi ce diagramme de Gantt.

Il y a 3 parties distinctes : une étude théorique permettant de bien comprendre le sujet et de choisir les différents composants que comportera la carte. La partie réalisation, qui comporte la fabrication de la carte électronique ainsi que le site Web. La troisième partie est la réalisation du rapport du projet.

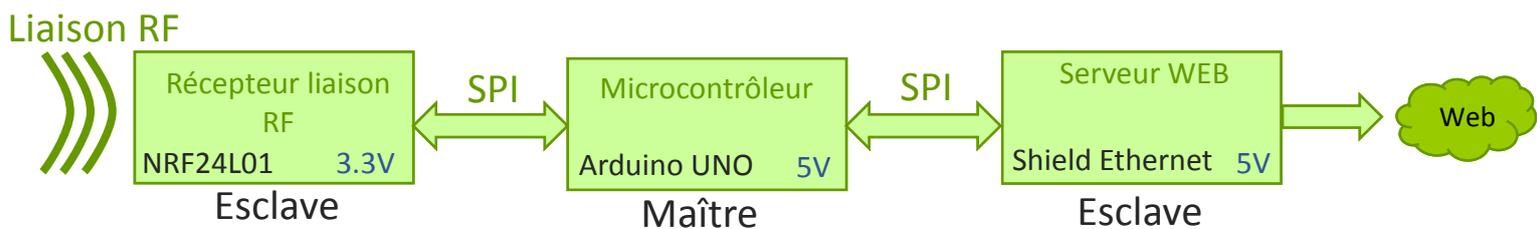
2.2 SYNOPTIQUE

2.2.1 Carte embarquée



Voici le principe de fonctionnement de la carte embarquée. Une trame CAN générée par l'ordinateur (ou par un calculateur de voiture) arrive sur le MCP2551. Cette trame va être transformée en trame CAN TTL (0-5V) ensuite grâce au MCP2515 elle va se mettre sous forme SPI pour finalement être traitée par le microcontrôleur Atmega328. L'identifiant et la donnée de la trame vont être envoyés par un module RF (NRF24L01)

2.2.2 Carte serveur

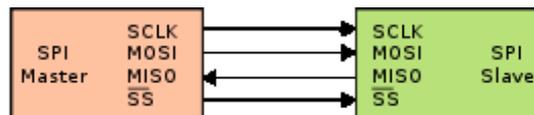


Une carte NRF24L01 reçoit en RF les identifiants ainsi que les données. L'Arduino, aidé de l'Ethernet Shield va transmettre ces informations sur un site Web (stocké dans une carte SD), grâce à la liaison Ethernet.

2.3 COMMUNICATION

2.3.1 Bus SPI

Le SPI (pour Serial Peripheral Interface) est un bus de données série qui opère en mode Full-duplex. Les circuits communiquent selon un schéma maître-esclave, où le maître s'occupe totalement de la communication. Plusieurs esclaves peuvent coexister sur un même bus, dans ce cas, la sélection du destinataire se fait par une ligne dédiée entre le maître et l'esclave appelée chip select.



Avec :

- SCLK — Serial Clock, Horloge (généré par le maître)
- MOSI — Master Output, Slave Input (généré par le maître)
- MISO — Master Input, Slave Output (généré par l'esclave)
- SS — Slave Select, Actif à l'état bas (généré par le maître)

2.3.2 Liaison RF

La liaison RF (Radio Frequency) est une liaison sans fils. Dans notre cas, nous allons utiliser un composant permettant de transmettre les données sur la fréquence 2,4 GHz avec une vitesse de transmission de 1 ou 2 Mb/s

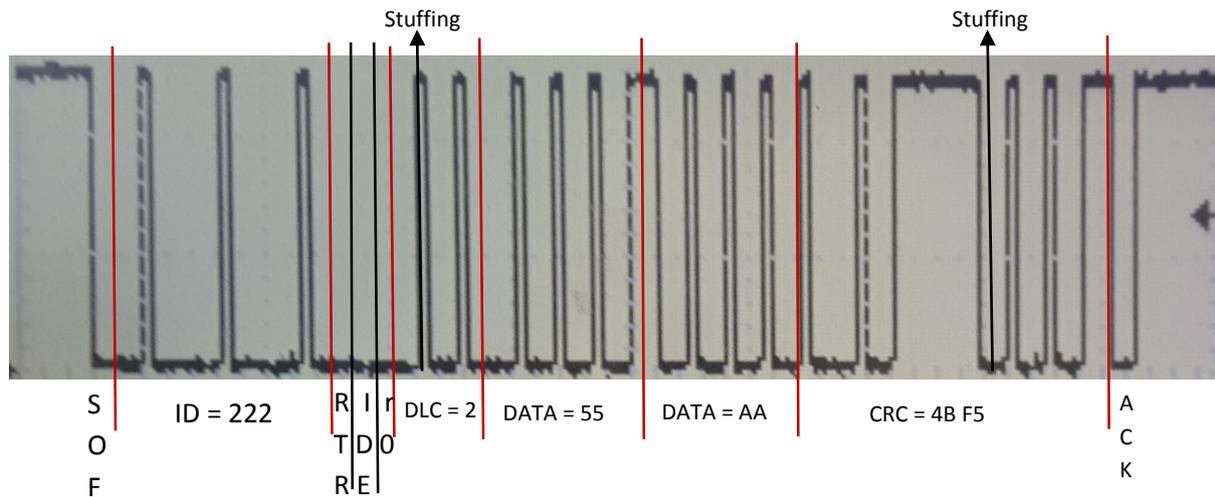
2.3.3 Liaison Ethernet

La liaison Ethernet permettra à la carte Arduino de communiquer avec le réseau Ethernet, afin de pouvoir accéder au site WEB embarqué sur la carte SD.

2.3.4 Bus Can

Le bus CAN (Controller Area Network) est un bus système série très répandu dans beaucoup d'industries, notamment l'automobile. Il met en application une approche connue sous le nom de multiplexage, et qui consiste à raccorder à un même câble (un bus) un grand nombre de calculateurs qui communiqueront ensemble.

Structure du bus CAN :



L'**identifiant** est l'adresse du message à transmettre.

Le **DLC** permet de définir le nombre d'octets envoyés dans le message.

DLC				Nb octets
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

Le **CRC** permet de contrôler la validité des données à transmettre.

Les **DATA** sont les octets à transmettre.

L'**ACK** permet d'indiquer que la trame CAN est bien reçue.

2.4 CHOIX DES COMPOSANTS

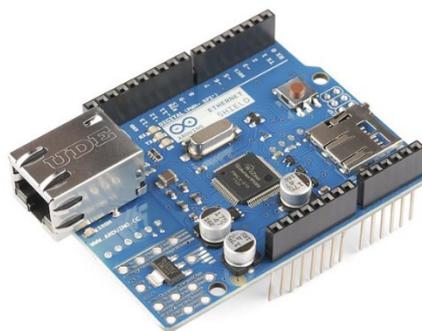
Arduino UNO (carte serveur)



Le choix de la carte Arduino Uno était imposé par le sujet. La carte Arduino Uno est une carte avec un microcontrôleur Atmega 328, il dispose de 14 broches numériques (entrée/sortie). Le microcontrôleur est programmé via câble USB et possède son langage de programmation Arduino.

Cette carte peut communiquer avec d'autres composants grâce au bus SPI.

Shield Ethernet (carte serveur)



Le Shield Ethernet permet de pouvoir héberger un serveur Web et pouvoir y accéder avec un ordinateur branché sur le réseau.



Atmega 328 (carte embaquée)



Le microcontrôleur Atmega 328, il dispose de 14 broches numériques (entrée/sortie). Il remplacera la carte Arduino, afin de pouvoir avoir un système autonome.

78L05 Régulateur interne de tension 5V (carte embarquée)



Il permet de transformer du 12 V continue en 5 V continue, ceci permet au système embarqué d'être autonome.

MCP1702 Régulateur interne de tension 3.3V (carte embarquée)



Il permet de transformer du 5 V continue en 3.3 V continue, ceci permet au système embarqué d'être autonome.



MCP2551 émetteur-récepteur CAN (carte embarquée)



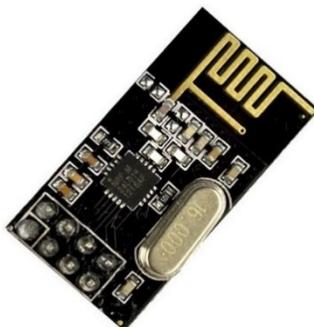
Le bus CAN en sortie de l'ordinateur n'est pas un signal 0-5V. Il faut donc, pour pouvoir le traiter, un composant permettant de mettre les niveaux de tension adéquats. Le MCP2551 est un émetteur-récepteur pouvant réaliser cette fonction.

MCP2515 : contrôleur CAN avec interface SPI (carte embarquée)



Une fois le bus CAN sous forme TTL (0-5V) il faut récupérer les données que contient la trame. Cette fonction sera réalisée par la carte Arduino pendant il faut convertir le bus CAN en bus SPI, le MCP2515 réalisera cette fonction.

NRF24I01 (carte embarquée et carte serveur)



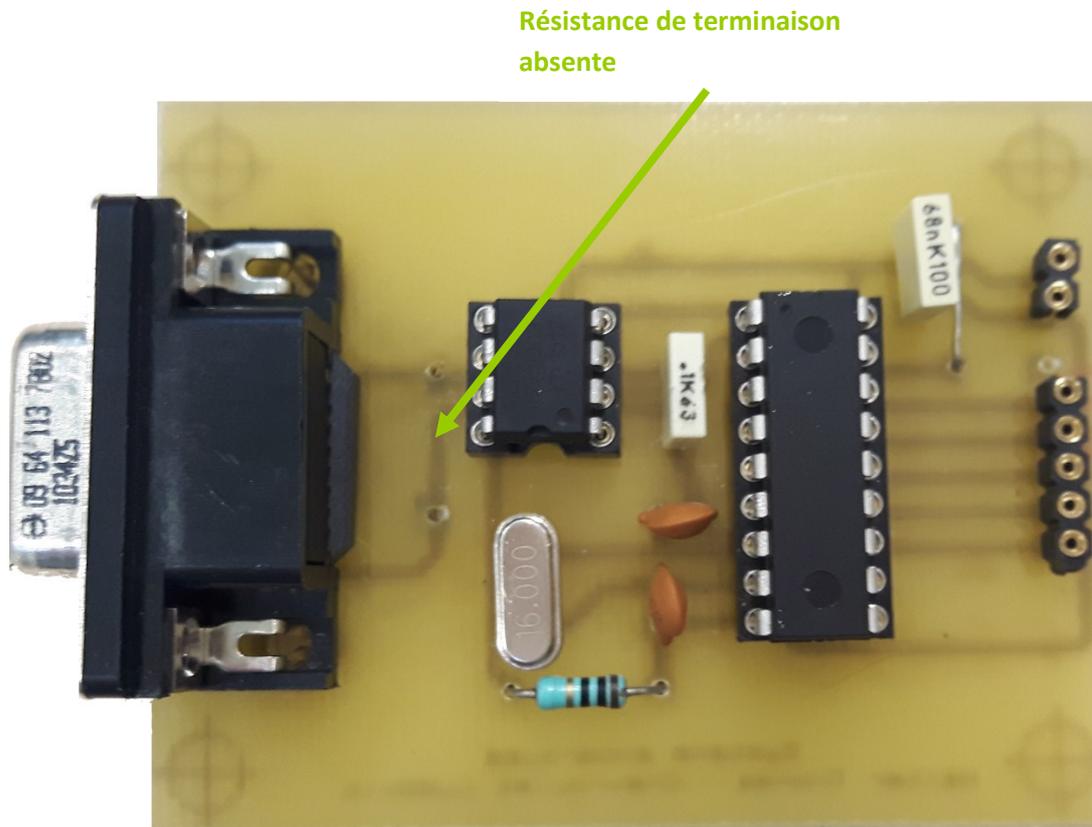
Carte de transmission et réception sans fils. Elle permettra la communication entre la carte embarquée et la carte serveur.



3. RÉALISATION

3.1 SCHÉMAS ÉLECTRIQUE DE LA PREMIÈRE ÉTAPE (SANS LIAISON RF)

3.1.1 Photo de la carte



Nous avons réalisé cette carte dans le but afin de pouvoir créer et tester le programme Arduino. Ceci nous a permis de mieux comprendre le fonctionnement des différents composants ainsi que le bus SPI. Nous avons aussi commencé la création du site web avec cette carte de test. Après validation de cette carte, nous avons pu envisager la réalisation d'une carte avec une liaison sans fils (liaison RF).

Nous avons constaté que notre carte ne fonctionnait pas, après une recherche de panne et de nombreuse tentative de correction, la carte a fonctionné.

3.1.2 Problèmes rencontrés

Problème : Lors de la transmission de la trame CAN, une erreur apparaît sur le logiciel

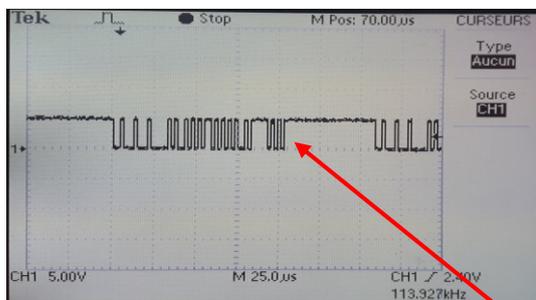
CANalyser :

Time	Chn	ID	Name	Event Type	Dir	DLC	Da...	Data
1.512302	CAN 1			CAN Error	TxEr			ECC: 11011001, Other Type of Error, Segment = ACK Slot
ECC : 11011001								
Code : Other Type of Error								
Segment: ACK Slot								

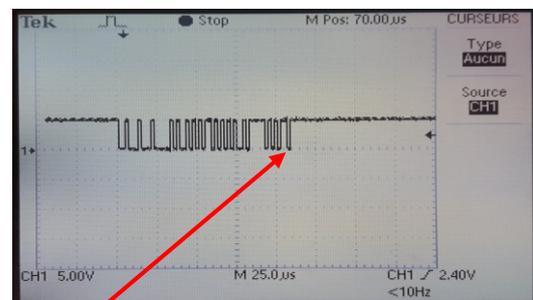
Recherche de solution :

- Notre premier montage possédait un quartz à 20 MHz, il a donc été changé à 16MHz pour correspondre à la fréquence de la carte Arduino
- Ajout d'une résistance R_s au MCP2551, permettant d'autoriser les fronts brutaux de tension.
- Remplacement du MCP2515.
- Grâce à l'achat d'une carte : CAN bus Shield (Arduino), nous pouvons mieux confirmer ce qu'il fonctionne et ce qu'il ne fonctionne pas sur notre montage :

Le problème vient du bit ACK qui n'est pas mis à 0 par le contrôleur CAN (MCP25 b15)



Carte embarquée



CAN Bus-Shield

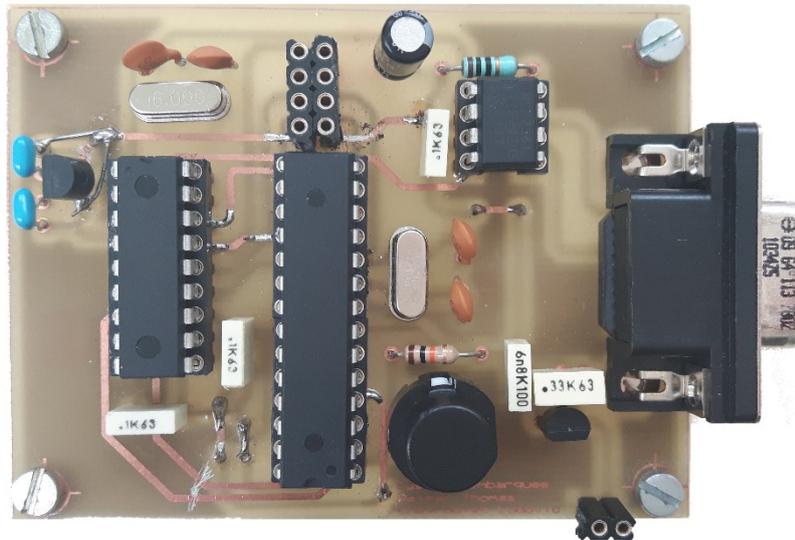
ACK

Solution :

- Nous avons retiré la résistance de terminaison de 120 Ω , et nous envoyons une trame de 125 kbit/s.

3.2 CARTE ÉLECTRIQUE DE LA SECONDE ÉTAPE (AVEC LIAISON RF)

3.2.1 Photo de la carte



3.2.2 Problèmes rencontrés

Problème : Problème de package sur 1702 (Régulateur de tension 3.3 V)

Solution : Rectification de la carte

Problème : Problème intermittent de la transmission RF

Solution : Ajout d'une capacité de filtrage entre le 3.3V et la masse

Problème : Lors de la transmission de la trame CAN, une erreur apparaît sur le logiciel

CANalyser :

Time	Chn	ID	Name	Event Type	Dir	DLC	Da...	Data
1.512302	CAN 1			CAN Error	TxEr			ECC: 11011001, Other Type of Error, Segment = ACK Slot
			ECC :					11011001
			Code :					Other Type of Error
			Segment:					ACK Slot

Recherche de solution :

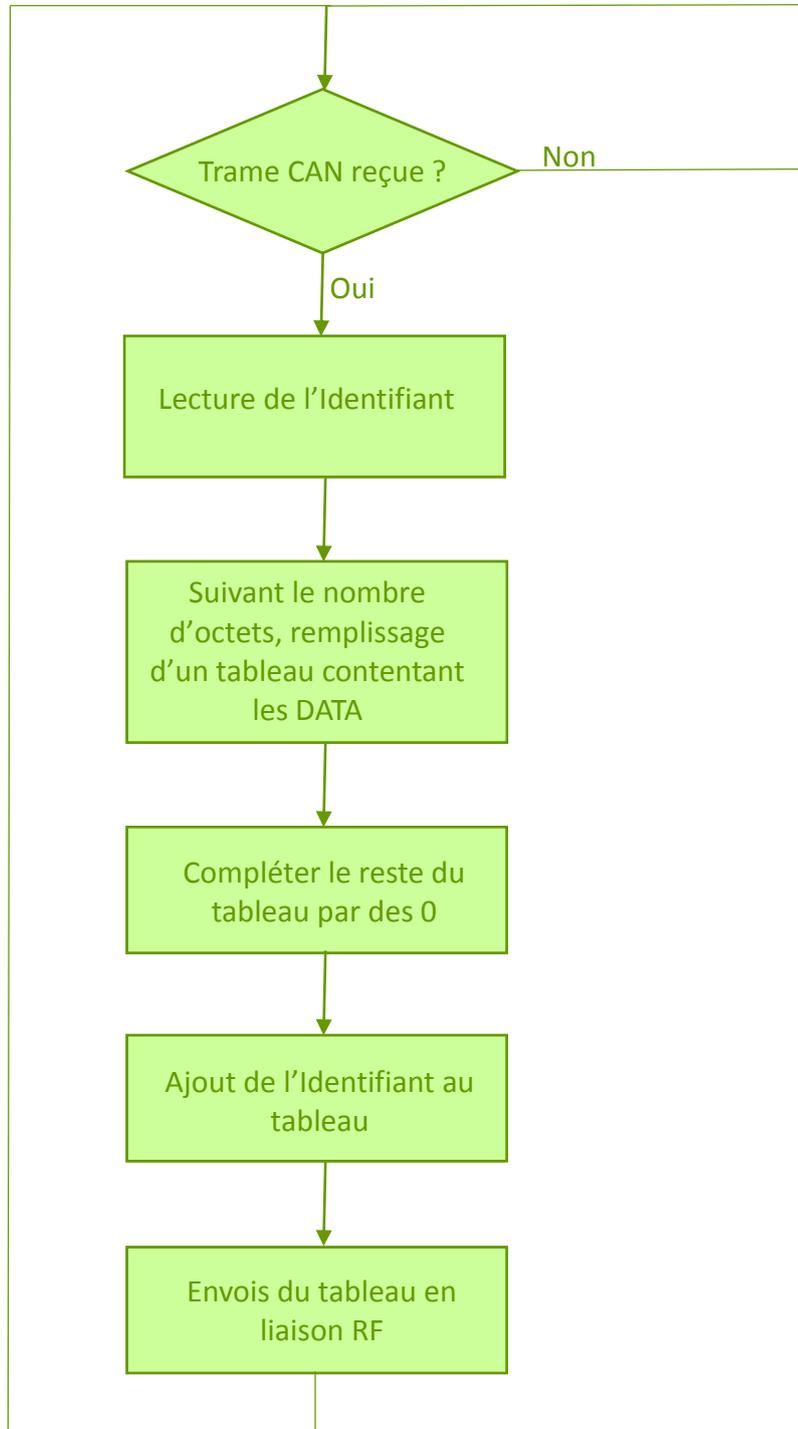
- Test des différents modules grâce à la carte fonctionnelle (MCP 2515, MCP2551 et Atmega) grâce à ces tests, nous avons pu déterminer que le problème venait de l'Atmega

Solution :

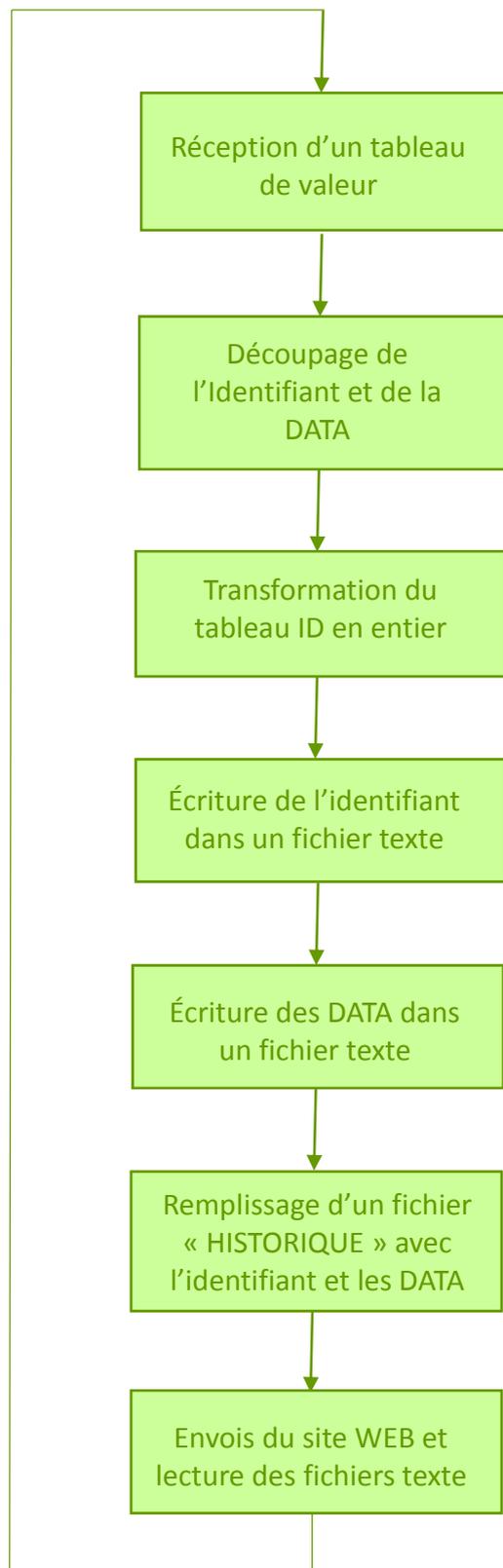
- Ajout d'un second quartz (avec ses condensateurs) afin d'en avoir un pour l'Atmega et un pour le MCP2515

3.3 ORGANIGRAMMES

3.3.1 Carte embarquée



3.3.2 Carte serveur



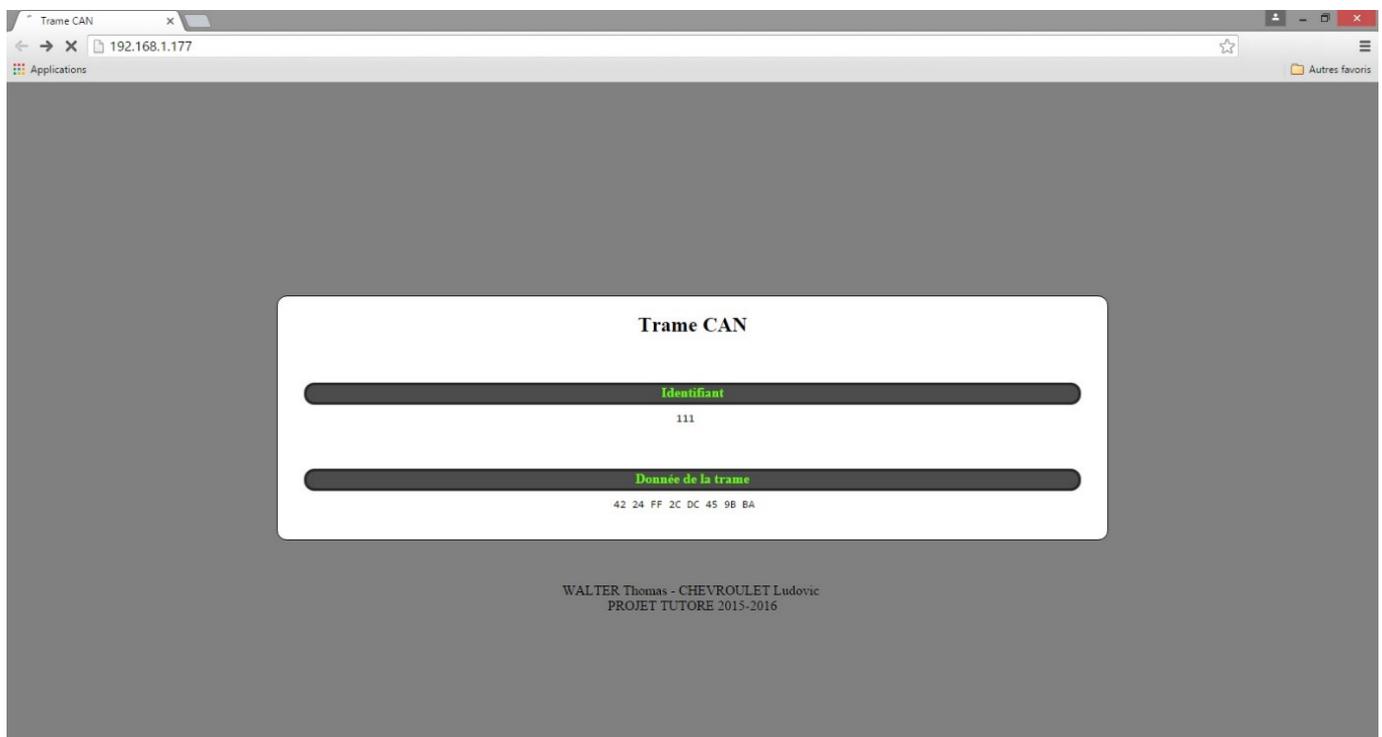
3.4 SITE WEB

3.4.1 Carte SD

 DATA.TXT	02/03/2016 09:01	Document texte	1 Ko
 HISTORIQUE.txt	02/03/2016 09:01	Document texte	9 Ko
 ID.TXT	02/03/2016 09:01	Document texte	1 Ko
 index.htm	02/03/2016 08:54	Chrome HTML Do...	3 Ko

La carte SD comporte 4 fichiers : le site HTML, contenant la structure du site, un fichier où est stocké le dernier identifiant envoyé, un fichier où est stocké les dernières DATA envoyée, et enfin un fichier regroupant toute les trame CAN envoyée.

3.4.2 Page WEB



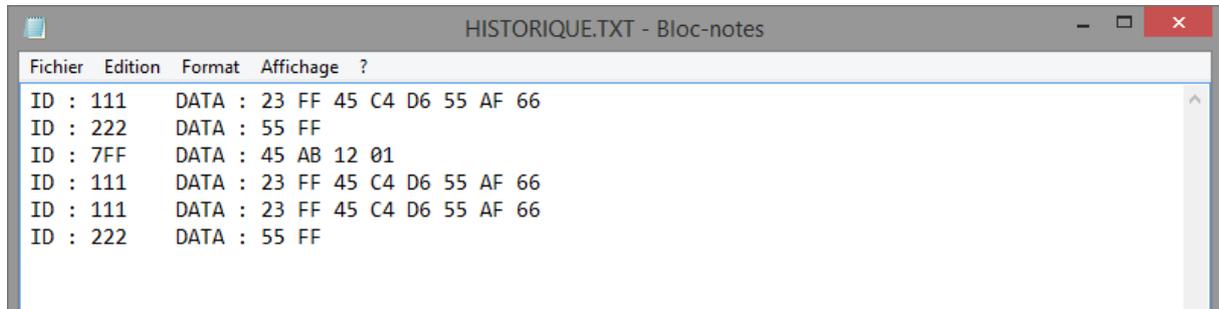
Pour accéder au site web embarqué sur la carte Arduino il faut connecter le câble Ethernet sur le réseau et inscrire l'adresse : 192.168.1.177 sur un navigateur internet.

Nous pouvons voir que l'adresse, ainsi que la data de la trame CAN dans les deux champs prévus à cet effet.

Le site Web se recharge automatiquement toutes les 5 secondes, étant donné que nous n'avons pas réussi à faire recharger que les valeurs des trames (ID et DATA). Pour ne perdre aucune trame, un fichier LOG (enregistré dans la carte SD) récupère toutes les trames envoyées par le réseau CAN.

3.4.3 Fichier « HISTORIQUE »

Le fichier « HISTORIQUE » a été créé dans le but de palier au temps de rechargement de la page WEB



```
HISTORIQUE.TXT - Bloc-notes
Fichier  Edition  Format  Affichage  ?
ID : 111  DATA : 23 FF 45 C4 D6 55 AF 66
ID : 222  DATA : 55 FF
ID : 7FF  DATA : 45 AB 12 01
ID : 111  DATA : 23 FF 45 C4 D6 55 AF 66
ID : 111  DATA : 23 FF 45 C4 D6 55 AF 66
ID : 222  DATA : 55 FF
```

Nous pouvons voir que chaque trame envoyée par le réseau est stockée ici. Chaque Identifiants possède sa DATA correspondant.

4. CONCLUSION

La réalisation de ce projet a été très intéressante, grâce à son application concrète sur le bus CAN. De plus, ce projet regroupe des fondamentaux appris au cours de la licence professionnelle électricité et électronique option VEGA, telle que l'informatique industrielle pour la partie programmation sur microcontrôleur, et également l'électronique pour la réalisation des cartes électroniques.

Nous avons pu nous rendre compte de la difficulté de tenir un calendrier établi à l'avance grâce au diagramme de Gant, suite à de nombreux problèmes rencontrés au cours de ce projet. Grâce à la persévérance et de l'aide de nos professeurs, nous avons résolu la quasi-totalité des problèmes et de proposer un système fonctionnel.

Malgré que le projet soit mené à terme, des améliorations peuvent être apportées. Il faudrait d'abord recréer une carte électronique propre (proposée en Annexe), car la carte électronique réalisée comportant de nombreuses corrections. Il manque également sur celle-ci, un interrupteur ainsi qu'un témoin lumineux pour indiquer si la carte est bien alimentée. Le site peut être amélioré en trouvant la solution pour actualiser seulement les données variantes : Identifiant et Data. La transmission RF pourrait être perfectionnée afin d'obtenir une distance de réception plus grande, nous possédons actuellement une distance d'environ 10 mètres.



5. BIBLIOGRAPHIE

Documentation technique de l'Atmega328 (ATMEL) :

http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

Consulté le : 07/10/2015

Documentation technique du MCP2551 (Microchip) :

<http://ww1.microchip.com/downloads/en/DeviceDoc/21667f.pdf>

Consulté le : 07/10/2015

Documentation technique du MCP2515 (Microchip) :

<http://ww1.microchip.com/downloads/en/DeviceDoc/21801G.pdf>

Consulté le : 07/10/2015

Note d'application du MCP2515 (Microchip) :

<http://ww1.microchip.com/downloads/en/AppNotes/00215c.pdf>

Consulté le : 08/10/2015

Schemas du CAN bus Shield (Arduino) :

http://www.seeedstudio.com/wiki/images/7/78/CAN-BUS_Shield_v0.9b.pdf

Consulté le : 05/01/2016

Documentation technique du MCP1702 (Microchip) :

<http://ww1.microchip.com/downloads/en/DeviceDoc/22008E.pdf>

Consulté le : 03/02/2016



Documentation technique du 78L05 (TI) :

<http://www.ti.com/lit/ds/symlink/lm78l05.pdf>

Consulté le : 03/02/2016

Documentation de l' Ethernet Shield (Arduino):

<https://www.arduino.cc/en/Main/ArduinoEthernetShield>

Consulté le : 05/01/2016

