



Cégep Limoilou

ÉLECTRONIQUE PROGRAMMABLE ET ROBOTIQUE

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

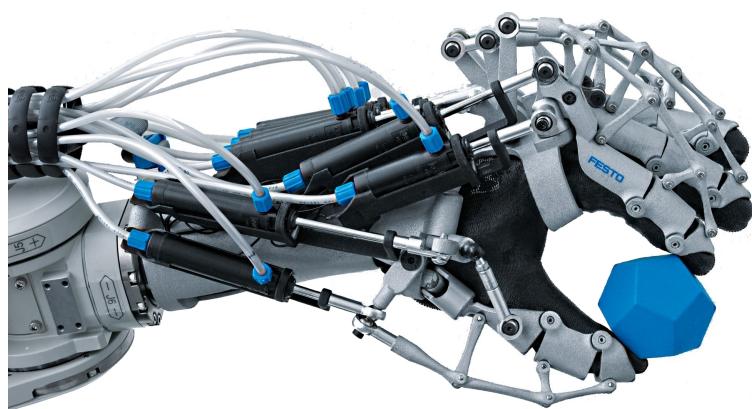
Projet de 5^e session

Étudiants :

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

Professeurs :

Ali Tadli
Alain Champagne
Stéphane Deschênes
Étienne Tremblay



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

21 novembre 2014

Table des matières

1 Présentation du projet	5
1.1 Explication du projet	5
1.2 Schéma bloc du système	5
1.2.1 Bloc 1	5
1.2.2 Bloc 2	5
1.2.3 Bloc 3	5
1.2.4 Bloc 4	5
1.3 Liste des logiciels	5
1.4 Liste des trames	7
2 Le matériel	8
2.1 Bloc 1	8
2.2 Bloc 2	8
2.3 Bloc 3	8
2.4 Bloc 4	8
2.5 Explication des types de liens	8
2.5.1 RS232	8
2.5.2 Xbee	8
2.6 Explication des trames	8
2.6.1 RS-232	8
2.6.2 CAN	9
2.6.3 XBEE	11
2.7 Liste des pièces	11
2.7.1 Liens web	11
2.7.2 Datasheets	12
3 Interface PC	13
3.1 Gestion de l'historique	15
3.1.1 Exemple d'historique typique	15
3.2 Structure du programme	15
3.2.1 Les Ghosts Labels	15
3.3 Explication des trames	15
3.4 Ordre de gestion des tâches	15
4 Logiciel du SOC8200	16
4.1 Description du programme	16
4.2 Schéma bloc	16
4.2.1 Du code	16
4.2.2 Du script shell	16
4.3 Gestion des processus et du temps de CPU	16
4.4 Format et récupération des logs	16
4.5 Liste des tests et logiciels	16
5 Logiciel de la station 1 et 2 et du bolide	17
5.1 La station no.1	17
5.2 La station no.2	17
5.3 Le bolide	17
5.4 Procédure de compilation sur IAR	18
5.5 Procédure de vérification	18

6 Logiciel du module PIC18F258	18
6.1 Description du fonctionnement du programme	18
6.2 Procédure de compilation sur MPLAB	18
6.3 Procédure de vérification	18
7 Calculs	19
7.1 Calcul du pas de conversion de la pile	19
7.2 Calcul du baudrate	19
8 Fichiers Gerbers	20
9 Conclusion	23
9.1 Ce que le projet m'a apporté	23
9.1.1 Vincent Chouinard	23
9.1.2 Hicham Safoine	23
9.1.3 Gabriel Fortin-Bélanger	23
9.1.4 Louis-Norman Ang-Houle	23
9.2 Difficultés et corrections	23
9.2.1 Vincent Chouinard	23
9.2.2 Hicham Safoine	23
9.2.3 Gabriel Fortin-Bélanger	23
9.2.4 Louis-Norman Ang-Houle	23
9.3 Ce que j'ai aimé ou pas	23
9.3.1 Vincent Chouinard	23
9.3.2 Hicham Safoine	23
9.3.3 Gabriel Fortin-Bélanger	23
9.3.4 Louis-Norman Ang-Houle	23
10 ANNEXE 1 : Code source du Bolide et de la station no.1	24
11 ANNEXE 2 : Code source du programme pour PC	25
12 ANNEXE 6 : Code source du programme PIC	26
13 ANNEXE 3 : Code source du programme du SOC8200	27
14 ANNEXE 5 : Code source de la table FESTO	28

Table des figures

1	Programme de contrôle principal	13
2	Options CAN avancées	14
3	Historique des actions	15
4	Choix de la cible sur IAR	18
5	Couche TOP	20
6	Couche BOT	20
7	Silk Screen TOP	21
8	Solder mask TOP	21
9	Drill	22

Liste des tableaux

1	Index des identifiants matériel CAN	9
2	Index des trames CAN	9
3	Index des communications CAN	9
4	Informations sur le bus I ₂ C du bolide	17

1 Présentation du projet

Le projet de la cinquième session consiste à réaliser

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

1.1 Explication du projet

1.2 Schéma bloc du système

1.2.1 Bloc 1

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

1.2.2 Bloc 2

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

1.2.3 Bloc 3

1.2.4 Bloc 4

1.3 Liste des logiciels

Terminaux

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

Gestion du projet

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

Compilateurs et IDE

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

Éditeur de texte

- | | | |
|---|---|---|
| <ul style="list-style-type: none">• Notepad++• gedit• medit 1.2.0 <p>Schémas électriques</p> <ul style="list-style-type: none">• OrCAD 16.2 <p>Système d'exploitation</p> | <ul style="list-style-type: none">• Windows 7 SP1• Windows 8.1• Windows XP SP3• Fedora 20• CentOS• Lubuntu 14.10 | <p>Autres</p> <ul style="list-style-type: none">• VMWare Workstation 10• TeXmaker 4.3• Dukto R6• Dia• Festo configuration tool |
|---|---|---|
-

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. Il n'y a rien à faire, mise à part brancher les bons câbles aux bons endroits.

2.2 Bloc 2

2.3 Bloc 3

2.4 Bloc 4

2.5 Explication des types de liens

2.5.1 RS232

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

2.5.2 Xbee

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

2.6 Explication des trames

2.6.1 RS-232

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

2.6.2 CAN

Chaque dispositif matériel, du Bolide au PC, dispose d'un identifiant CAN allant de 000 à 005. Chaque fonctionnalité dispose d'un code d'identification suivi de la donnée 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	TimeStamp
Démarre le véhicule	0x00	0x00	TimeStamp
Arrête le véhicule	0x00	0x01	TimeStamp
Le véhicule est arrêté	0x01	0x00	TimeStamp
Le véhicule est en marche	0x01	0x01	TimeStamp
Le véhicule est hors circuit	0x01	0x02	TimeStamp
Vitesse (0-100)	0x02	0x00 à 0x64	TimeStamp
Battre	0x03	0x00 à 0x64	TimeStamp
Couleur du bloc	0x04	0x00 à 0x02	TimeStamp
Poids du bloc	0x05	0x00 à 0x64	TimeStamp
Envoyer l'heure	0x06	à déterminer	TimeStamp
No. de la station	0x07	0x00 à 0x02	TimeStamp
Demande de l'historique	0xC0	0x00	TimeStamp
Direction horaire et antihoraire	0x08	0x00 à 0x01	TimeStamp

TABLE 3 – Index des communications CAN

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

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 μPSD), sur la station no.2 (la table FESTO) et la station no.6, c'est à dire le bolide. La particularité des Xbee est que lorsqu'ils sont adéquatement configurés, tout ce qu'envoie un Xbee est reçu et lu par tous les autres Xbee à proximité, et c'est pourquoi nous avons défini un système de trames.

2.7 Liste des pièces

- Carte Dallas
- Carte μPSD
- SOC 8200 (avec clavier)
- PIC18FmachinTruc
- Carte d'extension SPI
- Carte d'extension I2C
- Carte CAN MCP2515
- Xbee
- Table FESTO
- Carte d'extension IO
- Carte connecteur IO
- Carte connecteur DAC ADC
- Carte Xbee vers DB9
- Câble Ethernet croisé
- Câble Ethernet régulier
- Câble DB9
- Carte Xbee vers DB8
- Le Bolide
-
-

2.7.1 Liens web

Mettre ien vers GitHub

2.7.2 Datasheets

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

74HC14

Pb-Free Semiconductors		Product Specification			
Hex Inverting Schmitt trigger		74HCT14			
FEATURES					
<ul style="list-style-type: none"> • Device available by standard • I_Q: category B9 					
GENERAL DESCRIPTION					
<p>The 74HCT14 is a high-speed Schottky CMOS device and are pin compatible with low power Schottky TTL, ECL, TLL, HDTL, and EATTL devices.</p> <p>The 74HCT14 provides six inverting buffers with Schmitt-trigger action. They are capable of transforming logic levels changing fast signal into sharply defined, positive output signals.</p>					
APPLICATIONS					
Digital logic, logic gate, DTL, TTL, ECL, EATTL					
SYMBOL	PARAMETER		CONDITIONS		
			TYPICAL		
			HCT		
			UNIT		
Symbol	Inverter		ns		
I _Q (max)	DC (10-15°C, V _{DD} =5.5V)		ns		
I _Q (typical)	DC (25°C, V _{DD} =5V)		22		
Power dissipation	DC (25°C, V _{DD} =5V)		μW		
Notes					
1. Capacitor used to determine the dynamic power dissipation ($P_D = \frac{1}{2}C(V_{DD} - V_{DD})^2$)					
2. $I_Q = I_{Q1} + I_{Q2}$, where $I_{Q1} = I_{Q1}(V_{DD}, V_{SS}, V_{DD} - V_{SS})$					
3. Input frequency is 1 MHz					
4. $V_{DD} = 5V$ unless otherwise specified					
5. $V_{DD} = 5V$ unless otherwise specified					
6. For HCT condition $V_{DD} = 5V$, $V_{SS} = 0V$					
7. For HCT condition $V_{DD} = 5V$, $V_{SS} = 0V$					
ORDERING INFORMATION					
See "74HCT/HCT Logic Package Information".					

LineSensors

PCF8573

 <p>TPS701A</p> <p>Low Shutdown Current Consumption of 10 μA Maximum ±0.05% Linearity Error Open-Drain Output Expander Compatible With Most Microcontrollers</p>	 <p>TSSOP</p> <p>Pin Configuration 1 V_{DD} 2 GND 3 V_{REF} 4 GND 5 GND 6 GND 7 GND 8 GND 9 GND 10 GND 11 GND 12 GND 13 GND 14 GND 15 GND 16 GND</p>	 <p>DFN</p> <p>Pin Configuration 1 V_{DD} 2 GND 3 V_{REF} 4 GND 5 GND 6 GND 7 GND 8 GND 9 GND 10 GND 11 GND 12 GND 13 GND 14 GND</p>																
REBATE & BPI EXPANDER FOR PC-BUS																		
<p>Description/Ordering Information</p> <p>The TPS701A provides a bidirectional I_O/I_H response for the two-directional I_O/I_H response required by the PC-ibus interface.</p> <p>The PC-ibus protocol provides two parallel bytes for each port: one for reading and one for writing.</p> <p>The device features a bidirectional I_O/I_H response for the two-directional I_O/I_H response required by the PC-ibus interface.</p> <p>The use of a bidirectional signal allows signal A to be high or low. In the mode, the command source can force logic high or logic low. The bidirectional signal is controlled by the logic level of SCL. The bidirectional signal can be high or low when a register is written high or switched by the negative edge of SCL. The I_O should be high during write operations.</p> <p>ORDERING INFORMATION</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Part</th> <th>Packaging</th> <th>General Description</th> <th>Normal Lead Time</th> </tr> </thead> <tbody> <tr> <td>TPS701A</td> <td>SOT-23</td> <td>10 μA Shutdown Current, ±0.05% Linearity, 14-Pin SOT-23</td> <td>10 weeks</td> </tr> <tr> <td>TPS701A</td> <td>TSSOP</td> <td>10 μA Shutdown Current, ±0.05% Linearity, 16-Pin TSSOP</td> <td>10 weeks</td> </tr> <tr> <td>TPS701A</td> <td>DFN</td> <td>10 μA Shutdown Current, ±0.05% Linearity, 14-Pin DFN</td> <td>10 weeks</td> </tr> </tbody> </table> <p>Notes: Please be aware that all registered trademarks belong to their respective companies. © 1996, Texas Instruments Incorporated. All rights reserved. Printed in U.S.A.</p> <p>TI Logo</p> <p>TEXAS INSTRUMENTS</p> <p>Copyright © 1996, Texas Instruments Incorporated</p>			Part	Packaging	General Description	Normal Lead Time	TPS701A	SOT-23	10 μA Shutdown Current, ±0.05% Linearity, 14-Pin SOT-23	10 weeks	TPS701A	TSSOP	10 μA Shutdown Current, ±0.05% Linearity, 16-Pin TSSOP	10 weeks	TPS701A	DFN	10 μA Shutdown Current, ±0.05% Linearity, 14-Pin DFN	10 weeks
Part	Packaging	General Description	Normal Lead Time															
TPS701A	SOT-23	10 μA Shutdown Current, ±0.05% Linearity, 14-Pin SOT-23	10 weeks															
TPS701A	TSSOP	10 μA Shutdown Current, ±0.05% Linearity, 16-Pin TSSOP	10 weeks															
TPS701A	DFN	10 μA Shutdown Current, ±0.05% Linearity, 14-Pin DFN	10 weeks															

DAC6574

LM3914

Texas
Instruments

LM3914

uPSD3254A

DS89C450

OPT101

IBB Ibur-Brown Products
from Texas Instruments

OPT101

SIXTEEN-MONOLITHIC OPTICAL SENSORS
DECEMBER 1984 EDITION OCTOBER 1991

MONOLITHIC PHOTODIODE AND SINGLE-SUPPLY TRANSMISSION AMPLIFIER

FEATURES

- BIPOLAR PHOTODIODE
- PHOTOCURRENT: 1.6A (0.599 mA)
- INTERNAL 1MΩ PHOTOREGISTER
- INTERNAL 100Ω GAIN CONTROL
- BANDWIDTH: 100Hz to 1MHz
- LOW NOISE
- AVAILABLE IN SOT-89D OR D-CAD SURFACE-MOUNT PACKAGES

APPLICATIONS

- INDUSTRIAL PROCESS MONITORING
- LABORATORY INSTRUMENTATION
- AUTOMOTIVE SAFETY SENSORS
- PHOTOGRAPHIC ANALYZERS
- BARCODE SCANNERS
- IMAGE PROCESSING
- CURRENT CHANGERS

DESCRIPTION

The OPT101 is a monolithic photodiode with negative transconductance and digital current increase function. The device is designed for use as a single-supply photodetector, making it ideal for barcode reading, optical character recognition, and other applications.

The integrated combination of photodiode and transmission amplifier provides a low noise, high performance optoelectronic detector. The problem normally encountered in discrete designs such as a photodiode and operational amplifier is the difficulty in matching the two devices. The OPT101 overcomes this by using a single monolithic structure to achieve the best performance in every aspect. The 1.6A (0.599 mA) peak photocurrent is matched to the 100Ω internal gain control resistor to provide a low noise, high dynamic range and low dark current.

The operating current from 2.7V to 5.0V and current gain in excess of 10,000 is available in three surface-mount packages. The OPT101 is a high performance optoelectronic detector with a wide operating temperature range of -40°C to +70°C.

A WORD OF CAUTION: When using this or any bipolar junction photodiode, attention must be paid to potential breakdown of the reverse biased junction. Under certain conditions and under certain test conditions, breakdown voltage can be less than the reverse bias voltage.

For more information on important bipolar junction photodiode characteristics, see our general application note, "Characteristics and Applications of Bipolar Junction Photodiodes," available from the Application Engineering Department.

IBB is a registered trademark of Ibur-Brown Products, Inc. Texas Instruments is a registered trademark of Texas Instruments Incorporated. © 1991 Ibur-Brown Products, Inc. All rights reserved.

IBB Division of Texas Instruments

Texas Instruments

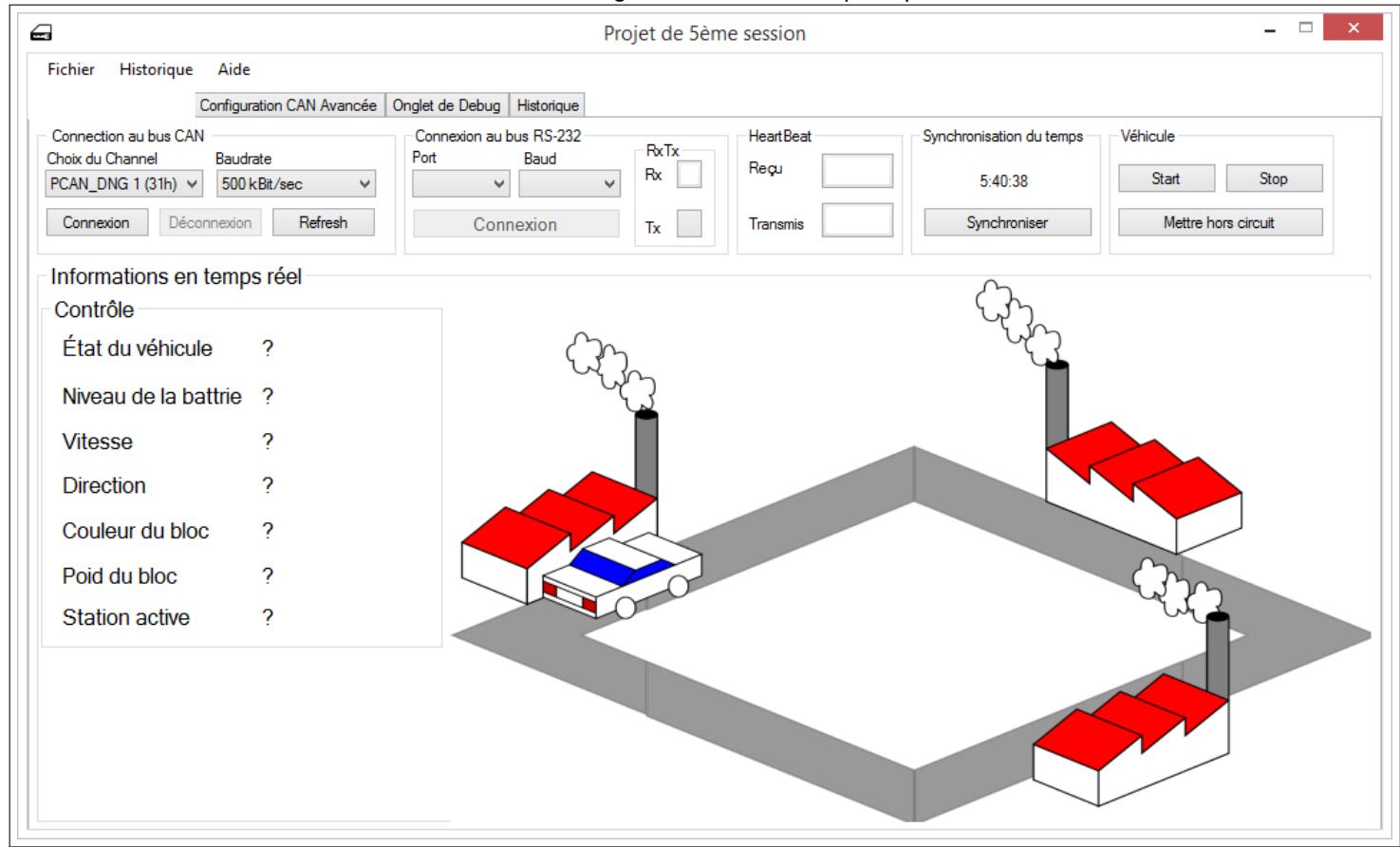
International Semiconductor

SaberTooth motor drive

1. 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.

3 Interface PC

FIGURE 1 – Programme de contrôle principal

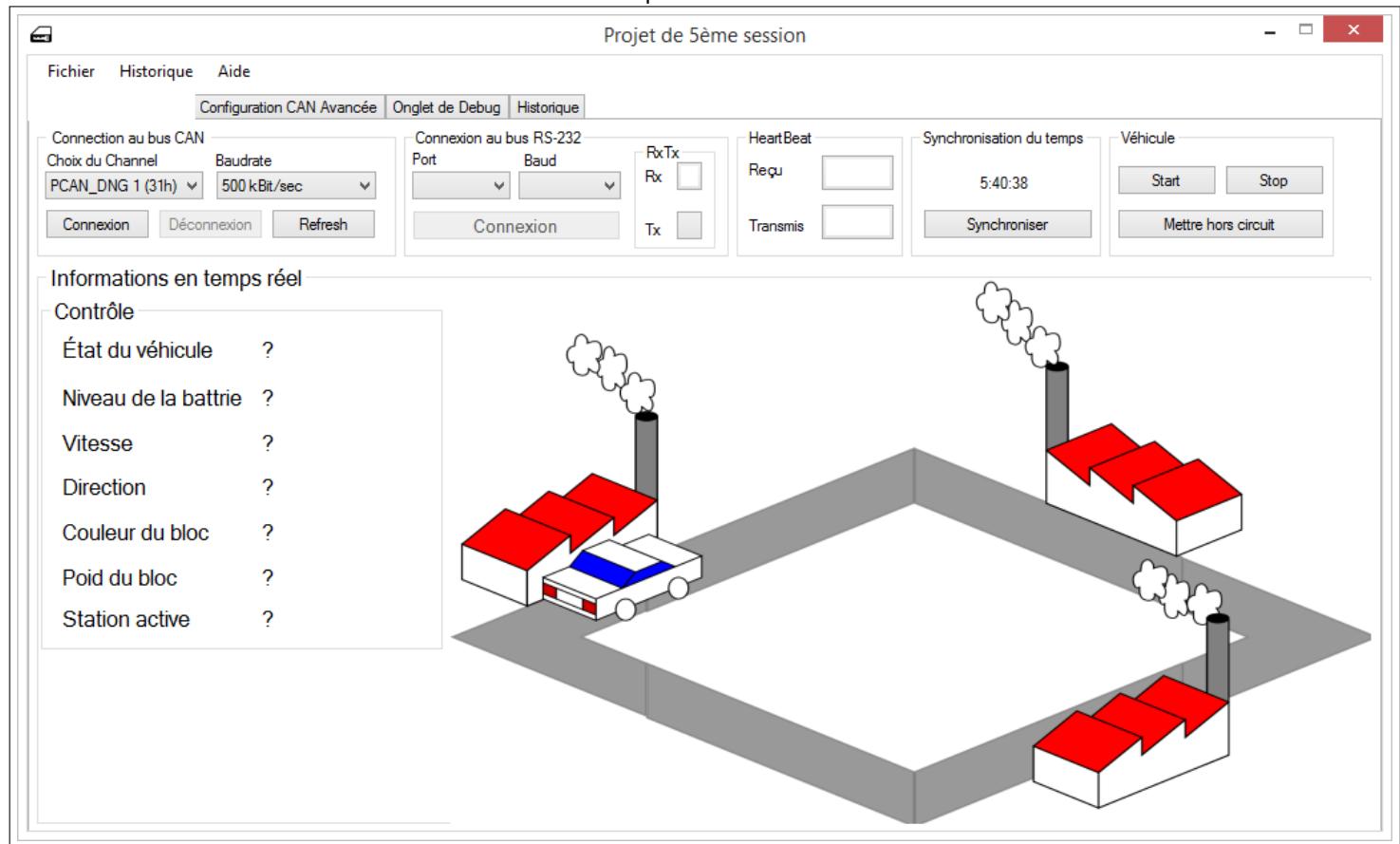


Notre programme, écrit en C# à l'aide de Visual Studio, peut se connecter au bus CAN via une carte SPI² et au bus RS232 via un câble DB9 ou USB³. La connexion RS232 sert à l'envoie et à la réception du HeartBeat afin d'informer le SOC8200 si l'ordinateur en venait à connaître une défaillance. De plus,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

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

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

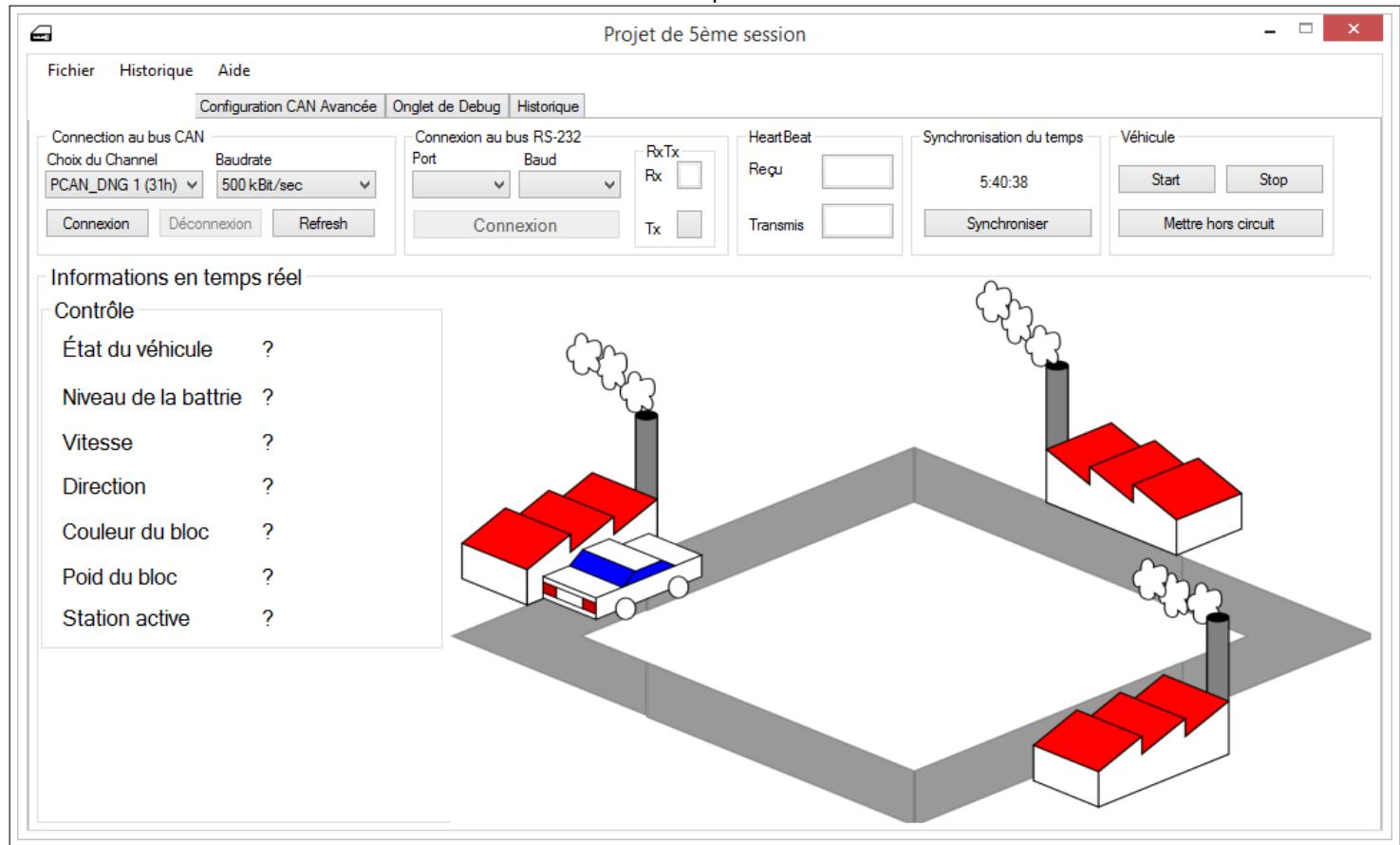
FIGURE 2 – Options CAN avancées



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

3.1 Gestion de l'historique

FIGURE 3 – Historique des actions



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

3.1.1 Exemple d'historique typique

wefsd

3.2 Structure du programme

3.2.1 Les Ghosts Labels

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

3.3 Explication des trames

3.4 Ordre de gestion des tâches

4 Logiciel du SOC8200

4.1 Description du programme

Le programme du SOC2800 est écrit en script Shell⁴ ⁵

4.2 Schéma bloc

4.2.1 Du code

Mettre Doxygen generated stuff

4.2.2 Du script shell

4.3 Gestion des processus et du temps de CPU

4.4 Format et récupération des logs

4.5 Liste des tests et logiciels

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

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

5 Logiciel de la station 1 et 2 et du bolide

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

5.1 La station no.1

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

5.2 La station no.2

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

5.3 Le bolide

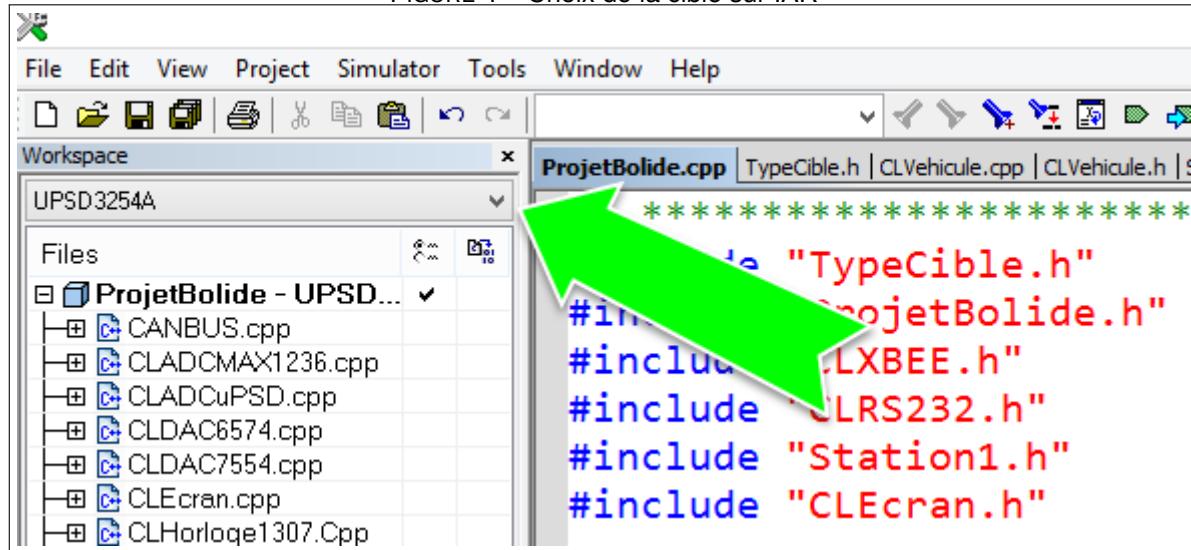
Composante	Adresse I2C	Description
MAX1236	0x68	Convertisseur analogique-numérique
DS1307	0xD0	Circuit d'horloge RTC
PCF8574	0x40	I/O Expander pour bus I2C
DAC6574	0x98	Convertisseur numérique-analogique
OPT101	0x50	Suiveur de ligne

TABLE 4 – Informations sur le bus I2C du bolide

5.4 Procédure de compilation sur IAR

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

FIGURE 4 – Choix de la cible sur IAR



De plus, des paramètres de compilation optionnelle vous permettent, via la décommentation, de compiler le code pour la carte Dallas ou μPSD , 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

```

5.5 Procédure de vérification

6 Logiciel du module PIC18F258

6.1 Description du fonctionnement du programme

6.2 Procédure de compilation sur MPLAB

6.3 Procédure de vérification

7 Calculs

7.1 Calcul du pas de conversion de la pile

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

7.2 Calcul du baudrate

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

Alors...

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

8 Fichiers Gerbers

Une carte d'extionsion à été réalisée avec OrCad 16.2 et à été gravée sur la rutilante LPKF. Voici les gouches gerber⁶.

FIGURE 5 – Couche TOP

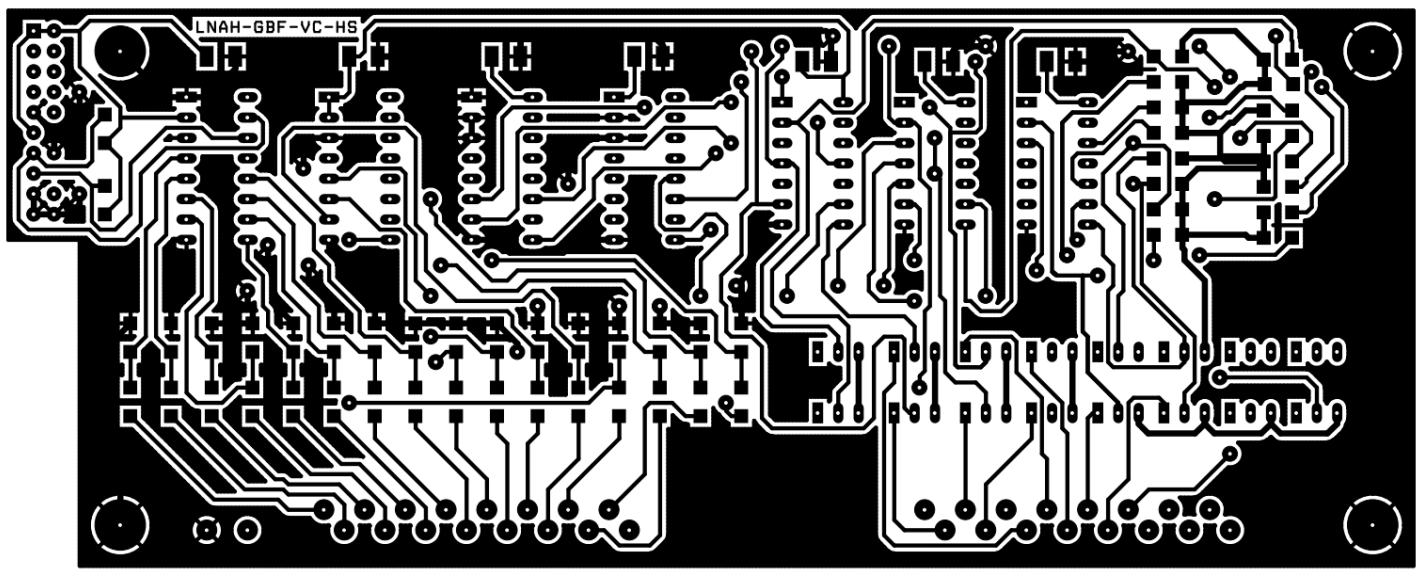
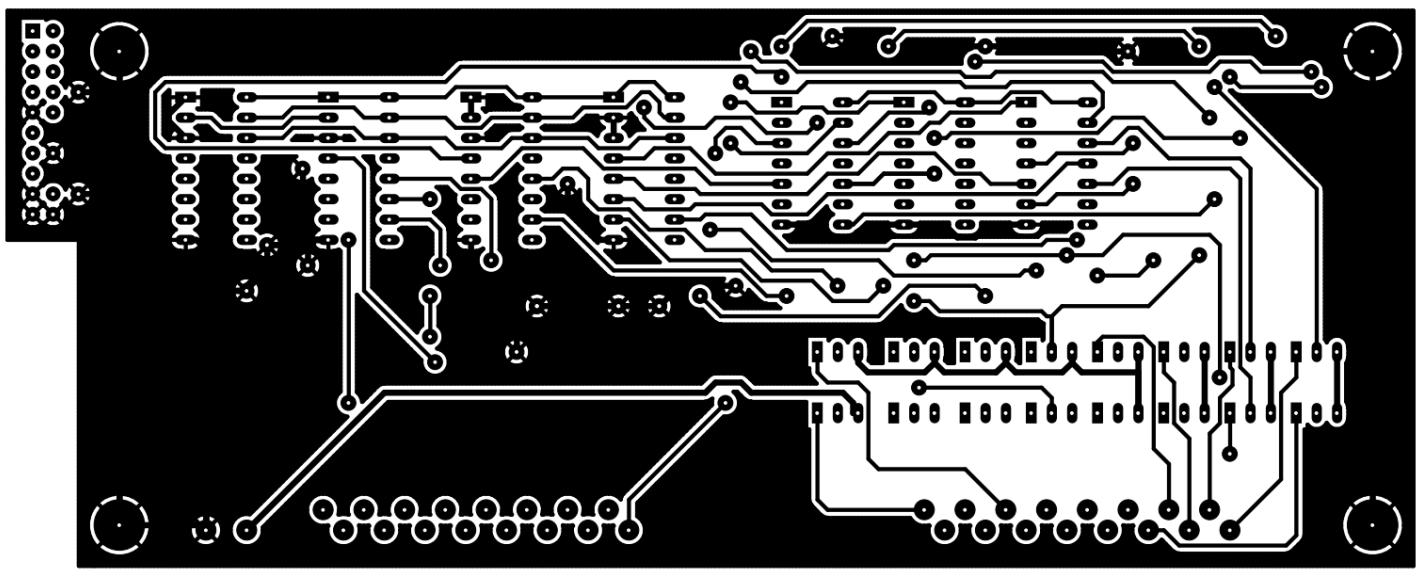


FIGURE 6 – Couche BOT



6. Les couches présentées ne sont pas à l'échelle

FIGURE 7 – Silk Screen TOP

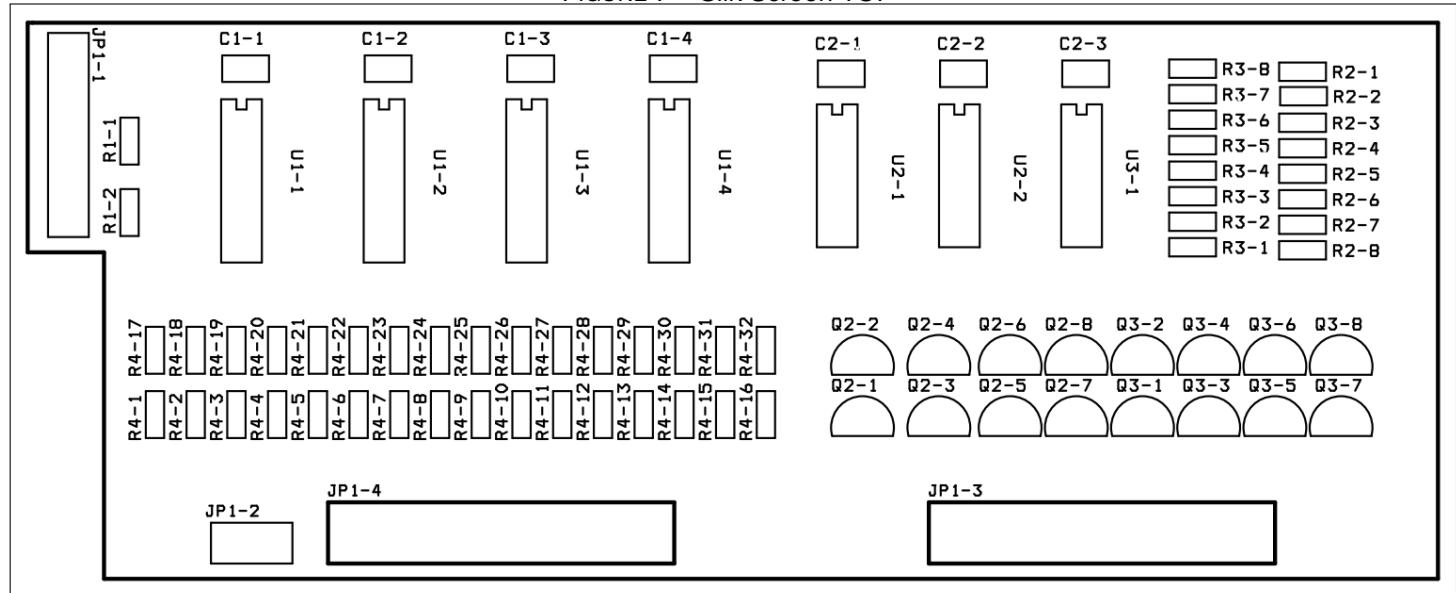


FIGURE 8 – Solder mask TOP

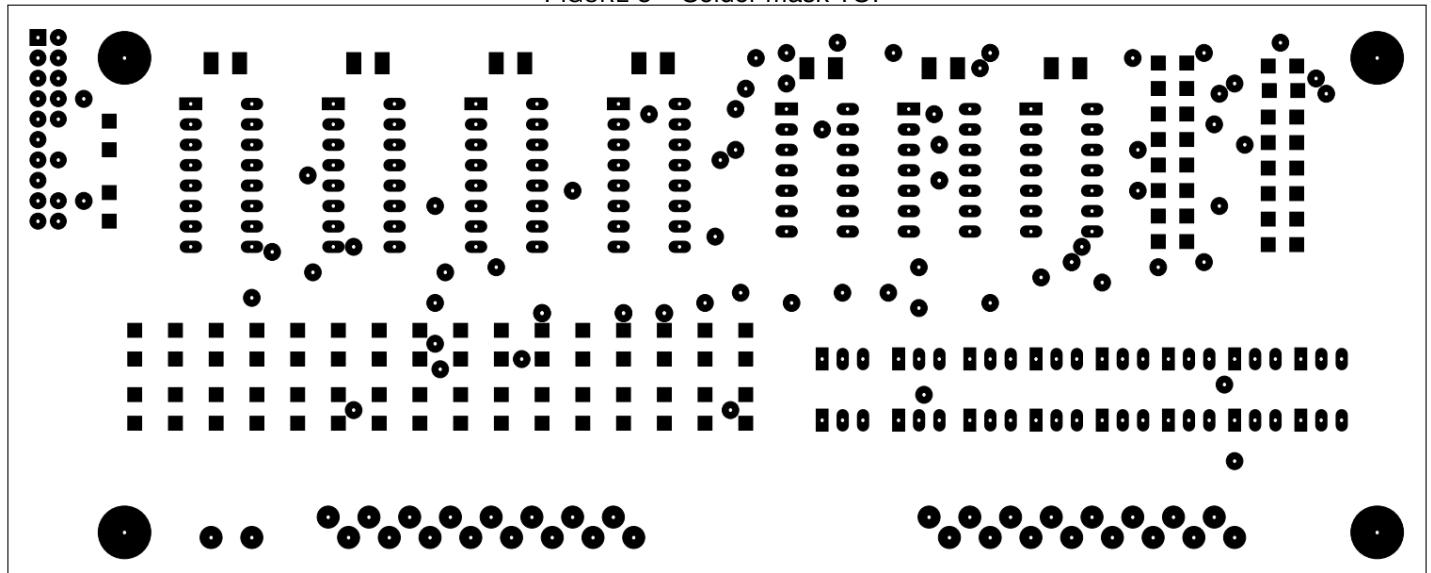
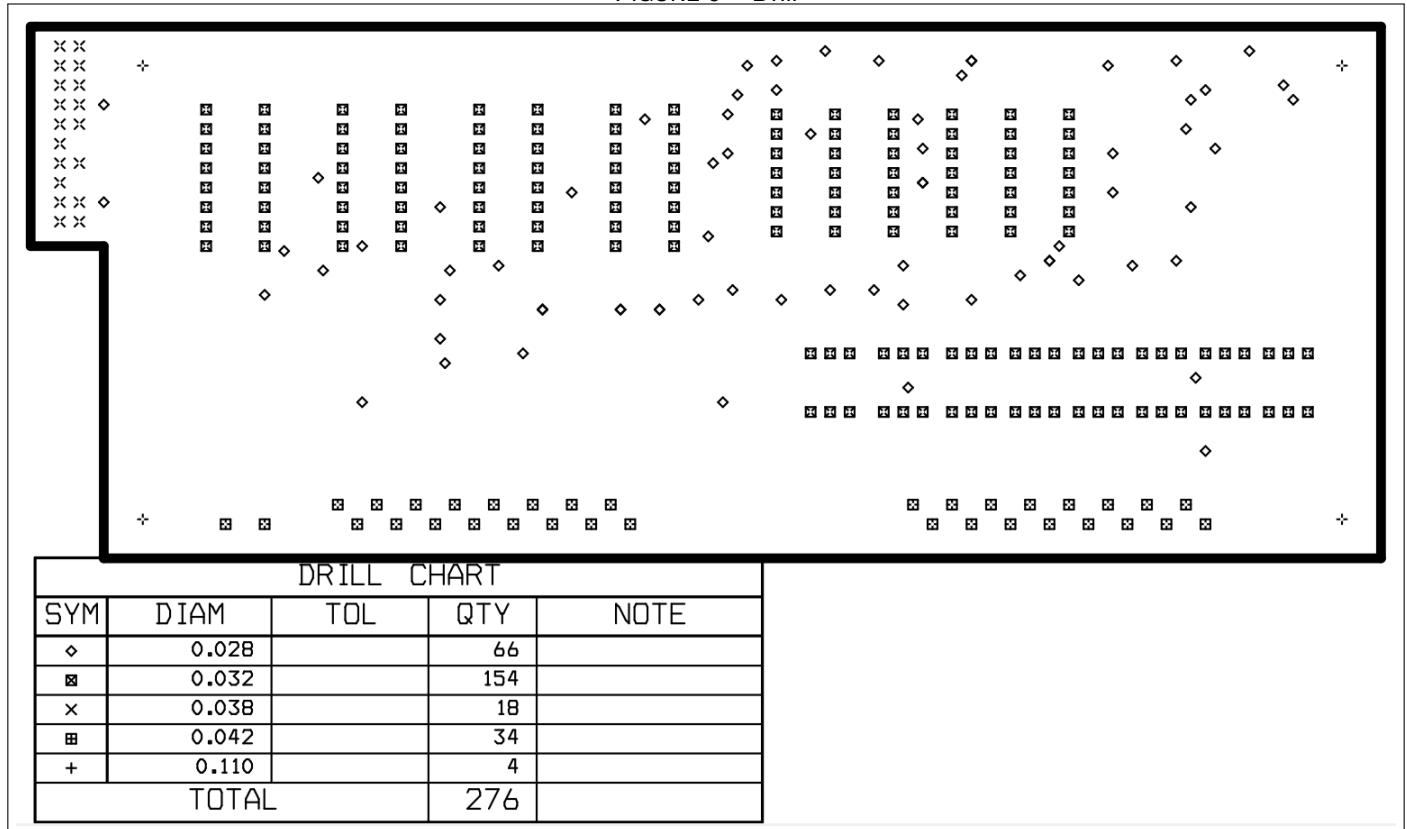


FIGURE 9 – Drill



9 Conclusion

9.1 Ce que le projet m'a apporté

9.1.1 Vincent Chouinard

9.1.2 Hicham Safoine

9.1.3 Gabriel Fortin-Bélanger

9.1.4 Louis-Norman Ang-Houle

9.2 Difficultés et corrections

9.2.1 Vincent Chouinard

9.2.2 Hicham Safoine

9.2.3 Gabriel Fortin-Bélanger

9.2.4 Louis-Norman Ang-Houle

9.3 Ce que j'ai aimé ou pas

9.3.1 Vincent Chouinard

9.3.2 Hicham Safoine

9.3.3 Gabriel Fortin-Bélanger

9.3.4 Louis-Norman Ang-Houle

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

```
using System;
```

11 ANNEXE 2 : Code source du programme pour PC

```
using System;
```

12 ANNEXE 6 : Code source du programme PIC

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

13 ANNEXE 3 : Code source du programme du SOC8200

```
using System;
```

14 ANNEXE 5 : Code source de la table FESTO

```
using System;
```