



**Cégep Limoilou**

ÉLECTRONIQUE PROGRAMMABLE ET ROBOTIQUE

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

## Projet de 5<sup>e</sup> session

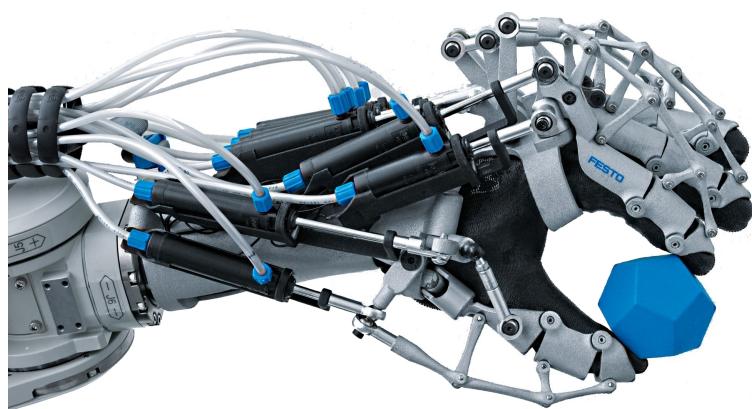
---

### Étudiants :

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

### Professeurs :

Ali Tadli  
Alain Champagne  
Stéphane Deschênes  
Étienne Tremblay



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

11 décembre 2014

## Table des matières

<b>1 Présentation du projet</b>	<b>5</b>
1.1 Schéma bloc d'ensemble du système . . . . .	5
1.1.1 Schéma bloc du bolide . . . . .	6
1.1.2 Schéma bloc de la table FESTO . . . . .	7
1.1.3 Schéma bloc du SOC8200 . . . . .	8
1.1.4 Schéma bloc de la station de pesée . . . . .	9
1.1.5 Schéma bloc de la station no.1 . . . . .	10
1.2 Liste des logiciels . . . . .	11
<b>2 Fonctionnement matériel</b>	<b>12</b>
2.1 Bloc 1 . . . . .	12
2.2 Bloc 2 . . . . .	12
2.3 Bloc 3 . . . . .	12
2.4 Bloc 4 . . . . .	12
<b>3 Explication des liens de communication entre les blocs</b>	<b>13</b>
3.1 RS232 . . . . .	13
3.2 Xbee . . . . .	13
3.3 CAN . . . . .	13
<b>4 Les trames expliquées en détail</b>	<b>14</b>
4.1 RS232 . . . . .	14
4.2 CAN . . . . .	14
<b>5 Liste des pièces</b>	<b>16</b>
5.1 Liens web . . . . .	16
5.2 Datasheets . . . . .	17
<b>6 Schémas de toutes les cartes et circuits utilisés dans le projet</b>	<b>19</b>
6.1 Schémas OrCAD de la carte d'extension . . . . .	20
6.2 Fichiers Gerbers de la carte d'extension . . . . .	24
6.3 Autres Schémas OrCAD . . . . .	27
<b>7 Interface PC</b>	<b>28</b>
7.1 Gestion de l'historique . . . . .	30
7.1.1 Exemple d'historique typique . . . . .	30
<b>8 Logiciel du SOC8200</b>	<b>31</b>
8.1 Description du programme . . . . .	31
8.2 Schéma bloc du script shell . . . . .	31
8.3 Gestion des processus et du temps de CPU . . . . .	31
8.4 Format et récupération des logs . . . . .	31
8.5 Liste des tests et logiciels . . . . .	31
<b>9 Logiciel du bolide et de la station uPSD</b>	<b>32</b>
9.1 Logiciel du module PIC18F258 . . . . .	32
9.2 Logiciel de la table FESTO . . . . .	32
9.3 Schéma des héritages de classes . . . . .	33
9.4 Procédure de compilation sur IAR . . . . .	34
9.5 Procédure de vérification . . . . .	34

<b>10 Logiciel du module PIC18F258</b>	<b>35</b>
10.1 Description du fonctionnement du programme . . . . .	35
10.2 Procédure de compilation sur MPLAB . . . . .	35
10.3 Procédure de vérification . . . . .	35
<b>11 Calculs</b>	<b>36</b>
11.1 Calcul du pas de conversion de la pile . . . . .	36
11.2 Calcul du baudrate . . . . .	36
<b>12 Conclusions</b>	<b>37</b>
12.1 Ce que le projet m'a apporté . . . . .	37
12.1.1 Vincent . . . . .	37
12.1.2 Hicham . . . . .	37
12.1.3 Gabriel . . . . .	37
12.1.4 Louis-Normand . . . . .	37
12.2 Difficultés et corrections . . . . .	37
12.2.1 Vincent . . . . .	37
12.2.2 Hicham . . . . .	37
12.2.3 Gabriel . . . . .	37
12.2.4 Louis-Norman . . . . .	37
12.3 Ce que j'ai aimé ou pas . . . . .	38
12.3.1 Vincent . . . . .	38
12.3.2 Hicham . . . . .	38
12.3.3 Gabriel . . . . .	38
12.3.4 Louis-Norman . . . . .	38
<b>13 ANNEXE 1 : Code source du programme pour PC</b>	<b>39</b>
<b>14 ANNEXE 2 : Code source du Bolide et de la station 1</b>	<b>40</b>
<b>15 ANNEXE 4 : Code source de la table FESTO</b>	<b>41</b>
<b>16 ANNEXE 5 : Code source du programme PIC</b>	<b>42</b>
<b>17 ANNEXE 4 : Script Shell du SOC8200</b>	<b>43</b>

## Table des figures

1	Vue d'ensemble du projet . . . . .	5
2	Schéma bloc du Bolide . . . . .	6
3	Schéma bloc de la table FESTO . . . . .	7
4	Schéma bloc du SOC8200 . . . . .	8
5	Schéma bloc du Bolide . . . . .	9
6	Schéma bloc du Bolide . . . . .	10
7	Schémas carte connecteur IO I2C, page 1 . . . . .	20
8	Schémas carte connecteur IO I2C, page 2 . . . . .	21
9	Schémas carte connecteur IO I2C, page 3 . . . . .	22
10	Schémas carte Iconnecteur O I2C, page 4 . . . . .	23
11	Couche TOP . . . . .	24
12	Couche BOT . . . . .	24
13	Silk Screen TOP . . . . .	25
14	Solder mask TOP . . . . .	25
15	Drill . . . . .	26
16	Programme de contrôle principal . . . . .	28
17	Options CAN avancées . . . . .	29
18	Historique des actions . . . . .	30
19	Héritage de la classe de contrôle du véhicule . . . . .	33
20	Qui hérite du SPI ? . . . . .	33
21	Qui hérite de l'I2C ? . . . . .	33
22	Héritage de la classe de contrôle de la station no.1 . . . . .	33
23	Choix de la cible sur IAR . . . . .	34
24	Compiler avec MicroC PRO pour PIC . . . . .	35

## Liste des tableaux

1	Index des identifiants matériel CAN . . . . .	14
2	Liste des trames CAN . . . . .	14

# 1 Présentation du projet

**Interface utilisateur contrôlant une station de chargement de blocs, une station de communication, un véhicule de livraison et une station d'acquisition.**

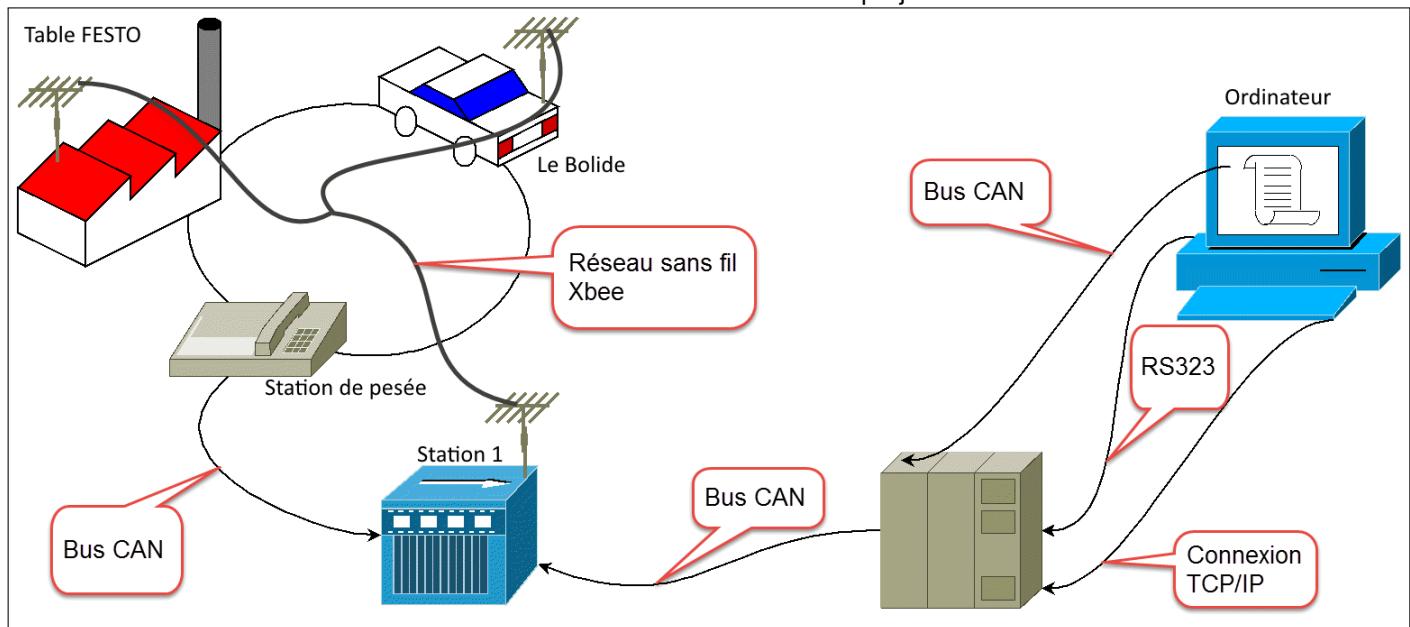
Nous devons réaliser l'automatisation d'un système industriel constitué d'une station de communication, d'une station de chargement, d'un véhicule de livraison, et d'un système embarqué, le tout, contrôlé à l'aide d'une interface utilisateur réalisée sur un ordinateur de type PC.

## Principales composantes :

- ⇒ 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

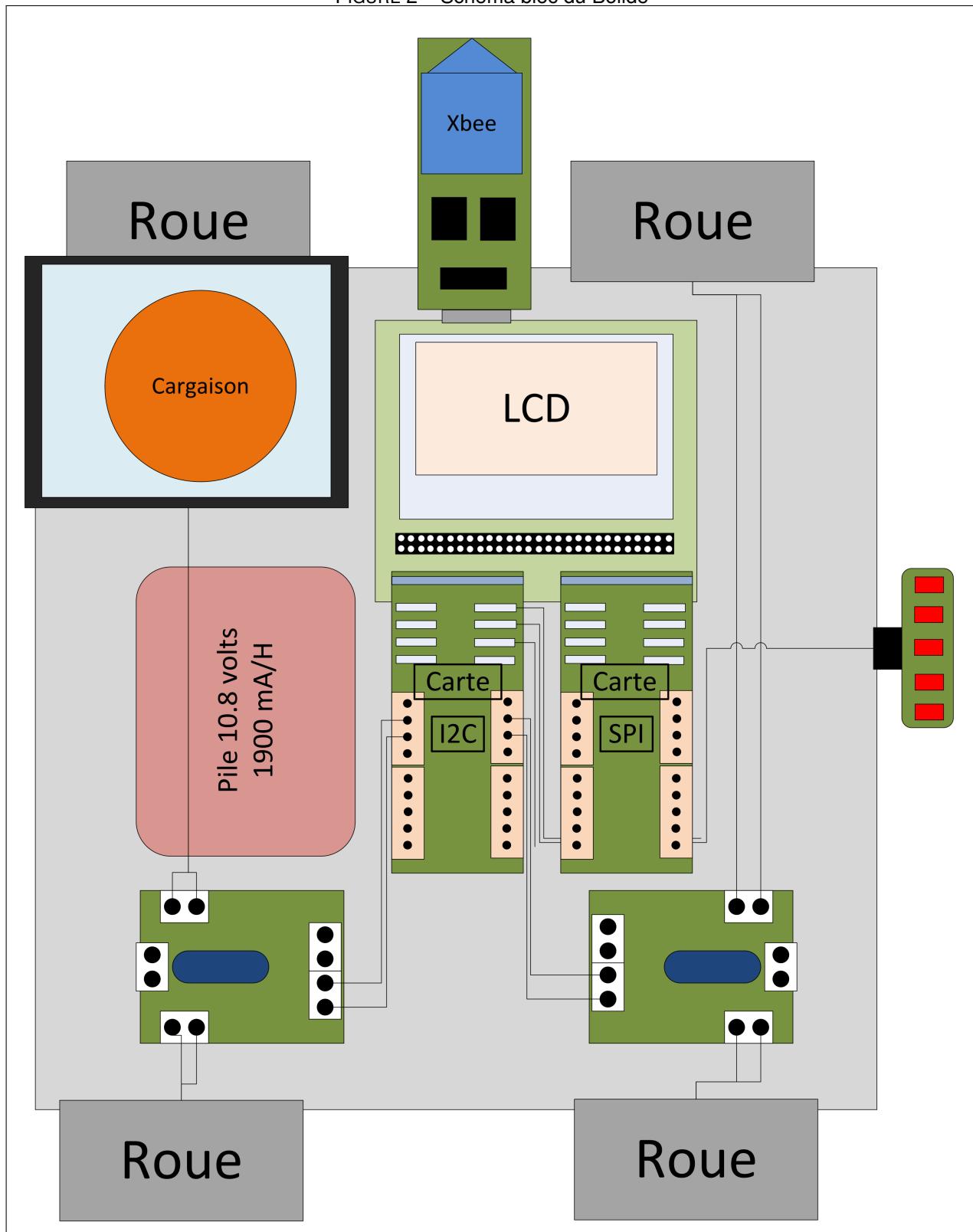
## 1.1 Schéma bloc d'ensemble du système

FIGURE 1 – Vue d'ensemble du projet



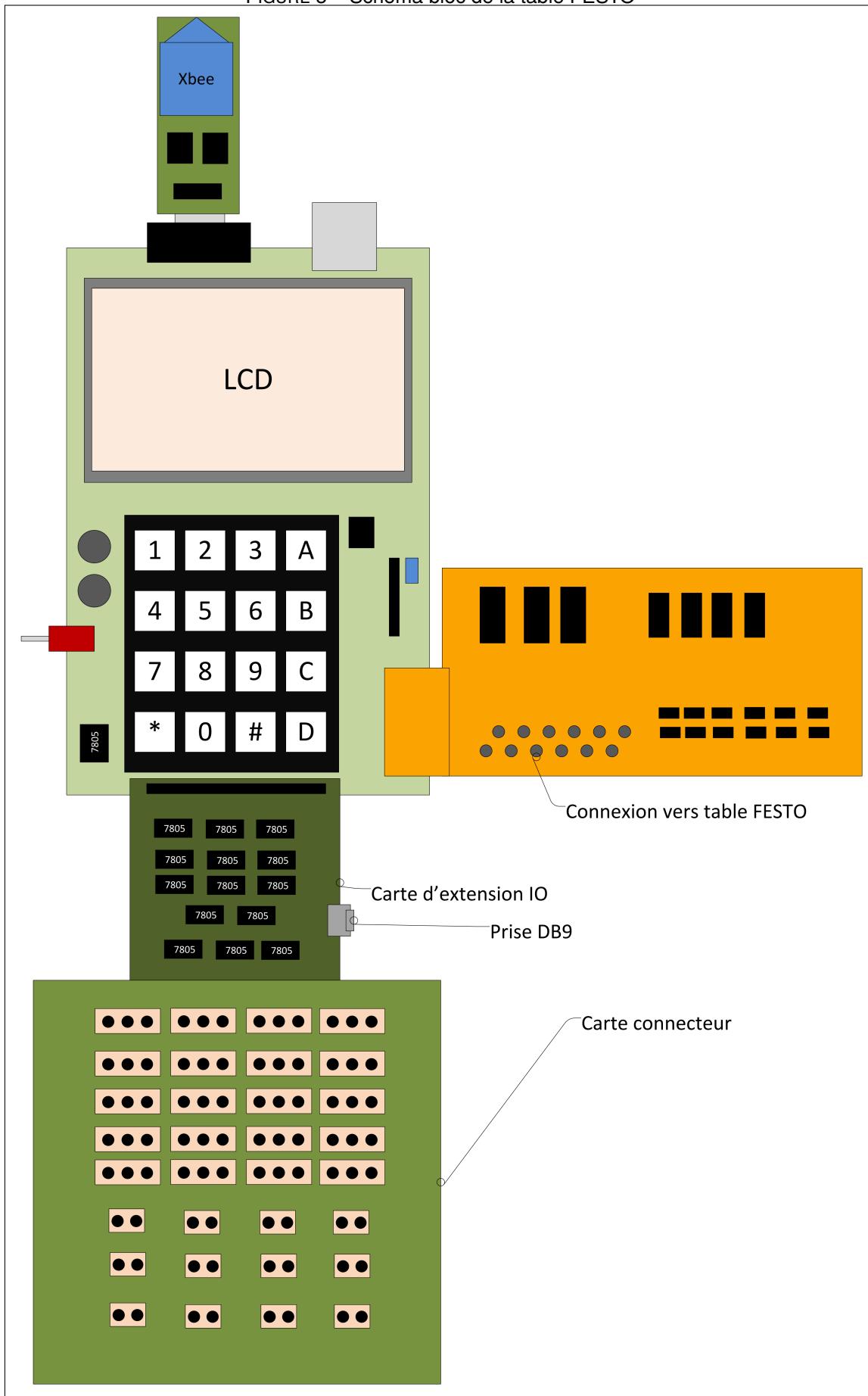
### 1.1.1 Schéma bloc du bolide

FIGURE 2 – Schéma bloc du Bolide



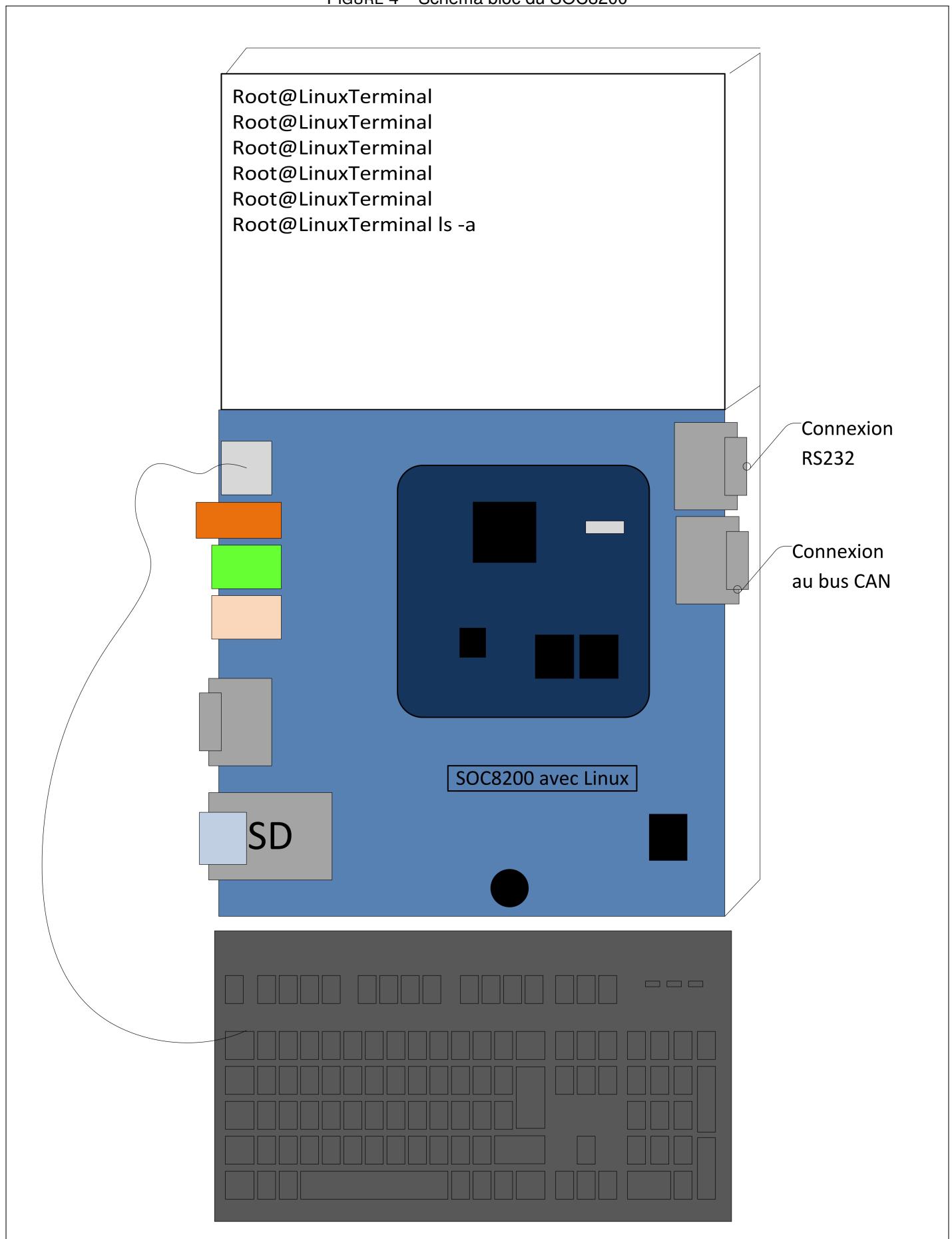
### 1.1.2 Schéma bloc de la table FESTO

FIGURE 3 – Schéma bloc de la table FESTO



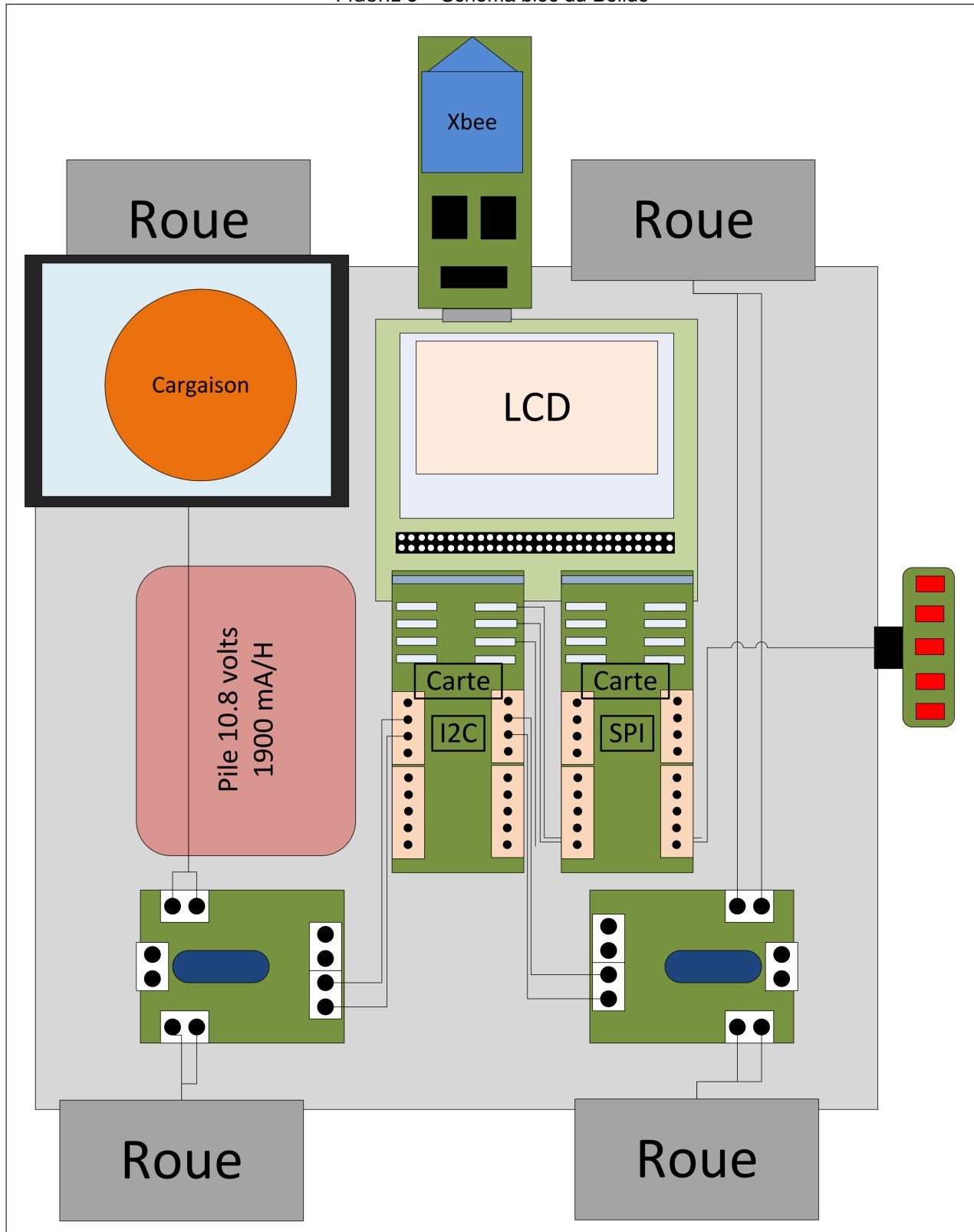
### 1.1.3 Schéma bloc du SOC8200

FIGURE 4 – Schéma bloc du SOC8200



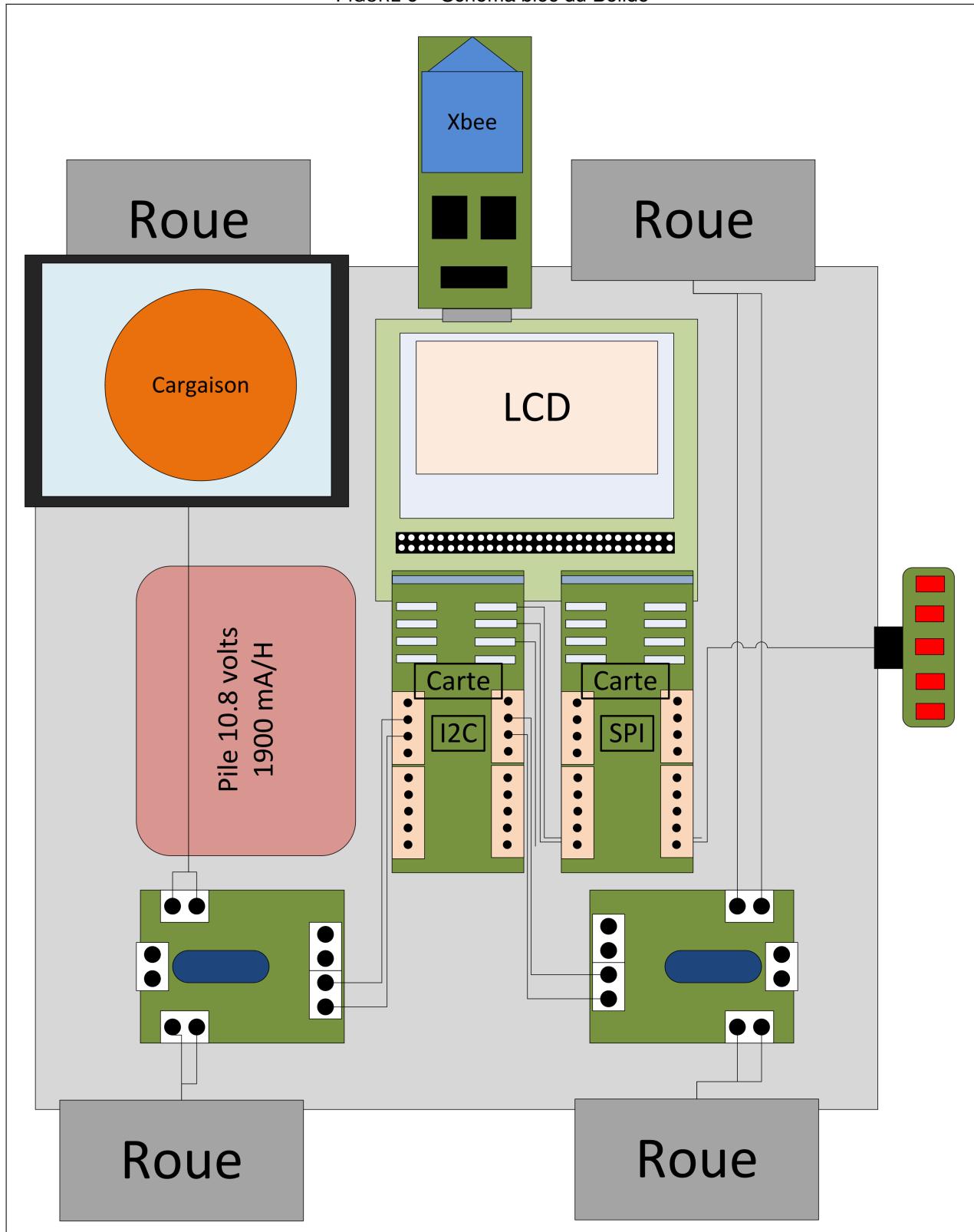
#### 1.1.4 Schéma bloc de la station de pesée

FIGURE 5 – Schéma bloc du Bolide



### 1.1.5 Schéma bloc de la station no.1

FIGURE 6 – Schéma bloc du Bolide



## 1.2 Liste des logiciels

---

<b>Terminaux</b>	<b>Compilateurs et IDE</b>	<b>Éditeurs de texte</b>
<ul style="list-style-type: none"><li>• UART Master 0.97</li><li>• Serializ3r 1.0.2</li><li>• TerraTerm</li><li>• Putty</li><li>• GTKterm 0.99.7-rc1</li><li>• Terminator</li><li>• CAPS</li><li>• tinyBootloader</li></ul>	<ul style="list-style-type: none"><li>• Visual Studio 2013</li><li>• Visual Studio 2010</li><li>• IAR 8.20</li><li>• MPLAB X version 6.00</li></ul>	<ul style="list-style-type: none"><li>• Notepad++</li><li>• BowPad</li><li>• medit 1.2.0</li></ul>
<b>Schémas électriques</b>	<b>Éditeurs de texte</b>	<b>Autres</b>
<ul style="list-style-type: none"><li>• OrCAD 16.2</li></ul>	<ul style="list-style-type: none"><li>• Notepad++</li><li>• BowPad</li><li>• medit 1.2.0</li></ul>	<ul style="list-style-type: none"><li>• VMWare Workstation 10</li><li>• TeXmaker 4.3</li><li>• X-ctu for Xbee</li><li>• Dukto R6</li><li>• Dia</li><li>• Microsoft Visio 2013</li><li>• Festo configuration tool</li><li>• Saleae Logic Analiser</li><li>• L<sup>A</sup>T<sub>E</sub>X(avec plug-in Doxygen)</li></ul>
<b>Gestionnaires de projet</b>	<b>Systèmes d'exploitation</b>	
<ul style="list-style-type: none"><li>• MS Project 2010</li><li>• Git Hub</li></ul>	<ul style="list-style-type: none"><li>• Windows 7 SP1</li><li>• Windows 8.1</li><li>• Windows XP SP3</li><li>• Arch Linux</li></ul>	

---

Mettre des photos ici...

## 2 Fonctionnement matériel

### 2.1 Bloc 1

D'un point de vue matériel, le bloc 1 est le plus simple, car il est composé d'un ordinateur Windows muni 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 composante à l'aide d'une application en C#. 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 consignes<sup>1</sup> 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 pesée<sup>2</sup>, de la table FESTO<sup>3</sup> et du bolide<sup>4</sup>. Ces informations sont systématiquement retransmises au PC via le bus CAN. Quant au bolide, il doit pouvoir fonctionner aussi bien en SPI qu'en I2C, il doit s'arrêter aux bon endroits, choisir son sens de rotation et tenir le système informé de ses moindres faits et gestes.

### 2.3 Bloc 3

Le Bloc 3 est composé de la table FESTO (incluant toutes ses pompes et ses capteurs) et d'une carte uPSD. Sur la carte uPSD est branché une carte connecteur PCF8570 que nous avons réalisé avec OrCAD, ainsi qu'une carte connecteur 5-24 volts et une carte d'extension avec convertisseurs DAC ADC. De plus, un Xbee est connecté au uPSD afin de relier ce dernier (et la table FESTO) au reste du montage. C'est cette station qui gère les opérations de la table FESTO, de la lecture des capteurs à l'activation des pompes en passant par la gestion du moteur pas à pas.

### 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'informations 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. De plus, le bloc 4 comporte une carte PIC (mettre modèle) et d'une balance (mettre modèle.) La balance pèse le poids de la cargaison et envoie l'information en RS232 à la carte PIC qui se charge de convertir la donnée au format CAN avant de l'expédier au PC.

---

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

### 3 Explication des liens de communication entre les blocs

#### 3.1 RS232

Un lien RS232 de 9600 bauds est établi entre l'ordinateur et le SOC8200. Ce lien sert à l'envoi et à la réception d'un HeartBeat, afin que le SOC8200 ou l'ordinateur soit informé de toute défaillance de l'autre. Un autre lien RS232, à XYZ bauds cette fois, relie la carte PIC-Machin à la balance. Ainsi, le PIC est informé de tout ce qui est pesé sur la balance.

#### 3.2 Xbee

Trois Xbee sont présents sur le système, un sur le bolide, un autre sur la table FESTO (Bloc 3) et le dernier sur la carte uPSD du Bloc 2. Dès que l'un des Xbee envoie une information, les deux autres la reçoivent systématiquement. La station du bloc 2 fait office de centralisateur en réalisant deux actions :

1. Transformer les données Xbee du bolide et de la table FESTO en trame CAN à destination du PC
2. Transformer les directives CAN du PC en directives Xbee intelligibles pour le bolide et la table FESTO.

#### 3.3 CAN

à écrire

## 4 Les trames expliquées en détail

### 4.1 RS232

Le protocole RS232 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" tous deux les HeartBeats, et si ces derniers ne leur parviennent pas, chaque dispositif tient pour acquis que l'autre est hors service et prend la relève de la gestion du bus CAN.

### 4.2 CAN

Chaque composante matérielle, 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 et de trois octets de TimeStamp. Seule exception à ce système, la pesée du bloc, qui ne dispose d'aucun octet d'identification, et c'est pourquoi nous filtrons l'ID émetteur afin de détecter l'ID matériel de la station de pesée (005). Quant à sa trame de 5 octets, elle se lit comme suit :

DIZAINE UNITÉ POINT DIXIÈME CENTIÈME.

TABLE 1 – Index des identifiants matériel CAN

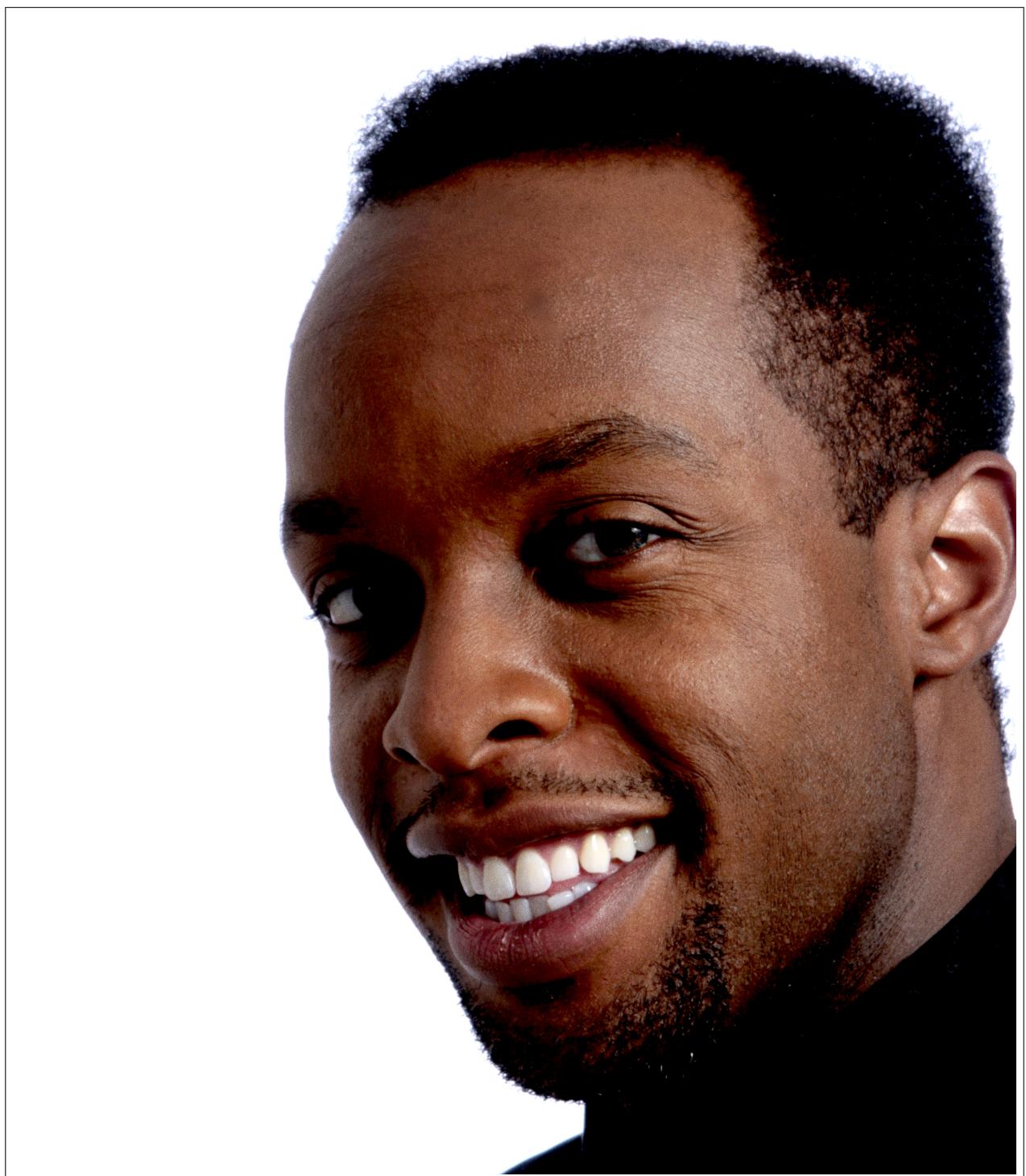
Device	ID matériel
Ordinateur	000
SOC8200	001
Station 1	002
Station 2	003
Véhicule	004
Station de pesée	005

Note sur les octets :

TS = TimeStamp ; HH = heure ; MM = minute ; SS = seconde ; [00-FF]= valeur allant de 0 à 255

TABLE 2 – Liste des trames CAN

Émetteur	Fonctionnalité	Trame CAN
Ordinateur	Démarrer le véhicule	08 00 TS TS TS 00 00
Ordinateur	Arrêter le véhicule	08 01 TS TS TS 00 00
Véhicule	Dit : je suis arrêté	01 00 TS TS TS 00 00
Véhicule	Dit : j'avance	01 01 TS TS TS 00 00
Véhicule	Dit : je suis hors circuit	01 02 TS TS TS 00 00
Véhicule	Dit : sa vitesse	02 [00-FF] TS TS TS 00 00
Véhicule	Dit : le niveau de sa batterie	03 [00-FF] TS TS TS 00 00
Table FESTO	Bloc = métal	04 00 TS TS TS 00 00
Table FESTO	Bloc = orange	04 01 TS TS TS 00 00
Table FESTO	Bloc = noir	04 02 TS TS TS 00 00
Table FESTO	La voiture est à la table FESTO	07 01 TS TS TS 00 00
Station de pesée	La voiture est à la station de pesée	07 00 TS TS TS 00 00
Ordinateur	Synchronisation temporelle	06 00 HH MM SS 00 00
Ordinateur	Demande le log au SOC8200	2B 1E TS TS TS 00 00
Ordinateur	Exige le sens horaire	08 00 TS TS TS 00 00
Ordinateur	Exige le sens anti horaire	08 01 TS TS TS 00 00
Ordinateur	Active la table FESTO	09 00 TS TS TS 00 00



Bon, **Hicham**, maintenant que j'ai ton attention, j'aimerais simplement te dire ce qu'il reste à faire :

Il faut réaliser un schéma câblage de la station FESTO et du véhicule

Trouver toutes les datasheets et leurs URL

Écrire tes 3 conclusions

Faire le schéma bloc de la station de pesée

## 5 Liste des pièces

---

### Cartes, kits et câbles

- 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
  - SaberTooth motor drive 2x5
- 

### Composantes, pièces et puces (liste à compléter)

- PCF8574
  - DAC65574
  - MCP2515
  - 74HC07
  - BS250
  - MCP2515
  - MCP2515
- 

### 5.1 Liens web

<http://datasheets.maximintegrated.com/en/ds/DS89C430-DS89C450.pdf>  
[http://www.datasheetcatalog.com/datasheets\\_pdf/U/P/S/D/UPSD3254.shtml](http://www.datasheetcatalog.com/datasheets_pdf/U/P/S/D/UPSD3254.shtml)  
[http://www.datasheetcatalog.com/datasheets\\_pdf/D/A/C/6/DAC6574.shtml](http://www.datasheetcatalog.com/datasheets_pdf/D/A/C/6/DAC6574.shtml)  
[http://www.datasheetcatalog.com/datasheets\\_pdf/L/M/3/9/LM3914.shtml](http://www.datasheetcatalog.com/datasheets_pdf/L/M/3/9/LM3914.shtml)  
[http://www.datasheetcatalog.net/datasheets\\_pdf/P/C/F/8/PCF8574.shtml](http://www.datasheetcatalog.net/datasheets_pdf/P/C/F/8/PCF8574.shtml)  
[http://www.datasheetcatalog.com/datasheets\\_pdf/O/P/T/1/OPT101.shtml](http://www.datasheetcatalog.com/datasheets_pdf/O/P/T/1/OPT101.shtml)  
<http://stackoverflow.com/>  
<http://stackoverflow.com/>

## 5.2 Datasheets

Note : Toutes les datasheets sont en format non-compressé. Vous pouvez zoomer sur le document PDF<sup>5</sup> afin de lire l'intégralité de leurs premières pages.

74HC14

Pinout Schematic		Product Specification					
		74HHC1T14					
<b>Hex Inverting Schmitt trigger</b>							
<b>GENERAL DESCRIPTION</b>							
The 74HHC1T14 are high-speed肖特基CMOS devices and are pin-compatible with the power Schottky TTL 74LS14. They have a low input current and a fast switching speed. The 74HHC1T14 provide six inverting buffers with Schmitt-trigger action. They are capable of transforming a changing digital signal into sharply defined, square output signals.							
<b>LOGIC SYMBOL</b>							
							
<b>SYMBOL</b>							
Input symbol:  A1-A6		Conditions		TYPICAL			
Input threshold: V <sub>IN(H)</sub> = 15 V, V <sub>DD</sub> = 5 V, T = 25°C		HC HCT		UNIT			
V <sub>IN(H)</sub> = 1.6 V		22 ns		ns			
V <sub>IN(L)</sub> = 0.8 V		32 ns		ns			
Output threshold: V <sub>OUT(H)</sub> = 15 V, V <sub>DD</sub> = 5 V, T = 25°C		2.0 V		V			
<b>Notes</b>							
1. C <sub>in</sub> is used to determine the dynamic power dissipation (P <sub>d</sub> ) in mW. P <sub>d</sub> = K <sub>1</sub> (V <sub>DD</sub> - V <sub>IN</sub> ) + K <sub>2</sub> (V <sub>DD</sub> - V <sub>OUT</sub> ) + K <sub>3</sub> where: K <sub>1</sub> = Input leakage current K <sub>2</sub> = Output load capacitance C <sub>in</sub> = Input load capacitance V <sub>DD</sub> = 5 V, V <sub>IN</sub> = 0.8 V, V <sub>OUT</sub> = 1.6 V 2. For HC conditions V <sub>DD</sub> = 3.6 V For HCT the condition V <sub>DD</sub> = 5 V and V <sub>DD</sub> = 15 V							
<b>ORDERING INFORMATION</b>							
See "74HHC Logic Devices Logic Package Information".							

## LineSensors

### Registers:

You can read the EC line sensor by issuing an EC read of the programmed address.

A single byte representing the status of the sensor will be returned, ranging from 0 to 31, where

0 = sensor off and 31 = sensor on.

	Address	Sensor	Address (A)	Address (B)	Address (C)	Address (D)	Address (E)
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	1
0	1	0	0	0	0	0	2
1	0	1	0	0	0	0	3
0	1	0	1	0	0	0	4
1	0	1	0	1	0	0	5
0	1	1	0	0	0	0	6
1	1	1	0	0	0	0	7
0	0	0	1	0	0	0	8
1	0	0	1	0	0	0	9
0	1	0	1	0	0	0	10
1	1	0	1	0	0	0	11
0	0	1	1	0	0	0	12
1	0	1	1	0	0	0	13
0	1	1	1	0	0	0	14
1	1	1	1	0	0	0	15
0	0	0	0	1	0	0	16
1	0	0	0	1	0	0	17
0	1	0	0	1	0	0	18

PCF8573

DAC6574

**DATAACQUISITION**

**INSTRUMENTATION**

**QUAD, 10-BIT, LOW-POWER, VOLTAGE OUTPUT,  
INTERFACE DIGITAL-TO-ANALOG CONVERTER**

---

FEATURES	DESCRIPTION
<ul style="list-style-type: none"> <li>• Bi-directional Operation: 500 µA ±15 V I<sub>in</sub></li> <li>• Fast Update Rates: 140 SPS</li> <li>• Four Analog Outputs: 0.01% Accuracy</li> <li>• Power-On Reset to Zero</li> <li>• Dual Power Sources: ±15 V</li> <li>• 16-bit Resolution</li> <li>• FC™ Interface for PC</li> <li>• Direct Memory Access</li> <li>• On-chip Digital Buffer Amplifier, Rail-to-Rail</li> <li>• Dual Differential Inputs</li> <li>• Analog-to-Digital Input for Four D/A6484</li> <li>• Synchronous Update Support for up to 16</li> <li>• Operation From +5VDC to 195°C</li> <li>• Immune to ESD Protection</li> </ul>	<p>The D/A6484 is a quad, 10-bit, low-power, voltage output, digital-to-analog converter. It has four analog outputs, each with a 16-bit resolution, 0.01% accuracy, and a 140 SPS update rate. The device is designed for use in data acquisition systems, instrumentation, and control applications. The D/A6484 takes an 8-bit digital input and converts it to a 10-bit analog output. It features a rail-to-rail output buffer amplifier and a direct memory access (DMA) interface mode with address support up to 16 bit. The device is powered by ±15 V power supplies.</p> <p>The D/A6484 uses a 16-bit digital-to-analog converter (DAC) with a 10-bit resolution. The DAC is followed by a current source circuit that converts the digital code into an analog current. This current is then converted into an analog voltage using a transconductance amplifier. The device also includes a digital-to-digital converter (DDC) and a digital-to-analog converter (DAC). The D/A6484 is designed to be used in a variety of applications, such as data converters with FC interface, data DACs, and 16-bit DACs.</p>
<ul style="list-style-type: none"> <li>• Process Control</li> <li>• Data Acquisition Systems</li> <li>• Clockless Serial Device Control</li> <li>• Peripherals</li> <li>• Portable Instrumentation</li> </ul>	<p>The low power consumption of the part is normal for a 10-bit DAC. The power consumption is dependent on the operating conditions, such as supply voltage, operating frequency, and digital input. The power consumption is approximately 10 mW at 100 SPS and 15 V supply voltage.</p> <p>It offers the ability of data converters with FC interface, data DACs, and 16-bit DACs. The D/A6484 is a quad, 10-bit, low-power, voltage output, digital-to-analog converter. It has four analog outputs, each with a 16-bit resolution, 0.01% accuracy, and a 140 SPS update rate. The device is designed for use in data acquisition systems, instrumentation, and control applications. The D/A6484 takes an 8-bit digital input and converts it to a 10-bit analog output. It features a rail-to-rail output buffer amplifier and a direct memory access (DMA) interface mode with address support up to 16 bit. The device is powered by ±15 V power supplies.</p>

## LM3914

uPSD3254A

DS89C450

OPT101

## 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 Semiconductor		Product Specification				
Hex Inverting Schmitt trigger		74HC/HCT14				
<b>FEATURES</b>						
• Output capability: standard						
• I <sub>OL</sub> : category SII						
<b>GENERAL DESCRIPTION</b>						
The 74HC/HCT14 are hex inverting Schmitt triggers. The devices are pin compatible with the lower threshold TTL/HTL. These devices are designed to be fully compatible with the standard CMOS logic families.						
The 74HC/HCT14 provide six inverting buffers with Schmitt-trigger action. They are capable of functioning as dual inverter buffers or as three input AND or OR gates, through the use of multiple inputs.						
<b>QUICK REFERENCE INFORMATION</b>						
AND ( $I_{OL} = 25 \mu A$ ), OR ( $I_{OL} = 1.5 \mu A$ )						
<b>SYMBOL</b>	<b>PARAMETER</b>		<b>CONDITION</b>		<b>TYPICAL</b>	
$I_{OL}$ ( $I_{OL} = 25 \mu A$ )	$I_{OL}$ ( $I_{OL} = 1.5 \mu A$ )		$I_{OL} = 15\% I_{OL}$ , $T_{J} = 25^{\circ}C$		$12$	
<b>L</b>	<b>Input impedance</b>		$T_{J} = 25^{\circ}C$		$10$	
<b>C</b>	<b>Output load capacitance per gate</b>		$V_{DD} = 5V$ , $T_{J} = 25^{\circ}C$		$3.2$	
<b>Notes</b>	<b>Notes</b>		<b>Notes</b>		<b>Notes</b>	
1. $I_{OL} = 25 \mu A$ is the output current for standard outputs ( $I_{OL} > 0$ ).	1. $I_{OL} = 25 \mu A$ is the output current for standard outputs ( $I_{OL} > 0$ ).		1. $I_{OL} = 25 \mu A$ is the output current for standard outputs ( $I_{OL} > 0$ ).		1. $I_{OL} = 25 \mu A$ is the output current for standard outputs ( $I_{OL} > 0$ ).	
2. Input leakage is $0.1 \mu A$ .	2. Input leakage is $0.1 \mu A$ .		2. Input leakage is $0.1 \mu A$ .		2. Input leakage is $0.1 \mu A$ .	
3. $I_{OL}$ is total output capacitance in pF.	3. $I_{OL}$ is total output capacitance in pF.		3. $I_{OL}$ is total output capacitance in pF.		3. $I_{OL}$ is total output capacitance in pF.	
4. $I_{OL} = 1.5 \mu A$ is the output current for low voltage outputs.	4. $I_{OL} = 1.5 \mu A$ is the output current for low voltage outputs.		4. $I_{OL} = 1.5 \mu A$ is the output current for low voltage outputs.		4. $I_{OL} = 1.5 \mu A$ is the output current for low voltage outputs.	
5. $I_{OL} = 1.5 \mu A$ is the output current for standard outputs ( $I_{OL} < 0$ ).	5. $I_{OL} = 1.5 \mu A$ is the output current for standard outputs ( $I_{OL} < 0$ ).		5. $I_{OL} = 1.5 \mu A$ is the output current for standard outputs ( $I_{OL} < 0$ ).		5. $I_{OL} = 1.5 \mu A$ is the output current for standard outputs ( $I_{OL} < 0$ ).	
<b>ORDERING INFORMATION</b>	<b>See <a href="#">74HC/HCT14 Standard Logic Package Information</a></b>					

## LineSensors

### Registers:

You can read the ADC input sensor by issuing an ADC read of the programmed address. A single byte representing the value of the four sensors will be returned, ranging from 0 to 31, where 0 means 4 sensors are valid and 31 means all sensors are black.

Sensor 1	Sensor 2	Sensor 3	Sensor 4	Sensor (S)	Decimal Number
0	0	0	0	0	1
1	0	0	0	0	2
0	1	0	0	0	3
1	1	0	0	0	4
0	0	1	0	0	5
1	0	1	0	0	6
0	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
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15
0	0	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 contains a 74HC164 shift register followed by a 'Data Register'. This feeds into a 'Control Logic' section containing a 74HC148 priority encoder and a 74HC138 decoder. The control logic also includes a 'Clock Generator' and a 'Power Down' section with a 74HC138. Below the control logic is a 'Digital-to-Analog' section featuring a 74HC4511 DAC. This section includes a 'Digital Filter' (74HC221) and a 'Digital-to-Analog' converter (74HC4511). The DAC output is connected to a 'Sample-and-Hold' section (74HC125) and a 'Filter' (74HC4508). The final output stage consists of a 'Driver' (74HC4040) and a 'Buffer' (74HC4040).

LM3914

 <b>TEXAS INSTRUMENTS</b> <hr/> <b>LML914 DotBar Display Driver</b> <i>Low Cost, High Resolution Dot Matrix Display</i>	<b>LML914</b> <hr/> <a href="http://www.ti.com/lm914">http://www.ti.com/lm914</a>
<hr/> <p><b>FEATURES</b></p> <ul style="list-style-type: none"> <li>• Drives 16x16 LCDs or vacuum fluorescent displays up to 16x24 pixels at 8 bits per pixel.</li> <li>• 16-bit parallel data bus for fast data access by user.</li> <li>• Operates in a variety of modes:</li> <ul style="list-style-type: none"> <li>• Internal voltage reference from 2.0V to 3.6V.</li> <li>• Operation with +5V power source.</li> <li>• Input voltage range from 3.0V to 3.9V.</li> <li>• Input current requirement of less than 10mA.</li> <li>• Output current requirement of less than 10mA.</li> <li>• Output current programmable from 2mA to 10mA.</li> </ul> </ul> <hr/> <p><b>DESCRIPTION</b></p> <p>The LML914 is a low cost, high resolution dot matrix display driver. It can drive 16x16 LCDs, providing a broad analog display, a single point change the display from a driving 16x24 at 8 bits per pixel. Unlike other dot matrix display drivers, the LML914 has a built-in 16-bit parallel data bus which allows the user to interface directly with the host system from 3.0V to 3.9V.</p> <p>This device is designed to be used with standard 8-bit microprocessors and acoustic substrate drivers. The high-resolution input buffer accepts signals down to 2.0V, yet it can accept voltages of 3.0V or above. The output buffer can also accept voltages down to 2.0V, yet it can tolerate voltages of 3.6V or above. This means that the LML914 can be held to typical 3.0V, even with a wide temperature range.</p> <p>Compatibility is provided with standard 8-bit microprocessors, as well as with the Zilog Z80 and Z8000. Standard and augmented bus structures are easily added on the display system. The LML914 can drive LCDs of many colors, or monochrome (infrared). The LML914 can also be used with vacuum fluorescent displays. The LML914 is a high resolution dot matrix display driver, and supports 16x16 pixels. It can support up to 16x24 pixels, if required. The LML914 can be driven by a 16-bit parallel data bus, or a 16-bit serial data bus.</p> <p>In the 16x16 mode, there is a small amount of overlap ("fence" time) between segments. This means that the LML914 can be used with a variety of dot matrix displays, such as LCDs and VFDs.</p> <p>Most of the display drivers found in the market today are addressed by logic controlled currents. Various effects can be achieved by controlling these currents. The individual outputs can be driven as well as all 16 outputs. The LML914 is a high resolution dot matrix display driver, and can be used with a variety of dot matrix displays.</p> <p>The LML914 is rated for operation from 1.0V to +3.9V. The LM914H is available in a leadless surface mount package.</p> <p>The LML914 is a high resolution dot matrix display driver, addressing the requirements of reduced cost, and proper pricing for accurate operation, and leading cost reductions.</p> <hr/> <p><b>A</b> Before use, ensure that an authorized dealer is carrying the latest version of the data sheet, and use it in all applications of the device. Texas Instruments reserves the right to make changes to the design, structure, and/or performance of the device without notice, and is not responsible for typographical or printed errors that may appear in this document. © 1989 Texas Instruments Incorporated. All rights reserved. No part of this document may be reproduced, stored in a retrieval system, or transmitted, in whole or in part, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the prior written permission of Texas Instruments Incorporated.</p>	

uPSD3254A

DS89C450

 <b>DALLAS</b> <b>MAXIM</b> www.maxim-ic.com	<b>DS89C430/DS89C440/DS89C450</b> Ultra-High-Speed Flash Microcontrollers																														
<b>GENERAL DESCRIPTION</b> <p>The DS89C430, DS89C440, and DS89C450 are the highest performance, lowest power consumption microcontrollers. They feature next-generation 16-bit RISC architecture, 32KB of flash memory, and 128KB of SRAM. The DS89C430 is the fastest member of the family, executing 1.5 times faster than the original 8051 of the same core. The DS89C440 and DS89C450 offer a 1.5 speed improvement over the DS89C430. All three microcontrollers support a wide range of memory sizes from 128KB to 256KB. The DS89C430, DS89C440, and DS89C450 are pin-compatible with the 8051 and are fully compatible with the 8051 instruction set. The DS89C430, DS89C440, and DS89C450 are the first members of the MAXIM family to support the 8051 compatibility mode.</p> <p>The Ultra-High-Speed Flash Microcontroller family should be used in applications where fast processing time and low power consumption are required. Applications include real-time control, motion control, and industrial automation.</p>																															
<b>FEATURES</b> <ul style="list-style-type: none"> <li>• High-Speed 8051 Architecture</li> <li>• 16-Bit RISC Instruction Set</li> <li>• 128KB SRAM</li> <li>• 32KB Flash Program Memory</li> <li>• Optional Large Length 16KB+ Access</li> <li>• 16-Bit A/D</li> <li>• 10-Bit D/A</li> <li>• 12-Bit DAC</li> <li>• Dual Data Ports with Automatic Transfer</li> <li>• 16-Bit Timer with 100% Modulo</li> <li>• 10-Bit Counter with 100% Modulo</li> <li>• On-Chip Memory</li> <li>• Watchdog Timer</li> <li>• Power-On Reset</li> <li>• Low-Power Mode</li> <li>• 100% Pin-Compatible Through Serial Port</li> <li>• 1024x1024 ROM</li> </ul>																															
<b>ORDERING INFORMATION</b> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Part</th> <th>Description</th> <th>Package</th> </tr> </thead> <tbody> <tr> <td>DS89C430-100</td> <td>DS89C430 100-pin PLCC</td> <td>PLCC</td> </tr> <tr> <td>DS89C430-100T</td> <td>DS89C430 100-pin TQFP</td> <td>TQFP</td> </tr> <tr> <td>DS89C440-100</td> <td>DS89C440 100-pin PLCC</td> <td>PLCC</td> </tr> <tr> <td>DS89C440-100T</td> <td>DS89C440 100-pin TQFP</td> <td>TQFP</td> </tr> <tr> <td>DS89C450-100</td> <td>DS89C450 100-pin PLCC</td> <td>PLCC</td> </tr> <tr> <td>DS89C450-100T</td> <td>DS89C450 100-pin TQFP</td> <td>TQFP</td> </tr> <tr> <td>DS89C430-100G</td> <td>DS89C430 100-pin Gullwing</td> <td>Gullwing</td> </tr> <tr> <td>DS89C440-100G</td> <td>DS89C440 100-pin Gullwing</td> <td>Gullwing</td> </tr> <tr> <td>DS89C450-100G</td> <td>DS89C450 100-pin Gullwing</td> <td>Gullwing</td> </tr> </tbody> </table>		Part	Description	Package	DS89C430-100	DS89C430 100-pin PLCC	PLCC	DS89C430-100T	DS89C430 100-pin TQFP	TQFP	DS89C440-100	DS89C440 100-pin PLCC	PLCC	DS89C440-100T	DS89C440 100-pin TQFP	TQFP	DS89C450-100	DS89C450 100-pin PLCC	PLCC	DS89C450-100T	DS89C450 100-pin TQFP	TQFP	DS89C430-100G	DS89C430 100-pin Gullwing	Gullwing	DS89C440-100G	DS89C440 100-pin Gullwing	Gullwing	DS89C450-100G	DS89C450 100-pin Gullwing	Gullwing
Part	Description	Package																													
DS89C430-100	DS89C430 100-pin PLCC	PLCC																													
DS89C430-100T	DS89C430 100-pin TQFP	TQFP																													
DS89C440-100	DS89C440 100-pin PLCC	PLCC																													
DS89C440-100T	DS89C440 100-pin TQFP	TQFP																													
DS89C450-100	DS89C450 100-pin PLCC	PLCC																													
DS89C450-100T	DS89C450 100-pin TQFP	TQFP																													
DS89C430-100G	DS89C430 100-pin Gullwing	Gullwing																													
DS89C440-100G	DS89C440 100-pin Gullwing	Gullwing																													
DS89C450-100G	DS89C450 100-pin Gullwing	Gullwing																													
<small>1. Order quantities are in thousands. Minimum order quantity is 1000 units. Lead times are subject to change without notice. Actual lead times will depend on the number of units ordered and the delivery point.</small>																															
<b>APPLICATIONS</b> <ul style="list-style-type: none"> <li>• Data Logging</li> <li>• Telephones</li> <li>• Billing Energy</li> <li>• Industrial Control</li> <li>• White Goods</li> <li>• HVAC</li> <li>• Power Management</li> <li>• Motor Control</li> <li>• Vending</li> <li>• Programmable Logic Controller</li> <li>• Building Safety</li> <li>• Consumer Electronics</li> <li>• Industrial Sensors</li> <li>• Resource/Sensor</li> </ul>																															
<small>2. Actual device does not always have exactly the same pinouts as their 8051 counterparts due to minor dielectric thickness differences of each device. For more detailed pinouts of each device see the individual data sheet.</small>																															

OPT101

**MONOLITHIC PHOTODIODE AND SINGLE-SUPPLY TRANSIMPEDANCE AMPLIFIER**

---

FEATURES	DESCRIPTION
<ul style="list-style-type: none"> <li>• SINGLE SUPPLY +2.7V TO +50V</li> <li>• PHOTOELECTRICITY</li> <li>• INTERNAL 10-MEGOHM RESISTOR</li> <li>• HIGH RESPONSIVENESS (4-EA/W)</li> <li>• LOW DARK CURRENT (10<sup>-12</sup>A)</li> <li>• BULKY (1.5MM X 1.5MM X 0.5MM) SURFACE-MOUNT PACKAGES</li> </ul>	<p>The OPT110 is a monolithic photodiode with a compact surface-mount package. It is designed for use in low-light applications with light intensity. The optoelement is designed to operate in single or dual-supply voltages. The internal 10-Megohm resistor provides a balanced current source for the photodiode.</p> <p>The OPT110 is a combination of photodiode and transimpedance amplifier in a single chip die structure. The photodiode is a p-n junction with a thin aluminum oxide passivation layer. The transimpedance amplifier is designed for logic, timer, error, sense, pulse, and pin-pushing due to its low power consumption and low cost. The OPT110 is intended to be used in the photodiode mode for certain frequency and amplitude ranges.</p> <p>The OPT110 operates from +2.7V to +50V and can be used in a variety of applications such as medical, industrial, scientific, and automotive.</p>
<b>APPLICATIONS</b> <ul style="list-style-type: none"> <li>• METICAL INSTRUMENTATION</li> <li>• LANDSCAPE ILLUMINATION</li> <li>• POSITION AND POSITION SENSORS</li> <li>• PROXIMITY AND POSITION SENSORS</li> <li>• BARCODE READERS</li> <li>• SMOKE DETECTORS</li> <li>• CURRENT CHAMBERS</li> </ul>	

---

**SPECTRAL RESPONSE**

Wavelength (nm)

Intensity (a.u.)

## SaberTooth motor drive

## 6 Schémas de toutes les cartes et circuits utilisés dans le projet

**Arrrg, faut-il vraiment mettre tout ça ?**

## 6.1 Schémas OrCAD de la carte d'extension

FIGURE 7 – Schémas carte connecteur IO I2C, page 1

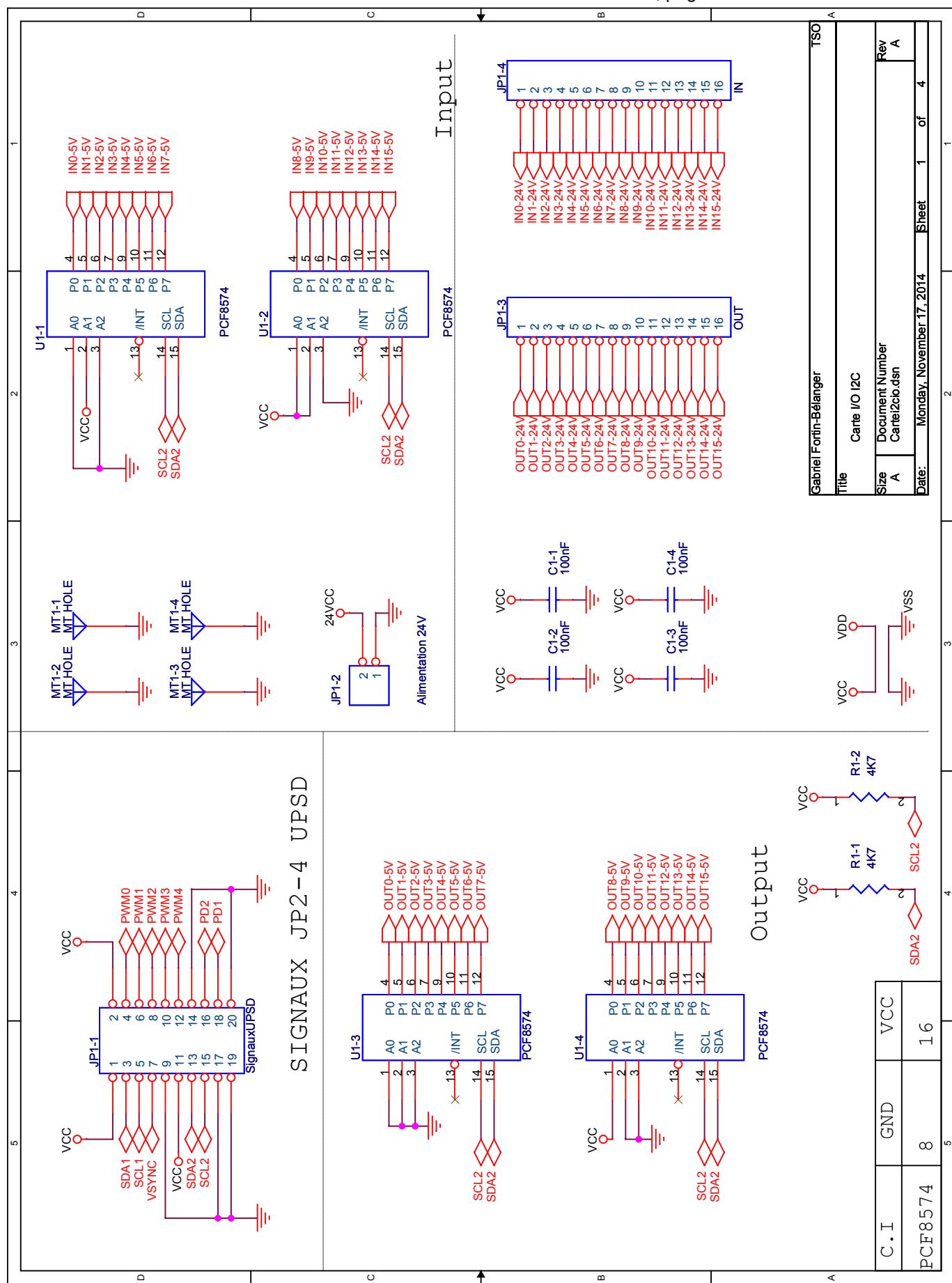


FIGURE 8 – Schémas carte connecteur IO I2C, page 2

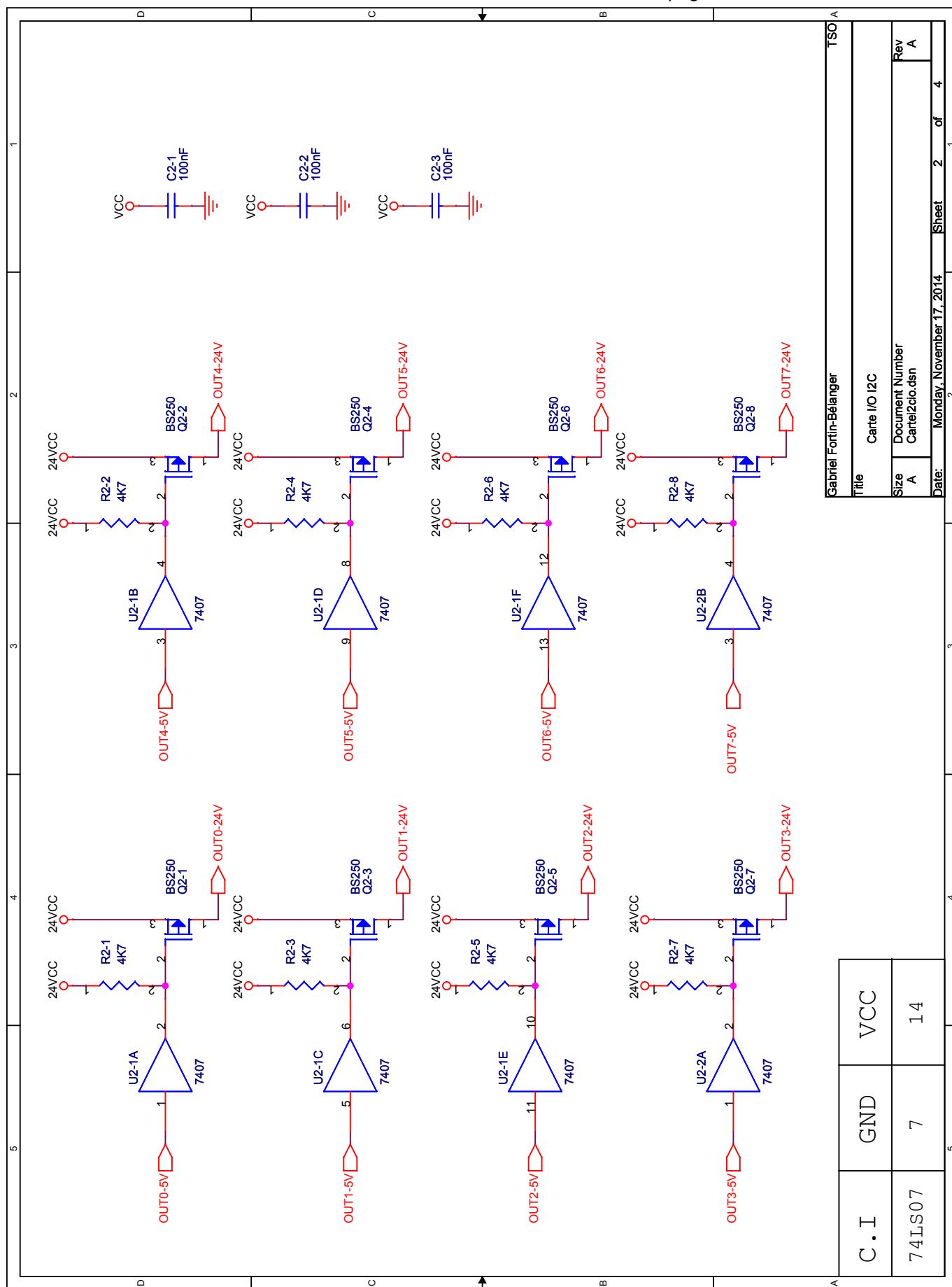


FIGURE 9 – Schémas carte connecteur IO I2C, page 3

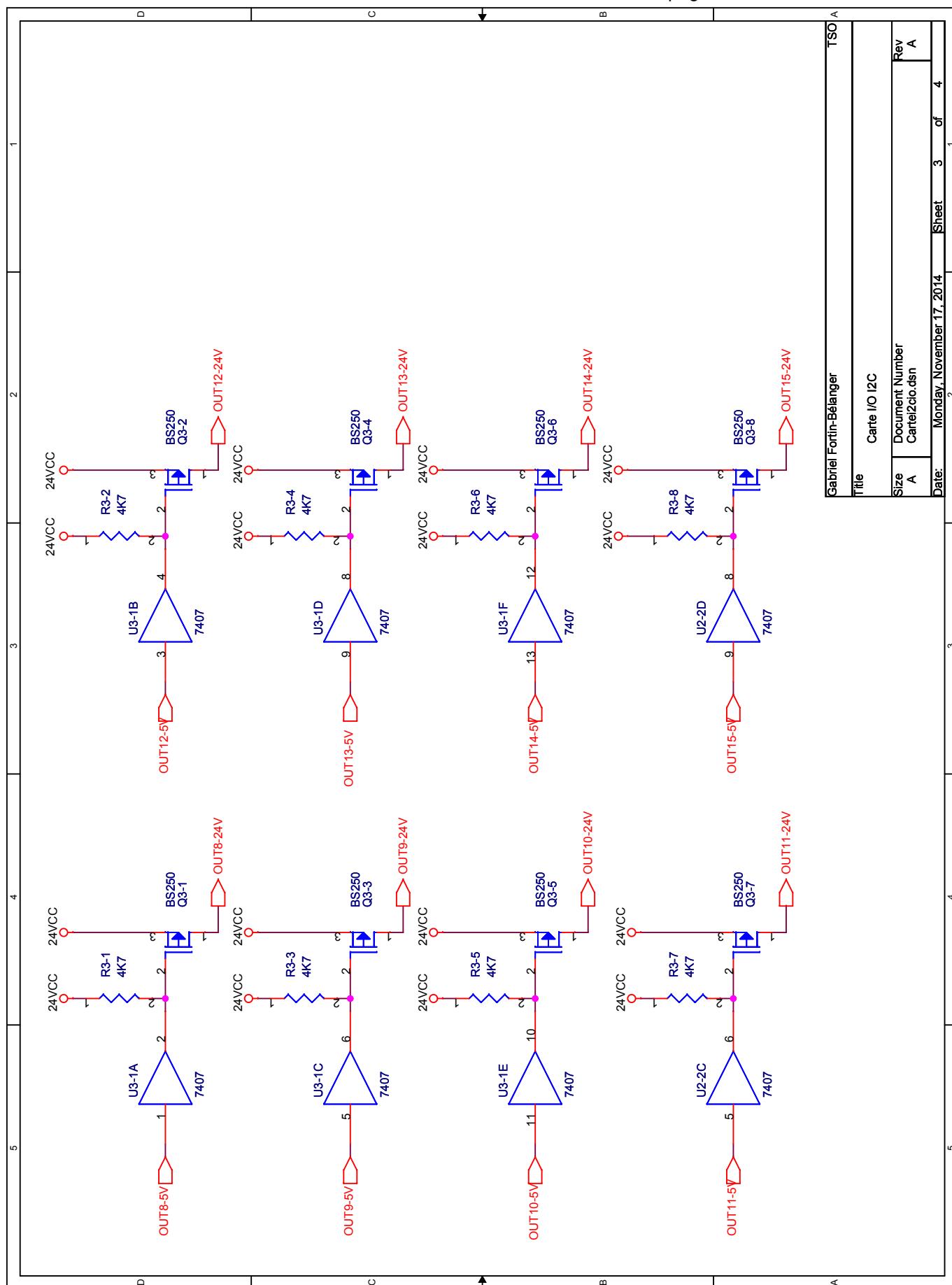
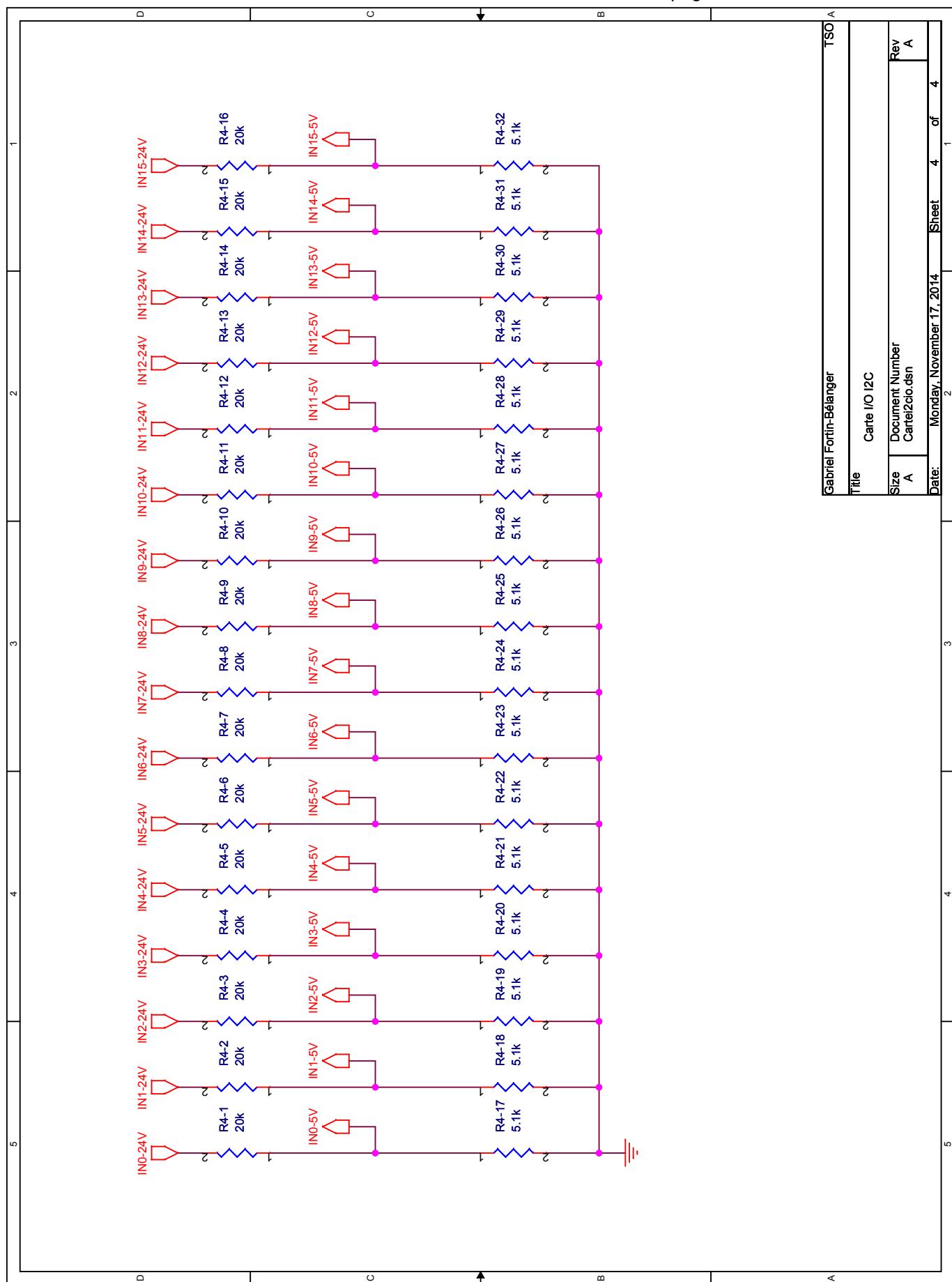


FIGURE 10 – Schémas carte Iconnecteur O I2C, page 4



## 6.2 Fichiers Gerbers de la carte d'extension

Une carte d'extension, dont voici les images GERBER<sup>6</sup>, à été réalisée avec OrCAD 16.2 et gravée à l'aide de la rutilante LPKF départementale.

FIGURE 11 – Couche TOP

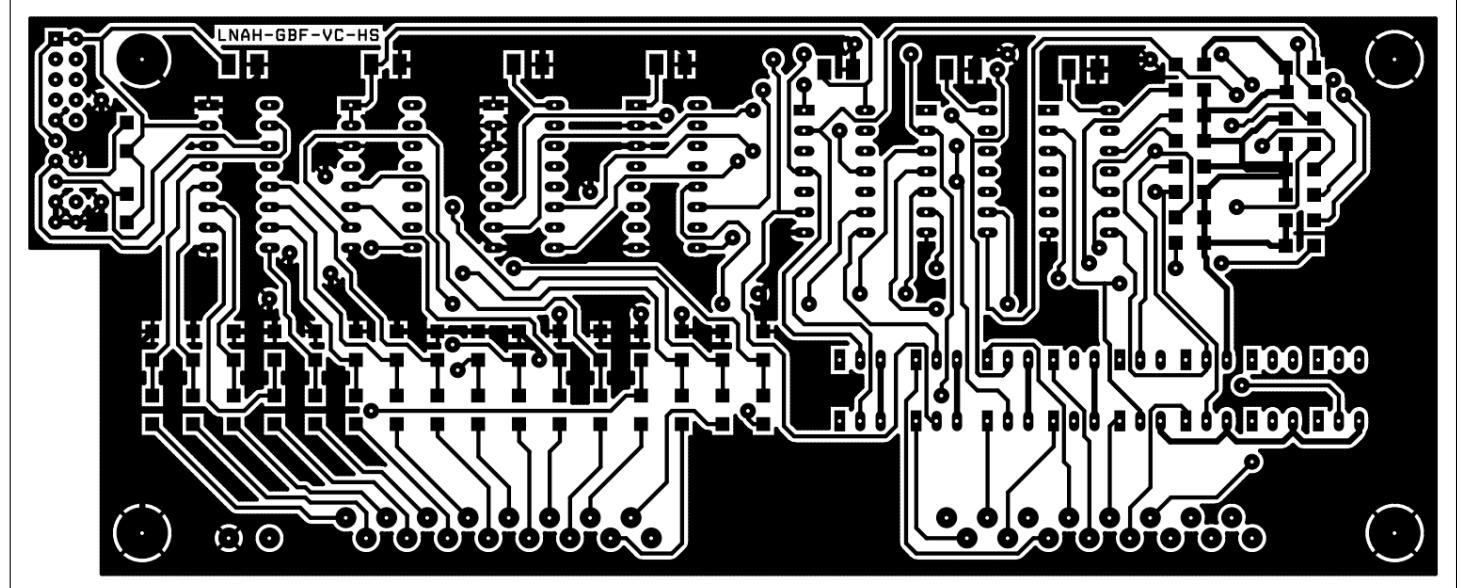


FIGURE 12 – Couche BOT

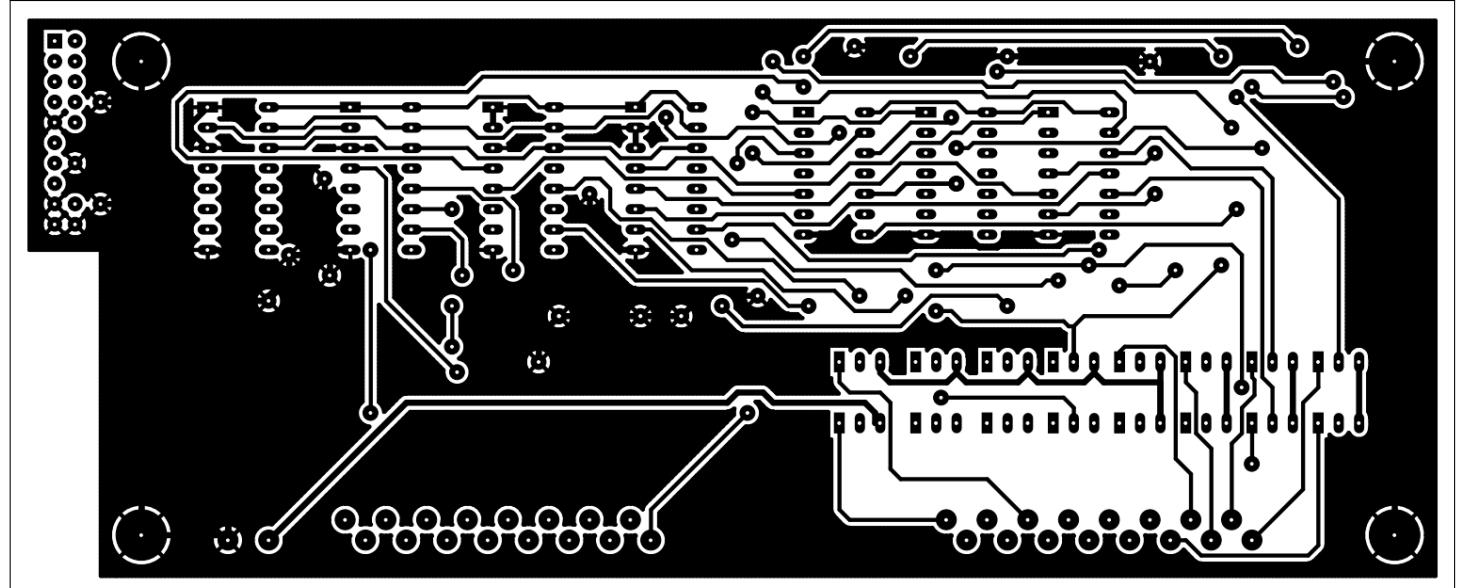


FIGURE 13 – Silk Screen TOP

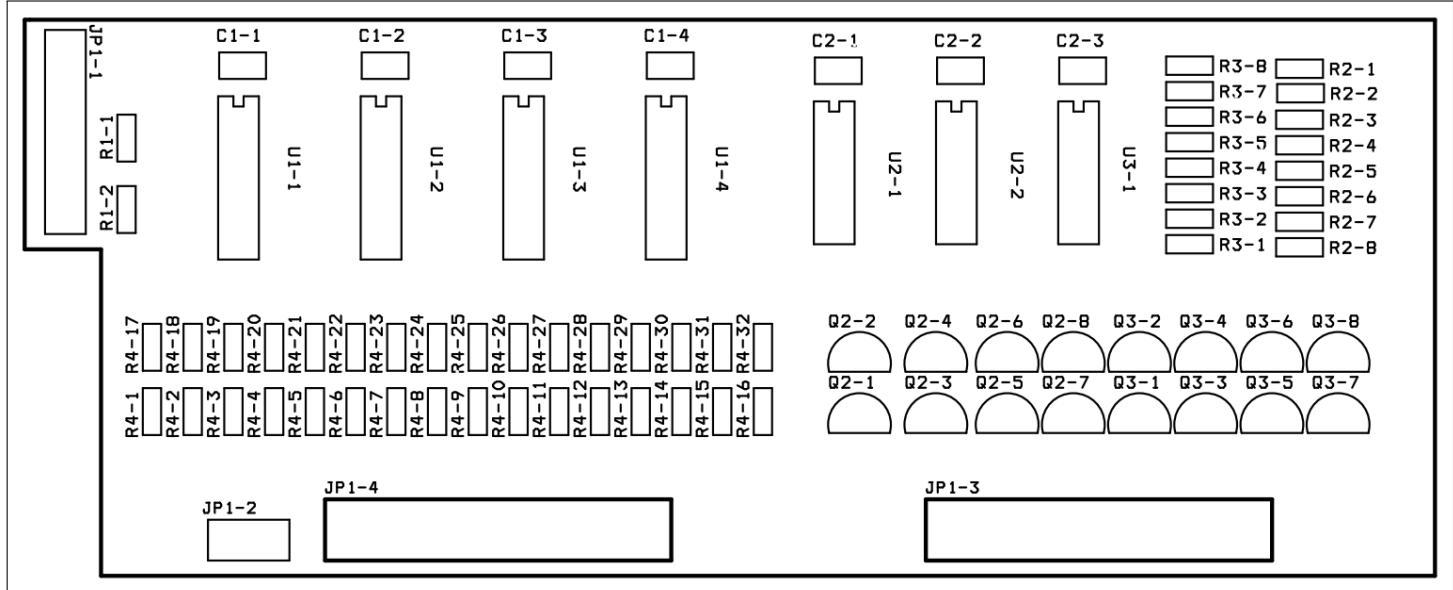


FIGURE 14 – Solder mask TOP

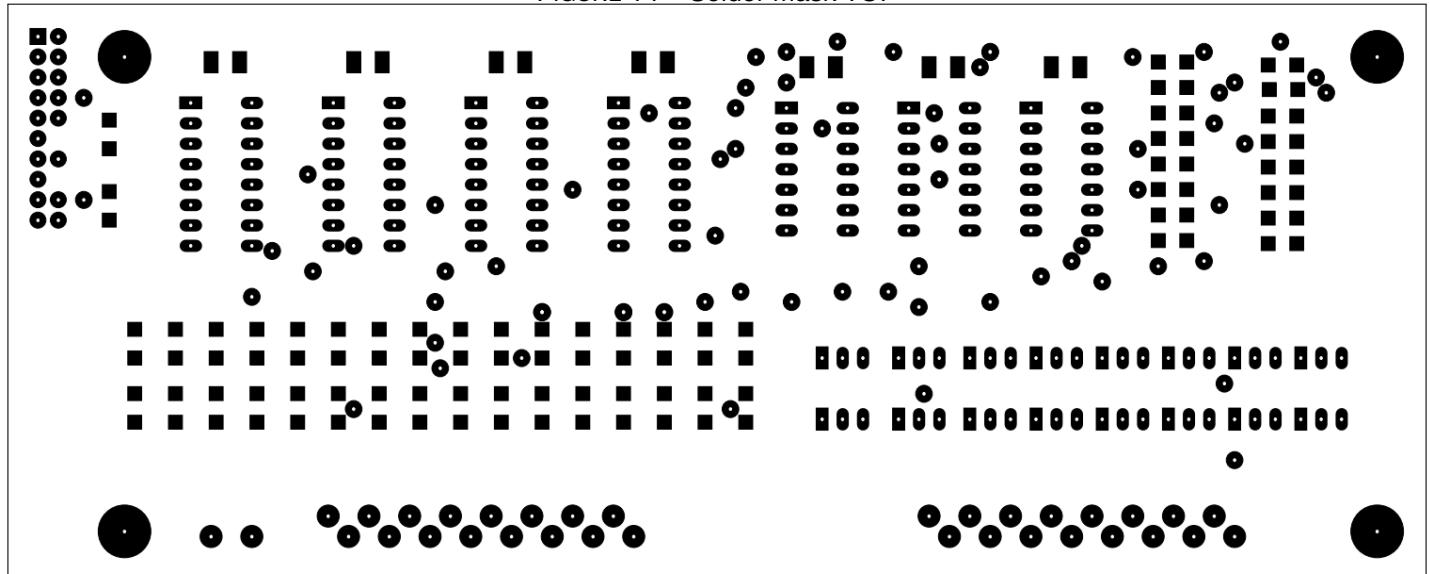
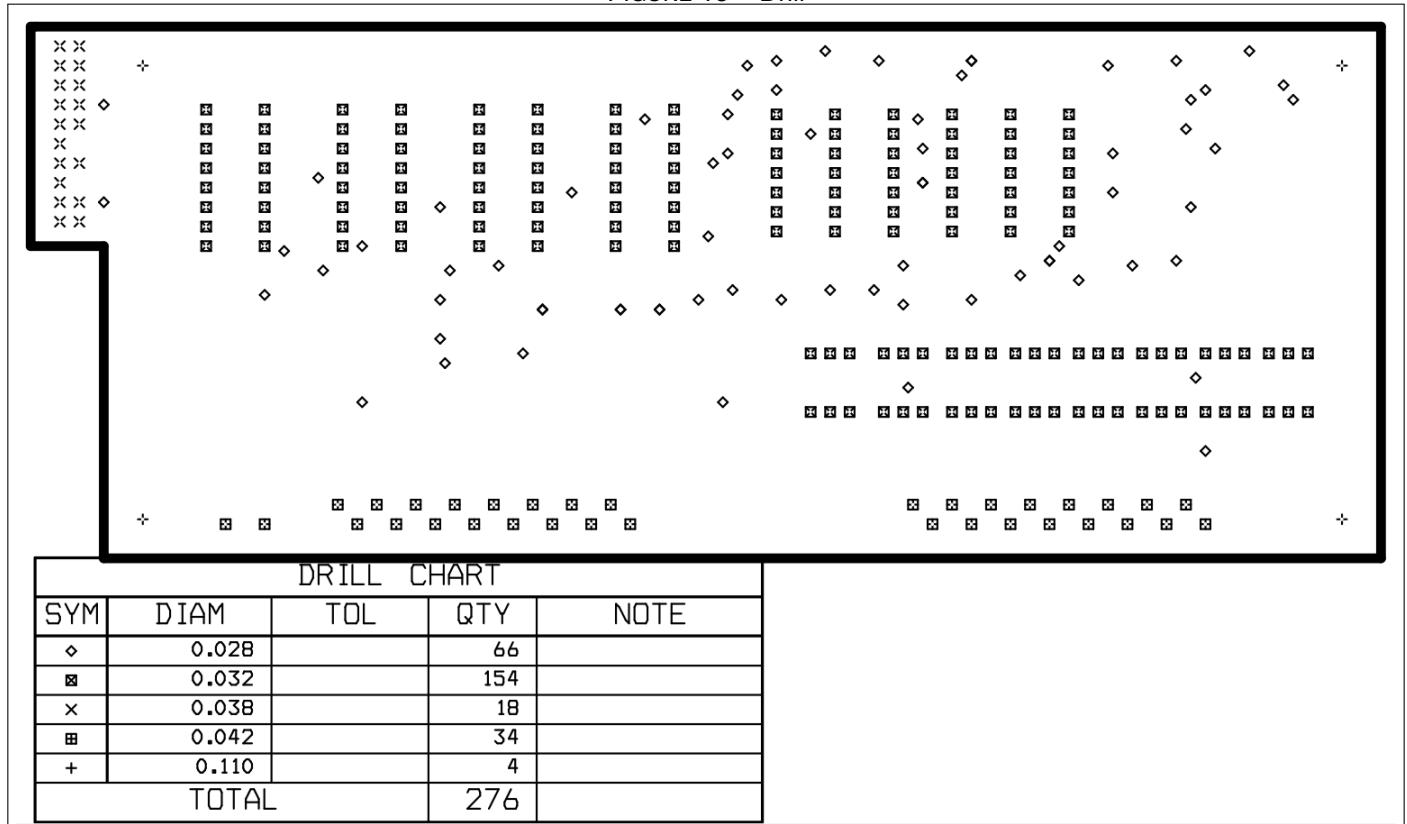
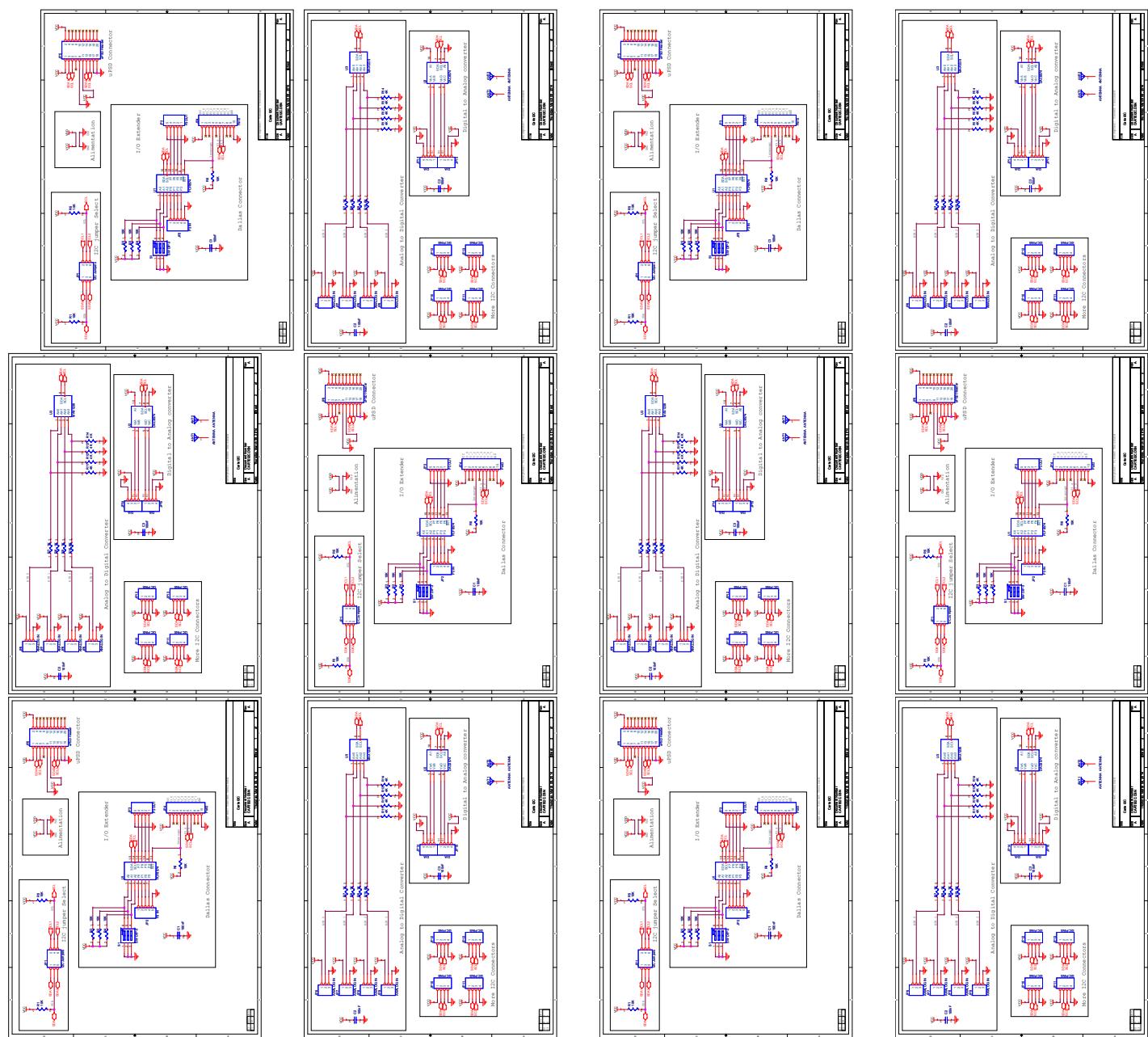


FIGURE 15 – Drill



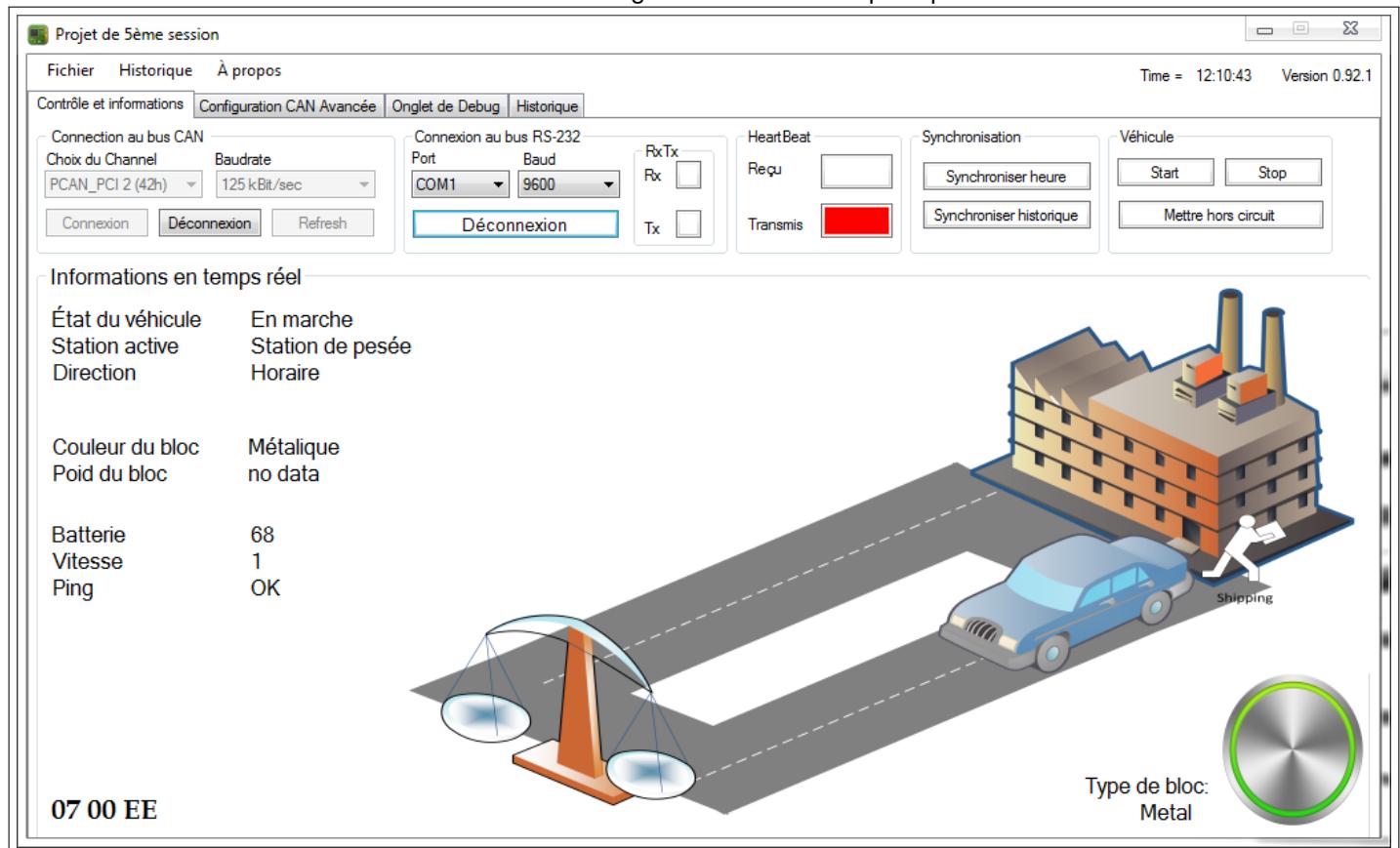
### 6.3 Autres Schémas OrCAD

Note : Les consignes demandent de mettre, et je cite : « les schémas de toutes les cartes de circuits utilisés dans le projet. » Puisque ces cartes ont déjà fait l'objet d'une évaluation par le passé ou qu'elles sont simplement fournies par le collège, j'ai pris la liberté de mettre de petites images. Cela dit, ces images sont non compressées, alors il suffit de zoomer sur le document PDF afin de pouvoir les apprécier dans leur moindre détails.



## 7 Interface PC

FIGURE 16 – Programme de contrôle principal

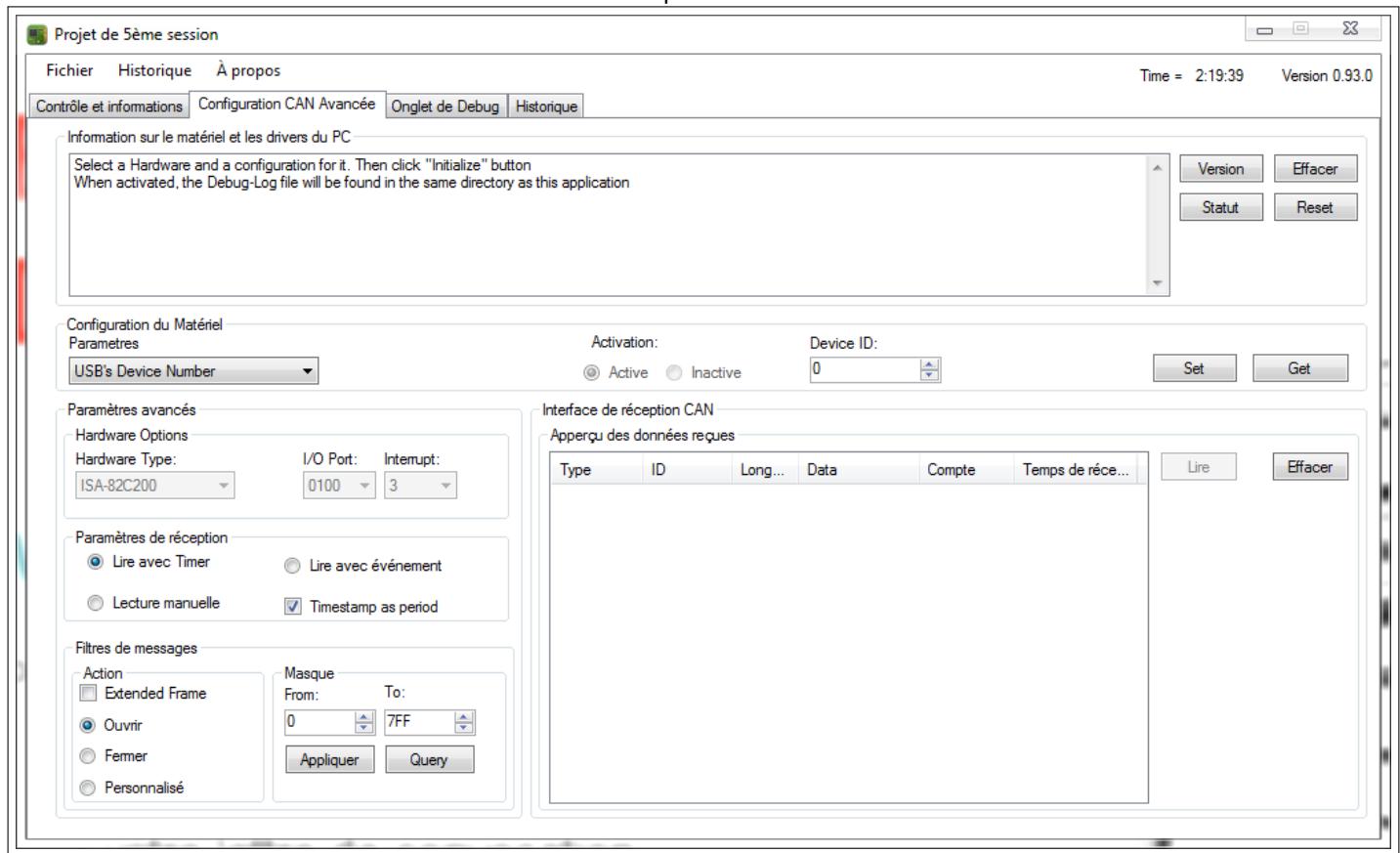


Notre programme, écrit en C# à l'aide de Visual Studio, peut se connecter au bus CAN via une carte PCI et au bus RS232 via un câble DB9 ou USB<sup>7</sup>. La connexion RS232 sert à l'envoi et à la réception du HeartBeat afin d'informer le SOC8200 si l'ordinateur en venait à connaître une défaillance. 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. Enfin, toutes les données pertinentes tel que l'historique, la direction, le poids du bloc, etc. sont affichées à l'écran.

Parler de l'ordre de gestion des tâches...

7. S'il y a présence d'un FTDI

FIGURE 17 – Options CAN avancées



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

## 7.1 Gestion de l'historique

FIGURE 18 – Historique des actions

Le véhicule est à la table FESTO  
Heartbeat Envoyé  
Heartbeat Reçu

2014-12-10 12:11:45  
Error sending data on CAN bus (try catch error)

2014-12-10 12:11:46  
Le bloc est orange  
Heartbeat Envoyé  
Heartbeat Reçu  
Heartbeat Envoyé  
Heartbeat Reçu

2014-12-10 12:11:47  
Le véhicule est à la station de pesée  
Heartbeat Envoyé  
Heartbeat Reçu

2014-12-10 12:11:49  
Le véhicule est à la table FESTO  
Heartbeat Envoyé  
Heartbeat Reçu

2014-12-10 12:11:50  
Transfert de l'historique du SOC vers le PC  
Heartbeat Envoyé  
Heartbeat Reçu  
Heartbeat Reçu

2014-12-10 12:11:51  
Le véhicule est en marche  
Heartbeat Envoyé

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

### 7.1.1 Exemple d'historique typique

Insérer copié-collé de l'historique ici

## 8 Logiciel du SOC8200

### 8.1 Description du programme

D'un commun accord de l'équipe, le programme du SOC2800 a été 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, elles ne viennent qu'aggraver la situation. De plus, son gestionnaire de licence<sup>8</sup> frustre quiconque souhaite l'utiliser. Quant au script shell, il est à la fois plus simple et permet de faire plus en moins de temps, enfin, je crois.

### 8.2 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 ou si l'utilisateur a appuyé sur la lettre A. Quant aux autres scripts, soit *connexion.sh*, *RxCan.sh*, *tcp.sh*, *can.sh* et *Envoi.sh*, ce ne sont que des fonctionnalités que *Projet.sh* appelle en asynchrone afin de lire et d'écrire sur le port série et sur le bus CAN.

### 8.3 Gestion des processus et du temps de CPU

Le seul processus synchrone est script principal *Projet.sh*. Tous les scripts appelés par *Projet.sh* ainsi que leurs processus enfants sont exécutés en asynchrone. Lorsqu'un script doit passer un paramètre à un autre script, un fichier est créé (à l'aide de la commande echo ou cat) afin de contenir le paramètre en question. Les processus impliqués se contentent de lire des fichiers.

### 8.4 Format et récupération des logs

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

```
can0      3  [7] 06 00 0E 0F 06 00 00
can0      3  [7] 00 00 0E 0F 08 00 00
can0      3  [7] 00 00 0E 0F 09 00 00
```

De plus, un autre fichier (histocandate) concatène l'heure à la date dans trames reçues.

```
*****
can0 3 [7] 00 00 0E 0F 09 00 00
Wed Dec 10 14:15:07 UTC 2014
```

```
*****
can0 3 [7] 00 00 0E 0F 0A 00 00
Wed Dec 10 14:15:08 UTC 2014
```

### 8.5 Liste des tests et logiciels

Beaucoup de débogage à la chandelle avec l'écran intégré et PCANview Basic.exe.

---

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

## 9 Logiciel du bolide et de la station uPSD

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.

### 9.1 Logiciel du module PIC18F258

Le programme de la station no.1 (appelée Bloc 2 dans le cahier de consignes) est écrit en C à l'aide de MPLABX 6.00...

### 9.2 Logiciel de la table FESTO

La station no.2 (qui s'appelle Bloc no.3 dans le cahier de consignes) est composée de la table FESTO, de la carte uPSD, de la carte d'extensions IO que nous avons réalisée avec OrCAD.

### 9.3 Schéma des héritages de classes

Quelles classes utilisent quelles autres classes dans notre code ?

FIGURE 19 – Héritage de la classe de contrôle du véhicule

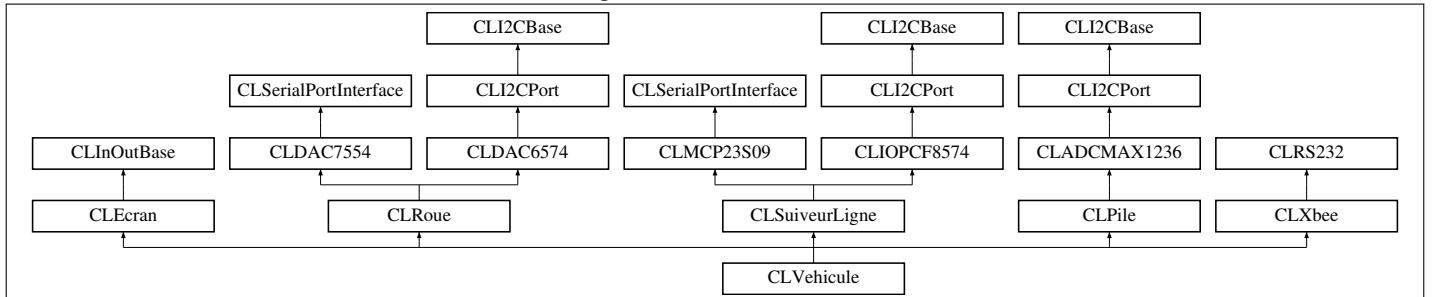


FIGURE 20 – Qui hérite du SPI ?

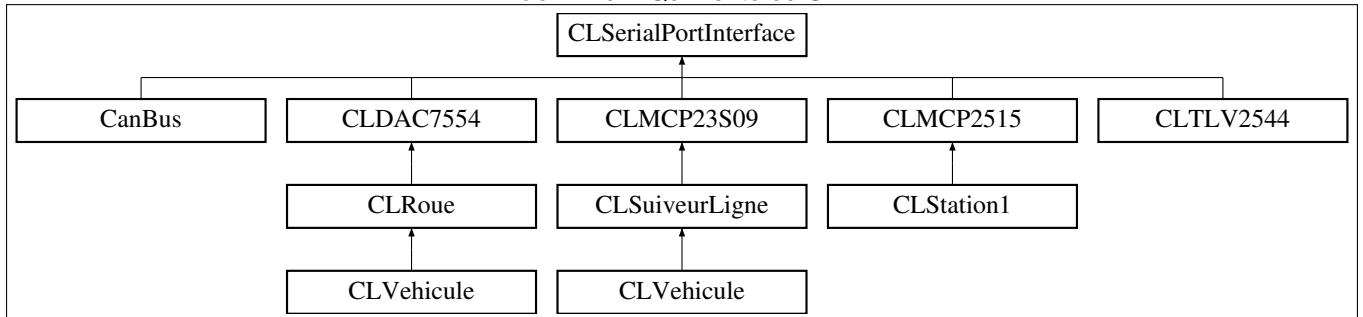


FIGURE 21 – Qui hérite de l'I2C ?

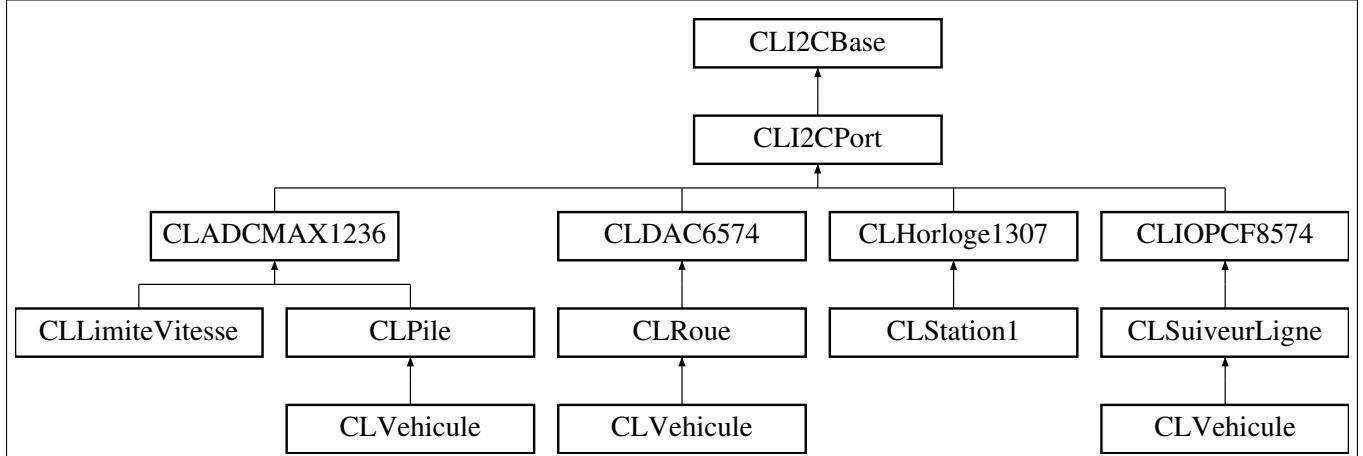
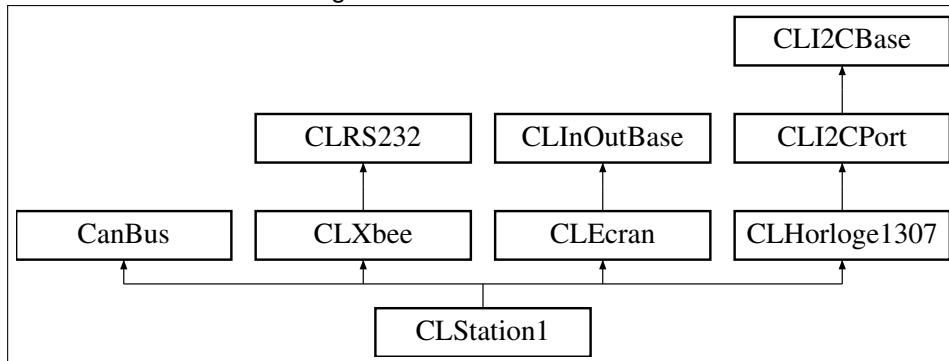


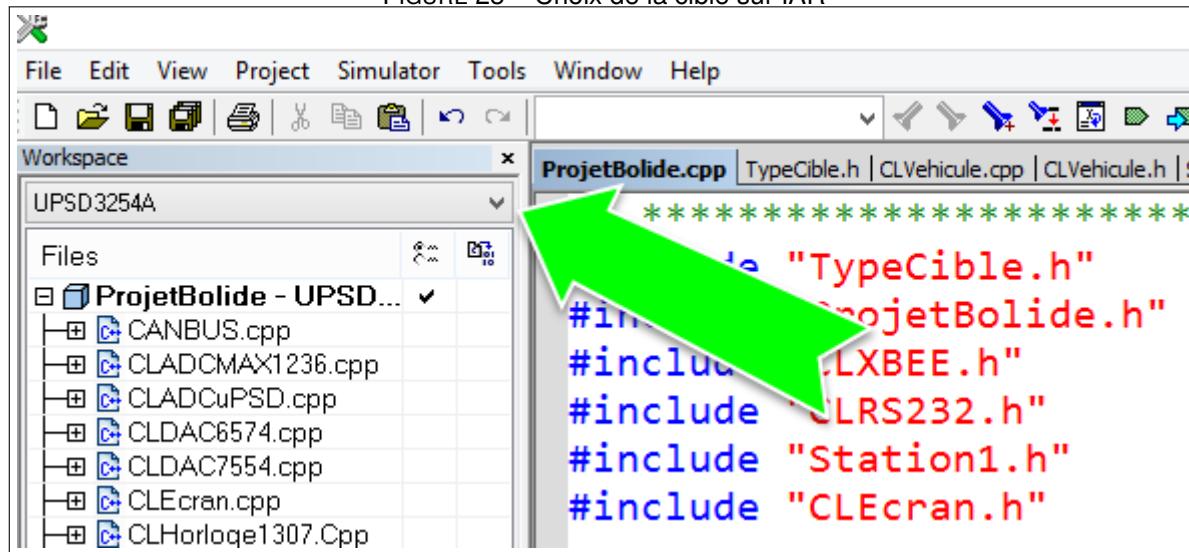
FIGURE 22 – Héritage de la classe de contrôle de la station no.1



## 9.4 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 uPSD. Pour lancer une compilation, rien de plus simple que d'appuyer sur F6.

FIGURE 23 – 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 UPSD3254A
#define DALLAS89C450
#define SPI.DALLAS
#define I2C.DALLAS
#define PCF.5.CAPTEURS
#define PCF.3.CAPTEURS

```

## 9.5 Procédure de vérification

Pour les vérification, rien de plus simple. Il suffit d'envoyer le fichier .hex dans le microcontrôleur et d'observer visuellement le fonctionnement du montage. Certains appellent cette technique « débogage à la chandelle<sup>9</sup>. » Cela étant dit, afficher des trames et autres données sur un écran LCD aide grandement.

9. Salut Étienne!!!

## 10 Logiciel du module PIC18F258

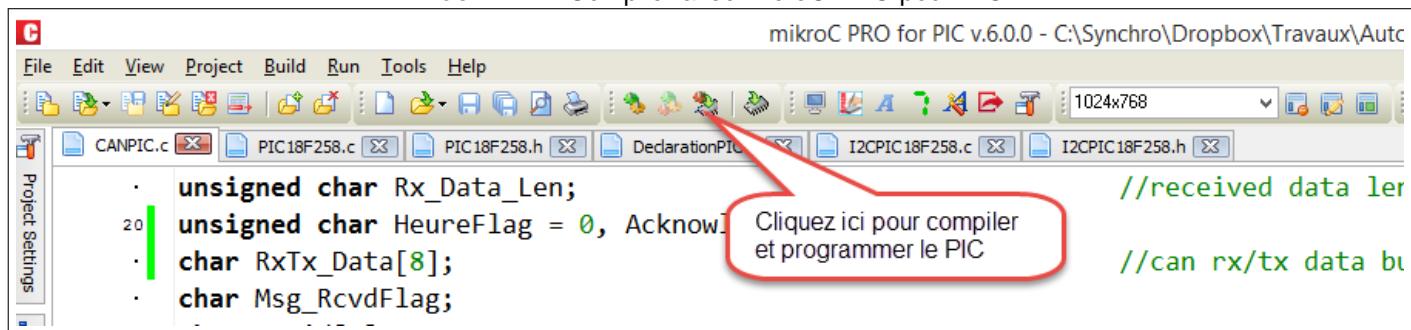
### 10.1 Description du fonctionnement du programme

La structure du programme est on ne peut plus classique, car il s'agit d'un programme écrit en C et disposant d'un bon vieux `while(1)` à l'intérieur du `void main(void)`. En bref, le programme initialise d'abord les différents registres du PIC, puis le bus CAN, I2C et RS232. Quant à la boucle, elle lit le poids du bloc, traduit les trames CAN en données RS232 et expédie le tout au... Où déjà ?

### 10.2 Procédure de compilation sur MPLAB

C'est simple, faites ce qui est indiqué sur l'image ci-dessous :

FIGURE 24 – Compiler avec MicroC PRO pour PIC



### 10.3 Procédure de vérification

## 11 Calculs

### 11.1 Calcul du pas de conversion de la pile

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

### 11.2 Calcul du baudrate

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

Alors...

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

## 12 Conclusions

### 12.1 Ce que le projet m'a apporté

#### 12.1.1 Vincent

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.1.2 Hicham

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.1.3 Gabriel

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.1.4 Louis-Normand

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

### 12.2 Difficultés et corrections

#### 12.2.1 Vincent

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.2.2 Hicham

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.2.3 Gabriel

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.2.4 Louis-Norman

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

### 12.3 Ce que j'ai aimé ou pas

#### 12.3.1 Vincent

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.3.2 Hicham

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.3.3 Gabriel

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

#### 12.3.4 Louis-Norman

Le Lorem Ipsum est simplement du faux texte employé dans la composition et la mise en page avant impression. Le Lorem Ipsum est le faux texte standard de l'imprimerie depuis les années 1500, quand un peintre anonyme assembla ensemble des morceaux de texte pour réaliser un livre spécimen de polices de texte.

---

## 13 ANNEXE 1 : Code source du programme pour PC

```
using System;
```

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

```
using System;
```

## 15 ANNEXE 4 : Code source de la table FESTO

```
using System;
```

## 16 ANNEXE 5 : Code source du programme PIC

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

## 17 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.