



Cégep Limoilou

ÉLECTRONIQUE PROGRAMMABLE ET ROBOTIQUE

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

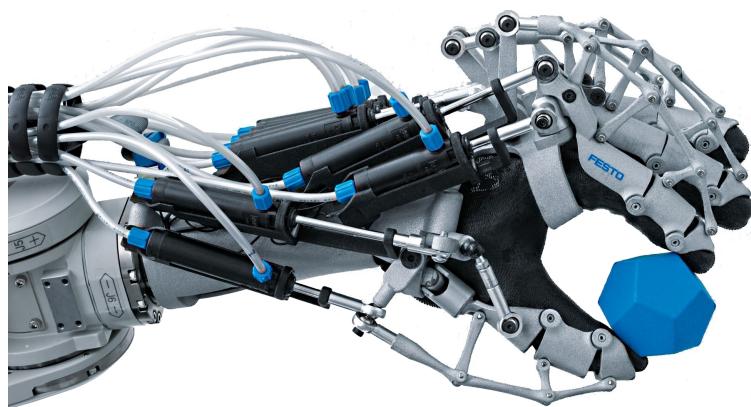
Projet de 5^e session

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 !

4 décembre 2014

Table des matières

1 Présentation du projet	5
1.1 Explication du projet	5
1.2 Schéma bloc vue d'ensemble du système	5
1.2.1 Schéma bloc du bolide	6
1.2.4 Schéma bloc de la station de pesée	7
1.2.2 Schéma bloc de la table FESTO	8
1.2.3 Schéma bloc du SOC8200	9
1.2.5 de la station no.1	10
1.3 Liste des logiciels	10
1.4 Liste des trames	11
2 Le matériel	12
2.1 Bloc 1	12
2.2 Bloc 2	12
2.3 Bloc 3	12
2.4 Bloc 4	12
2.5 Explication des types de liens	12
2.5.1 RS232	12
2.5.2 Xbee	12
2.6 Explication des trames	13
2.6.1 RS-232	13
2.6.2 CAN	14
2.6.3 XBEE	16
2.7 Liste des pièces	16
2.7.1 Liens web	16
2.7.2 Datasheets	17
3 Interface PC	19
3.1 Gestion de l'historique	21
3.1.1 Exemple d'historique typique	21
3.2 Structure du programme	21
3.2.1 Les Ghosts Labels	21
3.3 Explication des trames	21
3.4 Ordre de gestion des tâches	21
4 Logiciel du SOC8200	22
4.1 Description du programme	22
4.2 Schéma bloc du script shell	22
4.3 Gestion des processus et du temps de CPU	22
4.4 Format et récupération des logs	22
4.5 Liste des tests et logiciels	22
5 Logiciel de la station 1 et du bolide	23
5.1 La station no.1	23
5.2 La station no.2	23
5.4 Le bolide	24
5.3 Schéma des héritages de classes	25
5.5 Procédure de compilation sur IAR	26
5.6 Procédure de vérification	26

6 Logiciel du module PIC18F258	27
6.1 Description du fonctionnement du programme	27
6.2 Procédure de compilation sur MPLAB	27
6.3 Procédure de vérification	27
7 Calculs	28
7.1 Calcul du pas de conversion de la pile	28
7.2 Calcul du baudrate	28
8 Schémas OrCAD	29
9 Fichiers Gerbers	33
10 Conclusions	36
10.1 Ce que le projet m'a apporté	36
10.1.1 Vincent	36
10.1.2 Hicham	36
10.1.3 Gabriel	36
10.1.4 Louis-Norman	36
10.2 Difficultés et corrections	36
10.2.1 Vincent	36
10.2.2 Hicham	36
10.2.3 Gabriel	36
10.2.4 Louis-Norman	36
10.3 Ce que j'ai aimé ou pas	36
10.3.1 Vincent	36
10.3.2 Hicham	36
10.3.3 Gabriel	36
10.3.4 Louis-Norman	36
11 ANNEXE 1 : Code source du programme pour PC	37
12 ANNEXE 2 : Code source du Bolide et de la station 1	38
13 ANNEXE 4 : Code source de la table FESTO	39
14 ANNEXE 5 : Code source du programme PIC	40
15 ANNEXE 4 : Script Shell du SOC8200	41

Table des figures

1	Vue d'ensemble du projet	5
2	Schéma bloc du Bolide	6
3	Schéma bloc de la table FESTO	8
4	Schéma bloc du SOC8200	9
5	Programme de contrôle principal	19
6	Options CAN avancées	20
7	Historique des actions	21
8	Héritage de la classe de contrôle du véhicule	25
9	Héritage de la classe de contrôle de la station no.1	25
10	Qui hérite du SPI ?	25
11	Qui hérite de l'I2C ?	25
12	Choix de la cible sur IAR	26
13	Schémas carte IO I2C, page 1	29
14	Schémas carte IO I2C, page 2	30
15	Schémas carte IO I2C, page 3	31
16	Schémas carte IO I2C, page 4	32
17	Couche TOP	33
18	Couche BOT	33
19	Silk Screen TOP	34
20	Solder mask TOP	34
21	Drill	35
22	Correctif	35

Liste des tableaux

1	Index des identifiants matériel CAN	14
2	Index des trames CAN	14
3	Index des communications CAN	14
4	Informations sur le bus I2C du bolide	24

1 Présentation du projet

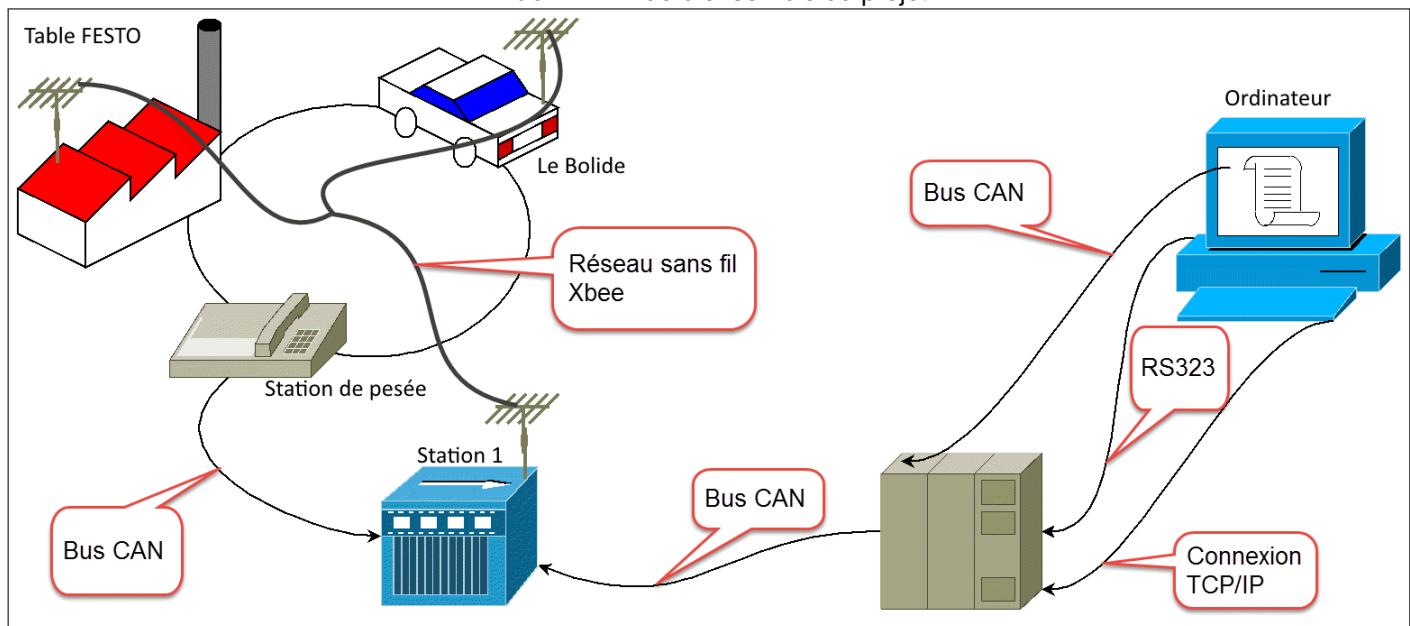
Le projet de la cinquième session consiste à réaliser (mettre un cours extrait des consignes)

- ⇒ Le Bolide
- ⇒ Carte Dallas DS89C450
- ⇒ Carte uPSD 3254A
- ⇒ SOC8200
- ⇒ Table FESTO
- ⇒ Carte PIC machin-chose-binouche
- ⇒ 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

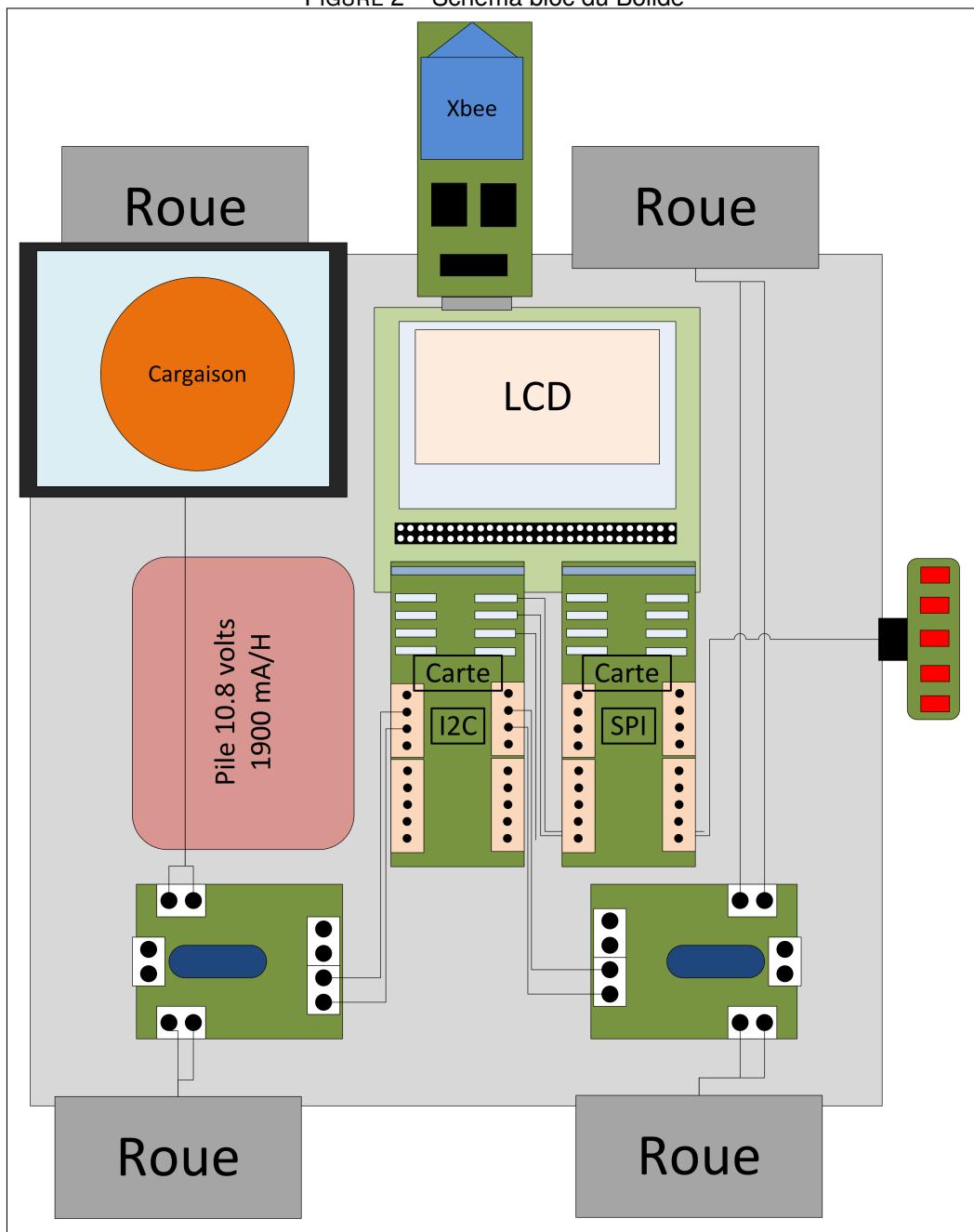
1.2 Schéma bloc vue d'ensemble du système

FIGURE 1 – Vue d'ensemble du projet



1.2.1 Schéma bloc du bolide

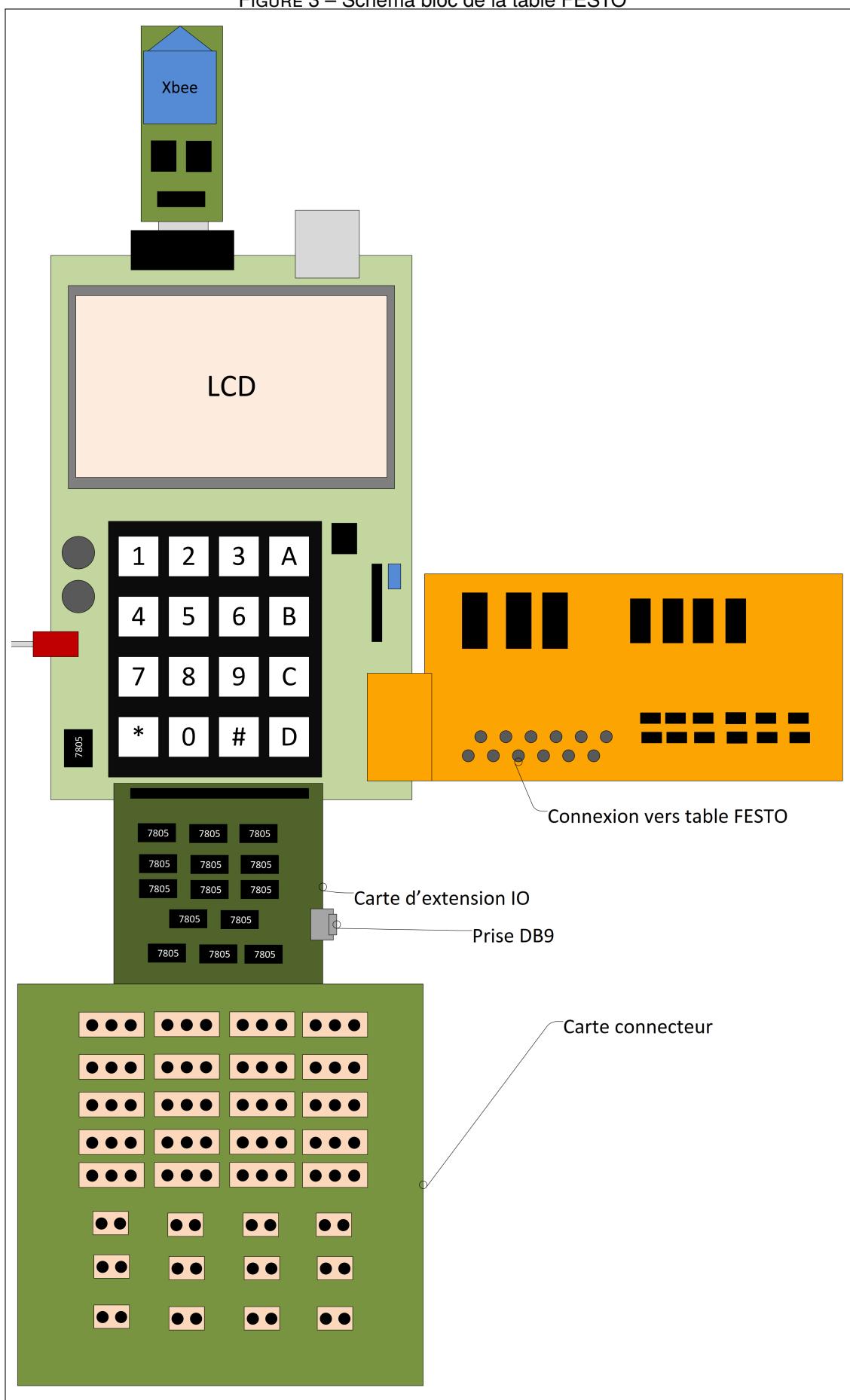
FIGURE 2 – Schéma bloc du Bolide



1.2.4 Schéma bloc de la station de pesée

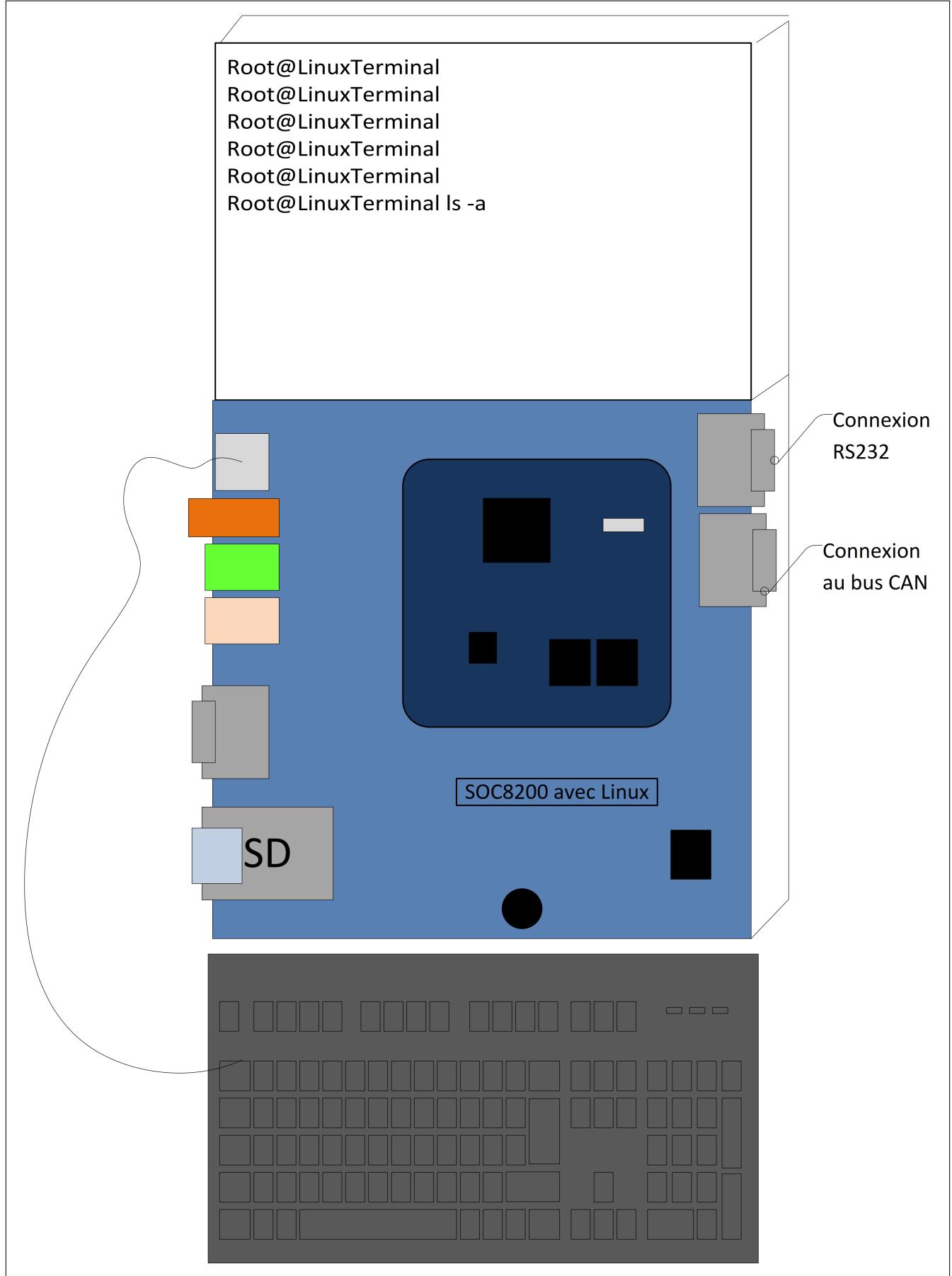
1.2.2 Schéma bloc de la table FESTO

FIGURE 3 – Schéma bloc de la table FESTO



1.2.3 Schéma bloc du SOC8200

FIGURE 4 – Schéma bloc du SOC8200



1.2.5 de la station no.1

1.3 Liste des logiciels

Terminaux

- UART Master 0.96
- Serializ3r 1.0.2
- TerraTerm
- Putty
- GTKterm 0.99.7-rc1
- Terminator
- CAPS
- tinyBootloader

Gestionnaires de projet

- MS Project 2010
- Git Hub

Compilateurs et IDE

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

Éditeurs de texte

- Notepad++
- BowPad
- medit 1.2.0

Schémas électriques

- OrCAD 16.2

Systèmes d'exploitation

- Windows 7 SP1

- Windows 8.1

- Windows XP SP3
- CentOS
- Lubuntu 14.10

Autres

- VMWare Workstation 10
- TeXmaker 4.3
- Dukto R6
- Dia
- Microsoft Visio 2013
- Festo configuration tool
- L^AT_EX(avec plug-in Doxygen)

1.4 Liste des trames

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 et d'une prise Ethernet. Il n'y a rien à faire, mise à part brancher les bons câbles aux bons endroits. Son rôle est de contrôler et diriger toute l'opération et de veiller au bon fonctionnement de chaque composantes à l'aide d'une application en Csharp. Le bloc 1 est le cerveau de l'usine.

2.2 Bloc 2

Le bloc 2 est composé du bolide et de la station no.1. Cette dernière, dont le cerveau est une carte uPSD, joue le rôle de centralisateur CAN. En effet, cette station reçoit des consignes¹ en provenance du PC, consignes qu'elle s'empresse d'expédier aux bons endroits via Xbee. De plus, cette station reçoit des informations de la station de pesée², de la table FESTO³ et du bolide⁴. Ces informations sont systématiquement retransmises au PC via le bus CAN.

2.3 Bloc 3

Le bloc 3 est composé d'une carte PIC (mettre modèle) et d'une balance (mettre modèle.) Comme son nom l'indique, la station de pesée pèse le bloc et envoie l'information (le poids) au PC et au SOC8200 via le bus CAN. La carte PIC fait office de convertisseur CAN vers RS232.

2.4 Bloc 4

Le bloc 4 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.

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.

-
1. Sous forme de trames
 2. Sous forme de trames CAN
 3. Via Xbee
 4. Idem

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

Chaque composante matériel, du Bolide au PC, dispose d'un identifiant CAN unique allant de 000 à 005. Chaque fonctionnalité dispose d'un code d'identification suivi de deux octets de données à transmettre.

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

TABLE 3 – Index des communications CAN

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

Exemples de trames CAN à transmettre au PC.

```
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
```

2.6.3 XBEE

Trois modules Xbee sont présent sur l'ensemble du projet, soit sur la station no.1 (la carte uPSD), sur la station no.2 (la table FESTO) et la station no.4, 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.

Note : mettre image d'un Xbee

2.7 Liste des pièces

- Carte Dallas
 - Carte uPSD
 - SOC 8200 (avec clavier et écran)
 - PIC18FmachinTruc
 - Carte d'extension SPI
 - Carte d'extension I2C
 - Carte CAN MCP2515
 - Xbee
 - Table FESTO
 - Carte connecteur IO 24 volts
 - Carte d'extension DAC ADC
 - Carte Xbee vers DB9
 - Câble Ethernet croisé
 - Câble Ethernet régulier
 - Câble DB9
 - Le Bolide
 -
 -
-

2.7.1 Liens web

Mettre lien vers GitHub

2.7.2 Datasheets

Note : Toutes les datasheets sont en format non-compressé. Vous pouvez zoomer sur le document PDF⁵ afin de lire l'intégralité de leurs premières pages.

74HC14

PNP Transistors		Product Specification																										
Hex Inverting Schmitt trigger		74HCT14																										
FEATURES																												
<ul style="list-style-type: none"> • Output capable of standard • 100% coverage SOT 																												
GENERAL DESCRIPTION																												
<p>The 74HCT14 is a high-speed Schmitt CMOS device and are pin compatible with the power Schottky TTL, ECL/TTL logic families.</p> <p>This device contains six hex inverting Schmitt trigger functions.</p> <p>The 74HCT14 provides six inverting buffers with Schmitt-trigger action. They are capable of transforming logic changing input signals into sharply defined, square output signals.</p>																												
LOGIC SYMBOL DATA																												
<table border="1"> <thead> <tr> <th>SYMBOL</th> <th>PARAMETER</th> <th>CONDITIONS</th> <th>TYPICAL</th> <th>UNIT</th> </tr> </thead> <tbody> <tr> <td>Max. Input</td> <td>Input voltage, V_{IN}, mA</td> <td>V_{IN} = 10, 15, 20, 25, 30 V</td> <td>±0.5</td> <td>mA</td> </tr> <tr> <td>I_{IN}</td> <td>Input current</td> <td>V_{IN} = 10, 15, 20, 25, 30 V</td> <td>±2.0</td> <td>μA</td> </tr> <tr> <td>C_{IN}</td> <td>Input capacitance</td> <td>—</td> <td>—</td> <td>—</td> </tr> <tr> <td>D_{OUT}</td> <td>Output capacitance, open gate</td> <td>—</td> <td>—</td> <td>—</td> </tr> </tbody> </table>				SYMBOL	PARAMETER	CONDITIONS	TYPICAL	UNIT	Max. Input	Input voltage, V _{IN} , mA	V _{IN} = 10, 15, 20, 25, 30 V	±0.5	mA	I _{IN}	Input current	V _{IN} = 10, 15, 20, 25, 30 V	±2.0	μA	C _{IN}	Input capacitance	—	—	—	D _{OUT}	Output capacitance, open gate	—	—	—
SYMBOL	PARAMETER	CONDITIONS	TYPICAL	UNIT																								
Max. Input	Input voltage, V _{IN} , mA	V _{IN} = 10, 15, 20, 25, 30 V	±0.5	mA																								
I _{IN}	Input current	V _{IN} = 10, 15, 20, 25, 30 V	±2.0	μA																								
C _{IN}	Input capacitance	—	—	—																								
D _{OUT}	Output capacitance, open gate	—	—	—																								
Notes																												
<ol style="list-style-type: none"> C_{IN} is used to determine the dynamic power dissipation (P_{DQ}) in μW. $P_{DQ} = C_{IN} \cdot V_{DD} \cdot (V_{DD} - V_{IN})^2 \cdot f_{IN}$, where f_{IN} = input frequency. L_{IN} = input load in pF. For HCT conditions: $V_{DD} = 5$ V, $V_{IN} = 0$ to 5 V. For HCT conditions: $V_{DD} = 5$ V, $V_{IN} = 0$ to 5 V. For HCT conditions: $V_{DD} = 5$ V, $V_{IN} = 0$ to 5 V. For HCT conditions: $V_{DD} = 5$ V, $V_{IN} = 0$ to 5 V. 																												
ORDERING INFORMATION																												
See "74HCT14/1400 Logic Package Information".																												

LineSensors

Registers:

You can read the I/O line status by issuing an I/O read of the programmed address.

A single byte representing the status of the 8 sensors. All bits are inverted, ranging from 0 to 255, where 255 means one or more sensors are triggered.

	Address	Description	Address #	Address #	Address #	Decimal Meaning
0	0	0	0	0	0	0
1	0	0	0	0	0	1
0	1	0	0	0	0	2
0	0	1	0	0	0	3
0	0	0	1	0	0	4
1	0	1	1	0	0	5
0	1	1	0	0	0	6
1	1	1	0	0	0	7
0	0	0	1	0	0	8
1	0	0	0	1	0	9
0	1	0	1	0	0	10
1	1	0	1	0	0	11
0	0	1	1	0	0	12
1	0	1	0	1	0	13
0	1	1	1	0	0	14
1	1	1	1	0	0	15
0	0	0	0	1	0	16
1	0	0	0	0	1	17
0	1	0	0	0	1	18

PCF8573

	REMOTE 4-BIT I/O EXPANDER FOR PC-BUS PCF8574															
<p>Product Summary</p> <p>The PCF8574 is a remote 4-bit bidirectional I/O expander designed for the PC-Bus interface. It features a built-in 4-bit bidirectional parallel port (PPI) which includes high-current drivers and receivers. The PCF8574 is a high-speed device with a maximum propagation delay of 10 ns. It also has a low power consumption of 1.5 mA at 5 V. The device is housed in a compact 16-pin DIP package.</p> <p>Features</p> <ul style="list-style-type: none"> • Low Standby Current Consumption of 1.5 mA at Maximum • PC-Bus Compatible • Open-Drain Input/Output • Compatible With Most Microcontrollers • Latched Outputs With High-Speed Drive Capability for Directly Driving LEDs • Logic Input Current Exceeds 100-nA Per JEDEC Specification Class II 																
<p>Pinouts</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 33%;">DW OR IN PACKAGE</td> <td style="width: 33%;">PIN NUMBER</td> <td style="width: 33%;">PIN PACKAGE</td> </tr> <tr> <td></td> <td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</td> <td></td> </tr> <tr> <td></td> <td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</td> <td></td> </tr> <tr> <td></td> <td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16</td> <td></td> </tr> </table> <p>Pin Descriptions</p> <p>DW OR IN PACKAGE</p> <ul style="list-style-type: none"> • A1: Address • A2: Address • A3: Address • A4: Address • IN+: Input • IN-: Input • PW: Power • GND: Ground <p>PIN NUMBER</p> <ul style="list-style-type: none"> • 1: A1 • 2: A2 • 3: A3 • 4: A4 • 5: IN+ • 6: IN- • 7: PW • 8: GND • 9: NC • 10: NC • 11: NC • 12: NC • 13: NC • 14: NC • 15: NC • 16: NC <p>PIN PACKAGE</p> <ul style="list-style-type: none"> • 1: A1 • 2: A2 • 3: A3 • 4: A4 • 5: IN+ • 6: IN- • 7: PW • 8: GND • 9: NC • 10: NC • 11: NC • 12: NC • 13: NC • 14: NC • 15: NC • 16: NC <p>Notes:</p> <p>(1) DW = No internal connection</p> <p>(2) PC = No internal connection</p>		DW OR IN PACKAGE	PIN NUMBER	PIN PACKAGE		1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16			1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16			1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16				
DW OR IN PACKAGE	PIN NUMBER	PIN PACKAGE														
	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16															
	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16															
	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16															
<p>Description/Ordering Information</p> <p>The PCF8574 is a bidirectional I/O expander for the PC-Bus interface. It is designed for the PC-Bus interface.</p> <p>The PCF8574 provides four remote bidirectional I/O expanders for most microcontroller families via the PC-Bus interface.</p> <p>The device features a built-in quad-buffered 16-bit port (PPI), including bidirectional outputs with high-current drive capability for direct driving of LEDs. The PPI port is controlled by a 4-bit address register. The address is set by the user of a bidirectional serial control signal, A₄. It provides the 4-bit bus ID in the mode, a 16-bit memory access to 4 Kbytes of memory, and a 16-bit bidirectional port. The PCF8574 is a high-speed device with a maximum propagation delay of 10 ns. It also has a low power consumption of 1.5 mA at 5 V. The device is housed in a compact 16-pin DIP package.</p> <p>Ordering Information</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <th style="width: 10%;">Part</th> <th style="width: 20%;">Package</th> <th style="width: 20%;">Temperature Range</th> <th style="width: 10%;">Voltage</th> <th style="width: 10%;">Ordering</th> </tr> <tr> <td>PCF8574</td> <td>16-pin DIP</td> <td>-40°C to +85°C</td> <td>5 V</td> <td>PCF8574</td> </tr> <tr> <td>PCF8574T</td> <td>16-pin DIP</td> <td>-40°C to +85°C</td> <td>5 V</td> <td>PCF8574T</td> </tr> </table> <p>Caution: Please be aware that an inductor noise cancellation diode is required, and one is circuit board layout of the device.</p>		Part	Package	Temperature Range	Voltage	Ordering	PCF8574	16-pin DIP	-40°C to +85°C	5 V	PCF8574	PCF8574T	16-pin DIP	-40°C to +85°C	5 V	PCF8574T
Part	Package	Temperature Range	Voltage	Ordering												
PCF8574	16-pin DIP	-40°C to +85°C	5 V	PCF8574												
PCF8574T	16-pin DIP	-40°C to +85°C	5 V	PCF8574T												

DAC6574

LM3914

Texas
Instruments

LM3914

uPSD3254A

DS89C450

OPT101

BB | Burr-Brown Products
from Texas Instruments

OPT101

MONOLITHIC PHOTODIODE AND SINGLE-SUPPLY TRANSIMPEDANCE AMPLIFIER

FEATURES <ul style="list-style-type: none"> ■ TRANSIMPEDANCE GAIN: 10^6 TO 10^9 ■ PHOTODIODE: 10^{-1} TO 10^{-9} AMPERES ■ HIGH RESPONSIVITY: CLEAN ROOM ■ BANDWIDTH: 100 Hz TO 10^6 Hz ■ LOW NOISE: 10^{-16} VOLTS ■ AVAILABLE IN SOT-89 DIP AND SOT-84 SMD PACKAGE 	DESCRIPTION <p>The OPT101 is a monolithic photodiode with matching transimpedance op-amp circuitry housed in a single integrated circuit package. The device is designed for fast photodiode response, making it ideal for optical sensing applications.</p> <p>The integrated combination of photodiode and transimpedance amplifier provides a solution to the problems commonly encountered in discrete designs such as noise, component matching, and layout. The device can be used in a variety of applications, including photoelectric sensors, photoelectric switches, and photoelectric comparators. The OPT101 is available in SOT-89 and SOT-84 packages and operates over a wide temperature range from -27°C to $+70^\circ\text{C}$. The device is also available in tape and reel.</p> <p>Temperature range: -27°C to $+70^\circ\text{C}$.</p>
--	--

APPLICATIONS

- MEDICAL INSTRUMENTATION
- LABORATORY INSTRUMENTATION
- INDUSTRIAL PROCESS CONTROLS
- PHOTOGRAPHIC ANALYZERS
- IMAGE PROCESSORS
- SWING STYLUS
- CURRENCY CHANGERS

ANSWER LINE Please let us know if an important article containing valuable information would be of interest in a future issue of *EE Times*. Your comments and suggestions are welcome.

DISCLAIMER The information contained in this document does not constitute legal disclaimers or warranties. © 1993 Texas Instruments Incorporated. All rights reserved. TI and the TI logo are registered trademarks of Texas Instruments Incorporated.

Texas Instruments

Copyright © 1993, Texas Instruments Incorporated

SaberTooth motor drive

5. 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.

74HC14

PNP Semiconductors		Product specification			
Hex inverting Schmitt trigger		74HC/HCT14			
FEATURES					
• Output capability: standard					
• I _{OL} : category SII					
GENERAL DESCRIPTION					
The 74HC/HCT14 are hex inverting Schmitt triggers. CMOS devices and are pin compatible with the lower threshold TTL/HTL. These devices have a wide operating voltage range from 2.7 V to 5.5 V.					
The 74HC/HCT14 provide six inverting buffers with Schmitt-trigger action. They are capable of functioning at clock rates up to 15 MHz. The outputs can drive standard logic, Schmitt-trigger logic, or other CMOS devices.					
QUICK REFERENCE DATA					
(V _D = 2.7 V to 5.5 V, T = 15°C)					
SYMBOL		PARAMETER	CONDITIONS		
			TYPICAL		
I_{OL} (I_{OSS})		Output current (sink/source) at $V_O = 0.8V_{DD}$	52		
t_{PD}		Power dissipation at $V_O = 0.8V_{DD}$	17		
t_{R}		Invert response	1.0		
t_{IOL}		Output load time constant for gate inputs (inverter and Z)	3.2		
t_{IOH}		Output load time constant for gate inputs (open drain)	1.7		
Notes					
1. t_{R} is measured at $V_O = 0.8V_{DD}$ for a 100% duty cycle variation of V_O .					
2. $P_D = G(V_{DD} - V_O)^2 \times t_{PD}$, where $G = 1.0 \mu A/V^2$ at $V_O = 0.8V_{DD}$.					
3. Input response time is $t_{R} + t_{IOL}$.					
4. I_{OL} is total output load current.					
5. I_{OSS} is total output sink current.					
6. t_{IOL} is output load time constant for gate inputs.					
7. t_{IOH} is output load time constant for open drain outputs.					
8. The HC/HCT condition is $V_{DD} = 2.7V$ to $5.5V$ and $V_{SS} = 0V$.					
9. The HCT condition is $V_{DD} = 3.6V$ to $5.5V$ and $V_{SS} = 0V$.					
ORDERING INFORMATION					
See 74HC/HCT14 Standard Logic Package Information .					

LineSensors

Registers:

You can read the ADC-Analog sensor by issuing an ADC-read of the programmed address. A single bit representing the status of the line sensors will be selected, ranging from 0 to 31, where 0 means 1 sensor is active and 31 means all sensors are 100%.

Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor 5	Decimal Meaning
0	0	0	0	0	1
0	0	0	0	0	2
1	0	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
0	1	1	1	0	13
1	1	1	1	0	14
0	0	0	0	1	15
0	1	0	0	1	16
1	0	0	0	1	17
0	1	0	0	1	18

PCF8573

DAC6574

The figure shows a detailed block diagram of the DAT4517. At the top, a 'Digital Input' section includes a 'Digital Filter' and a 'Digital-to-Analog Converter'. The DAC output is connected to a 'Digital-to-Optical Converter' (labeled 'Digital-to-Optical'). This converter has two outputs: one to an 'Optical Transmitter' and another to a 'Digital-to-Optical Receiver'. The receiver's output is connected to a 'Digital-to-Analog Converter' (labeled 'Digital-to-Analog'), which then feeds into an 'Analog Filter'. Below the digital sections, there is a 'Power Management' block containing a 'Power Control' section with a 'Power Control Logic' and a 'Power Control DAC'. A 'Power Control DAC' also receives input from the 'Digital-to-Optical Receiver'. The 'Power Management' block is connected to an 'AC/DC Power Supply' and an 'AC/DC Power Supply' for the 'Digital-to-Optical Receiver'. The 'Digital-to-Optical Receiver' also receives power from an 'AC/DC Power Supply'. The 'Digital-to-Optical Receiver' and 'Digital-to-Analog Converter' are connected to a 'Digital-to-Optical Converter' at the bottom, which is labeled 'Digital-to-Optical'.

LM3914

uPSD3254A

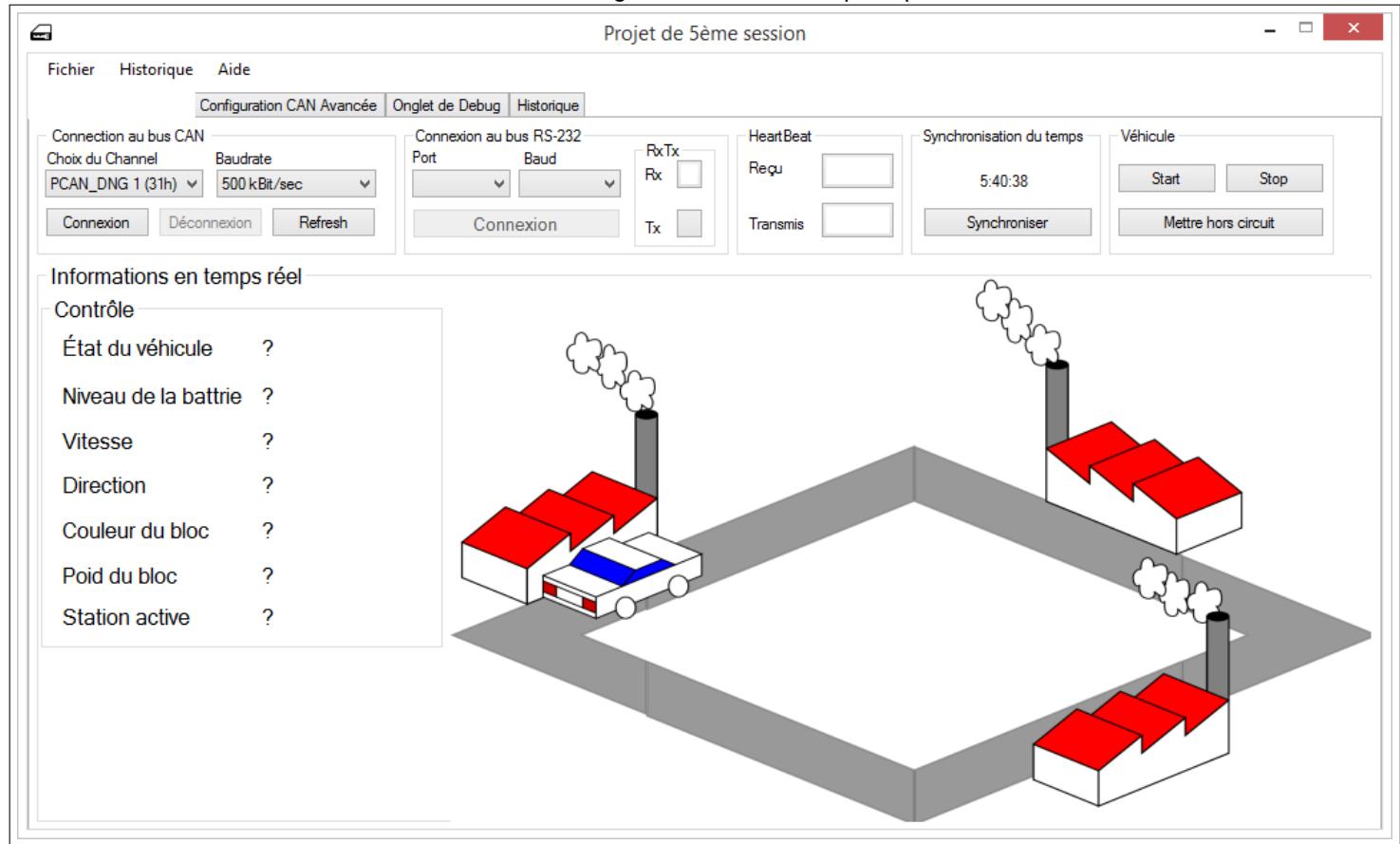
DS89C450

OPT101

SaberTooth motor drive

3 Interface PC

FIGURE 5 – Programme de contrôle principal

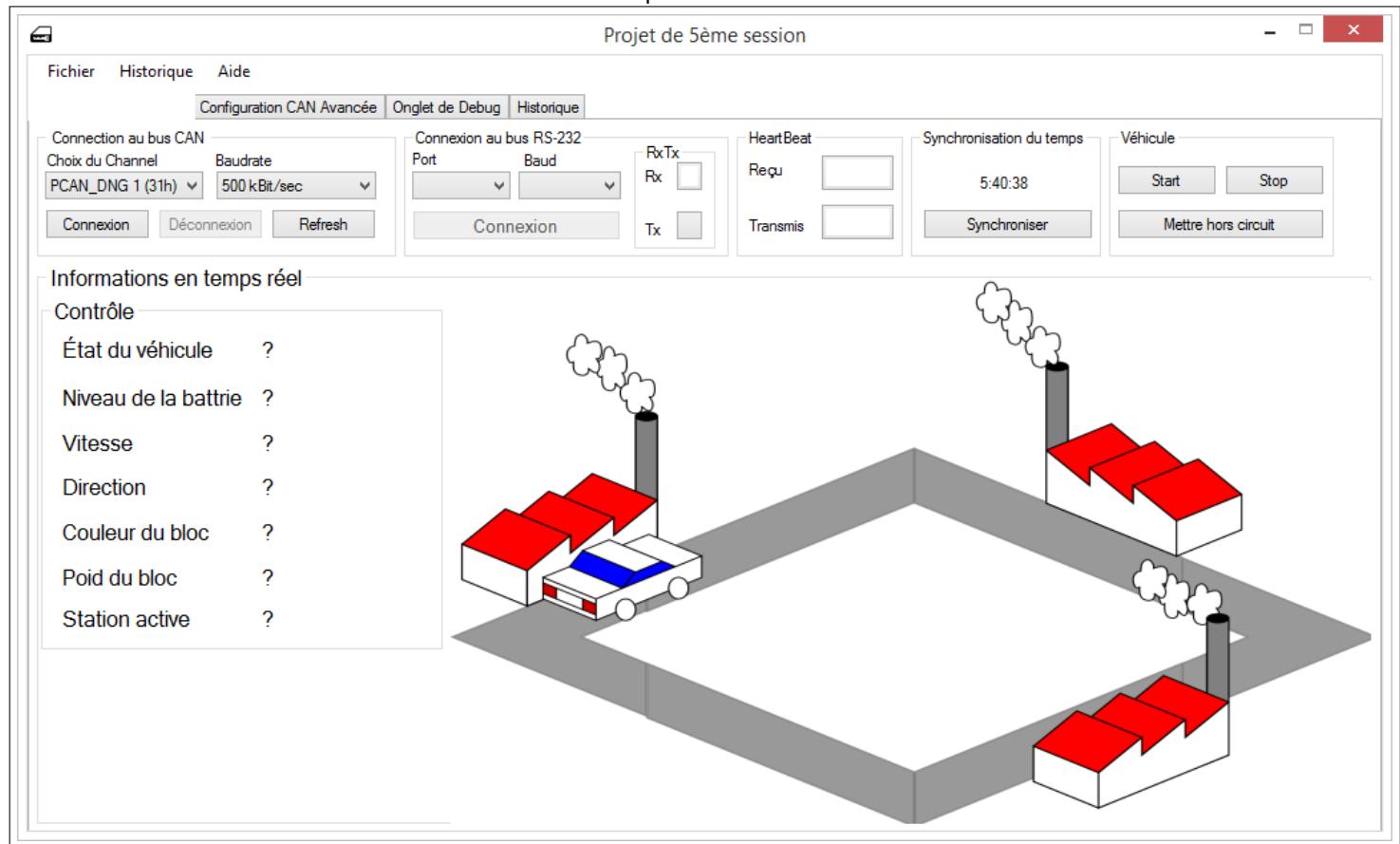


Notre programme, écrit en C# à l'aide de Visual Studio, peut se connecter au bus CAN via une carte SPI⁶ et au bus RS232 via un câble DB9 ou USB⁷. La connexion RS232 sert à l'envoi et à la réception du HeartBeat afin d'informer le SOC8200 si l'ordinateur en venait à connaître une défaillance. De plus, 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 », inscrire son heure de référence sur la station no.1 via le bus CAN.

6. Spécifier le fabricant

7. S'il y a présence d'un FTDI

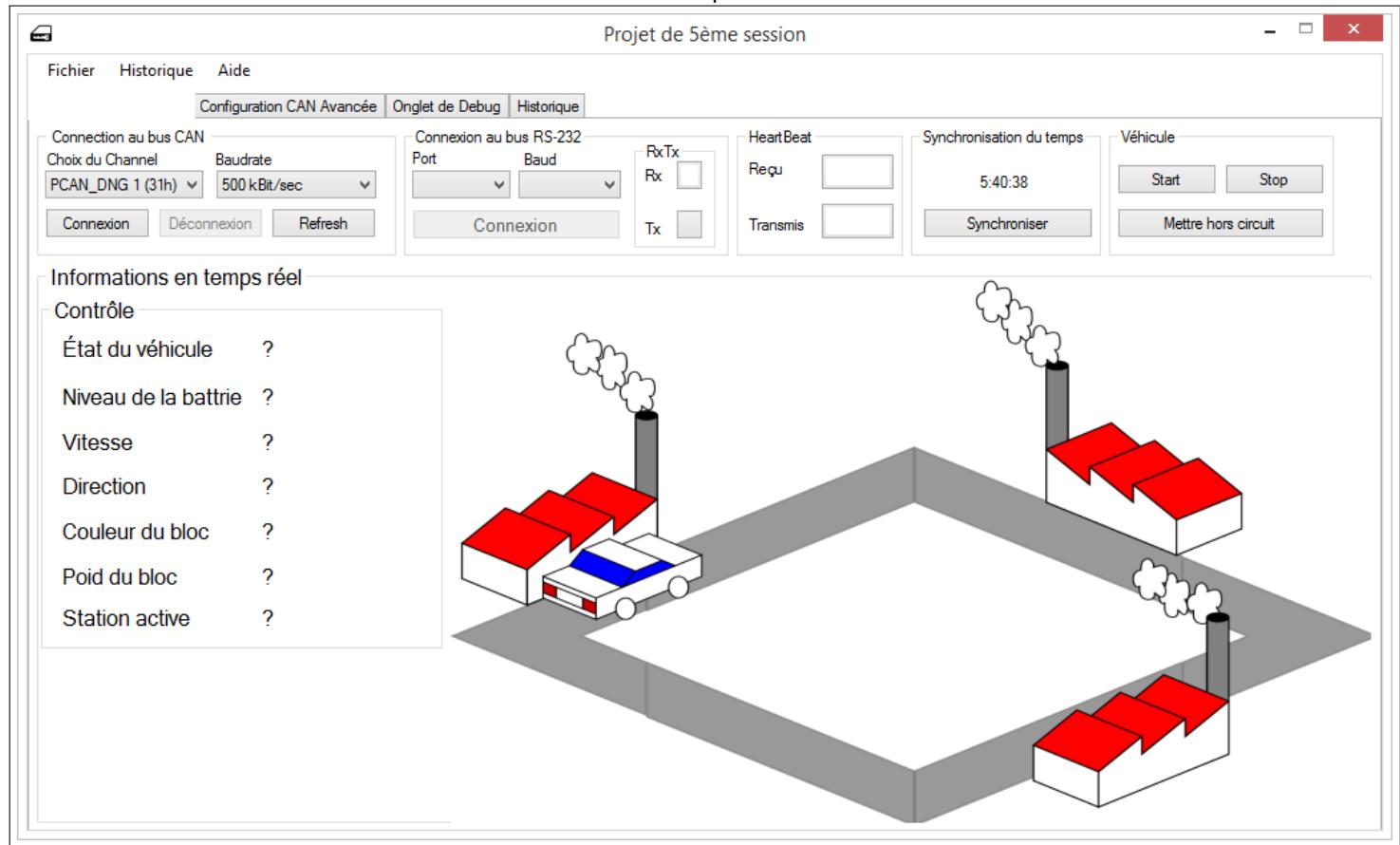
FIGURE 6 – Options CAN avancées



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 7 – Historique des actions



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

3.1.1 Exemple d'historique typique

Insérer copié-collé de l'historique ici

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

D'un commun accord de l'équipe, le programme du SOC2800 est écrit en script Shell. La principale raison de ce choix est Sourcery Codebench lui-même. La gestion des projets avec Sourcery est un cauchemar. Quant à la nécessité de sauvegarder pour compiler et d'utiliser une machine virtuelle, elle ne viennent qu'agraver la situation. De plus, son gestionnaire de licence⁸ frustré quiconque souhaite l'utiliser. L'utilisation du script shell est à la fois plus simple et permet de faire plus en moins de temps.

4.2 Schéma bloc du script shell

De tous les scripts, *Projet.sh* est le maître, et contient l'équivalent du main. Ce fichier initialise le port série, le bus can, la lecture et l'envoi des heartbeat en asynchrone, ainsi que la lecture du clavier USB. Le script *Projet.sh* peut aussi prendre la relève du bus CAN si le PC est dans l'incapacité d'assurer ses fonctions. Quant aux autres scripts, soit *connexion.sh*, *RxCan.sh*, *tcp.sh* et *Envoi.sh*, ce ne sont que des fonctionnalité que que *Projet.sh* appelle en asynchrone.

4.3 Gestion des processus et du temps de CPU

La passe-passe du coyote

4.4 Format et récupération des logs

Toutes les trames CAN reçues sont enregistrées dans le fichier «histocan» dont voici un court aperçu :

```
can0    1  [3] 08 00 FF
can0    4  [3] 04 00 FF
can0    1  [3] 08 01 FF
can0    4  [3] 04 00 FF
can0    1  [3] 08 01 FF
can0    4  [3] 04 00 FF
can0    1  [3] 08 01 FF
can0    4  [3] 04 00 FF
```

De plus, un autre fichier (histocandate) contient l'heure et la date des trames reçues.

```
Thu Jan  1 00:05:28 UTC 1970
can0 4 [3] 04 01 FF
Thu Jan  1 00:06:13 UTC 1970
can0 4 [3] 04 00 FF
Thu Jan  1 00:06:16 UTC 1970
can0 4 [3] 04 02 FF
Thu Jan  1 00:06:18 UTC 1970
can0 4 [3] 04 02 FF
Thu Jan  1 00:06:50 UTC 1970
```

4.5 Liste des tests et logiciels

8. 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 du bolide

Le programme de la station 1 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 deux 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. et ses divers capteurs.

5.1 La station no.1

La station no.1 est composé de uPSD 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 (qui s'appelle Bloc no.3 dans le cahier de consignes) est composé de la table FESTO, de la carte uPSD, de la carte d'extensions IO que nous avons réalisés et d'un Xbee.

5.4 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

5.3 Schéma des héritages de classes

Quelles classes utilisent quelles autres classes dans notre code ?

FIGURE 8 – Héritage de la classe de contrôle du véhicule

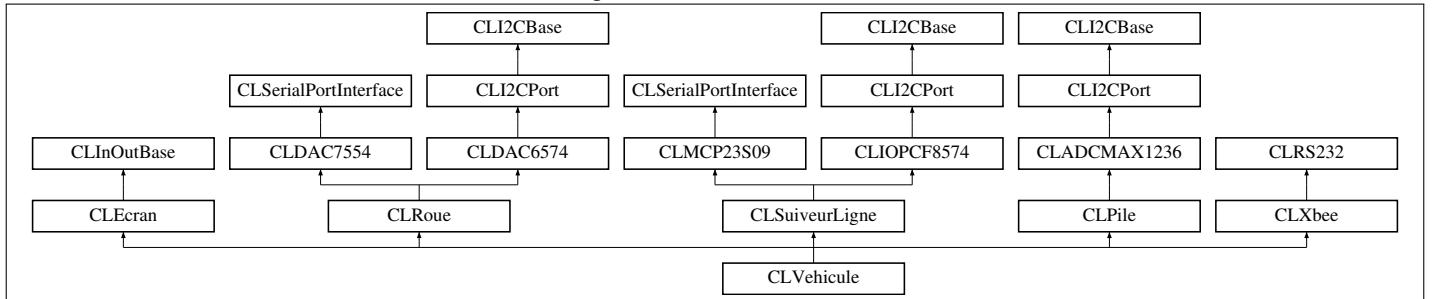


FIGURE 9 – Héritage de la classe de contrôle de la station no.1

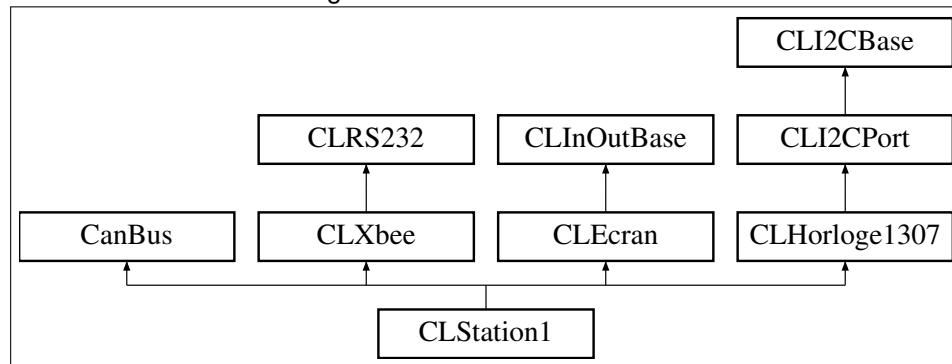


FIGURE 10 – Qui hérite du SPI ?

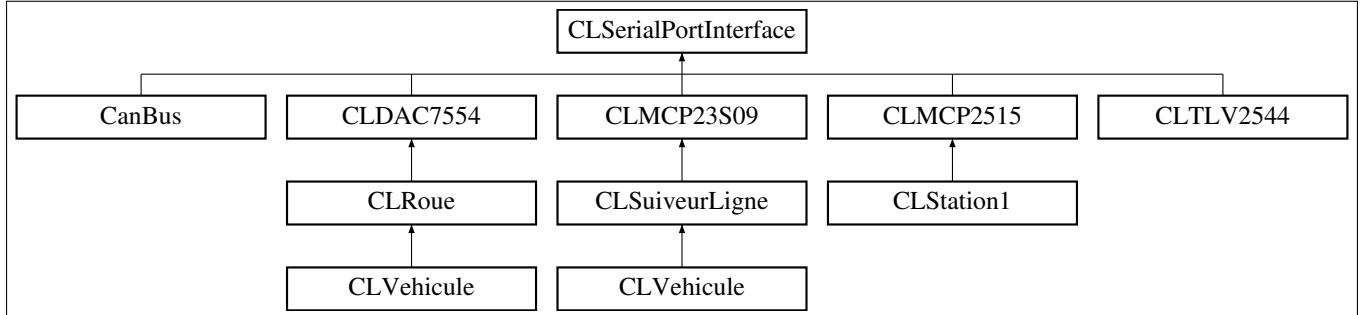
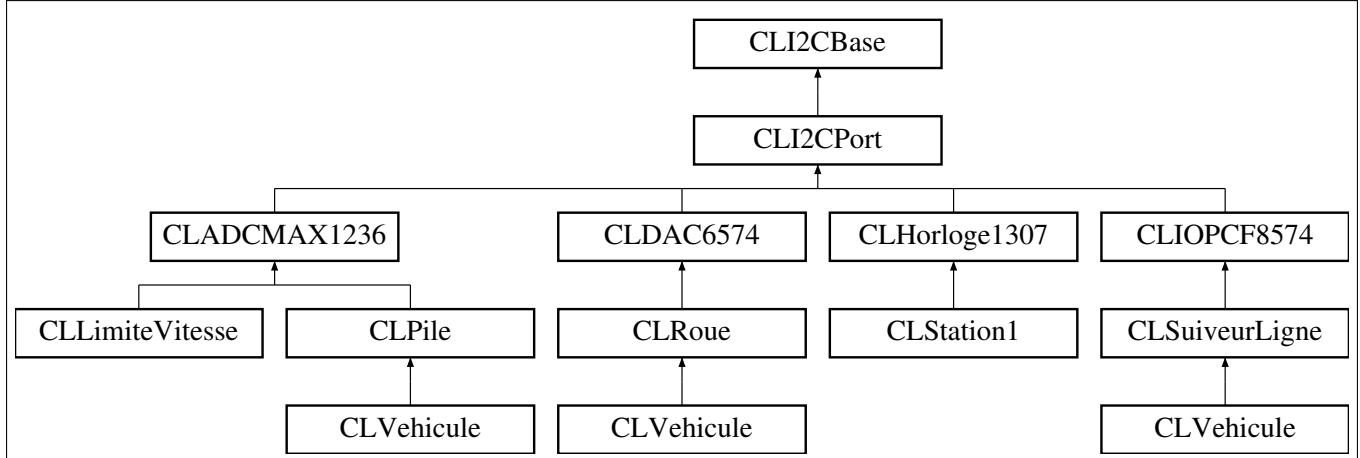


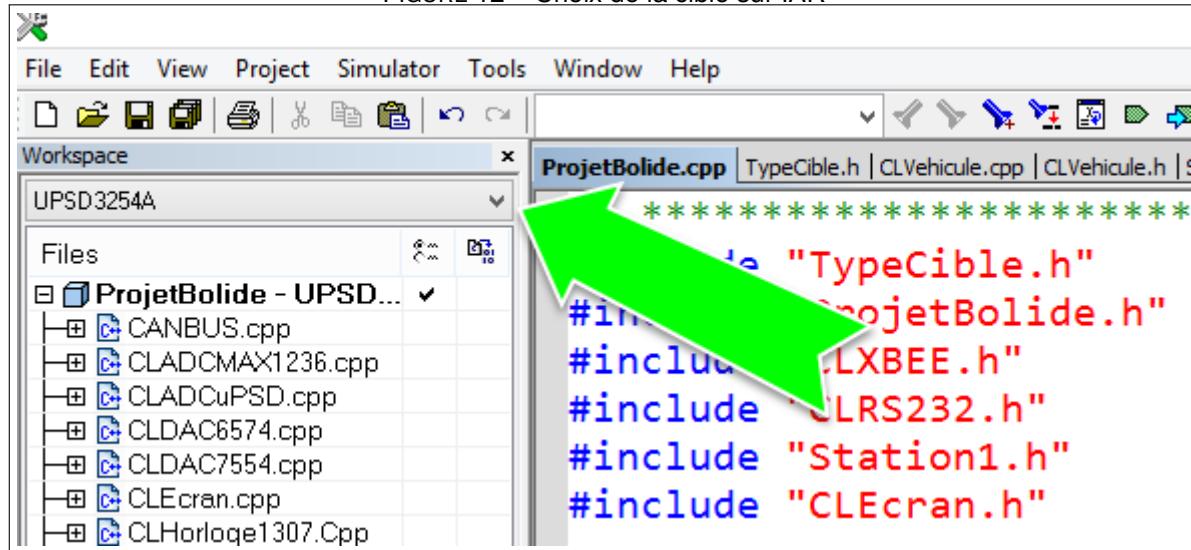
FIGURE 11 – Qui hérite de l'I2C ?



5.5 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 uPSD.

FIGURE 12 – 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 uPSD, 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

```

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

```

5.6 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...

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

8 Schémas OrCAD

FIGURE 13 – Schémas carte IO I2C, page 1

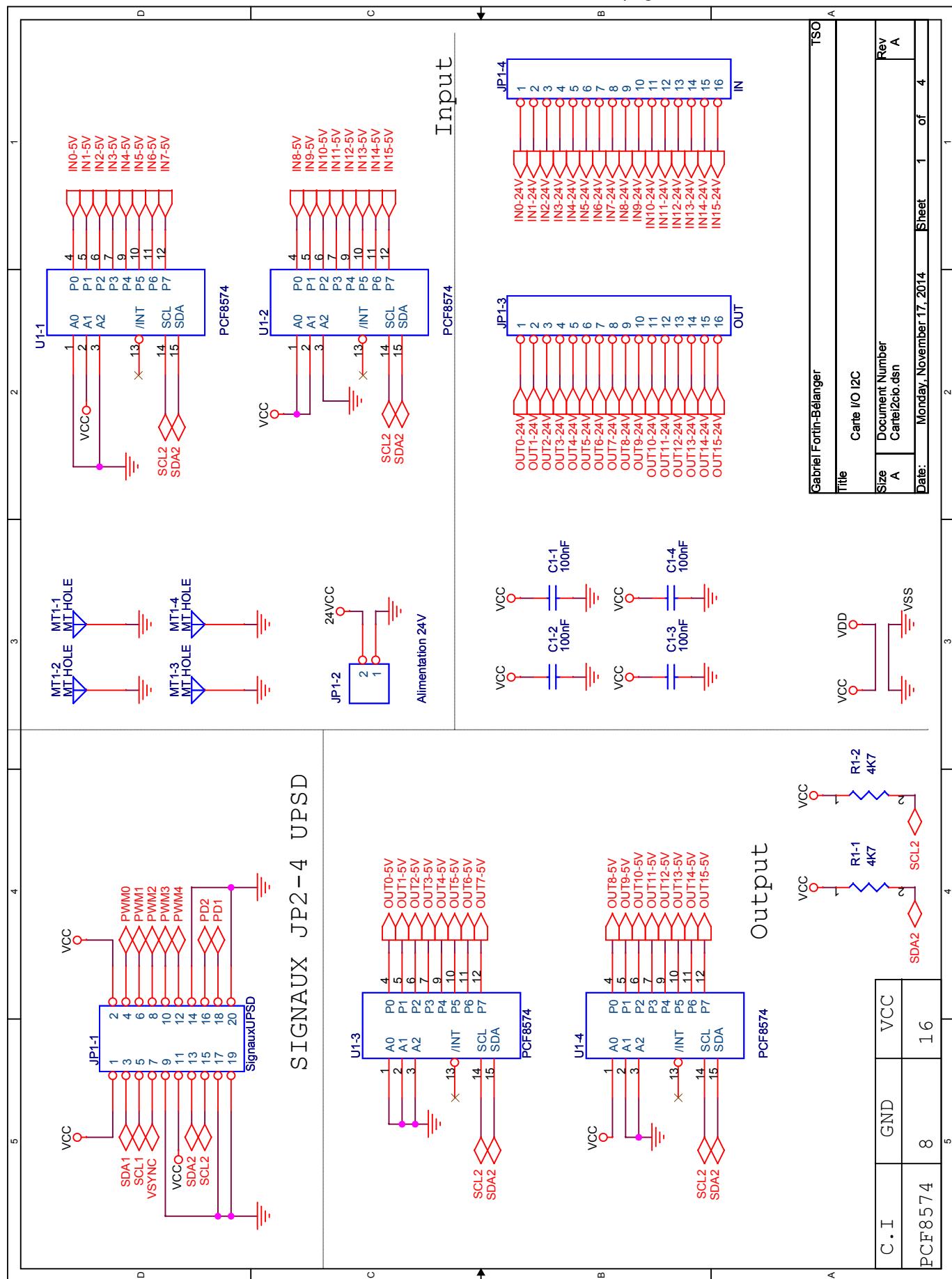


FIGURE 14 – Schémas carte IO I2C, page 2

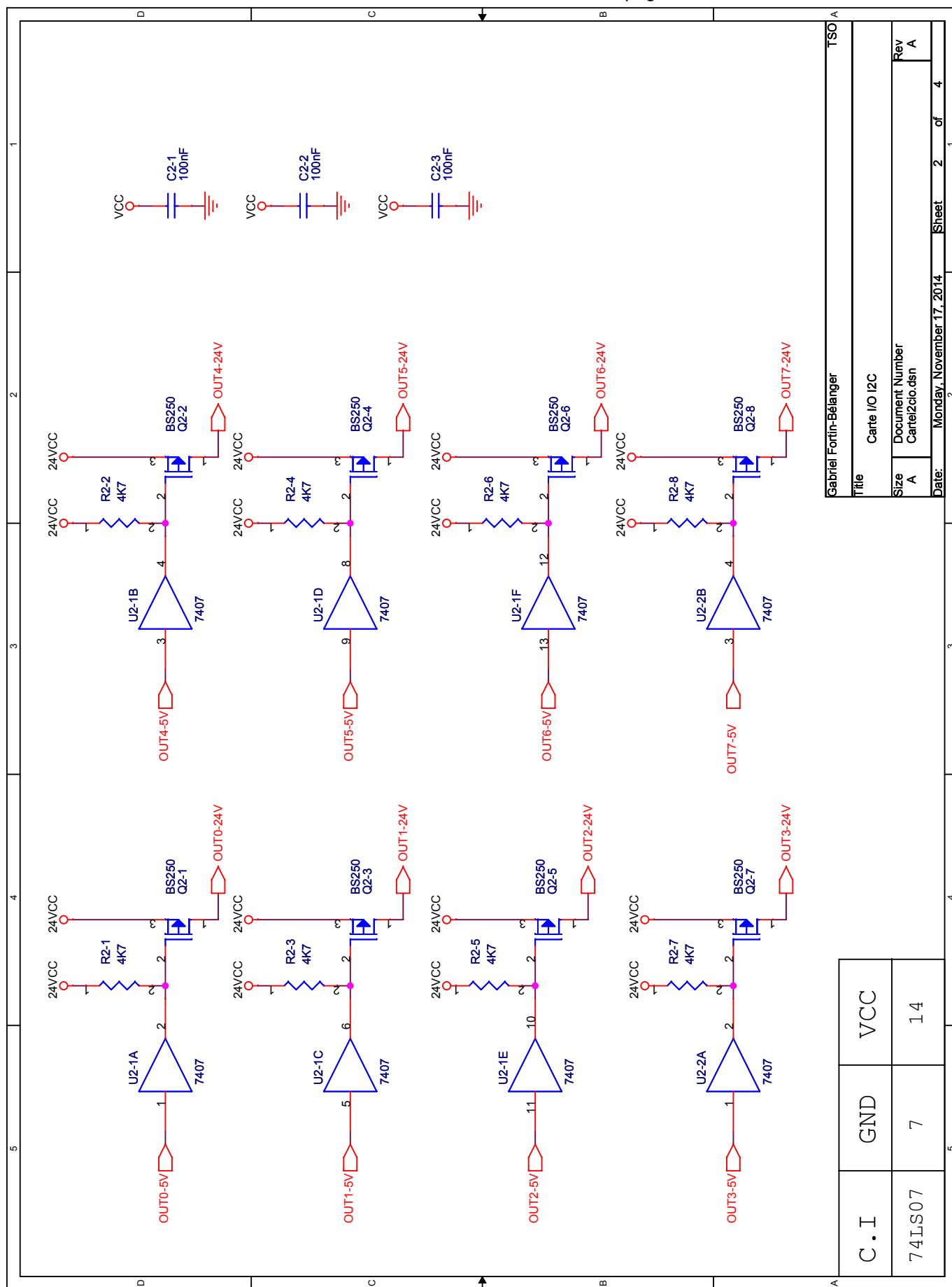


FIGURE 15 – Schémas carte IO I2C, page 3

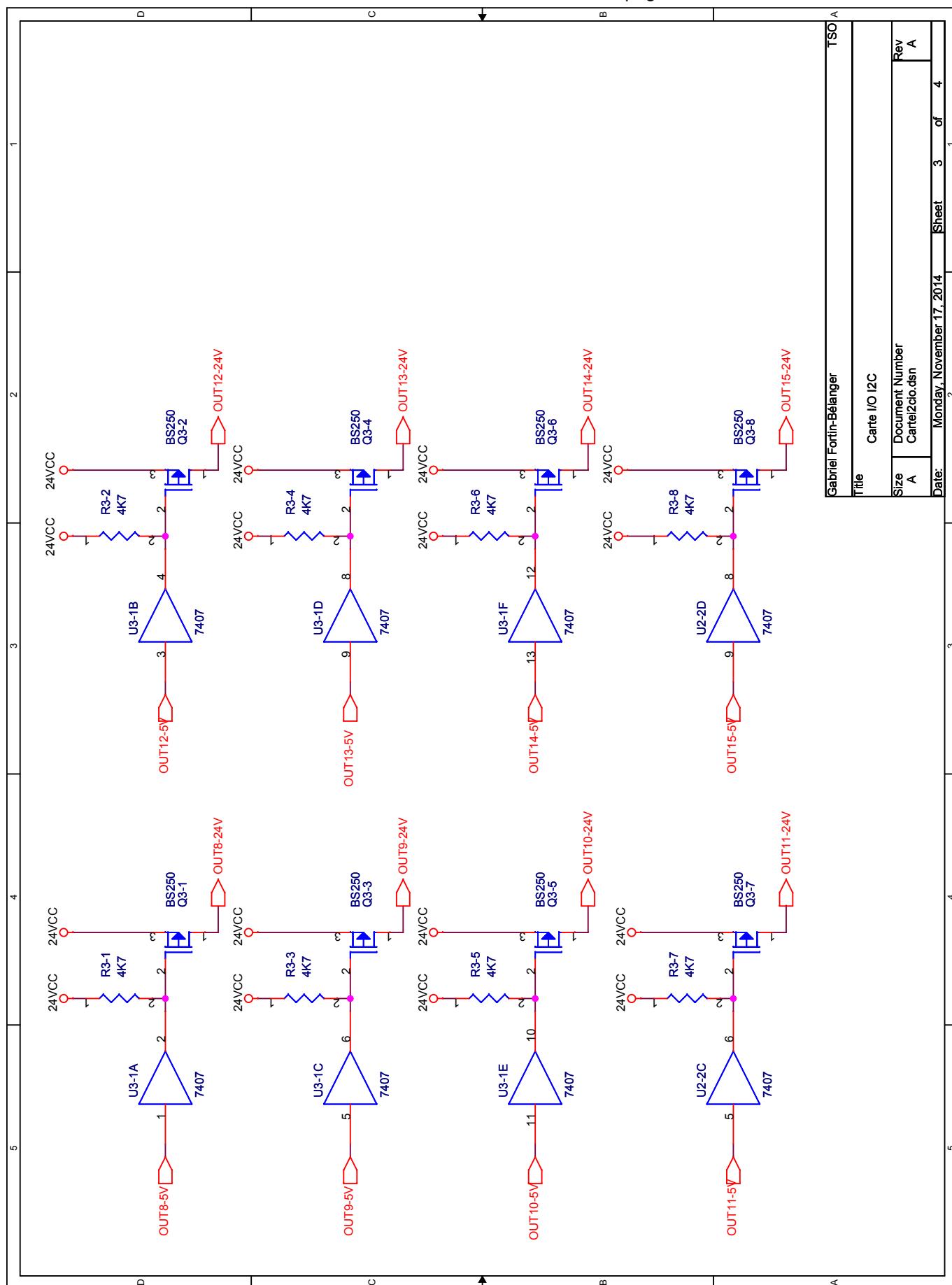
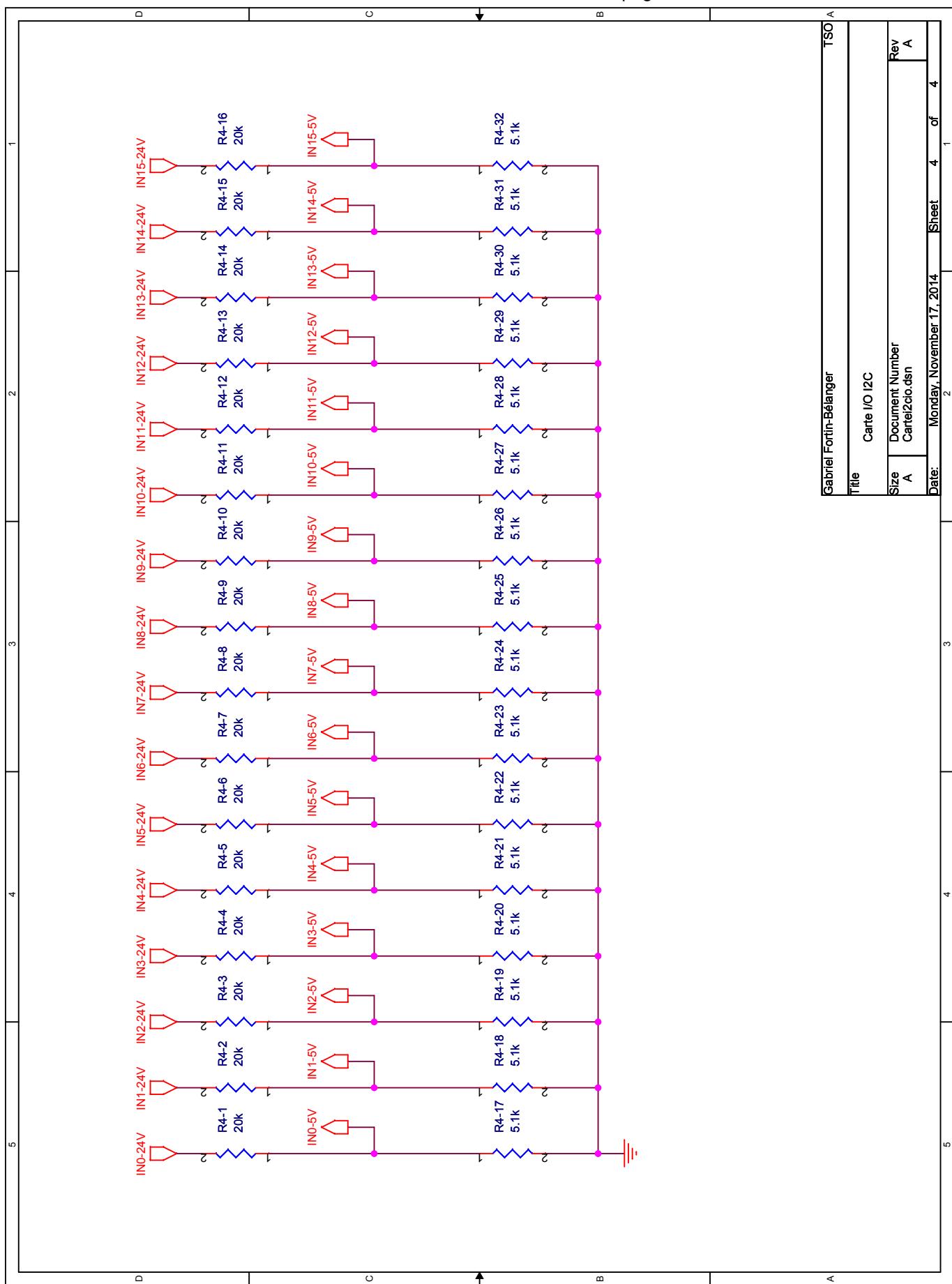


FIGURE 16 – Schémas carte IO I2C, page 4



Gabriel Fortin-Bélanger		TSO A
Title	Carte I/O I2C	
Size	Document Number	Rev
A	CarteI2Ci.dsn	A
Date:	Monday, November 17, 2014	Sheet 4 of 4

9 Fichiers Gerbers

Une carte d'extension, dont voici les images GERBER⁹, à été réalisée avec OrCAD 16.2 et gravée à l'aide de la rutilante LPKF départementale.

FIGURE 17 – Couche TOP

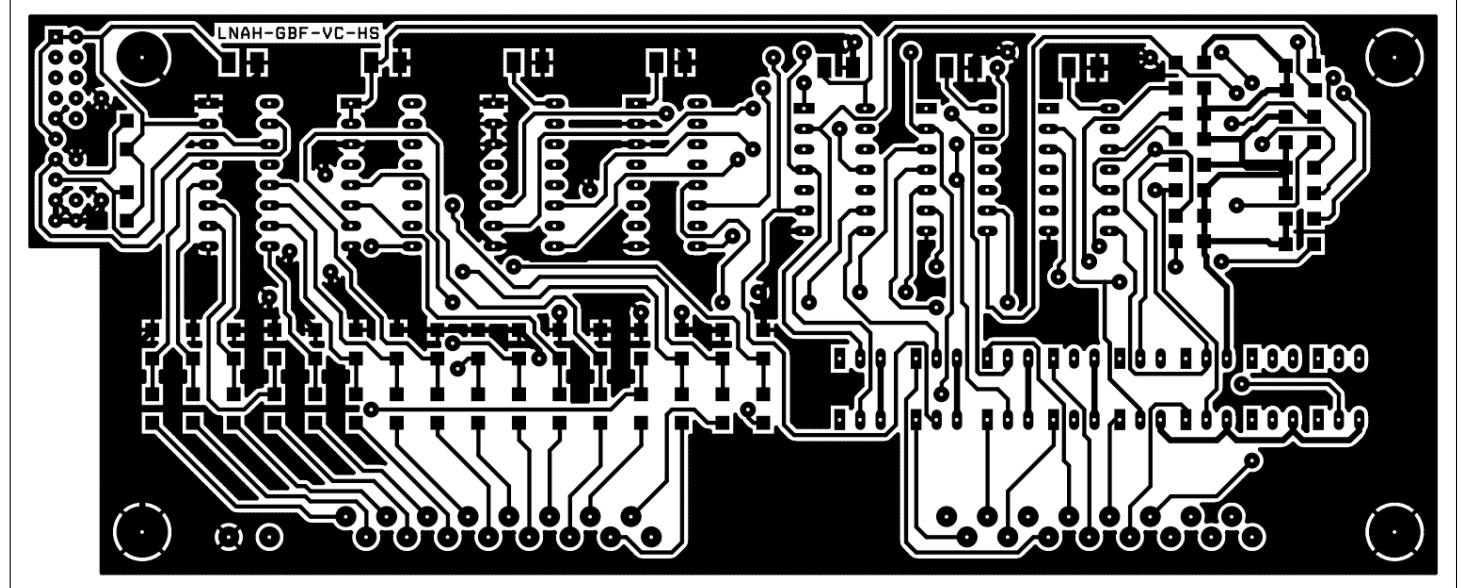


FIGURE 18 – Couche BOT

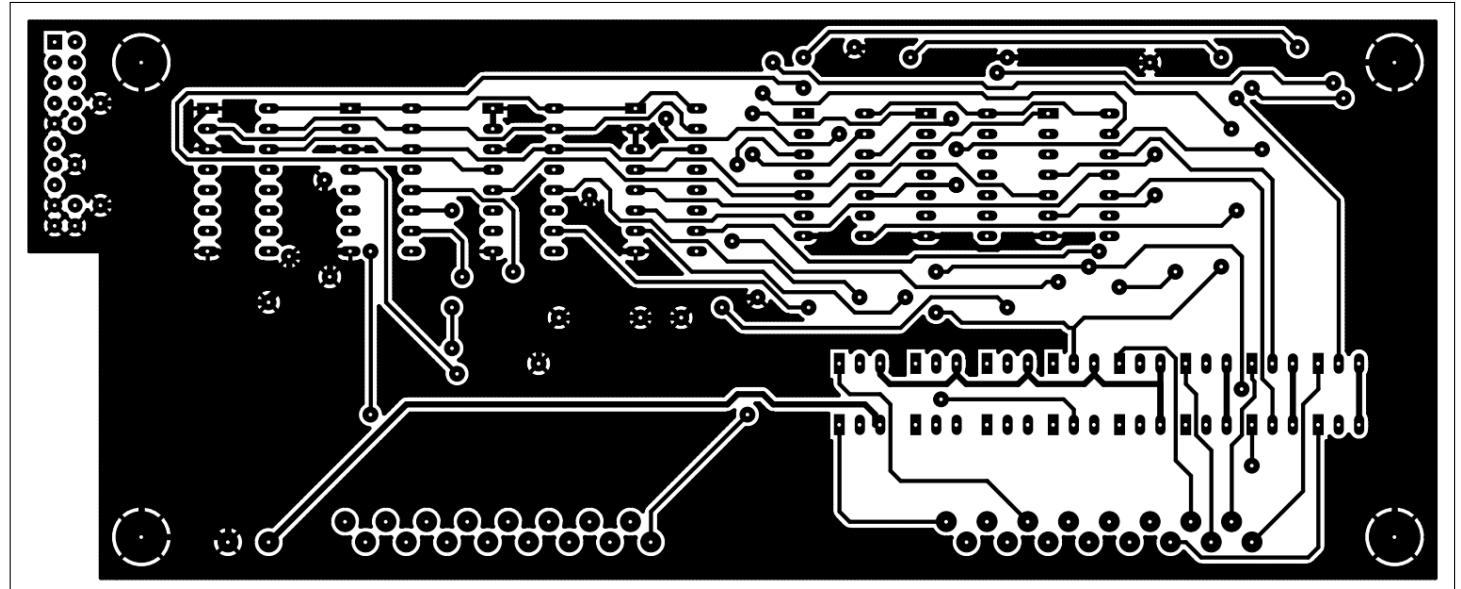


FIGURE 19 – Silk Screen TOP

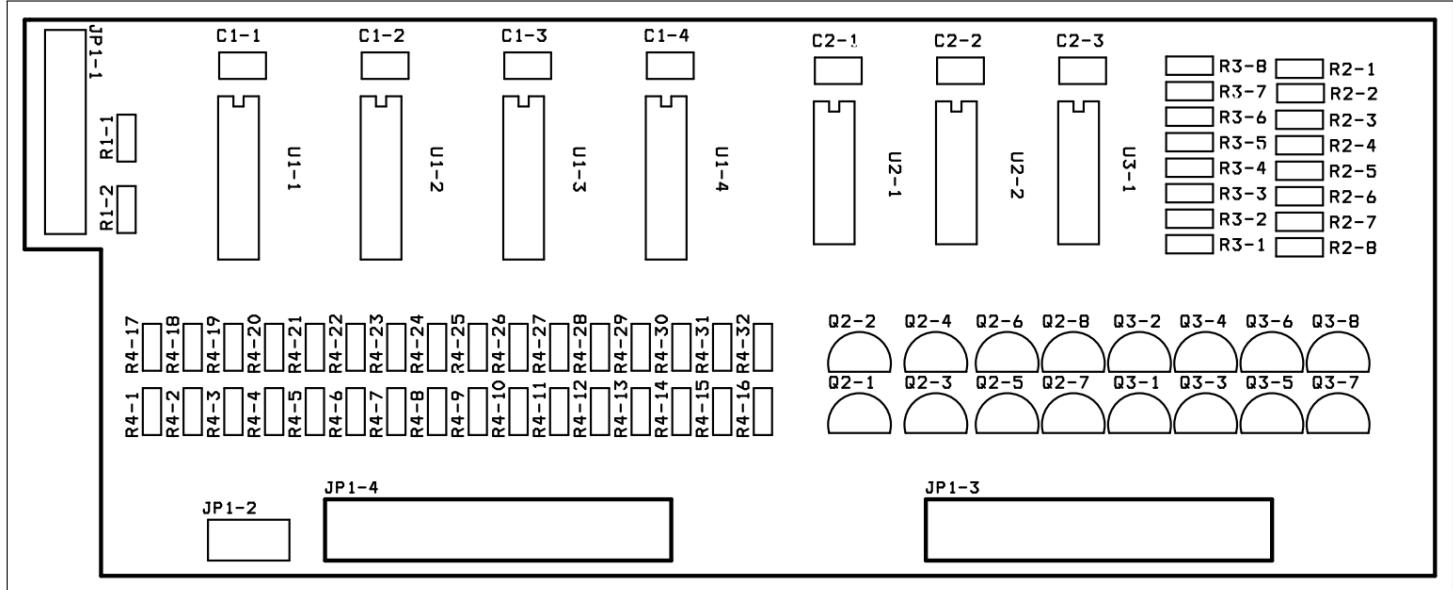


FIGURE 20 – Solder mask TOP

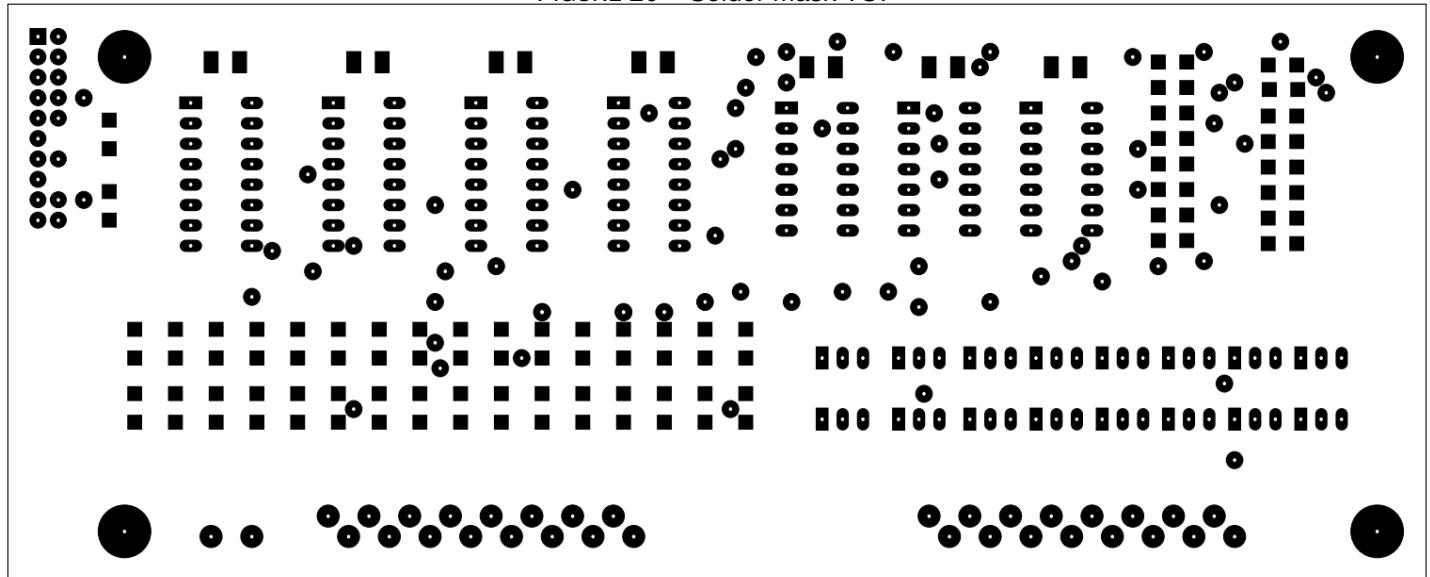


FIGURE 21 – Drill

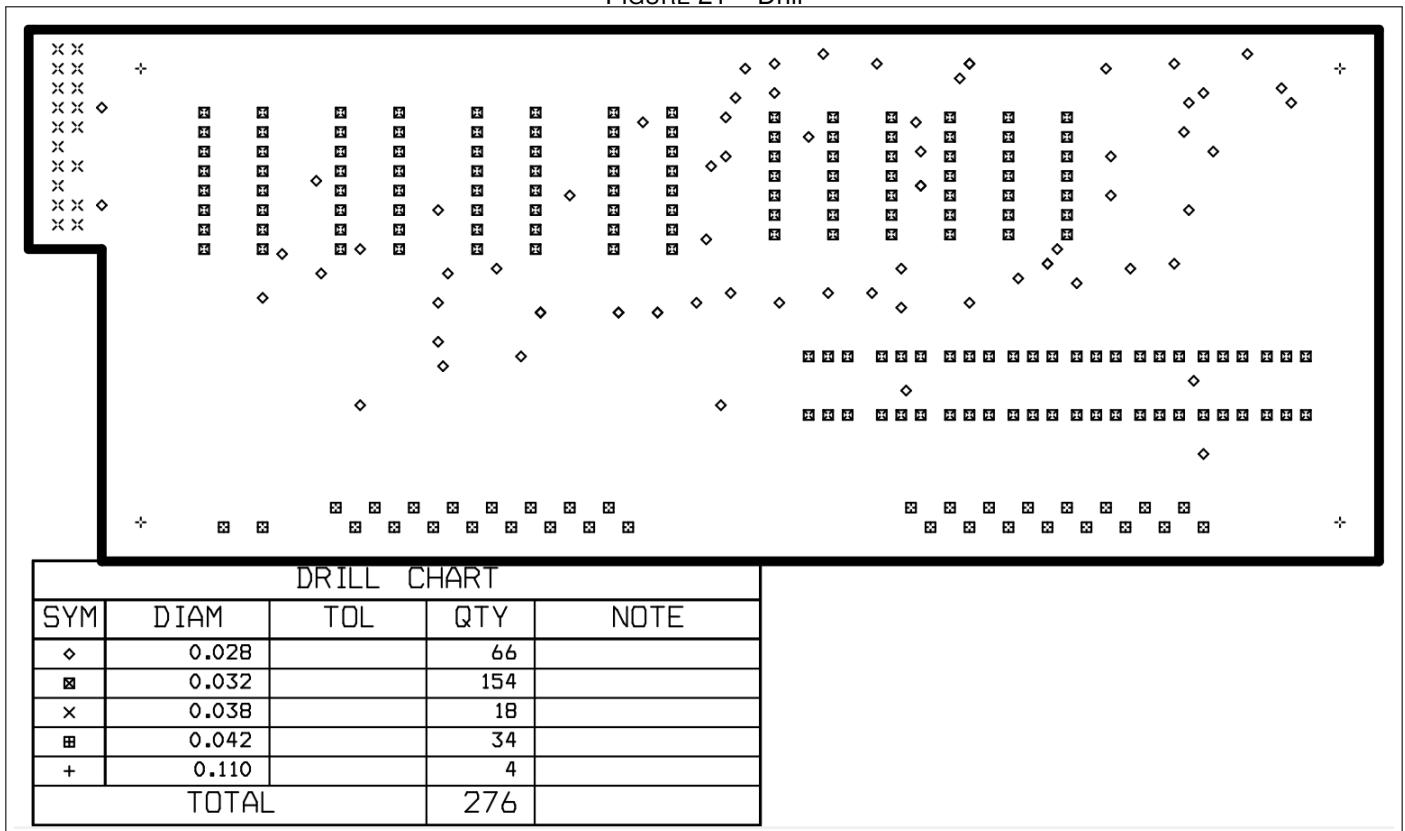
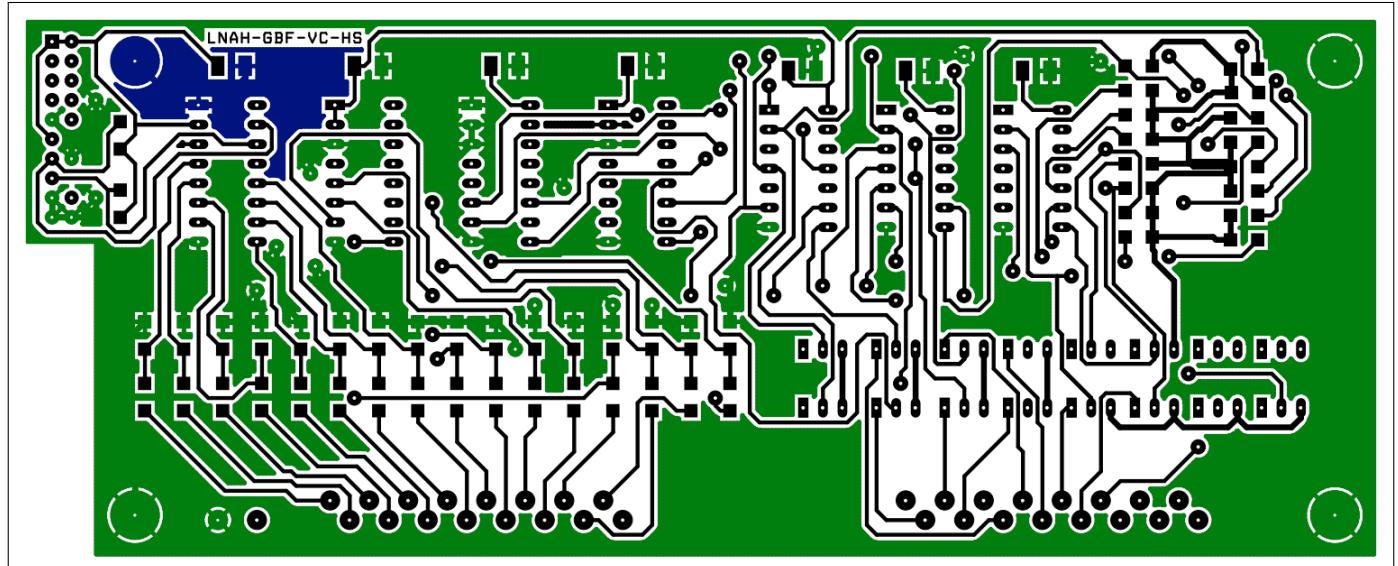


FIGURE 22 – Correctif



Il est à noter que nous avons dû réaliser un correctif lors de la soudure de la carte IO du projet. La zone de ground plane en bleu devait être reliée à GND, c'est-à-dire la zone verte, mais ce n'était pas le cas. C'est pourquoi un bout de fil a été soudé afin de pailler à ce manque. De plus, quelques ponts de soudure sont manifestés là et là, grillant de ce fait nos PCF8574. Rien d'insurmontable.

10 Conclusions

10.1 Ce que le projet m'a apporté

10.1.1 Vincent

dd

10.1.2 Hicham

dd

10.1.3 Gabriel

dd

10.1.4 Louis-Norman

dd

10.2 Difficultés et corrections

10.2.1 Vincent

ss

10.2.2 Hicham

ss

10.2.3 Gabriel

ss

10.2.4 Louis-Norman

ss

10.3 Ce que j'ai aimé ou pas

10.3.1 Vincent

a

10.3.2 Hicham

a

10.3.3 Gabriel

a

10.3.4 Louis-Norman

a

11 ANNEXE 1 : Code source du programme pour PC

```
using System;
```

12 ANNEXE 2 : Code source du Bolide et de la station 1

```
using System;
```

13 ANNEXE 4 : Code source de la table FESTO

```
using System;
```

14 ANNEXE 5 : Code source du programme PIC

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

15 ANNEXE 4 : Script Shell du SOC8200

prog.sh

```
compteur=0
```

read.sh

```
using System;
```

tcp.sh

```
using System;
```

can.sh

```
using System;
```

PortSerie.sh

```
while true
do
    head -1 /dev/ttysCMA0 > ./junk
    echo "Allo" > ./hbeat
done
```

RxCAN.sh

```
candump can0 >> ./histocan &
nbligne=1
ancienvar=0

while true
do
    var='tail -1 ./histocan'
    if [ "$var" != "$ancienvar" ]
    then
        echo $var
        echo $var >> ./histocandate
        date >> ./histocandate
        ancienvar=$var
    fi
# nbligne='wc -l ./histocan'
done
```

MachinChouette.sh

```
using System;
```

Bonus

*54 · 43. $\vdash .\alpha, \beta \in 1. \supset: \alpha \cap \beta = \Lambda. \equiv .\alpha \cup \beta \in 2$

Dem.

$$\begin{aligned}
 & \vdash . * 54 \cdot 26. \supset \vdash .\alpha = \iota'x.\beta = \iota'y. \supset: \alpha \cup \beta \in 2. \equiv .x \neq y. \\
 & [\ast 51 \cdot 231] \quad \equiv .\iota'x \cap \iota'y = \Lambda. \\
 & [\ast 13 \cdot 12] \quad \equiv .\alpha \cap \beta = \Lambda \tag{1} \\
 & \vdash .(1). * 11 \cdot 11 \cdot 35. \supset \\
 & \quad \vdash: .(\exists x, y).\alpha = \iota'x.\beta = \iota'y. \supset: \alpha \cup \beta \in 2. \equiv .\alpha \cap \beta = \Lambda \tag{2} \\
 & \vdash .(2). * 11 \cdot 54. * 52 \cdot 1. \supset \vdash .Prop
 \end{aligned}$$

From this proposition it will follow, when arithmetical addition has been defined, that $1 + 1 = 2$.

Cette démonstration mathématique prouve hors de tout doute que $1 + 1 = 2$.
Si les mathématiques sont vrais, alors notre projet devrait aller.