



Cégep Limoilou

ÉLECTRONIQUE PROGRAMMABLE ET ROBOTIQUE

247-6[1-2-3-4]7-LI

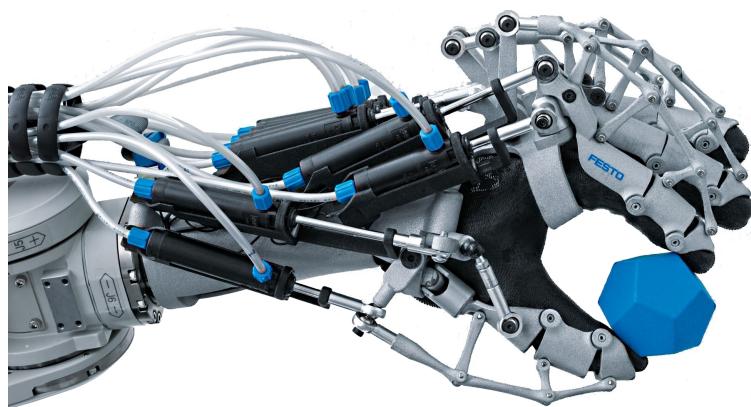
Projet de 5^e session

Étudiants :

Vincent Chouinard
Hicham Safoine
Gabriel Fortin-Bélanger
Louis-Nomand Ang-Houle

Professeurs :

Ali Tadli
Alain Champagne
Stéphane Deschênes
Étienne Tremblay



L'usine à gaz, et le gaz, c'est de l'air !

21 novembre 2014

Table des matières

1 Présentation du projet	5
1.1 Explication du projet	5
1.2 Schéma bloc du système	5
1.2.1 Bloc 1	5
1.2.2 Bloc 2	5
1.2.3 Bloc 3	5
1.2.4 Bloc 4	5
1.3 Liste des logiciels	5
1.4 Liste des trames	7
2 Le matériel	10
2.1 Bloc 1	10
2.2 Bloc 2	10
2.3 Bloc 3	10
2.4 Bloc 4	10
2.5 Explication des types de liens	10
2.5.1 RS232	10
2.5.2 Xbee	10
2.6 Explication des trames	10
2.6.1 RS-232	10
2.6.2 CAN	10
2.6.3 XBEE	10
2.7 Liste des pièces	10
2.7.1 Liens web	11
2.7.2 Datasheets	11
3 Interface PC	13
3.1 Gestion de l'historique	15
3.1.1 Exemple d'historique typique	15
3.2 Structure du programme	15
3.2.1 Les Ghosts Labels	15
3.3 Explication des trames	15
3.4 Ordre de gestion des tâches	15
4 Logiciel du SOC8200	16
4.1 Description du programme	16
4.2 Schéma bloc	16
4.2.1 Du code	16
4.2.2 Du script shell	16
4.3 Gestion des processus et du temps de CPU	16
4.4 Format et récupération des logs	16
4.5 Liste des tests et logiciels	16
5 Logiciel de la station 1 et 2 et du bolide	17
5.1 La station no.1	17
5.2 La station no.2	17
5.3 Le bolide	17
5.4 Procédure de compilation sur IAR	18
5.5 Procédure de vérification	18

6 Logiciel du module PIC18F258	18
6.1 Description du fonctionnement du programme	18
6.2 Procédure de compilation sur MPLAB	18
6.3 Procédure de vérification	18
7 Calculs	19
7.1 Calcul du pas de conversion de la pile	19
7.2 Calcul du baudrate	19
8 Conclusion	21
8.1 Ce que le projet m'a apporté	21
8.1.1 Vincent Chouinard	21
8.1.2 Hicham Safoine	21
8.1.3 Gabriel Fortin-Bélanger	21
8.1.4 Louis-Norman Ang-Houle	21
8.2 Difficultés et corrections	21
8.2.1 Vincent Chouinard	21
8.2.2 Hicham Safoine	21
8.2.3 Gabriel Fortin-Bélanger	21
8.2.4 Louis-Norman Ang-Houle	21
8.3 Ce que j'ai aimé ou pas	21
8.3.1 Vincent Chouinard	21
8.3.2 Hicham Safoine	21
8.3.3 Gabriel Fortin-Bélanger	21
8.3.4 Louis-Norman Ang-Houle	21
9 ANNEXE 1 : Code source du Bolide et de la station no.1	22
10 ANNEXE 2 : Code source du programme pour PC	23
11 ANNEXE 6 : Code source du programme PIC	24
12 ANNEXE 3 : Code source du programme du SOC8200	25
13 ANNEXE 5 : Code source de la table FESTO	26

Table des figures

1 Programme de contrôle principal	13
2 Options CAN avancées	14
3 Historique des actions	15
4 Choix de la cible sur IAR	18

Liste des tableaux

1 Index des identifiants matériel CAN	7
2 Index des trames CAN	7
3 Index des communications CAN	7
4 Informations sur le bus I2C du bolide	17

1 Présentation du projet

Le projet de la cinquième session consiste à réaliser

- ⇒ Le Bolide
- ⇒ Carte Dallas DS89C450
- ⇒ Carte uPSD 3254A
- ⇒ SOC8200
- ⇒ Table FESTO
- ⇒ Carte PIC16F88
- ⇒ Carte d'extension I₂C
- ⇒ Carte d'extension SPI
- ⇒ Une pile de 10.8 volts
- ⇒ Quatre moteurs et autant de pneus

1.1 Explication du projet

1.2 Schéma bloc du système

1.2.1 Bloc 1

Le bloc 1 est composé d'un ordinateur muni d'une carte d'extension PCI vers bus CAN. Son rôle est de contrôler et diriger toute l'opération et de veiller au bon fonctionnement de chaque composante à l'aide d'une application en Csharp. Le bloc 1 est le cerveau de l'usine.

1.2.2 Bloc 2

Le bloc 2 est composé d'un système embarqué Linux basé sur le SOC8200. Son rôle principal est d'agir comme sniffeur d'information et d'afficher sur son écran toutes les données qui transitent sur le bus CAN. Toutefois, ce dernier est en mesure de détecter une défaillance du PC via un gestionnaire de HeartBeat et de prendre la relève en tant que cerveau de l'opération. Le SOC8200 agit comme vice-président du bus CAN.

1.2.3 Bloc 3

1.2.4 Bloc 4

1.3 Liste des logiciels

Terminaux

- UART Master 1.0.3
- Serializ3r 1.0.2
- TerraTerm
- Putty
- GTKterm 0.99.7-rc1

Gestion du projet

- xTerminator
- CAPS
- tinyBootloader
- MS Project 2012
- Git Hub

Compilateurs et IDE

- Visual Studio 2013
- Visual Studio 2010
- IAR 8.20
- MPLAB

Éditeur de texte

- Notepad++
 - gedit
 - medit 1.2.0
 - Schémas électriques**
 - OrCAD 16.2
 - Système d'exploitation**
 - Windows 7 SP1
 - Windows 8.1
 - Windows XP SP3
 - Fedora 20
 - CentOS
 - Lubuntu 14.10
 - Autres**
 - VMWare Workstation 10
 - TeXmaker 4.3
 - Dukto R6
 - Dia
 - Festo configuration tool
-

1.4 Liste des trames

TABLE 1 – Index des identifiants matériel CAN

Device	ID matériel
Ordinateur	000
SOC8200	001
Station 1	002
Station 2	003
Station 3	004
Véhicule	005

TABLE 2 – Index des trames CAN

Fonctionnalité	Composante	Données	TimeStamp
Démarre le véhicule	0x00	0x00	TimeStamp
Arrête le véhicule	0x00	0x01	TimeStamp
Le véhicule est arrêté	0x01	0x00	TimeStamp
Le véhicule est en marche	0x01	0x01	TimeStamp
Le véhicule est hors circuit	0x01	0x02	TimeStamp
Vitesse (0-100)	0x02	0x00 à 0x64	TimeStamp
Battre	0x03	0x00 à 0x64	TimeStamp
Couleur du bloc	0x04	0x00 à 0x02	TimeStamp
Poids du bloc	0x05	0x00 à 0x64	TimeStamp
Envoyer l'heure	0x06	à déterminer	TimeStamp
No. de la station	0x07	0x00 à 0x02	TimeStamp
Demande de l'historique	0xC0	0x00	TimeStamp
Direction horaire et antihoraire	0x08	0x00 à 0x01	TimeStamp

TABLE 3 – Index des communications CAN

Émetteur	Action	ID receveur	Donnée envoyée	TimeStamp	Récepteur	Erreur
Ordinateur	Démarrer le véhicule	004	00 00	TimeStamp	Véhicule	F1
Ordinateur	Arrêter le véhicule	004	00 01	TimeStamp	Véhicule	F2
Véhicule	Dit : je suis arrêté	000	01 00	TimeStamp	Ordinateur	F3
Véhicule	Dit : j'avance	000	01 01	TimeStamp	Ordinateur	F4
Véhicule	Dit : je suis hors circuit	000	01 02	TimeStamp	Ordinateur	F5
Véhicule	Dit sa vitesse	000	02 [00 à 64]	TimeStamp	Ordinateur	F6
Véhicule	Dit le niveau de sa batterie	000	03 [00 à 64]	TimeStamp	Ordinateur	F7
Station 1	Dit bloc = métal	000	04 00	TimeStamp	Ordinateur	F8
Station 1	Dit bloc = orange	000	04 01	TimeStamp	Ordinateur	F9
Station 1	Dit bloc = noir	000	04 02	TimeStamp	Ordinateur	FA
Station 1	Dit le poid du bloc	000	05 [00 à 64]	TimeStamp	Ordinateur	FB
Voiture	Dit qu'elle est à la station 1	000	07 00	TimeStamp	Ordinateur	FC
Voiture	Dit qu'elle est à la station 2	000	07 01	TimeStamp	Ordinateur	FD
Ordinateur	Envoie l'heure	003	06 à déterminer	TimeStamp	Station 1	FE
Ordinateur	Demande le LOG	001	C0 00	TimeStamp	SOC8200	E0
Ordinateur	Exige Horaire	004	08 00	TimeStamp	Véhicule	E1
Ordinateur	Exige Antihoraire	004	08 01	TimeStamp	Véhicule	E2

Note : Il faut définir les TimeStamps et la checkSUM

Note : La station no.1 relaie les données entre l'ordinateur et la station no.3 (pesage), entre l'ordinateur et la station no.2 (table Festo) via xbee et entre la voiture et le PC via Xbee.

Note : FF, c'est la checkSUM, mais elle n'a pas encore été faite

Note : Il faut ajouter le TimeStamp

```
CAN.SendToPC( "0100FF" ); // Arrêté
CAN.SendToPC( "0101FF" ); // En marche
CAN.SendToPC( "0102FF" ); // Hors circuit
CAN.SendToPC( "02xxFF" ); // Vitesse de xx
CAN.SendToPC( "03xxFF" ); // Batterie chargée à xx %
CAN.SendToPC( "0400FF" ); // Bloc métallique
CAN.SendToPC( "0401FF" ); // Bloc noire
CAN.SendToPC( "0402FF" ); // Bloc orange
CAN.SendToPC( "050064" ); // Le bloc est lourd
CAN.SendToPC( "0700FF" ); // Rendu à la station 1
CAN.SendToPC( "0701FF" ); // Rendu à la station 2
CAN.SendToPC( "0702FF" ); // Rendu à la station 3
```

f

2 Le matériel

2.1 Bloc 1

D'un point de vu matériel, le bloc 1 est le plus simple, car il est composé d'un ordinateur Windows munie d'une carte PCI vers CAN et d'une prise RS232. Il n'y a rien à faire, mise à part brancher les bons câbles aux bons endroits.

2.2 Bloc 2

2.3 Bloc 3

2.4 Bloc 4

2.5 Explication des types de liens

2.5.1 RS232

Un lien RS232 9600 Bauds est établi entre l'ordinateur et le SOC8200. Ce lien sert à l'envoie et à la réception de HeartBeat, afin que le SOC8200 ou l'ordinateur soit informé de toute défaillance de l'autre.

2.5.2 Xbee

Lorsque les modules Xbee sont adéquatement configurés, ils font office de remplacement au câble RS232. En effet, nos Xbee discutent entre eux à l'aide du protocole de communication RS232 à 9600 bauds.

2.6 Explication des trames

2.6.1 RS-232

Le protocole RS-232 sert à envoyer et à recevoir des HeartBeat. Le PC et le SOC 8200 s'envoient tous deux un HeartBeat par seconde à 9600 bauds. Un HeartBeat, c'est simplement le mot "Allo". Le PC et le SOC2800 "écoutent" les HeartBeats, et si ces derniers ne sont pas entendus, chaque dispositif prend pour acquis que l'autre est hors-service et prend la relève de la gestion du bus CAN.

2.6.2 CAN

2.6.3 XBEE

Trois modules Xbee sont présent sur l'ensemble du projet, soit sur la station no.1 (la carte μ PSD), sur la station no.2 (la table FESTO) et la station no.6, c'est à dire le bolide. La particularité des Xbee est que lorsqu'ils sont adéquatement configurés, tout ce qu'envoie un Xbee est reçu et lu par tous les autres Xbee à proximité, et c'est pourquoi nous avons défini un système de trames.

2.7 Liste des pièces

- Carte Dallas
- Carte uPSD
- SOC 8200
- PIC18Fmachin
- Carte d'extension SPI
- Carte d'extension I2C
- Carte CAN MCP2515
- XBEE
- Table FESTO
- Carte d'extension IO
- Carte connecteur DAC ADC
- Carte Xbee vers DB9
- •

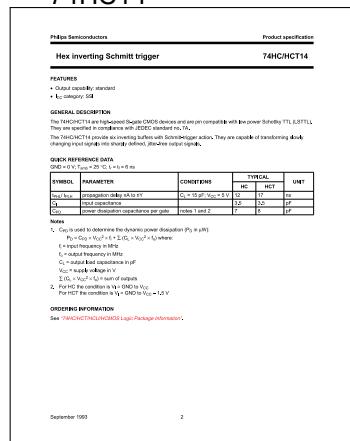
2.7.1 Liens web

Mettre ien vers GITHUB

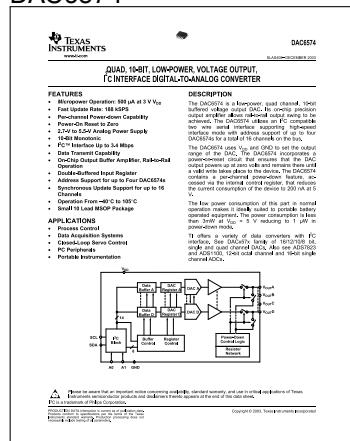
2.7.2 Datasheets

Note : Toutes les datasheets sont en format non-compressé. Vous pouvez zoomer sur le document PDF¹ afin de lire l'intégralité de leurs première page.

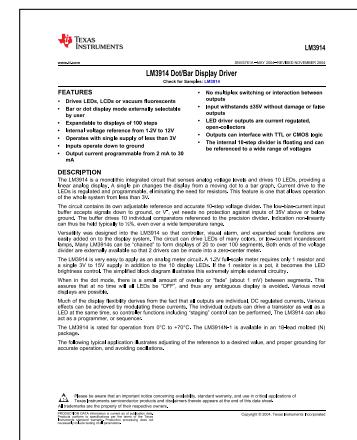
74HC14



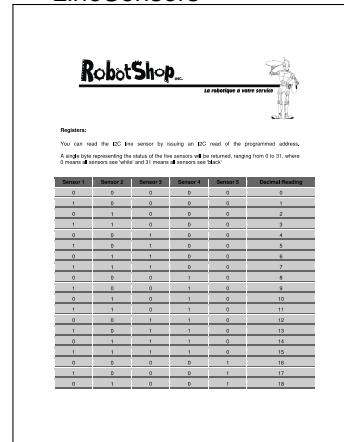
DAC6574



DS89C450



LineSensors



LM3914

PCF8573

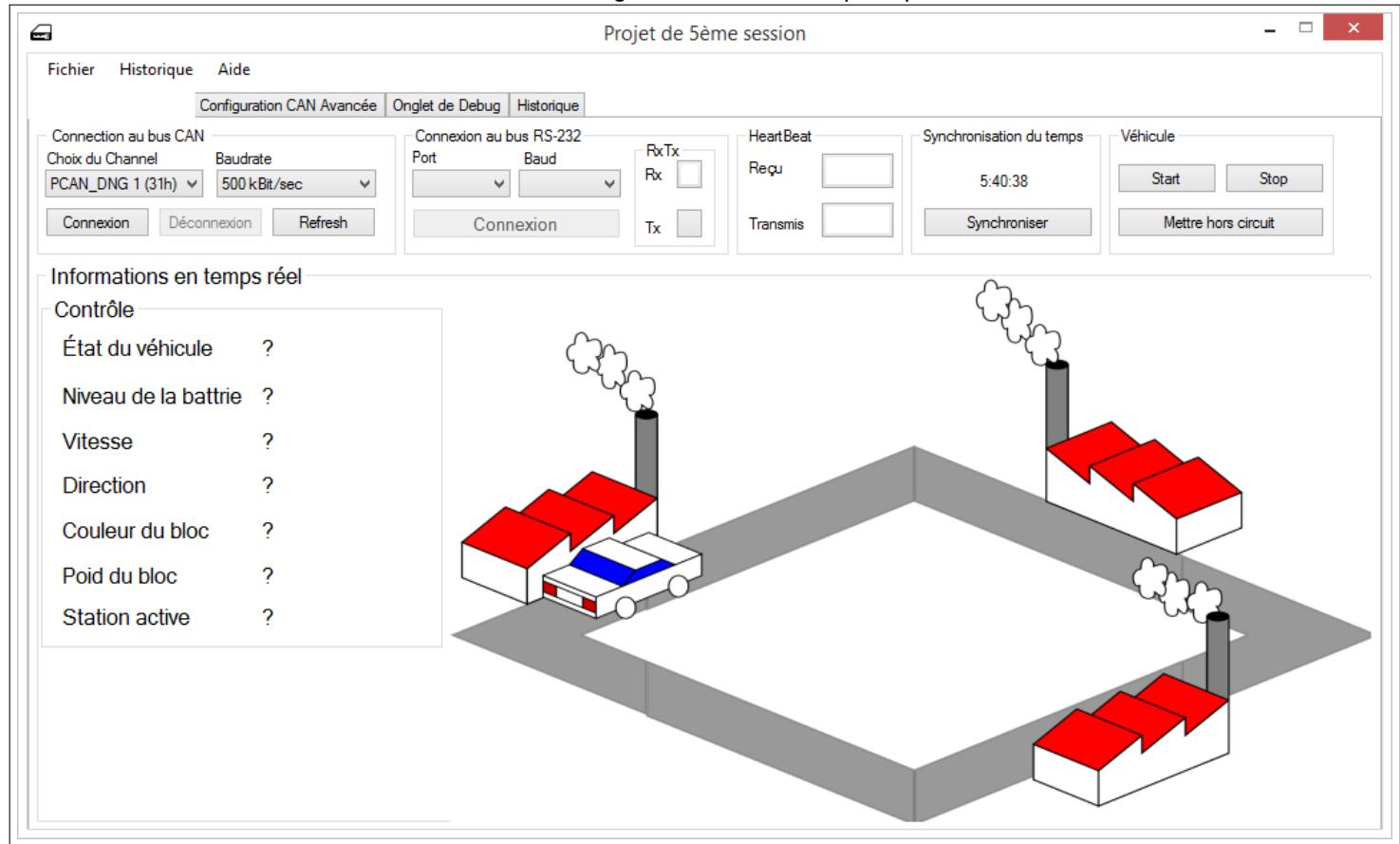
- Présenter les datasheets de la sorte économise du papier, donc des arbres, mais requiert de consulter le document .PDF afin de lire adéquatement les datasheets.

<p>REMOVAL & FIT TO EXPAND FOR PCB BUS</p> <p>LOW PROFILE LOW COST LOW CURRENT CONSUMPTION PCB TO PCB INTERCONNECT SMD SMT COMPATIBLE WITH MOST MICROCONTROLLERS</p> <p>ON BOARD PACKAGE (TOP VIEW)</p> <p>POST PACKAGE (TOP VIEW)</p> <p>DOOR OR PCB PACKAGE (TOP VIEW)</p> <p>NO = not required or not connected</p> <p>DESCRIPTION/ORDERING INFORMATION</p> <p>This document reproduced (PCB) experience for the most advanced IC's designed to ZIF-DOJ connectors.</p> <p>The PCPZ574A provides pin-to-pin compatibility with PCPZ574 for maximum compatibility. Refer to the PCZ technical note for more information.</p> <p>The device features an 8-bit quad serial port (QSPI) [P0/P1/P2], including serial inputs with high-speed data capability by direct driving. Each quad serial port can be used as serial input or output. The serial port is fully compatible with SPI, QSPI, QIPI, and QSPI-4. The serial port has a bidirectional serial bus YZQ to enable additional memory or ROM to be easily added to the system. It also has a bidirectional serial bus XZQ to enable additional memory or ROM to be easily added to the system. This device has a high level of compatibility with other TI products.</p> <p>ORDERING INFORMATION</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Part No.</th> <th>Package</th> <th>Orderable</th> <th>Orderable</th> </tr> </thead> <tbody> <tr> <td>PCPZ574A</td> <td>SMD SMT</td> <td>YES</td> <td>YES</td> </tr> <tr> <td>PCPZ574A</td> <td>PCB</td> <td>YES</td> <td>YES</td> </tr> <tr> <td>PCPZ574A</td> <td>PCB</td> <td>NO</td> <td>NO</td> </tr> <tr> <td>PCPZ574A</td> <td>PCB</td> <td>NO</td> <td>NO</td> </tr> </tbody> </table> <p>IMPORTANT: To ensure that an application meets environmental, standard, and use-in-condition requirements of the Texas Instruments product specification, it must be tested and approved by Texas Instruments.</p> <p>DISCLAIMER: The information contained in this document is subject to change without notice. Texas Instruments reserves the right to make changes at any time, without notice, which may affect the performance or reliability of the products described. Texas Instruments does not assume any responsibility or liability for any errors or inaccuracies that may appear in this document.</p> <p>TRADE SHOWS: Texas Instruments is participating in the following trade shows: TI booth #1000, SEMICON West, San Jose, CA, July 15-19, 2001 TI booth #1000, DesignCon, San Jose, CA, January 29-February 2, 2002</p> <p>TEXAS INSTRUMENTS www.ti.com www.ti.com/tisolutions</p> <p>Copyright © 2001, Texas Instruments Incorporated</p>	Part No.	Package	Orderable	Orderable	PCPZ574A	SMD SMT	YES	YES	PCPZ574A	PCB	YES	YES	PCPZ574A	PCB	NO	NO	PCPZ574A	PCB	NO	NO
Part No.	Package	Orderable	Orderable																	
PCPZ574A	SMD SMT	YES	YES																	
PCPZ574A	PCB	YES	YES																	
PCPZ574A	PCB	NO	NO																	
PCPZ574A	PCB	NO	NO																	

SaberTooth motor drive

3 Interface PC

FIGURE 1 – Programme de contrôle principal

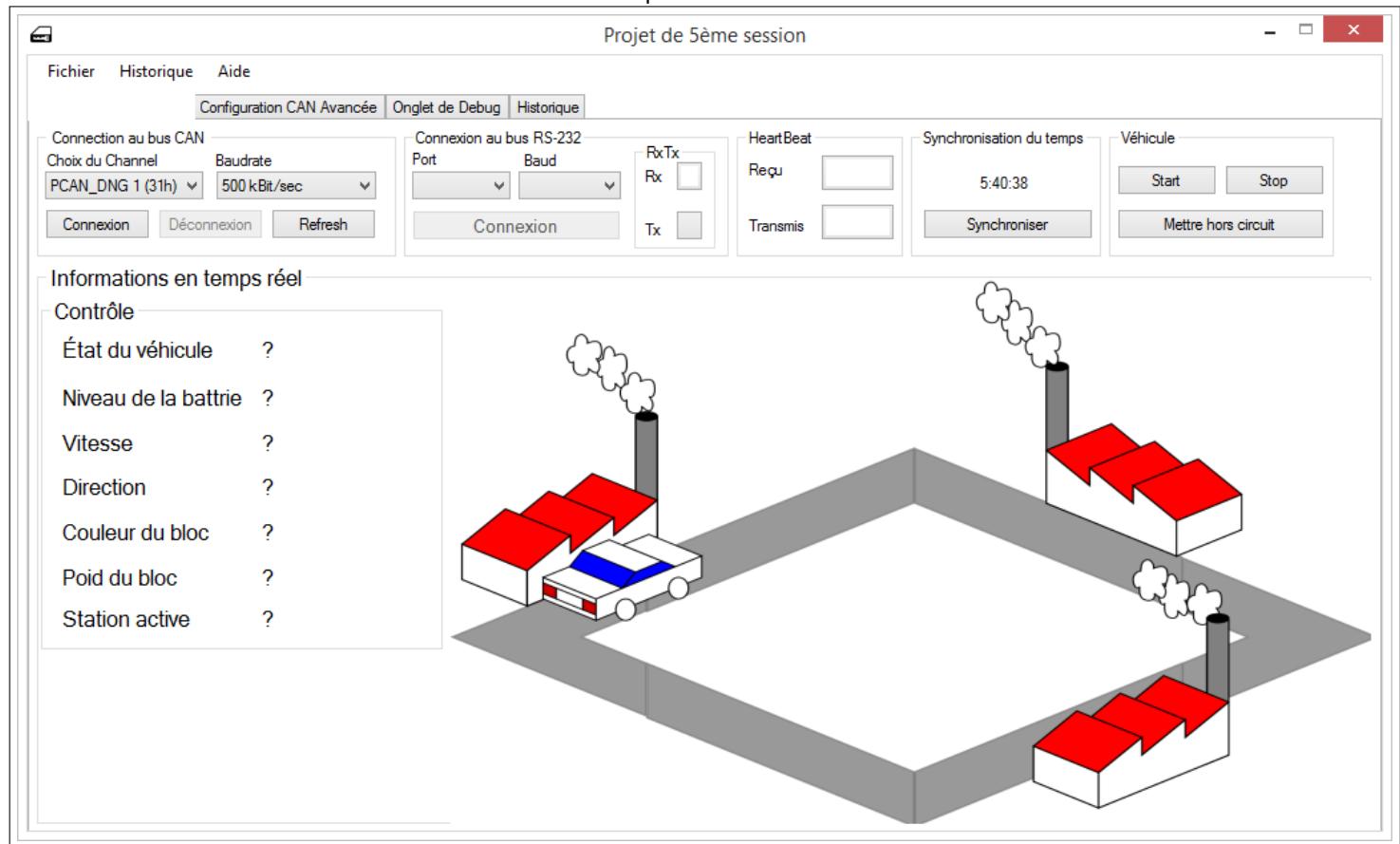


Notre programme, écrit en C# à l'aide de Visual Studio, peut se connecter au bus CAN via une carte SPI² et au bus RS232 via un câble DB9 ou USB³. La connexion RS232 sert à l'envoie et à la réception du HeartBeat afin d'informer le SOC8200 si l'ordinateur en venait à connaître une défaillance. De plus, des témoins lumineux s'allument en présence de données transmises et reçues.

Le programme peut lire l'heure interne du PC et, par un simple clic sur le bouton «Synchroniser», ajuster l'heure de référence de la station no.1.

2. Spécifier le fabricant
3. S'il y a présence d'un FTDI

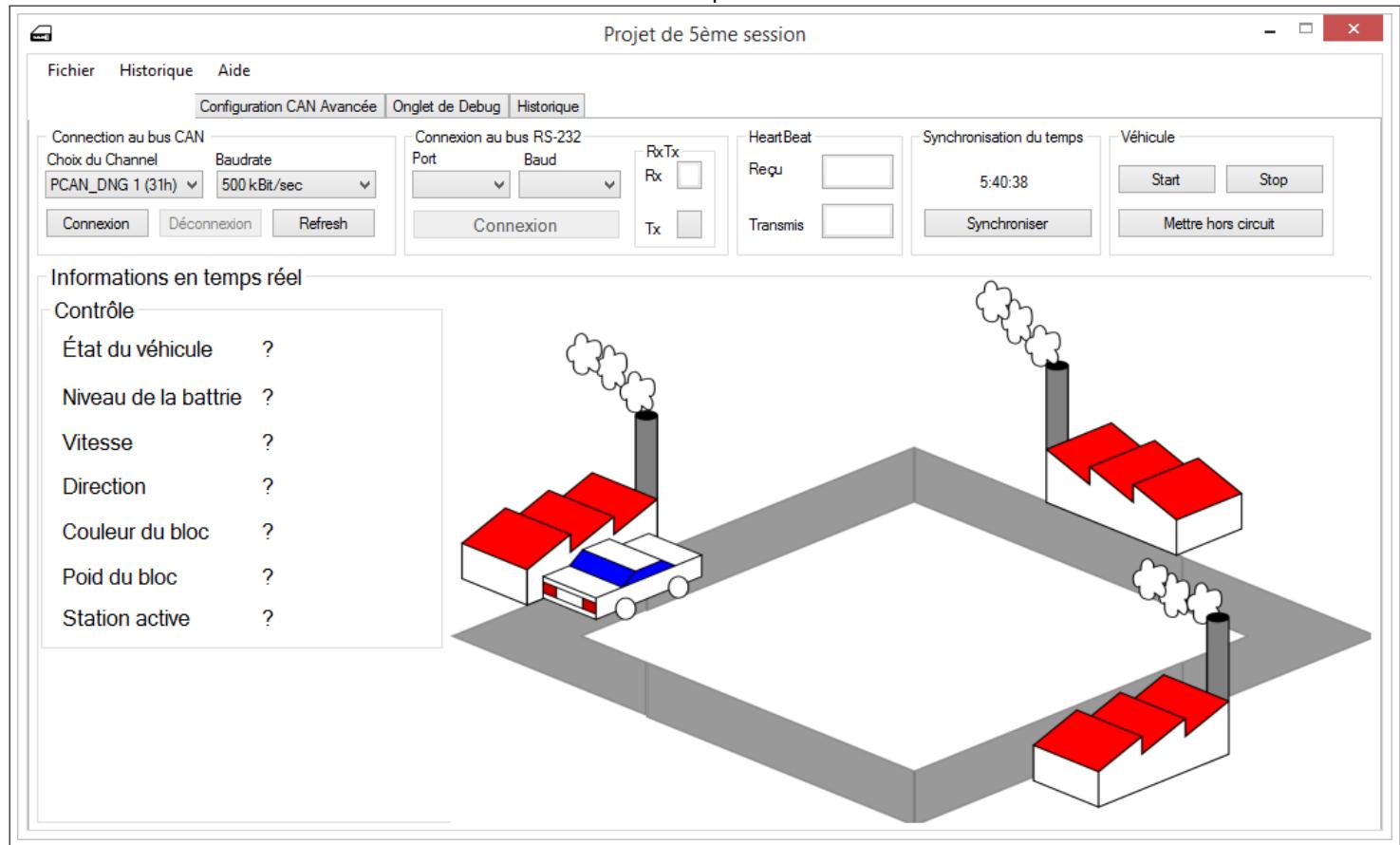
FIGURE 2 – Options CAN avancées



Il est possible d'utiliser des fonctionnalités CAN avancées tels que les masques et filtres de données. De plus, cette fenêtre permet de visualiser les données CAN reçues à l'état brutes et non traitées, ce qui peut s'avérer utile pour du débogage.

3.1 Gestion de l'historique

FIGURE 3 – Historique des actions



Toute action effectué via le programme ainsi que toute donnée ayant transité sur le bus CAN et RS232 est catalogué en bonne et due forme dans l'onglet historique qu'il est possible de consulter et sauvegarder à tout moment.

3.1.1 Exemple d'historique typique

wefsd

3.2 Structure du programme

3.2.1 Les Ghosts Labels

Un ghost label est un label de texte présent sur l'interface, mais définit comme invisible. Il est donc impossible pour l'usager de le voir et d'y accéder. Leurs principales utilités est de faire office de variable globale afin de passer des paramètres entre fonctions et de déclencher des événements système lorsqu'ils sont lus ou modifiés.

3.3 Explication des trames

3.4 Ordre de gestion des tâches

4 Logiciel du SOC8200

4.1 Description du programme

Le programme du SOC2800 est écrit en script Shell⁴ ⁵

4.2 Schéma bloc

4.2.1 Du code

4.2.2 Du script shell

4.3 Gestion des processus et du temps de CPU

4.4 Format et récupération des logs

4.5 Liste des tests et logiciels

4. Car c'est plus simple que de se battre avec le gestionnaire de licence de Sourcery Codebench

5. L'un des membres de l'équipe fait dire que les licences sont une horreur inacceptable sur un système Linux

5 Logiciel de la station 1 et 2 et du bolide

Le programme de la station 1, 2 et du bolide est écrit en C++ à l'aide d'IAR WorkBench 8.20 et la compilation conditionnelle offre de le compiler pour chacune des trois stations mentionnées. De plus, la compilation conditionnelle permet au bolide d'utiliser soit une carte d'extension I2C, soit une carte d'extension SPI pour contrôler ses moteurs.

5.1 La station no.1

La station no.1 est composé de μ PSD et s'appelle Bloc no.2 dans le cahier de consignes. Cette station reçoit les directives du PC par le BUS CAN et les expédie sur le bus CAN (et vice-versa) aux endroits appropriés. C'est aussi à cette station qu'incombe la tâche de communiquer avec le bolide et la table FESTO via des Xbee

5.2 La station no.2

La station no.2 est composé de la table FESTO et s'appelle Bloc no.3 dans le cahier de consignes

5.3 Le bolide

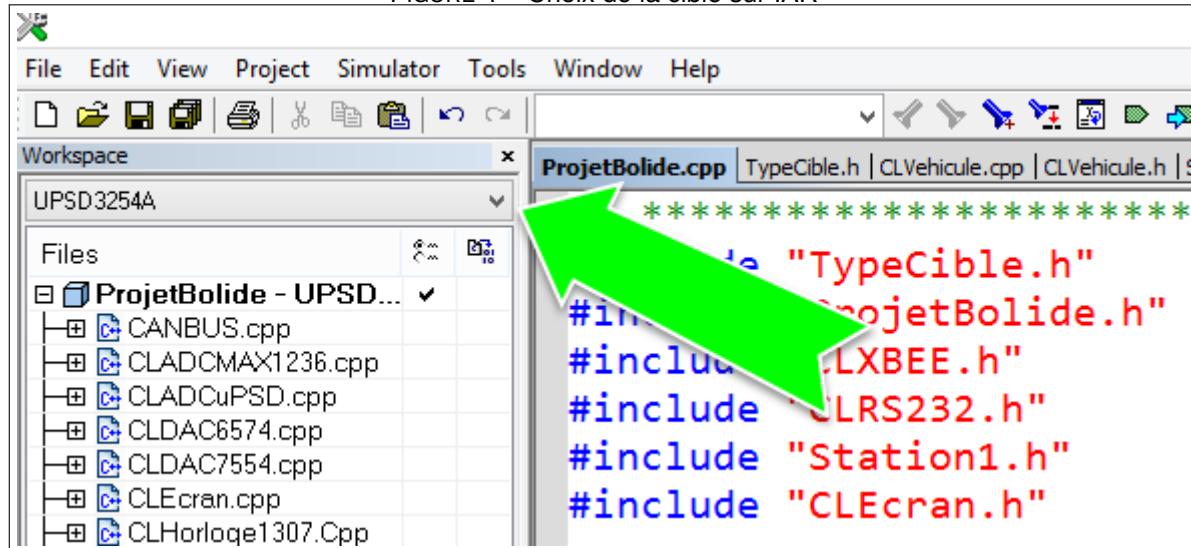
Composante	Adresse I2C	Description
MAX1236	0x68	Convertisseur analogique-numérique
DS1307	0xD0	Circuit d'horloge RTC
PCF8574	0x40	I/O Expander pour bus I2C
DAC6574	0x98	Convertisseur numérique-analogique
OPT101	0x50	Suiveur de ligne

TABLE 4 – Informations sur le bus I2C du bolide

5.4 Procédure de compilation sur IAR

Sur IAR, vous pouvez utiliser le menu déroulant, illustré à la figure suivante, afin de compiler le code pour la carte Dallas ou pour la carte μ PSD.

FIGURE 4 – Choix de la cible sur IAR



De plus, des paramètres de compilation optionnelle vous permettent, via la décommentation, de compiler le code pour la carte Dallas ou μ PSD, pour la carte d'extension I2C ou SPI et pour un capteur de ligne à 3 ou à 5 photorécepteurs.

Appercu des directives de compilation conditionnelles

```
///#define UPSD3254A  
///#define DALLAS89C450  
///#define SPI.DALLAS  
///#define I2C.DALLAS  
///#define PCF.5.CAPTEURS  
///#define PCF.3.CAPTEURS
```

5.5 Procédure de vérification

6 Logiciel du module PIC18F258

6.1 Description du fonctionnement du programme

6.2 Procédure de compilation sur MPLAB

6.3 Procédure de vérification

7 Calculs

7.1 Calcul du pas de conversion de la pile

$$V_{MAX} = 10.8V \Rightarrow MAX1236 = 12bit \Rightarrow Pas = \frac{4K\Omega \cdot \left(\frac{10.8V}{10K\Omega} \right)}{2^{12}(pas)} = 1.33(mV/pas)$$

7.2 Calcul du baudrate

$$Baud = \frac{2^{SMOD}}{32} \cdot \frac{Crystal(Hz)}{12 \cdot (256 - TH1)}$$

Alors...

$$9600 = \overbrace{\frac{2^1}{32} \cdot \frac{24 \cdot 10^6(Hz)}{12 \cdot (256 - 243)}}^{uPSD3254} \Leftarrow \& \Rightarrow 9600 = \overbrace{\frac{2^0}{32} \cdot \frac{11.0597 \cdot 10^6(Hz)}{12 \cdot (256 - 253)}}^{DS89C450}$$

8 Conclusion

8.1 Ce que le projet m'a apporté

8.1.1 Vincent Chouinard

8.1.2 Hicham Safoine

8.1.3 Gabriel Fortin-Bélanger

8.1.4 Louis-Norman Ang-Houle

8.2 Difficultés et corrections

8.2.1 Vincent Chouinard

8.2.2 Hicham Safoine

8.2.3 Gabriel Fortin-Bélanger

8.2.4 Louis-Norman Ang-Houle

8.3 Ce que j'ai aimé ou pas

8.3.1 Vincent Chouinard

8.3.2 Hicham Safoine

8.3.3 Gabriel Fortin-Bélanger

8.3.4 Louis-Norman Ang-Houle

9 ANNEXE 1 : Code source du Bolide et de la station no.1

```
using System;
```

10 ANNEXE 2 : Code source du programme pour PC

```
using System;
```

11 ANNEXE 6 : Code source du programme PIC

```
void main( void )  
{  
}
```

12 ANNEXE 3 : Code source du programme du SOC8200

```
using System;
```

13 ANNEXE 5 : Code source de la table FESTO

```
using System;
```