RAPPORT DE STAGE

Réalisation d'une interface TCVL

Lucien RENAUD

Juin 2013

Maitre de stage :
Mr Christophe GUERBER

Tuteur de stage :
Mr Boutaieb DAHHOU

Remerciements

Je tiens à remercier tout d'abord mon maître de stage, Mr Christophe GUERBER de m'avoir permis de réaliser ce projet. Je remercie également toutes les personnes qui m'ont aidé pour la réalisation de ce projet en particulier Mr Christophe DUMONT et Mr Jean-Daniel GRIL pour les conseils qu'ils m'ont apportés. Je remercie Mr Henri DENIS ainsi que l'ensemble du personnel pour leur accueil à l'ENAC.

Merci

SOMMAIRE

ENAC

INTRODUCTION

Afin de finaliser mon DUT GEII, j'ai été amené à réaliser un stage en entreprise d'une durée de trois mois, afin de mettre en pratique les connaissances acquises durant ma formation.

J'ai effectué ce stage à l'ENAC (École National de l'Aviation Civile) à Toulouse, plus précisément dans la subdivision réseau aéronautique, du 8 avril au 28 juin 2013.

Mon stage consiste à concevoir et à fabriquer un boîtier d'interface pour un TCVL (Télécommande Communications Vocales et Liaisons de données) et un PC. Dans un premier temps je présenterai l'ENAC ainsi que la subdivision où j'ai travaillé, puis dans un deuxième temps, le projet que j'ai réalisé.

I. Présentation de l'entreprise

1. L'ENAC

L'École nationale de l'aviation civile ou ÉNAC est une grande école française fondée le 28 août 1949 qui a pour mission d'assurer la formation initiale et le perfectionnement des cadres et principaux acteurs de l'aviation civile. Elle est membre de la conférence des Grandes Ecoles, membre associé de l'université de Toulouse, membre du pôle de compétitivité Aerospace Valley, du groupement des écoles d'aéronautique et l'un des cinq membres fondateurs de France Aérotech.

L'ÉNAC assure plusieurs formations orientées vers le domaine de l'aéronautique civile et en particulier vers les activités du transport aérien : compagnies aériennes, aéroports, équipementiers, construction aéronautique et spatiale, administrations et organismes français et internationaux de l'aviation civile. L'école propose environ 25 programmes de formations différents, incluant des formations d'ingénieurs (IESSA ...), de contrôleurs aériens, de pilotes de ligne, de gestionnaires, de techniciens supérieurs (TSEEAC et TSA) et d'instructeurs avions.

En accueillant chaque année plus de 7 500 stagiaires qui participent aux 600 stages annuellement organisés par l'école, avec un chiffre d'affaires de 15 millions d'euros, l'ÉNAC est aujourd'hui le premier organisme aéronautique européen de formation continue. Les activités de formation continue de l'ÉNAC se sont naturellement développées dans les domaines où l'ÉNAC a su acquérir et faire reconnaître des compétences particulières : circulation aérienne, électronique, informatique, techniques aéronautiques, pilotage avions (instructeur), etc. Ces activités s'adressent aussi bien aux entreprises et organismes français et étrangers qu'aux personnels de la direction générale de l'Aviation civile.

Moyens

Humains :
- 957 personnels permanents dont 500 enseignants et instructeurs
-1000 professeurs vacataires,

Pédagogiques :
- Une flotte de 134 avions,
- Simulateurs de vol,
- Simulateurs de contrôle du trafic aérien,
- laboratoires d'aérodynamique, d'électronique, d'informatique, de langues,
- 4 laboratoires de recherche, etc.

Implantations : Toulouse (siège administratif), Biscarosse (40), Carcassonne et Castelnaudary (11), Grenoble (38), Montpellier (34), Melun (77), Muret (31), Saint-Auban (06), Saint-Yan (71)

2. Présentation du département.

L'ENAC est divisée en plusieurs départements qui sont eux-mêmes divisés en divisions, qui sont, elles-mêmes divisées en subdivisions. Mon stage se déroule dans la subdivision RRR (réseau radionavigation radiocommunication) là où exerce mon maître de stage Christophe GUERBER. Je travaille en collaboration avec des personnes de la subdivision électronique (ELE) notamment avec Christophe DUMONT pour des conseils de conception/fabrication et des personnes de la subdivision Système Architecture Réseau (SAR). Notamment, Jean-Daniel GRIL qui s'est occupé du développement logiciel et Henri DENIS chef de la subdivision SAR.

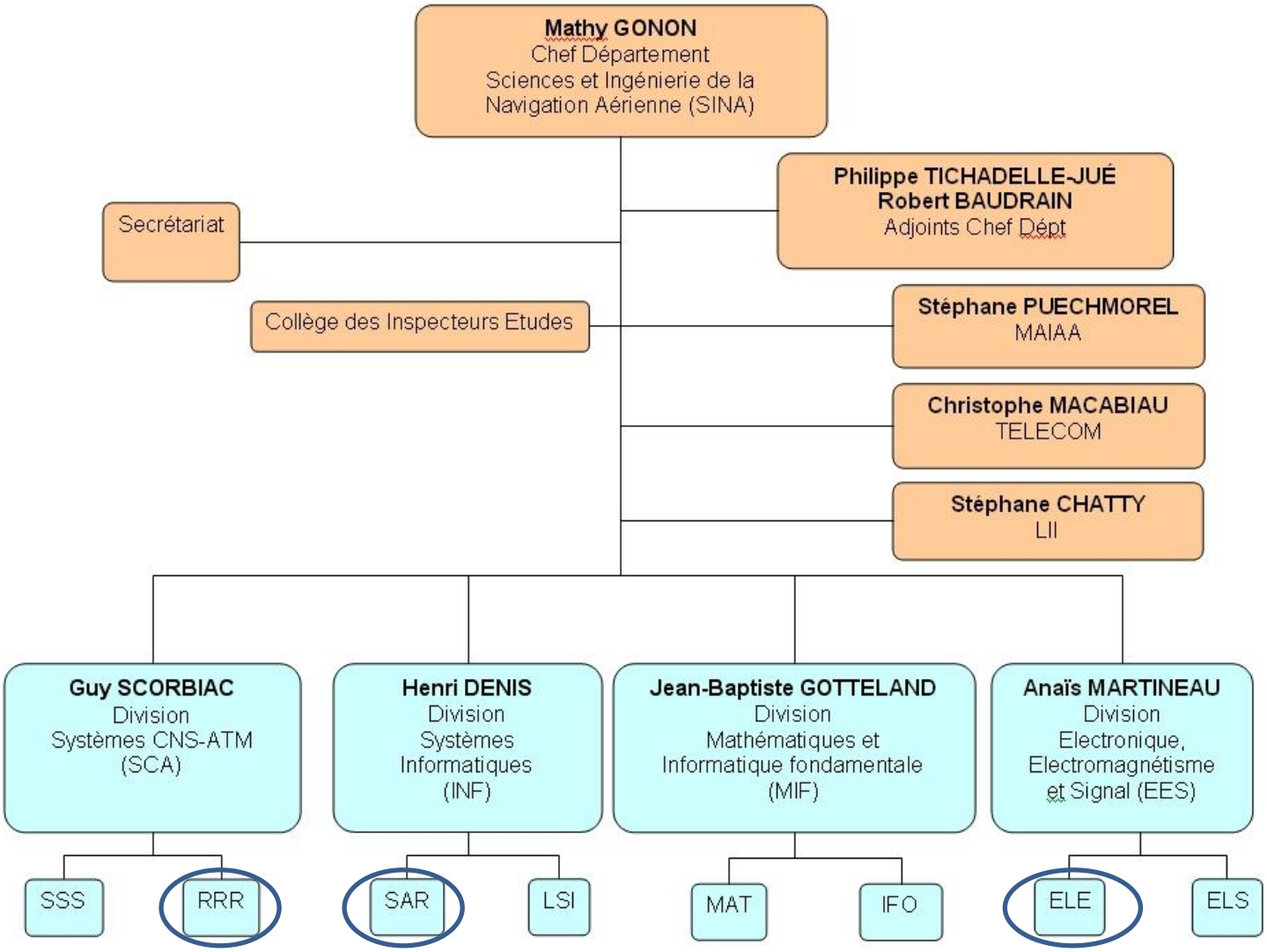


Figure 1 : Organigramme du département SINA

II. Le projet

1. Le cadre

Afin de réaliser un contrôle permanent du trafic aérien, les pilotes et les contrôleurs aériens ont besoin de communiquer entre eux : en France on utilise la VHF (Very High Frenquency). Certains espaces à contrôler sont parfois très distants des salles de contrôle, il est alors nécessaire d'utiliser des « antennes avancées » (terme qui désigne le bâtiment regroupant les émetteurs/récepteurs, les antennes…). Pour permettre un niveau de sécurité optimale, il est nécessaire de pouvoir superviser ces antennes avancées en temps réels. Ainsi il est possible de configurer les émetteurs/récepteurs selon le besoin à n'importe quel moment (émission/réception simple, doublée ou couplée). Les matériels qui assurent cette supervision sont les TC892 (Télécommande 892) et les PAE892 (Panneau d'extension 892). La TC892 permet la communication vocale entre les émetteurs/récepteurs et le centre de contrôle ainsi que la supervision des émetteurs/récepteurs. Le PAE892 est une extension permettant de retourner au centre de contrôle des informations sur l'environnement des antennes avancées (porte ouverte, climatisation activée, alimentation solaire…). Les antennes avancées sont reliées au centre de contrôle par des liaisons filaires qui transportent les informations sous format analogique.

Dans le cadre de la réglementation Ciel Unique Européen, les communications entre les antennes avancées et les centres de contrôles vont évoluer et utiliser la technologie de la voix sur IP (même principe que le logiciel Skype). Cette fonction sera assurée par les VCX (réalisés par Frequentis). La fonction de supervision sera, quant à elle, effectuée par des TCVL (Télécommande Communications Vocales et Liaisons de données) (réalisés par Telerad). La TCVL peux remplacer jusqu'à douze TC892 et six PAE892.

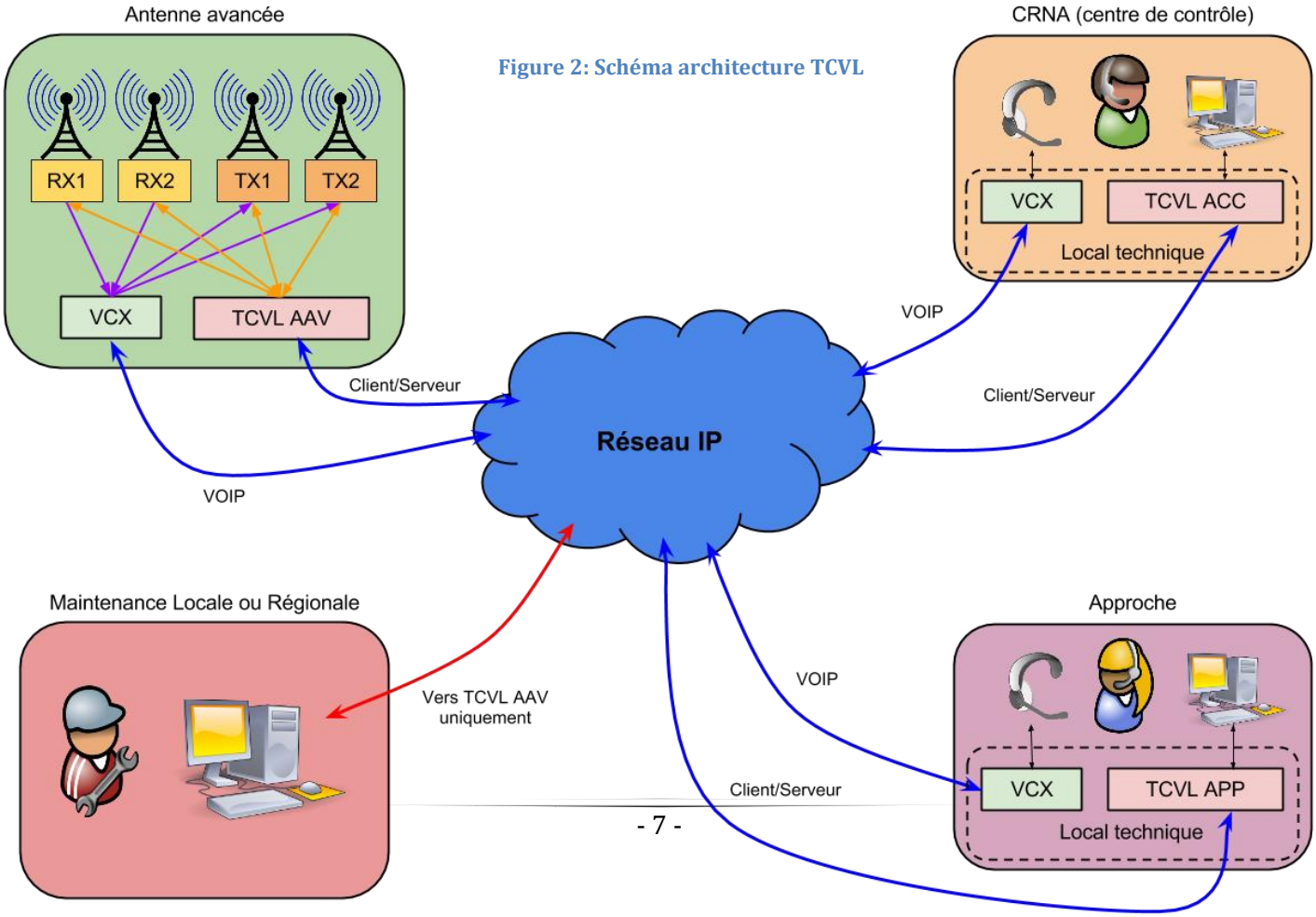


Figure 2: Schéma architecture TCVL

Des IESSA (Ingénieur Électronicien des Systèmes de la Sécurité Aérienne) sont amenés dans leur métier à utiliser et à faire de la maintenance sur des équipements TCVL. Ils doivent alors suivre une formation continue à l'ENAC où ils réaliseront des travaux pratiques sur ces équipements. Les émetteurs/récepteurs que supervisent les TCVL ne sont pas disponibles dans les salles de TP, nous devons alors trouver un moyen de simuler ces émetteurs/récepteurs.

Le but de ce stage est de réaliser une interface entre un boîtier TCVL et un PC, et de réaliser un logiciel d'interface homme-machine. Ce PC simulera plusieurs émetteurs/récepteurs afin de réaliser des travaux pratiques.

2. La réalisation

Pour la réalisation de ce boitier d'interface, j'ai procédé en plusieurs étapes : définition du cahier des charges (recherche documentaire pour bien comprendre le cadre d'utilisation de l'interface), recherches de solutions techniques, conception et fabrication.

a) Définition du cahier des charges

Cette partie se résume à une recherche documentaire sur les équipements et à des discussions avec mon maître de stage afin de clarifier au maximum le cahier des charges du projet.

Afin de choisir la meilleure solution il est nécessaire de bien connaitre le matériel, j'ai donc d'abord étudié la TCVL et fait un bilan complet des entrées/sorties grâce à la documentation du TC892 et de la PAE892. Normalement, la TCVL est compatible avec la TC892 et le PAE892.

Figure 4 : TCVL face avant

Figure 3 : TCVL (2)

- Les interfaces du TCVL:

La TCVL possède douze ports DB37 mâles (figure 6 page 7), un port DB37 est équivalent à une TC892, il comporte aussi six ports DB9 (figure 7 page 7), un port étant équivalent à un port d'un PAE892.

Deux autres connecteurs DB9 mâle et femelle permettent à un utilisateur de configurer la TCVL grâce au protocole JBUS. Mais généralement la configuration se fait à distance grâce au port Ethernet qui est connecté au réseau IP. Ces deux connecteurs ne sont en fait pas implémentés, mais ils peuvent être un axe d'évolution de la TCVL.

La TCVL est alimentée en 24V via un connecteur CTA 3 broches mâles. Nous ne disposons pas de ce connecteur. Pour la réalisation des TP, il faut nous en procurer cinq sur le moyen terme (5 mois), et au moins un immédiatement pour pouvoir effectuer des tests. Nous avons contacté le distributeur Farnell pour en commander, mais ces connecteurs ne seraient disponibles qu'après un délai de six mois et avec une quantité minimum de vingt exemplaires (à une vingtaine d'euros l'unité). Nous avons cependant réussi à trouver un connecteur, à la DTI (Direction Technique et Industrielle de l'Aéronautique).

Figure 5 : Connecteur d'alimentation

Le but du projet est de pouvoir réaliser toutes les configurations possibles d'interconnexions entre les TCVL et les émetteurs/récepteurs, pour cela il est nécessaire qu'un TCVL puisse gérer au minimum trois antennes (soit trois connecteurs DB37) et un PAE avec 4 entrées. Cela semble suffisant pour les besoins des TPs.

Pour un connecteur DB37 (TC892) :

- 11 E/S, 6 sorties et 5 entrées (8 E/S utile pour la TCVL)

Seuls les signaux de commandes marche/arrêt et les signaux des défauts sont traités par la TCVL.

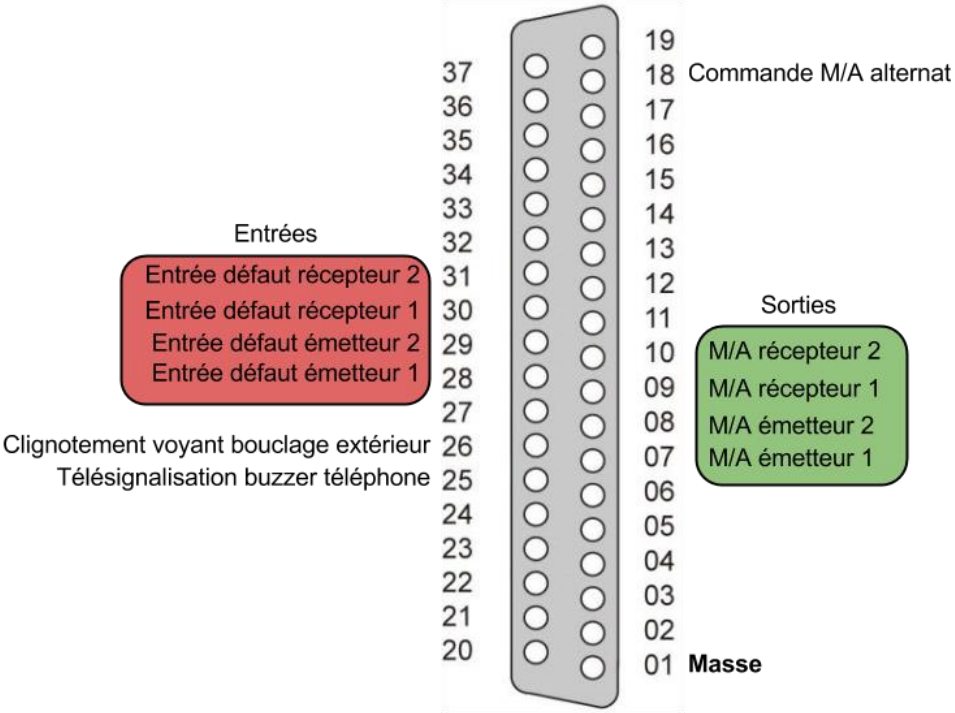


Figure 6 : Connecteur TC892 DB37

Connecteur Sub-D 9 (PAE892)

- 8 entrées

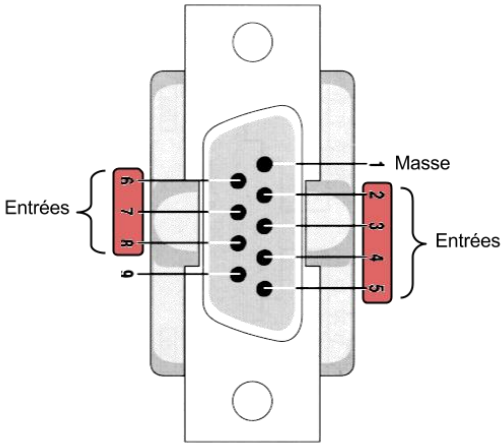


Figure 7 : Connecteur PAE892 DB9

Seules 4 entrées seront utilisées pour les Travaux pratiques des IESSA.

Il y a donc en tout pour un boitier TCVL (trois connecteurs DB37 et 1 connecteur DP9), 12 sorties et 16 entrées qu'il faut interfacer.

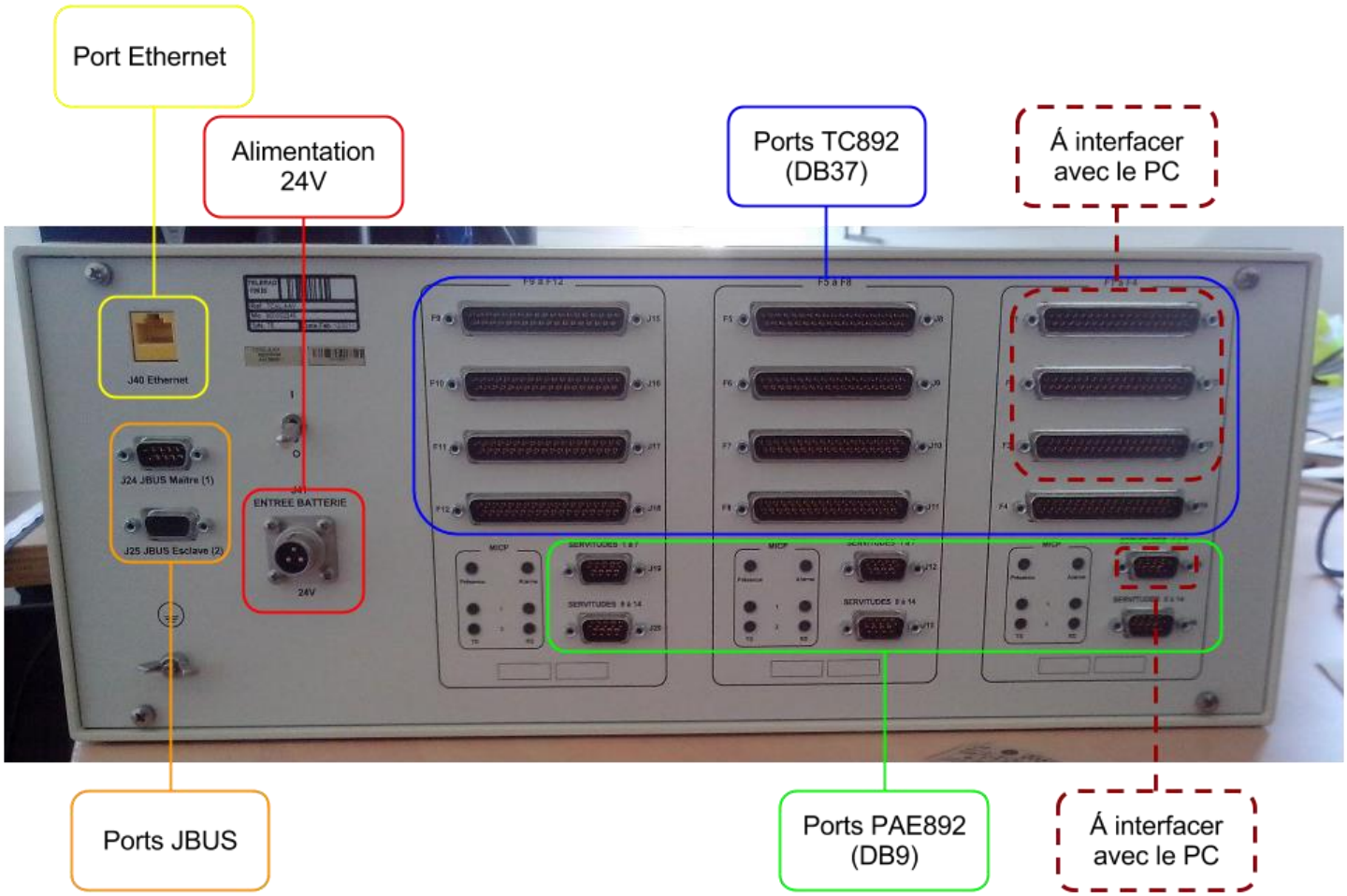


Figure 8 : Face arrière TCVL

b) Recherche de solutions techniques

Nous avons trouvé deux solutions qui peuvent répondre au cahier des charges :

- Utilisation d'interrupteurs et de LED pour interagir sur les entrées/sorties du TCVL.

Des LED et des interrupteurs sont câblés directement aux entrées/sorties du TCVL.

- Pas de programmation de composant étant donné que tout est câblé directement (les sorties du TCVL à des LED et les entrées à des interrupteurs)
- Prix des interrupteurs
- Trop d'interrupteurs et de LED nécessaires (16 interrupteurs et 12 LED).

- Utilisation d'un ordinateur pour contrôler la TCVL

Pour cela il faut utiliser une interface entre le PC et la TCVL. Cette interface a pour rôle d'adapter les niveaux de tension (passer des sorties 24V du TCVL en 5V pour les entrées d'un microcontrôleur et inversement). Il doit aussi établir une liaison entre les E/S du TCVL et le port USB du PC, pour cela il doit pouvoir recueillir les données envoyées par le PC, les décoder, écrire sur les E/S du TCVL et inversement.

La deuxième solution a été choisie, car son coût est inférieur à la première et parce qu'elle offre un moyen plus flexible de configuration ainsi que plus d'évolutivité.

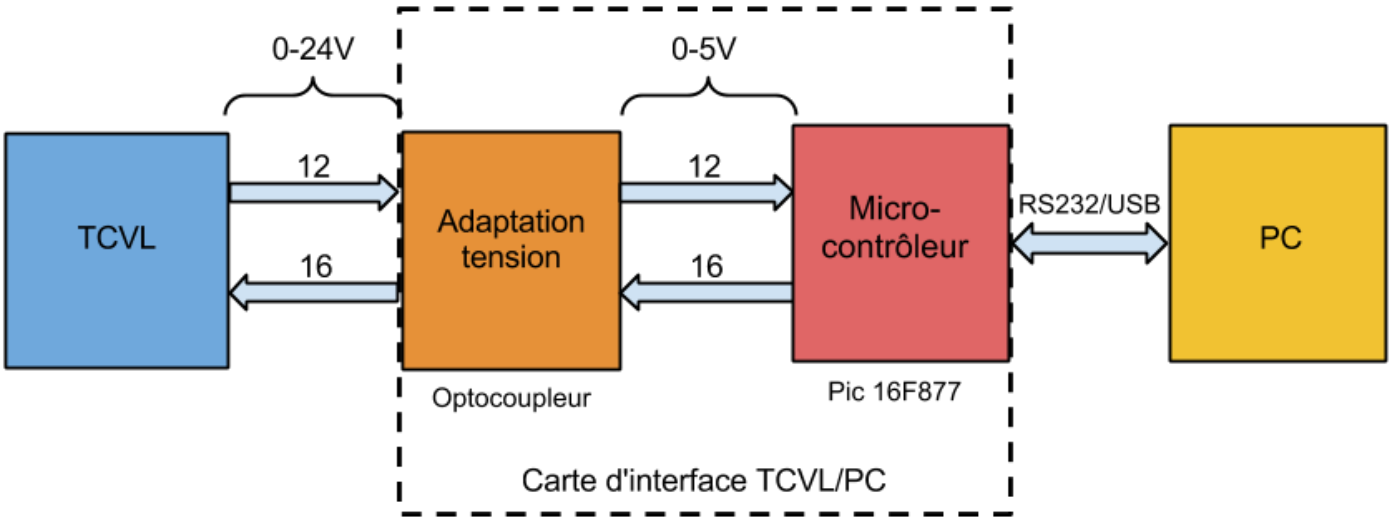


Figure 9 : Synoptique du système

1. Adaptation des tensions

Nous avons utilisé des optocoupleurs afin d'établir des liaisons dotées d'une isolation galvanique. Le schéma d'adaptation est assez simple à mettre en œuvre du fait qu'il n'y a pas besoin de beaucoup de composants.

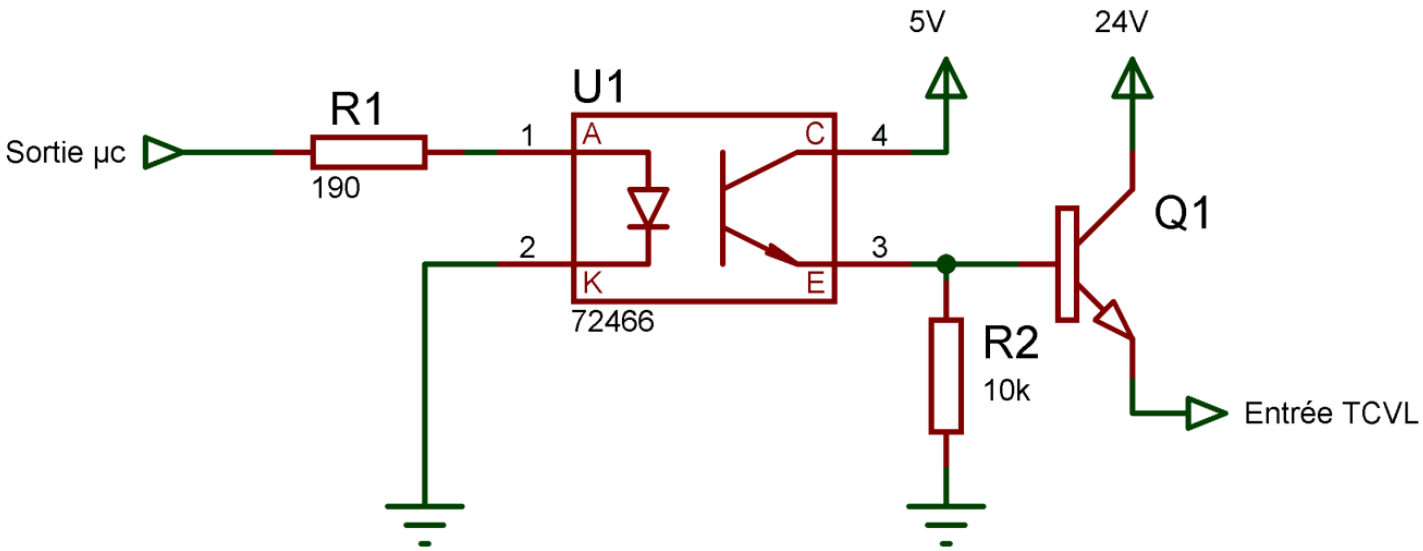


Figure 10 : Interface Sortie microcontrôleur

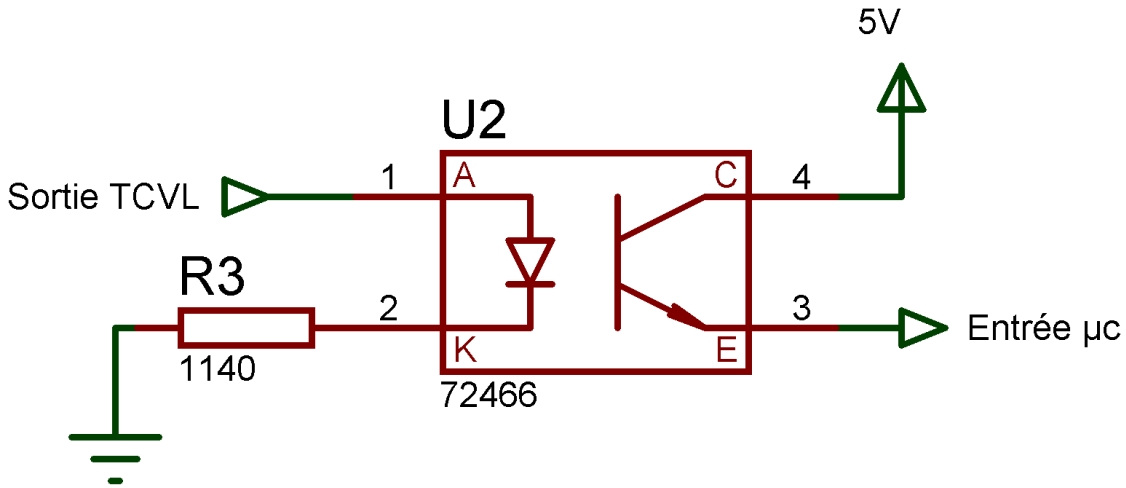


Figure 11 : Interface Entrée microcontrôleur

2. Le microcontrôleur

Après avoir fait des recherches sur des microcontrôleurs, je me suis aperçu que la plupart des microcontrôleurs avec plus de 28E/S coûtent relativement cher. Je me suis alors tourné vers une solution qui consiste à utiliser un microcontrôleur couplé à un expanseur I2C.

Un expanseur I2C sert à ajouter des E/S supplémentaires à un microcontrôleur grâce à une communication I2C. Cette communication utilise seulement deux fils ce qui est un avantage, d'autant plus qu'il est possible de s'en procurer gratuitement en tant qu'échantillons chez Maxim Integrated (fabriquant de circuits intégrés).

L'expanseur I2C que nous allons utiliser est le MAX7300AAX : 28 E/S, 36 broches, package SSOP36. Chaque broche peut être configurée indépendamment, les entrées peuvent disposer d'une résistance de pullup interne, ce qui permet d'économiser des résistances.

Figure 12 : Max7300

Pour le choix du microcontrôleur, ne connaissant pas au début les microcontrôleurs disponibles dans la subdivision électronique, mon choix s'est porté sur un petit microcontrôleur à 8 broches, l'ATtiny85 de chez Atmel. Il dispose de 6 E/S, ce qui est suffisant pour notre cas, car nous n'avons besoin que de 4 E/S (2 pour le bus I2C et 2 pour la communication série avec l'ordinateur). Ce microcontrôleur est compatible avec le logiciel Arduino, un logiciel libre et gratuit qui permet de programmer aisément des microcontrôleurs en C, grâce aux nombreuses librairies disponibles. Pour ce qui est de la plateforme de développement seul un convertisseur USB vers série RS232 TTL (ftdi) est nécessaire. Son coût est d'environ 1,50€ l'unité chez Farnell.

Cependant, on m'a proposé d'utiliser un microcontrôleur PIC (16F88), car tout le matériel pour la mise en œuvre est disponible, et en cas de problème je pouvais recevoir de l'aide plus facilement. Ce microcontrôleur comporte 18 broches et a un module UART (pour la communication série RS232) et I2C intégrés.

Ce PIC est programmé en assembleur grâce au logiciel MPLAB, déjà utilisé à l'IUT, mais avec la version en C. Comme ce Microcontrôleur est utilisé dans l'enseignement des IESSA j'ai eu accès à de nombreux cours détaillés et à des explications données par professeur Christophe DUMONT.

Afin de prendre en main le microcontrôleur, j'ai réalisé des programmes simples sur les maquettes de TP (chennillard à LED, communication I2C avec un RTC (Real Time Clock), un PIO (expanseur I2C), communication série avec le PC).

Figure 13 : PIC16F88

3. *La liaison entre le microcontrôleur et l'ordinateur*

Le microcontrôleur communique grâce à une liaison série type RS232 niveau TTL (un 1 logique est représenté par une tension de 5V et un 0 logique par une tension de 0V). Il existe des câbles USB avec une sortie RS232 TTL à environ 2€ (pas chez Farnell ou Radiospares, mais sur ebay à 12€ les 5). Cependant, on ne peut pas acheter facilement sur des sites autres que Farnell et Radiospares.

Figure 14 : Câble convertisseur USB/Série

Après quelques discussions, il s'est avéré que l'on disposait déjà de câbles convertisseurs USB ver RS232, mais pas TTL, il faut donc adapter les signaux du microcontrôleur. Pour cela j'ai utilisé un max232, un circuit intégré servant à adapter les niveaux de tension.

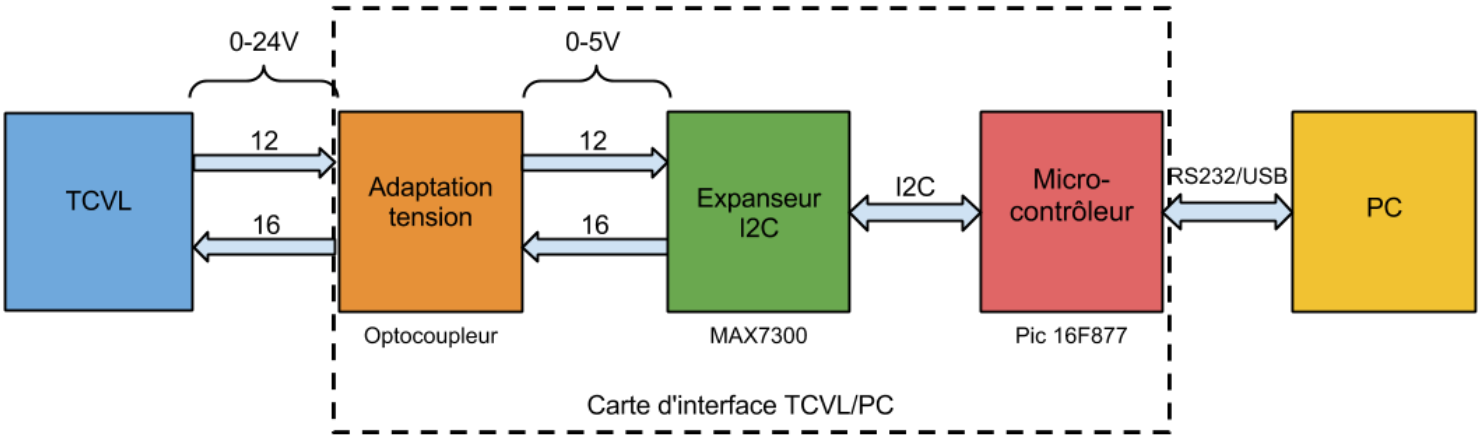


Figure 15 : Synoptique carte d'interface

c) Conception

La conception des schémas électroniques a été effectuée sur le logiciel ISIS de la suite PROTEUS. J'ai utilisé ce logiciel, car c'est un outil que je connais, je l'ai utilisé en première année et lors de projets personnels pour réaliser des cartes.

J'ai beaucoup appris sur ce logiciel en très peu de temps (création de composants et de leurs empreintes, utilisation de label, maniabilité en général, etc.).

Après avoir fait la carte sur l'ordinateur, j'ai pu faire une approximation de sa taille et choisir un boîtier dans le stock de la subdivision électronique. J'ai ensuite modélisé cette boîte sur l'ordinateur grâce au logiciel SolidWorks. Grâce à cette « numérisation », j'ai pu exporter des plans précis dans le logiciel Ares (logiciel de la suite Proteus qui permet de router les pistes sur un circuit). Ceci m'a permis de router les pistes en ayant une taille de carte définie et d'indiquer les zones de perçage (afin d'éviter de tracer des pistes par-dessus).

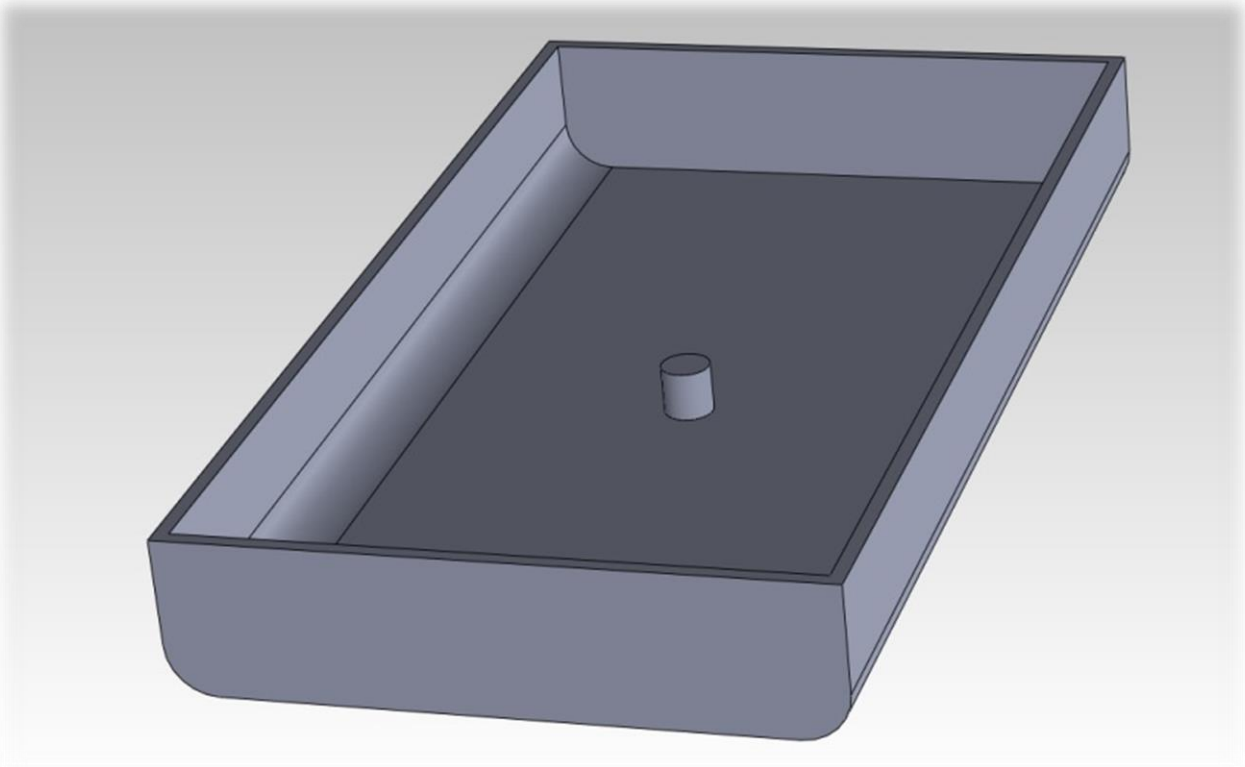


Figure 16 : Boîtier (sans le couvercle) réalisé avec SolidWorks

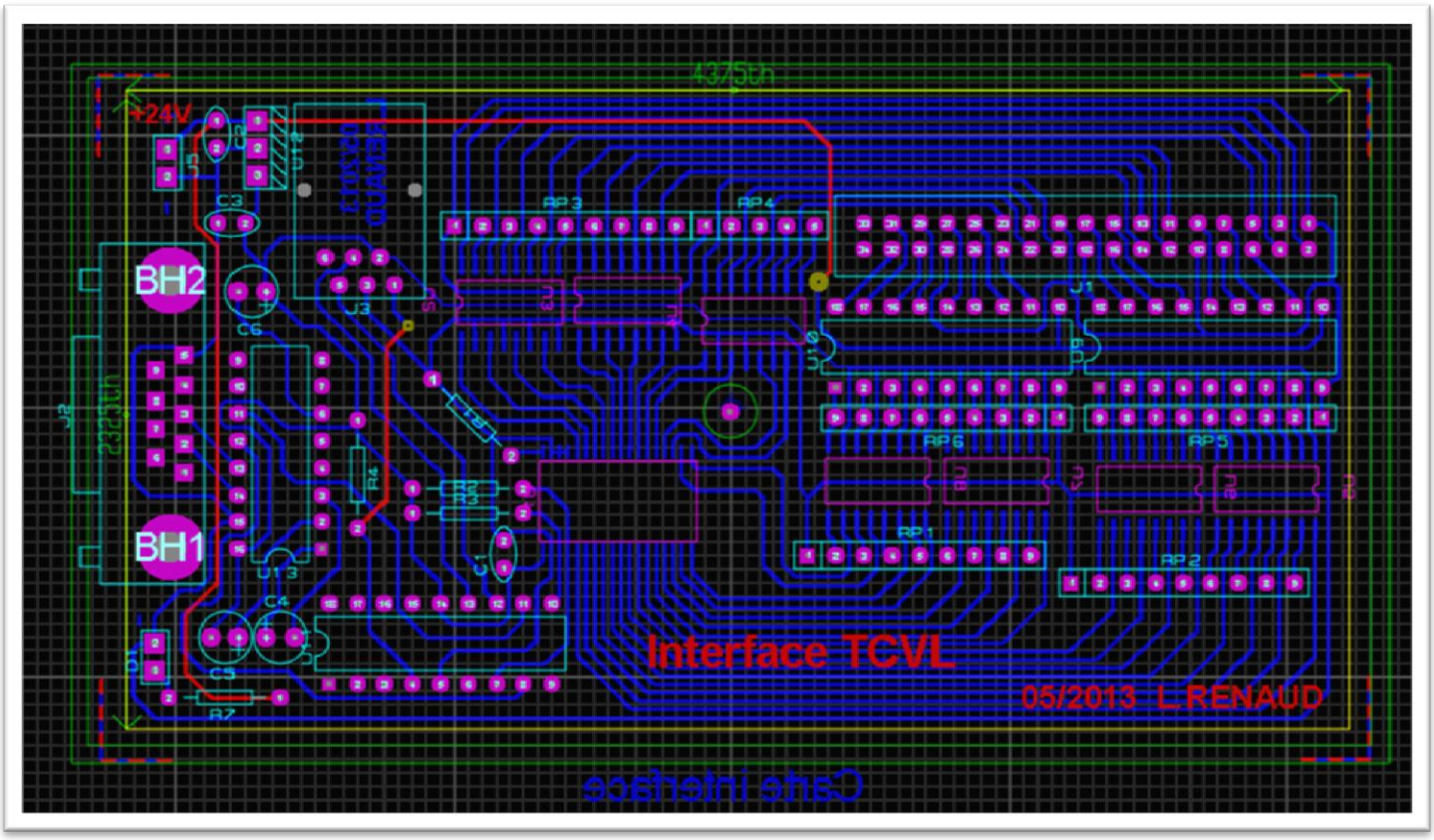


Figure 17 : Routage carte d'interface (Proteus)

Image du routage de la carte (logiciel ARES)

- En vert, le tracé du boîtier
- En jaune, le bord de la carte
- En bleu, les pistes en dessous de la carte
- En rouge, les pistes au-dessus de la carte (remplacées par des fils pour ne pas avoir à utiliser une plaque double faces)
- En violet les composants sous la carte (les CMS)
- En bleu clair les composants sur la carte

(Le schématique complet est disponible en annexe page XX)

Cette version est proche de la version finale, elle a subi de nombreux changements au cours de l'avancement du projet.

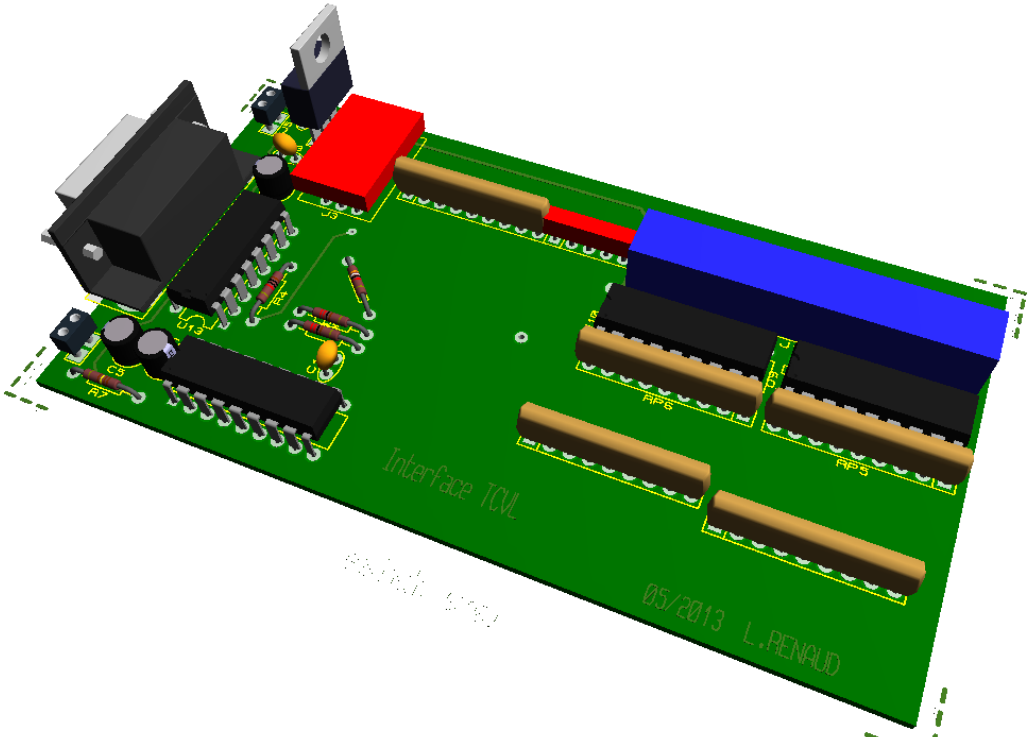


Figure 18:Vue 3D du dessus de la carte (sous ARES)

Une fois les plans terminés, nous avons commandé les composants chez Farnell et Maxim

Etant donné que la commande chez Maxim ne concernait des échantillons gratuits, celle-ci s'est faite rapidement sans passer par l'approbation des supérieurs. Le colis est arrivé dans la semaine suivante. Chez Farnell la commande a mis un peu plus de temps, le plus long est d'attendre l'approbation de la commande (elle nous a pris environs deux semaines) ensuite la commande en elle-même est très rapide, elle est arrivée le lendemain.

Figure 20 : Composants de chez Farnell

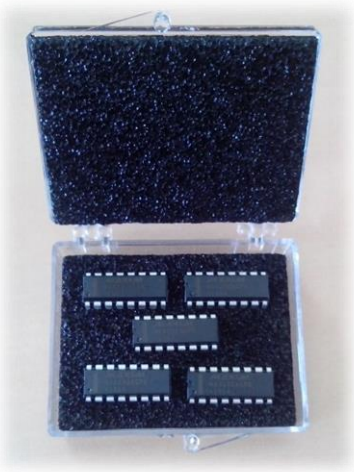


Figure 19 : MAX232 (MAXIM)

Figure 21 : MAX7300 (MAXIM)

d) La fabrication

Après avoir reçu tout le matériel commandé chez Farnell, j'ai pu commencer la phase de fabrication.

1. Fabrication du circuit imprimé

Une fois les typons imprimés sur des transparents, je suis allé tirer les cartes avec Christophe DUMONT.

Premièrement on pose la carte photosensible et le typon dans l'insoleuse, c'est un dispositif qui éclaire la carte avec des ultraviolets pendant un laps de temps défini. Les ultraviolets vont enlever l'encre sur la couche photosensible qui n'est pas protégée par le typon.

Figure 22 : Insoleuse

On passe la carte dans un bac rempli de révélateur, ce qui sert à enlever complètement la couche photosensible qui n'a pas été protégée.

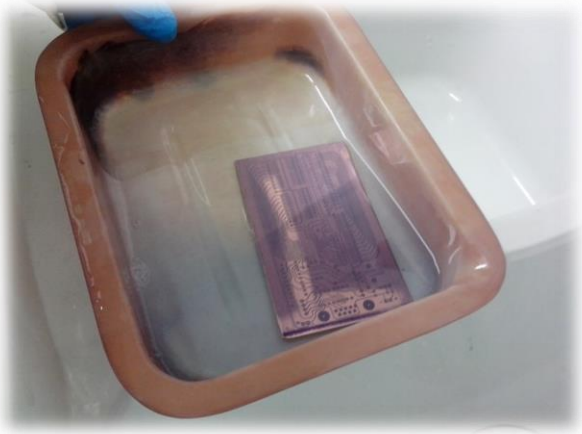


Figure 23 : Carte dans bac de révélateur

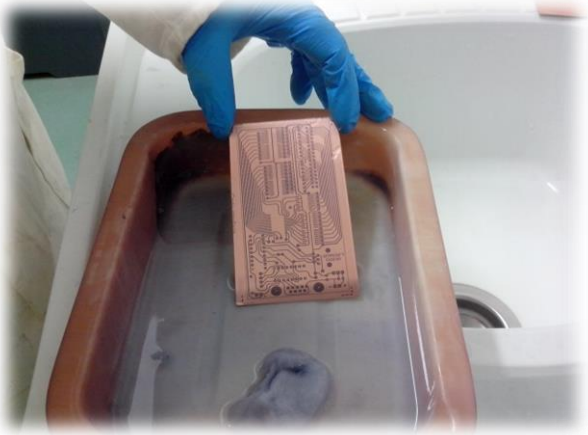


Figure 24 : Carte dans le bac de révélateur(2)

La carte est ensuite insérée dans une graveuse, c'est une machine qui projette du perchlorure de fer sur la carte. Une réaction chimique se produit entre le perchlorure de fer et le cuivre qui n'est pas protégé par la couche photosensible, le cuivre réagit complètement jusqu'à ce qu'il n'y en reste plus (exepté sur les zones où le cuivre est protégé), la carte est ensuite rincée à l'eau .

Figure 25 : Graveuse

On nettoie ensuite la carte avec du solvant pour enlever complètement la couche photosensible sur le cuivre.

Enfin on met la carte dans un bac pour faire de l'étamage à froid. Une fois sortie du bain les pistes de la carte sont recouvertes d'étain, pour empêcher que le cuivre ne s'oxyde.

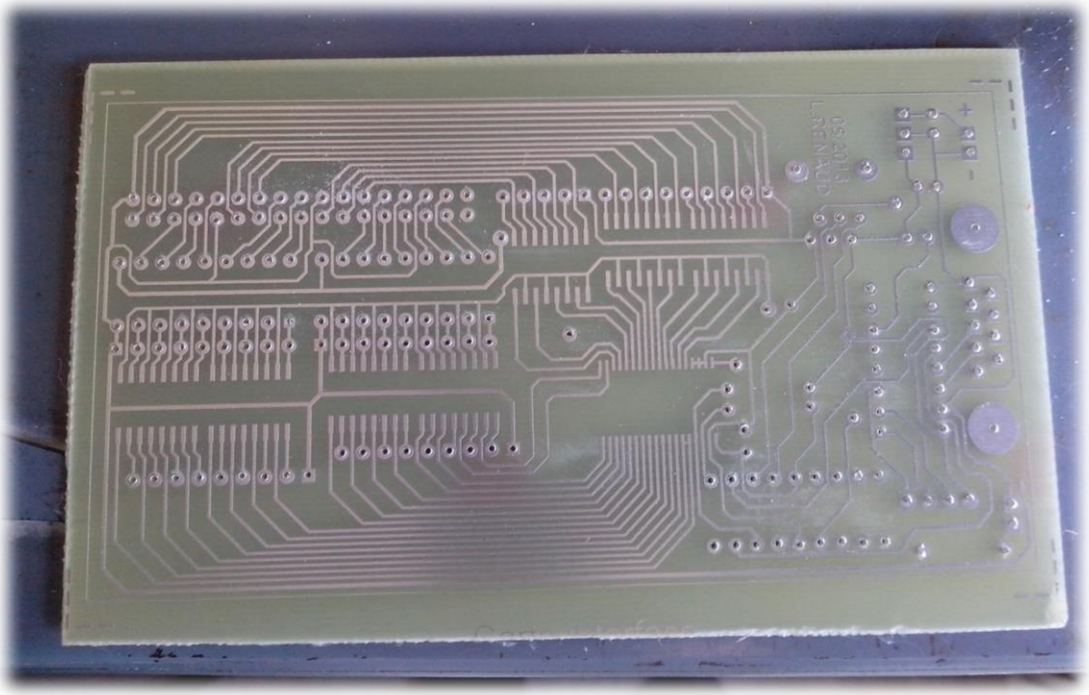


Figure 26 : Circuit imprimé

Une fois terminé, il ne reste plus qu'à percer des trous. L'ENAC est équipé d'une machine peu commune pour effectuer ce travail.

La carte est percée par le dessous, en appuyant sur une pédale. Il y a un rétroprojecteur qui projette la carte sur un écran rond. Il faut juste centrer la carte au centre à l'aide du réticule sur l'écran, et appuyer sur la pédale pour percer. Avant que le foret sorte pour percer la carte, la carte est immobilisée grâce à un vérin.

Cette machine permet de percer des trous rapidement avec une très grande précision.

Figure 28 : Perceuse

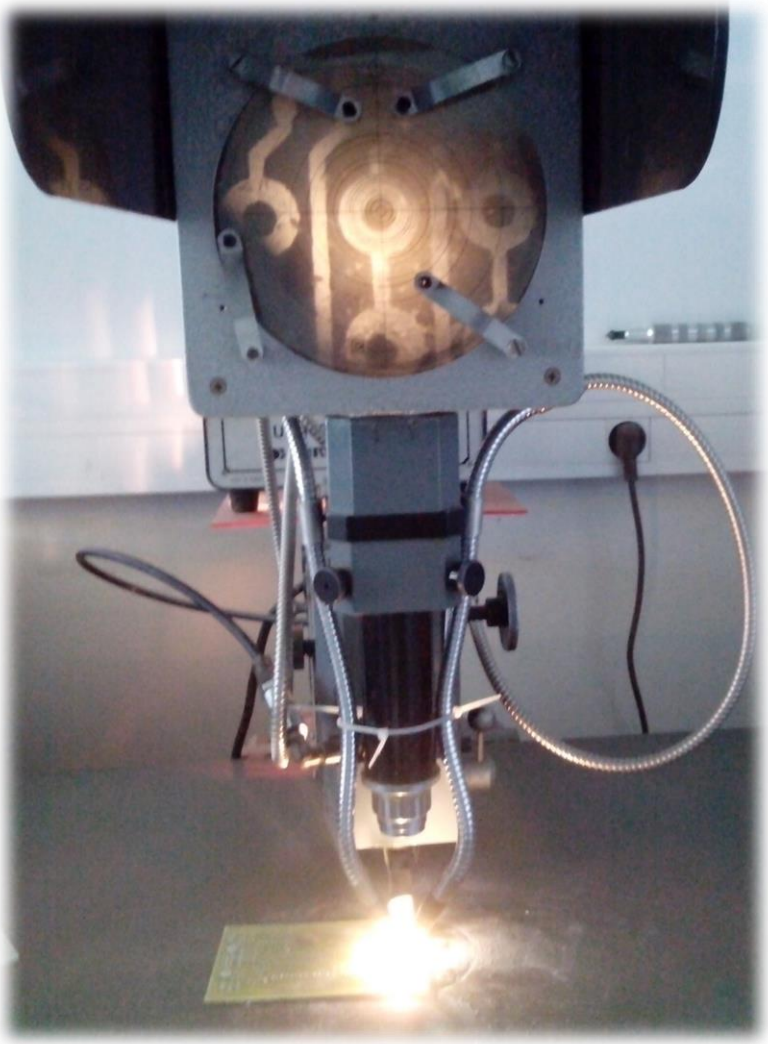


Figure 27 : Perceuse

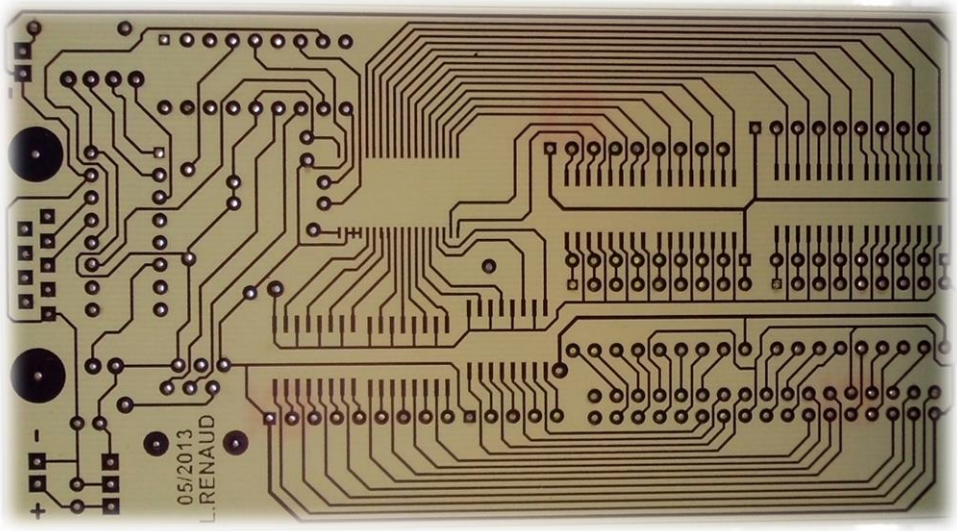


Figure 29 : Circuit imprimé terminé

2. Soudure des CMS

J'utilise des composants aux formats CMS (l'expanseur I2C et les optocoupleurs), ils ont l'avantage d'être très petits, mais très difficiles à souder au fer. J'ai donc utilisé un four à refusions pour les souder.

Tout d'abord il faut mettre de la pâte à souder sur les pistes. C'est la partie qui demande le plus d'attention, car il faut être très précis sur l'endroit où déposer la pâte et sur la quantité à mettre. J'ai effectué cette opération avec un binoculaire ce qui a amélioré la visibilité.

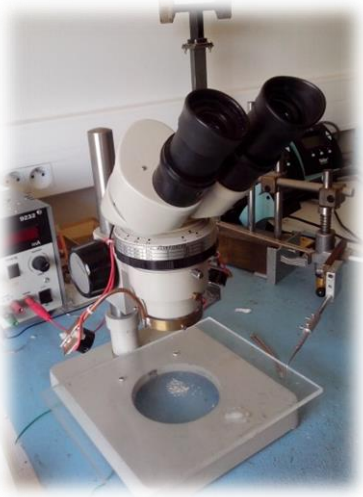


Figure 31 : Binoculaire

Figure 30 : Four à refusions

Une fois la pâte déposée il faut positionner les composants sur la plaque avec une pince à épiler. Ensuite il ne reste plus qu'à mettre la plaque dans le four et à attendre 5 minutes. Le four suit une courbe de chauffe précise qui est définie selon la composition de la pâte à souder utilisée.

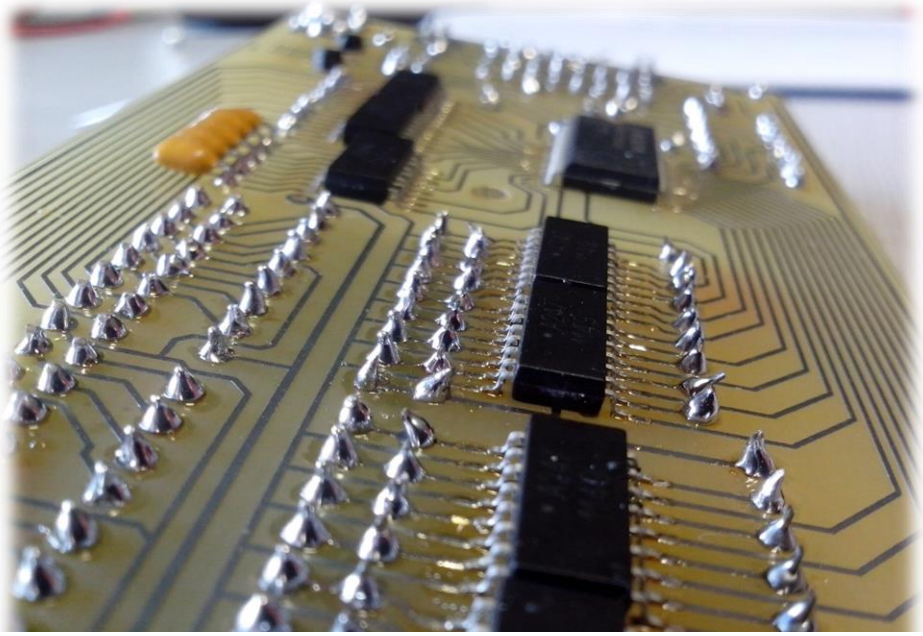


Figure 32 : Soudures carte d'interface

3. Soudure des composants

La soudure des composants de taille DIP se fait de manière traditionnelle avec un fer à souder et de l'étain.

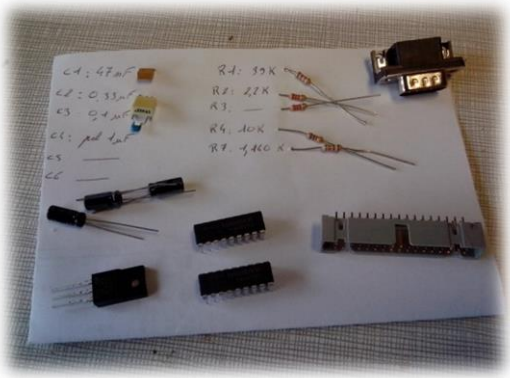


Figure 34 : Composants à souder

Figure 33 : Fer à souder

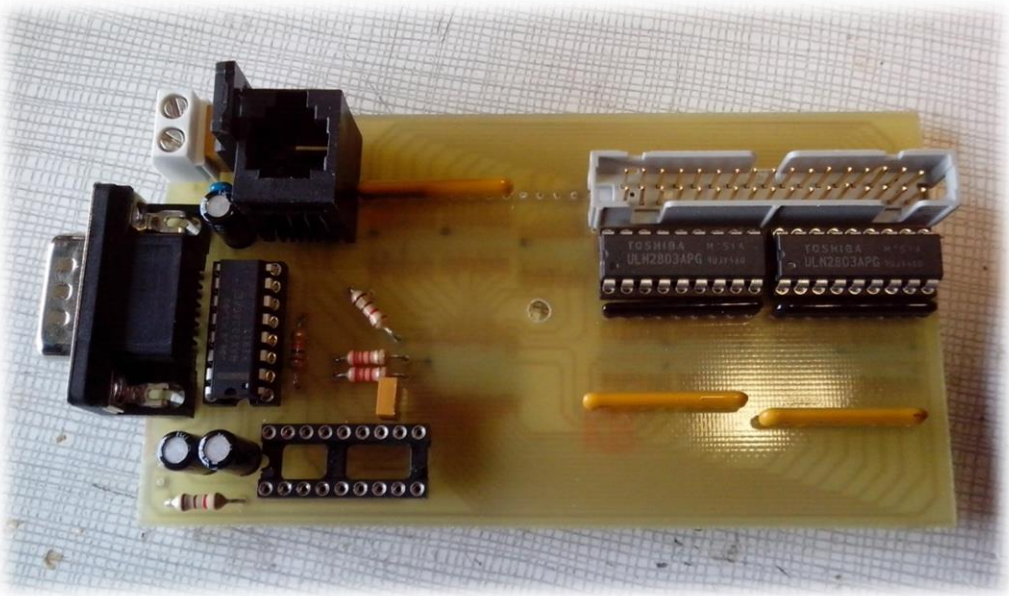


Figure 35 : Carte d'interface avec ces composants

4. Mise en boîte

Afin de protéger la carte contre toutes les agressions extérieures, il est nécessaire de protéger la carte grâce à un boîtier. J'ai commencé à usiner la boîte pour qu'elle puisse accueillir la carte avec ces connecteurs. J'ai réalisé l'ouverture pour le connecteur du port série au cutter et à la lime, et les trous pour les fils d'alimentation et de la LED à la perceuse à colonne. Après avoir positionné la carte dans la boîte, j'ai eu l'agréable surprise de voir que tout rentrait parfaitement. Le connecteur du port série arrive exactement là où je l'avais prévu et le trou pour la vis se trouve bien en face du plot.

Figure 36 : carte d'interface dans le boîtier

5. Fabrication du cordon

J'ai fabriqué un câble afin de relier la carte d'interface et le boîtier TCVL. Le câble part de la carte d'interface et se divise en quatre, 3 connecteurs DB37 et un DB9.

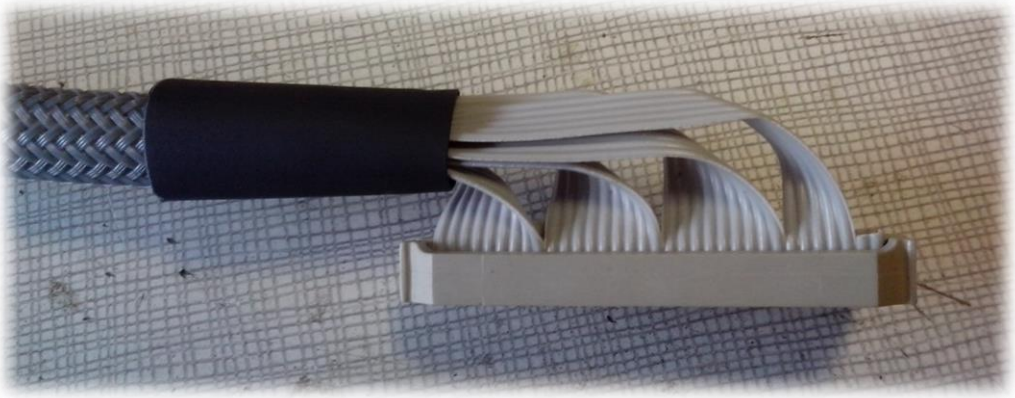


Figure 37 : Connecteur (HE10) qui vient se connecter sur la carte d'interface

Le câble plat est protégé par une gaine tressée en nylon, puis de la gaine thermo rétractable est mise au bout pour fixer la tresse et éviter qu'elle s'effiloche.

Figure 38 : Cordon (1)

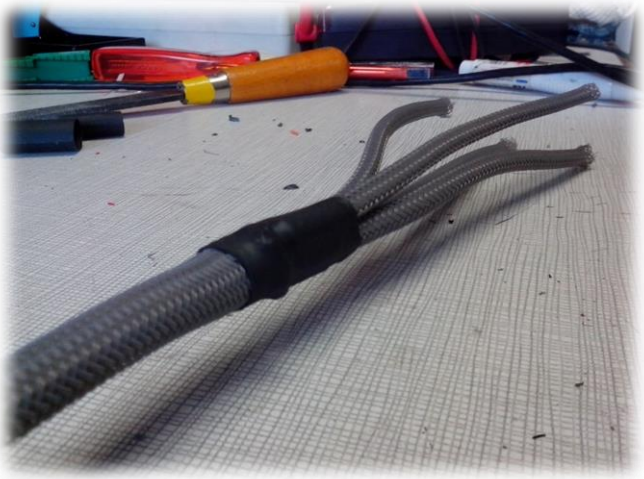


Figure 39 : Cordon (2)

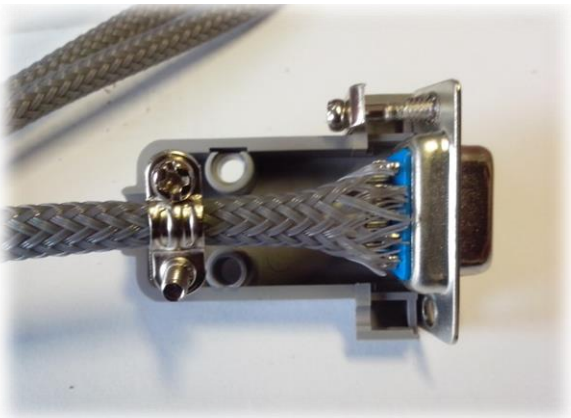


Figure 41 : Cordon (connecteur DB9)

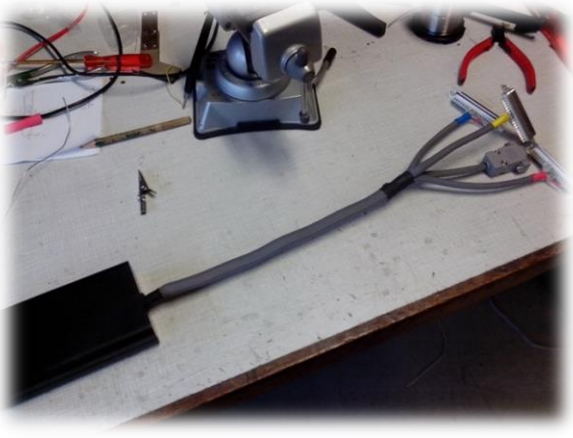


Figure 40 : Cordon terminé

e) Programmation du microcontrôleur

Maintenant que la carte est fabriquée, il faut à présent la programmer. Pour cela j'ai d'abord fabriqué une carte module de l'expanseur I2C qui s'adapte aux maquettes de TP de l'ENAC. Grâce à elle j'ai pu m'entrainer à configurer et à utiliser l'expanseur I2C en reliant directement les entrées sorties de l'expanseur aux LED ou aux interrupteurs de la maquette.

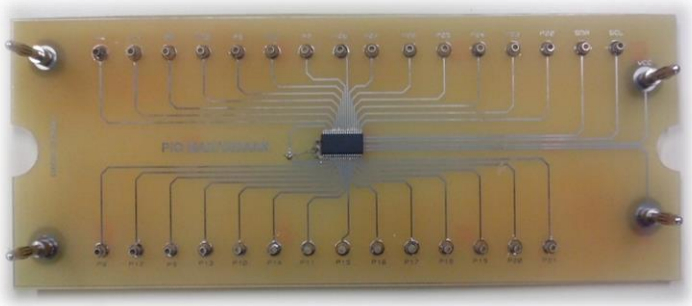


Figure 42 : Module MAX7300

J'ai eu des problèmes pour programmer le PIC 16F88, car sa structure interne n'est pas exactement la même que le 16F877, (microcontrôleur que j'ai programmé dans un premier temps pour me familiariser avec le matériel). En effet le 16F88 ne gère pas l'I2C en tant que maître de manière matérielle comme le 16F877, il a donc fallu implémenter l'I2C de manière logicielle.

Pour la mise au point de la communication I2C entre le microcontrôleur et l'expanseur, j'ai pu utiliser un analyseur logique afin de vérifier les trames échangées sur la liaison.

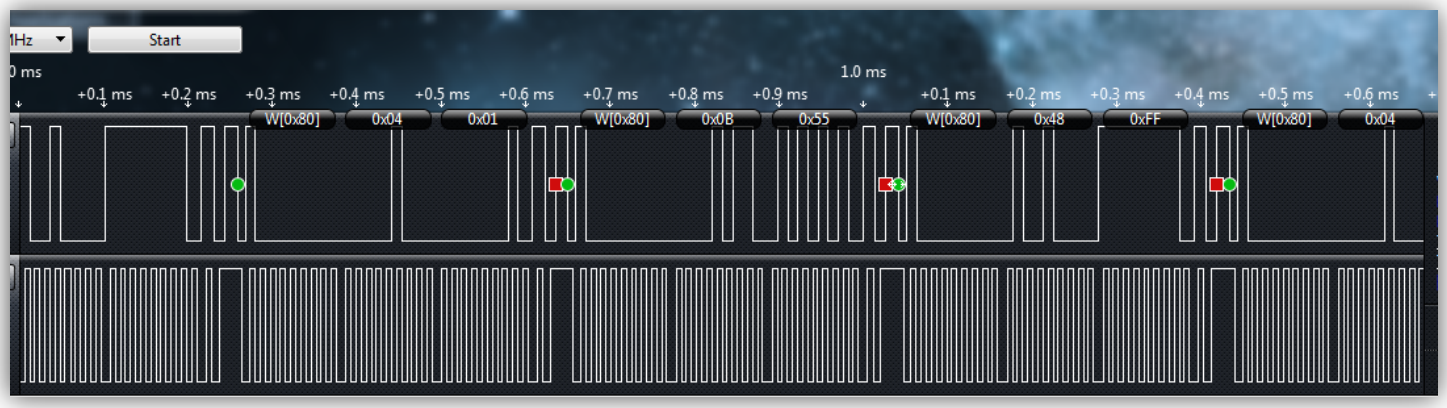


Figure 43 : Capture d'écran du logiciel « Logic » qui permet de visualiser les deux signaux de la liaison I2C

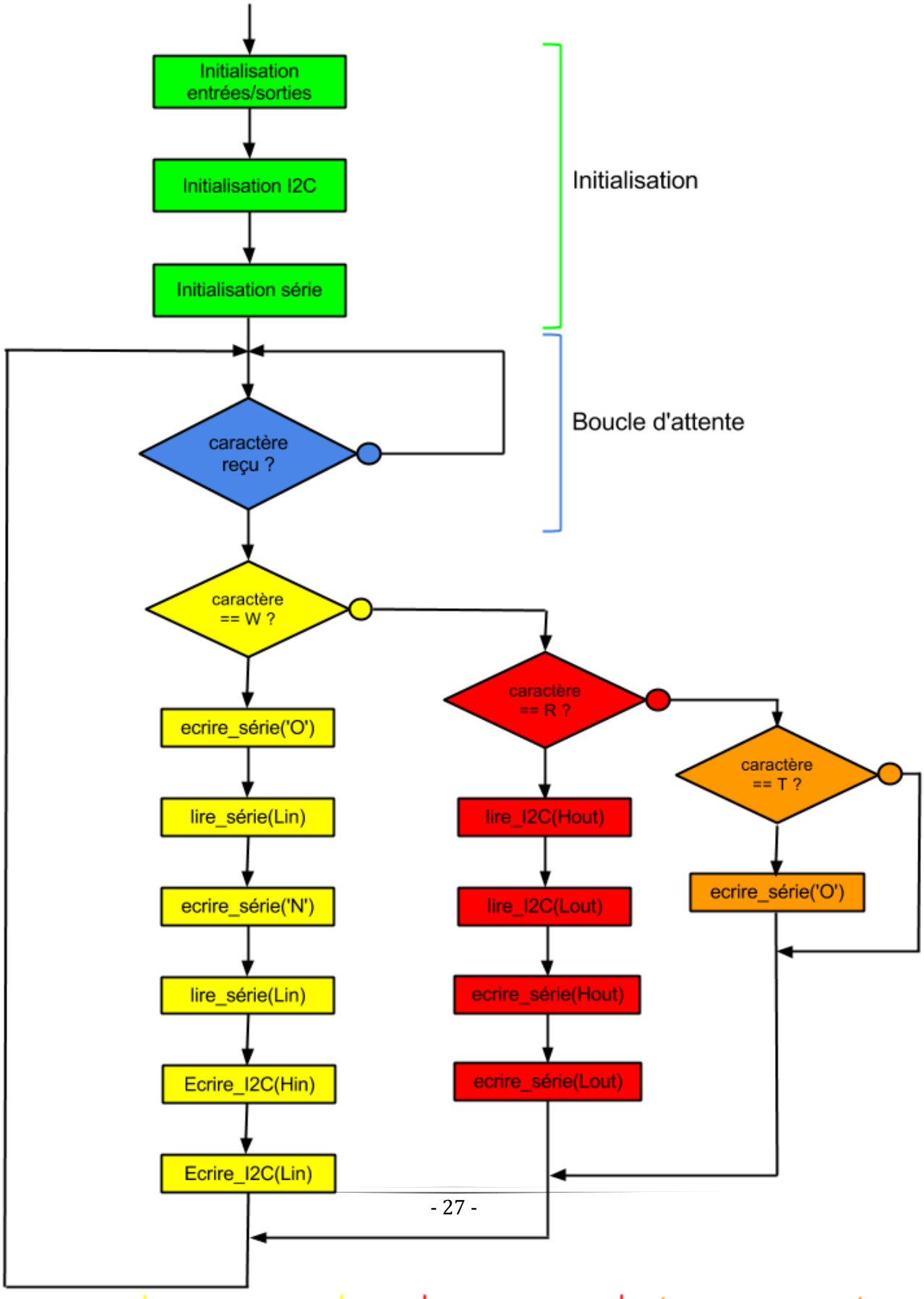
Ce dispositif m'a beaucoup aidé pour le débogage, il permet de visualiser directement en hexadécimal le contenu des trames I2C et indique aussi les conditions de Start et de Stop (en vert en on rouge sur la capture d'écran si dessus).

Après de nombreux essais, j'ai enfin réussi à configurer et à échanger des informations correctement avec l'expanseur I2C.

J'ai eu des difficultés au niveau des interruptions du microcontrôleur sur le 16F88 : elle détruisait la liaison I2C (mais pas sur le 16F877). Après de longues heures à essayer de corriger ce problème avec Christophe Dumont, nous avons décidé de ne pas utiliser l'interruption. Selon Jean-Daniel Gril (Collègue de Christophe Guerber, nous travaillons tous les trois dans le même

bureau), le problème venait peut-être du fait qu'on ne sauvegardait pas toutes les pages de la mémoire.

Algorithme microcontrôleur



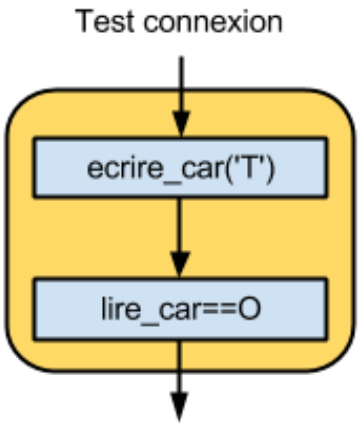
3. Programmation ordinateur

a) Le protocole de communication entre la carte d'interface et l'ordinateur

Afin de pouvoir communiquer avec la carte d'interface, il est nécessaire d'établir un protocole de communication. Il a pour but de définir les réactions des appareils à chaque bout de la liaison. Ayant beaucoup travaillé sur la liaison I2C, j'ai décidé que l'ordinateur pourrait interagir avec la carte d'interface en utilisant la notion de maître/esclave (comme la liaison I2C). C'est l'ordinateur, le maître, qui interroge la carte d'interface (l'esclave).

L'ordinateur à trois actions possibles

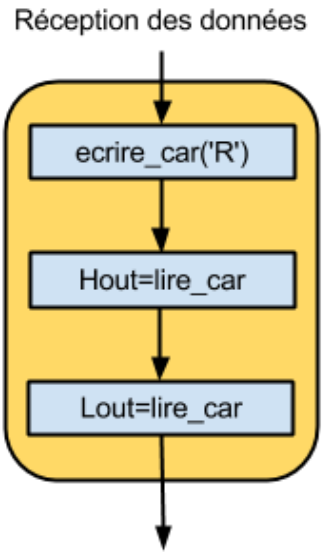
- Tester la connexion
  - Permet de s'assurer que le boitier d'interface est bien connecté.
  - Permet d'effectuer une détection automatique du numéro de port où est connectée la carte d'interface. Lors du lancement du logiciel, le logiciel envoie un caractère sur tous les ports série disponibles. Si la carte d'interface est connectée, elle va renvoyer un caractère défini. Ceci permet de connaitre le numéro de port et d'identifier la carte d'interface.



L'ordinateur envoie le caractère T sur l'interface série

La carte d'interface répond en envoyant le caractère O.
(Le caractère renvoyé (ici un O) permet d'identifier la carte.)

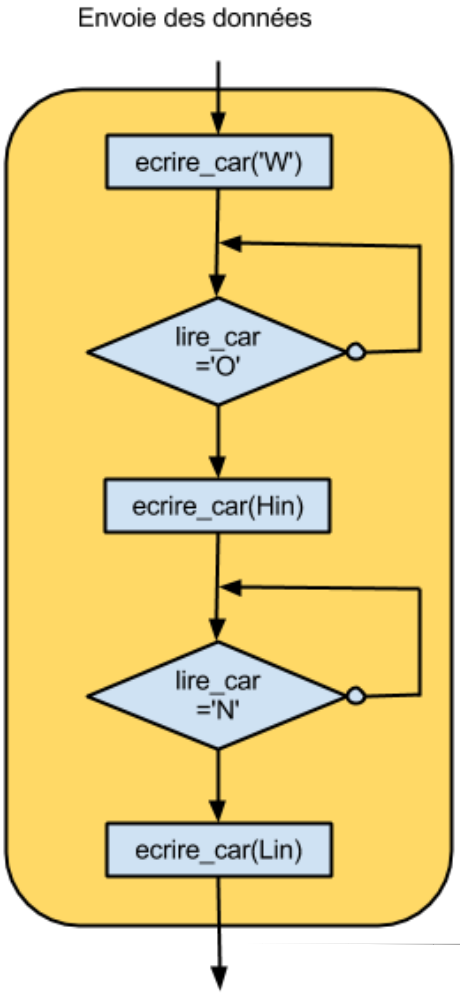
- Lire les données du TCVL
  - Permet de connaître les ordres envoyés par la TCVL



Pour lire, le PC envoie le caractère R (Read).

Il reçoit ensuite les deux octets qui contiennent les données

- Envoyer des données
  - Envoie les informations des interrupteurs à l'écran sur les entrées du TCVL



Envoie le caractère W (Write)

Attend que la carte d'interface lui envoie un O (pour signaler que la carte est prête)

Envoie la première partie des données

Attend que la carte d'interface lui envoie un N (pour signaler que la carte est prête et lui demander la suite des données)

Envoi de la seconde partie des données

b) L'IHM

Le logiciel sur l'ordinateur sert d'IHM (interface homme machine), il nous permet de visualiser les sorties du TCVL et de lui envoyer des informations. La programmation de l'interface utilisateur a été faite par Jean-Daniel, en python. Un langage basé sur la programmation orientée objet.

L'interface se compose de 12 LED indiquant les sorties du TCVL, et 16 boutons pour renseigner les entrées de la TCVL

Lors de la conception de la carte, j'ai privilégié le routage pour en limiter la complexité. En effet, la numérotation des ports de l'expanseur I2C (max7300) n'est pas linéaire (voir figure 41). (Ex : broche 5 = port 8 ; broche 6 = port 12). Cela entraine une interconnexion des ports sans logique entre l'expanseur I2C et la TCVL.

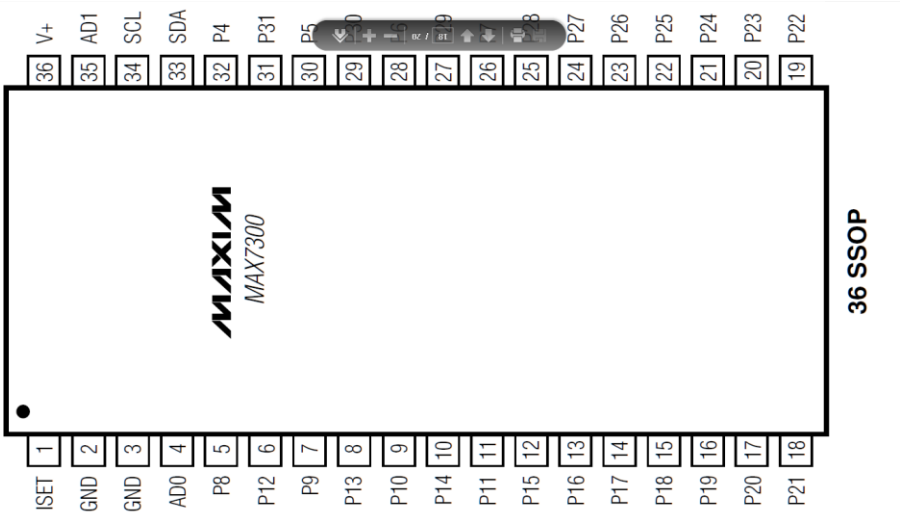


Figure 46 : Brochage MAX7300

Ceci n'a pas d'importance pour l'expanseur I2C, car tous ses ports sont des Entrées/Sorties indépendantes. (Il n'y a pas de ports réservés pour les entrées et d'autres pour les sorties). J'ai donc tracé les pistes sur la carte d'interface sans me préoccuper du numéro de port, cela a pour conséquence de mélanger les bits dans les données transmises. Il a donc fallu créer un tableau (annexe) qui indique pour chaque broche la position de leurs bits respectifs dans les octets de données. Ainsi on a pu faire le lien entre les LEDs affichées à l'écran et les bits des données reçues de la carte d'interface, et de manière analogue pour les boutons.

4. Test sur la TCVL

Les TCVL que nous disposons sont des équipements tout neufs qui n'ont jamais été mis sous tension. Ils sont équipés d'un mini PC tactile sous Windows XP en façade. Nous avons eu des problèmes pour configurer la TCVL. Une fois le PC démarré, un message apparaissait, nous indiquant que la clé de sécurité est incompatible… Après avoir contacté la DTI, ils nous ont donné la solution pour corriger ce problème et nous avons enfin réussi à configurer la TCVL.

Figure 47: écran de la TCVL

Figure 48 : Connecteurs arrière de la TCVL

Après avoir effectué les tests, nous avons constaté que la TCVL ne réagissait pas comme ce qui est indiqué dans la documentation de la TC892. Nous avons fait l'hypothèse que les sorties de la TCVL sont mises en œuvre avec des relais (à cause du bruit caractéristique des relais). Nous avons fait des tests pour confirmer cette hypothèse et modifié la carte d'interface en conséquence. Une fois les modifications effectuées, tout a fonctionné normalement.

Les changements d'état de la TCVL se traduisent par une mise à la masse du contact ou une mise en haute impédance. Les relais offrent une liaison galvanique, il est donc inutile de garder les optocoupleurs en entrée de l'expanseur I2C.

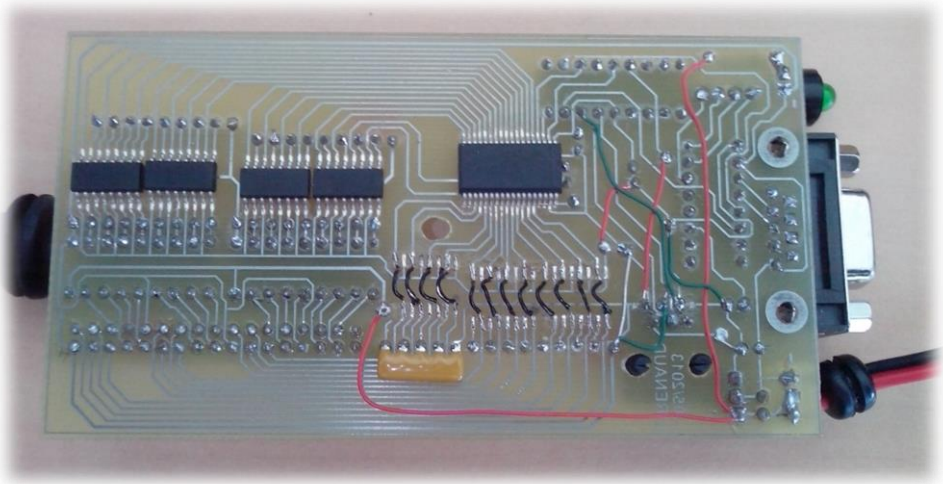


Figure 49 : Carte d'interface après corrections

5. Bilan des coûts et du temps de fabrication

En termes de coût, la carte d'interface coûte environ 20€, sans compter le câble convertisseur USB/Série. Ce câble coûte à lui seul environ 15€.

Pour le temps de fabrication, je pense que pour une personne qui n'est pas trop expérimentée (qui ne sait pas où se trouve le matériel, et n'a pas l'habitude de l'utiliser…) une carte peut être fabriquée en moins d'une journée en supposant que tout le matériel soit disponible :

- 3h pour la fabrication du circuit imprimé (perçage compris)
- 1h pour la soudure des CMS
- 15 minutes pour la soudure au fer
- 1h pour la fabrication du cordon
- 15 minutes pour « l'usinage » de la boîte
- 15 minutes pour l'assemblage et la finition

Figure 53 : L'atelier (1)

Figure 52 : L'atelier(2)

Figure 51 : L'atelier (3)

Figure 50 : L'atelier (4)

Conclusion

Ce projet à l'ENAC m'a fait évoluer de manière autonome dans de nombreux domaines, qui appartiennent à la conception complète d'un produit à partir d'un cahier des charges.

En électronique, j'ai pu réaliser une carte électronique de A à Z, ce qui implique de créer des schématiques, choisir les composants les mieux adaptés, s'adapter en fonction des contraintes de fabrications, etc. J'ai beaucoup appris dans la fabrication concrète d'un circuit électronique, la fabrication du circuit imprimé ('insolation de la plaque photosensible, la gravure…), la soudure de CMS etc.

En mécanique, j'ai réutilisé SolidWorks, un logiciel que j'utilisais au lycée, pour pouvoir définir les contraintes dimensionnelles lors de la conception du circuit électronique. J'ai fait un peu d'usinage sur le boitier (perçage, évidage…) et beaucoup de bricolage général fait avec « du bon sens ». J'ai d'ailleurs eu beaucoup de plaisir à travailler dans l'atelier de l'ENAC, il y a beaucoup de matériels de très bonne qualité à disposition ce qui permet un travail agréable et de qualité.

En programmation, j'ai programmé un PIC en assembleur. Un assembleur qui est assez diffèrent de celui utilisé à l'IUT. Je n'ai cependant pas eu de mal à m'adapter à cet assembleur et j'ai pu rapidement commencer à coder normalement (sans regarder la documentation toutes les minutes). Pour la programmation sur ordinateur, j'ai réalisé quelques programmes de test en C avec le logiciel Processing. J'ai travaillé avec Jean Daniel Gril pour le développement de l'IHM et de la communication série en Python, il m'a beaucoup appris sur la programmation-objet.

Table d'illustration :

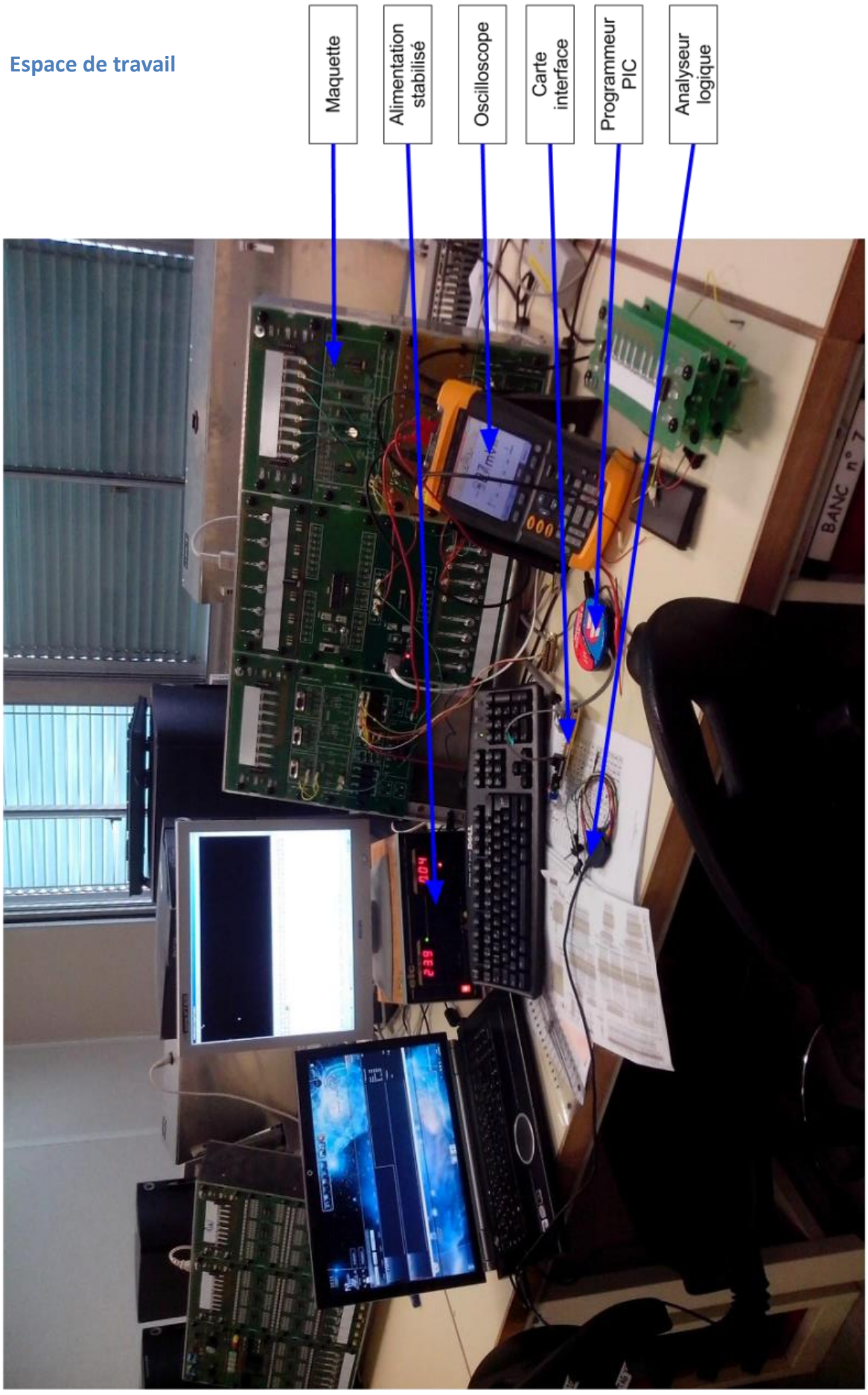
Annexes

Liste des composants pour une seule carte

Nom	référence	Quantité	Prix
Microcontrôleur	16F88	1	2,50€
PIO	max7300	1	échantillon gratuit où environ 2€
MAX232	MAX232	1	0,50€
Optocoupleur	ACPL-247-500E	4	1,23€
régulateur 5V	L7805	1	0.50€
Réseau de résistance	8res1com 1 kohm	2	0.50€
Réseau de résistance	8res1com 200 ohm	2	0,50€
Résistance	2.2k	2	0.03€
Résistance	1.2k	1	0.03€
Résistance	39K	1	0.03€
Condensateur non pol	0.1µf	1	0.05€
Condensateur non pol	0.33µf	1	0.05€
Condensateur non pol	47nf	1	0.05€
Condensateur pol	1µf	3	0.05€
Connecteur mâle 34 broches	HE10, 34 broches Mâle pour CI	1	1€
Connecteur femelle 34 broches	HE10, 34 broches Femelle à sertir	1	1€
Connecteur 37 broches	SUB-D 37 broches femelle	2	1,22€
Connecteur 9 broches	SUB-D 9 broches femelle	1	0,48€
Capot connecteur 37 broches	capot pour sub-d 37 broches	2	1,50€
Capot connecteur 9 broches	capot pour sub-d 9 broches	1	1€
Câble USB/RS232	câble convertisseur USB/RS232	1	14€ (possible de trouver moins chers)
Boîtier	Boîtier 3x5x10cm	1	3€

Prix total: environ 35€

Espace de travail



Analyse trame I2CMAX7300

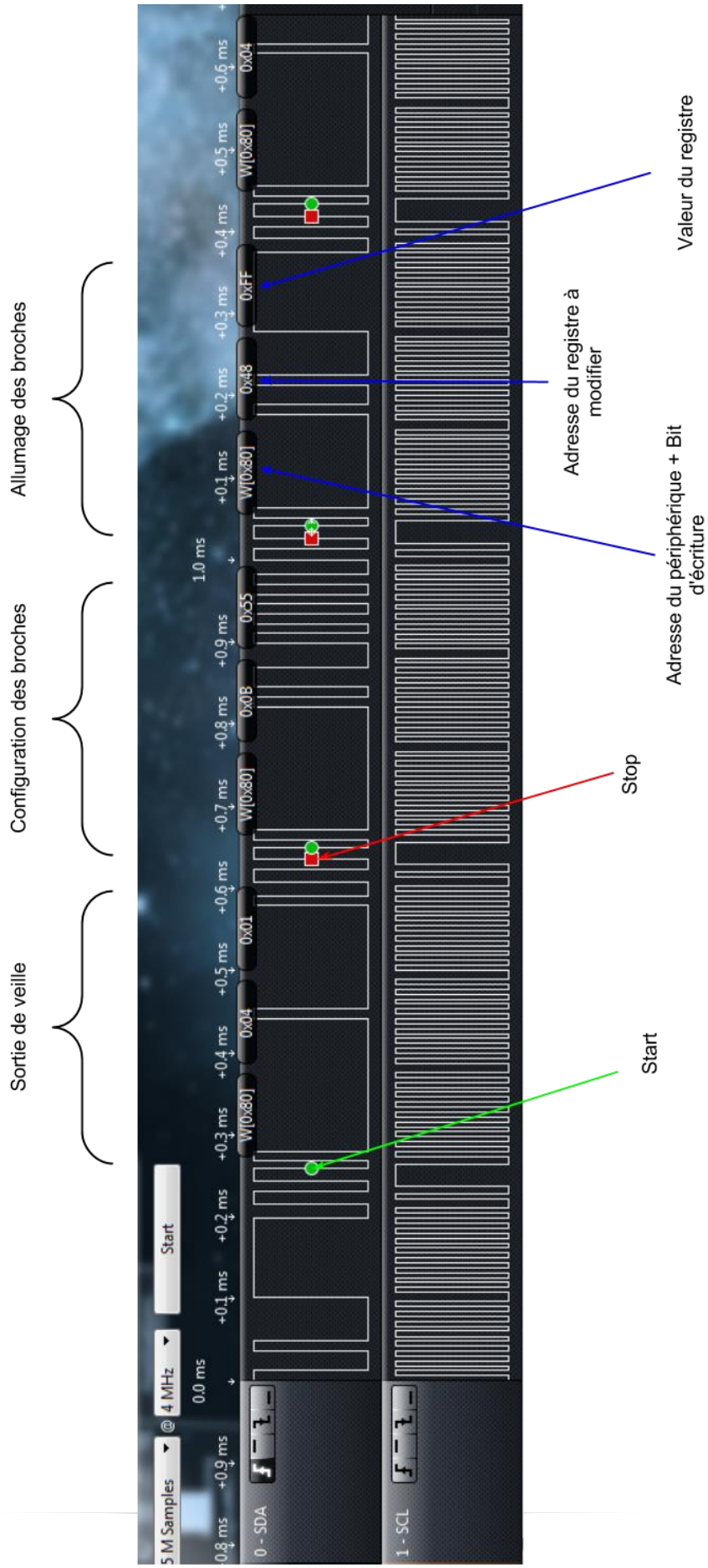
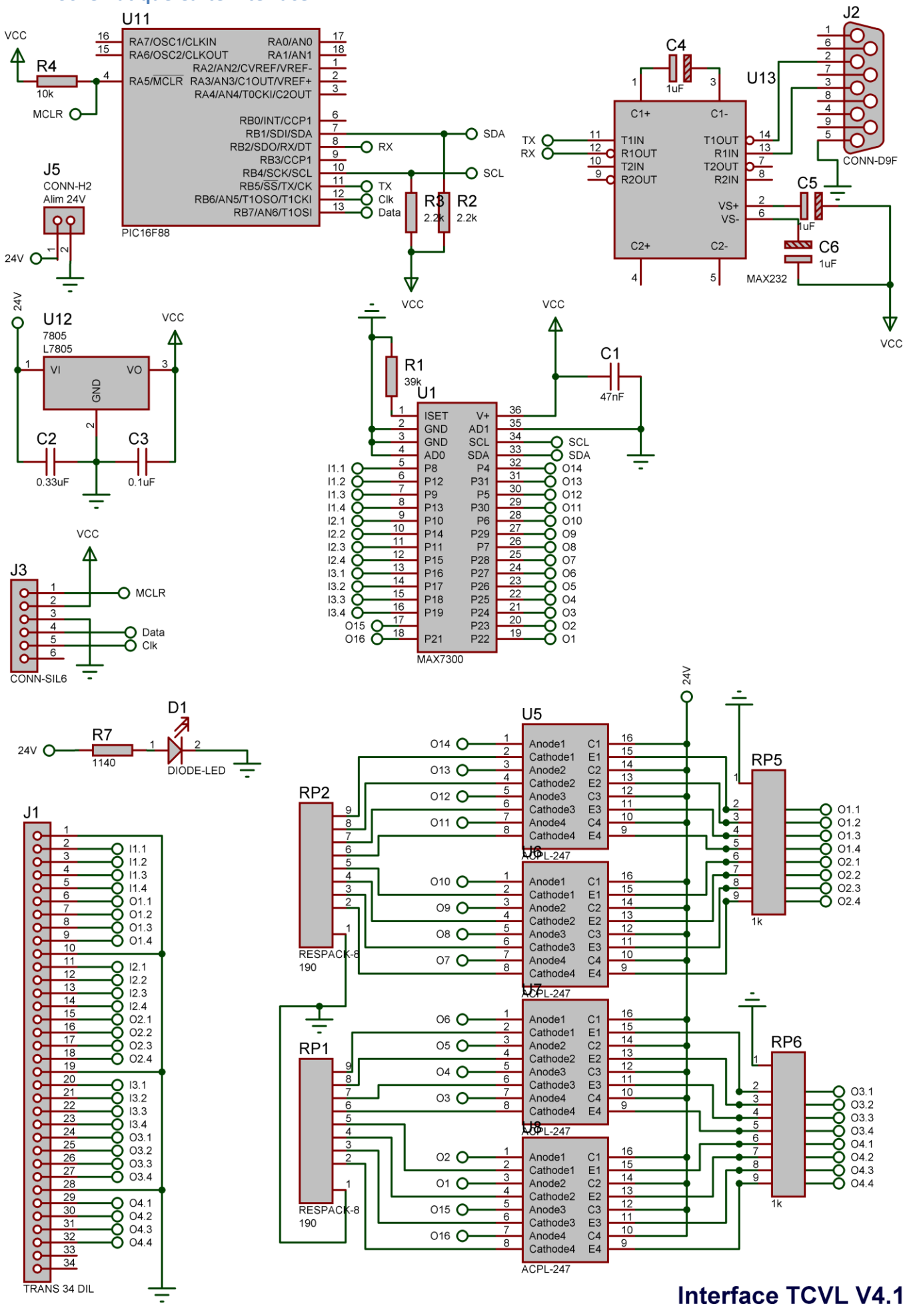


Tableau de correspondance bit/broches

	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Lout	O4.1	O4.2	O4.4	O4.3	O2.3E	O2.1R	O1.3E	O1.1R
Hout (Bouton pousoir PC)	O1.2E	O1.4R	O2.2R	O2.4E	O3.1R	O3.2R	O3.3E	O3.4E
Lin	I2.3E	I2.1R	I1.4R	I1.2E	I2.4E	I2.2R	I1.3E	I1.1R
Hin (LED PC)	0	0	0	0	I3.4E	I3.3E	I3.2R	I3.1R
XN.KZ	X = O sortie ou I entrée		3E = Emetteur 1					
	N = Numéro du Connecteur (1 à 3 TC892, 4=PAE892)		4E = Emetteur 2					
	K = Numéro de la broche 1 à 4							
	Z = Emetteur ou Récepteur							

Schématique Carte interface



Interface TCVL V4.1

Journal de bord

- 1er jour: 08/04/13
  - Papier administratif (badge, compte ...)
  - Explication du cadre du projet.
  - Lecture des documents sur le cadre et l'utilisation du boitier TCVL.
  - Bilan des connexions (25 E/S)
- 2éme jour: 09/04/13
  - Choix de la solution finale: (utilisation d'un microcontrôleur pour gérer l'interface entre le boitier TCVL et un PC)
  - Recherche d'un microcontrôleur adapté
  - Recherche d'une solution pour adapter les entrées/sorties (expanseur, mux ...) à fixer
  - Recherche solution pour interface usb série RS232 TTL (câble, FTDI, max232...) à fixer
- 3éme jour: 10/04/13
  - Recherche d'une solution pour adapter les niveaux logiques, entre le 5V du µp et le 24V du TCVL.
  - Rencontre avec le mec de l'électronique, choix de microcontrôleur (picxxx), cours sur ce µp, qui sera programmé en ASM
  - Prise en main du matériel (programmation de programme simple sur le pic, chenillard...)
- 4éme jour: 11/04/13
  - Mise en œuvre d'une communication série RS232 entre le pic et un PC
  - Mise en œuvre d'une communication I2C entre le pic et un périphérique I2C
  - Mise au point d'un protocole de communication pour échanger les données entre le PIC et le PC
  - Mise en œuvre du protocole de com sur le PIC et test avec un PIO 8 bits
- 5éme jour: 12/04/13
  - Mise au propre des algos
  - Commence à chercher solutions pour le PC (compilateur ? IDE ? ... QT, TK...)
  - Proposition solution pour adaptation tension: optocoupleur a6400
  - Mise au point du circuit d'adaptation des tensions avec les optocoupleurs
- 6eme jour: 17/04/13
  - simulation du circuit d'adaptation des tensions
- 7eme jour: 18/04/2013
  - Réalisation d'un schéma électrique globale sur ISIS
  - Recherche optocoupleur alternatifs pour réduire le coût

- 8eme jour: 19/04/13
  - Vérification du schéma d'adaptation des tensions par Christophe Dumont
  - Routage de la carte sous le logiciel proteus
- 9ème jour: 22/04/13
  - Routage de la carte (optimisation de la surface utile)
  - Recherche boîtier adapté
- 10ème jour: 23/04/13
  - Discussion avec Christophe Dumont sur le Boîtier et les connecteurs à utiliser.
  - Routage de la carte avec les bons connecteurs.
- 11ème jour: 24/04/2013
  - rédaction de la liste de composants
  - Amélioration du routage
- 12ème jour: 25/04/13

    - Commande des composants qui ne sont pas en stock
        - maxim
        - farnell
        - électronique diffusion
    - Recherche connecteur d'alimentation
- 13eme jour: 26/04/13
    - modification de l'interface entre le µp et le pc, utilisation d'un câble convertisseur USB vers RS232
    - implémentation de l'adaptation des niveaux de tensions pour passer du rs232 ttl en 12V grâce au max232
    - Réalisation du nouveau schéma électrique en utilisant les labels pour plus de clarté.
- 14eme jour : 29/04/13
    - routage de la carte
    - vérification de la taille de la carte (simu sur SolidWorks)
- 16eme jour : 30/04/13
    - routage carte module max7300
- 17eme jour : 06/05/13
    - réception des max7300 de chez Maxim
        - vérification de l'empreinte
    - recherche connecteur d'alim dans d'autre service de l'ENAC.
    - vérification typon du module max7300
    - Chercher un connecteur d'alim à la DTI
- 18eme jour: 07/05/13
    - Correction erreur schématique (empreinte du max7300 mal numéroté)

- 19eme jour: 13/05/13
    - changement du placement des composants (pour éviter de futurs problèmes de soudure) reroutage complet de la carte.
    - réception max232

- 20eme jour: 14/05/13
    - routage de la carte (version 2.7)
    - impression des typons

- 21eme jour: 15/5/2013
    - correction schéma de la carte (V3.1)

- 22eme jour: 16/05/13
    - reroutage et impression du typon
    - réception des composants de chez Farnell

- 23eme jour: 17
    - impression des circuits imprimés
    - perçage

- 24eme jour: 21/05/2013
    - soudure des composants
    - usinage du boitier
    - fabrication du cordon
- 25 eme jour : 22/05/2013
    - soudure du cordon

  - test des connections

- 26eme jour : 23/05/2013
  - test de communication entre le programmeur et la carte
  - correction du connecteur de programmation

- 27 eme jour: 24/05/2013
  - programmation de la réalisation série de la carte d'interface
  - impression de la carte module max7300
  - soudage des composants

- 28eme jour:27/05/2013
  - programmation de la communication entre le max7300 et un pic
  - fabrication d'un analyseur logique avec l'arduino

- 29eme jour: 28/05/2013
  - mise au point de la liaison I2C sur la carte interface (non supportée de manière matérielle)
  - programmation de la communication entre la carte d'interface et le module max7300
  - structuration du programme (pas d'utilisation d'interruptions, car provoque des bugs)

- 29eme jour: 29/05/2013
  - configuration des E/S du max
  - programmation du programme final

30eme jour: 30/05/2013
- test des entrées de la carte
  - modification, ajout résistances de pulldown
- correction du code

31eme jour : 31/05/2013
- développement logiciel avec Jean-Daniel Gril
  - en Python
  - et processing pour des tests

32eme jour : 3/06/2013
- idem
- définition des tableaux de correspondance avec les octets et les broches

33eme jour : 4/06/2013
- vérification du tableau grâce à des tests sur la carte d'interface

34eme jour : 06/06/2013
- fabrication du câble d'alimentation du TCVL
- Mise en marche du TCVL
  - problème message d'erreur ("clé de sécurité incompatible")

Programme assembleur de la carte d'interface

```
;****************************************
;**               TCVL 5               **
;**          Oscillateur  8Mhz         **
;**          Renaud - mai 2013         **
;****************************************
;                  ____________
;               <>-|PA0    PC0|-<>
;               <>-|PA1    PC1|-<>
;               <>-|PA2    PC2|-<>
;               <>-|PA3    PC3|-<>
;               <>-|PA4    PC4|-<>
;               <>-|PA5    PC5|-<>
;                  |       PC6|-<>
;               <>-|PE0    PC7|-<>
;               <>-|PE1       |
;               <>-|PE2       |
;                  |       PD0|-<>
;               <>-|PB0    PD1|-<>
;               <>-|PB1    PD2|-<>
;               <>-|PB2    PD3|-<>
;               <>-|PB3    PD4|-<>
;               <>-|PB4    PD5|-<>
;               <>-|PB5    PD6|-<>
;               <>-|PB6    PD7|-<>
;               <>-|PB7       |
;                  ------------
nolist
#INCLUDE reg88.inc
list
;*************************************
;******** RESERVATION MEMOIRE **********
;*************************************
CBLOCK  0x20
        carac   ; variable caractère
;variables a definir dans le programme principal:

        I2C_BOOL                                ; I2C : - 8 bits = 8 variables booleennes.
        I2C_dat                         ; I2C : - Octet de donnee.
        I2C_adrHIGH                             ; I2C : - Adresse : octet de poids faible.
        I2C_adrLOW                              ; I2C : - Adresse : octet de poids fort.
        I2C_VAR1                                ; I2C : - 3 variables temporaires :
        I2C_VAR2                                ; I2C :   sauvegarde de W, compteurs, etc.

        Hin     ; Variable où sont stockés les information sur les entrées/sorties du TCVL
        Lin
        Hout
        Lout
        R0
        R1
        R2
        R7

ENDC
;*************************************
;************ EQUIVALENCES *************
;*************************************
SAUVEW  EQU     h'007F' ; Sauvegarde registre W
SAUVES  EQU     h'007E' ; Sauvegarde registre STATUS

;*************************************
;**************** CODE *****************
;*************************************
        ORG h'0000'
        GOTO PROG
        ORG h'0005'
; CONFIG entrees sorties ---------------
PROG    PAGE1

        movlw   b'00000100'     ;TX 5 en sortie et RX 2 en entrée 0000 0100
        MOVWF   TRISB

; CONFIG oscilateur
```

```
        PAGE1
        BSF     IRCF2   ; Oscillateur interne, Tcy = 0.5us   8Mhz
        BSF     IRCF1
        BSF     IRCF0

;=============================================================================
; PROGRAMME PRINCIPAL ========================================================
;=============================================================================

        PAGE0

        MOVLW   "*"
        movwf   Hout
        MOVLW   "$"
        movwf   Lout

        call    INIT_SERIE
        call    INIT_I2C

        CALL    I2C_START       ; envoi un START en I2C             Allumage du max7300 (sortie
de veille)
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x04
        CALL    I2C_EMISS               ;turn on
        movlw   0x01
        CALL    I2C_EMISS               ;
        CALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C

        CALL    I2C_START               ; envoi un START en I2C       config broches sortie
4,5,6,7 en sortie
        MOVLW   b'10000000'             ; @ max7300 en ecriture
        CALL    I2C_EMISS               ;
        movlw   0x09            ;registre
        CALL    I2C_EMISS               ;
        movlw   0x55            ;config
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK               ; envoi un NOACK en I2C
        CALL    I2C_STOP                ; envoi un STOP en I2C


        CALL    I2C_START       ; envoi un START en I2C             config broches entrées
[8..11]
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x0a            ;registre
        CALL    I2C_EMISS               ;
        movlw   0xff            ;config        INPUT AVEC pullup
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C

        CALL    I2C_START       ; envoi un START en I2C             config broches entrées
[12..15]
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x0b            ;registre
        CALL    I2C_EMISS               ;
        movlw   0xff            ;config        INPUT AVEC pullup
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C

        CALL    I2C_START       ; envoi un START en I2C             config broches entrées
[16..19]
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x0c            ;registre
        CALL    I2C_EMISS               ;
        movlw   0xff            ;config        INPUT AVEC pullup
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C
```

```
        CALL    I2C_START               ; envoi un START en I2C       config broches sortie
[20..23] en sortie
        MOVLW   b'10000000'             ; @ max7300 en ecriture
        CALL    I2C_EMISS               ;
        movlw   0x0d            ;registre
        CALL    I2C_EMISS               ;
        movlw   0x55            ;config
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK               ; envoi un NOACK en I2C
        CALL    I2C_STOP                ; envoi un STOP en I2C

        CALL    I2C_START               ; envoi un START en I2C       config broches sortie
[24..27] en sortie
        MOVLW   b'10000000'             ; @ max7300 en ecriture
        CALL    I2C_EMISS               ;
        movlw   0x0e            ;registre
        CALL    I2C_EMISS               ;
        movlw   0x55            ;config
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK               ; envoi un NOACK en I2C
        CALL    I2C_STOP                ; envoi un STOP en I2C

        CALL    I2C_START               ; envoi un START en I2C       config broches sortie
[28..31] en sortie
        MOVLW   b'10000000'             ; @ max7300 en ecriture
        CALL    I2C_EMISS               ;
        movlw   0x0f            ;registre
        CALL    I2C_EMISS               ;
        movlw   0x55            ;config
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK               ; envoi un NOACK en I2C
        CALL    I2C_STOP                ; envoi un STOP en I2C

DEBUT   PAGE0

        BTFSC   RCIF    ;test buffer eception
        goto    COM     ;si buffer plein go com

        GOTO    DEBUT


COM     call    LIRE_CAR        ;lit le caractère reçu
        MOVWF   R0      ;sauvegarde du caractère reçue (R ou W ou T)
        BCF     Z
        sublw   "W"
        BTFSS   Z       ;test si le caractère reçu est un w sinon goto test_carR
        goto    test_carR

                        ;receptions des entrées
        MOVLW   "O"     ;envoie O pour demander l'emission du PC
        call    ECRIRE_CAR
        call    LIRE_CAR
        MOVWF   Hin
        MOVLW   "N"     ;envoi N pour demander la suite au PC
        call    ECRIRE_CAR
        call    LIRE_CAR
        MOVWF   Lin
        movfw   Lin
        movwf   R2

        movFW   Lin
        ANDLW   b'00001111'     ;on garde que les quatres premiers bits (au cas ou)
        movWF   Lin

        CALL    I2C_START       ; envoi un START en I2C             ecriture sur les sortie
[7..4]
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x40            ;registre
        CALL    I2C_EMISS               ;
        movFW   Lin
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C
```

```
        RRF     R2      ;décalage à droite (les 4 bits de poids forts sont déjà présents dans
Lout)
        RRF     R2
        RRF     R2
        RRF     R2
        movfw   R2
        ANDLW   b'00001111'     ;on garde que les quatres premiers bits
        movWF   R2

        CALL    I2C_START       ; envoi un START en I2C             ecriture sur les sortie
[27..20]
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x54            ;registre
        CALL    I2C_EMISS               ;
        movFW   R2
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C

        CALL    I2C_START       ; envoi un START en I2C             ecriture sur les sortie
[31..24]
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x58            ;registre
        CALL    I2C_EMISS               ;
        movFW   Hin
        CALL    I2C_EMISS               ;port
        CALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C

        goto    fin_com

test_carR
        movFW   R0
        SUBLW   "R"     ;test si on a reçu 'R' sinon goto fin
        BTFSS   Z
        goto    test_carT

        CALL    I2C_START       ; envoi un START en I2C             lecture des entrées 8..15
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x48            ;registre
        CALL    I2C_EMISS

        CALL    I2C_START       ; envoi un START en I2C             lecture des entrées
        MOVLW   b'10000001'     ; @ max7300 en ecriture
        call    I2C_EMISS
        CALL    I2C_RECEP
        movwf   Lout            ;déplacement des données lues dans Lout
        cALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C

        CALL    I2C_START       ; envoi un START en I2C             lecture des entrées 12..19
        MOVLW   b'10000000'     ; @ max7300 en ecriture
        CALL    I2C_EMISS       ;
        movlw   0x4c            ;registre
        CALL    I2C_EMISS

        CALL    I2C_START       ; envoi un START en I2C             lecture des entrées
        MOVLW   b'10000001'     ; @ max7300 en ecriture
        call    I2C_EMISS
        CALL    I2C_RECEP
        movwf   Hout            ;déplacement des données lues dans Lout
        cALL    I2C_NOACK       ; envoi un NOACK en I2C
        CALL    I2C_STOP        ; envoi un STOP en I2C

        RRF     Hout    ;décalage à droite (les 4 bits de poids forts sont déjà présents dans
Lout)
        RRF     Hout
        RRF     Hout
        RRF     Hout

        movFW   Hout
        ANDLW   b'00001111'     ;on garde que les quatres premiers bits
```

```
        movWF   Hout

        movFW   Hout    ; envoie des données
        call    ECRIRE_CAR
        movFW   Lout
        call    ECRIRE_CAR
        goto    fin_com

test_carT
        movFW   R0
        SUBLW   "T"     ;test si on a reçu 'T' sinon goto fin
        BTFSS   Z
        goto    fin_com

        movlw   "O"
        call    ECRIRE_CAR

fin_com
        goto    DEBUT

;*******************************************************************************
; SOUS PROGRAMMES ==============================================================
;*******************************************************************************

;--Sous programme------------------ INIT_SERIE ---------------------------------
; Entree -> rien
; Sortie <- rien
INIT_SERIE
; CONFIG liaison serie -----------------
        PAGE1
        MOVLW   d'51'
        MOVWF   SPBRG     ; vitesse = 9600 Bauds
        BSF     BRGH      ; vitesse rapide
        BCF     SYNC      ; mode asynchrone
        BSF     TXEN      ; mise en service du Tx
        PAGE0
        BSF     SPEN      ; mis en service de l'USART
        BSF     CREN      ; réception continue

        RETURN
;--Sous programme------------------ LIRE_CAR -----------------------------------
; Entree -> nul
; registre w détruit
; Sortie <- W
LIRE_CAR        PAGE0
test
        BTFSS   RCIF      ; teste flag réception
        goto    test
        MOVFW   RCREG     ; l'octet reçu est lu
        BCF     RCIF

        RETURN

;--Sous programme------------------ ECRIRE_CAR ---------------------------------
; Entree -> w
; Sortie <- nul
ECRIRE_CAR      PAGE0

FREE0
        BTFSS   TXIF      ; teste flag émission
        GOTO    FREE0      ; flag=0 registre Tx plein
        MOVWF   TXREG     ; emission de la data reçue: écho
        BCF     TXIF

        RETURN


;-------------------------------------------------------------------------------;
;                                                                               ;
;********************************** I2C ***************************************;
                      ;
;                                                       ;
;-------------------------------------------------------------------------------;

;-------------------------------------------------------------------------------;
```

```
;                 sous-fonctions permettant de realiser                 ;
;                les chronogrammes d'acces a la memoire :                ;
;------------------------------------------------------------------------;

INIT_I2C
       I2C_PORT       EQU PORTB      ; I2C : Port sur lequel est branché le bus I2C
       I2C_SCL              EQU 4          ; I2C : Broche 'Serial CLock' du bus I2C.
       I2C_SDA        EQU 1          ; I2C : Broche 'Serial DAta' du bus I2C.
       I2CACK         EQU 0          ; I2C :le bit '0' de l'adresse BOOLEEN, qui sert a
                                     ;      verifier si l'acces a la memoire I2C est correct.
       BSF   I2C_PORT,I2C_SCL               ; Initialiser a '1' les
       BSF   I2C_PORT,I2C_SDA               ; deux lignes du bus I2C.
          CLRF  I2C_BOOL                            ; Initialiser d'un coup a '0' les  8 bits
de la variable booleenne ;
       RETURN                                ; en fait, seul le bit 0 est utilise.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Envoie une sequence 'START' au peripherique I2C ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
I2C_START      BSF       I2C_PORT,I2C_SCL   ; Au debut, SCL
              BSF       I2C_PORT,I2C_SDA    ; et SDA sont a '1'.
              CALL      Tempo5us            ; On attend 5 us,
              BCF       I2C_PORT,I2C_SDA    ; puis SDA passe a '0',
              CALL      Tempo5us            ; puis on attend 5 us,
              BCF       I2C_PORT,I2C_SCL    ; puis SCL passe a '0',
              CALL      Tempo5us            ; puis on attend 5 us.
              BSF       I2C_PORT,I2C_SDA    ; SDA doit etre remis a '1' pour pouvoir
              RETURN                        ; voir les donnees venant de la memoire.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Envoie une sequence STOP au peripherique I2C ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
I2C_STOP       BCF       I2C_PORT,I2C_SCL   ; Au debut, SCL
              BCF       I2C_PORT,I2C_SDA    ; et SDA sont a '0'.
              CALL      Tempo5us            ; On attend 5 us,
              BSF       I2C_PORT,I2C_SCL    ; puis SCL passe a '1',
              CALL      Tempo5us            ; puis on attend 5 us,
              BSF       I2C_PORT,I2C_SDA    ; puis SDA passe a '1',
              GOTO      Tempo5us            ; on attend 5 us...

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Envoie un octet contenu dans W au peripherique I2C en 'serie' : ;;
;; (attention, cela modifie les variables VAR1 et VAR2)            ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; L'octet a envoyer est mis dans VAR1 ; on va alors faire subir 8 fois
; a cet octet un decalage vers la gauche, en recopiant a chaque fois au
; prealable le bit B7 sur la broche SDA (avant d'appliquer une impulsion
; d'horloge sur SCL). La variable VAR2 sert de compteur : elle part de 8
; et est decrementee 8 fois jusqu'a 0 afin de compter le nombre de
; decalages :
I2C_EMISS      MOVWF     I2C_VAR1            ; - Octet a envoyer mis dans VAR1.
              MOVLW     8                   ; - 8 (nb de decalages) dans VAR2.
              MOVWF     I2C_VAR2            ;
NextBitI2Cout BCF       I2C_PORT,I2C_SCL    ; - SCL mis a '0'.
              BTFSS     I2C_VAR1,7          ; - Bit B7 de VAR1 a '1' ?
              BCF       I2C_PORT,I2C_SDA    ;   si non : mettre SDA a '0'.
              BTFSC     I2C_VAR1,7          ; - Bit B7 de VAR1 a '0' ?
              BSF       I2C_PORT,I2C_SDA    ;   si non : mettre SDA a '1'.
              CALL      Tempo5us            ; - On applique l'impulsion sur SCL :
              BSF       I2C_PORT,I2C_SCL    ;   attente 5 us, SCL passe a '1',
              CALL      Tempo5us            ;   attente 5 us, SCL repasse a '0'.
              BCF       I2C_PORT,I2C_SCL    ;
              RLF       I2C_VAR1,F          ; - On decale VAR1 un coup a gauche.
              DECFSZ    I2C_VAR2,F          ; - On decremente VAR2 ; si ce n'est
              GOTO      NextBitI2Cout               ;   pas la 8ieme fois, on recommence.
              BSF       I2C_PORT,I2C_SDA    ; - On remet SDA a '1' (ce qui permet
test_ack
           call      I2C_TEST_ACK
           BTFSS          I2CACK,0
           goto           test_ack

              RETURN                        ;   de scruter SDA comme une entree).

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Lit un octet depuis le peripherique I2C et le met dans W : ;;
```

```
;; (attention, cela modifie les variables VAR1 et VAR2)       ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Cette routine ressemble beaucoup a la precedente ; mais au lieu
; de recopier le bit B7 sur SDA et de decaler l'octet vers la gauche
; et ce huit fois de suite, on commence par decaler l'octet, puis on
; recopie SDA sur B0, toujours huit fois de suite. Les bits ne sont
; plus retires un a un, mais au contraire empiles les uns apres les
; autres dans VAR1 :
I2C_RECEP     MOVLW     8                   ;-On met 8 dans VAR2, variable compteur
              MOVWF     I2C_VAR2            ; qui va comptabiliser les 8 decalages.
              CLRF      I2C_VAR1            ;- VAR1 initialisee a 0 (= 00000000b).
NextBitI2Cin  BCF       I2C_PORT,I2C_SCL    ;- SCL mis a '0'.
              BSF       I2C_PORT,I2C_SDA    ;- SDA mis a '1' pour etre 'lisible'.
              CALL      Tempo5us            ;- attente de 5 us.
              BSF       I2C_PORT,I2C_SCL    ;- SCL passe a '1'.
              CALL      Tempo5us            ;- attente de 5 us.
              RLF       I2C_VAR1,F          ;- decalage de un coup vers la gauche.
              BCF       I2C_VAR1,0          ;- bit B0 initialise a '0'.
              BTFSC     I2C_PORT,I2C_SDA    ;- on teste SDA : vaut-il '0' ?
              BSF       I2C_VAR1,0          ;  si non (SDA='1'), on met B0 a '1'.
              DECFSZ    I2C_VAR2,F          ;- huitieme cycle ?
              GOTO      NextBitI2Cin        ;  si non, on reboucle : cycle suivant.
              BCF       I2C_PORT,I2C_SCL    ;- apres le dernier cycle : SCL a '0'.
              MOVF      I2C_VAR1,W          ;- VAR1 (l'octet lu) est copie dans W.
              RETURN                        ;Fin de la routine

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Verifie si le peripherique I2C renvoie un 'Acknowledge' ; si celui-ci ;;
;; est present, le bit I2CACK est mis a '1', sinon I2CACK est mis a '0'. ;;
;; (la seule variable modifiee est BOOLEEN)                              ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
I2C_TEST_ACK  BSF       I2C_PORT,I2C_SDA    ; On s'assure que SDA est a '1'.
              BCF       I2C_PORT,I2C_SCL    ; On genere l'impulsion positive
              CALL      Tempo5us            ; sur SCL : SCL a '0', attente de 5
              BSF       I2C_PORT,I2C_SCL    ; us, SCL a '1', attente de 5 us.
              CALL      Tempo5us            ; La, on va lire la valeur de SDA :
              BCF       I2C_BOOL,I2CACK            ; - On met le bit I2CACK a '0'.
              BTFSS     I2C_PORT,I2C_SDA    ; - On verifie l'ACK (SDA a '0' ?)
              BSF       I2C_BOOL,I2CACK            ;   si oui, OK: on met I2CACK a '1'.
              BCF       I2C_PORT,I2C_SCL    ; Puis SCL repasse a '0'.
              RETURN                        ; Et la routine est terminee...
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Envoie un bit 'Acknowledge' au peripherique I2C. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
I2C_ACK     BCF       I2C_PORT,I2C_SCL      ; au debut, SCL est a '0'.
              BCF       I2C_PORT,I2C_SDA    ; SDA est mis a '0' (on applique ACK).
              CALL      Tempo5us            ; on attend 5 us.
              BSF       I2C_PORT,I2C_SCL    ; Impulsion positive sur SCL :
              CALL      Tempo5us            ; SCL passe a '1' pendant
              BCF       I2C_PORT,I2C_SCL    ; 5 us, puis repasse a '0'.
              BSF       I2C_PORT,I2C_SDA    ; Il faut penser a remettre SDA a '1' !
              RETURN                        ; Fin de la routine, retour au programme
principal.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Envoie un 'No Acknowledge' au peripherique I2C. ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
I2C_NOACK   BCF       I2C_PORT,I2C_SCL      ; au debut, SCL est a '0'
              BSF       I2C_PORT,I2C_SDA    ; et SDA a '1' (car pas d'ACK).
              CALL      Tempo5us            ; on attend 5 us.
              BSF       I2C_PORT,I2C_SCL    ; Impulsion positive sur SCL :
              CALL      Tempo5us            ; SCL passe a '1' pendant
              BCF       I2C_PORT,I2C_SCL    ; 5 us, puis repasse a '0'.
              RETURN                        ; Fin de la routine, retour au programme
principal.

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Temporisation de 5 micro-secondes pour respecter ;;
;; les temps d'acces au peripherique I2C.           ;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
Tempo5us      NOP       ; La tempo dure 5 us, avec le "CALL Tempo5us".
           NOP
              RETURN    ; (CALL=2us) + (NOP=1us) + (RETURN=2us) = 5 us.

       END
```

N° d'archivage
(ne pas remplir)

FICHE DE BILAN ET D'ARCHIVAGE DU RAPPORT DE STAGE

Cette fiche, disponible sur le site web du département dans la rubrique **"Espace Etudiant"**, est à remplir par chaque étudiant(e) et doit être :

- envoyée par mail **obligatoirement** avant la date de remise du rapport à **cathy.ibanez@iut-tlse3.fr**
- insérée en annexe du rapport de stage
- portée en un exemplaire papier le jour de la soutenance

NOM de l'étudiant(e) : Lucien RENAUD

Titre du rapport : Réalisation d'une interface TCVL

A - Nom de l'entreprise (expliciter les sigles)
ENAC Ecole National de l'Aviation Civile

Adresse :
ENAC
7 avenue Edouard Belin
31055 toulouse

Domaine(s) d'activité(s) :

Aéronautique

Enseignant Tuteur : Boutaieb DAHHOU

B - Critères de classement de stage

Lieu*	Type*	Spécialité*
1 ☒ Toulouse	1 ☒ Etude	1 ☐ Automatique
2 ☐ Midi-Pyrénées	2 ☒ Production	2 ☐ Electronique analogique
3 ☐ Métropole	3 ☐ Maintenance	3 ☐ Electronique de puissance
4 ☐ DOM-TOM	4 ☒ Intégration	4 ☐ Traitement du signal
5 ☐ Etranger	5 ☐ Autre : (préciser)	5 ☐ Hyperfréquences
		6 ☒ Informatique industrielle
		7 ☐ Réseaux
		8 ☒ Développement logiciel
		9 ☐ Autre : (préciser)

** Cocher les cases correspondantes*

C - Informations diverses sur le stage

Offre d'emploi prolongeant le stage ☐ OUI ☒ NON
Si oui type et durée :

D – Travail effectué par le (la) stagiaire

- Libellé technique de l'objet du stage : (510 caractères maximum sur 6 lignes maximum)

Le stage consiste à réaliser un boitier qui permet d'afficher l'état des commandes marche/arrêt et de pouvoir positionner les valeurs des données en entrée (alarmes et servitudes) d'un TCVL sur un PC.

- Libellé vulgarisé et explicatif : (510 caractères maximum sur 6 lignes maximum)

Avec l'arrivée prochaine de la voix sur IP pour les communications sol-sol de la navigation aérienne, de nouveaux équipements vont être déployés, et notamment les TCVL qui ont en charge la télécommande des radios des antennes avancées.

Afin de nous permettre de fournir des formations sur ces matériels, nous devons pouvoir simuler le comportement des radios connectées aux TCVL.

- Domaine(s) technique(s) et scientifique(s)
  - A quelle(s) discipline(s) ou technique(s) enseignée(s) au département peut-on rattacher ce travail ? (510 caractères maximum sur 6 lignes maximum)

Programmation
Étude et réalisation
Communication série
Assembleur

  - A quelle(s) discipline(s) ou technique(s) non enseignée(s) au département peut-on rattacher ce travail ? (510 caractères maximum sur 6 lignes maximum)

Contraintes de fabrication des circuits imprimés
Connaissance des composants courants
Réalisation complète d'un projet

- Matériel utilisé (préciser) : (510 caractères maximum sur 6 lignes maximum)

Microcontrôleur PIC 16F88
Oscilloscope numérique : fluke 196C
Multimètre numérique : MX570
Alimentation stabilisée : RACAL-DANA 9232
Analyseur logique : Saleae

E – Remarques personnelles : (510 caractères maximum sur 6 lignes maximum)

Ce projet très intéressant recouvre une grande partie des matières enseignées à l'IUT, il m'a demandé une très grande autonomie pour la réalisation complète du projet. (Choix des composants, choix de la solution technique, fabrication, etc.)

Résumé : (1700 caractères maximum sur 20 lignes maximum)

Des IESSA (Ingénieur Électronicien des Systèmes de la Sécurité Aérienne) sont amenés dans leur métier à utiliser et à faire de la maintenance sur des équipements TCVL. Ils doivent alors suivre une formation continue à l'ENAC où ils réaliseront des travaux pratiques sur ces équipements. Les émetteurs/récepteurs que supervisent les TCVL ne sont pas disponibles dans les salles de TP, nous devons alors trouver un moyen de simuler ces émetteurs/récepteurs.

Le but de ce stage est de réaliser une interface entre un boîtier TCVL et un PC, et de réaliser un logiciel d'interface homme-machine. Ce PC simulera plusieurs émetteurs/récepteurs afin de réaliser des travaux pratiques.

J'ai utilisé un expanseur I2C pour augmenter le nombre de broches du microcontrôleur, afin de pouvoir récupérer les informations fournies en TOR (tout ou rien) par le TCVL. J'ai implémenté des circuits pour adapter les niveaux de tensions entre les différents matériels. J'ai réalisé entièrement la fabrication et la programmation moi-même.

Mots clés : (5 maximum / 170 caractères / 2 lignes maximum)

PIC 16F88 / assembleur / TCVL / Interface / I2C

Abstract : (1700 characters in a maximum of 20 lines)

Some IESSA are brought into their business to use and do maintenance on TCVL equipments. They must undergo continuing training ENAC where they will carry out practical work on this equipment. Transmitters / receivers supervised by TCVL are not available in the rooms, then we must find a way to simulate these transmitters / receivers.

The purpose of this course is to provide an interface between a PC and a TCVL and achieve a human-machine interface software. This PC simulate multiple transmitters / receivers to carry out practical works.

I used an I2C expander to increase the number of pins on the microcontroller, in order to retrieve information in digital (all or nothing) by the TCVL. I implemented shematics to adjust voltage levels between the differents materials. I realized fully the manufacturing and programming by myself.

Key-words : (5 maximum / 170 characters / 2 lines maximum)

PIC 16F88 / assembly / TCVL / Interface / I2C

Schémas ou Photos illustrant votre stage : (pas plus d'une page)[1]

[1] Attention ce document ne doit pas faire plus de quatre pages et sa taille ne doit pas excéder 5 Mégaoctets !