Présentation TER 2015

Protocole SimpleCAN pour bus à base d'Arduino

Romain Le Forestier
M1 SICLE

Sommaire

- **Contexte**
- **Présentation du protocole CAN**
  - Historique
  - Caractéristiques physique
  - Principe du protocole
- **Présentation de la carte**
- **Programmation d'une carte**
- **Présentation de SimpleCAN**

Contexte

- **Sujet : Protocole SimpleCAN pour bus à base d'Arduino**
  - Exploiter un projet existant
  - Utilisation de matériel existant
  - Mise en œuvre du réseau

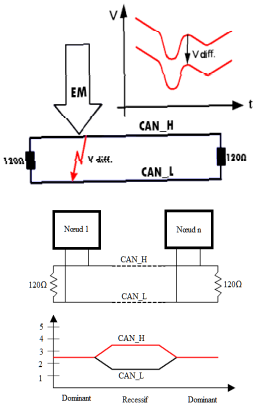
Projet s’inscrivant dans le cadre du travail d'étude et de recherche
Le Sujet du Ter consistait a réutiliser un projet existant, le Kévin Bruget, étudiant en 2013 à l'ENSTA Bretagne, pour son son projet de fin d'études pour lequel il avait étudié la possibilité de remplacer un système d'assistance à la navigation centraliser par un système réparti via un réseau CAN à base de carte Arduino,
pour voir sa compatibilité avec notre projet pour un réseaux de capteur pour un auto pilote de voilier et exploiter le matériel existant :
5 carte arduino avec une interface CAN
Protocole et méthode existante (simpleCan)
Le but final étant d'avoir un réseau de capteur utilisant un bus CAN

Le protocole CAN

- **Historique**
  - Développé par Bosch et l'Université de Karlsruhe dans les années 1980
  - Standardisé par un standard ISO au début de l'année 1990
  - Formation de CAN in Automation en 1992

Le protocole CAN(Controller Area Network) est un bus de terrain développer à l'origine pour le secteur de l'automobile par Bosch et l'Université de Karlsruhe dans les années 1980 et standardisé par les standards ISO au début des années 1990.
Le Développement de ce bus avait pour but de simplifier le câblage et réduire le nombre de câbles dans les voitures qui dans les années 1980 se complexifiait, principalement à cause de l'accroissement du nombre de capteur et de système électronique afin de permettre aux voitures de répondre aux exigences environnementales, de sécurité (ABS, ESP, AIR-BAG...) et une demande de confort (climatisation automatique, système de navigation …).
(CIA), une organisation a but non lucratif dont l'objectif est de fournir des informations techniques et promouvoir le protocole CAN.
Actuellement composer de 560 entreprises
Le protocole CAN c'est répandu du domaine de l'automobile pour lequel il fut développé au autres domaine du transport (aérospatiale, engin de chantier, bateau), au domaine industriel (automate, gestion de la génération de courant...), dans le bâtiment (ascenseur, porte automatique, air conditionné...), l'agriculture (machine de traite, gestion de l'alimentation des bêtes...), le domaine médical, la communication, le domaine financier(caisse enregistreuse, ATM...), le domaine du spectacle (rampe d'éclairage,robot), le domaine scientifique (équipement de laboratoire, télescope).

Caractéristiques physique du bus CAN



conservation de la différence de potentiel quand le réseau est soumis à une perturbation électromagnétique

ISO11898
High Speed CAN 125Kbps - 1Mb/s

Longueur (m)	30	50	100	250	500	1000	2500	50000
Vitesse (kb/s)	1000	800	500	250	125	62,5	20	10

Les nœuds sont câblés sur le bus de façon à effectué des opérations logiques de type "ET", ce qui se correspond en cas d'émission simultanée sur le bus que la valeur 0 écrasera la valeur 1, ce qui se donne dans la terminologie CAN :
- l'état logique 0 est l'état dominant
- l'état logique 1 est l'état récessif

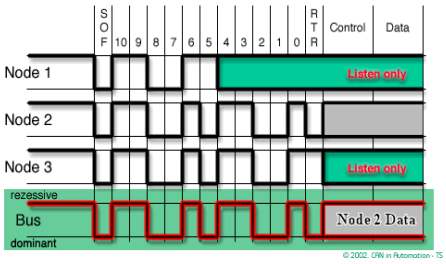
Principe du protocole

Composition d'une trame CAN

Nom du champ		Taille (bit)	
SOF		1	Indique le début d'une trame, dominant (0)
Champ d'arbitrage	Identifiant	11 | 29	Identifiant de message unique, permet de déterminer la priorité du message
	RTR	1	Doit être dominant (0) pour les trames de données et récessif (1) pour les trames de requêtes.
Champ de contrôle	Identifiant d'extension	1	Doit être dominant (0) pour les trames standard et récessives (1) pour les trames étendues.
	Champ réservé	1	bit réservé, non utilisé, doit être dominant (0)
	DLC	4	(Data Lenght Code) indique le nombre d'octets dans le champ data (0-8 octets)
Champ de données		0-64	Les données qui doivent être transmises.
Champs CRC	CRC	15	Contrôle de Redondance cyclique
	Délimiteur du CRC	1	Le bit de délimitation qui est toujours récessif (1)
Champ ACK	ACK	1	Le bit d'acquittement est toujours récessif (1) pour l'émetteur
	ACK délimiteur	1	Le bit de délimitation de ACK, récessif(1)
EOF		7	Indique la fin de la trame, tous les bits sont récessifs (1)

Principe du protocole

Arbitrage accès au bus

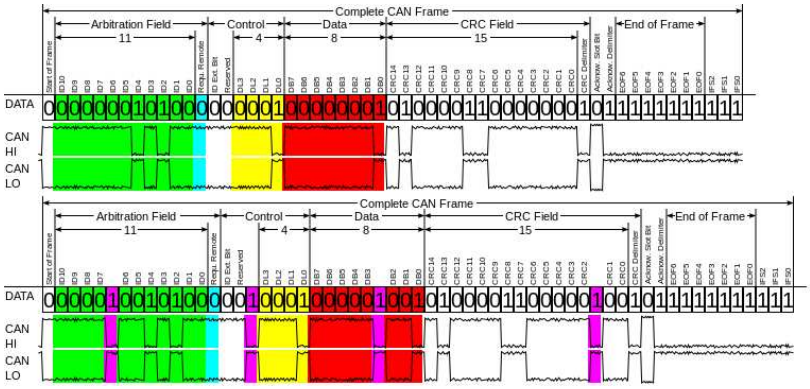


	Start Bit	Identificateur										RTR	Champ Contrôle	Données
		10	9	8	7	6	5	4	3	2	1	0			
Station 1	0	1	1	0	0	1	1	Écoute du bus							
Station 2	0	1	1	0	0	1	0	1	1	0	0	1	0	X	X
Station 3	0	1	1	0	0	1	0	1	1	0	0	1	1	Écoute du bus	
Signal sur le Bus	0	1	1	0	0	1	0	1	1	0	0	1	0	X	X

Le champ d'identification permet donc en mode standard de codé 2^{11} soit 2048 combinaisons de messages différents, et en mode étendu 2^{29} soit 536 870 912 combinaisons, le mode étendu est donc plus utile sur un réseau composer d'un grand nombre de nœuds, alors que les trames standard seront suffisantes pour de petits réseaux.

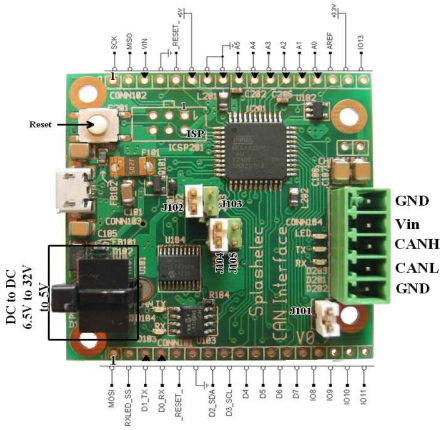
Principe du protocole

Ajout de bit



La détection d'erreur se fait au niveau binaire (la couche 1 pour le modèle OSI) via 2 principes :
- La surveillance qui consiste au suivi par chaque station du bus des données circulant sur le bus, l'émetteur observe aussi le bus ce qui lui permet de détecter les différences entre les bits envoyer et ceux reçus et lui permet de déterminer si l'erreur est locale ou globale.
- L'ajout de bit, les bits envoyer par CAN utilise la méthode NRZ, soit pas de retour a zéros entre les bits, pour permettre une synchronisation, après l'envoi de 5 bits identique consécutif, l'émetteur ajoute un bit de remplissage dans le flot binaire. Ce bit de remplissage est complémentaire de bit précédent, ce bit est ensuite supprimé par les receveurs.

Présentation de la carte



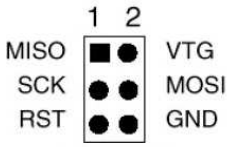
La carte Arduino utilisée

La carte de base pour chaque nœud du réseau composer d'un microcontroleur identique a une carte Leonardo et de Puce CAN MCP2515 et MCP2551 que l'on retrouve sur les shield CAN

Programmation de la carte

Programmateur USBtinyISP

Connecteur ICSP

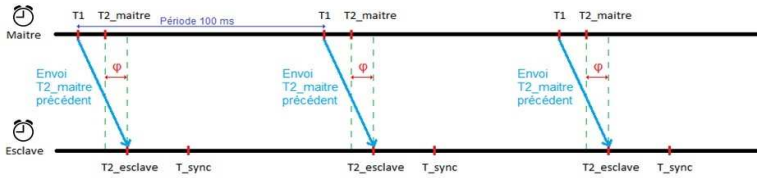


ISP6PIN

Pour reprogrammer la carte on peut utiliser un programmateur dédier, par exemple celui utiliser lors du projet, le programmateur USBtinyISP, qu'il faut sélectionner dans la liste des programmateurs dans le logiciel Arduino.
Le programmateur ce connect au port ICSP de la carte arduino

Protocole SimpleCAN

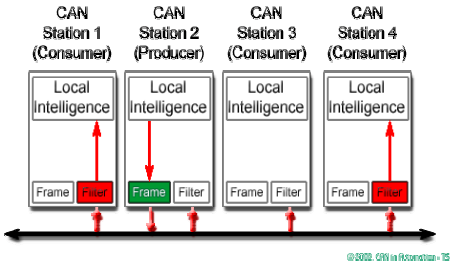
- **Protocole de haut niveaux**
  - Synchronisation du bus
  - Élection d'un leader (maître)



Cela évite qu'une carte avec un programme moins complexe n'émette plus souvent qu'une autre carte du réseau avec un programme plus complexe.
Le nœud maître synchronise toutes les cartes toutes les 100ms pour éviter une dérive des timers entres les cartes.
Le nœud avec l'identifiant le plus faible du réseau est désigner leader du réseau, pour ce faire chaque nœud du réseau émet sont identifiant sur le réseau et le compare a celui qu'il reçoit, s'il a l'identifiant le plus faible, il détermine qu'il est le maître, sinon il détermine qu'il est un esclave.
Ce système d'élection est très gourmand en taux d'occupation du bus, chaque nœud émettant son identifiant cela peux saturer le bus.

Protocole SimpleCAN

- **Protocole de haut niveaux**
  - Filtrage des messages



Mask Bit n	Filter Bit n	Message Identifier bit	Accept or Reject bit n
0	X	X	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

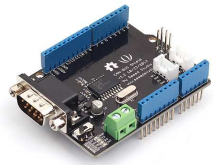
Note: X = don't care

Pour ne pas saturer le microcontrôleur utilisé, on peut utiliser les filtres disponibles sur la puce CAN MCP2515. Les buffers de réception de la puce CAN dispose d'un masque et de 6 filtres qui permettent d'ignorer les messages qui circulent sur le bus s'il ne correspond pas au filtrage appliqué.
SimpleCan apport un support de ces filtres, et permet de déterminer des plages de messages accepter ce qui évite de limiter à 6 messages différents par nœud, ce filtrage s'effectuant sur l'identifiant des messages de la trame CAN.

Présentation de la libraire CAN utilisée

Librairie CAN bus Shield de Seeed-Studio

- Basée sur une puce identique
- Mise à jour récemment
- Fonctions documentées et simple
- Gestion des filtres et des masques



Le shield CAN de la librairie

comporte plusieurs exemples d'utilisation de ses fonctions, ce qui permet une prise en main rapide. utilisant les interruptions du programme Arduino.

Les fonctions disponibles sont :

```
MCP_CAN CAN(SPI_CS_PIN);
CAN.begin(CAN_500KBPS) 5kb/s à 1000kb/s,
 CAN.init_Mask(unsigned char num, unsigned char ext, unsigned char ulData);
CAN.init_Filt(unsigned char num, unsigned char ext, unsigned char ulData);
CAN.checkReceive()
CAN.getCanId()
CAN.sendMsgBuf(INT8U id, INT8U ext, INT8U len, data_buf);
CAN.readMsgBuf(unsigned char *len, unsigned char *buf);
CAN.isRemoteRequest();
CAN.isExtendedFrame();
```

Présentation du NMEA

- Norme NMEA 183
- Utilisé dans les système de navigation
- Système de trame

$GPRMC,225446,A,4916.45,N,12311.12,W,000.5,054.7,191194,020.3,E*68

$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a,m*hh
Field #
1 = UTC time of fix
2 = Data statuts (A=Valid position, V=navigation receiver warning)
3 = Latitude of fix
4 = N or S of longitude
5 = Longitude of fix
6 = E or W of longitude
7 = Speed over ground in knots
8 = Track made good in degrees True
9 = UTC date of fix
10 = Magnetic variation degrees (Easterly var. subtracts from true course)
11 = E or W of magnetic variation
12 = Mode indicator, (A=Autonomous, D=Differential, E=Estimated, N=Data not valid)
13 = Checksum

Le NMEA ou NMEA 183 est une spécification de communication fonder en 1957
Les trames NMEA sont composées de plusieurs champs séparer par des virgules, chaque trame commence par un identifiant dont le premier caractère est toujours $, suivit d'un identifiant du récepteur composer de 2 lettres :
- GP pour le GPS,
- GL pour le système GLONASS équivalent russe du GPS,
- LC pour les récepteurs Loran-C qui est un système de radio navigation,
- OM pour les récepteurs de navigation de type oméga qui est un système de radio navigation par triangulation de la distance des différentes ondes reçues émise par les stations OMEGA dont la position est connue,
- II pour instrument intégrer, par exemple AutoHelm de Seatalk système.

Certain fabricant propriétaire utilise leur propre identifiant en indiquant P pour propriétaire suivit d'un code de 3 lettres pour identifier le fabricant, par exemple Garmin donne $PGRM.
Le reste identifie la trame, pour le GPS, on trouve donc entre autres les trames :
- GGA pour GPS fixe date
- GLL pour la position géographique latitude, longitude
- GSA pour les données de précision
- GSV pour la liste des satellites en vue
- VTG pour le cap par rapport au plan terrestre et la vitesse au sol
- RMC pour les données minimales recommandées de spécification GPS.

La taille d'une trame NMEA ne peut pas dépasser 80 caractères.

Conclusion

- Problème rencontrer
- Projet intéressant

Probleme

Documentation

Probleme de librairie

Problème matériel

Un problème matériel est aussi survenu sur une des cartes, la carte d'interface USB ce qui empêchait tout débogage de la programmation, ce problème matériel était dû à un conflit au niveau du SPI de la carte. En effet la puce CAN le MCP2515 était connecter au bus MISO et MOSI que le programmateur utilise pour reprogrammer la carte, hors le MCP2515 envoyai des données en même temps que le programmateur se qui provoquait des erreurs de transmission.

Manque de temps

Conclusion

Ce projet m'a permis d'améliorer mon autonomie et améliorer mes connaissances sur les bus de terrain CAN et des cartes Arduino que je connaissais déjà un peu.

Ce projet était relativement intéressant par rapport a mon projet d'étude, car il concernait le domaine de l'embarqué et correspondait à des contraintes et un objectif concret de mise en œuvre d'un système dans son ensemble.

Vous avez des questions?