Institut Supérieur
d'**I**nformatique de
Modélisation et de
leurs **A**pplications

Complexe des Cézeaux
BP 10125
63173 Aubière Cedex

Institut de recherche
pour l'ingénierie de
l'agriculture et de
l'environnement

24 Avenue des Landais
63170 Aubière

Rapport de stage de 2ème année ISIMA
Informatique des systèmes embarqués

Conception et réalisation d'un capteur sans fil évolutif pour l'acquisition de données agri-environnementales

Présenté par : **Lionel Moënne-Loccoz**
Responsable ISIMA : **Kun-Mean Hou**
Responsables entreprise : **Gil De Sousa et Aurélien Jacquot**

Du 12 avril au 11 septembre 2010

Remerciements

Je remercie le personnel du Cemagref, pour l'accueil qu'ils m’ont réservé tout au long de mon stage. L'ambiance était chaleureuse et conviviale. J’ai été très bien intégré au sein de cette équipe. J’ai pu prendre confiance en moi.

Je tiens à remercier particulièrement Aurélien JACQUOT et Gil DE SOUSA pour m'avoir aidé dans de nombreuses circonstances. Ils m'ont permis d'avancer sur mon travail lorsque j'ai rencontré des difficultés. Ils m'ont soutenu et leur aide et leur expérience m’ont été précieuses.

Je remercie également Philippe RAMEAU, Arnaud ABONNAT et Jean-Pierre CHANET pour les suggestions et leur aide qu'ils m'ont apporté lors de la programmation.

Glossaire

Atmel :	Fabricant de composants à semi-conducteurs, crée en 1984 par Georges Perlegos. http://www.atmel.com/
CIRAD :	Centre de coopération Internationale en Recherche Agronomique pour le Développement est un établissement public à caractère industriel et commercial français crée en 1984.
CNRS :	Centre National de la Recherché Scientifique, organisme public français de recherche scientifique. C'est un établissement public à caractère scientifique et technologique.
CMS :	Composant Monté en Surface. La structure des composants a été modifiée pour obtenir de petites terminaisons métalliques, pour qu'ils puissent être brasés directement sur la carte.
I²C :	Inter Integrated Circuit Bus, développé par Philips. Ce bus de communication nécessite deux fils.
IAR :	IAR Embedded Workbench est un logiciel de programmation de nombreux types de processeurs. Le code compilé est adapté au processeur choisi par l'utilisateur.
ITK :	Entreprise spécialisée dans la réalisation d'outils d'aide à la décision basés sur modèle de simulation dans les domaines de l'agronomie et la biologie.
INRA :	Institut National de la Recherche Agronomique est un établissement public à caractère scientifique et technologique.
ISIMA :	Institut Supérieur de l'Informatique de Modélisation et de leurs Applications.
Langage C :	Langage de programmation impératif, de bas niveau inventé en 1970.
LIMOS :	Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes est une unité mixte de recherche. Il est rattaché aux deux universités clermontoises.
Microchip :	Fabricant de semi-conducteurs, fondé en 1989. http://www.microchip.com/
MPLAB :	Outil de développement et de compilation sur des processeurs du fabricant Microchip.
PCB :	Printed Circuit Board signifie circuit imprimé. Il s'agit généralement d'une plaque permettant de relier électriquement différents composants électronique entre eux.

RSCF :	Réseau de Capteur Sans Fil
SCK :	Fil du bus I²C, ce fil sert à donner la fréquence de la communication, c'est l'horloge du bus.
SDA :	Fil du bus I²C, ce fil permet de transmettre les données, c'est le fil de donnée.
SPI :	Serial Peripheral Interface. Bus de communication inventé par Motorola. Ce bus de communication utilise quatre fils.
Timer :	Un timer est un périphérique matériel permettant de mesurer des durées (généralement inclus dans les microcontrôleurs). Son rôle est de permettre la synchronisation des opérations que le microcontrôleur est chargé d'effectuer.
TSCF :	Technologies et Système d'information pour les agrosystèmes de Clermont Ferrand.
UART :	Universal Asynchronus Receiver Transmitter est un bus de communication par liaison série. Ce bus nécessite un port série avec qu'un seul fil transfert les données.
Wi-Fi :	Protocole de communication sans fil, il permet de relier plusieurs appareils informatiques les informations sont transmises par des ondes.
XBee :	Circuit développé par la société Maxstream, permettant la communication sans fil. Certaines puces possèdent une antenne intégrée.
ZigBee :	Protocole de communication de haut niveau permettant la communication de petites radios à consommation réduite.

Résumé

Depuis dix ans, les recherches sur la technologie du **réseau de capteurs sans fil** se succèdent sans pour autant proposer un système fiable répondant à toutes les contraintes subies. Un des intérêts de cette technologie est, de proposer une solution de supervision et de contrôle à distance **d'infrastructures agricoles** par exemple. Cela permet à un utilisateur de minimiser son temps de déplacement et de gestion pour se consacrer à une autre activité.

Mon travail a consisté à contribuer à l'avancée de cette technologie qui est encore dans le domaine de la recherche. Afin de permettre l'addition de tout type de cartes, qui constituent le nœud, j'ai choisi un **protocole de communication** que j'ai ensuite implémenté en **langage C**. Puis, j'ai réalisé une **carte de mesure** à l'aide du logiciel libre KiCad. Cette carte a pour but de réaliser une mesure rapide et elle doit pouvoir intégrer les nœuds conçus au Cemagref. Aujourd'hui, la carte de mesure effectue correctement son rôle, le protocole de communication fonctionne mais plusieurs cartes ne peuvent pas encore communiquer ensemble. Des corrections sont à apporter pour que cette communication fonctionne et le protocole peut être amélioré pour qu'il gère au mieux le processeur lorsqu'il se met au repos.

Mots clefs : réseau de capteurs sans fil, agriculture, protocole de communication, langage C, carte de mesure.

Abstract

During the last decade, many researches in **wireless sensor network** technology have been led; however, no reliable developed system satisfies all of the met constraints. This technology aims at the development of a more scalable system. For example, one of the interests of this technology is to offer a remote supervision and control solution for **agricultural infrastructures.** This allows user to minimize his time of movement and control in order to another activity.

My work contributed to the development of this technology, which is still a research area. To allow the addition of different kinds of electronic cards, forming a wireless sensor, I chose a **communication protocol**, which I implemented using the **C programming language**. Then, I realized an **acquisition card** with the free software KiCad. This card aims to realize a quick acquisition and must be able to be integrated with the sensors designed in Cemagref. Nowadays, the acquisition card performs correctly its function, the communication protocol operates but several cards can not yet communicate together. Corrections must be realised in order to achieve correct communication and, the protocol can be improved to control the processor when it enters sleep mode.

Key words: wireless sensor network, agriculture, communication protocol, C programming language, acquisition card.

Tables des matières

Table des figures

Introduction

Dans le cadre de ma deuxième année d'étude à l'Institut Supérieur d'Informatique, de Modélisation et de leurs Applications, j'ai effectué un stage au sein de l'institut de recherche en sciences et technologies pour l'environnement, nommé Cemagref. Ce stage a duré cinq mois entre avril et septembre 2010.

Dans le cadre du projet Disp'Eau, le Cemagref a pour mission d'établir un réseau de nœuds sans fil pour effectuer des mesures de l'environnement. Différentes versions de nœuds ont déjà été développées par le passé en collaboration avec le laboratoire de recherche LIMOS (Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes). Cependant, l'équipe de recherche désire en développer une nouvelle version permettant d'élargir le champ d'application.

Le nœud, anciennement utilisé, s'appuie sur la plate-forme de Réseaux de Capteurs Sans Fil (RCSF) LiveNode. Ce nœud dispose naturellement d'une connexion sans fil et on peut lui ajouter des appareils de mesure. Protégé dans un boîtier étanche, ce nœud a été utilisé pour effectuer des relevés de données sur l'humidité du sol. Ce nœud, bien qu'il fonctionne correctement, nécessite quelques améliorations. En outre, l'équipe du Cemagref désire un système plus évolutif, capable d'être modifié rapidement. De plus, les mesures prises par les appareils doivent être plus rapides. Il est donc nécessaire de concevoir une carte dédiée à la mesure de l'humidité du sol plus performante.

Je présenterai dans une première partie le contexte du stage. Dans la seconde partie, j'expliquerai l'architecture du système étudié. Ensuite, dans la troisième partie, je présenterai les différentes étapes dans la réalisation de la carte. Enfin, dans la quatrième et dernière partie, je détaillerai le fonctionnement de mes programmes et leurs validations.

I. PRESENTATION DU STAGE

I.1. Contexte du stage

I.1.1. Présentation du Cemagref

J'ai effectué mon stage au sein du Cemagref qui est un établissement public à caractère scientifique et technologique (EPST). Le Cemagref possède 9 centres sur la France et 2 antennes, une en Martinique et une à Strasbourg illustré dans la Figure 1.

Figure 1. Présentation des centres du Cemagref en France.

Le Cemagref possède 20 unités de recherche et 5 unités mixtes de recherche. Les unités de recherche sont créées au sein du Cemagref tandis que les unités mixtes (UMR) de recherche sont créées par la signature d'un contrat d'association d'un ou de plusieurs laboratoires ou organismes de recherche ou établissement d'enseignement supérieur. Il y a donc 1600 personnes qui travaillent au Cemagref réparties dans tous ces centres, parmi eux, il y a environ 200 doctorants.
J'ai été affecté à l'unité de recherche Technologies et Système d'information pour les agrosystèmes de Clermont-Ferrand. (TSCF)

I.1.2. L'unité de recherche TSCF

L'unité de recherche TSCF appartient au centre de Clermont-Ferrand (63) qui a la particularité d'être sur deux sites :

- Site d'Aubière, département du Puy-de-Dôme.
- Site de Montoldre, département de l'Allier.

Ce dernier site dispose d'une ferme expérimentale. Régulièrement les agents sont amenés à effectuer des essais ou des relevés sur ce site afin de mieux tester leurs solutions et appareils en milieu agricole.

L'unité de recherche TSCF est composée de trois équipes de recherche qui sont réparties sur les deux sites. L'équipe TEAM, Technologie Epandage Agroéquipements Mobilité, a pour but d'améliorer les techniques d'épandage pour un meilleur résultat sans détériorer l'environnement, et d'apporter des solutions pour améliorer les travaux des exploitants par l'intermédiaire de robots mobiles autonomes coopératifs. L'équipe CARAC'TERRE s'occupe de la caractérisation des matériaux et des différents milieux dans le domaine agricole. La troisième équipe que j'ai rejointe se nomme l'équipe COPAIN.

I.1.3. L'équipe COPAIN

L'équipe travaille sur les Systèmes d'Information Agro-environnementaux et Communicants. L'équipe est chargée de concevoir et réaliser des systèmes d'information dédiés à la gestion de données agro-environnementales. Ces systèmes sont alimentés par des données environnementales collectées et permettent de qualifier et d'avoir une meilleure compréhension de l'environnement et des pratiques agricoles. De plus, les échanges de données entre différents systèmes d'information sont également étudiés afin d'exploiter au mieux les informations recueillies. La démarche de la réalisation commence par l'analyse des besoins de l'utilisateur, la spécification des systèmes d'information, la modélisation et la conception d'une solution et enfin la gestion des sources de données.

L'équipe est composée de 12 personnes permanentes au sein du Cemagref, une personne post-doctorante et deux personnes doctorantes. L'équipe développe trois thématiques de recherche :

- Les réseaux de capteurs sans fil (RSCFs).
- Les entrepôts de données.
- Les systèmes d'information géographique (SIGs).

Mon stage s'intègre dans la thématique des RCSFs. Un nœud sans fil est un système composé d'une ou plusieurs cartes électroniques supportant des capteurs pour mesurer des grandeurs environnementales et les communiquer, vers une station de collecte de données, via une communication sans fil. Un RCSF est un ensemble de nœuds dont un ou plusieurs servent de passerelles entre un utilisateur (ou un autre réseau) et le réseau.

De plus, il y a une collaboration avec le laboratoire du LIMOS situé à l'ISIMA et l'équipe COPAIN. Ensemble ils ont mis au point une plateforme de nœud sans fil : le LiveNode. L'équipe développe de nouveaux procédés afin d'améliorer cette plateforme en augmentant ses capacités de mesure. Mon maître de stage, Gil DE SOUSA, travaillent au sein de cette équipe ainsi que mes encadrants Aurélien JACQUOT et Jean-Pierre CHANET qui supervise cette équipe.

I.2. Projet de stage

Le projet Disp'Eau a pour objectif de définir un nouvel outil logiciel innovant d'aide à la décision pour la viticulture (voir Figure 2). Ce projet a pour ambition de piloter une irrigation de précision pour la vigne ou de son mode de conduite afin d'optimiser la gestion de la

ressource en eau. Ce projet regroupe 10 acteurs, dont le Cemagref, la société ITK ainsi que le CIRAD et l'INRA. Pour répondre aux objectifs, nous proposons de promouvoir les réseaux de capteurs sans fil dans une chaîne de décision afin de réaliser des préconisations automatiques en fonction des relevés des capteurs.

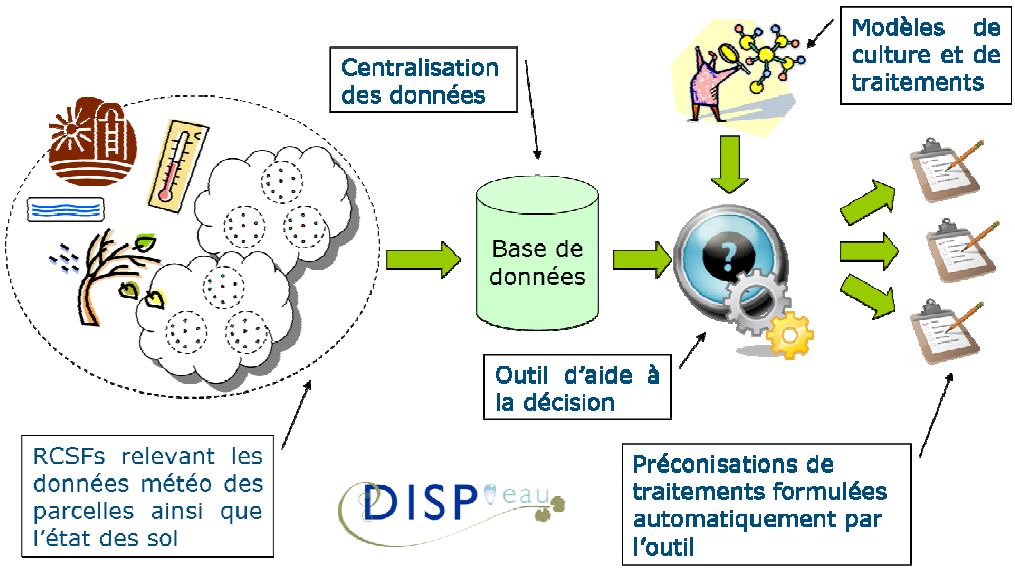


Figure 2. Présentation du projet Disp'Eau

Les RCSFs permettent de contrôler à distance l'irrigation d'une vigne. Les différents nœuds sont disséminés dans toute la vigne pour avoir un relevé détaillé. En complément, différentes données météorologiques peuvent être mesurées puis traiter afin d'indiquer la nécessité d'irriguer en fonction de l'humidité du sol. Le RCSF peut aussi permettre de comparer les relevés des différentes périodes de l'année sur plusieurs années, puisqu'une base de données stocke les informations utiles de ces relevés. Avec ces relevés, l'utilisateur peut analyser la qualité du sol de son exploitation grâce à un outil d'aide à la décision. L'outil fournit des conseils à l'utilisateur à partir des données recueillies et des modèles de culture et de traitements.

Par exemple, le sol n'absorbe pas l'eau uniformément sur toute une exploitation. Elle est alors découpée en parcelles modélisées dans l'outil afin de fournir un rapport sur l'état du sol. Ainsi, l'utilisateur peut donc irriguer uniquement les parcelles qui en ont besoin et économiser de l'eau sur les autres. Chacun des nœuds communique avec le nœud qui a pour rôle de passerelle. En fonction de la taille du réseau, ce nœud conserve son rôle de mesure ou sera uniquement dédié à cette tâche. La passerelle communique avec une station centrale de collecte pour alimenter une base de données. Pour fonctionner, les nœuds disposent de batteries qui doivent leur permettre d'avoir une autonomie de plusieurs mois sans intervention. Les énergies renouvelables comme le solaire peuvent aider à améliorer la durée de vie de chaque nœud du réseau.

Pour ce projet, l'humidité du sol est mesurée par l'intermédiaire de trois capteurs placés à trois profondeurs différentes afin de connaître la pénétration de l'eau dans le sol. Le premier capteur est placé à environ 30 cm du sol. Celui-ci détecte le début de l'arrivée de l'eau aux racines de la plante. Le deuxième est situé à 60 cm du sol et permet de savoir si l'irrigation a été suffisante pour s'infiltrer suffisamment dans le sol. Le troisième est à 1 m du sol et permet de relever la fin de l'irrigation et éventuellement son efficacité.

I.3. Objectifs du stage et planning

Les objectifs de mon stage sont les suivants :
La première partie du stage a pour but d'aider à améliorer les nœuds sans fil à partir de leur architecture. Plusieurs cartes composent le nœud et pour que celui-ci fonctionne, il est nécessaire que ces cartes puissent communiquer entre elles sans que l'efficacité du nœud soit perturbée. Les cartes doivent respecter un protocole spécifique et générique pour qu'il puisse être accepté par les différents processeurs qui composent chacune des cartes.

La deuxième partie du stage a pour but de concevoir et réaliser une carte dédiée à la mesure de l'humidité du sol. Il faudra rechercher un processeur pour que la carte puisse fonctionner avec les différentes contraintes de qualités et d'économies imposées par le cahier des charges du projet. A ces contraintes, il faut ajouter celles de conception qui est réalisée à l'aide d'un logiciel libre. La réalisation est effectuée, dans un premier temps, au sein du Cemagref pour permettre la phase de test. Cette phase permet de valider ou corriger la première partie du stage. De plus, elle permet d'apporter des modifications à la carte notamment la théorie utilisée pour le relevé des mesures d'humidités.

Le planning de mon stage est le suivant :
Mon stage a commencé le 12 avril 2010. Durant le mois d'avril, j'ai dû prendre en main les différents éléments utilisés comme le LiveNode et me documenter sur les communications possibles entre cartes. Lors des mois de mai et juin, j'ai proposé et développé un protocole de communication. Pour le début du mois de juin, j'ai préparé une présentation de mon stage avec ses objectifs et le protocole de communication proposé. Lors des mois de juin et juillet, j'ai conçu et réalisé la carte de mesure. Aux mois de juillet et août, j'ai validé son fonctionnement, testé la communication entre deux cartes et, conçu les méthodes permettant la lecture des chaînes de mesure.

II. SOLUTIONS ENVISAGEES

II.1. Contraintes liées à l'application

Les nœuds sans fil doivent, en premier lieu, avoir une autonomie fonctionnelle, ce qui prend en considération les conditions de déploiement. Ils doivent pouvoir être mis en place rapidement et facilement. De plus, ils doivent pouvoir fonctionner dans des conditions climatiques et environnementales parfois difficiles. Enfin, les nœuds sans fil doivent être réutilisables. Les nœuds doivent pouvoir changer de réseau facilement. S'il nous manque une mesure spécifique sur un nœud, il faut pouvoir adapter le nœud pour accueillir cette mesure.

A cela, le nœud se doit d'avoir une autonomie énergétique. Il doit avoir une durée de fonctionnement de plusieurs mois. Le nœud est une association de cartes, donc les exigences de durée de fonctionnement qui sont valables pour une carte, le sont pour le nœud dans son ensemble. La carte, que je dois réaliser, doit permettre de respecter la durée de fonctionnement. Au niveau de la consommation énergétique, les différents éléments qui entrent en jeu sont les suivants :

- Le processeur.
- Les composants sur la carte.
- Le programme implémenté.
- La communication entre le nœud et la station de collecte.

Le processeur est le cœur de la carte. Il permet de relier l'ensemble des appareils de mesure sur cette carte aux autres cartes. Le processeur doit communiquer avec une carte et doit analyser les données recueillies, c'est-à-dire effectuer les calculs sur les informations de la chaîne de mesure.

Les composants d'une carte sont de différents types. Il y a les composants de type traversant et ceux de types montés en surface (CMS). Les CMS sont petits et légers, mais plus compliqués à souder et dissipent moins bien la chaleur du fait de leur taille. Tandis que les composants traversant, appelés ainsi parce qu'ils possèdent des pattes qui traversent la carte pour être soudés de l'autre côté, ont une bonne fixation et ne sont pas limités en taille. Cependant, ils consomment plus d'énergie que les composants CMS.

Le programme doit faire en sorte que le processeur consomme le moins possible. Il doit donc s'exécuter assez rapidement et gérer le mode repos du processeur (gestion du mode veille). Il y a plusieurs options disponibles en fonction des caractéristiques du processeur que l'on a choisi. Le programme doit utiliser au mieux chacune de celles-ci pour améliorer le fonctionnement du nœud. Par exemple, l'utilisation du mode veille peut consister à ne garder que quelques éléments matériels actifs, pour améliorer considérablement la durée de vie du nœud.

Par ailleurs, l'encombrement du nœud doit être minimal. Afin de limiter les coûts, il est courant de choisir des boîtiers standards du commerce dans lequel on positionne au mieux la carte. De plus, d'autres éléments doivent prendre place à l'intérieur du boîtier comme les piles, les régulateurs de tension pour un panneau solaire. Il faut faire correspondre, d'un côté, les dimensions de la carte et de ces autres éléments et, de l'autre, celles du boîtier pour obtenir le

meilleur dispositif possible. La légèreté et la facilité de transport sont des contraintes moindres mais à prendre en compte pour minimiser les risques de pannes.

De plus, par définition, un nœud sans fil emploie un médium d'accès sans fil pour diffuser ses données. Les échanges sont sujets à perturbations extérieures ainsi qu'aux obstacles.

Enfin, la dernière contrainte, est la communication entre chaque composant du nœud. Pour que le nœud puisse évoluer, il doit pouvoir accepter différents éléments. Ces éléments doivent pouvoir communiquer entre eux. Dans le cadre de mon stage, cet aspect a été traité par l'implémentation d'un protocole de communication.

II.2. Architectures possibles

Différents types d'architecture de nœud existent :
- Carte spécifique à une application.
- Carte mère avec des extensions enfichables.
- Carte mère avec des cartes filles empilables.

La première solution consiste à réaliser une carte spécifique. Dans celle-ci, il faut alors avoir un module de communication sans fil, le module d'alimentation et le module de mesure (voir Figure 3). L'avantage de cette solution est qu'elle est rapide à réaliser. De plus, chaque nœud est conçu au plus près de l'application, ce qui permet d'avoir de meilleures performances énergétiques et une meilleure utilisation des fonctionnalités de la carte par le processeur.
En revanche, ce dispositif sera peu ou pas évolutif alors que, dans notre cas, il nous faut un nœud réutilisable, pour des applications différentes. La contrainte d'autonomie de déploiement peut difficilement être respectée puisque la carte a un nombre limité d'appareils de mesures. Le nœud ne pourra être réutilisé que pour l'application ou une autre assez proche pour laquelle il a été conçu. En outre, cette solution nécessite la conception d'une carte complète. De plus, la possibilité d'associer deux cartes, correspondant chacune à un nœud sans fil, pour en former un plus évolué serait une fonctionnalité assez complexe à mettre en œuvre. Enfin, la contrainte d'encombrement fait qu'il n'est pas possible d'avoir de nombreux appareils de mesure contrairement aux autres solutions.

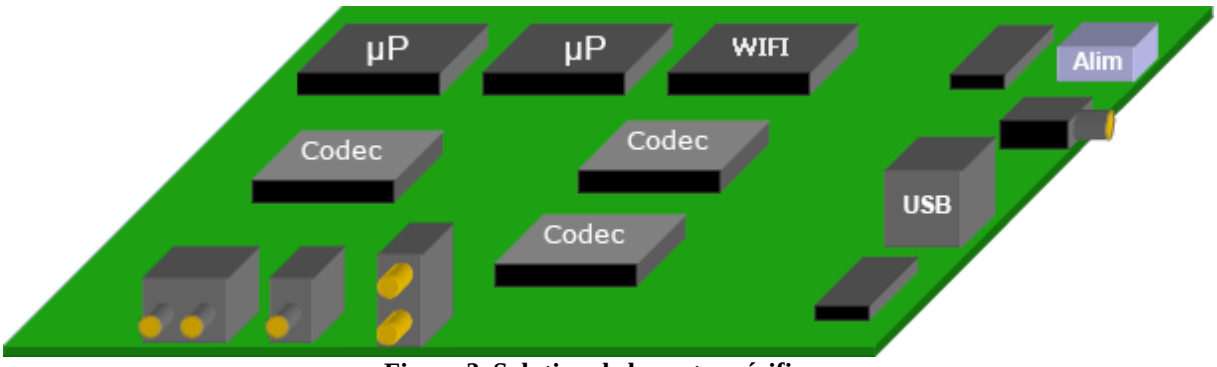


Figure 3. Solution de la carte spécifique

Une autre solution consiste à concevoir une carte mère avec des extensions possibles (voir Figure 4). La carte mère gère les éléments nécessaires à tous les nœuds comme les processeurs et l'alimentation électrique. On peut également ajouter quelques appareils de mesures suivant leurs tailles. Les extensions gèrent des modules spécifiques comme la

communication sans fil. Pour chaque extension, il faut prévoir un module de décodage ou de transmission sur la carte mère pour qu'elle puisse communiquer avec elle.
L'avantage de ce système est la possibilité de modifier rapidement, pour une application donnée, un nœud sans fil du réseau grâce à une extension. La contrainte d'autonomie de déploiement est aussi respectée.
L'inconvénient majeur est la limitation du nombre d'extensions possibles supportées par la carte mère. Il est donc nécessaire ici d'effectuer une étude préalable sur le nombre d'extensions possibles par nœud avant de concevoir la carte mère. De plus, ce type de positionnement des extensions, implique un encombrement plus élevé.

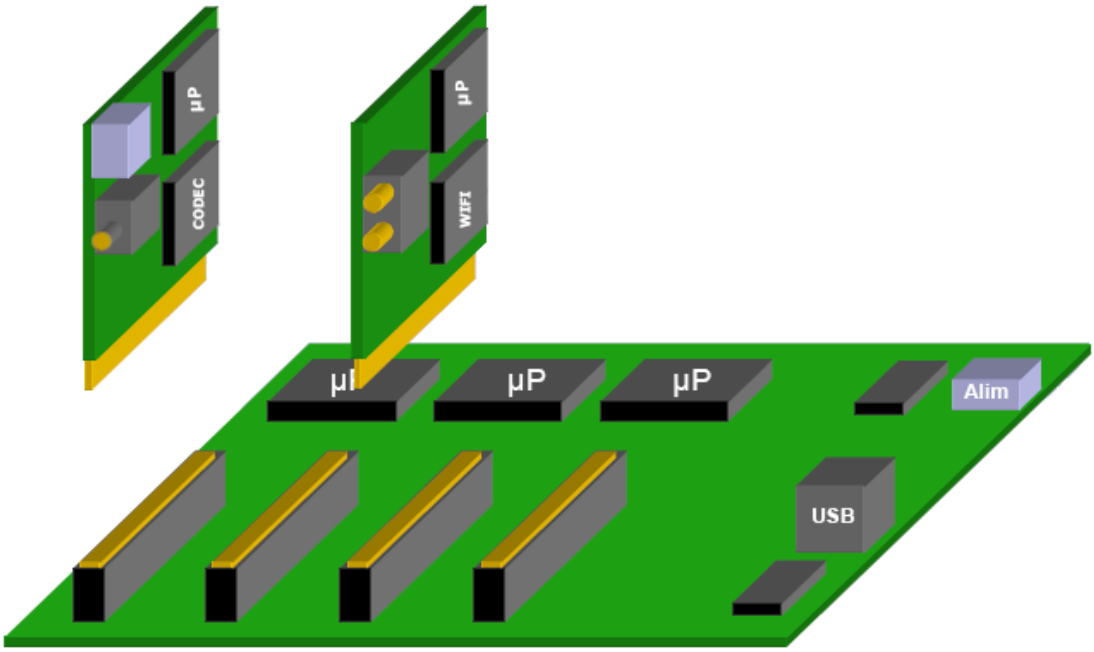


Figure 4. Solution carte mère avec extensions.

Enfin, la dernière solution utilise l'aspect de carte mère avec extension empilable. Cependant, afin de ne pas limiter le nombre d'extensions, il paraît judicieux de pouvoir créer un empilement de carte. Les cartes empilées sont nommées cartes filles. Cette idée, d'ajouter des cartes dédiées à une application, a permis d'imaginer et de concevoir une nouvelle architecture : une carte mère à laquelle on peut ajouter ou enlever une ou plusieurs cartes filles (voir Figure 5). Chacune des cartes filles possèdent une application spécifique et il est possible d'avoir plusieurs cartes identiques. La carte mère possède les éléments nécessaires au bon fonctionnement du nœud sans fil. L'avantage de cette solution est que le nœud est entièrement modifiable donc constamment adapté à tout type d'application, sous réserve du nombre de cartes que l'on peut empiler. Les inconvénients de cette solution sont, tout d'abord, l'encombrement en hauteur des différentes cartes filles. Ensuite, le développement de ce type de nœud nécessite la réalisation de deux cartes minimum puisque la carte mère ne peut pas accueillir des appareils de mesure trop imposants. De plus, la communication entre toutes les cartes est plus difficile à réaliser puisque les cartes filles partagent le même connecteur pour communiquer.

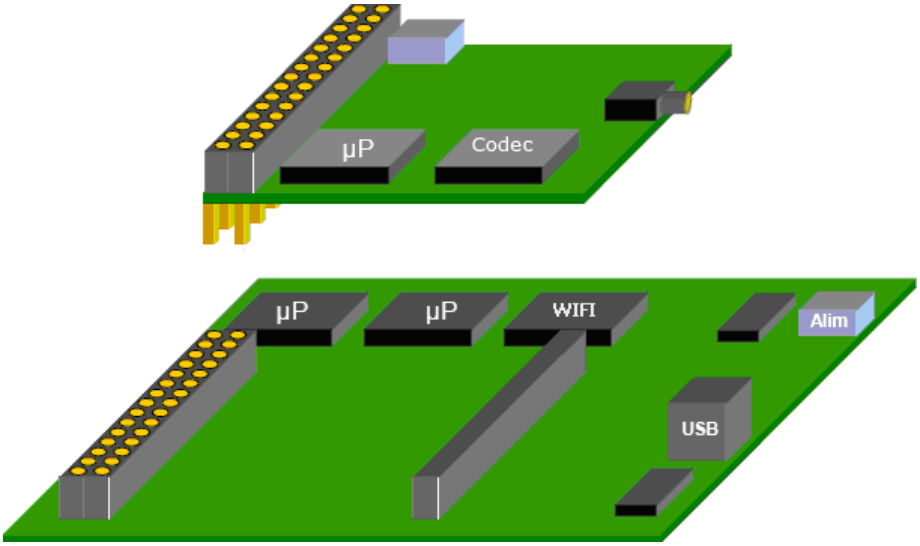


Figure 5. Solution carte mère et carte fille.

Les trois solutions présentées ci-dessus dispose d'un certain nombre d'avantages et d'inconvénients répertoriés dans la Figure 6.

Solution	Avantages	Inconvénients
Carte spécifique	-Vitesse de réalisation -Encombrement adapté à l'application -Communication directe avec les appareils de mesure	-Nombre d'applications très limité -Nœud non modifiable -Choix de la taille d'un boîtier standard
Carte mère avec des extensions	-Nœud modifiable	-Réalisation d'au moins deux cartes pour une application -Nombre d'extensions limité -Communication avec les extensions à définir
Carte mère avec cartes filles	-Nœud très évolutif	-Compatibilité carte mère – carte fille -Communication entre la carte mère et les cartes filles de différents types à définir

Figure 6. Tableau récapitulatif des solutions.

Au vue de cette étude, la troisième solution est la plus adaptée à notre problématique.

Celle-ci sera basée sur le nœud réalisé au LIMOS : le LiveNode. Ce nœud sans fil est une plate-forme de prototypage qui utilise une communication sans fil soit IEEE 802.15.4 ZigBee soit IEEE 802.11b Wi-Fi. Ce nœud a permis de réaliser un premier réseau de capteurs sans fil (RCSF).
Dans la Figure 7, nous pouvons voir une version du LiveNode développée au LIMOS, en vue de face avec le connecteur pour programmer le processeur, qui est situé juste en-dessous, ainsi qu'en vue de dos avec une puce XBee pour une communication sans fil ZigBee. Le LiveNode est composé d'un processeur Atmel AT91SAM7S256 32bits, dont la fréquence de fonctionnement peut varier de 500Hz à 50MHz. Il embarque 256 ko de mémoire flash ainsi que 64 ko de mémoire RAM. La consommation en veille du nœud est de 2mA et, en fonctionnement, celle-ci varie de 55mA à 215mA lors d'une émission de données. Le médium

de communication sans fil est basé sur le protocole ZigBee. Ce protocole autorise des portées de communication allant de 300 à 1600 mètres pour un débit maximal de 256 ko/s.
Le LiveNode possède un capteur de température et deux borniers de huit fils chacun. A l'aide de ces borniers, plusieurs cartes peuvent être branchées au LiveNode afin d'effectuer des tâches plus complexes. Une fois le LiveNode testé et validé, il a servi de base pour tester certains dispositifs ou cartes de mesure. Le LiveNode illustre le concept proposé par la troisième solution c'est-à-dire une carte mère avec, dans ce cas, une carte fille de mesure qui a pour but la mesure de l'humidité du sol à l'aide de trois sondes tensiométriques Watermark.

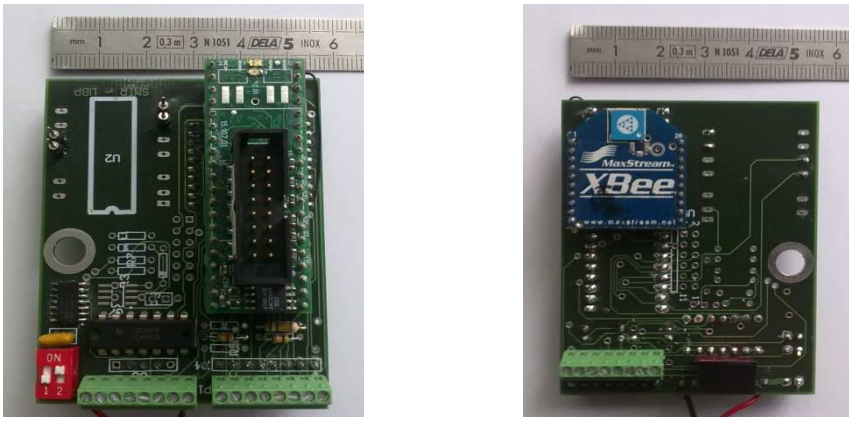


Figure 7. Le LiveNode.

Comme nous l'avons déjà indiqué, la nouvelle carte mère a été conçue à partir du nœud LiveNode. Différentes modifications ont cependant été apportées. Tout d'abord, un connecteur a été ajouté pour pouvoir empiler plusieurs cartes. Celui-ci possède 40pins, en prévision de la connexion de différents appareils de mesure. Il est peu encombrant en hauteur.
La carte mère, qui est composée essentiellement des éléments du LiveNode, a été élargie pour pouvoir accepter l'ajout de différents types de cartes appelées carte fille. Le nœud a donc la possibilité d'être modifiable et est très évolutif.
La carte mère possède les éléments indispensables pour s'intégrer et fonctionner comme un nœud sans fil dans un RCSF avec la présence, par exemple, d'une alimentation, d'un médium de communication sans fil, et d'un processeur supervisant le fonctionnement général du nœud. Les cartes filles s'ajoutent en s'empilant sur le connecteur (voir Figure 5). Chaque carte fille peut proposer des fonctionnalités différentes. Comme l'on peut ajouter ou retirer des cartes filles à la demande, le nœud est réutilisable pour différentes applications.

En revanche la communication entre toutes ces cartes est nettement plus difficile puisque chaque carte peut avoir un processeur de type et de marques différentes. Il est alors nécessaire de concevoir un protocole de communication standard permettant aux différents éléments de cette architecture matérielle de communiquer. La définition de ce protocole est l'objet du paragraphe suivant.

II.3. Communication entre les différentes cartes

Le nœud sans fil est composé d'une carte mère et d'une ou plusieurs cartes filles (troisième solution) qui possèdent les différents appareils de mesure. Ces appareils de mesure génèrent des données numériques qui sont traitées par la carte fille ou directement par la carte mère. Pour que ces différents éléments puissent communiquer entre eux, il est nécessaire d'avoir un bus de communication avec un protocole standard et connu.

Les caractéristiques du protocole de communication recherché sont les suivantes :

- L'émetteur doit s'adresser directement à chacune des cartes. Ces cartes disposent d'un identifiant unique que l'émetteur doit connaître.
- Le protocole doit permettre à l'émetteur de déterminer si le récepteur a bien reçu le message ou s'il faut le réémettre. Il s'agit de la gestion de l'acquittement.
- Il faut pouvoir adresser un nombre suffisant de cartes que l'on pourrait ajouter physiquement sur la carte mère.

Le but de la communication est que la carte mère puisse obtenir rapidement une mesure pour éventuellement la traiter, la stocker ou l'envoyer. Différents protocoles existants (SPI, I²C, UART) peuvent répondre à ce besoin.

II.3.1. Le protocole SPI (Serial Peripheral Interface)

Le protocole SPI, conçu par Motorola, permet d'avoir un très bon débit de communication (10Mbits/s) (voir Figure 8). Pour adresser plusieurs esclaves, il est nécessaire d'avoir une commande par composant esclave. De plus, la gestion du port SPI sur les processeurs se limite généralement à 2 voire 3 esclaves. Par ailleurs, il n'y a pas de gestion matérielle d'acquittement ce qui signifie que l'émetteur ne sait pas si le récepteur a reçu un message.

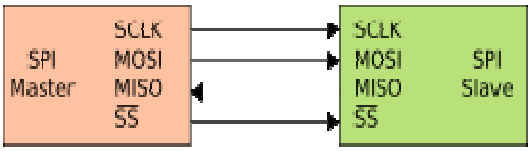


Figure 8. Liaison SPI maître - esclave.

II.3.2. Le protocole UART (Universal Asynchronous Receiver Transmitter)

Le protocole UART permet d'avoir un débit de communication important (3Mbits/s) (voir Figure 9). Cependant, la communication entre deux cartes se fait en mode « peer-to-peer » (point à point). En effet, la communication fonctionne uniquement de carte à carte. Il est donc impossible à la carte mère de s'adresser directement à plusieurs cartes filles, ce qui a pour effet de compliquer les algorithmes pour obtenir rapidement une information. Ce protocole n'est donc pas adapté à notre architecture et ne sera pas retenu pour la communication interne.

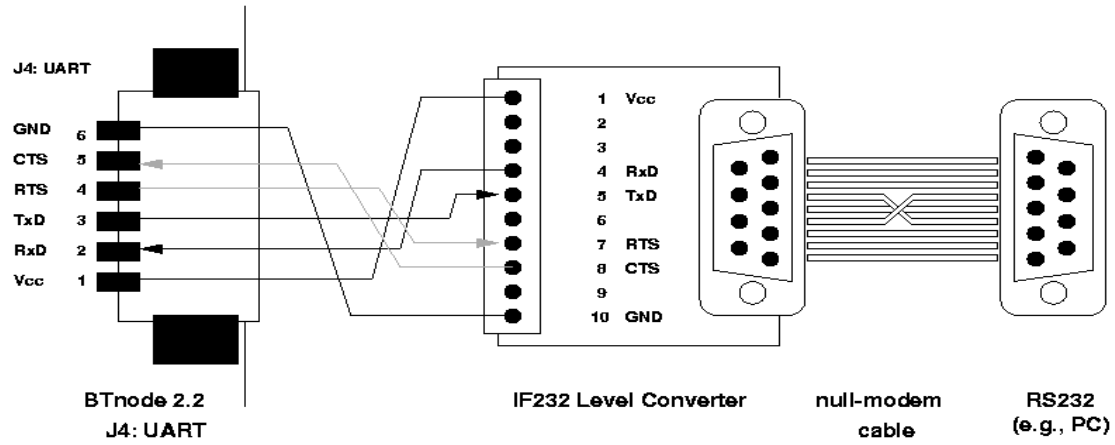


Figure 9. Liaison UART.

II.3.3. Le protocole I²C (Inter Integrated Circuit Bus)

Le protocole I²C conçu par Philips possède un débit inférieur aux protocoles précédents (400Kbits/s) (voir Figure 10). Cependant, il permet de s'adresser, de façon directe, à de nombreuses cartes. L'adresse peut être codée sur 7 ou 10 bits, ce qui laisse une grande possibilité d'ajout de cartes filles (127 adresses ou 1024 adresses avec les adresses réservées). Ce protocole est basé sur le principe de « maître-esclave ». Seul le maître peut initier la communication en s'adressant à une carte fille via son adresse. De plus, il n'y a que deux types de trames possibles : une trame de lecture et une autre d'écriture. Sur de courtes distances, les probabilités d'erreur de communication sont minimes ce qui constitue un atout non négligeable. Le maître et l'esclave partagent une horloge commune générée par le premier. Elle est aussi utilisée pour l'acquittement des trames et la synchronisation entre le maître et l'esclave.

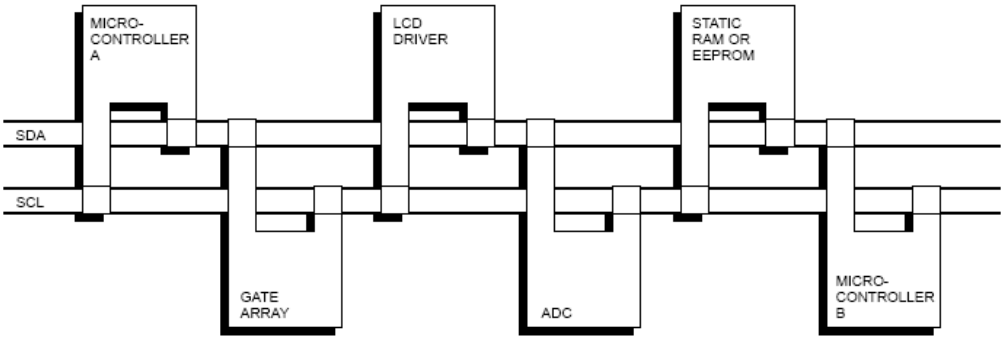


Figure 10. Liaison I²C.

II.4. Choix du processeur

Pour réaliser ma carte, j'ai opté pour un microcontrôleur de la gamme PIC de Microchip. Ces microcontrôleurs ont des caractéristiques intéressantes tant sur les interfaces disponibles que sur leurs performances énergétiques. J'ai donc effectué une recherche pour comparer les différents processeurs disponibles avec comme critères de sélection : la fréquence

d'utilisation, les mémoires embarquées, les interfaces de communication, ainsi que la consommation et le type du processeur.

La fréquence d'utilisation n'a pas été un élément de sélection majeure puisque la gamme de produits sélectionnés fonctionne à des fréquences largement suffisantes pour la plupart des opérations qui seront supportées par la carte fille. La fréquence que nous souhaitons est de 4MHz alors que les processeurs fonctionnent de 32kHz à 32MHz. Ce choix a été effectué en prenant en compte la vitesse de calcul ainsi que la consommation du processeur à cette vitesse.

Par ailleurs, la consommation, notamment la possibilité d'avoir un mode de veille, est un des critères importants pour le choix du processeur. Ce mode de fonctionnement permet au processeur de diminuer considérablement sa consommation lorsqu'il n'est pas actif. Dans cette configuration, seuls quelques éléments fonctionnent avant que les autres ne reçoivent l'ordre de se réveiller et de passer en mode actif.

Un autre critère de sélection est les interfaces embarquées. Le processeur doit comporter au minimum un bus I²C, SPI et UART. Pour certains processeurs, il n'est pas possible d'avoir à la fois le bus I²C et le bus SPI puisqu'ils utilisent les mêmes pins. Nous souhaitons avoir ces interfaces pour balayer la grande majorité des capteurs que l'on pourrait utiliser.

Nous avons ensuite décidé du type de processeur. Les processeurs de type 8 bits ont été jugées non adaptés pour notre application car leur mémoire embarquée n'est pas suffisante. Les processeurs de type 32 bits consomment trop d'énergie notamment à cause de leur mémoire. Les processeurs qui ont donc été retenus sont de type 16 bits.

Les processeurs PIC de la famille 24F sont les seuls de types 16 bits qui répondent aux critères précédents. Leur mémoire est plus grande que celle des processeurs de types 8 bits. Leur consommation en revanche est légèrement plus élevée mais acceptable à condition d'avoir une vitesse assez basse et d'utiliser les modes de veille.

Par conséquent, j'ai donc choisi un processeur de la famille 24FJ. Le processeur choisi est le PIC24FJ32GA002 avec 32ko de mémoire flash et 8Ko de mémoire RAM (voir Figure 11). Ce processeur possède 21 pins pilotables, ce qui est suffisant pour notre application. La mémoire permet de programmer correctement le processeur et de pouvoir stocker les différentes informations qui pourraient s'avérer être nécessaires. L'une des caractéristiques intéressantes de ces processeurs est la possibilité de remapper certains périphériques sur différentes pins du microcontrôleur. Il est alors possible de configurer les différentes interfaces et de les disposer sur les pins que l'on souhaite. Ceci permet de faciliter la réalisation de la carte.

PIC24FJ Device	Pins	Program Memory (bytes)	SRAM (bytes)	Remappable Peripherals						I^2C™	10-Bit A/D (ch)	Comparators	PMP/PSP	JTAG
				Remappable Pins	Timers 16-Bit	Capture Input	Compare/ PWM Output	UART w/ IrDA®	SPI					
32GA002	28	32K	8K	16	5	5	5	2	2	2	10	2	Y	Y

Figure 11. Caractéristiques du processeur PIC24FJ32GA002.

II.5. Description de la solution choisie

II.5.1. Protocole choisi

Nous avons donc choisi le protocole I²C, car ce protocole ne nécessite, entre autres, que deux fils pour communiquer. L'un des fils sert à transmettre les données (SDA) et l'autre fil sert d'horloge pour la communication (SCK). Les possibilités d'adressage sont grandes et suffisantes pour notre application. Malgré un débit relativement faible, ce protocole est suffisant pour réaliser la communication interne. De plus, le protocole permet la gestion de l'acquittement, afin de nous assurer que les informations transmises arrivent à leur destinataire. Enfin, il est possible de réduire la consommation à l'aide du mode veille du processeur. En effet, ce mode veille peut être désactivé par le bus I²C. Ainsi, une fois la communication terminée, le processeur peut repasser de nouveau en mode veille.
Son fonctionnement est basé sur le principe de « maître-esclave ». Ce principe empêche tout esclave de démarrer une conversation sans l'aval du maître. Un esclave ne peut donc donner directement une information à un autre esclave ou au maître tant qu'il n'en a pas reçu l'ordre ou l'autorisation. La communication avec un autre esclave passe nécessairement par le maître. Le maître contrôle donc entièrement les échanges. Un échange se déroule de la manière suivante :

- Le maître envoie un signal de départ (START) qui permet de réserver le bus.
- Le maître envoie l'adresse « matérielle » de l'esclave avec lequel il souhaite communiquer, suivit d'un bit qui indique le type de la conversation c'est-à-dire soit la lecture, soit l'écriture.
    - Si le type de la conversation est la lecture, le reste de la trame qui suit est la réception des informations envoyées par l'esclave et ceci jusqu'à ce que le maître termine la conversation.
    - Si le type de la conversation est l'écriture, le maître envoie des informations à l'esclave.
- Une fois les informations envoyées, le maître met un terme à la communication par le biais d'un signal de fin (STOP).

II.5.2. Architecture de la solution

Sur chacune des cartes filles, il y a des appareils de mesure nommés capteurs. Le capteur est un élément qui permet de traduire électriquement une grandeur physique ou matérielle en information digitale ou analogique.

Les cartes filles relient les capteurs à la carte mère. Leur but est de décoder et de stocker les données pour décharger le processeur de la carte mère. Le nœud sans fil est donc constitué de deux ensembles (carte mère et carte fille) et d'un bus de communication pour relier les deux, ce qui donne le schéma synoptique présenté par la Figure 12.

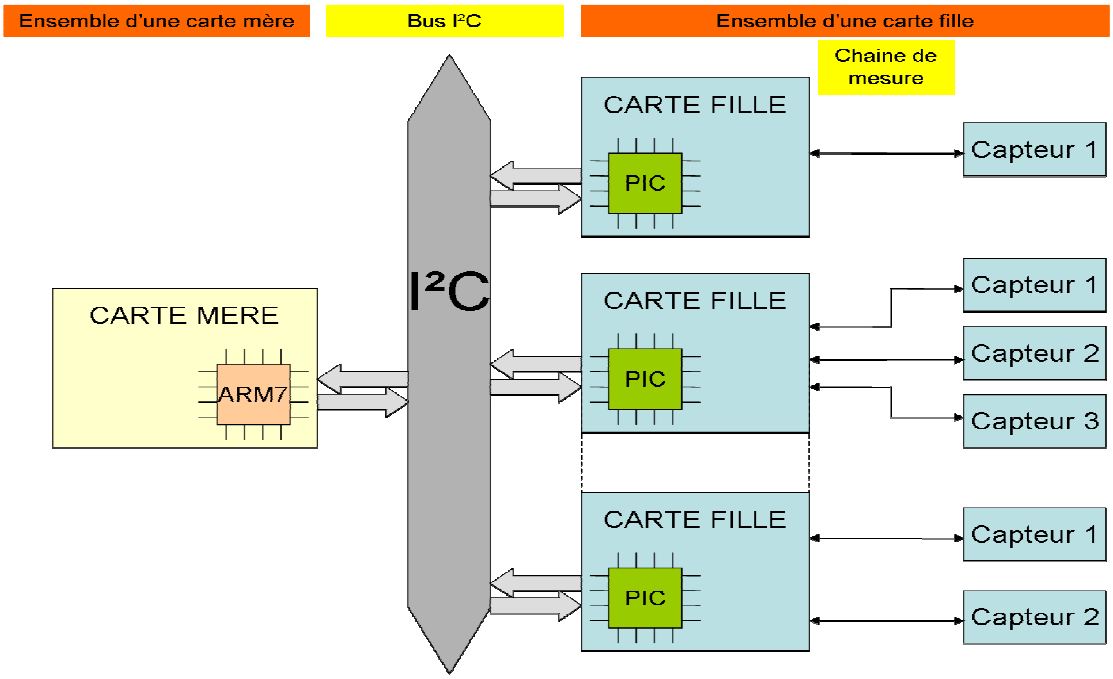


Figure 12. Schéma synoptique de l'architecture de la solution proposée.

II.5.3. Fonctionnement de la chaîne de mesure

La chaîne de mesure retenue pour relever les informations se base sur la charge et la décharge d'un circuit RC. Le capteur interfacé est un capteur d'humidité du sol ou sonde tensiométrique de type Watermark (voir Figure 13) qui voit sa résistance varier en fonction de la teneur en eau du sol. La chaîne de mesure génère une fréquence fixe qui charge un condensateur. Ensuite, durant la phase de décharge, le condensateur envoie son énergie dans le capteur ce qui a pour effet de générer un signal de fréquence variable selon la teneur en eau. Pour limiter la plage de fréquences, des composants annexes ont été ajoutés. Les fréquences mesurées varient de 50Hz lorsque le capteur est sec, à 12.5kHz lorsque le capteur est trempé. Après avoir mesuré la fréquence issue de la chaîne de mesure, le processeur applique des algorithmes pour, tout d'abord, trouver la résistance du capteur et, ensuite, la valeur de l'humidité du sol. Pour cela, à partir de la résistance, on applique la formule suivante :

Humidité = (3.213 * résistance + 4.093) / (1- 0.009733 * résistance – 0.01205 * température)

Cette formule est fonction de la température du sol. Dans un premier temps, la température employée est fixée à 20°C. Avec une telle chaîne de mesure, il est possible d'avoir la mesure en quelques millisecondes ce qui constitue un avantage non négligeable pour le bilan énergétique de la carte fille.

Figure 13. Capteur Watermark.

II.5.4. Aperçu de la solution

L'architecture globale de la solution est présentée dans la Figure 14. On y retrouve la carte mère équipée d'une carte fille de mesure de l'humidité du sol. La carte mère dispose du processeur AT91SAM7S256 qui sera défini comme le maître du nœud et aura le rôle de gérer les communications avec la station de collecte. La carte fille est conçue pour relever trois capteurs ou sondes tensiométriques Watermark ainsi que la température interne du nœud. Concernant l'alimentation, la carte mère dispose d'un régulateur ajustant la tension de 3 piles AA standard à 3.3V. La carte fille utilise cette source d'énergie pour fonctionner.

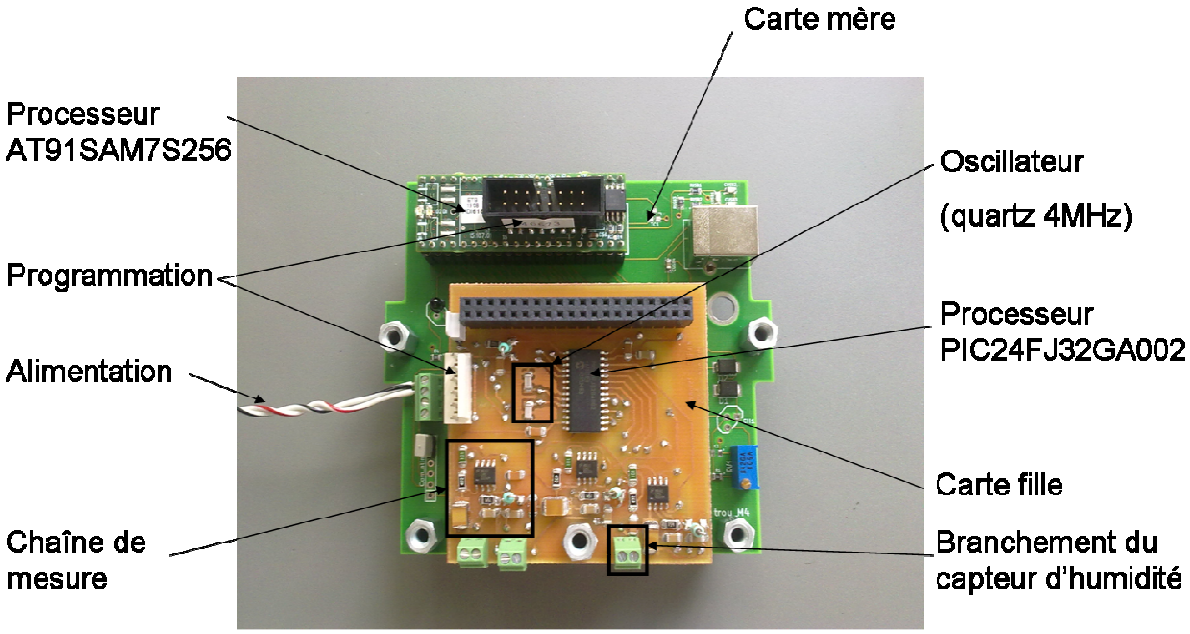


Figure 14. La carte mère et la carte fille ensemble.

III. REALISATION DE LA CARTE

III.1. Conception de la carte

III.1.1. Les contraintes

- *Contrainte d'encombrement*

La carte que je dois concevoir est une carte fille qui prendra place sur la carte mère déjà développée. La taille de cette carte est directement liée au choix du boîtier employé (130x130mm) et aux différents éléments à y mettre (batterie, antenne, etc.). La taille de ma carte devra avoir une dimension minimale de 60x60mm pour pouvoir être positionnée sur le connecteur prévu et être fixé au boîtier. Du fait, de la conception de la carte mère, elle peut avoir une largeur légèrement supérieure (jusqu'à 80x60mm). La longueur est, par contre, fixe afin qu'elle puisse toujours passer dans le boîtier sans introduire de nouvelle contrainte pour le passage des câbles.

- *Contrainte de placement des composants*

La carte fille a donc une taille limitée. Sur cette surface, on a des composants qui doivent avoir une position fixe. On distingue deux éléments dont le placement est important : le connecteur inter-carte et le trou de fixation. Le connecteur, qui sert à empiler les cartes et qui intègre le bus de communication, occupe une surface de 50x5mm. Le trou de fixation d'un diamètre de 4mm, avec une zone d'isolement autour, occupe la surface d'un disque de 6mm. Ce trou permet de placer une entretoise afin de fixer la carte fille avec l'ensemble des autres cartes filles ou avec la carte mère. Les positions relatives de ces deux éléments sont fixées afin de correspondre à celles de la carte mère. Le trou est positionné au milieu du connecteur à 5mm du bord bas de la carte fille, et le connecteur est situé à 5mm du bord haut de la carte fille. De plus, pour programmer le microcontrôleur embarqué, je dois prendre en compte la taille du programmateur (connecteur six broches) et lui trouver une place facile d'accès et peu contraignante pour le reste de la carte.

- *Contrainte de réalisation*

La carte est fabriquée, dans un premier temps, au Cemagref ce qui impose des techniques de conception en conséquence. La taille des pistes doit être de 0.6mm minimum. Les via auront un diamètre de 1.7mm. Pour gérer les trois capteurs d'humidité, chacun d'entre eux disposera d'une chaîne de mesure dédiée pour exploiter les données recueillies. Le nombre de composant est alors multiplié par 3 ce qui a pour effet de compliquer le placement de l'ensemble dans une surface restreinte.

III.1.2. Logiciels de conception utilisés

Le logiciel KiCad est une suite open source pour la réalisation de schéma électronique et de circuit imprimé. Ce logiciel possède quatre applications : EESchema, CVpcb, PCBnew et GerbView.

- GerbView 

L'application GerbView sert à visualiser une carte en 3D. Cette application n'a pas été utilisée lors de la conception cependant elle fait partie du logiciel.

- EESchema 

Cette application aide à la réalisation des schémas électroniques (voir Figure 15), ce qui permet de mettre en relation les composants pour faciliter la génération des pistes. On peut créer de nouveaux composants que l'on insère dans notre schéma. De plus, lorsque l'on crée un projet, les schémas qui font partie du projet sont automatiquement liés. En effet, j'ai réalisé, dans deux fichiers séparés, d'une part le connecteur et le processeur avec les éléments nécessaires à son bon fonctionnement et, d'autre part, la chaîne de mesure. Une fois le schéma fini, il faut générer une liste d'empreintes (netlist) avec le logiciel CVpcb.

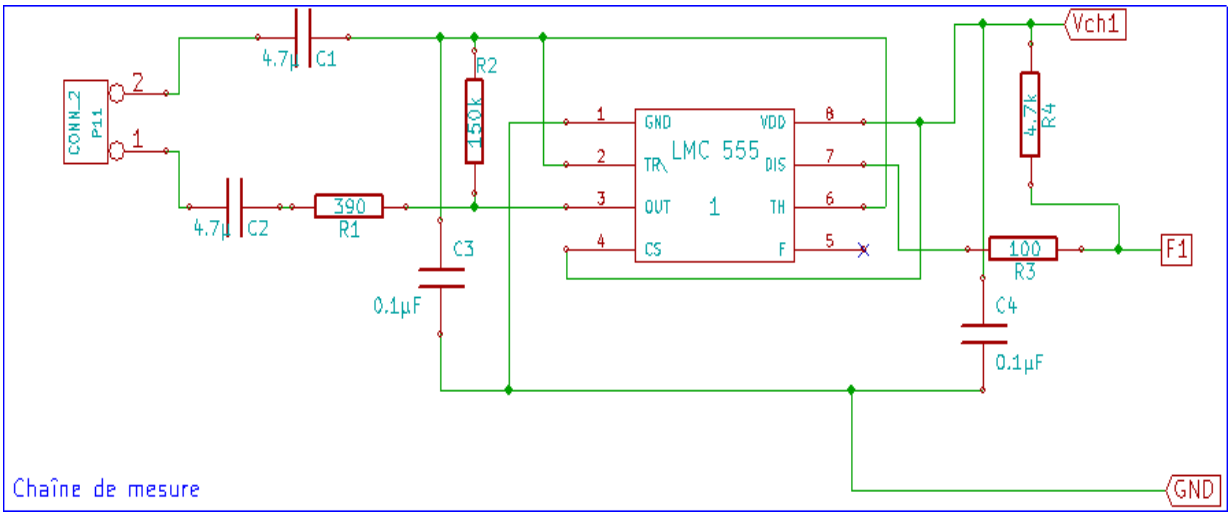


Figure 15. Exemple de schéma électronique réalisé pour la chaîne de mesure.

- CVpcb 

Cette application associe à chaque schéma électrique une empreinte. On peut créer de nouvelles empreintes. Le fichier généré est appelé netliste (voir Figure 16) et contient les informations concernant les différentes associations composant/empreinte. La partie « bleu » est la liste des composants présents sur le schéma et la partie « verte » correspond à la librairie des empreintes fournie avec le logiciel. Une fois cette netliste créée, il faut placer les composants et faire le routage avec PCBnew.

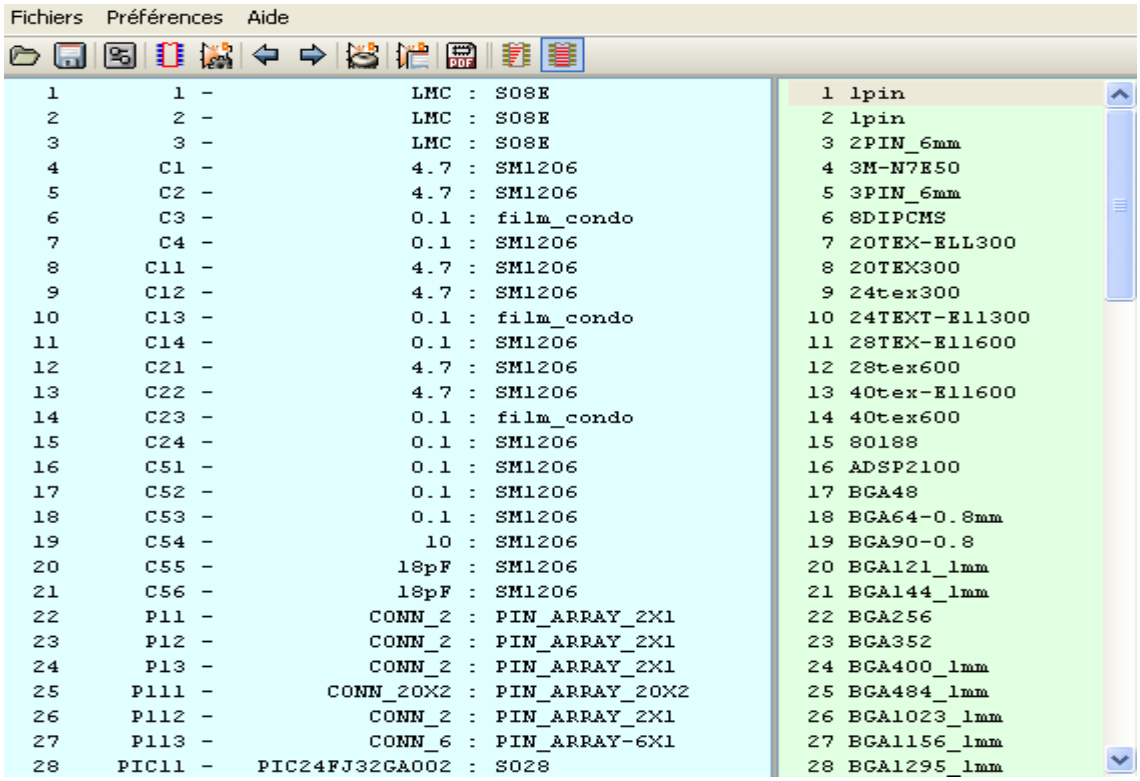


Figure 16. Exemple de netliste généré pour la chaîne de mesure.

- PCBnew 

Ce logiciel permet de placer les composants pour réaliser une carte. Les relations entre les composants étant faites à la première étape et les relations des composants avec leurs empreintes étant faites à la seconde étape, le routage peut se faire aisément, il n'y a que le placement et l'orientation des composants à ajuster.
Le logiciel permet un routage automatique, malheureusement celui-ci n'est pas vraiment optimal. Il est préférable de le réaliser manuellement. Cependant dans les cas où il y a plus de trois couches, le routage automatique permet de gagner du temps bien qu'il faille retoucher les pistes.

III.1.3. Schémas réalisés

Avant de commencer la réalisation des schémas électroniques, je dois prendre en compte tous les composants et leur rôle au sein de la carte. Pour m'aider dans cette tâche, j'ai réalisé le schéma synoptique de la carte fille (voir Figure 17). L'alimentation des trois chaînes de mesure, Vch1, Vch2 et Vch3, est contrôlée par le processeur PIC24FJ32GA002 sur le port des pins remappables (RP) respectivement RP11, RP13, RP15. Le processeur est relié au le bus I²C sur les ports RP prévus; le fil de donnée SDA est relié à RP9 et le fil de l'horloge SCK est relié à RP8.

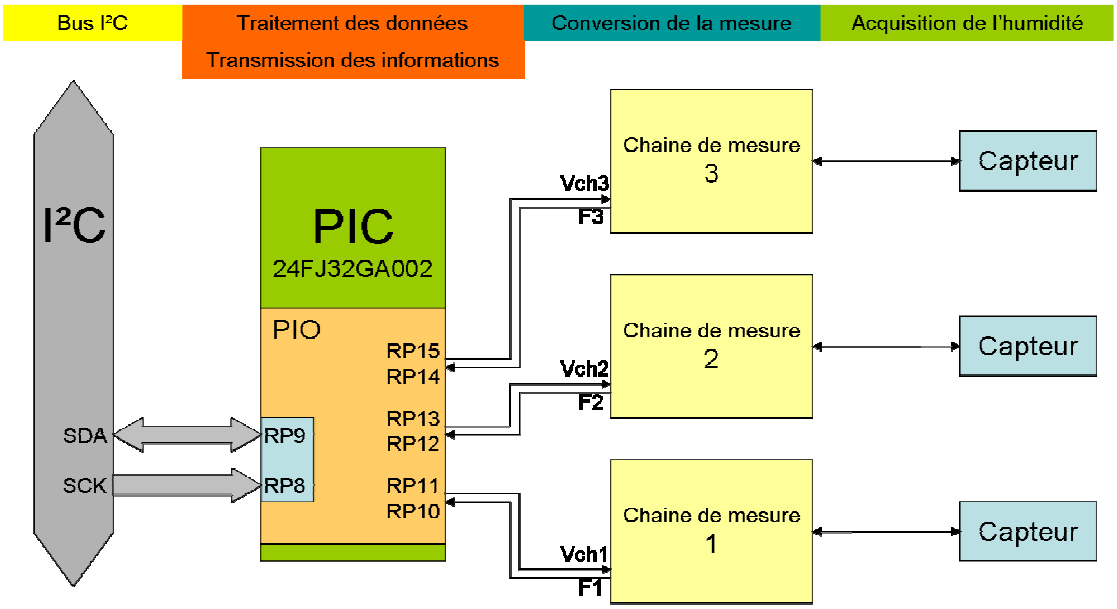


Figure 17. Schéma synoptique de la carte de mesure d'humidité.

J'ai commencé par réaliser la chaîne de mesure. Celle-ci (voir Figure 15) est composée d'un timer LMC555, de quatre résistances et condensateurs, d'un connecteur CONN_2 qui relie le capteur d'humidité à la chaîne de mesure, d'une alimentation Vch1 pilotée par le processeur et enfin, d'un circuit délivrant une fréquence F1 qui est traitée par le processeur. J'ai réalisé le schéma du composant LMC555.

Le processeur nécessite plusieurs éléments utiles à son bon fonctionnement. J'ai donc créé un schéma électronique (voir Figure 18) avec le processeur PIC24FJ32GA002 ainsi que le dessin du composant. Le processeur est relié à un connecteur CONN_6 qui permet de programmer un quartz 'CRYSTAL' accompagné de deux condensateurs C55 et C56 fixant la fréquence du processeur et enfin, de trois condensateurs C52, C53 et C54 nécessaire pour que le processeur puisse fonctionner.

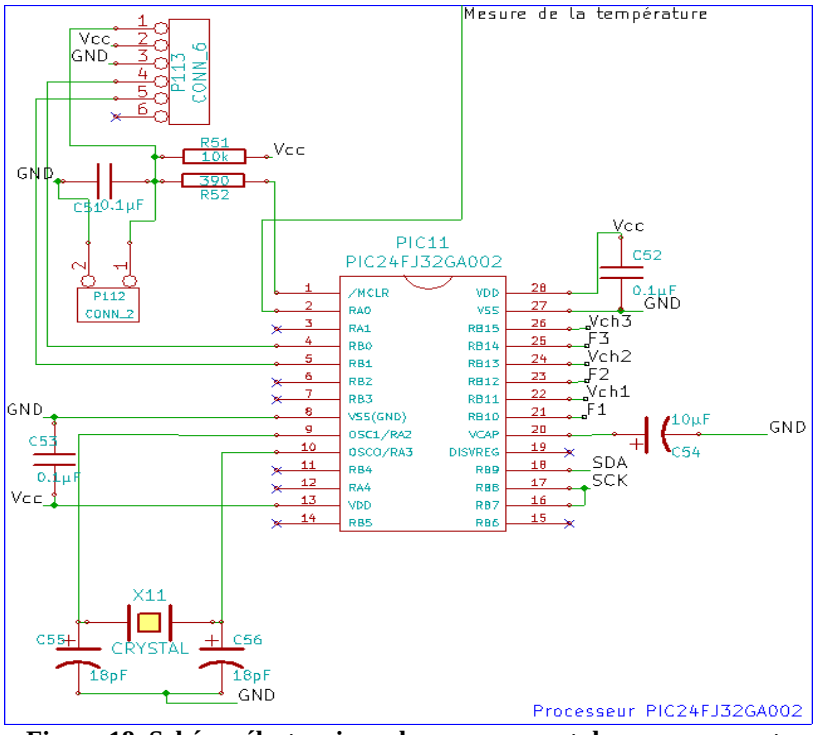


Figure 18. Schéma électronique du processeur et de ses composants

Enfin, j'ai réalisé le schéma du connecteur et j'ai ajouté un capteur de température que j'ai dû créer. Une fois tous les schémas électroniques complets, j'ai commencé la réalisation des empreintes. J'ai réalisé la netliste en affectant une empreinte à chaque composant. Pour certains composants, l'empreinte n'était pas disponible dans la librairie. J'ai donc réalisé l'empreinte à partir des dimensions du composant qui nous sont fournies par le fabriquant. Pour créer une empreinte, il faut réaliser un tracé de contour du composant et indiquer les pads du composant. Entre temps, j'ai listé les composants nécessaires à la réalisation de la carte. Puis, à partir de cette liste, j'ai rédigé le bon de commande afin d'obtenir les composants pour valider les empreintes et pouvoir, ensuite, démarrer la fabrication.

III.1.4. Réalisation des empreintes

Dans un premier temps, j'ai évalué la taille de tous les composants. Cela m'a permis de constater que les composants de type traversant ne sont pas, dans notre cas, adaptés à cause de leur taille. Par conséquent, le type de composant retenu est CMS (Composant Monté en Surface). Ces composants sont largement moins encombrants. Cependant, l'opération de soudure sur la carte est nettement plus délicate. Après avoir placé les composants le plus soigneusement possible, il faut réaliser le routage c'est-à-dire la connexion entre les différents composants à l'aide d'un conducteur de courant, ce que l'on appelle des pistes. Il faut respecter certaines règles de conception. Une fois que le routage est fini et que les règles sont respectées, il ne reste plus qu'à imprimer les schémas pour une vérification manuelle.

Les règles de conception sont les suivantes :

- Il ne doit pas avoir d'angles droit pour les pistes d'une même couche mais des angles proche de 45°. En effet, les angles droits sont plus difficiles à réaliser et ils sont plus fragiles car si le coin de l'angle est abimé, suivant la taille de la piste, cela peut la couper. De plus, les électrons circulent mal avec des angles droits car une partie est dissipée avant de changer de direction.

- Les pistes et les empreintes doivent avoir une distance d'écart minimale pour isoler les pistes. En effet, si les pistes sont trop proches, des perturbations magnétiques peuvent gêner le fonctionnement de la carte. Ce risque est accentué lorsque la carte fonctionne à haute fréquence. Dans ce cas, il est nécessaire d'ajouter un plan de masse qui permet d'isoler chaque piste.

- Les pistes doivent être les plus courtes possibles. Pour éviter de perturber un signal, il est préférable de limiter la longueur d'une piste. Cependant dans notre cas, le signal n'est pas perturbé parce que les dimensions de la carte sont suffisamment petites. En revanche, il faut minimiser l'espace utilisé par les pistes. Il faut éviter d'utiliser trop de vias. Les vias sont de grande taille par rapport aux pistes et, d'un point de vue électrique, il y a une atténuation du signal à chaque via. De plus, la réalisation des vias est particulièrement longue puisqu'au Cemagref, il faut les percer et les souder manuellement.

La couche du dessus est dessinée en rouge par défaut et la couche du dessous est dessinée en vert. Les composants sont entourés d'un liséré bleu. Les vias sont les disques blancs. Les disques verts sont les pads des composants traversant.

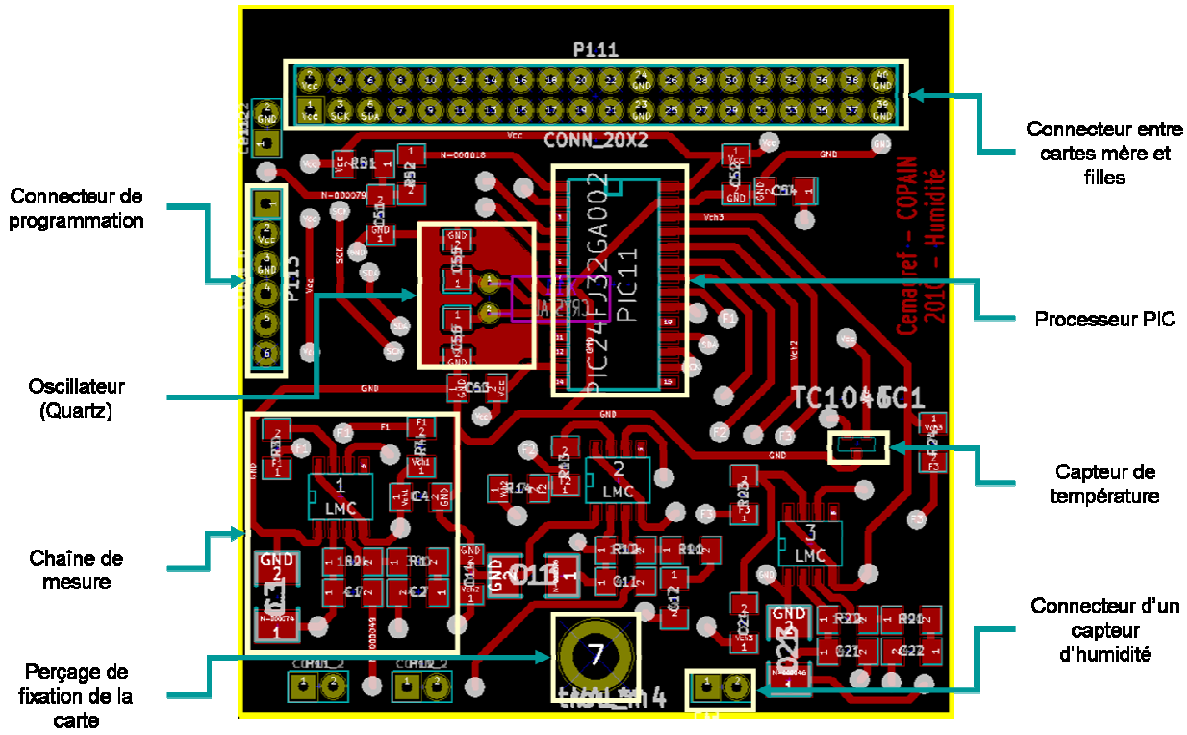


Figure 19. Schéma PCB de la carte fille.

III.2. Fabrication de la carte

III.2.1. 1ère étape : Réalisation du typon.

Le typon est un élément nécessaire lors de la fabrication de la carte. Un typon représente les pistes de cuivre que l'on souhaite fixer sur la plaque d'époxy photosensible. La réalisation du typon commence par la validation de la conception de la carte. Pour la valider, il faut s'assurer que les pistes respectent bien les règles de conception, que le schéma est correct avec le typon et que les empreintes des différents composants sont correctes et bien orientées. En parallèle, il faut commander les composants afin de vérifier le typon avec les composants réels avant sa réalisation. Dans notre cas, le typon est imprimé sur une feuille transparente (voir Figure 20).

Nous avons rencontrés des difficultés pour obtenir une bonne qualité de typon, En effet, pour que l'étape suivante fonctionne bien, le transparent doit avoir un fort contraste et être très foncé. Or la qualité de l'encre de l'imprimante a fait que le transparent n'était pas assez foncé pour générer une carte correcte. La solution a été de doubler les transparents (deux de chaque côté) tout en faisant très attention à leurs positionnements.

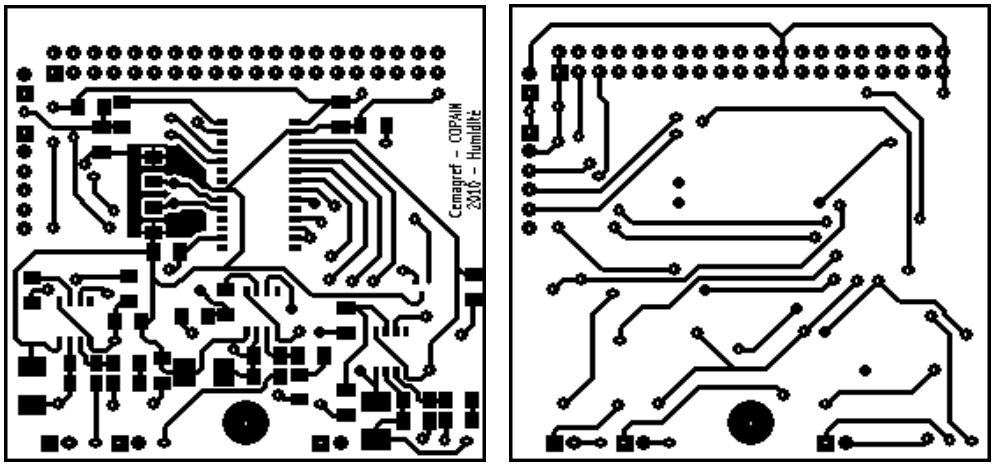


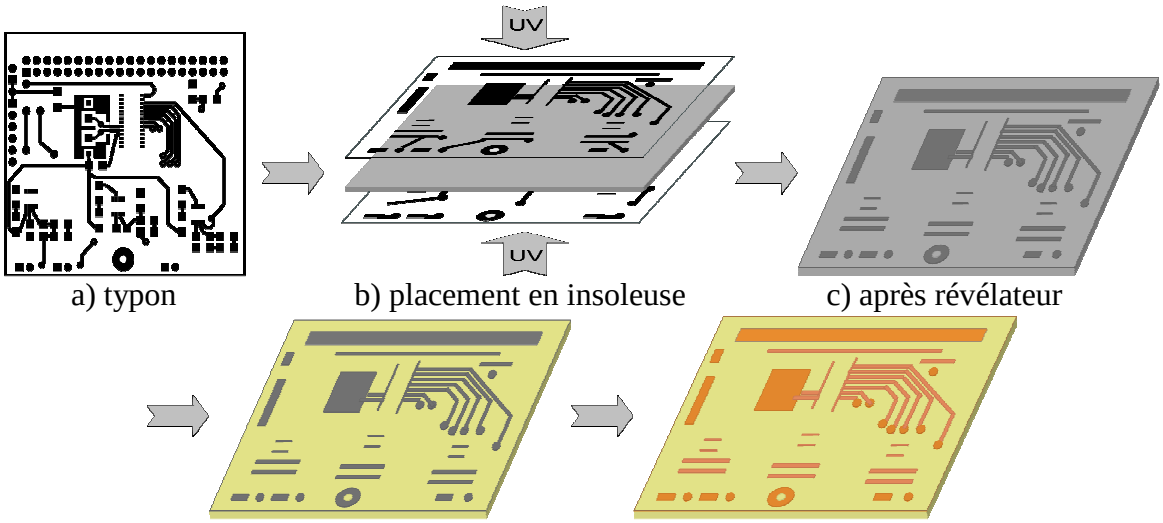
Figure 20. Typons couche du dessus et couche du dessous.

III.2.2. 2ème étape : Le travail de la plaque de cuivre.

Cette étape consiste à fabriquer les pistes (voir Figure 21). Pour cela, nous récupérons le typon que l'on place sur une plaque à vernis photosensible (voir Figure 21b). Ensuite, cette plaque est placée dans une insoleuse. L'insoleuse est une machine qui envoie des rayons Ultraviolets (UV) afin de sensibiliser le vernis qui protège les parties de cuivre à retirer correspondantes aux zones transparentes du typon.

Après une minute et trente secondes sous l'insoleuse, nous plaçons la plaque dans un révélateur afin d'enlever uniquement le vernis qui a été exposé aux UV. Une fois que le vernis ne protège plus le cuivre à enlever, nous plaçons la plaque dans la graveuse. La graveuse envoie du perchlorure de fer sur la plaque pendant trois minutes et trente secondes. Le cuivre en trop est donc retiré de la plaque. Avant de récupérer la plaque, la graveuse effectue un rinçage car le perchlorure de fer est particulièrement corrosif.

Ensuite, nous la plaçons dans un bain d'éliminateur. Ce liquide permet d'enlever le vernis qui protégeait le cuivre. Enfin, il ne reste plus qu'à rincer la plaque. Après quelques tests sur de petits bouts de cartes, j'ai déterminé les temps nécessaires pour chacune des opérations à effectuer pour que le cuivre ne soit pas retiré là où il devrait être présent.



a) typon b) placement en insoleuse c) après révélateur

d) placement dans la graveuse e) après le bain d'éliminateur

Figure 21. Etapes de la réalisation de la carte fille.

III.2.3. 3ème étape : Insertion des composants.

Parmi les composants à insérer, la plupart des composants sont de type CMS même s'il reste des composants de type traversant pour les gros composants comme le connecteur et les borniers. Tout d'abord, nous commençons par le placement des composants CMS. Pour cela, il faut appliquer de la pâte à braser sur les zones où l'on doit placer ces composants. Cette pâte permet de coller le composant et, lors du chauffage, la pâte va fondre et réaliser la soudure entre le composant et la piste. Cette opération s'appelle le brasage.

Le brasage est effectué par un four avec deux étapes dans la cuisson. La première dure deux minutes et trente secondes à 170°C et la seconde dure une minute et quinze secondes à 250°C. Il faut attendre une demi-heure avant de pouvoir sortir la carte, le temps qu'elle refroidisse.

Pour placer les composants traversant et pour réaliser les vias, il faut percer. Le perçage est de diamètre 0.8 mm. Une via a pour but de relier plusieurs pistes sur des couches différentes. Puisque la carte ne possède que deux couches, il est plus simple de réaliser la via à l'aide d'un fil que l'on soude des deux côtés de la carte dans notre cas. Lors d'une réalisation industrielle de la carte, les vias sont déjà métallisées, réalisant par la même occasion, la liaison entre les différentes couches. Ensuite, il faut placer les composants traversant dans le bon sens et sur le bon côté de la carte. Puis, il n'y a plus qu'à les souder au fer à souder. Une fois tous les composants soudés, il faut vérifier si la carte est correcte.

III.3. Tests et validation de la carte fille

Avant de réaliser l'insertion des composants, il y a une première vérification des pistes sur la carte. Tout d'abord, il faut vérifier que les pistes ne sont pas coupées entre elles. Pour cela, j'ai utilisé un multimètre pour mesurer la résistance des pistes sur le plus petit calibre.
Une résistance nulle signifie qu'il n'y a pas de coupures sur la piste. Dans le cas contraire, il faut repérer où la piste est coupée et, si cela est possible, il faut rétablir la connexion.
Ensuite, il faut vérifier qu'il n'y a pas de court-circuit entre deux pistes et surtout entre deux alimentations. Pour cela, il nous faut positionner le multimètre sur le calibre de la résistance la plus haute. L'affichage ne doit pas indiquer de résistance nulle. Les premières pistes à être testées sont l'alimentation et la masse. Si ce test est validé, je passe à l'étape d'insertion des composants.

Après l'insertion des composants, je vérifie les soudures et principalement celles des composants CMS puisque pour les composants traversant, la soudure est faite au fer à souder et, si la soudure est mal faite, je peux la rectifier rapidement. Cependant, les composants CMS sont collés à la carte du même côté de la piste donc rien ne me garantit que la pâte à braser ne s'est pas étalée jusqu'à court-circuiter le composant. Je m'assure, dans un premier temps, que les composants sont bien soudés et qu'ils ne se décollent pas. Je répète l'opération précédente pour détecter les courts-circuits. Si j'en détecte un, je dois dessouder le composant, enlever l'étain qui fait le court-circuit et enfin je dois le ressouder.

Une fois ces étapes de validation réalisées, je vérifie que la carte exécute correctement un programme simple. J'ai donc commencé par programmer le processeur pour configurer les pins et les faire changer d'état.

La première difficulté est venue des pins 21 et 22. Ces pins sont, au départ, associés à la programmation et je les ai rattachés à une chaîne de mesure. Même en changeant les valeurs du port, la tension relevée par l'oscilloscope était de 2.6V ce qui ne correspond pas à un état logique. La cause venait de la configuration générale du processeur. Après reconfiguration, les pins 21 et 22 ont pu être associés à la chaîne de mesure.

La seconde difficulté est due à une erreur. Pour diminuer la consommation de la chaîne de mesure, quelques modifications ont été apportées. Toutes ces modifications ont été testées sur une carte de test où l'on doit souder les composants et, à l'aide de l'oscilloscope, vérifier que la fréquence reçue est conforme au fonctionnement du capteur d'humidité. Nous avons donc effectué la modification sur un schéma et ensuite j'ai commencé la conception de la carte. Lors de la réalisation du schéma, nous nous sommes trompés sur une résistance de rappel. On faisait un rappel de la sortie de fréquence à la masse alors qu'il fallait faire un rappel à l'alimentation. Cette erreur était facile à corriger mais plus difficile à détecter. En effet, bien que la chaîne de mesure ait été alimentée, la fréquence en retour était toujours nulle. Dans un premier temps, j'ai pensé que ce dysfonctionnement pouvait être lié à la première difficulté rencontrée sur les pins 21 et 22. Après plusieurs réglages, j'en ai déduit qu'il fallait revérifier les schémas électriques ainsi que les pistes. Et finalement, en reprenant la chaîne de mesure testée, nous avons trouvé l'erreur. Une fois réparée, le processeur reçoit bien une fréquence conforme à nos attentes.

La troisième difficulté est la programmation de la communication entre les processeurs AT91SAM7S256 et le PIC24FJ32GA002 via le protocole I²C. La programmation n'est pas la même pour les deux processeurs puisque les registres, les fonctions C et la configuration sont différents. Une fois la communication établie, j'ai testé l'envoi et la réception d'un caractère entre les deux cartes. Cette étape étant validée, je dois désormais réaliser le programme permettant de faire fonctionner la carte pour son application.

IV. PROTOCOLE DE COMMUNICATION

IV.1. Environnement de programmation

L'environnement de programmation est adapté en fonction du matériel à programmer. Les processeurs qui doivent être programmés sont le processeur AT91SAM7S256 de chez Atmel et le processeur PIC24FJ32GA002 de chez Microchip.
L'Atmel se programme avec le logiciel IAR Embedded Workbench (voir Figure 22). Pour programmer, il faut tout d'abord créer un projet avec différents fichiers que l'on peut rattacher au projet. Tous les fichiers qui sont rattachés au projet seront compilés et injectés dans le processeur. Pour injecter le code compilé dans le processeur, il faut posséder un programmateur JTAG comme le J-link de SEGGER qui se branche sur un port USB et sur le port JTAG du processeur. Pour programmer le processeur, il faut que celui-ci soit alimenté. En effet, l'alimentation doit être gérée indépendamment de l'appareil de programmation.

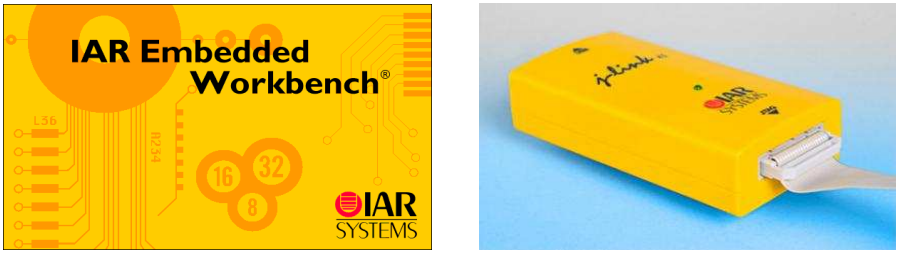


Figure 22. Logo du logiciel et programmateur J-link.

Le PIC se programme avec le logiciel MPLAB IDE v8.50 (voir Figure 23). Ce logiciel est initialement conçu pour programmer en assembleur tous les processeurs du fabricant Microchip. Nous avons choisi le langage C pour programmer le PIC ce qui implique l'ajout du compilateur C30 spécialement conçu pour les familles PIC24 et dsPIC30. De la même manière que le logiciel IAR, MPLAB gère les fichiers par projet. Le code compilé est injecté dans le processeur par le programmateur MPLAB ICD 2. Il est relié à l'ordinateur par le port USB et au processeur sur un des ports de programmation.

Figure 23. Logo du logiciel et programmateur MPLAB ICD 2.

Les deux processeurs ne sont pas du même fabricant ce qui implique une différence entre les configurations et les fonctions de gestion des interfaces. L'Atmel est un processeur 32 bits et le PIC est un processeur 16 bits. Cette différence d'architecture impose de faire attention au stockage des informations qui sera différente. Par exemple, un entier sera codé sur 2 octets pour le PIC tandis que sur l'ATMEL il sera codé sur 4 octets. On a donc une limitation implicite de la plage de valeurs pour ce type.

IV.2. Utilisation du protocole I²C

L'ensemble du programme développé pour la carte fille, durant le stage, doit réaliser les mesures, les stocker et les transmettre à la carte mère. J'ai commencé par implémenter un programme générique qui permet la communication entre les deux cartes.

IV.2.1. Stockage des données

Le premier programme, que j'ai implémenté, a pour but de gérer le stockage des informations de deux manières différentes. La première solution (voir Figure 24) est réalisée à partir d'un tableau de listes chainées. L'avantage de cette méthode est qu'elle est rapide à programmer et elle permet de revoir efficacement le principe des listes chaînées. Cependant, elle ne permet pas de distinguer les informations propres aux capteurs de celles propres à la carte fille, ni de connaître facilement les informations d'un capteur en particulier.

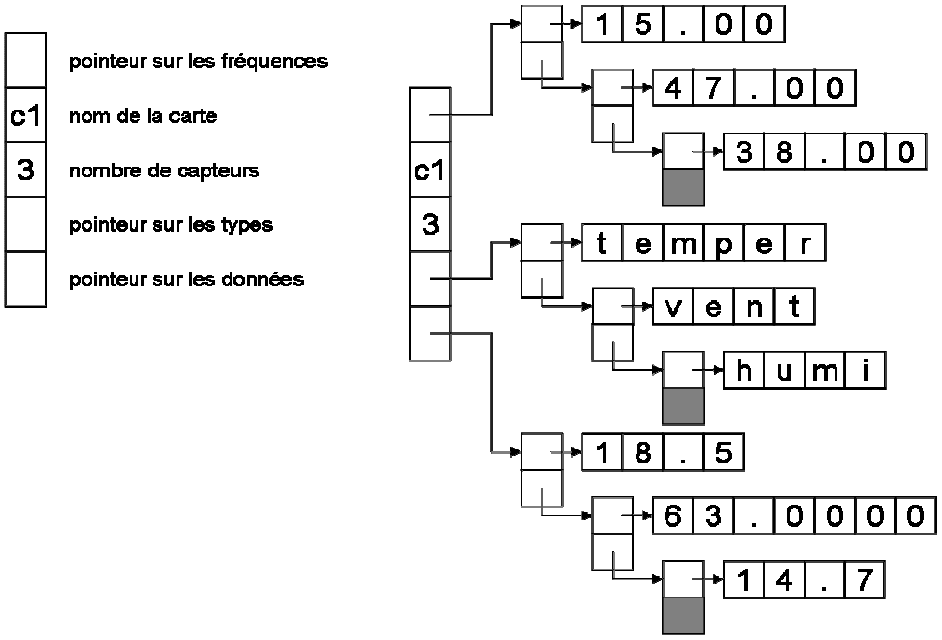


Figure 24. Stockage des informations. Solution 1.

La deuxième solution (voir Figure 25), qui a été retenue, permet de distinguer les informations propres aux capteurs et les informations de la carte fille grâce à l'utilisation d'une structure en C. Pour alléger cette structure, nous avons décidé qu'un capteur ne permet d'avoir qu'une seule donnée même si physiquement un capteur peut en fournir plusieurs. Par contre, en augmentant le nombre de structure chaînée pour chaque capteur, on répètera certaines informations qui risquent d'encombrer inutilement la mémoire du processeur de la carte mère. Avec cette solution, il nous est plus facile d'avoir l'identifiant du capteur pour récupérer une donnée en particulier.

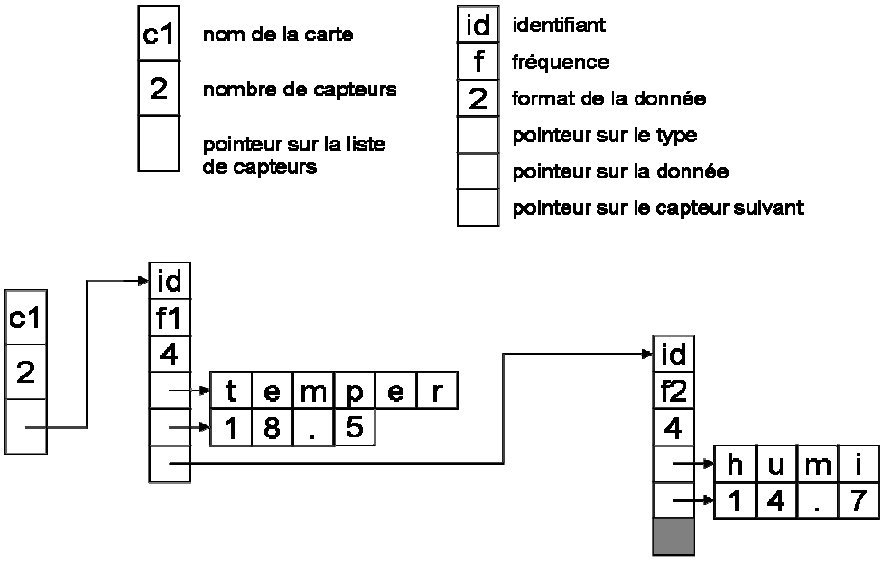


Figure 25. Stockage des informations. Solution 2.

Dans la Figure 26, le code de la structure correspond à la solution retenue. La structure du capteur contient l'identifiant du capteur, une fréquence d'acquisition, le format de la donnée, le type de la donnée, la valeur de la donnée et enfin le pointeur sur la structure du capteur suivant. La structure de la carte fille contient l'identifiant de la carte, le nombre de capteur et le pointeur sur la liste des structures des capteurs.

```
typedef struct Def_capteur
{
  int            id;                  // identifiant du capteur
  float          freq;                // fréquence du capteur
  int            format;              // ce nombre est en nombre d'octet, il définit la taille de la donnée
  int            type;                // indique le type de la donnée
  char         * donnee;              // valeur de la donnée
  struct         Def_capteur * next;  // pointeur sur le capteur suivant

}Def_capteur_t;

typedef struct Def_carte
{
  Int            id;                  // identifiant de la carte
  int            nb_capteur;          // nombre de structure capteur
  Def_capteur_t *capteur;             // pointeur sur la liste chaînée de capteur
}Def_carte_t;
```

Figure 26. Structure de la carte fille.

Une fois le stockage des données réalisé dans chacune des cartes, j'ai commencé à travailler sur le transfert de ces données. J'ai donc implémenté le protocole de communication choisi suite à l'étude présentée à la section II.3.

IV.2.2. Implémentation du protocole I²C

- Les Trames I²C

Les trames I²C utilisées sont sous la forme suivante :

- La première trame (voir Figure 27) a pour but de récupérer une ou plusieurs informations. Comme il n'y a que le maître qui peut mettre un terme à la communication, le programme doit permettre de savoir quand la communication va être terminée.

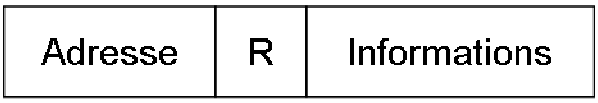


Figure 27. Trame de réception d'informations.

- La seconde trame I²C (voir Figure 28) a pour but de demander une information. La trame indique que le maître va envoyer des codes à la carte fille via le bit d'écriture (W). Le code de la commande permet de définir les informations que l'on recherche et l'ID du capteur permet de différencier si l'on s'adresse à la carte fille, à un capteur en particulier ou à tous les capteurs.

Figure 28. Trame de demande d'informations.

Pour permettre une meilleure gestion de la part de la station centrale de collecte, elle peut s'adresser au nœud et demander d'effectuer des changements qui devront être répercutés sur les différents capteurs. Les cartes mères doivent donc demander des changements à la structure d'une carte fille. Un exemple d'utilisation est celui d'une carte fille que l'on retire d'un nœud sans fil pour être placée sur un autre nœud sans fil. Le problème est que la carte fille a des identifiants capteurs identiques à d'autres cartes filles du nouveau nœud sans fil auquel elle est associée. Il faut donc les changer avec la trame suivante de la Figure 29. Elle permet donc de modifier des informations, comme le nombre de capteurs d'une carte fille, les identifiants et les fréquences d'acquisitions. La trame de modification est une trame de demande d'informations qui contient un champ avec la nouvelle valeur de l'élément à modifier.

Figure 29. Trame de modification d'une information.

- Les Opérations

L'utilisation de ces trames permet de réaliser plusieurs opérations.
La première opération est l'initialisation (voir Figure 30). Elle permet à la carte mère de créer la structure de la carte fille qu'elle interroge, ainsi que celle des capteurs qui lui sont associés. L'opération se termine une fois que toutes les informations ont été reçues. L'initialisation sert à configurer la structure du côté de la carte mère.

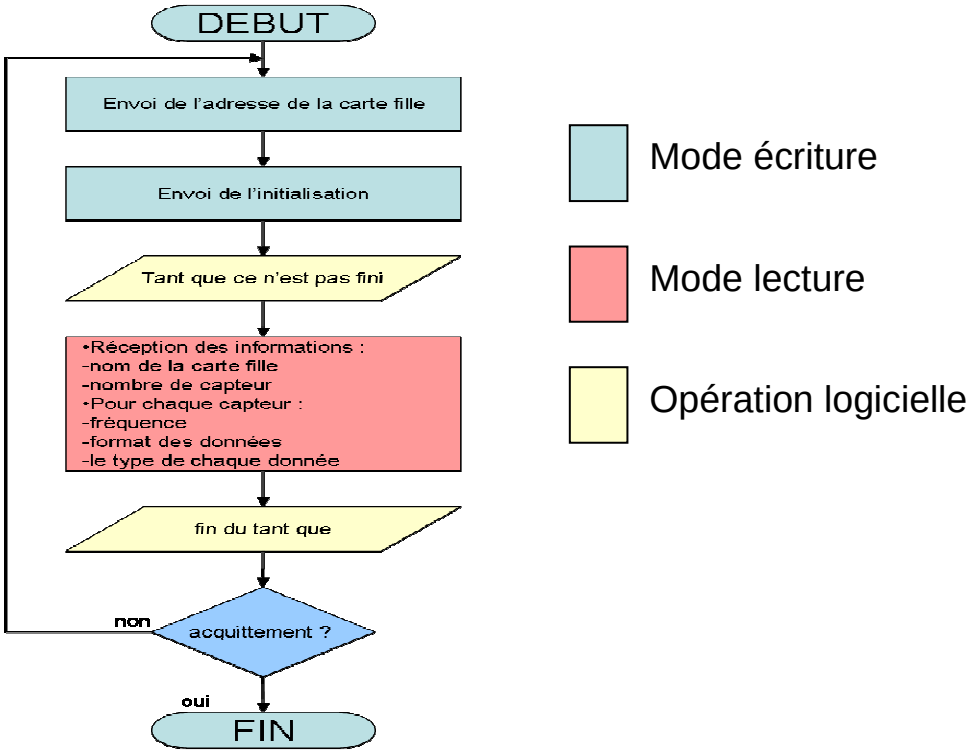


Figure 30. Organigramme de la commande d'initialisation.

L'opération suivante permet de récupérer une information de la structure stockée en mémoire du processeur de la carte fille. L'opération de réception (voir Figure 31) est la plus utilisée puisque l'initialisation n'a lieu qu'au démarrage ou au redémarrage de la carte mère. Cette opération a pour but de recueillir les données des capteurs.

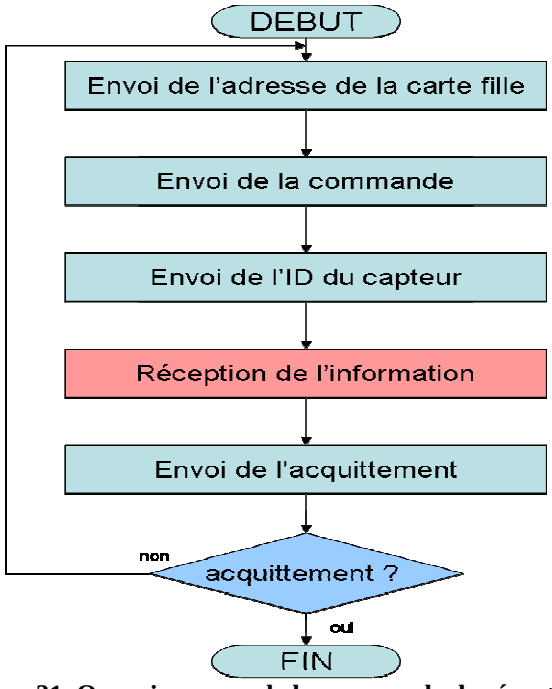


Figure 31. Organigramme de la commande de réception.

La dernière opération permet de modifier la structure stockée en mémoire du processeur de la carte fille. L'opération de changement (voir Figure 32) peut être appelée en cours de

fonctionnement. L'opération est délicate puisqu'il faut s'assurer que le changement a bien fonctionné. En effet, dans le cas du changement de l'identifiant du capteur, si la carte fille ne renvoie pas le bon identifiant, la carte mère ne peut plus contacter le bon capteur à partir de l'ancien identifiant. Pour que la carte mère puisse contacter à nouveau le capteur, elle doit utiliser l'identifiant qu'elle a reçue de la carte fille. La carte fille effectue le changement d'identifiant directement à chaque fois qu'elle reçoit la commande. La carte mère valide le changement une fois qu'elle a reçue le bon identifiant.

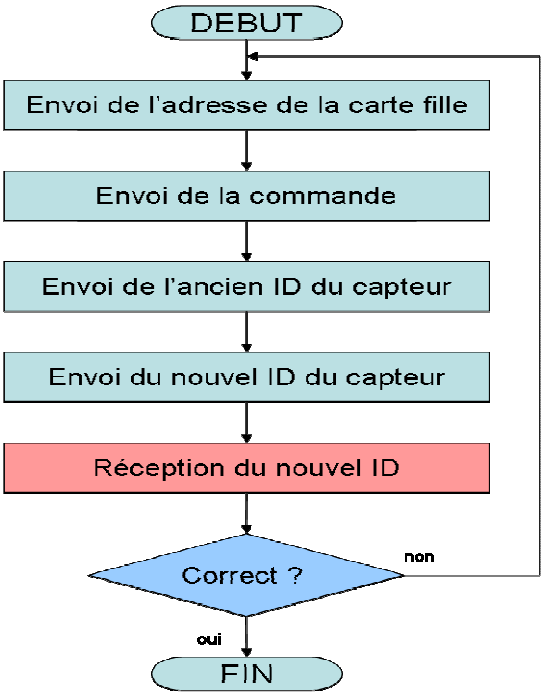


Figure 32. Organigramme de la commande de changement d'identifiant.

IV.3. Fonctions diverses

Afin de tester le fonctionnement du programme, avant d'avoir un exemplaire de carte fille, j'ai installé une machine virtuelle Linux via le logiciel Virtual Box développée par Sun. La communication a été simulée à l'aide des sockets en langage C. La communication par socket nécessite de convertir tout ce que l'on souhaite envoyer en des chaînes de caractères. J'ai donc créé des fonctions de conversions des types de variables en chaîne de caractères et inversement. Les simulations à l'aide des sockets m'ont permis de valider l'ensemble de l'implémentation du protocole de communication.

IV.3.1. La fonction de sélection de commande

J'ai réalisé la fonction de sélection de commande (voir Figure 33) qui permet, à la carte mère, d'envoyer une commande à une carte fille et de réceptionner l'information en conséquence. Cette fonction possède une fonction jumelle qui permet à la carte fille d'identifier la commande reçue et de fournir l'information désirée.
Cette fonction renvoie « 0 » si la ou les informations n'ont pas été reçues et « 1 » dans le cas contraire. La fonction commence par une structure conditionnelle sur le type de l'identifiant capteur à envoyer. La commande peut en effet s'adresser à un capteur en particulier ou à une carte fille. Une fois ce test effectué, le numéro de la commande permet de déterminer

l'information que la carte fille doit fournir et celle que la carte mère va recevoir pour mettre à jour la structure stockée en mémoire.
La fonction sélection est composée de plusieurs petites fonctions. La fonction « ecriture_i2c » permet d'envoyer, via le bus I²C, un paramètre. La carte fille récupère ce paramètre grâce à la fonction « lecture_i2c ». Les fonctions qui commencent par « recep » permettent de récupérer l'information de la carte fille et de modifier la structure stockée en mémoire. Le corps complet de cette fonction est fournit en annexe.

```
int selection(Def_carte_t * pcarte, int code_commande, int id_capteur, char * addr, char * buffer)
{
  int select =0;
  ecriture_i2c(addr);
  ecriture_i2c(code_commande);
  ecriture_i2c(id_capteur);

  if(id_capteur == 0)
  {
    /* cas où la commande ne s'adresse qu'à la carte fille */
    switch(code_commande)
    {
      case 1:
        /* indique le nom de la carte */
        select = recep_carte_id(pcarte,buffer);
      break;
      case 2:
        /* indique le nombre de capteurs */
        select = recep_carte_nbcapteur(pcarte,buffer);
      break;
      default:
        /* indique toutes les données des capteurs */
        select = recep_capteur_total(pcarte,buffer);
      break;
    }
  }
  else
  {
    /* cas où seul les informations des capteurs sont intéressants */
    switch(code_commande)
    {
      default:
        /* indique la donnée */
        select = recep_capteur_donnee(pcarte,id_capteur,buffer);
      break;
    }
  }
  return select;
}
```

Figure 33. Fonction sélection en langage C.

La fonction « sélection » ne traite que les commandes liées à la demande d'une ou de plusieurs informations. Les autres types de commandes sont traités par des fonctions spécifiques.

IV.3.2. La fonction de changement

J'ai implémenté une fonction spécifique permettant de modifier l'identifiant d'un capteur de la carte fille nommée « changement_id » (voir Figure 34). La commande est envoyée à la carte fille ciblée. Une fois que celle-ci a bien identifié le capteur et qu'elle l'a bien modifiée, la carte fille envoie le nouvel identifiant pour s'assurer que la modification a bien été effectuée et

qu'elle a été correcte. La carte mère compare cet identifiant pour s'assurer qu'il s'agit bel et bien de l'identifiant qu'elle a envoyé. Dans le cas contraire, la commande est envoyée à nouveau avec pour identifiant celui qui a été reçu précédemment.

```
void changement_id(Def_carte_t * pcarte, char * addr , int id_capteur, int buffer, int new_id_capteur)
{
  int       trouve = 0,
            fin = 0;
  Def_capteur_t ** ppcapteur_cour = NULL;                                   // pointeur de recherche
  ppcapteur_cour = (Def_capteur_t **) malloc(sizeof(Def_capteur_t *));      // allocation

  buffer = id_capteur;

  while(!fin)
  {
    ecriture_i2c(addr);                             // adresse de la carte fille
    ecriture_i2c(CHG_ID);                           // code de commande de changement d'un ID
    ecriture_i2c(buffer);                           // identifiant du capteur
    ecriture_i2c(new_id_capteur);                   // nouvel identifiant du capteur

    lecture_i2c(buffer);

    if(buffer == new_id_capteur)                    // vérification que la carte fille a bien modifié l'id du capteur
    {
      (*ppcapteur_cour) = parcours(*pcarte,id_capteur,&trouve);

      if(trouve == 1)
      {
        (*ppcapteur_cour)->id = new_id_capteur;
        fin = 1;
      }
    }
  }
}
```

Figure 34. Fonction changement_id en langage C.

Lorsque le protocole de communication et la carte ont été réalisés, j'ai commencé la programmation du processeur de la carte fille. J'ai effectué les configurations des différents pins du processeur pour qu'il puisse fonctionner avec le reste de la carte.

IV.3.3. La lecture de la chaîne de mesure

L'une des difficultés de la programmation a été celle de « l'input capture ». Celui-ci permet de calculer la fréquence que renvoie la chaîne de mesure. « L'input capture » est un type d'entrée du processeur qui se programme sur un pin. Ce module a été développé par le fabricant Microchip afin de faciliter la mesure de la fréquence et de dater précisément les changements d'état d'une broche. A partir des fichiers exemples fournit sur le site de Microchip, j'ai configuré le processeur pour obtenir des interruptions sur chaque front montant du signal issu de la chaîne de mesure. Malgré cela, je n'ai eu aucune interruption même avec un changement d'état sur la broche. Nous avons finalement trouvé la solution en configurant le « timer » que l'on choisi pour la mesure, en plus de la configuration de « l'input capture ».

Lors de cette configuration (voir Figure 35), il faut d'abord préciser la broche qui va être utilisée _IC1R = 14. Ici, il s'agit du pin 14 qui fait parti du port des pins remappables(RP). Pour le choix du « timer 2 », il faut configurer le registre IC1CON. Il faut aussi indiquer le mode : une interruption à chaque capture et, une datation tous les 4 fronts montants sur l'entrée. En configurant "l'input capture" avec IC_EVERY_4_RISE_EDGE le temps mesuré

correspond au temps écoulé entre 4 fronts montants soit trois périodes du signal. Enfin, il faut configurer l'état de « l'input capture », _IC1IF indique s'il y a une interruption, _IC1IP indique la priorité et _IC1IE indique que « l'input capture » est prête à fonctionner.
La configuration du « timer 2 » permet de créer les interruptions. Le processeur que je programme est de type 16bits, donc il faut configurer le « timer 2 » pour que sa valeur ne dépasse pas 16bits. La fréquence du « timer » est définie par la variable T2_SOURCE_INT. Dans cette configuration, la fréquence du « timer » est égale à la moitié de la fréquence du processeur. Ceci signifie que le « timer » s'incrémente tous les deux coups d'horloge du processeur. PR2 indique la valeur maximale du « timer ». La valeur du « timer » est initialisée à 0. Le statut du « timer » est initialisé. Il est enfin prêt à fonctionner.

```
void configInputCapture1(void)
{
  _TRISB14 = 1;                                  // configuration du pin 14 en entrée
  _IC1R = 14;                                    // affecte l'input capture au pin 14
  IC1CON = IC_TIMER2_SRC |                       //Timer2 source
           IC_INT_1CAPTURE |                     //Interrupt every capture
           IC_EVERY_4_RISE_EDGE;                 //capture every 4th edge
  _IC1IF = 0;
  _IC1IP = 1;                                    //pick a priority
  _IC1IE = 1;                                    //enable
 }

void  configTimer2(void)
{
  T2CON = T2_OFF | T2_IDLE_CON | T2_GATE_OFF
          | T2_32BIT_MODE_OFF
          | T2_SOURCE_INT
          | T2_PS_1_1 ;                          //1 tick = 1.6 us at FCY=40 MHz
  PR2 = 0xFFFF;                                  //maximum period
  TMR2  = 0;                                     //clear timer2 value
  _T2IF = 0;                                     //clear interrupt flag
  T2CONbits.TON = 1;                             //turn on the timer
}
```

Figure 35. Fonction de configuration de l'input capture en langage C.

IV.4. Tests et validation du protocole I²C

Les tests du bus I²C ont permis de valider la carte fille. Les tests sont effectués avec les programmateurs et un port de communication UART avec un terminal disponible pour Windows. Le terminal (voir Figure 36) permet de récupérer des messages envoyés par le processeur Atmel. Ainsi, je peux envoyer une information au processeur PIC que celui-ci me renvoie aussitôt (cela nécessite deux trames par communication).

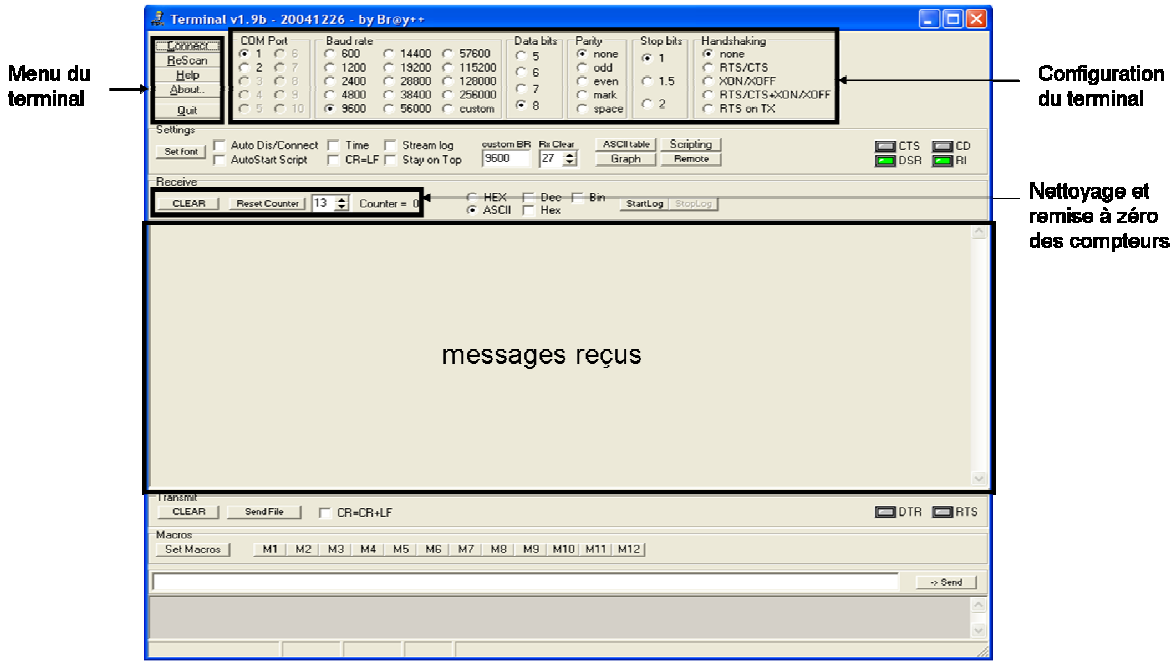


Figure 36. Image du terminal de communication UART.

Le premier programme que j'ai testé a pour but de vérifier la configuration de l'I²C. Le processeur Atmel envoie des trames de commande d'un octet suivi d'une trame de réception pour récupérer l'information de la part du PIC, qui renvoie l'octet de commande qu'il a reçu. Une boucle permet d'envoyer en continu les différentes trames. En variant les trames, j'ai remarqué qu'il y avait un décalage au démarrage du processeur, il faut attendre deux envois pour que le PIC se synchronise.

J'ai effectué de nombreux tests pour la configuration de « l'input capture ». Les premiers tests avaient pour but d'allumer les différents pins d'alimentation des chaînes de mesure. La configuration générale du processeur m'a empêché de pouvoir alimenter la troisième chaîne de mesure. En effet, ces pins sont réservés en priorité pour la programmation du processeur et il faut donc désactiver cette utilisation de ces pins avant de pouvoir leurs affecter une autre fonction. Une fois que j'ai pu piloter toutes les chaînes de mesure, j'ai observé la fréquence en retour à l'aide de l'oscilloscope et ceci en faisant osciller la tension d'alimentation de la chaîne de mesure. J'ai observé une tension nulle anormale. J'en ai donc conclu qu'une erreur a été commise lors de la réalisation du schéma électronique. C'est à ce moment là que nous avons modifié la résistance de rappel qui était mal placée. Une fois cette erreur corrigée, j'ai commencé la lecture de cette fréquence à l'aide de « l'input capture ».

Pour tester « l'input capture », j'ai utilisé le mode Debug de MPLAB (voir Figure 37), ce qui m'a au départ permis de remarquer que l'interruption ne déclenchait pas. Après avoir corrigé et configuré le « timer2 », j'ai pu récupérer la valeur du « timer2 » et la période de la mesure. La fenêtre en haut à gauche de la Figure 37 indique la valeur de tous les registres du processeur. La fenêtre du bas permet d'indiquer les variables du programme que l'on souhaite ainsi que les valeurs des registres qui nous intéressent. La fenêtre de droite contient le programme que l'on débugge. Je peux, en outre, placer des breakpoints qui permettent de stopper l'exécution du processeur sur une ligne du programme. Les valeurs sont, par défaut, en hexadécimal mais elles peuvent être mises en binaire ou en décimal ou, en caractère avec le code ASCII.

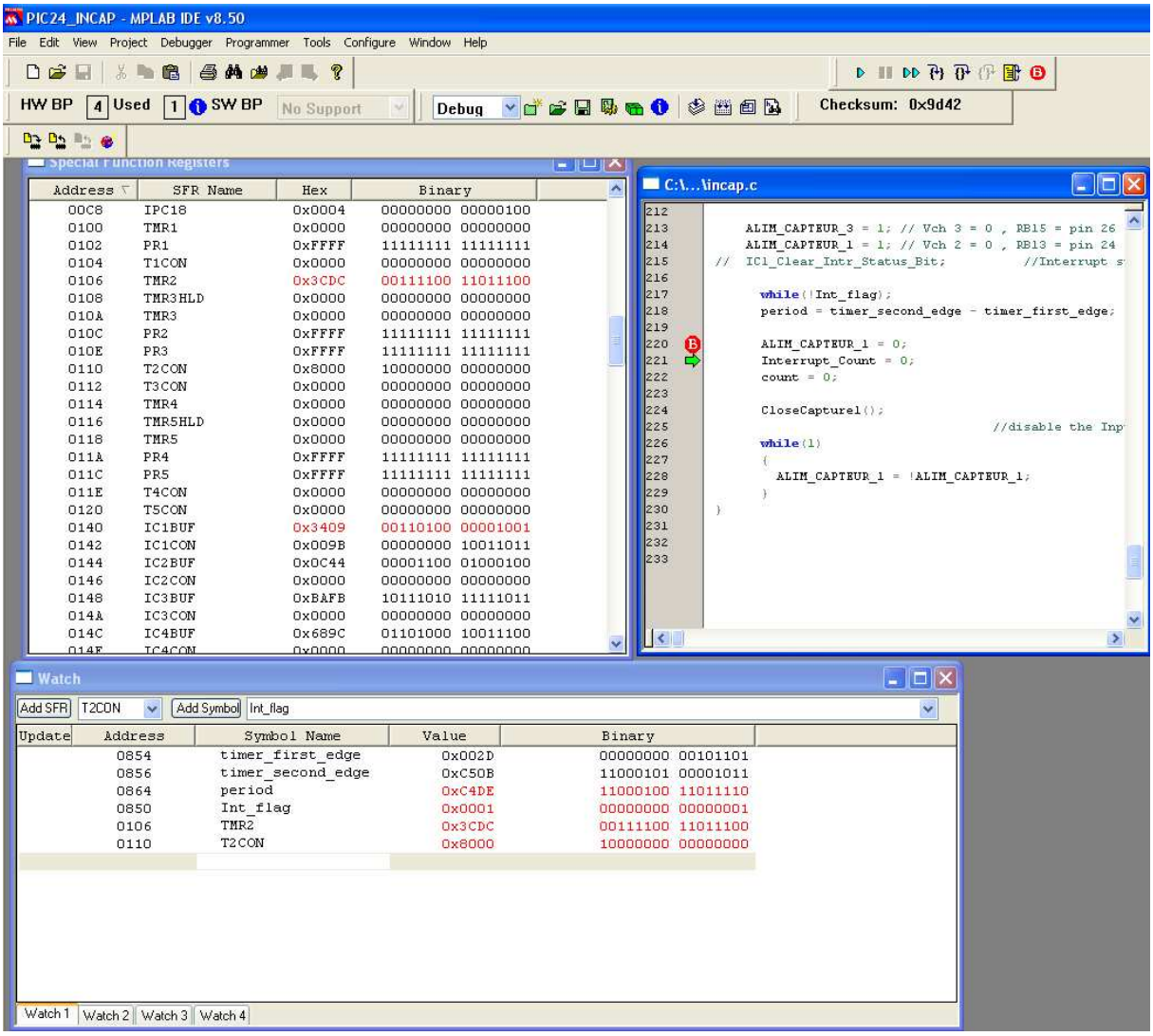


Figure 37. Mode Debug de MPLAB IDE v8.50.

IV.5. Bilan de la programmation

Dans un premier temps, le protocole a été développé, par l'intermédiaire de sockets, afin de simuler deux processeurs qui communiquent. Les fonctions de lecture et d'écriture ne sont pas contraignantes puisqu'elles nécessitent uniquement une chaîne de caractère comme paramètre. Ce protocole fonctionne correctement.

Les processeurs peuvent s'échanger des trames de type I²C. Pour le moment, celles-ci ne peuvent pas avoir plus d'un octet en plus de l'adresse de la carte fille. Mais, cela me permet tout de même d'effectuer des tests pour récupérer une information envoyée.

La fonction « d'input capture » me permet de relever une fréquence conforme au cahier des charges de la chaîne de mesure. Cependant, une seule chaîne de mesure a été modifiée donc aucun test de « l'input capture » n'a été fait pour les deux autres chaînes. De plus, je dois m'assurer de pouvoir configurer trois « inputs capture » et observer leur fonctionnement au même moment.

Pour tester le fonctionnement du protocole I²C avec les deux processeurs, j'ai utilisé à la fois le mode Debug de MPLAB et le terminal UART. Cette partie des tests est en cours. Ces différents éléments fonctionnent séparément. Les principaux problèmes sont liés aux contraintes des systèmes temps réel. Les deux processeurs ne doivent pas se décaler en cours d'exécution. S'il y a un décalage, il faut que les processeurs se synchronisent à nouveau. Les quelques tests, que j'ai pu réaliser, ne sont pas assez nombreux et la communication ne fonctionne pas correctement. J'ai pu détecter un problème dans le programme des fonctions de lecture et d'écriture. Lorsque j'ai testé le protocole, je n'ai pas apporté de modification sur ces deux fonctions. Or j'ai modifié les paramètres d'entrée de ces fonctions pour qu'elles puissent être adaptées aux fonctions I²C et leurs contenus. Des erreurs persistent. Je compte tester uniquement ces deux fonctions pour chacun des processeurs. Une fois que ces deux fonctions fonctionneront correctement, j'effectuerai de nouveaux tests sur le protocole I²C.

Conclusion

L'objectif de mon stage était de réaliser une carte dédiée à la mesure de l'humidité du sol, cette carte devant s'intégrer dans un nœud sans fil.
La maîtrise d'une nouvelle architecture mise au point pour les nœuds sans fil a été nécessaire.
- Le choix du protocole de communication et son adaptation à notre architecture ont été implémentés.
- La conception et la réalisation de la carte avec le choix des composants ont été correctement effectuées. Cette carte permet bien d'effectuer un relevé d'humidité du sol.
- L'intégration dans le nœud n'est pas complètement opérationnelle. Le protocole de communication n'est pas complètement intégré compte tenu d'un planning restreint par rapport aux objectifs.

Cependant, la carte de mesure remplit son rôle. Elle peut effectuer des relevés de façon autonome. Elle est désormais une base solide pour réaliser de nombreux tests indispensables à la correction du protocole de communication.

Le manque d'expérience dans la programmation des processeurs, principalement la configuration des différents ports, ont été un frein à l'avancée de mon travail. Cela a occasionné un retard dans ce planning très restreint.

Au delà du cadre de mon stage, mon protocole de communication pourra être encore amélioré en perfectionnant les tests de vérification afin de s'assurer que l'information reçue est correcte. En effet, le protocole prévoyant le fait que l'on puisse détecter les erreurs, il serait ainsi intéressant d'établir une gestion de ces erreurs en fonction des informations erronées. Enfin, les interruptions générées par la communication I^2C peuvent servir à réveiller le processeur. L'action d'endormir le processeur permet de diminuer considérablement la consommation de la carte et donc du nœud. Cet élément est crucial dans le domaine du développement actuel des réseaux de capteurs sans fil.

Bibliographie

JACQUOT A.; Réalisation d'un système modulaire multisupports multiprocesseurs : MobiPlus (rapport de stage de 3ème année, Clermont-Ferrand, 2006)

JACQUOT A., Supervision de réseaux d'objets intelligents communicants sans fil, Thèse en informatique, Université Blaise Pascal, 2010.

[Philips Semiconductors 1995] Philips « The I²C-bus and how to use it (including specifications) »

Webographie

[Microchip Data Sheet] Microchip « PIC24FJ64GA104 Family Data Sheet »
http://ww1.microchip.com/downloads/en/DeviceDoc/39881D.pdf

[Microchip Data Sheet] Microchip « PIC16F882/883/884/886/887 Data Sheet »
http://ww1.microchip.com/downloads/en/DeviceDoc/41291F.pdf

[JYOTI Shrinivas et HARSHA.J.M] Microchip Code Examples « CE349 Input Capture » et « CE338 I2C Slave »
http://www.microchip.com/CodeExamplesByFunc.aspx

[Microchip Library] Microchip «I2C™ Master Library Module (Interrupt-driven) »
http://ww1.microchip.com/downloads/en/DeviceDoc/i2cmint.readme.pdf

[Emesystems] Emesystems « SMX: soil moisture transmitter »
http://www.emesystems.com/smx.htm

[ABCELECTRONIQUE] forum de concepteur électronique,
http://www.abcelectronique.com/forum/

[FUTURA-SCIENCE] forum d'aide électronique,
http://forums.futura-sciences.com/electronique/

[WIKIPEDIA] Encyclopédie libre, http://fr.wikipedia.org/wiki/

Annexes

Schéma électronique de la carte fille.

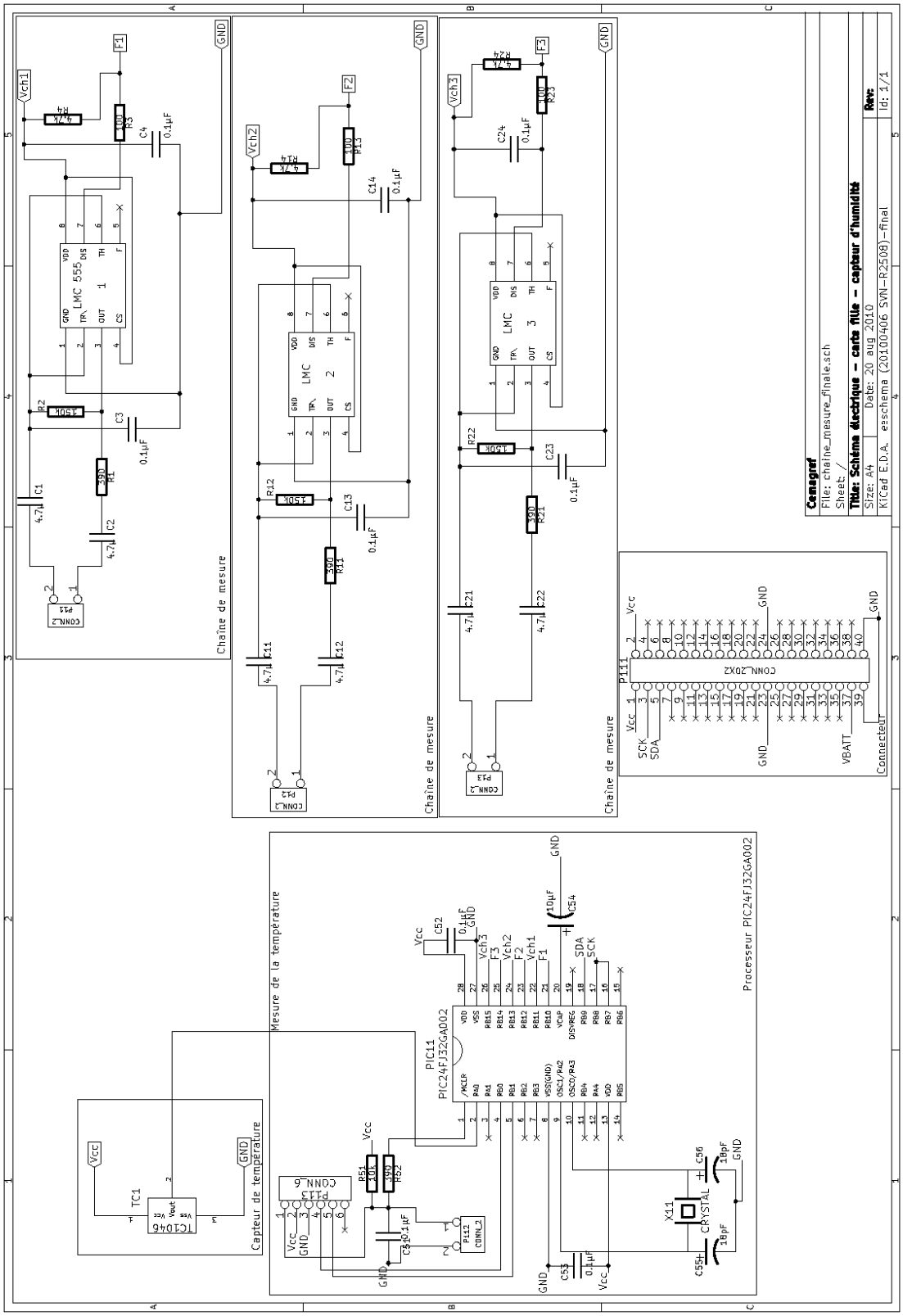


Schéma du PCB de la carte fille, vue de dessous.

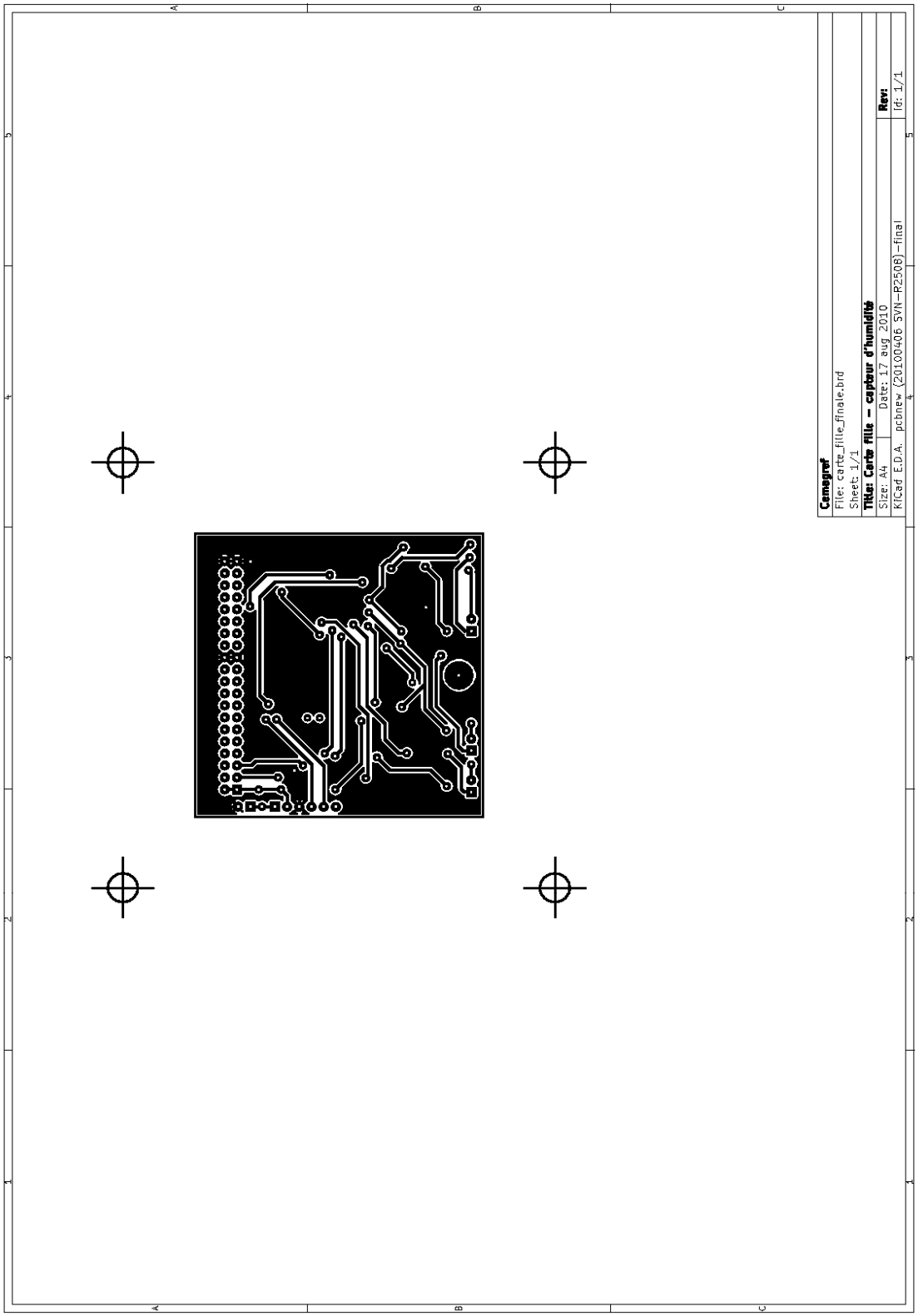
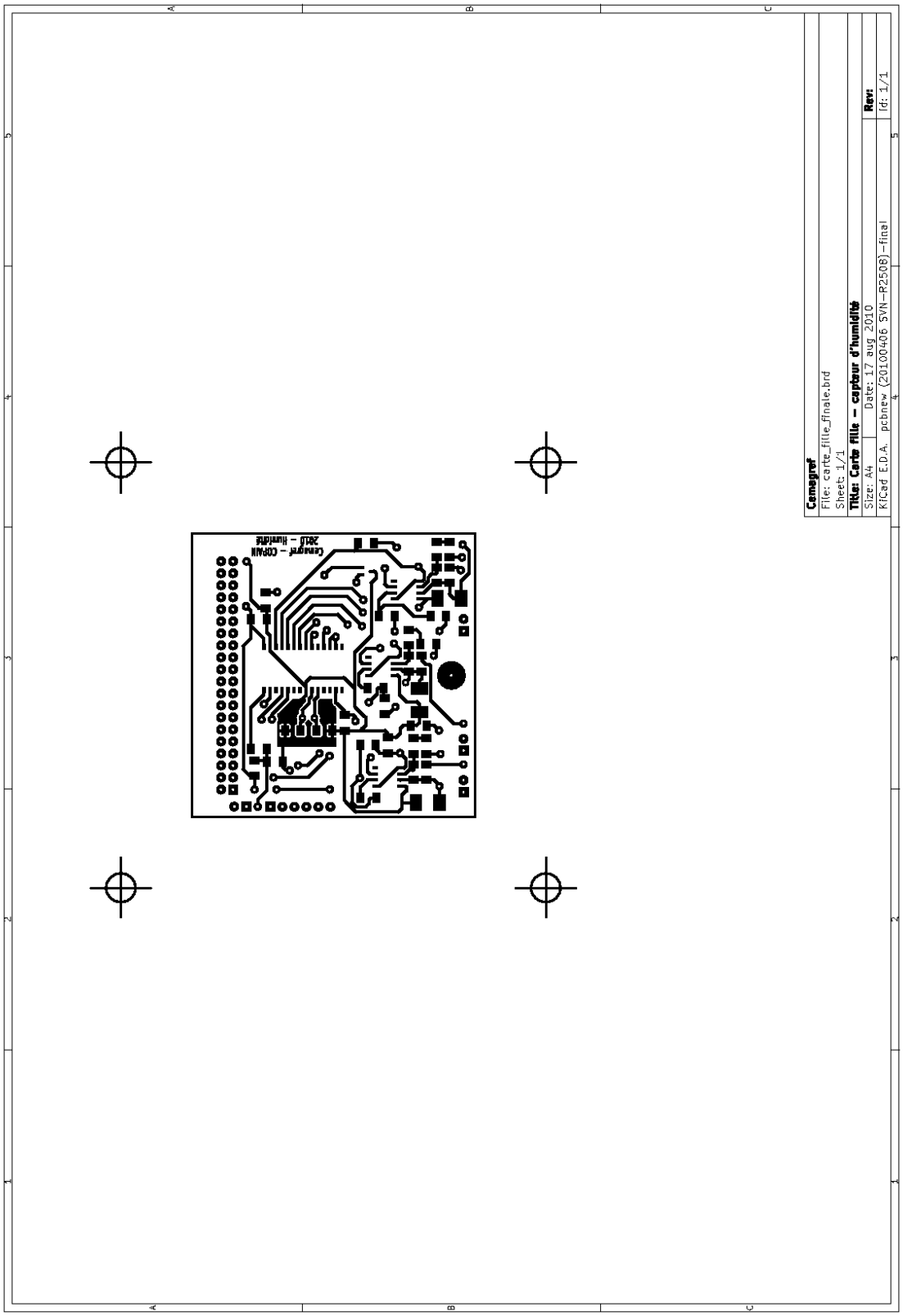
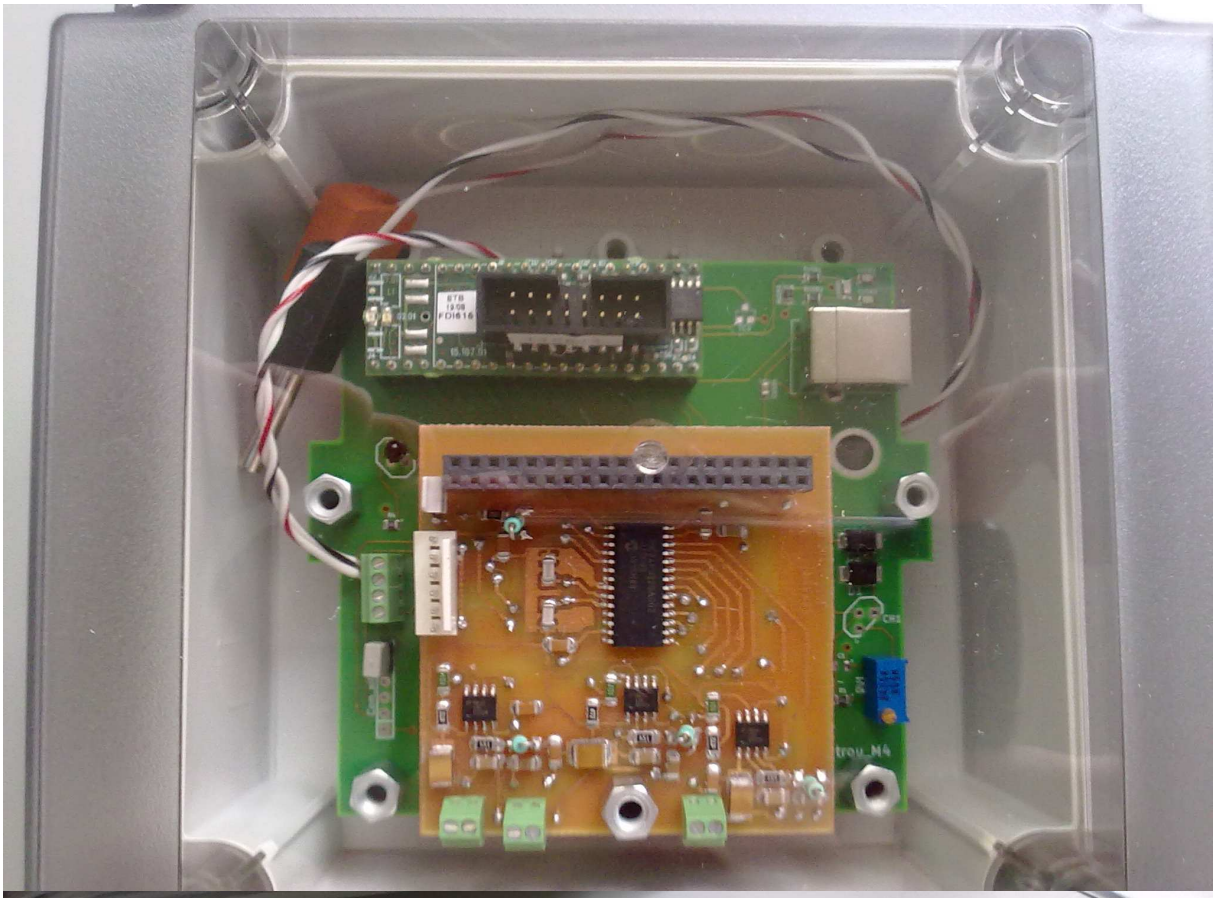
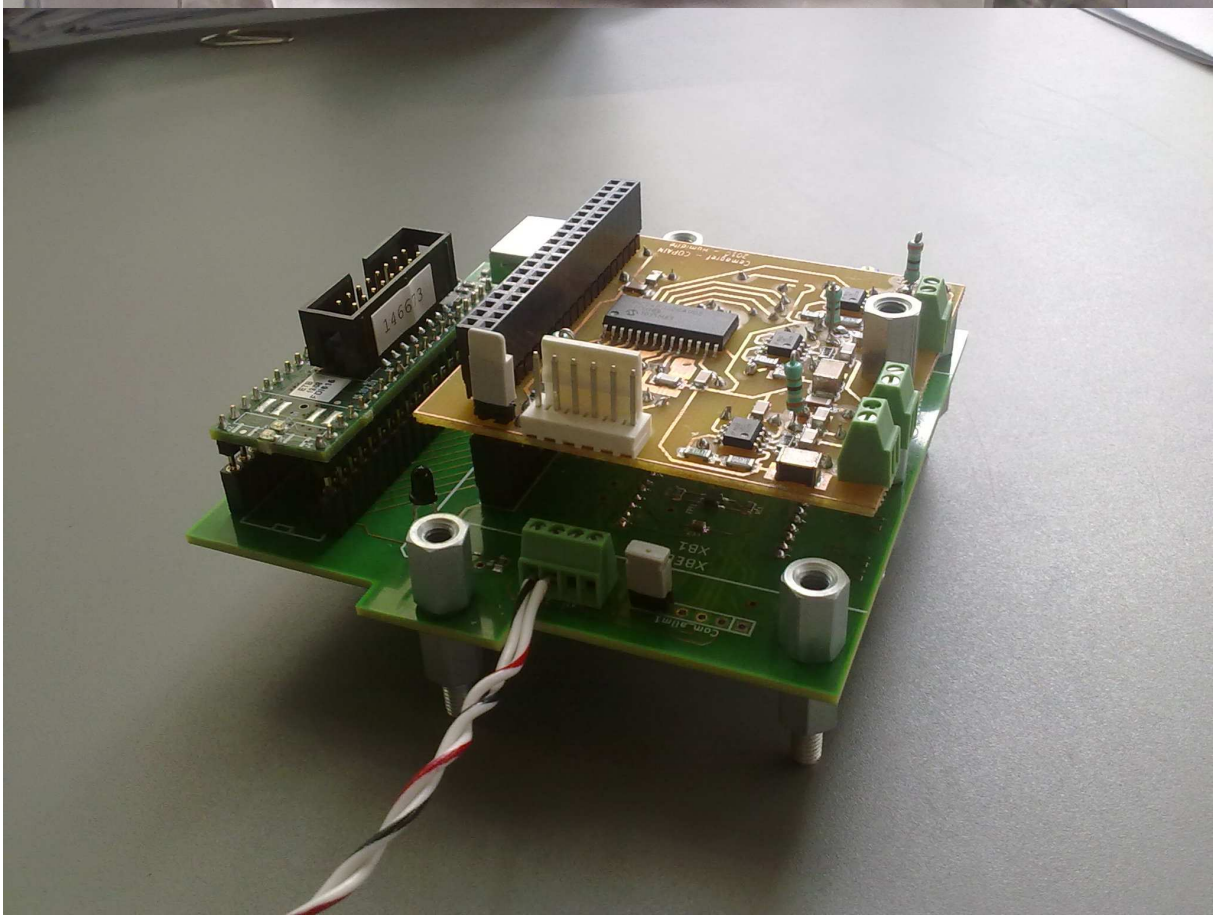


Schéma du PCB de la carte fille, vu de dessus.



Vues de la carte fille avec la carte mere.





Programme du protocole de communication.
Fonction gestion_envoi_changement.

```
void gestion_envoi_changement(Def_carte_t carte, char * buffer)
{
  int       new_id_capteur = 0,
            code_commande,
            id_capteur;
  float     frequence = 0.0;

  lecture_i2c(buffer);
  code_commande = char_to_int(buffer,0);
  lecture_i2c(buffer);
  id_capteur = char_to_int(buffer,0);

  if(code_commande == INIT)
  {
    initialisation(carte,buffer);
  }
  else if(code_commande == CHG_ID)
  {
    lecture_i2c(buffer);
    new_id_capteur = char_to_int(buffer,0);
    chg_id(new_id_capteur, &carte, id_capteur, buffer);
  }
  else if(code_commande == CHG_FREQ)
  {
    lecture_i2c(buffer);
    frequence = char_to_float(buffer,frequence);
    chg_freq(frequence, &carte, id_capteur, buffer);
  }
  else
  {
    envoi(carte,code_commande,id_capteur,buffer);
  }
}
```

Fonction initialisation.

```
void initialisation (Def_carte_t carte,char * buffer)
{
  int       res = 0;
 Def_capteur_t * pcour = NULL;

 pcour = (Def_capteur_t *) malloc(sizeof(Def_capteur_t));

 envoi_carte_id(carte,buffer);
 envoi_carte_nbcapteur(carte,buffer);
 pcour = carte.capteur;

 while(pcour != NULL)
 {
   envoi_capteur_id(carte,pcour->id,buffer);
   envoi_capteur_frequence(carte,pcour->id,buffer);
   envoi_capteur_format(carte,pcour->id,buffer);
   envoi_capteur_type(carte,pcour->id,buffer);
   envoi_capteur_donnee(carte,pcour->id,buffer);

   pcour = pcour->next;
 }
 free(pcour);
}
```

Fonction selection.

```
int selection(Def_carte_t * pcarte, int code_commande, int id_capteur, char * addr, char * buffer)
{
  int select =0;
  ecriture_i2c(addr);
  ecriture_i2c(code_commande);
  ecriture_i2c(id_capteur);

  if(id_capteur == 0)
  {
    /* cas où la commande ne s'adresse qu'à la carte fille */
    switch(code_commande)
    {
      case 0:
        /* toutes les informations à récupérer */
        select = recep_carte_total(pcarte,buffer);
      break;

      case 1:
        /* indique le nom de la carte */
        select = recep_carte_id(pcarte,buffer);
      break;

      case 2:
        /* indique le nombre de capteurs */
        select = recep_carte_nbcapteur(pcarte,buffer);
      break;

      default:
        /* indique toutes les données des capteurs */
        select = recep_capteur_total(pcarte,buffer);
      break;
    }
  }
  else
  {
    /* cas où seul les informations des capteurs sont demandées */
    switch(code_commande)
    {
      case 1:
        /* indique l'identifiant */
        select = recep_capteur_id(pcarte,id_capteur,buffer);
      break;

      case 2:
        /* indique la fréquence */
        select = recep_capteur_frequence(pcarte,id_capteur,buffer);
      break;

      case 3:
        /* indique le format de la donnée capteurs */
        select = recep_capteur_format(pcarte,id_capteur,buffer);
      break;

      case 4:
        /* indique le type d'une donnée */
        select = recep_capteur_type(pcarte,id_capteur,buffer);
      break;

      default:
        /* indique la donnée */
        select = recep_capteur_donnee(pcarte,id_capteur,buffer);
      break;
    }
  }
  return select;
}
```

Fonction envoi.

```
int envoi(Def_carte_t carte, int code_commande, int id_capteur, char * buffer)
{
  int select=0;

  if(id_capteur == 0)
  {
    /* cas où la commande ne s'adresse qu'à la carte fille */
    switch(code_commande)
    {
      case 0:
        /* toutes les informations à récupérer */
        envoi_carte_total(carte,buffer);
      break;

      case 1:
        /* indique le nom de la carte */
        envoi_carte_id(carte,buffer);
      break;

      case 2:
        /* indique le nombre de capteurs */
        envoi_carte_nbcapteur(carte,buffer);
      break;

      case 3:
        /* indique toutes les données des capteurs */
        envoi_capteur_total(carte,buffer);
      break;
    }
  }
  else
  {
    /* cas où seul les informations des capteurs sont demandées */
    switch(code_commande)
    {
      case 1:
        /* indique l'identifiant */
        envoi_capteur_id(carte,id_capteur,buffer);
      break;

      case 2:
        /* indique la fréquence */
        envoi_capteur_frequence(carte,id_capteur,buffer);
      break;

      case 3:
        /* indique le format de la donnée capteurs */
        envoi_capteur_format(carte,id_capteur,buffer);
      break;

      case 4:
        /* indique le type d'une donnée */
        envoi_capteur_type(carte,id_capteur,buffer);
      break;

      default:
        /* indique la donnée */
        envoi_capteur_donnee(carte,id_capteur,buffer);
      break;
    }
  }
  return select;
}
```