Table des matières

Introduction

Ayant déjà par le passé effectué un stage en informatique et effectué un CDD en électronique d'un mois à la suite de celui-ci, j'ai pu découvrir les compétences qui y sont recherchées pour leurs projets. Le fait que ce centre me propose un stage en électronique m'a donc tout de suite intéressé, puisque celui-ci porte sur la réalisation d'application de gestion et de communication de machines, utilisées dans les élevages expérimentaux pour la collecte de diverses informations.
La biologie étant un domaine qui m'attire, travailler entouré de personne y travaillant m'a donc encouragé à me mettre à leur service.
En effet j'ai travaillé dans un service qui allie divers domaines tels que l'informatique, l'électronique ou encore la mécanique dans le but de subvenir aux besoins matériels spécifiques des scientifiques dans leurs élevages.
Donc pour ce stage, les applications réalisées seront utilisées, ou mise à disposition, de l'Unité Expérimentale de Bourges qui collecte des informations sur les ovins, caprins et palmipèdes. Ces données sont utilisées par la SAGA, Station d'Amélioration Génétique des Animaux, l'unité dans laquelle j'effectue mon stage.
Les différentes applications devront permettre de contrôler de diverses façons des machines à distance pour la collecte d'informations.
La mis en place et l'utilisation de techniques de communication est donc au cœur de mon stage.

1 Présentation de l'INRA

1.1 Généralités

L'INRA (Institut National de la Recherche Agronomique) est un établissement public à caractère scientifique et technologique(E.P.S.T). Fondé en 1946, il est placé sous la double tutelle du Ministère de la Recherche et du Ministère de l'Agriculture et de la Pêche. C'est le deuxième institut de recherche publique français et le premier institut de recherche agronomique européen. C'est un des trois premiers organismes mondiaux dans les domaines de l'agriculture, de l'alimentation, et de l'environnement.

1.1.1 Missions

Il est chargé d'organiser et d'exécuter toute la recherche scientifique concernant l'agriculture avec pour mission de produire et diffuser des connaissances scientifiques, concevoir des innovations et des savoir-faire pour la société, éclairer, par son expertise, les décisions des acteurs publics et privés, développer la culture scientifique et technique et participer au débat science/société et enfin former à la recherche et par la recherche.

1.1.2 Organisation

L'INRA est dirigé par un Président Directeur Général et par deux directeurs adjoints, eux-mêmes supplées par des directeurs scientifiques. Il existe également :

- un conseil d'administration définissant les orientations de l'institut tout en assurant sa gestion humaine et financière
- un conseil scientifique axé sur la recherche qui suit le bon déroulement des programme de recherche.

1.1.3 Structure et moyens

Homme et femmes : 1 839 chercheurs, 2 572 ingénieurs, 4 121 techniciens, 1 891 thésards, 1 519 stagiaires accueillis chaque années dans les laboratoires.

Organisation : 14 départements scientifiques, 19 centres de recherche régionaux, plus de 150 implémentations.
Budget : 772 Millions d'euros

1.2 Le centre de recherche de Toulouse

1.2.1 Présentation

Le centre de Toulouse a été ouvert en 1970. Il regroupe actuellement 22 laboratoires dont les activités de recherche concernent :

Génome et biotechnologies : génie génétique et enzymatique des micro-organismes, relations entre les plantes et les micro-organismes, méthodes de transformation génique des plantes, élaboration de la carte génique de plantes et d'animaux.

Sécurité des aliments : apporter aux pouvoirs publics des éléments objectifs d'information nécessaires à l'exercice de leur pouvoir réglementaire, accroître la protection des consommateurs, fournir aux industriels concernés les élément analytique de contrôle.

Territoire et produits : élaboration de produits de qualité (amélioration génétique des races animales), occupation de l'espace rural et aménagement du territoire (relation entre les activités agricoles, l'utilisation de l'espace et de l'environnement).

1.2.2 La SAGA

La SAGA ou Station d'Amélioration Génétique des Animaux, est née en 1975 de la fusion des laboratoires de Génétique des Petits Ruminants et de la Méthodologie Génétique. Elle est constituée de 50 agents permanents, répartis dans 4 équipes thématiques (suivant les espèces étudiées) interagissant dans 5 axes transversaux :

- Équipes thématiques :
  - → Lapin
  - → Palmipèdes
  - → Petits ruminants allaitant (ovins et caprins)
  - → Petit ruminants laitiers (ovins et caprins)
- Axes transversaux :
  - → Informatique et automatismes
  - → Méthodes
  - → Reproduction
  - → Résistance aux maladies
  - → Phanères

2 Le travail réalisé

Pour ce stage Mr Ricard m'a confié trois projets à réaliser et un projet à terminer. Tous ces projets on rapport avec la communication des micro-contrôleurs avec l'extérieur via divers protocoles.

Contexte

Les unités expérimentales s'occupant de collecter de nombreuses informations sur divers animaux, il est nécessaire de leur fournir les outils adaptés pour leur venir en aide.
C'est à ces besoins que le service, dans lequel j'ai effectué mon stage, répond. Il fournit des solutions au travers de divers domaines tels que la mécanique, l'électronique, ou l'informatique. Pour ne citer que quelques exemples :

- DACs (mangeoires automatisées)
- Applications pour PDA (cf : PDA page 34)
- Administration de Bases de données

Pour ce stage les différents projets ont été réalisés pour répondre à la demande de l'Unité Expérimentale de Bourges qui réalise ses expérimentations sur les ovins, caprins et bovins.
Les différents projets sont des outils que nous fournissons à cette Unité Expérimentale pour lui permettre de contrôler des mangeoires à distance. Trois des 4 applications réalisées pourront être utilisées pour d'autres projets, seul une des application est spécifique.
Cette dernière application permet de recevoir et d'afficher des valeurs de pesées d'animaux, le module demandé ajoute la fonctionnalité de communiquer via Bluetooth avec les PDAs (cf : PDA page 34), pour récupérer ces valeurs.

2.1 Présentation de l'UART et de la RS-232

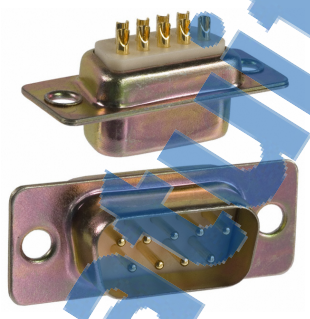
Les quatre projets, sur lesquels j'ai travaillé, utilisent un moyen de communication commun pour dialoguer avec un ordinateur ou d'autres machines. Je commencerai donc par vous introduire à celui-ci. Ce moyen regroupe deux technologies complémentaires, l'UART et la

RS-232.

2.1.1 La RS-232

La RS-232 est une norme de standardisation pour un bus de communication série, elle définit les caractéristiques électriques du bus de communication telles que les tensions, ou les intensités.
Les bus de communication sont des interfaces, ou connecteurs, permettant de relier des appareils entre eux dans le but d'échanger des informations binaires.

Connecteur Femelle



Connecteur mâle

C'est deux images nous montre le type de connecteur DE-9 utilisé pour ce type d'interface, qui est communément appelée port COM.

On qualifie un bus de communication "série" quand celui-ci envoie les informations bit à bit, un bit à la fois. Au contraire ceux dit "parallèles" permettent d'envoyer plusieurs bits à la fois, mais demande un nombre de connexions, fils, plus importantes. De ce fait les interfaces séries sont moins encombrantes, mais moins rapides, alors que les interfaces parallèles sont plus encombrantes, mais plus rapides.
L'intérêt de cette connexion c'est qu'elle ne nécessite que trois fils minimum, des fils supplémentaires peuvent être utilisés, mais dans les projets présentés ici ils ne sont pas utilisés.
Pour un des projet j'utilise un fil pour une utilisation non standard, et que je nomme VDD, qui permet l'alimentation d'un montage si celui-ci n'a pas sa propre alimentation.
Les trois fils minimums standards utilisés sont :

GND : Permet d'avoir une masse commune entre les montages
TX : Permet l'envoie de données
RX : Permet la réception de données

On qualifie aussi un tel câble de "croisé" quand on inverse les fils TX (pin 3) et RX (pin 2) à une extrémité, ce qui permet d'avoir le fil de transmission de l'un sur le fil de réception de l'autre composant et vis-versa :

Câble croisé

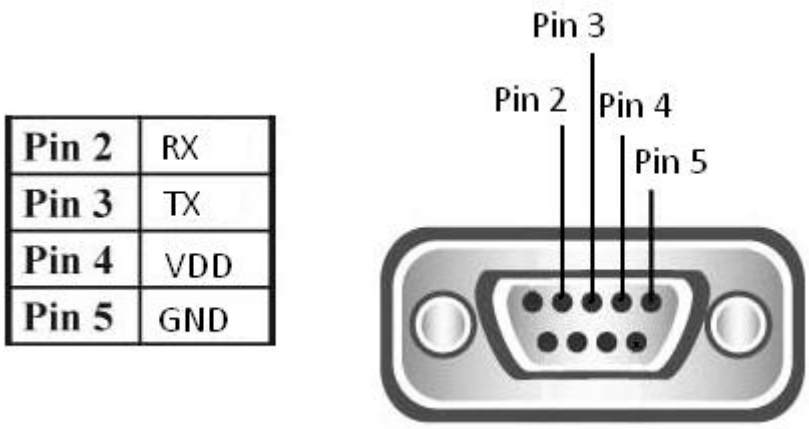


FIGURE 2.1 – Schéma de placement des fils de la RS-232, ainsi que VDD

Extrémité A		Extrémité B
TX(pin 3)	$\longrightarrow$	RX (pin 2)
RX(pin 2)	$\longleftarrow$	TX (pin 3)

A l'inverse un câble dit "droit" n'inverse pas les pins de données.

Câble droit

Extrémité A		Extrémité B
TX (pin 3)	$\longleftrightarrow$	TX (pin 3)
RX (pin 2)	$\longleftrightarrow$	RX (pin 2)

2.1.2 L'UART

L'UART Universal Asynchronous Receiver Transmitter (Émetteur Récepteur Asynchrone Universel). C'est un circuit électrique implémenté dans les micro-contrôleurs permettant l'envoie et la réception de données avec d'autres composants ou montages extérieurs.
L'UART comme son nom l'indique est une connexion asynchrone, elle permet la communication entre 2 appareils qui ont leurs horloges qui ne sont pas synchronisées. Ces horloges servent à l'émission ainsi qu'à la réception des données.

Son fonctionnement : Ce dispositif permet au micro-contrôleur de communiquer simplement avec une interface RS-232.
Le micro-contrôleur envoie et reçoit les données en parallèles (8 bits par 8 bits pour nos projets).
L'interface RS232 étant un bus de communication série, l'UART est là pour convertir l'information entre les 2, appelé aussi sérialisation (parallèle $\rightarrow$ série) et désérialisation (série $\rightarrow$ parallèle) des données.
Ces deux mécanismes sont mis en place au moyen de registres à décalages.

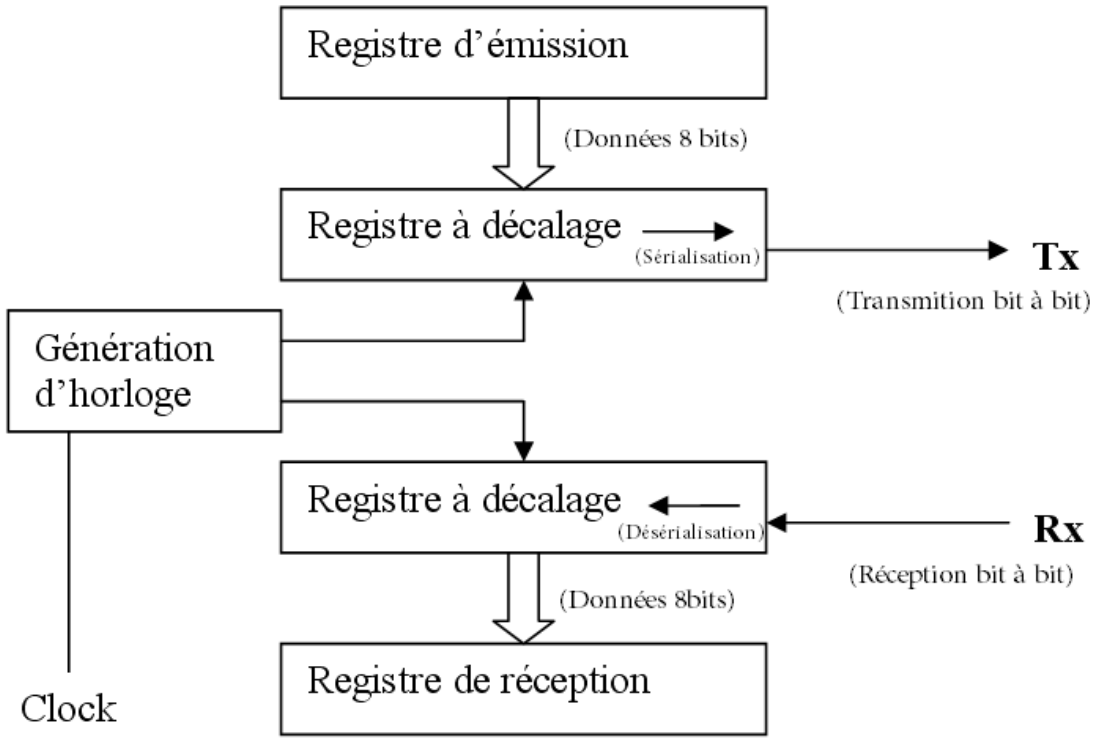


FIGURE 2.2 – Schéma de fonctionnement de l'UART

Pour qu'une telle connexion fonctionne il est donc nécessaire de les configurer au préalable avec les mêmes paramètres dans les différents systèmes reliés par une même connexion de ce type.
Pour cela on règle notamment la taille d'un caractère(ex : 8 ou 9 bits), ou la vitesse de communication, qui sont des valeurs standardisées (110,300,9600,38400, etc) en bauds/s (nombre de caractères à la seconde).
Pour nos projets la taille des caractères est de 8 bits.

2.2 Les outils utilisés

MPLAB IDE : IDE, environnement de développement des applications sur micro-contrôleurs pic de microchip.

Keil v1 : IDE, environnement de développement des applications sur micro-contrôleur 80C51.

Keil v2 : IDE, environnement de développement des applications sur micro-contrôleur 80C51.

MPLAB ICD 2 : Outils permettant de relier l'ordinateur à la carte où se situe le micro-contrôleur pour le programmer ou le déboguer. S'utilise exclusivement pour des micro-contrôleurs pic de Microchip.

OpenSSL : Logiciel d'ordinateur, open-source et disponible gratuitement sur internet, qui permet d'établir une connexion sécurisée, avec le protocole SSL (page 21), utilisé pour le projet du serveur TCP/IP, pour le tester ainsi que pour communiquer avec le serveur de façon sécurisée.

Flash 51 : Logiciel d'ordinateur permettant de programmer le micro-contrôleur 80C51 embarqué dans le AGPA (cf : AGPA page 33) du projet de gestionnaire de communication (page 32). Il utilise un port RS-232(cf : RS-232 page 11), un port COM de l'ordinateur, pour communiquer avec le AGPA. Ce logiciel est fournit par la société BALEA, constructeur des AGPA.

Notepad++ : Éditeur de texte avec coloration syntaxique, utilisé pour le projet sur 80C51.

2.3 Interface de communication entre 2 machines

Mon tuteur m'a demandé pour ce sujet de permettre la communication entre deux machines en formatant, adaptant, les données entre les deux, pour qu'ils puissent dialoguer, au travers d'un micro-contrôleur.
Le micro-contrôleur utilisé est un pic18F96J60 monté sur une carte électronique (cf : Annexe A page 42).

Contexte

Le lecteur de puce RFID permet la lecture de l'identifiant d'un animal passant dans le champs de l'antenne.
Celui-ci est placé dans des mangeoires pour savoir quel animal vient manger, et de là connaitre à quel animal correspondent les informations qui sont alors collectées par la mangeoire.
L'identification électronique des ovins est devenu obligatoire, et il a donc été nécessaire de changer les puces RFID des animaux ainsi que le lecteur de puces associé.
L'application sur ordinateur qui permet de récupérer les données sur les animaux est une logiciel propriétaire. De ce fait il est impossible de modifier ce logiciel pour qu'il puisse récupérer l'identifiant des animaux correctement.
L'autre problème c'est que le lecteur de puce ne peut pas non plus être modifié pour envoyer les identifiants de la manière voulue par le logiciel sur ordinateur.
C'est pourquoi j'ai été chargé de créer une application embarquée sur une carte électronique qui servira d'intermédiaire entre le lecteur de puces et l'ordinateur. Cette carte fonctionnera donc de manière autonome, aucune interaction possible avec une personne durant son fonctionnement.

2.3.1 Introduction à la RFID

La RFID, ou Radio Frequency Identification (Identification par fréquence radio), est une technologie qui permet de mémoriser et récupérer des informations à distance au moyen de "radio-étiquette".
Ces radio-étiquettes sont de petits objets contenant une antenne et une puce, qui mémorise les informations.

Principe : Quand un lecteur de puce est suffisamment proche d'une l'étiquette celle-ci ce met à fonctionner et transmet au lecteur les données qu'elle contient. L'énergie nécessaire à son fonctionnement est fournit par le lecteur de puce via radio fréquence.

Intérêt : Les principaux intérêts de cette technologie sont que les étiquettes, ou puces, peuvent être de taille très réduite, selon les utilisations, donc non encombrante, et deuxièmement ne nécessite aucune alimentation (ex : pile).

FIGURE 2.3 – Puce sous-cutanée pour animaux (à gauche), Grain de riz (à droite)

Cas d'utilisation : On utilise cette technologie dans de nombreux domaines :

- Télé-péage
- Puce d'identification sous-cutanée d'animaux domestiques
- Antivol dans les magasins

2.3.2 Présentation des machines

La première machine : c'est un lecteur de puce RFID qui utilise la RS-232 pour transmettre l'identifiant de la puce en cours de lecture. Le lecteur ne transmet aucune information s'il n'y pas de puces, sinon il envoie en continue la valeur de l'identifiant de la puce présente.
Il ne fait donc qu'émettre et sans rien recevoir sur la RS-232.
Cette machine ne possédant pas d'alimentation, elle est alimentée directement par la RS-232.

FIGURE 2.4 – Lecteur de puce RFID

FIGURE 2.5 – Puce RFID portée par un animal

La seconde machine : C'est un ordinateur qui lui ne fait que recevoir sur la RS-232 sans rien émettre.

La carte électronique : La carte devra être placés entre les deux pour formater les données reçues sur la RS-232 provenant du lecteur de puce, et envoyé les données formatées sur la RS-232 à l'ordinateur.

2.3.3 La liaison RS-232

La carte électronique ne possédant qu'un seul connecteur pour la RS-232, il a donc été nécessaire de fabriquer un câble adapté pour communiquer avec les deux machines avec un même port RS-232.
Le connecteur de la carte électronique est une femelle, câble croisé (La RS-232 page 11).
Le connecteur du lecteur de puce est mâle, câble croisé (La RS-232 page 11) et est alimenté par la RS-232.
Le connecteur du PC est une femelle mâle, câble droit.

Donc on utilise trois connecteurs pour créer notre câble :

PC : Un connecter mâle
Lecteur : Un connecteur femelle
Carte : Un connecteur mâle

Pour améliorer les explications qui suivront les trois connecteurs cités précédemment seront appelés respectivement, "connecteur PC", "Connecteur lecteur" et "Connecteur carte".

Les trois connecteurs ont tous leurs pins 5 (GND) câblées entre elles pour avoir une masse commune.
La pin 4 du connecteur lecteur est directement câblé à la borne VDD du générateur de la carte pour alimenter le lecteur de puce.

Connexion entre carte et lecteur : Le lecteur ne faisant que transmettre il n'utilise pas le fil de réception (fil RX). Le câble du lecteur étant croisé, la pin utilisé sur connecteur lecteur est la pin 2 (TX en croisé).
La carte étant croisée, il faut câbler la pin 3 (RX en croisé) de connecteur carte.
$\Longrightarrow$ Connecteur carte pin 3 (RX) $\longleftarrow$ Connecteur lecteur pin 2 (TX)

Connexion entre carte et PC : Le PC ne faisant que recevoir, il n'utilise pas le fil de transmission (fil TX). Le câble du PC n'étant pas croisé, la pin utilisée sur connecteur PC est la pin 2 (RX en droit).
La carte étant croisée, il faut câbler la pin 2 (TX en croisé) de connecteur carte.
$\Longrightarrow$ Connecteur carte pin 2(TX) $\longrightarrow$ Connecteur PC pin 2 (RX)

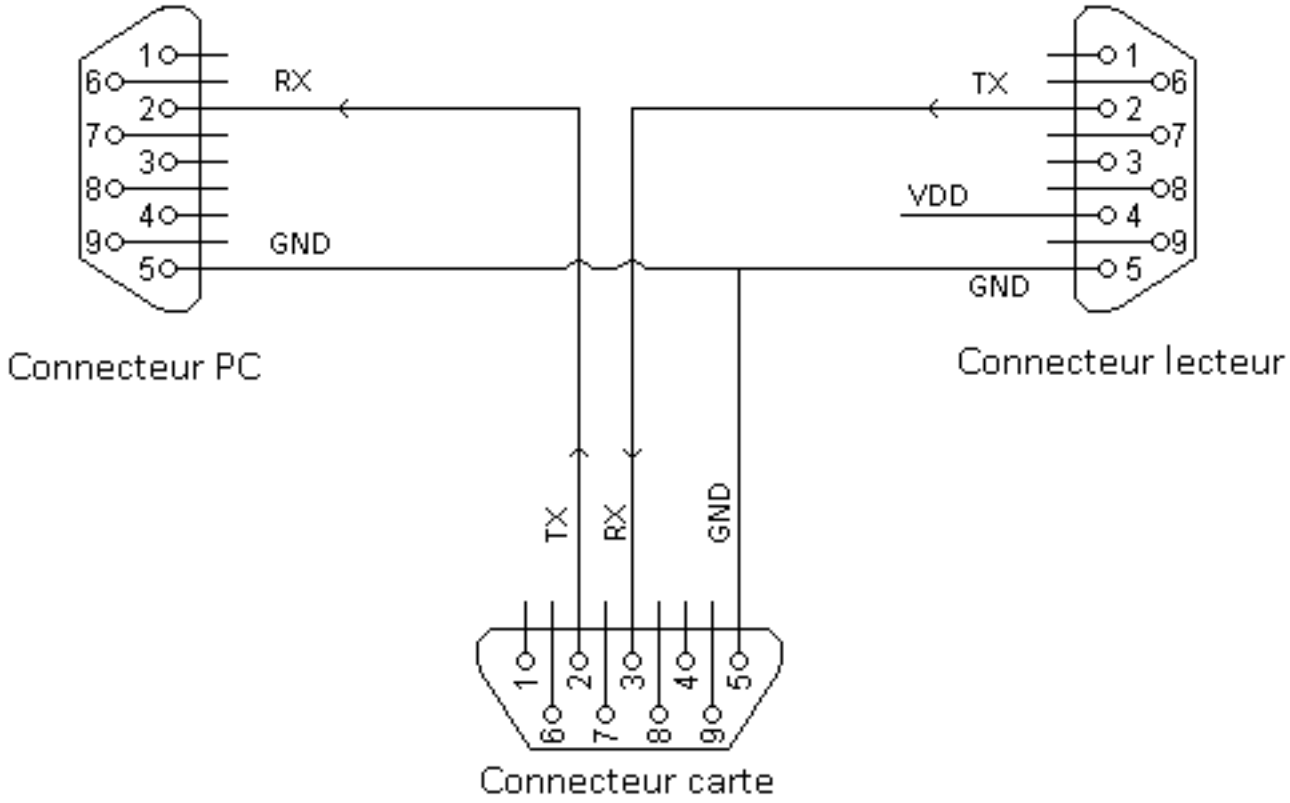


FIGURE 2.6 – Schéma récapitulatif du câblage entre les connecteurs

2.3.4 Fonctionnement de l'application

Le micro-contrôleur attend qu'un identifiant complet soit reçu sur l'UART (cf : L'UART page 10) puis le traite et envoie le résultat sur l'UART. La récupération de l'identifiant ayant déjà été codé l'année dernière durant mon CDD, je ne la présenterai pas ici.

Format des données du lecteur

N° octets :	0	1	2	3 à 5	6	7 à 18
Données :	'L'	'A'	' '	3 chiffres	' '	12 chiffres

Taille : 19 octets
Chiffres : seulement de '0' à '9'.
Une lettre (ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Format de donnée désirées

On utilise les 5 derniers chiffres du message reçu du lecteur de puce pour composer le nouveau message.

N° octets :	0	1 à 3	4 à 8	9
Données :	02h	'0' '0' '0'	les 5 derniers chiffres	03h

Tailles : 10 octets
Chiffres : seulement de '0' à '9'.
xxh : Nombre hexadécimal
Une lettre (ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Problème technique important rencontré

Au début de projet j'ai rencontré de très gros problèmes pour la compilation de l'application (transformation du code source en code machine). En effet j'ai passé énormément de temps à chercher la solution en pensant que j'étais responsable de ces erreurs. Mais après de longues recherches sur Internet, et notamment sur des forums, j'ai découvert que le problème venait de Microchip, le constructeur du micro-contrôleur. En effet avec la version du compilateur (exécutable transformant le code source en code machine) que j'utilisais, le fichier d'entête, correspondant au micro-contrôleur de la carte utilisée, était erroné.
J'ai donc récupéré un fichier d'entête pour ce micro-contrôleur dans une version antérieure du compilateur.
Le fichier d'entête contient de nombreuses constantes essentielles pour l'utilisation d'un micro-contrôleur.
La version du compilateur que j'utilisais : 3.37.1
La version du compilateur où j'ai récupéré le fichier d'entête : 3.36

2.4 Connexion Ethernet d'une carte électronique

Pour ce projet mon tuteur m'a demandé de permettre de contrôler une carte électronique via un réseau Internet. Par la suite nous avons ajouté une fonctionnalité permettant de sécuriser l'accès à la carte.

Contexte

La carte utilisée (cf : Annexe A page 42) embarque un pic18F97J60 a la possibilité de gérer une connexion Internet, et possède un port Ethernet pour la connecter à un réseau.
C'est pourquoi mon tuteur m'a demandé de mettre en place une application permettant de contrôler cette carte à distance via le réseau interne de l'INRA.
Intérêts : Le premier est qu'une telle application permet de contrôler les cartes à distances, tant que la machine utilisée pour envoyer des commandes soit sur le même réseaux (ici celui de l'INRA) que la carte. Le réseau de l'INRA comprend tous les centres INRA en France.
Un deuxième intérêt est qu'on libère ainsi le port de la RS-232, auparavant utilisé pour contrôler la carte. Ce qui permet à présent de l'utiliser pour brancher un autre dispositif pour collecter des informations par exemple.
Il m'a aussi été demandé de faire une application qui soit générique, dans le sens où il sera facile de l'utiliser dans d'autre projet, sur cette même carte électronique.
Pour ce projet Microchip, le constructeur du micro-contrôleur utilisé, fournit les bases pour la création d'un serveur TCP/IP embarqué.
Pour sécuriser la connexion j'ai utilisé le protocole de cryptage SSL fourni par Microchip.

2.4.1 Présentation de l'Ethernet et du TCP/IP

Ethernet, TCP et IP sont des protocoles de communication, qui font notamment partie d'un ensemble de protocoles utilisés pour Internet.
Voici un petit tableau explicatif pour mieux comprendre à quel moment ces protocoles sont utilisés lors d'une liaison à un réseau Internet, ce lit de haut en bas et les protocoles sont utilisés dans le même ordre. :

Niveau	Couche	Protocole
1	Physique	Connexion filaire(Ethernet)
2	Liaison	Ethernet
3	Réseau	IP
4	Transport	TCP

Couche Physique : Elle définit la façon dont les " symboles " (petits groupes de bits d'informations) seront convertis en signaux (exemple : électriques, optiques, radio) pour être transportés, ainsi que le support de ce transport (exemple : cuivre, fibre optique).

FIGURE 2.7 – Câble Ethernet

FIGURE 2.8 – Port RJ-45, utilisé pour Ethernet

Couche Liaison : Elle permet l'envoi et la réception de paquets d'informations, appelés aussi trames, entre deux équipements voisins tout en gérant le partage du même support physique à plusieurs(plusieurs machines branchées sur une même machine).
Réseaux : Elle ajoute la notion de routage, route de transport, des paquets d'information depuis une adresse source et en les transférant de proche en proche vers une adresse destination.
Transport : Elle gère les communications de bout en bout entre application. Le plus souvent cette communication se fera octet par octet et sera fiable, ou alors le processus sera prévenu de la perte de la connexion. Cette couche prend donc à sa charge la retransmission d'octets en cas de besoin.

Un serveur TCP/IP : C'est une machine qui reliée à un réseaux, permet de fournir des services aux autres machines de ce même réseaux. Les machines ce connectant à ce serveur sont appelées clients.

2.4.2 Présentation du protocole de cryptage SSL

SSL, Secure Sockets Layer (ou Couche de sécurité), est un protocole permettant de sécuriser une communication entre deux machines, une cliente et l'autre serveur. Dans le modèle des couches des protocoles Internet vu précédemment (page 20), SSL s'utilise juste après la couche 4, celle du transport (TCP).
Ce protocole utilise deux types d'algorithmes pour crypté l'information :

→Un algorithme de cryptage asymétrique

↪ Cet algorithme utilise 2 clés, une pour crypté l'information, appelée clé publique, et une autre pour décrypter l'information, appelée clé privée. La clé publique est donnée à toute machine voulant entamer une "conversation" pour que celle-ci puisse crypter ses messages à envoyer. La clé privée quant à elle n'est jamais divulguée.

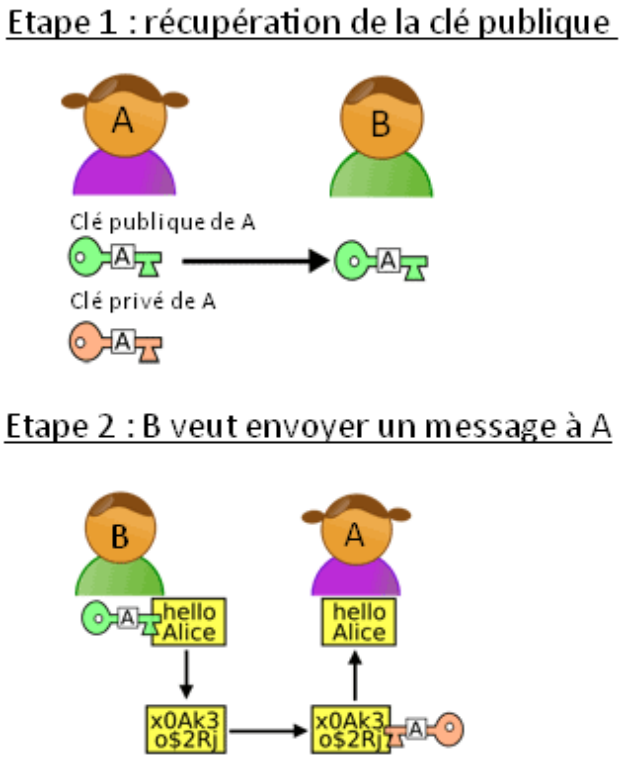


FIGURE 2.9 – Envoie d'un message avec un cryptage asymétrique

Comme on peut le voir, si A veut alors répondre à B, B doit créer lui aussi un jeu de clés et envoyer sa clé publique à A.

→Un algorithme de cryptage symétrique

↪ Cet algorithme n'utilise lui qu'une seule clé, pour crypter et décrypter et doit donc être connue par les deux interlocuteurs.

Le cryptage asymétrique étant gourmand en calcul, SSL l'utilise pour communiquer la clé pour l'algorithme de cryptage symétrique, qui lui demande beaucoup moins de ressources et qui sera utilisé par la suite pour crypter les informations.

Le protocole SSL permet aussi de vérifier l'intégrité, la non-corruption, d'un paquet (ou message) reçu au moyen d'un autre algorithme, appelé algorithme de hachage. Cet algorithme prend le message et créer une "empreinte numérique" de celui-ci. Cette empreinte à une taille fixe quelle que soit la taille du message.
Son principal intérêt est que la moindre petite modification d'un message, le nouveau message aura une empreinte numérique totalement différente.

Dans le cas de SSL le message est envoyé avec son empreinte, et pour vérifier qu'aucune modification n'a eu lieu, il suffit de calculer l'empreinte du message reçu avec l'empreinte fournie avec le message.
La combinaison de ces troix algorithmes nous permet d'assurer la sécurité (personne ne connait le contenu des messages mis à part les deux interlocuteurs), l'intégrité (les messages corrompus sont détectés) et donc son authenticité.
Avant de pouvoir échanger des informations entre le client et le serveur, le protocole SSL établie un "handshake" (ou "poignée de main"), pendant lequel les deux machines se concertent pour la configuration de la liaison (exemple : choix des algorithmes, échange des clés, ...).

L'Handshake :

Cette phase permet d'initialiser différents paramètres entre le client et le serveur avant de commencer l'échange d'informations.
Après s'être concerté sur la version du protocole SSL utilisée (v1, 2, ou 3), le serveur à obligation de donner un certificat d'authentification contenant diverses informations (exemples : pays, ville, entreprise, ...). Si l'utilisateur demande l'accès à des ressources nécessitant une authentification, il demande au client de lui envoyer un certificat.
A ce stade le client (et si nécessaire le serveur) vérifie la validité du certificat qu'il a reçu grâce une 3^{e}entité, que sont les AC, Autorité de Certification.
Les AC peuvent être des entreprises ou le client ou le serveur lui-même. Ils sont ici considérés comme des entités de confiance. Si les certificats ne sont pas validé par les AC, la connexion au serveur échoue.
Si elle réussie, le client et le serveur choisissent les différents algorithmes nécessaires (cryptage et hachage) en fonction de ceux que chacun peut utiliser.
Ensuite ils créent en commun la clé pour l'algorithme de cryptage symétrique et se la partage.
À ce stade l'Handshake est terminé, la connexion sécurisée peut commencer.

Certificat de validation (ou d'authentification) :

Contient plusieurs informations obligatoires, et à la possibilité d'en contenir d'autres si désiré. Pour que le certificat puisse être utilisé, il est nécessaire de le faire certifier auprès d'une AC, Autorité de Certification, qui le signe avec une paire de clé publique/clé privée qui lui est propre. Cette clé publique de l'AC peut à son tour être signée par une autre clé formant ainsi une chaîne. Tout en haut de ces chaînes se trouve le certificat racine, le plus important.
Certificat racine : C'est une clé publique non signée, ou auto-signée, dans laquelle repose la confiance. Si l'AC et le créateur de la clé publique ne font qu'un, le certificat est dit auto-signé.

2.4.3 Fonctionnement de l'application

Dans cette partie je vous explique le fonctionnement et les différents modules de l'application.
J'ai récupéré l'application fournit par Microchip qui permet d'implémenter la création d'un serveur TCP/IP sécurisé. À celui-ci j'y ai ajouté un module d'authentification de l'utilisateur (login et mot de passe), ainsi qu'un module de contrôle des entrées/sorties de la carte électronique via ce serveur.

Les entrées/sorties de la carte :

- 4 bornes d'entrées

FIGURE 2.10 – Photo des bornes de la carte

- 2 relais

FIGURE 2.11 – Photo des relais de la carte

- 4 LEDs (Petites lumières), l'une d'entre elles permet de contrôler le rétro éclairage d'un afficheur LCD.

Cette application ce devant d'être générique, servir de support à d'autre application nécessitant la mise en place d'un serveur, le développeur à la possibilité d'activer ou désactiver, ou modifier différents paramètres du serveur de façon très simple.
Pour cela je me suis inspiré de la méthode de Microchip a utilisé, c'est-à-dire que ces différentes options sont regroupées dans un fichier d'entête et implémentées au travers de définitions de constantes.

```
2  /***CONFIGURATION***/
3
4  //utiliser le protocole SSL pour crypter la communication
5  //commenter pour ne pas la crypter
6  #define GENERIC_USE_SSL
7
8  //utiliser pour activer les commandes de controle
9  // des entrees/sorties ainsi que les commandes de
10 // demande d'information sur celle-ci.
11 //commenter pour desactiver
12 #define GENERIC_USE_IO_CONTROL
13
14 //pour afficher les differents etats du serveur sur la RS-232
15 //commenter pour desactiver
16 #define GENERIC_DEBUG_ON_RS232
17
18 //utiliser une authentification avec login et mot de passe
19 //commenter pour desactiver l'authentification
20 #define GENERIC_USE_AUTHENTIFICATION
21   #define GENERIC_AUTH_DEFAULT_LOGIN      "qqun"
22   #define GENERIC_AUTH_DEFAULT_PASS       "qqch"
23   #define GENERIC_AUTH_BUFF_MAX_LENGTH    20
24
25 //Port de la connexion securisee
26 #define GENERIC_PORT 23
27
28 //Port de la connexion non securisee
29 #define GENERIC_SSL_PORT 4433
30
31 //taille du buffer de reception
32 #define GENERIC_GETBUFF_LENGTH 20
33
34 //caractere ou valeur, mettant fin a la connexion
35 #define GENERIC_BREAK_CAR ('~')
36
37 /***FIN CONFIGURATION***/
```

Listing 2.1 – Une partie du fichier d'en-tête de configuration du serveur

Problèmes rencontrés

Pour l'utilisation du protocole SSL il a été nécessaire d'acheter à Microchip une bibliothèque. Cette bibliothèque contient les codes sources pour utiliser ce protocole. Le code étant donc propriétaire.
Un autre problème rencontré a était la création des clé publique/privé pour le serveur ainsi que de son certificat. En effet c'est avec OpenSSL que j'ai généré chacun d'eux, en suivant les instructions fournit par Microchip pour cela. La difficulté a résidé dans le fait que OpenSSL génère des clés formaté en Big Endian, et le micro-contrôleur est en Little Endian.
Big Endian : C'est l'ordre dans lequel les octet sont organisés en mémoire : plus l'octet est significatif plus son adresse mémoire sera petite.
Little Endian : C'est l'ordre inverse du Big Endian. Plus l'octet est significatif plus son adresse mémoire sera grande.
Exemple : le nombre hexadécimal F09A28A1h (4 octets)

Adresse mémoire	0	1	2	3
En Big Endian	F0h	9Ah	28h	A1h
En Little Endian	A1h	28h	9Ah	F0h

OpenSSL génère 5 clés a intégrés dans le code sources de l'application.
Le certification n'a pas besoin d'etre convertie en Little Endian.
Le fait de convertir de Big Endian vers Little Endian puis de formater cela pour l'intégrer directement dans le code source, de même pour le formatage du certificat, étant très longue et fastidieuse, j'ai réalisé une petite application pour ordinateur en C++ pour créer le code source correspondant aux clés et au certificat. Ce code source généré peu être directement utilisé ensuite dans l'application du micro-contrôleur.

Débugage sur la RS-232

Quand il est activé, permet de fournir, via la RS-232, des informations sur l'état du serveur.

Module d'authentification

Ce module permet de sécuriser l'accès au serveur grâce un login et un mot de passe.
À la connexion d'un utilisateur le serveur demande son login, puis demande son mot de passe, si les deux sont correctes, l'utilisateur à accès aux fonctionnalités du serveurs, sinon le serveur met fin à la connexion.
En cas de problème lors de la connexion (ex : déconnexion), le serveur supprime la session de l'utilisateur, ce qui l'obligera à se ré-authentifier.

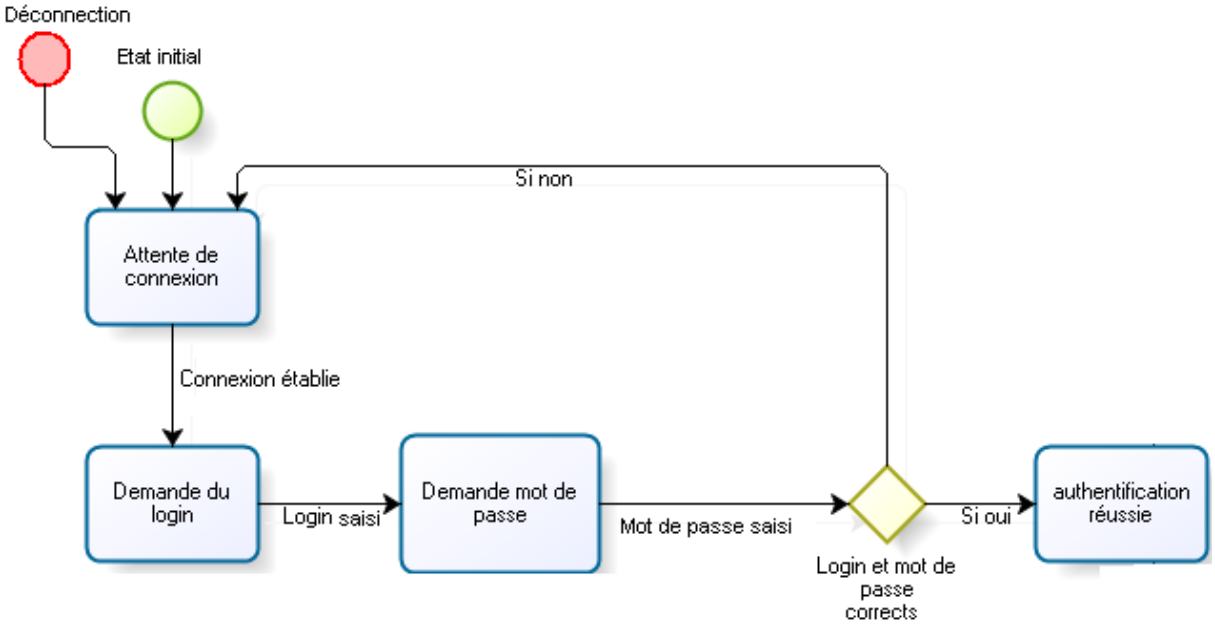


FIGURE 2.12 – Diagramme résumant le fonctionnement du module d'authentification

Module de contrôle des entrées/sorties

Ce module est disponible pour l'utilisateur après l'authentification, si elle a été activée. Il permet à l'utilisateur de contrôler l'état des entrées et des sorties présentent sur la carte électronique. Pour cela plusieurs commandes ont été crées permettant d'allumer/éteindre les LED, activer/désactiver les relais, connaître l'état des 4 entrées, connaître l'état des LEDs et finalement connaître l'état des relais.
En annexe B(page 43) est présenté le code source pour récupérer l'état des LED et activer les LEDs. Ces codes sources sont présentés en exemples représentatifs, les autres commandes fonctionnant quasiment de la même façon, leurs codes sources ne sont donc pas présentées.
Connaître l'état des LEDs :
Commande à émettre :

→ Taille : 1 octet

→ Valeur : 't' (valeur ASCII de la lettre t)

Réponse du serveur :

→ Taille : 1 octet

→ Format :

Bits :	bits 7 à 4	bit 3	bit 2	bit 1	bit 0
Valeurs :	3h	État LED 3	État LED 2	État LED 1	État LED 0

LED 0 : LED orange
LED 1 : LED rouge 1
LED 2 : LED rouge 2
LED 3 : Les 2 LEDs de rétro-éclairage de l'écran LCD

xh : Nombre hexadécimal

Connaître l'état des entrées :

Commande à émettre :

→ Taille : 1 octet

→ Valeur : 'e' (valeur ASCII de la lettre e)

Réponse du serveur :

→ Taille : 1 octet

→ Format :

Bits :	bits 7 à 4	bit 3	bit 2	bit 1	bit 0
Valeurs :	3h	État entrée 4	État entrée 3	État entrée 2	État entrée 1

xh : Nombre hexadécimal

Connaître l'état des relais :

Commande à émettre :

→ Taille : 1 octet

→ Valeur : 'a' (valeur ASCII de la lettre a)

Réponse du serveur :

→ Taille : 1 octet

→ Format :

Bits :	bits 7 à 4	bit 3	bit 2	bit 1	bit 0
Valeurs :	3h	0	0	État relai 2	État relai 1

xh : Nombre hexadécimal

Activer/Désactiver les relais :

Commande à émettre :

→ Taille : 2 octets

→ 1er octet pour activer : 'S'(valeur ASCII de la lettre S)

→ 1er octet pour désactiver : 's'(valeur ASCII de la lettre s)

→ Format du 2^{e} octet :

Bits :	bits 7 à 4	bit 3	bit 2	bit 1	bit 0
Valeurs :	3h	0	0	Relai 2	Relai 1

xh : Nombre hexadécimal

Si on est en activation : un bit à 1 signifie l'activation du relais correspondant.

Si on est en désactivation : un bit à 1 signifie la désactivation du relais correspondant.
Dans les 2 cas un bit à 0 pour un relais signifie qu'il conservera son état.

Réponse du serveur : Aucune

Activer/Désactiver les LEDs :

Commande à émettre :

→ Taille : 2 octets

→ 1er octet pour activer : 'L'(valeur ASCII de la lettre L)

→ 1er octet pour désactiver : 'l'(valeur ASCII de la lettre l)

→ Format du 2e octet :

Bits :	bits 7 à 4	bit 3	bit 2	bit 1	bit 0
Valeurs :	3h	LED 3	LED 2	LED 1	LED 0

xh : Nombre hexadécimal
Si on est en activation : un bit à 1 signifie l'activation de la LED correspondante.
Si on est en désactivation : un bit à 1 signifie la désactivation de la LED correspondante.
Dans tous les cas un bit à 0 pour une LED signifie qu'elle conservera son état.

Réponse du serveur : Aucune

2.5 Contrôle d'une carte via RS-232

Pour ce projet mon tuteur m'a fait travailler sur carte embarquant un micro-contrôleur pic16F628A présenté en annexe D (page 47). Il m'a demandé de fournir une application permettant de contrôler ses entrées et sorties via la RS-232 (cf : La RS-232 page 11).

Contexte

Cette carte, moins chère que celle embarquant un pic18F97J60, permet de fournir des services similaires, l'Ethernet mis à part. Elle servira elle aussi pour le contrôle de mangeoires dans les Unités Expérimentales mais de façon non-autonome : elle sera directement branchée, via la RS-232, à un ordinateur pour piloté des mangeoires. Elle servira donc de périphérique à un ordinateur.

2.5.1 Fonctionnement de l'application

L'application consiste en une boucle infinie attendant des ordres, venant de la RS-232 (cf : La RS-232 page 11), et les exécutants.
Cette application se rapproche énormément du module de gestion des entrées/sorties pour l'Ethernet (page 27), à la différence qu'ici les informations et les commandes transitent par la RS-232 et non par Ethernet.
Cette application fournit les commandes permettant d'activer/désactiver des relais, connaître l'état des relais, et connaître l'état des entrées.

Présentation des entrées/sorties :

- 4 bornes d'entrées

FIGURE 2.13 – Photo des bornes d'entrées

- 4 relais

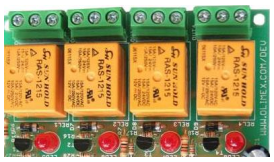


FIGURE 2.14 – Photo des relais

Présentation des commandes

Le traitement des commandes est similaire au traitement du module de contrôle des entrées/sorties par Ethernet (page 27). En annexe C (page 45) est présenté 2 exemples représentatifs de code source du traitement des commandes, l'un de récupération de l'état des entrées, l'autre pour l'activation des relais.

Connaître l'état des entrées :

Commande à émettre :

→ Taille : 1 octet

→ Valeur : 'i' (valeur ASCII de la lettre i)

Réponse du serveur :

→ Taille : 1 octet

→ Format :

Bits :	bits 7 à 4	bit 3	bit 2	bit 1	bit 0
Valeurs :	3h	État entrée 4	État entrée 3	État entrée 2	État entrée 1

xh : Nombre hexadécimal

Connaître l'état des relais :

Commande à émettre :

→ Taille : 1 octet

→ Valeur : 'r' (valeur ASCII de la lettre r)

Réponse du serveur :

→ Taille : 1 octet

→ Format :

Bits :	bits 7 à 4	bit 3	bit 2	bit 1	bit 0
Valeurs :	3h	État relai 4	État relai 3	État relai 2	État relai 1

xh : Nombre hexadécimal

Activer/Désactiver les relais :

Commande à émettre :

→ Taille : 2 octets

→ 1er octet pour activer : 's'(valeur ASCII de la lettre s)

→ 1er octet pour désactiver : 'c'(valeur ASCII de la lettre c)

→ Format du 2^{e} octet :

Bits :	bits 7 à 4	bit 3	bit 2	bit 1	bit 0
Valeurs :	3h	Relai 4	Relai 3	Relai 2	Relai 1

xh : Nombre hexadécimal
Si on est en activation : un bit à 1 signifie l'activation du relais correspondant.
Si on est en désactivation : un bit à 1 signifie la désactivation du relais correspondant.
Dans les 2 cas un bit à 0 pour un relais signifie qu'il conservera son état.

Réponse du serveur : Aucune

Problème technique important rencontré

L'application en elle-même ne fut pas la plus importante difficulté rencontrée au cours de ce projet. Le plus gros problème a été de résoudre le fait que la carte ne ce lançait pas tout le temps quand elle était mise sous tension.
La carte avait un fonctionnement très erratique et donc n'était pas utilisable. Après plusieurs essais de méthodes différentes pour y mettre un terme(debugage, ajout de condensateur sur l'alimentation, configuration du micro-contrôleur), j'ai fini par découvrir le problème.
Il a été nécessaire d'activer un bit servant à la configuration du micro-contrôleur qui permet de détecter les chutes de tension, et de désactiver le micro-contrôleur quand elles surviennent.

2.6 Gestionnaire de communication

Pour ce projet mon tuteur m'a demandé de compléter une application embarquée sur une machine, qui gère des balances pour des pesées. Cette application fut la plus difficile et longue à réaliser dû aux nombreux problèmes rencontrés.

Contexte

La machine appelée AGPA (présenté ci-dessous), qui permet de gérer des balances, est présente dans de nombreux élevages. Le problème qui c'est produit, c'est que cette machine ne se fabrique plus.
Plutôt que de remplacer tous les AGPAs par une autre machine, il a été décidé de les modifier pour qu'ils se comportent exactement de la même façon que la nouvelle machine, le TEO, qui sera désormais installé dans les élevages. Ainsi les machines coexisterons dans les élevages, avec un comportement identique ce qui permettra de communiquer de manière uniforme avec les deux. Les 2 machines utilisent le Bluetooth, communication sans fil, pour être contrôlé et échanger des informations à une 3^{e}machine, le PDA, présenté ci-dessous.
Dans le cas du AGPA on lui ajoute sur le port de la RS-232 une clé Bluetooth pour lui permettre de communiquer via le Bluetooth.

2.6.1 Présentation des machines

Le TEO

Machine permettant de contrôler 2 balances, ou plateaux, via des liaisons UART (cf : L'UART page 10). Le TEO communique par Bluetooth pour recevoir des ordres ou transmettre des informations sur des traitements qu'il effectue avec les plateaux.

FIGURE 2.15 – Photo d'un TEO

Le AGPA

Cette machine à la même utilité que le TEO, à la différence qu'elle ne dispose pas du Bluetooth intégré pour recevoir des ordres ou émettre des informations. Pour cela elle dispose d'un port RS-232 (cf : la RS-232 page 11) sur lequel a été branché une clé Bluetooth.

FIGURE 2.16 – Photo d'un TEO

FIGURE 2.17 – Photo de la clé Bluetooth

Le PDA

Le PDA, ou Personal Digital Assitant, est un appareil qui permet d'aider au quotidien des personnes. Cet appareil généralement de la taille d'un gros téléphone mobile, peut être comparé à un petit ordinateur et peut servir à de nombreuse tâches : agenda, carnet d'adresses, répertoire, etc.
Dans les Unités Expérimentales de l'INRA il permet d'aider au quotidien les chercheurs, au travers d'applications spécialisées pour les PDAs, de communiquer avec les machines ou les bases de données, par exemple, directement sur le terrain.

FIGURE 2.18 – Photo d'un TEO

2.6.2 Introduction au Bluetooth

C'est une technologie permettant la communication sans fil sur de courtes distances. Elle utilise le support radio pour transporter les informations. Cette spécification a été développée dans le but de supprimer les câbles des périphériques reliés à un ordinateur (ex : imprimantes, souris, clavier, etc.). De nos jours avec les progrès qui ont été réalisés, on retrouve cette technologie dans de nombreux appareils tels que les téléphones portables, PDAs, kits main-libre, lecteurs de code barres, etc.).

Intérêts :

- Très faible consommation
- Faible portée
- Bon marché et peu encombrant

Un mode d'utilisation que permet le Bluetooth, appelé "RFCOMM", permet d'utiliser le dispositif implémentant le Bluetooth comme une liaison UART. Ceci permet, une fois le dispositif Bluetooth configuré, de simplifier la communication entre les 2 machines utilisant le Bluetooth pour dialoguer.
Le dispositif Bluetooth récupère, via l'UART, les informations à envoyer, et fais tout le nécessaire pour envoyer ces informations à l'autre machine par le Bluetooth. Quand il reçoit des informations de l'autre machine par le Bluetooth il les envoie sur l'UART.
Ainsi pour la machine, l'utilisation du Bluetooth lui est totalement inconnue et simplifie le travail du développeur qui à juste à configurer le dispositif Bluetooth et à le brancher à la

liaison UART de la machine sans se soucier du fonctionnement complexe du Bluetooth.
Le développeur a donc juste à gérer l'UART du micro-contrôleur pour dialoguer avec l'autre machine, et de même pour l'autre machine.

2.6.3 Fonctionnement

Le TEO, la nouvelle machine, dispose de nombreuses commandes qui font son comportement.
Ainsi le AGPA doit pouvoir répondre de la même façon, et pouvoir effectuer les mêmes traitements que le TEO, aux ordres qu'il reçoit.
En annexes F (page : 53) est présentée la liste des commandes du TEO ainsi que les réponses à ces commandes.
Le AGPA disposant d'une interface graphique, il a été nécessaire de la gérer.

Problèmes techniques importants rencontrés

L'application a été développée à l'origine pour une version de compilateur très ancienne.

Le problème qui a eu lieu c'est que l'on c'est retrouvé dans l'impossibilité de porter le projet sur une version plus récente du compilateur, ni de l'éditeur de code source et non plus sur un autre ordinateur.(Je vais travailler sur ce problème durant mon CDD suivant ce stage). L'éditeur de code source ayant des fonctionnalités très limitées j'ai dû utiliser un 2^{e}ordinateur pour analyser le code source, et utiliser le 1erpour éditer et compiler.

Un autre problème qui a été rencontré était que l'application utilise de nombreuses variables globales, et l'espace mémoire restant étant très limité, j'ai dû les utiliser ce qui m'a pausé de nombreux problèmes d'effet de bord tout au long du projet. Un effet de bord représente un comportement non prévu dans l'application dû à l'utilisation de ressources partagées entre différents traitements, dans notre cas ces ressources sont des variables.

Pour terminer un dernier problème a été rencontré, celui de l'espace limité de la mémoire pour le programme. Vers la fin du projet j'ai rencontré ce problème qui a été résolu en analysant mon code source pour le réduire au maximum pour pouvoir ainsi continuer et finir le projet.

Point technique : La pesée stabilisée

J'ai choisi la commande de pesée stabilisée comme poins technique à aborder car elle regroupe et représente de nombreux problèmes auxquels j'ai dû faire face.
Je ne présente pas le code source dans sa totalité, mais seulement la description complète de

la fonction chargée de ce traitement. Le code source complet est fourni et décrit en annexe E (page 48).

La pesée stabilisée consiste à réaliser une série de N pesée et d'en faire la moyenne.

Plusieurs paramètres peuvent être utilisés pour configurer une pesée stabilisée qui sont :

- L'amplitude : Différence maximum entre le poids maximum et le poids minimum mesurés au cours de la série
- Le nombre de mesures d'une série

Ce traitement recommence une nouvelle série si l'amplitude maximum est dépassé.

```
1   /* @details Permet de faire une pesee stabilisee, et d'afficher
       une bare de progression sur l'ecran du AGPA.
2    *
3    * @param unsigned char num Numero du plateau (PLT1 ou PLT2)
4    * @param unsigned int* raterTotal Pointeur sur une variable
       permettant de stocke le nombre de series qui ont echoue
5    * @param unsigned int* raterInf5 Pointeur sur une variable
       permettant de stocke le nombre de serie qui ont echoue
       avant 5 pesees
6    *
7    * @pre (pres-conditions)
8    *- La variable globale ampproto contient l'amplitude voulu
       pour la pesee stabilisee
9    *- La variable globale nbremes contient le nombre de pesee
       pour une serie de pesee
10   *- CAL_TARE contient la valeur de la tare du plateau sur
       lequel on  fait la tare
11   *
12   * @pos (post-conditions)
13   *- CAL_POIDS contient la valeur de la pesee stabilisee
14   *- CAL_PMIN  contient la valeur la plus faible enregistree
       durant la derniere serie de la pesee
15   *- CAL_PMAX  contient la valeur la plus elevee enregistree *
                  durant la derniere serie de la pesee
16   *
17   * @return :
18   *0x00 pesee stabilisee effectuee
19   *0x01 probleme de tare pesee non effectuee
20   *0x02 erreur de pesee, code de l'erreur dans trvl0, pesee non
       effectuee
21   *0x03 Message d'arret de la pesee recu, pesee stabilisee non
       effectuee
22   */
```

Listing 2.2 – Code source : Description de la fonction permettant la pesée stabilisée

Comme le montre cette description, les différents problèmes cités précédemment sont mis-en évidences : variable globale et possible effets de bord causé par les nombreuses variables globales utilisées pour ce traitement et qui servent, pour certaines, à des traitements totalement différents pas la suite.
Elle montre aussi que l'interface graphique est utilisée pour l'affichage d'une barre de progression, qui indique le nombre de mesures effectuées de la série de pesées en cours.

Conclusion

Les quatre projets ont été terminés avec succès. Les deux applications permettant respectivement de créer un serveur TCP/IP et celle de contrôle d'une carte électronique par un ordinateur, seront utilisées dans divers projets à venir. En effet, suivant ce stage j'effectue un CDD durant lequel je travaillerai sur une application spécifique pour un projet, en utilisant l'application de création de serveur.

D'un point de vue professionnel ce stage m'a apporté des connaissances sur la mise en place de différents systèmes de communication ainsi que d'outils que sont :

- La mise en place de serveur TCP/IP sécurisé, HTTP et HTTPS pour le contrôle via un réseaux tel qu'Internet
- L'UART pour le contrôle ou la communication via port série
- SSL pour la sécurisation de communication sur un réseau

D'un point de vue plus personnel j'ai pu affiner mes qualités en tant que personne au sein d'une entreprise afin de mener un projet à son terme :

- L'autonomie : on m'a donné plusieurs missions et on m'a laissé libre de mes options (cycle de développement, organisation des tâches, des outils, des tests, etc.) durant des délais impartis
- Responsabilité : par la confiance que l'entreprise a placé en moi, et plus particulièrement mon tuteur, pour réaliser des objectifs professionnels dans des un délais imparties

Bibliographie

SSL :

- http://www.openssl.org/ : Application OpenSSL et sa documentation
- http://fr.wikipedia.org/wiki/Secure_Sockets_Layer : documentation
- http://en.wikipedia.org/wiki/Transport_Layer_Security : documentation
- http://fr.wikipedia.org/wiki/X.509 : documentation
- http://sebsauvage.net/comprendre/ssl/ : documentation
- http://www.madboa.com/geek/openssl/ : Ensemble de petits tutoriels pour OpenSSL
- http://www.microchipdirect.com/productsearch.aspx?Keywords=sw300052 : Bibliothèque vendue par Microchip pour implémenter SSL

Serveur TCP/IP :

- http://fr.wikipedia.org/wiki/Suite_des_protocoles_Internet : documentation
- TCPIP Stack Help.pdf : documentation fourni avec les exemples de serveurs fournis par Microchip
- http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2680&dDocName=en537041 : Sources pour la création d'un serveur

Data sheet utilisées :

- http://ww1.microchip.com/downloads/en/DeviceDoc/39762f.pdf : pic micro-contrôleur 18F96J60
- http://ww1.microchip.com/downloads/en/DeviceDoc/40044G.pdf : micro-contrôleur pic 16F628A

Cartes électroniques :

- `http://www.olimex.com/dev/pic-maxi-web.html` : documentation de la carte embarquant le pic 18f96J60
- `http://www.olimex.com/dev/pic-io.html` : documentation de la carte embarquant le pic 16F628A

Annexes

A Carte électronique avec pic18f97J60

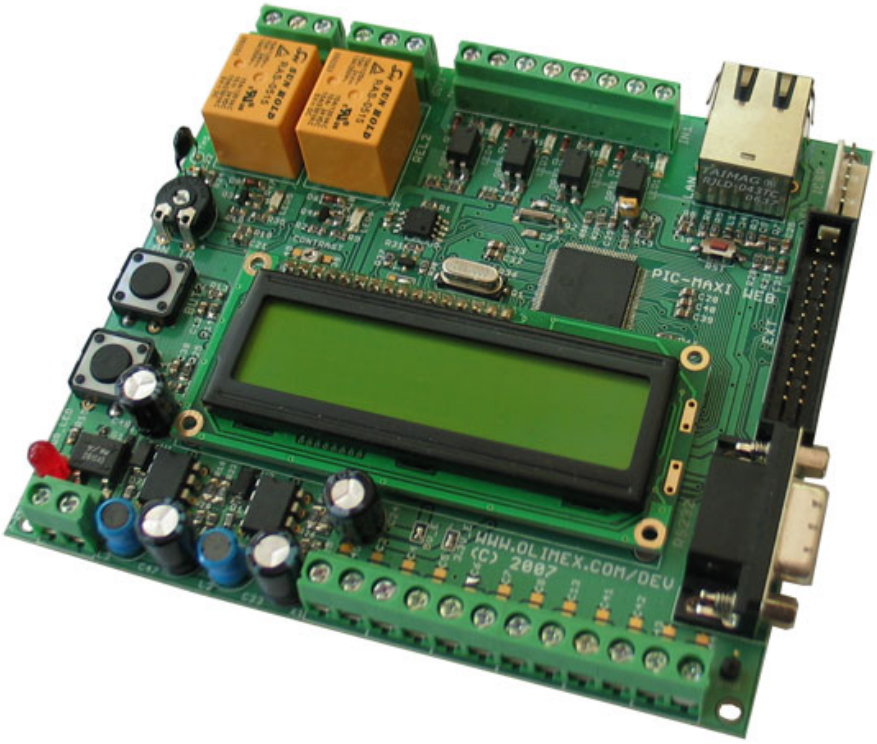


FIGURE A.1 – Carte électronique

Cette carte électronique a été fournie par l'entreprise Olimex.
Pour nos projets nous utilisons de cette carte :

- 1 LED orange
- 2 LEDs rouges
- 1 Afficheur LCD avec 2 LEDs de rétro-éclairage
- 2 Relais
- 4 Entrées
- 1 Port Ethernet
- 1 Port pour la RS-232
- 2 Boutons

B Présentation de code source : récupération des informations des LEDs et activations de celle-ci (projet Serveur TCP/IP sur carte)

```
1  //TCP_SOCKET* MySocket : socket courant, il permet d'envoyer le
       message
2  void GenericIOControl_InfoLed(TCP_SOCKET* MySocket)
3  {
4    uchar ret = 0x30; // '0'
5
6    //Etat LED0 (orange)
7    if(LED0)
8      ret += 0x08;
9
10   //Etat LED1 (rouge)
11   if(LED1)
12     ret += 0x04;
13
14   //Etat LED2 (rouge)
15   if(LED2)
16     ret += 0x02;
17
18   //Etat LED3 et LED4(LCD)
19   if(LED3)
20     ret += 0x01;
21   //envoi du message a l'utilisateur
22   TCPPut(*MySocket, ret);
23 }
```

Listing B.1 – Code source : Récupération de l'état des LEDs

```
1  //BYTE car : Caractere recu contenant les informations sur les LEDs
       a activer
2  void GenericIOControl_setLed(BYTE car)
3  {
4    if(car <= 0x30 || car > 0x3F)
5      return;
6    //set LED 0 (orange)
7    if((car&0x08) == 0x08)
8      LED0 = 1;
9
10   //set LED 1 (rouge 1)
11   if((car&0x04) == 0x04)
12     LED1 = 1;
13
14   //set LED 2 (rouge 2)
15   if((car&0x02) == 0x02)
16     LED2 = 1;
17
18   //set LED 3 et 4 (LCD)
19   if((car&0x01) == 0x01)
20     LED3 = 1;
21 }
```

Listing B.2 – Code source : Activation des LEDs

C Présentation de code source : récupération des informations des entrées et activations des relais (projet Contrôle d'une carte via RS-232)

```
1  void infoIn()
2  {
3    uchar ret = 0x30;
4
5    //IN 1
6    if(!IN1)
7      ret += 0x08;
8
9    //IN 2
10   if(!IN2)
11     ret += 0x04;
12
13   //IN 3
14   if(!IN3)
15     ret += 0x02;
16
17   //IN 4
18   if(!IN4)
19     ret += 0x01;
20
21   //envoie du message sur la RS-232
22   WriteByte(ret);
23 }
```

Listing C.1 – Code source : Récupération de l'état des entrées

```
1  void setRelai()
2  {
3    uchar car;
4    //attente de la reception d'un octet sur la RS-232
5    while(!RCIF)
6    {
7    }
8    car = RCREG;  //recuperation de l'octet
9    if(car <= 0x30 || car > 0x3F)
10     return;
11   //set relai 1
12   if((car&0x08) == 0x08)
13     REL1 = 1;
14
15   //set relai 2
16   if((car&0x04) == 0x04)
17     REL2 = 1;
18
19   //set relai 3
20   if((car&0x02) == 0x02)
21     REL3 = 1;
22
23   //set relai 4
24   if((car&0x01) == 0x01)
25     REL4 = 1;
26 }
```

Listing C.2 – Code source : Activation des relais

D Carte électronique avec pic16F628A

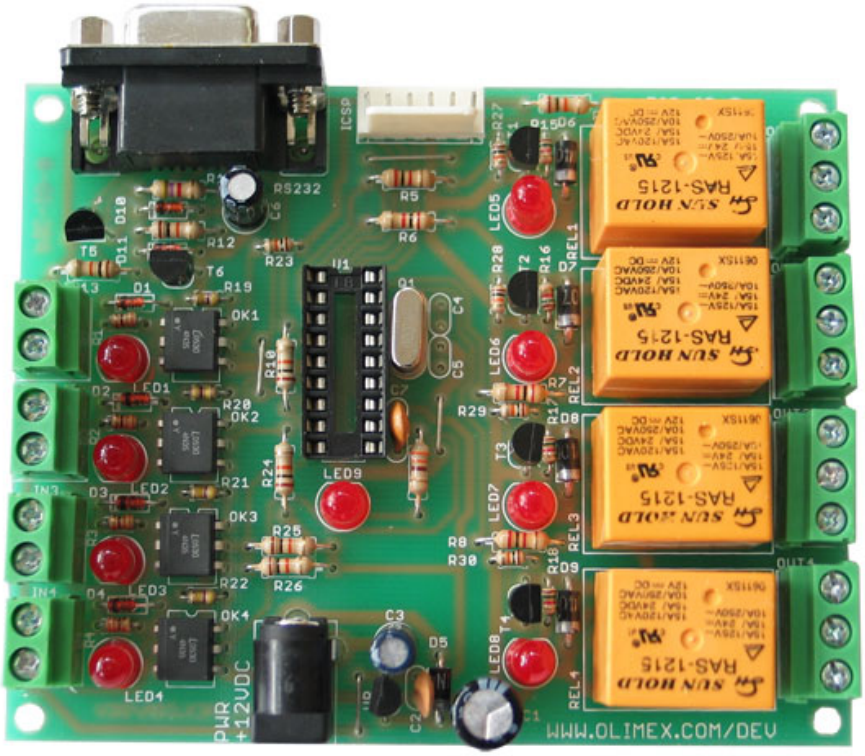


FIGURE D.1 – Carte électronique

Cette carte électronique a été fournie par l'entreprise Olimex.
Pour nos projets nous utilisons de cette carte :

- 1 LED rouge
- 4 Relais
- 4 Entrées

E Présentation code source : La commande de pesée stabilisée

Pour le fonctionnement détaillé ainsi que les commentaires concernant les paramètres et les retours de la fonction, cf : "Point technique : La pesée stabilisée" page 35.

```
1  unsigned char pmesure_rs(unsigned char num, unsigned int*
       raterTotal, unsigned int* raterInf5)
2  {
3    //nombre de mesure
4      xdata unsigned char dummy;
5      //somme des mesures
6      xdata unsigned long somme = 0;
7      //utiliser pour introduire un delai
8      //et laisser le temps au uP de recevoir
9      //sur la RS-232
10     xdata unsigned int rs;
11     //nombre de caractre lu sur la RS-232
12     unsigned char indexMsg;
13     //longeur de la bare de progression
14     unsigned char carl;
15
16   //remise a zero du nombre d'echec
17     *raterTotal = 0;
18     *raterInf5 = 0;
19   //remise a zero de la bare de progression
20     carl = 0;
21   //si le nombre mesure pour une serie est trop faible
22   //l'initialiser avec un minimum : 2
23     if(nbremes == 0 || nbremes == 1)
24         nbremes = 2;
25
26   // conversion amplitude en int
27   //ampproto est une chaine de caractere
```

```
28        AMP = conv_amp(ampproto);
29
30        // nbremes mesure + 1, pour le 1er Pmin et et Pmax
31        for (dummy = 0; dummy < nbremes + 1; dummy++)
32        {
33            SCL = ~SCL; //relance du watchdog
34
35        //selection du port RS-232 de reception
36        //de commande
37            select_rs(MICRO, BD_9600);
38            //remis a zero du bit indiquant qu'un
39            //caractere a ete recu sur la RS-232
40            RECEPT_FCT = 0x00;
41
42            for(rs = 0; rs < 0x1FFF; rs++)
43            {
44              // un caractere a ete lu
45                if(RECEPT_FCT != 0x00)
46                {
47                    RECEPT_FCT = 0x00;
48            // si un debut de message a ete lu
49                    if(CAR == 0x02)
50                    {
51                        RECEPT_FCT = 0x00;
52                        //remis a zero de l'index du
53                        // buffer de reception
54                        indexMsg = 0;
55
56                        //boucle de reception du reste du message
57                        while(1)
58                        {
59                            if(RECEPT_FCT != 0x00)
60                            {
61                                RECEPT_FCT = 0x00;
62                                //si c'est la fin du message
63                                if(CAR == 0x0D)
64                                {
65                                    RECEPT_FCT = 0x00;
66                                    //si demande d'arret de la pesee
                                        stab
67                                    if(pdstrent[0] == 'i')
68                                    {
```

```
69                                      if(num == PLT1)
70                                        // effacement bare de
                                             progression PLT1
71                                    Af_Blanc(13, 3, carl+1);
72                                  else
73                                    // effacement bare de progression PLT2
74                                    Af_Blanc(13, 7, carl+1);
75                                          return 0x03;
76                                      }
77                                      break;
78                                    }
79                                    pdstrent[indexMsg] = CAR;
80                                    indexMsg++;
81                                    RECEPT_FCT = 0x00;
82                                    // si le message est trop long
83                                    if(indexMsg == 2)
84                                      // on sort de la boucle de reception
85                                      break;
86                                }
87                              }
88                          }
89                      }
90                  }
91          // effectue une pesee
92              trvl0 = lectpoi(num);
93          // si il y a une erreur
94              if (trvl0 != 0)
95              {
96                  if(num == PLT1)
97                    // effacement bare de progression PLT1
98                      Af_Blanc(13, 3, carl+1);
99                  else
100                   // effacement bare de progression PLT2
101                     Af_Blanc(13, 7, carl+1);
102                 return(0x02);
103             }
104         // Probleme de tare
105             if (CAL_POIDS < CAL_TARE)
106             {
107                 if(num == PLT1)
108                   // effacement bare de progression
109                     Af_Blanc(13, 3, carl+1);
```

```
110                 else
111                   // effacement bare de progression
112                     Af_Blanc(13, 7, carl+1);
113
114                 return(0x01);
115             }
116             // 1ere pesee rangement en Pmin et Pmax
117             if (dummy == 0)
118             {
119                 CAL_PMIN = CAL_POIDS;
120                 CAL_PMAX = CAL_POIDS;
121             }
122             else
123             {
124
125                 if(num == PLT1)
126                     CursorXY(13+carl,3);
127                 else
128                     CursorXY(13+carl,7);
129                 //affichage bare de progression
130                 CarSED('*');
131                 carl++;
132
133                 if (CAL_POIDS < CAL_PMIN)
134                     CAL_PMIN = CAL_POIDS;
135                 else if (CAL_POIDS > CAL_PMAX)
136                     CAL_PMAX = CAL_POIDS;
137
138         // PMAX - PMIN f(plateau)
139                 if (defplat < 3)
140                     CAL_AMP = CAL_PMAX - CAL_PMIN;
141                 else if ((defplat > 2) && (defplat < 6))
142                     CAL_AMP = (CAL_PMAX - CAL_PMIN) * 10;
143                 else if ((defplat > 5) && (defplat < 0x0A))
144                     CAL_AMP = (CAL_PMAX - CAL_PMIN) * 100;
145                 else
146                     CAL_AMP = (CAL_PMAX - CAL_PMIN) * 1000;
147
148         // test amplitude
149                 if (CAL_AMP > AMP)
150                 {
151
```

```
152                     if(num == PLT1)
153                         Af_Blanc(13, 3, carl+1);
154                     else
155                         Af_Blanc(13,7, carl+1);
156 
157                     carl = 0;
158 
159                     (*raterTotal)++;
160                     if((dummy) < 5)
161                         (*raterInf5)++;
162             // recommence la serie de 10, avec
163             //l'incrementation en fin de boucle dummy reviendra a 0
164                     dummy = 0xFF;
165                     somme = 0;
166                 }
167                 else
168                   // somme des poids
169                     somme = somme + (long)CAL_POIDS;
170             }
171         }//fin mesure stab
172 
173       // Calcul de la moyenne des poids mesures
174         somme = somme / (nbremes);
175         // tranfert et resultat dans CAL_POIDS
176         CAL_POIDS = somme;
177         // detarage
178         CAL_POIDS = CAL_POIDS - CAL_TARE;
179         CAL_PMAX = CAL_PMAX - CAL_TARE;
180         CAL_PMIN = CAL_PMIN - CAL_TARE;
181         if(num == PLT1)
182           //effacement de la bare de progression PLT1
183             Af_Blanc(13, 3, carl+1);
184         else
185           //effacement de la bare de progression PLT2
186             Af_Blanc(13,7, carl+1);
187 
188         return(0x00);
189     }
```

Listing E.1 – Code source : Code source de la pesée stabilisée

Comme on peut le voir dans ce code source l'utilisation de variable globale est abondante.

F Liste des commandes du TEO

Tare plateau 1 ou 2 :

Description : Permet de demander au TEO de faire la tare du plateau 1 ou du plateau 2 et de récupérer la valeur de la tare.

Commande à émettre :

→ Taille : 3 octet

→ Format :

Octets :	0	1	2
Valeurs :	02h	't'(Plateau 1) ou 'z'(Plateau 2)	0Dh

xxh : Nombre hexadécimal

Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Réponse 1.1 : Tare échouée

→ Taille : 6 octets

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	'1' ou '2'	N° du plateau
2	xxh	Type du plateau
3	xxh	Type de l'erreur
4	'0' ou '1'	'0' si le poids est nul, '1' sinon
5	0Dh	Fin du message

xxh : Nombre hexadécimal

Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Réponse 1.2 : Tare réussie

→ Taille : Variable

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	'1' ou '2'	N° du plateau
2	xxh	Type du plateau
3	'0'	Aucune erreur
4	'0' ou '1'	'0' si le poids est nul, '1' sinon
5 à X	Nombre flottant (ASCII)	Valeur de la tare
X+1	0Dh	Fin du message

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Zéro plateau 1 ou 2 :
Description : Permet de demander au TEO de faire le zéro sur le plateau 1 ou le plateau 2.
Commande à émettre :

→ Taille : 3 octets

→ Format :

Octets :	0	1	2
Valeurs :	02h	'z'(Plateau 1) ou 'd'(Plateau 2)	0Dh

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).
Réponse : aucune

Passer en mode attente d'ordre :
Description : Permet de demander au TEO de rentrer en mode " attente d'ordre ".
Commande à émettre :

→ Taille : 3 octets

→ Format :

Octets :	0	1	2
Valeurs :	02h	'a'	0Dh

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).
Réponse :

→ Taille : 5 octets

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	' '	Espace
2	' '	Espace
3	52h	Accusé de réception
4	0Dh	Fin du message

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Demande de pesée stabilisé sur le plateau 1 ou 2 :
Description : Permet de demander une pesée stabilisé sur le plateau 1 ou 2. Une pesée stabilisé consiste en à faire la moyenne du poids de plusieurs pesée pour obtenir une pesée plus proche de la réalité si l'animal pesée bouge sur le plateau.
Elle prend plusieurs paramètre comme l'amplitude qui correspond à l'écart maximum autorisé entre le poids minimum et le poids maximum enregistrés, de la série de pesée en cours. Si l'amplitude est dépassée (ex : un animal trop nerveux) alors une nouvelle série recommence.
Durant ce traitement le TEO peut recevoir un ordre d'annulation de la pesée stabilisée.
Commande à émettre :

→ Taille : 9 octets

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	'h'	Commande
2	'1' ou '2'	N° du plateau
3 à 8	Nombre flottant (ASCII)	Valeur de l'amplitude maximum
9	'0' à '9' ou 'A'	Nombre de pesée pour une pesée ('A' = 10)
10	0Dh	Fin du message

Réponse 1 : Accusée de réception de la commande

→ Taille : 5 octets

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	' '	Espace
2	' '	Espace
3	53h	Accusé de réception
4	0Dh	Fin du message

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).
Réponse 2.1 : Pesée stabilisée échouée

→ Taille : 6

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	'1' ou '2'	N° du plateau
2	xxh	Type du plateau
3	xxh	Type de l'erreur
4	'0' ou '1'	'0' si le poids est nul, '1' sinon
5	0Dh	Fin du message

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).
Réponse 2.2 : Pesée stabilisée réussie

→ Taille : Variable

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	'1' ou '2'	N° du plateau
2	xxh	Type de plateau
3	'0'	Pas d'erreur
4	'1'	Poids > 0
5 à 11	Nombre flottant (ASCII)	Valeur de la pesée
12	'M'	Délimiteur poids maximum
13 à 19	Nombre flottant (ASCII)	Valeur de la pesée maximum
20	'm'	Délimiteur poids minimum
21 à 27	Nombre flottant (ASCII)	Valeur de la pesée minimum
28	'D'	Délimiteur heure de début de pesée
29 à 34	hhmmss (ASCII)	Heure du début de la pesée
35	'F'	Délimiteur heure de fin de pesée
36 à 41	hhmmss (ASCII)	Heure de fin de la pesée
42	'S'	Délimiteur nombre de séries de pesées ratées
43 à X	Nombre entier (ASCII)	Nombre de séries de pesées ratées
X+1	's'	Délimiteur nombre de séries ratées avant 5 pesées
X+2 à X+2+Y	Nombre entier (ASCII)	Nombre de séries ratées avant 5 pesées
X+2+Y+1	0Dh	Fin de message

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Demande d'arrêt de la pesée stabilisée :

Description : Permet d'arrêter une pesée stabilisée en cours. Cette commande ne fonctionne que si le TEO effectue une pesée stabilisée.
Commande à émettre :

→ Taille : 3 octets

→ Format :

Octets :	0	1	2
Valeurs :	02h	'i'	0Dh

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).
Réponse : aucune

Demande de simple pesage sur le plateau 1 ou 2 :

Description : Permet de demander au TEO d'effectuer un seul pesage sur le plateau 1 ou 2 et de communiquer le poids mesuré.

Commande à émettre :

→ Taille : 3 octets

→ Format :

Octets :	0	1	2
Valeurs :	02h	'c'(Plateau 1) ou 'f'(Plateau 2)	0Dh

xxh : Nombre hexadécimal

Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Réponse 1.1 : Pesée simple échouée

→ Taille : 6 octets

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	'1' ou '2'	N° du plateau
2	xxh	Type du plateau
3	xxh	Type de l'erreur
4	'0' ou '1'	'0' si le poids est nul, '1' sinon
5	0Dh	Fin du message

xxh : Nombre hexadécimal

Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Réponse 1.2 : Pesée simple réussie

→ Taille : Variable

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	'1' ou '2'	N° du plateau
2	xxh	Type du plateau
3	'0'	Aucune erreur
4	'0' ou '1'	'0' si le poids est nul, '1' sinon
5 à X	Nombre flottant (ASCII)	Valeur de la pesée
X+1	0Dh	Fin du message

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

Demande de passage en mode simple pesage du plateau 1 :
Description : Permet de demander au TEO de rentrer dans le mode " simple pesage " où il doit transmettre en continue le poids mesuré par le plateau 1.
Durant ce mode il peut recevoir certain ordre qui sont : passer en mode " attente d'ordre ", quitter mode " simple pesage", tare plateau 1, zéro plateau 1, demande de métrologie plateau 1.
Commande à émettre :

→ Taille : 3 octets

→ Format :

Octets :	0	1	2
Valeurs :	02h	'b' ou 'k'	0Dh

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).
Réponse : Dans ce mode le TEO envoi en continue la valeur du poids, le format de chaques messages envoyés, qu'il y est une erreur ou non, sont identiques aux formats des réponses en simple pesage.

Demande de métrologie plateau 1 :
Description : Permet de demander au TEO de transmettre des informations concernant le plateau 1, tel que son n° de série et la date de sa dernière vérification.

→ Taille : 3 octets

→ Format :

Octets :	0	1	2
Valeurs :	02h	's'	0Dh

xxh : Nombre hexadécimal
Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).
Réponse :

→ Taille : Variable (7, 12 ou 18 octets)

→ Format :

Octet	Valeur	Description
0	02h	Début de message
1	'1'	N° du plateau
2	FFh	Valeur fixe sans signification mais obligatoire
3	xxh	Type de l'erreur, '0' si aucune erreur
4	'1'	Valeur fixe sans signification mais obligatoire
5	' '	Espace
+6 octets	Nombre entier(ASCII)	N° de série, si le TEO la reçu
+6 octets	jjmmaa(ASCII)	Dernière date de vérification, si le TEO l'a reçu
X(= 6, 11 ou 17)	0Dh	Fin du message

xxh : Nombre hexadécimal

Une lettre(ou chiffre) entre ' correspond à la valeur de celle-ci en ASCII. (' ' correspond à la valeur de l'espace).

G Fiche bilan de stage