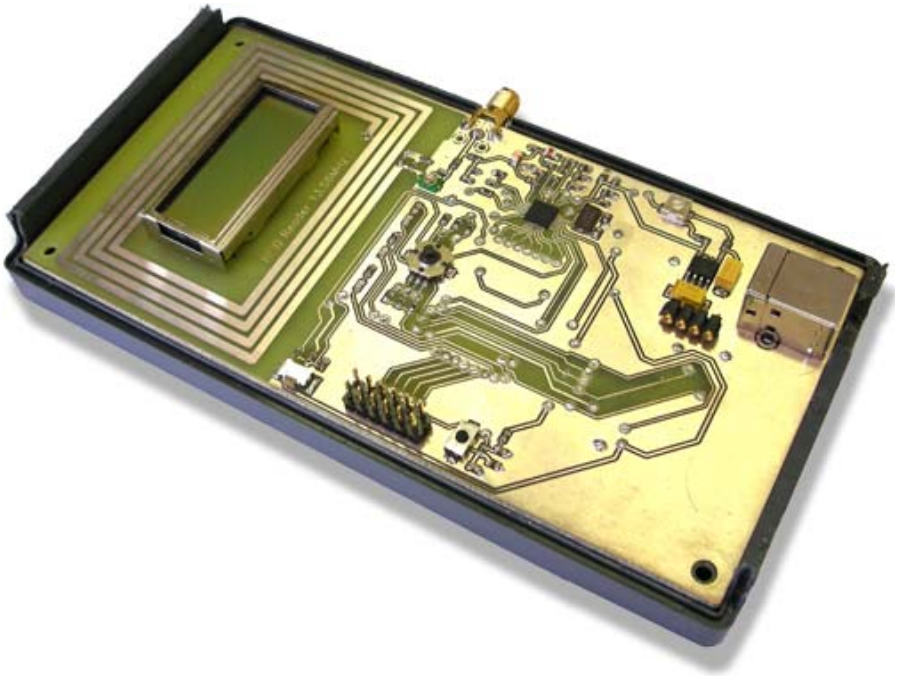

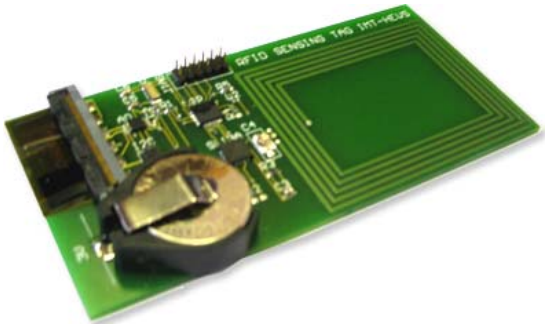
RFID SENSING TAG IMT-HCUS

Table des matières

Ce document a été rédigé comme rapport de projet de diplôme à la HES-SO // Valais dans la section Infotronics. Après une brève introduction aux systèmes RFID, il présente le développement d'un système RFID complet (lecteur et transpondeur) dans la bande de fréquence des 13,56MHz.

Le projet est réalisé en collaboration avec l'institue de microtechnique IMT de l'université de Neuchâtel. Le but du projet est de mettre en oeuvre un lecteur RFID portable à bas prix capable de lire un tag sur lequel se trouvent des capteurs sur substrat en polyimide conçus par l'IMT. L'issue du projet consiste à disposer d'une plateforme afin de démontrer les faisabilités d'un lecteur RFID portable ainsi que pour promouvoir les capteurs.

1 INTRODUCTION

L'identification par radiofréquence (RFID) est une technologie utilisée pour transmettre de l'information via les champs électromagnétiques. Les communications RFID sont effectuées très brièvement comparées à ses homologues que sont le Wifi ou le Bluetooth par exemple. La raison est que la RFID a été conçue pour récupérer uniquement une petite quantité d'information comme un identifiant. L'avantage de la RFID réside ainsi dans le fait qu'elle utilise des appareils capables de s'alimenter le temps de la communication avec le champ électromagnétique qui a été émit, sans utiliser une quelconque alimentation.

Les applications RFID se composent d'un lecteur et d'un ou plusieurs transpondeurs (tags). Le tag, bien que ce soit un mot anglais, est également employé par la littérature française pour désigner les appareils qui sont interrogés par le lecteur. « Tag » sera utilisé tout au long du document pour éviter toute ambiguïté avec la littérature.

Le tag peut être soit actif, soit passif, soit encore semi-actif. Un tag actif signifie qu'il est alimenté par une batterie pour gagner en distance de lecture, alors qu'un tag semi-actif emploie une batterie uniquement pour faire des logs des données. A l'inverse, un tag passif utilise uniquement l'énergie du champ électromagnétique pour fonctionner. Le choix de l'une ou l'autre technique se fait en fonction de l'application désirée.

La technologie RFID devient de plus en plus commune et les applications se diversifient continuellement. On retrouve la RFID tout particulièrement pour les applications de logistiques pour identifier les produits. Le secteur de l'industrie dispose également d'une large utilisation de la technologie RFID. La plate forme de démonstration conçue ici trouve ainsi son domaine de prédilection dans cette dernière catégorie.

Le document aborde tout d'abord une vu générale de l'application. La RFID est ensuite brièvement décrite à travers les différentes technologies et standards qui existent. La réalisation des appareils est ensuite expliquée en trois grands chapitres: architectures, réalisation électronique et programmation des logiciels. Le document se termine par un chapitre sur les améliorations et une conclusion.

2 APPLICATION GENERALE

La **figure 1** présente une vue d'ensemble du système RFID. Le lecteur au milieu est muni d'une batterie, d'une mémoire, d'un écran LCD, de trois indicateurs à led et d'un joystick pour relever et stocker les informations des capteurs. La connexion USB permet de recharger la batterie et de transférer les informations récupérées vers un ordinateur pour le rendu graphique et l'archivage.

Le tag est semi-actif. Cela signifie qu'il enregistre les données des capteurs à des intervalles réguliers grâce à la pile. Celle-ci est employée uniquement pour cela. Les données sont alors stockées sur une mémoire qui garde les informations même si la pile est à plat. La communication entre le lecteur et le tag se fait par induction magnétique.

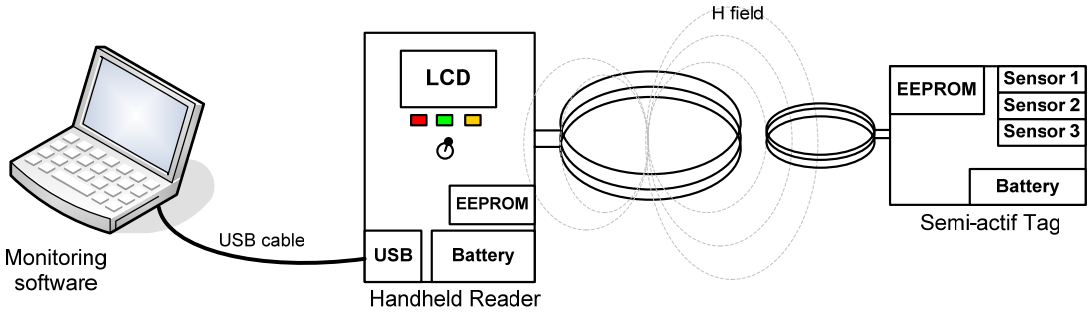

Figure 1 Application générale

La technologie RFID étant à l'heure actuelle assez vaste, il est nécessaire d'introduire quelques notions supplémentaires avant de pouvoir expliquer les choix et comprendre l'ambition du projet.

3 TECHNOLOGIES ACTUELLES

La section suivante présente un rapide tour d'horizon des technologies RFID. C'est utile pour savoir dans quelle catégorie se situe le système présenté ci-dessus mais aussi pour mieux cerner les avantages et les inconvénients.

La production de puces RFID opérant à la fréquence de 13,56MHz a subis une augmentation certaine comparé aux autres technologies qui sont à des fréquences ISM différentes. Le succès de cette fréquence est en grande partie due au bon compromis qu'elle offre entre vitesse de transfert, distance et sensibilité à l'environnement. En effet, la fréquence de 13,56MHz (HF) permet un transfert des données plus rapide comparé aux systèmes 125kHz (LHF) ce qui lui permet d'intégrer des fonction comme l'anticollision et le cryptage des données. Elle offre aussi une meilleure résistance aux perturbations et une meilleure pénétration de la matière comparée aux ultras hautes fréquences (UHF) telle que le 868MHz.

Les systèmes 125kHz et 13,56MHz opèrent à des distances très courtes. Ces distances ne dépassent habituellement pas les 1,5 mètres. Le 868MHz comme le 2,45GHz permettent quant à eux d'atteindre des distances jusqu'à 15 mètres.

La raison en est que la première catégorie de fréquences utilise le champ magnétique, avec comme technique l'induction magnétique (loi de Faraday) pour communiquer et récupérer l'énergie tandis que les systèmes à ultras haute fréquence se

servent de la propagation des ondes électromagnétiques avec comme technique la réflexion et le rayonnement de l'antenne pour communiquer et récupérer l'énergie.

Le dimensionnement des antennes des systèmes à ondes électromagnétique est beaucoup plus critique que celui des antennes inductives. En effet, la longueur de l'antenne électrique doit être proportionnelle à la longueur d'onde pour permettre à celle-ci d'être excitée par la fréquence, alors qu'il suffit aux antennes inductives d'être bouclées et d'avoir un champ variable en leur sein pour obtenir un courant induit dans le conducteur. Les systèmes inductifs sont ainsi beaucoup plus flexibles en ce qui concerne la taille des antennes. Toutes ces raisons font que le 13,56MHz est un très bon choix pour le projet actuel.

3.1 Les standards

L'accroissement de la technologie RFID a eu pour conséquence d'établir rapidement, de la part des organisations de standardisations, plusieurs règles garantissant l'interopérabilité entre les différents appareils (lecteurs et tags). C'est principalement le cas pour la norme *EPCGlobal* (*Electronic Product Code*) qui a très vite été inventée pour essayer d'instaurer un système d'identification unique pour chaque objet, de façon à simplifier les transactions commerciales entre les continents surtout.

D'autres normes ont été rédigées pour êtres utilisés dans des applications plus générales. Ce sont celles-ci qui nous intéressent ici. Ces dernières ont été établies par l'organisation international des standardisations (ISO) et la commission internationale d'électrotechnique (IEC).

Une quantité importante de ces normes ont vue le jour. Cette section n'a pas pour but de toutes les présenter bien entendu. Néanmoins une légère introduction à celles qui nous intéressent sera suffisante pour appréhender le sujet. Le **tableau 1** présente une vue d'ensemble des différentes normes qui sont applicables à la RFID.

Tableau 1 Les standards RFID

RFID Standards as applied to Frequency					
	Frequency Spectrum				
	LF 125/134.2 kHz	**HF 13.56 MHz +/- 7 kHz**	**HF 433 MHz**	**UHF 860-960MHz**	**UHF 2.45 GHz**
ISO	ISO 11784 ISO/IEC 18000-2A ISO/IEC 18000-2B	ISO/IEC 14443A ISO/IEC 14443B ISO/IEC 15693 ISO 18000-3	ISO 18000-7	ISO 18000-6A ISO 18000-6B ISO 18000-6C	ISO 18000-4 ISO/IEC 24730-2
EPCglobal				Class 0 Class 1 Class 1 Gen 2	

Celles qui nous intéressent se trouvent dans la colonne HF 13,56MHz. La norme *ISO/IEC 18000-x* a été définie pour décrire l'interface « air » entre les équipements. Elle a pour but de garantir l'interopérabilité entre les matériels en spécifiant les modes de fonctionnement et les fréquences de la porteuse entre autres. Le **tableau 2** résume ces

différentes fréquences. La norme *ISO/IEC 18000* fournie en quelque sorte la couche physique de l'application.

Tableau 2 RFID Air Interface Standards

18000-1	Generic Parameters
18000-2	< 135 kHz
18000-3	13.56 MHz
18000-4	2.45 GHz
18000-5	5.8 GHz
18000-6	860 to 960 MHz
18000-7	433 MHz

D'autres normes viennent compléter le 13,56MHz. Il s'agit des standards *ISO/IEC 14443* et *ISO/IEC 15693.* Le *14443* décrit le standard pour les cartes avec une détection très proche (jusqu'à 15cm) et *15693* pour les cartes avec une détection plus éloignée (jusqu'à 1,5m). Mais l'utilisation du *15693* n'implique pas nécessairement que la distance de lecture soit obligatoirement supérieure au 15 cm du *14443.* Il s'agit plutôt d'une convention. Le choix de l'une ou l'autre norme se fait en fonction de l'application et des performances recherchées.

Le choix d'une norme nécessite de la part du concepteur le respect des règles pour que la communication puisse avoir lieu avec n'importe qu'elle appareil de la même catégorie. La conception d'une interface RF avec des composants discrets reste possible mais risque d'être très fastidieuse. Il est plutôt conseillé d'utiliser des circuits intégrés qui sont compatibles avec standards les plus utilisés. La section suivante présente quelques circuits intégrés qui fonctionnent avec le standard ISO 15693 entre autres.

La documentation de la norme ISO 15693 doit être achetées. Mais il existe des résumés [1] qui peuvent être téléchargés sur Internet et qui décrivent suffisamment bien les grands principes du standard.

3.2 Circuits intégrés

Les lecteurs peuvent être conçue rapidement grâce d'une panoplie de puces qui réalisent complètement l'interface RF de l'appareil. Ces dernières contrôle automatiquement le format, les temps et tout ce qui est nécessaire pour être conforme aux règles dictées par les normes. Il suffit de les connecter à un microcontrôleur pour pouvoir les piloter et les rendre opérationnelles sur un lecteur.

Les tags sont eux généralement construits à l'aide d'une seule puce qui contient à la fois la section RF, digitale et la mémoire. Ces dernières n'ont plus qu'à être connectées à une antenne adaptée pour fonctionner. Elles ne permettent donc pas la modularité de pouvoir les connecter avec un microcontrôleur supplémentaire pour concevoir des tags plus sophistiqués. Elles sont généralement conçues dès le début pour une application particulière : ceci bien évidement pour des soucis d'économie d'énergie. Il existe malgré tout, mais cela reste pour le moment assez rare, des circuits intégrés pour tag qui disposent d'une interface pour communiquer avec un microcontrôleur. C'est ce type de puce qui a été utilisée pour le tag de ce projet.

3.2.1 Circuit intégré pour le lecteur

Comme circuit intégré pour un lecteur RFID à 13,56MHz, il existe la puce ***EM4094*** de *EM Microelectronic*. Cette dernière intègre une partie analogique qui permet de

restituer au microcontrôleur l'information modulée sous forme digital, et inversement bien sûre. Il est possible de la configurer pour les protocoles *ISO 15693* et *ISO 14443*.

Une autre puce beaucoup plus intéressante c'est la puce ***TRF7960*** de *Texas Instruments*. Celle-ci est capable de tout ce que fait la précédente mais elle intègre en plus de la partie analogique (Analog Front End) une partie digitale qui lui permet de gérer les trames transmises et reçues. Elle est donc capable d'ajouter automatiquement le SOF (Start Of Frame), le EOF (End Of Frame) et même de calculer le CRC pour le rajouter à la fin de la trame. Grâce à cela elle permet de rejeter les trames qui contiennent des erreurs et de générer une interruption capable d'avertir le microcontrôleur. Le code de ce dernier s'en trouve ainsi énormément simplifié.

Les possibilités supplémentaires qu'elle offre comparé à la puce précédente ont fait qu'elle a été préférée pour le projet. Son fonctionnement sera décrit au fur et à mesure dans les sections concernées car elle participe fortement au fonctionnement global de tout le lecteur (pour la gestion de l'énergie notamment).

Le circuit intégré choisi pour le projet est ***IDS-R13MP*** de *IDS Microchip* qui ressemble au *TRF7960*.

3.2.2 Circuit intégré pour le tag

IDS Microchip a eu raison de concevoir la puce modulaire ISO 15693 pour le tag cité précédemment car elle est très importante pour la concrétisation du projet. Son nom c'est ***IDS-SL13A***. Cette puce dispose en effet d'une interface SPI pour communiquer avec un microcontrôleur. Ce dernier pourra ainsi interfacer autant de capteurs qu'il en faut et communiquer avec le lecteur RFID via cette interface SPI.

La puce *IDS-SL13A* ne fonctionne que pour le standard ISO 15693. Il existe au sein de ce standard plusieurs configurations possibles (porteuses, débits, …) qui doivent être paramétrées avant chaque communication. La puce du tag détecte automatiquement quelle configuration a été choisie. C'est du côté du lecteur que les différentes options sont sélectionnées.

4 DESCRIPTION DES ARCHITECTURES

Le fonctionnement général des deux appareils (lecteur et tag) est expliqué à travers les architectures de ces derniers dans ce chapitre. La section suivante abordera quant à elle la schématique en détail et les explications sur les choix réalisés.

4.1 Schéma bloc du lecteur

Le schéma bloc de la **figure 2** présente globalement la composition du lecteur RFID. Tous les éléments principaux y sont représentés avec les directions des interactions. Les flèches en traits tillés représentent les alimentations et les flèches pleines les signaux.

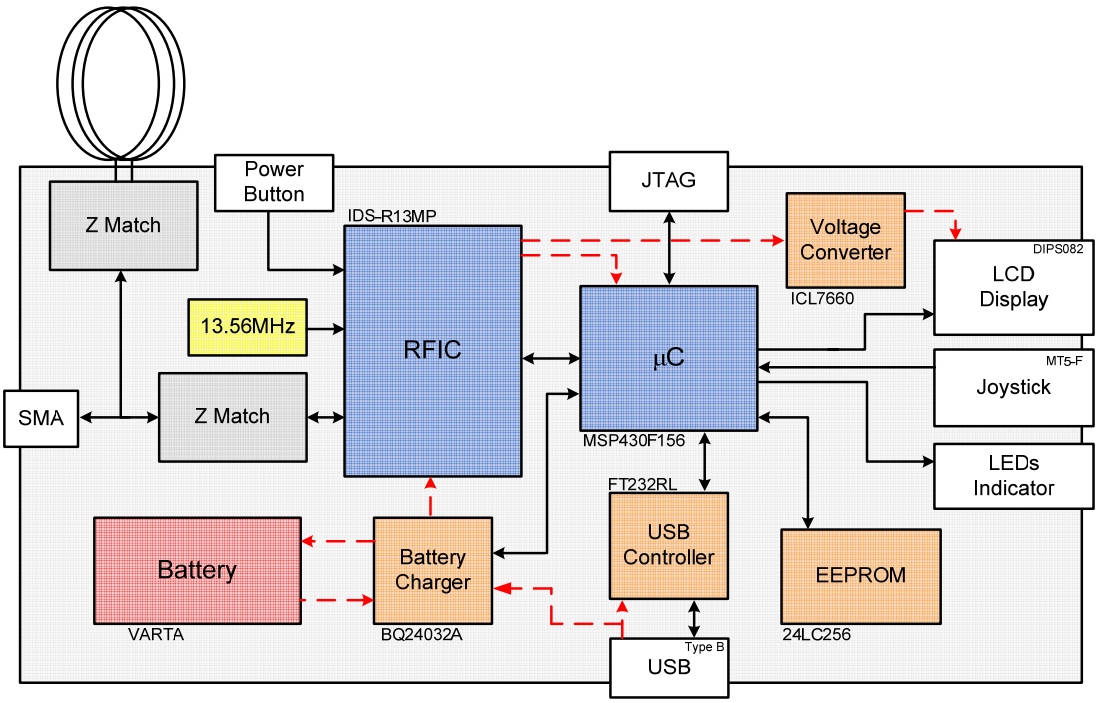

Figure 2 Architecture fonctionnelle du lecteur

Pour communiquer avec le tag, le lecteur utilise une interface RF (partie de gauche) qui est complètement réalisée, au moyen de quelques composants externes (blocs Z Match), par un circuit intégré qui est représenté sur le schéma par le bloc RFIC. Il s'agit de la puce ***IDS-R13MP*** qui a été présentée précédemment.

Le RFIC utilisé ici est très intéressant pour les systèmes embarqués car il dispose de deux régulateurs intégrés : un pour la partie digitale (<20mA) et un pour la partie analogique (>20mA). Ces régulateurs peuvent être activés et désactivés séparément. Cette petite particularité permet de réaliser des économies d'énergie considérable sur un système embarqué car on peut grâce à cela déconnecter toute la partie analogique tout en gardant le microcontrôleur (MCU) activée.

Le MCU utilisé est un ***MSP430F156***. Il possède suffisamment de ports de connexion et ne consomme que très peu de courant. Il est alimenté par le régulateur du RFIC. Il reçoit également l'horloge de la part de ce dernier. La communication entre ces deux composant se fait par une interface parallèle.

La mémoire externe ***24LC256*** de 256kB est prévue pour stocker les informations des capteurs de manière à les préserver même si la batterie est à plat.

Le connecteur USB Type B sert d'une part à communiquer avec le PC et d'autre part à recharger la batterie. L'USB a été choisie à la place d'un connecteur DB9 parce que ce dernier ne possède pas de broche V_{CC} pour recharger la batterie. Le contrôleur USB ***FT232RL*** est utilisé pour convertir le signal série UART du MCU en signaux différentiels (D_+, D_-) utilisés par l'interface USB. La puce s'occupe de toute l'implémentation du protocole USB.

Grâce au chargeur de batterie sophistiqué qu'est le ***BQ24032A***, l'appareil peut est être alimenté par la connexion USB en même temps que la batterie **Varta** de 660mAh se recharge.

Pour finir, un joystick cinq axes (***MT5-F***), trois LEDs basse consommation et un écran LCD 2x8 (***DIPS082***) sont utilisés pour permettre l'interactivité avec l'utilisateur. Le convertisseur de tension ***ICL7660*** est utilisé pour créer une tension négative nécessaire à l'écran LCD pour fonctionner s'il n'est pas alimenté avec une tension de 5V.

L'interface **JTAG** est utilisée pour programmer le microcontrôleur. Un bouton **POWER** pour allumer et éteindre le lecteur est connecté sur une entrée du RFIC qui est spécialement conçue pour cela. Le connecteur **SMA** sert dans un premier temps à régler l'adaptation, puis dans un deuxième temps il permet d'y connecter une autre antenne adaptée 50Ω.

4.2 Schéma bloc du tag

Le tag du système RFID a deux modes de fonctionnement : soit semi-actif soit complètement passif. La présence de la pile détermine dans lequel de ces modes le tag se trouve. Dans le mode semi-actif, le tag doit être capable d'enregistrer périodiquement les valeurs des capteurs, de les sauvegarder dans sa mémoire et de les restituer au lecteur lorsque celui-ci vient l'interroger. Il peut également fournir les valeurs instantanées des capteurs lorsque le lecteur le lui demande. Dans le mode passif, le tag s'alimente uniquement avec le champ RF et doit seulement fournir les valeurs instantanées des capteurs.

La puce RF n'étant pas capable de gérer plus d'un capteur branché à l'extérieur, un microcontrôleur (MCU) est nécessaire. Avec ce dernier il est possible de brancher autant de capteurs qu'il en est capable d'interfacer sur ses ports.

La **figure 3** présente la composition du tag RFID. La puce RF (RFIC) qui est la ***IDS-SL13A*** (présenté précédemment) réalise la partie RF du tag. Le MCU est un ***MSP430F2122***. Il est chargé de piloter les capteurs. Les capteurs utilisés pour le projet étant capacitifs, il est nécessaire d'utiliser un composant qui convertie la valeur de la capacité dans un format compréhensible par le MCU. Le composant ***AD7150*** réalise cette tâche. Il transforme la valeur de la capacité en une valeur digitale d'une résolution de 12 bits qui est ensuite transmise au MCU.

Si la batterie est présente, elle alimente en même temps le MCU et le RFIC. Dans le cas contraire, le champ RF alimente d'abord le RFIC qui peut ensuite fournir une tension régulée au MCU. Le MCU est alors capable d'alimenter le convertisseur quand bon lui semble depuis un de ses ports.

Une adaptation est nécessaire pour régler la fréquence de résonance du tag et pour ajuster le facteur de qualité. Le connecteur JTAG est utilisé pour programmer le MCU.

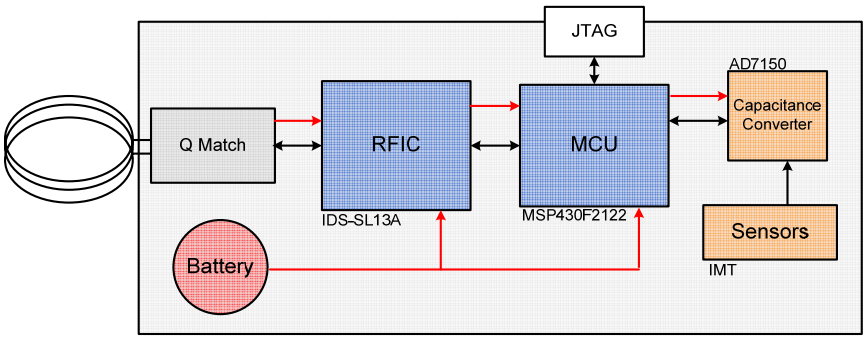

Figure 3 Architecture du tag

Le choix des composants ainsi que leur fonctionnement est décrit dans le chapitre suivant.

5 SCHEMATIQUE ET FONCTIONNEMENT

Cette section aborde en détail la réalisation du lecteur et du tag.

5.1 Réalisation du lecteur

Cette section décrit la réalisation électronique du lecteur.

5.1.1 Interface RF

Pour que le RFIC transmette à l'antenne sa puissance maximale, il faut que la sortie RF de la puce « voie » une impédance de 50Ω. On règle cette contrainte grâce à un étage d'adaptation d'impédance. L'effet du circuit d'adaptation doit être calculé en prenant également en considération les deux diviseurs capacitifs. Ces derniers sont utilisés dans le chemin inverse pour acheminer le signal RF reçu sur les deux entrées RF_IN du RFIC. Pour faire une adaptation optimale, le circuit est simulé avec *Ansoft Designer 3.5* (**Figure 4**).

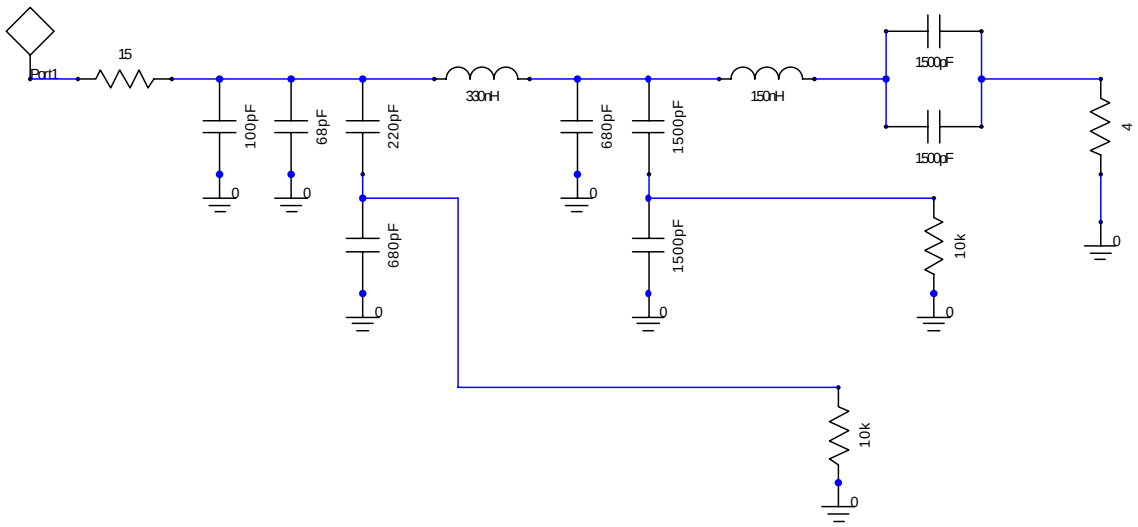

Figure 4 Circuit d'adaptation d'impédance avec les deux filtres pour les entrées AM et PM du RFIC qui sont à 10kΩ. La sortie RF est à 4Ω.

La sortie RF du RFIC a une résistance de 4Ω lorsqu'il est configuré en puissance maximale (Full Power). Cette résistance est de 8Ω lorsque le chip est configuré pour

transmettre seulement la moitié de la puissance (Half Power). L'adaptation utilisée ci-dessus est prévue pour être utilisée avec la puissance maximale. Si le chip est configuré en « Half Power », toute la puissance fournie en sortie ne sera donc pas transmise.

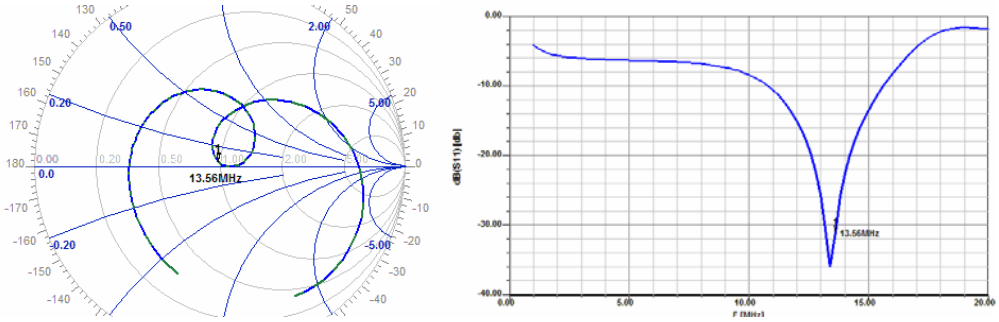

Figure 5 Simulation avec *Ansoft Designer 3.5* de l'étage d'adaptation d'impédance pour l'interface RF du RFIC.
Graphique de gauche : impédance sur l'abaque de Smith
Graphique de droite : paramètre S11 en dB

Les valeurs des composants utilisés dans la simulation sont choisies de telle manière qu'on peut directement les trouver dans des assortiments de composants. Les résultats de la simulation sur la **figure 5** montre que l'impédance mesurée à l'endroit ou est placé le port est bien de 50Ω. Le graphique de droite montre que le paramètre de réflexion S11 a une très bonne atténuation de 30dB à 13,56MHz.

Dimensionnement de la spirale

La conception de la spirale, c'est-à-dire l' « antenne » du lecteur (ce n'est pas vraiment une antenne), passe par une étape où il est nécessaire de prévoir l'emplacement des éléments constituants le lecteur. Plusieurs configurations ont été dessinées sur le papier avant d'en adopter la meilleure.

Pour gagner de la place, l'écran est placé au milieu de l'antenne. Il est tout à fait possible de concevoir ce genre de chose mais il faut faire attention, dans ce cas là, à la désadaptation : l'antenne n'aura pas les mêmes caractéristiques que s'il n'y avait pas l'écran.

Pour connaître l'influence de l'écran sur l'antenne, un PCB de la taille finale du lecteur a été réalisé avec seulement l'antenne, l'écran, le circuit d'adaptation et un connecteur SMA pour mesurer les caractéristiques (**figure 6**).

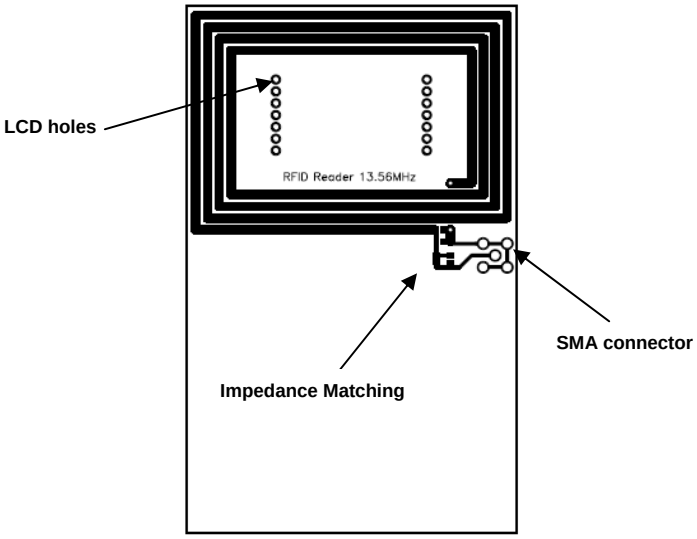

Figure 6 Mesure des caractéristiques et adaptation de l'antenne du lecteur

Avant de monter le circuit d'adaptation, la spirale est mesurée toute seule avec un analyseur réseau (Agilent E5071C) pour en connaître son impédance. Quatre mesures différentes ont été réalisées : avec et sans l'écran, dans la main ou non. Les quatre impédances mesurées sont présentées sur la **figure 7**.

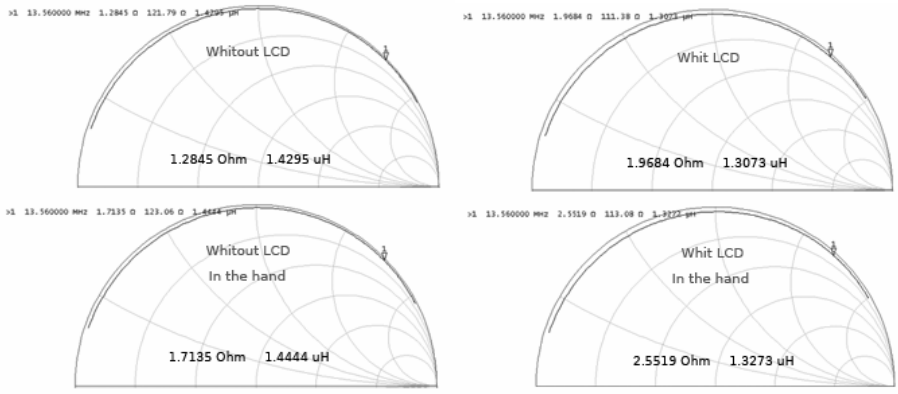

Figure 7 Impédance de l'antenne dans quatre configurations différentes

Le modèle de la spirale consiste en une inductance avec une résistance en série. Les différentes impédances mesurées sont réécrites en plus grand à l'intérieur des figures. On remarque premièrement que le fait de prendre dans les mains l'antenne influence uniquement la résistance série. Dans les deux cas elle augmente d'environ 0,5Ω. L'effet de l'écran augmente également la résistance d'environ 0.7Ω, mais par contre, il agit aussi sur la valeur de l'inductance. On voit que celle-ci diminue légèrement de 0.1µH.

Les valeurs qui nous intéressent se trouvent dans le dernier graphique : celui où l'appareil est pris dans les mains avec l'écran. Ces valeurs nous permettent de trouver les capacités qu'il faut utiliser pour adapter l'antenne à 50Ω et la résistance qu'il faut mettre en parallèle pour diminuer le facteur de qualité [2].

La facteur de qualité de l'antenne se calcul de la façon suivante :

$$Q = \frac{\omega L}{R_S} = \frac{2\pi \cdot 13{,}56MHz \cdot 1{,}33\mu H}{2{,}5\Omega} = 45$$

L'utilisation d'une sous porteuse de 423kHz (ISO 15693) impose que ce facteur de qualité ne dépasse pas une certaine valeur [3]. Cette limite est donnée par :

$$Q \le \frac{f_c}{2 \times BW} = \frac{13{,}56MHz}{2 \times 423kHz} \cong 16$$

Le facteur de qualité de la spirale étant trop élevé, on le diminue avec une résistance en parallèle. Le calcul de cette résistance est présenté ci-dessous.

On transforme d'abord la résistance série de l'inductance (c-à-d de l'antenne) en une résistance parallèle (**figure 8**). Ceci afin de faciliter les calculs.

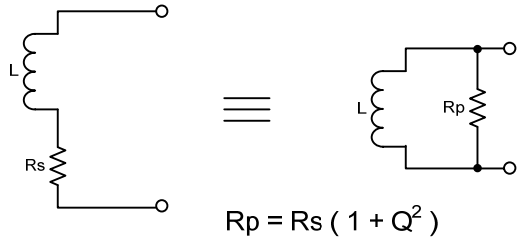

Figure 8 Transformation modèle série->parallèle

On applique l'équation :

$$R_p = 2{,}5\,\Omega \cdot (1 + 45^2) = 5\,k\Omega$$

on trouve une résistance parallèle de 5kΩ. Pour avoir un facteur de qualité de 16, la résistance en parallèle doit être :

$$Q = \frac{R_p}{\omega L} \Rightarrow R_p = Q \cdot \omega L = 16 \cdot 2\pi 13{,}56MHz \cdot 1{,}33\mu H = 1{,}813\,k\Omega$$

Pour obtenir cette résistance, on doit rajouter en parallèle une résistance de

$$1{,}813\,k\Omega = 5k\Omega \,//\, R \Rightarrow R = 2{,}84\,k\Omega$$

Ensuite pour adapter l'antenne à 50Ω on ajoute une capacité en parallèle et une capacité en série. Le circuit complet a été simulé avec *Ansoft Designer 3.5* (**figure 9**).

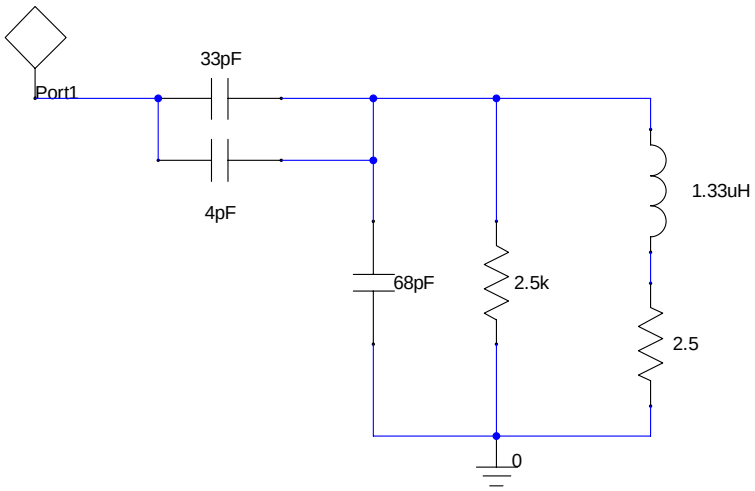

Figure 9 Simulation de l'adaptation d'impédance et réglage du facteur de qualité de la spirale du lecteur

L'analyse du circuit (**figure 10**) confirme que l'antenne est presque adaptée à 50Ω. Un condensateur variable sera dans tous les cas utilisé pour régler correctement l'adaptation. On voit également sur la l'image de droite que le cœfficient de réflexion S11 est parfaitement atténué à 13,56MHz.

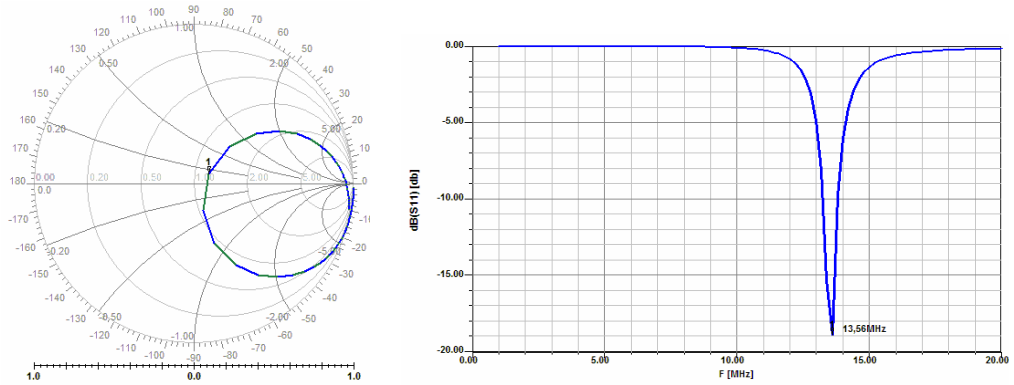

Figure 10 Simulation avec *Ansoft Designer 3.5* de l'étage d'adaptation de la spirale.

On peut également avec à la simulation vérifier que le facteur de qualité soit juste. Ceci se fait en désactivant les condensateurs. On peut alors constater (**figure 11)** que le facteur de qualité est parfaitement égale à 16 à une fréquence de 13,56MHz.

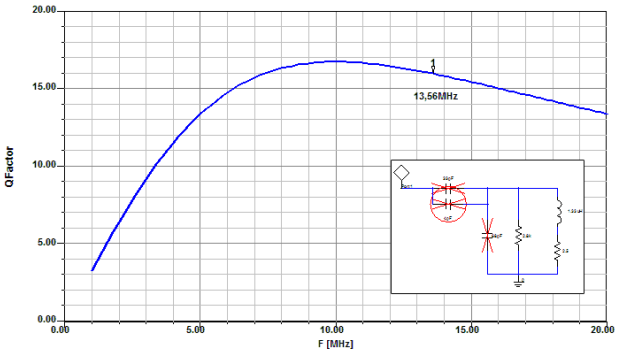

Figure 11 Simulation avec *Ansoft Designer 3.5* du facteur de qualité de la spirale du lecteur

5.1.2 Le microcontrôleur

Le MCU a comme rôle important sur le lecteur de : contrôler la puce RF, transmettre et recevoir les informations avec le tag, configurer le chargeur de batterie, piloter l'écran, les LEDs et le joystick, écrire sur la mémoire externe et pour finir communiquer avec le PC à travers l'interface USB.

Le MCU utilise pratiquement la totalité de ses six ports. Seul les modules internes UART et SPI sont utilisés. Le reste des pins sont utilisées comme simple entrés/sorties. La connexion de chaque élément avec le microcontrôleur est expliquée en détails ci-dessous.

Connexion avec la puce RF

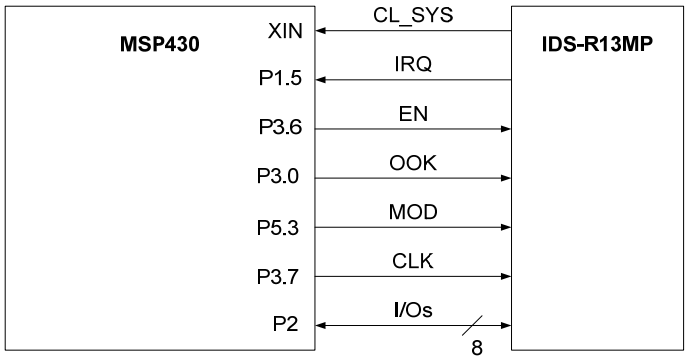

Figure 12 Connexion du microcontrôleur avec la puce RF

L'horloge du MSP430 est fournie par le *IDS-R13MP* à travers **CL_SYS** qui peut être ou non divisé par 2, 4 ou par 8. Le MSP430 possède des interruptions sur les ports 1 et 2. C'est pour cette raison que le signal **IRQ** est connecté sur une patte du port 1. La puce RF averti le MCU par des interruptions lorsqu'il y a eu des erreurs de transmission (CRC, parité, collision) mais aussi dans le cas où les données à envoyer dépassent la taille du tampon et ainsi avertir le MCU d'envoyer la suite dès que le FIFO est vide. Une interruption est également générée lorsque la transmission ou la réception est terminée. Pour savoir le type d'interruption, le MCU doit aller lire un registre particulier dans le *IDS-R13MP*.

Le signal **EN** est utilisé pour activer/désactiver le chip. Puisque l'alimentation du MCU est fournie par le RFIC, lorsque l'appareil est éteint, le MCU n'est pas capable de mettre un niveau haut à ce signal. Le signal EN2 sur le RFIC est utilisé pour cela. Il faut se reporter à la section consacrée à l'alimentation pour les explications sur la mise sous tension du lecteur.

Le signal **OOK** sert à dire au RFIC d'utiliser directement la modulation 100% sans passer le registre de configuration. Le signal **MOD** permet un accès direct à la modulation du signal RF sans passer par le système de protocole inclus dans le RFIC. Le SOF, le EOF et le CRC ne sont alors plus gérés. Le signal **CLK** est utilisé pour contrôler la transmission des données qui passent par les entrées/sorties **I/Os**.

Connexion avec le chargeur de batterie

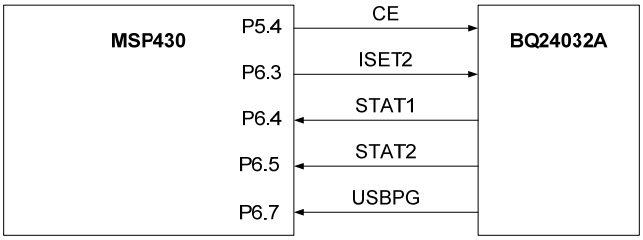

Figure 13 Connexion du microcontrôleur avec le chargeur de batterie

Le signal **CE** active/désactive de chargeur de batterie. Un niveau bas sur CE met en veille le chargeur pour qu'il consomme moins de courant. Le chip coupe ainsi l'arrivé du courant qui vient de l'USB mais continue à fournir le courant de la batterie sur sa sortie. Ceci permet de réduire le courant si le mode suspendu de l'interface USB est utilisé. Le signal **ISET2** est utilisé pour configurer le courant de charge de la batterie lorsque le câble USB est utilisé (soit 0.1A soit 0.5A). Un niveau bas indique 0.1A et le niveau haut 0.5A. Les signaux **STAT1** et **STAT2** indiquent l'état de charge de la batterie. Il faut se rendre à la partie consacrée au chargeur de batterie pour connaître la signification de ces deux signaux. **USBPG** (USB Power Good) sert à indiquer au MCU que le câble USB est branché : un niveau bas indique que le câble USB est branché.

Connexion avec le contrôleur USB et la mémoire

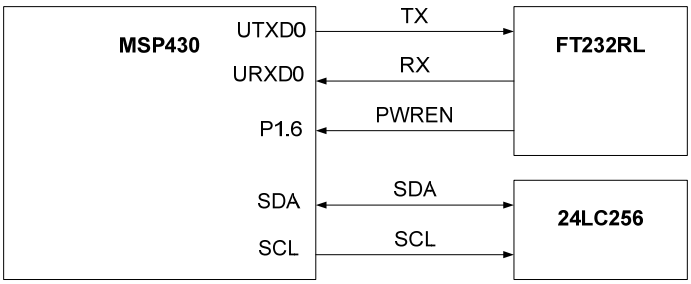

Figure 14 Connexion du microcontrôleur avec le contrôleur USB et la mémoire EEPROM

Le MCU communique avec le contrôleur USB, donc avec le PC, à travers son interface série UART. Les données sont envoyées et reçues sur les signaux **TX** et **RX**. Le signal **PWREN** est fourni par le contrôleur USB pour indiquer lorsque le bus passe en mode suspendu. Ce mode est utilisé pour économiser de l'énergie : s'il n'y a plus d'activité sur le bus pendant 3ms, le bus doit passer en mode suspendu. L'appareil doit alors réduire sa consommation de courant en dessous de 2,5mA. Le MCU est capable grâce à ce signal d'éteindre, ou mettre en mode veille les périphériques pour respecter les consignes du protocole.

Le MCU lit et écrit sur la mémoire EEPROM avec l'interface I^2C. Cette interface requière uniquement deux signaux : **SDA** pour les données et **SCL** pour l'horloge.

Connexion avec l'écran, le joystick et les LEDs

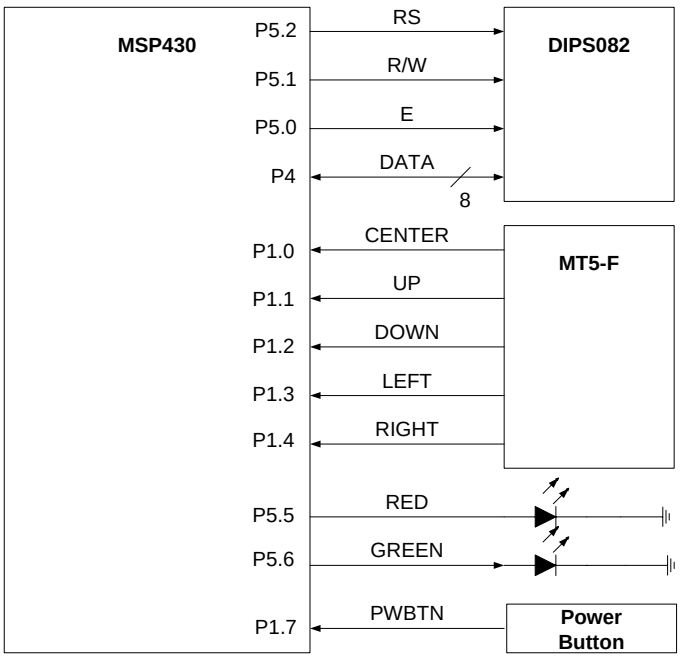

Figure 15 Connexion du microcontrôleur avec l'écran LCD, le joystick et les LEDs

L'écran LCD fait partie des écrans à interface HITACHI. Cela veut dire que tous les écrans possédant ce genre d'interface fonctionnent de la même manière. Il s'agit plus précisément de l'interface HD44780. De nombreuses explications peuvent être trouvées sur Internet à propos de son fonctionnement.

L'écran peut être branché de deux façon : soit avec 4 fils soit avec 8 fils de données. Le modèle complet a été utilisé ici. Trois signaux de contrôle en plus des données sont utilisés pour faire fonctionner l'écran. Le chapitre consacré à l'implémentation du logiciel explique plus en détail l'utilisation de chacun de ces signaux.

Un joystick à cinq axes a été choisi plutôt que de simples boutons. Ceci permet de rendre l'appareil plus compact. Les cinq signaux en provenance du joystick sont amenés sur le port 1 pour permettre les interruptions si cela s'avère nécessaire.

Deux diodes électroluminescentes de couleurs sont pilotées par le MCU pour avertir l'utilisateur des succès ou des échecs des opérations réalisées par le lecteur. Ces dernières sont un meilleur indicateur que l'écran LCD.

Le signal du bouton d'allumage est amené sur une entrée avec interruption du MCU pour permettre d'éteindre l'appareil également à l'aide ce bouton.

5.1.3 Chargeur de batterie

Avant de choisir le chargeur de batterie il fallait décider de la batterie. Comme le composant le plus important à alimenter sur le lecteur est la puce RF *IDS-R13MP* (puisque c'est elle qui demande la plus grande tension qui est de 5V au maximum) il fallait choisir, pour avoir la plus grande distance de lecture, une batterie en conséquence. Mais un autre critère plus important est de prendre en compte le fait que la

batterie doit pouvoir être rechargée à n'importe quelle moment. Entre les différents types de batterie qu'il existe, toutes ne possèdent pas la propriété adéquate pour satisfaire cette exigence. Les quatre grandes familles de batterie qu'il existe sur le marché sont :

- Nickel Cadmium (NiCd)
- Nickel Metal Hydride (NiMH)
- Lithium Ion (Li-ion)
- Lithium Polymer (LiPo)

Seul les batteries au lithium ne possèdent pas d'effet mémoire : elles peuvent être rechargées sans devoir attendre qu'elles soient complètement déchargées. Les batteries au nickel ne peuvent donc pas être utilisées sur le lecteur. De plus, celles au lithium ont une densité d'énergie 4 à 5 fois plus élevée que celles au nickel. La tension d'un élément Li-ion est de 3,6V est celle de la LiPo de 3,7V.

Lorsque le lecteur est transporté dans la main, on doit donc se contenter d'une alimentation de 3,7V fournie par la batterie lithium pour alimenter la puce RF ainsi que le lecteur. Mais cette tension monte jusqu'à 4,2V lorsque la batterie est complètement chargée. Elle redescend par contre très vite à 3,7V.

Le chargeur de batterie *BQ42032A* a été spécialement conçu pour recharger les batteries lithium avec une tension de 4,2V. Il a surtout été choisi par rapport à d'autres chargeurs parce qu'il est capable, grâce à un système appelé « *Dynamic Power-Path Management* » (DPPM), de recharger la batterie en même temps qu'il alimente le système lorsque le câble USB est branché. Cette particularité est un plus non négligeable pour le système embarqué dont il est question ici.

Le chargeur de batterie, malgré qu'il soit extrêmement petit, possède une multitude de ports qui sont utilisés pour configurer le chip. Certain sont configurés par le MCU d'autre emploie une résistance à l'extérieur pour réglé le courant (**figure 16**).

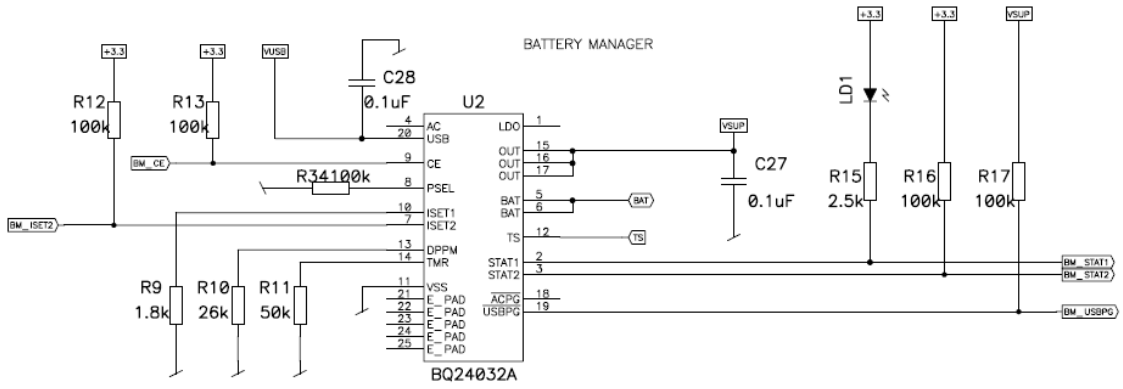

Figure 16 Schéma électrique du chargeur de batterie BQ24032A

Puisque l'adaptateur AC n'est pas utilisé sur le lecteur, le signal **PSEL** est configuré au niveau bas : il est tiré à la masse par une résistance. Ce signal indique la priorité entre le USB et le AC. Un niveau bas indique que l'USB est prioritaire.

Le courant de charge pour la batterie se règle avec les signaux **ISET1** et **ISET2**. ISET1 règle le courant pour une utilisation avec le AC alors que ISET2 règle le courant pour le USB. Le signal ISET2 est contrôlé par le MCU. Le courant qui provient du connecteur USB est soit 0,1A soit 0,5A. Un niveau bas sur ISET2 indique que la charge se fait avec 0,1A, avec un niveau haut elle est de 0,5A. En règle générale, ISET1 se règle pour un courant de charge supérieur à 0,5A. Ici, même si le AC n'est pas utilisé, on règle le courant à 580mA. La formule ci-dessous pour calculer la résistance $\mathbf{R_{SET}}$ (R9) est donnée dans la documentation du chargeur. $V_{(SET)}$ et $K_{(SET)}$ sont donnée dans la table de spécification du chargeur.

$$I_{O(BAT)} = \frac{V_{(SET)} \times K_{(SET)}}{R_{SET}} \Rightarrow R_{SET} = \frac{2{,}5V \times 425}{580mA} = 1{,}83k\Omega$$

La résistance $\mathbf{R_{DPPM}}$ (R10) règle le seuil de la tension à partir du quel le courant de charge de la batterie sera réduit pour compenser la baisse de la tension en sortie. Si le système à la sortie demande beaucoup trop de courant, est que le USB n'est pas capable de le fournir, le chargeur met également à contribution la batterie. La tension en dessous de laquelle la sortie ne doit pas descendre sous peine de réduire le courant de charge est réglée à 3V. $I_{(DPPM)}$ et SF sont donnée dans la table de spécification du chargeur.

$$V_{(DPPM-REG)} = I_{(DPPM)} \times R_{(DPPM)} \times SF \Rightarrow R_{(DPPM)} = \frac{3V}{100\mu A \times 1{,}150} = 26k\Omega$$

La dernière résistance de configuration $\mathbf{R_{TMR}}$ (R11) concerne le temps à partir du quel la batterie sera considérée comme morte si elle n'a pas été complètement rechargée. Ce temps est configuré à 5 heures.

$$t_{(CHR)} = K_{(TMR)} \times R_{(TMR)} \Rightarrow R_{(TMR)} = \frac{(60 \times 60 \times 5)s}{0{,}360\frac{s}{\Omega}} = 50k\Omega$$

Si la batterie possède une résistance qui varie avec la température, sa sortie peut être connecté sur l'entrée **TS** du chargeur. Ce dernier sera ainsi capable d'arrêter le chargement si la température augmente beaucoup trop.

La sortie **OUT** du chargeur de batterie est régulée à 4,4V. Il s'agit de l'alimentation générale du lecteur. C'est de là qu'est fourni tout le courant nécessaire au lecteur. La sortie peut fournir jusqu'à 4A. L'alimentation provient soit de la batterie soit du connecteur USB.

La signification des signaux **STAT1** et **STAT2** est reportée dans le **tableau 3**.

Tableau 3 Signification des signaux STAT1 et STAT2

CHARGE STATE	STAT1	STAT2
Precharge in progress	ON	ON
Fast charge in progress	ON	OFF
Charge done	OFF	ON
Charge suspend (temperature), timer fault, and sleep mode	OFF	OFF

Ce sont les deux des signaux « Open Drain ». Cela permet de connecter une led en pull-up sur le signal STAT1 pour indiquer l'état de charge de la batterie. En état de charge la led sera allumée puisque le transistor open-drain tirera à la masse. Dès que la charge sera terminée, la led s'éteindra. Des résistances en pull-up sont également ajoutées sur les signaux STAT2 et USBPG de manière à fournir un état haut lorsque les transistors en interne sont à OFF.

5.1.4 Le contrôleur USB

Le contrôleur USB convertie de signal UART en signal différentiel USB et possède une logique à l'intérieur qui implémente complètement le protocole USB. Le programmeur n'a donc à aucun moment besoin de se soucier des trames du protocole. Tout ce qu'il a besoin de faire c'est d'envoyer dans le tampon du module UART du MCU les données qu'il souhaite transmettre au PC. Le contrôleur USB se charge du reste.

La puce possède un oscillateur interne de 12MHz qui, grâce à un multiplicateur et un diviseur, permet d'obtenir toutes les fréquences habituellement utilisées par les interfaces USB, à savoir : 6, 12, 24 et 48MHz. Elle intègre également la résistance pull-up de 1,5kΩ sur le signal différentiel D- qui permet à l'appareil d'être détecté sur le bus même si celui-ci est éteint. Un régulateur de 3,3V est également incorporé dans la puce.

La mémoire EEPROM qui se trouve à l'intérieur de la puce contient la configuration. Les données qui y sont stockées sont par exemple le « Vendor ID », le « Product ID » mais aussi la description de l'appareil qui apparaît sur l'ordinateur lorsque l'on branche l'appareil sur le port USB. Toutes ces données se configurent avec un logiciel sur le PC (voir le chapitre consacré à la programmation).

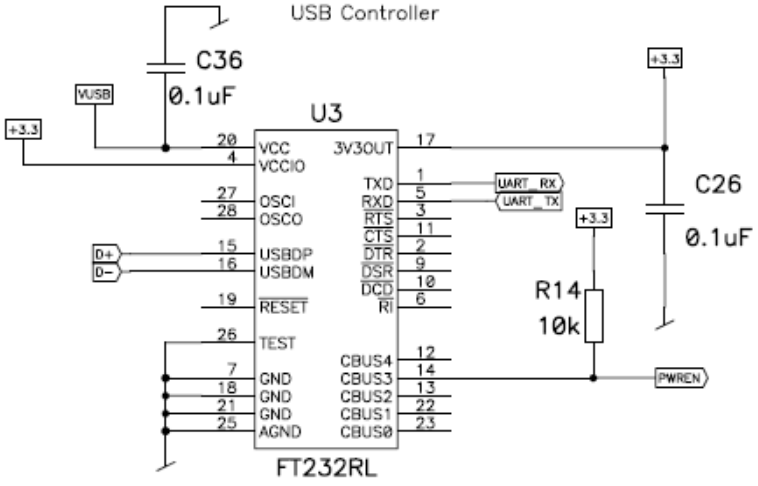

Figure 17 Schéma électrique du contrôleur USB FT232RL

La schématique du contrôleur USB est présentée sur la **figure 17**. Il est alimenté uniquement lorsque le câble USB est branché. Le régulateur de 3,3V est utilisé pour alimenter les résistances pull-up du circuit. Il est également utilisé pour spécifier le niveau des signaux qui proviennent du microcontrôleur (VCCIO).

5.1.5 Alimentation du lecteur

Le lecteur dispose au total de six régulateurs. Seulement cinq sont utilisés pour alimenter les différentes parties de l'appareil. La puce RF en possède trois, le chargeur de batterie deux et le contrôleur USB en possède un (**figure 18**). Cette partie décrit comment ces régulateurs sont utilisés et à quel moment. Savoir quel régulateur alimente quelle partie du lecteur et à quel moment fait partie des choix primordiaux pour une utilisation optimale de l'énergie consommée par le lecteur.

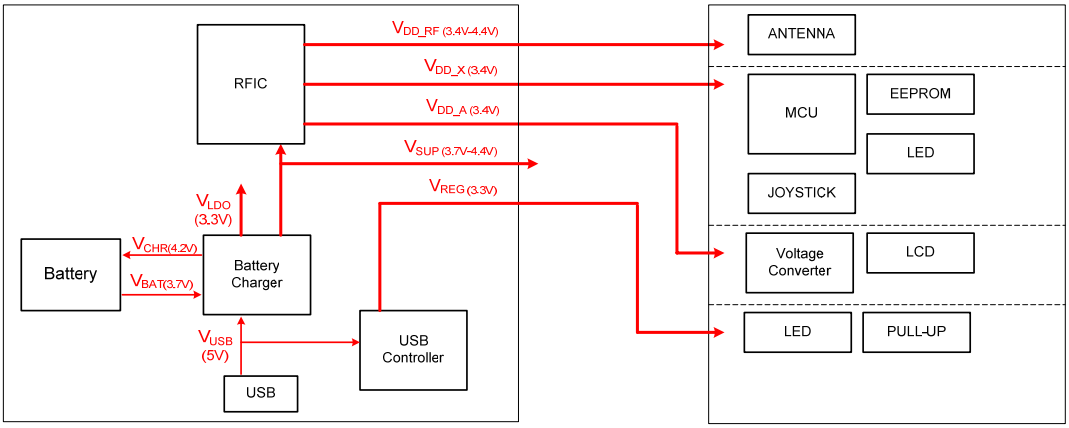

Figure 18 Régulateurs disponibles sur le lecteur avec les parties qui sont alimentés.

L'alimentation générale du lecteur provient soit du câble USB soit de la batterie. La tension qui provient de l'USB est de 5V. La tension de la batterie est de 3,7V. Ces deux tensions sont acheminées par le chargeur de batterie sur sa sortie qui régule la tension à 4,4V. C'est pourquoi V_{SUP} est soit la tension de la batterie 3,7V soit la tension du câble USB régulée à 4,4V. V_{SUP} devient ainsi l'alimentation générale du lecteur. Mais elle se charge uniquement d'alimenter le RFIC qui, grâce à ces nombreux régulateurs, alimente le reste de l'appareil.

Le régulateur V_{DD_RF} est utilisé pour réguler l'alimentation de l'antenne. Lorsque la batterie est utilisée la tension sur l'antenne est de 3,4V et 4,4V lorsque le câble USB est branché.

Le régulateur V_{DD_X} est sensé alimenter les parties digitales alors que le régulateur V_{DD_A} les parties analogiques. Ce choix a effectivement été adopté. Le joystick et la mémoire ont étaient branchés sur V_{DD_X} de manière à ce qu'ils puissent fonctionner avec le microcontrôleur lorsque le câble USB est branché et que le lecteur est éteint. Les scénarios de mise sous tension expliqués ci-dessous permettront de comprendre ces choix.

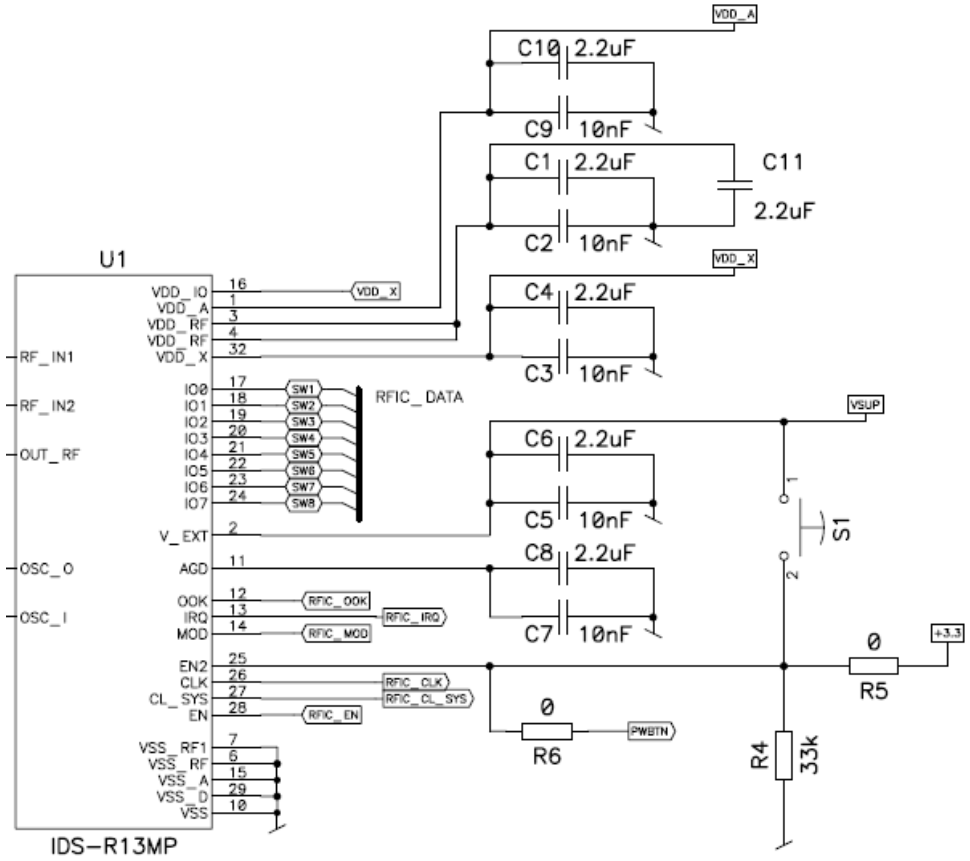

Figure 19 Les capacités de découplages de chaque régulateurs du chip RF et le circuit de démarrage du lecteur sont représentés sur cette figure.

Scénario d'allumage sans le câble USB

Le bouton impulsion S1 du circuit de la **figure 19** est le bouton pour allumer le lecteur. Lorsqu'il est appuyé, une impulsion arrive sur EN2. Cette impulsion a pour effet d'enclencher uniquement le régulateur V_{DD_X} et un oscillateur auxiliaire de 60kHz sur CL_SYS pour que le MCU puisse s'enclencher. A partir de ce moment, ce dernier à 100µs pour mettre EN au niveau haut et ainsi finir le processus de démarrage : enclenchement de tous les régulateurs et activation du quartz 13,56MHz sur CL_SYS. Dans le cas contraire le RFIC retourne s'endormir. Pour éteindre le lecteur toujours depuis ce bouton, l'impulsion du bouton est amenée avec R6 sur une entré du MCU capable d'interruption. Ce dernier peut alors remettre au niveau bas EN pour éteindre tous les régulateurs et ainsi éteindre le lecteur.

Scénario d'allumage avec le câble USB

Le câble USB se branche dans le lecteur soit pour recharger la batterie, soit pour récupérer les données stockées sur la mémoire. Dans le cas de la recharge, il faut que le MCU s'allume pour pouvoir lire les signaux STAT1 et STAT2 du chargeur de batterie et enregistrer sur la mémoire une défaillance de la batterie si cela se produit. C'est la raison pour laquelle la mémoire est également alimentée par V_{DD_X}. Lorsque l'utilisateur branche le câble USB, le lecteur s'allume automatiquement grâce à la résistance R5 qui amène un niveau haut sur EN2 (la tension +3.3V provient du contrôleur USB). V_{DD_X} et le 60kHz s'active alors ce qui donne le choix au MCU de finir le processus d'enclenchement avec le signal EN ou non. Le RFIC ne retourne pas

s'endormir comme précédemment puisque EN2 reste au niveau haut. Puisque l'utilisateur ne souhaite que recharger la batterie, le MCU détecte le câble USB grâce au signal USBPG et n'active pas le signal EN.

Si l'utilisateur souhaite récupérer les données de la mémoire, le logiciel sur le PC, grâce à une commande, pourra se charger d'activer le quartz 13,56MHz grâce au signal EN. Le MCU sera alors cadencer plus vite si un taux de transferts plus grand est nécessaire. Sinon, le MCU peut rester avec 60kHz et transmettre les données de la mémoire au PC.

Maintenant, si l'utilisateur souhaite pouvoir lire des tags, sans l'aide du logiciel, lorsque le câble USB est branché, il n'est pas capable d'allumer le lecteur avec le bouton Power On/Off puisque ce dernier est camouflé par la tension +3,3V (voir circuit de la figure 19). C'est la raison pour laquelle le joystick est alimenté avec V_{DD_X}. On peut de cette façon allumer complètement le lecteur grâce à ce dernier. On pourra régler le lecteur pour qu'il retourne en mode veille après un certain temps d'inactivité.

La schématique du lecteur se trouve en **annexe 1**.

5.2 Réalisation du tag

Cette section décrit la réalisation électronique du tag.

5.2.1 L'interface RF

Contrairement au lecteur, l'interface RF du tag est bien plus simple à réaliser. En effet, l'antenne est directement connectée à la puce RF. Un simple circuit d'ajustement est connecté en parallèle à cette antenne afin de respecter certaines exigences. La spirale et le circuit d'ajustement ont été étudiés à l'aide d'un logiciel de simulation avant d'être réalisés.

Les dimensions de la spirale peuvent être choisies au hasard tout en veillant à quelques consignes qui sont dictées par la théorie. En effet, cette dernière [4] nous renseigne que la taille de la spiral (la bobine) du tag doit être dans les proportions de celle du lecteur : le diamètre de la bobine du tag doit, pour le plus juste, être légèrement inférieur à celui de la bobine du lecteur. On assure ainsi premièrement qu'elle sera complètement « immergée » par le champ du lecteur lorsque celui-ci vient assez proche pour l'interroger et deuxièmement pour avoir un maximum de coefficient de couplage.

Le coefficient de couplage *k* a été introduit pour avoir une mesure du couplage qui soit indépendante de la géométrie des bobines, c'est-à-dire de leur inductance. Il est donné par l'équation ci-dessous [5] :

$$k = \frac{r_R^{\,2} \cdot r_T^{\,2} \cdot \cos(\theta)}{\sqrt{r_R \cdot r_T} \cdot \left(r_R^{\,2} + x^2\right)^{\frac{3}{2}}}$$

Le coefficient *k* varie entre 0 et 1. Un couplage de 100% (*k*=1) est atteint lorsque : la distance est de x=0, les bobines sont parfaitement parallèles et leur rayon est identique (r_R=r_T).

Les dimensions de la spirale du tag ont été choisies environ 30% inférieures aux dimensions de la spirale du lecteur. Une analyse des caractéristiques de la spirale a été réalisée à l'aide du logiciel ***SonnetLite***. Ce programme est une version gratuite de la suite *SonnetSuite* distribuée par Sonnet. Il permet de générer des spirales et de les analyser assez rapidement. La version est par contre limitée à 16Mo d'utilisation de la mémoire. L'antenne du lecteur n'a pas pu être analysée pour cette raison.

Deux types d'antenne ont été analysés. La taille extérieure (52x40cm) et le nombre de tours (5) sont restés identiques pour les deux antennes. Il n'y a que la largeur et l'espace entre les conducteurs qui ont été variés. La **figure 20** présente une vue 3D des deux antennes qui ont été analysées.

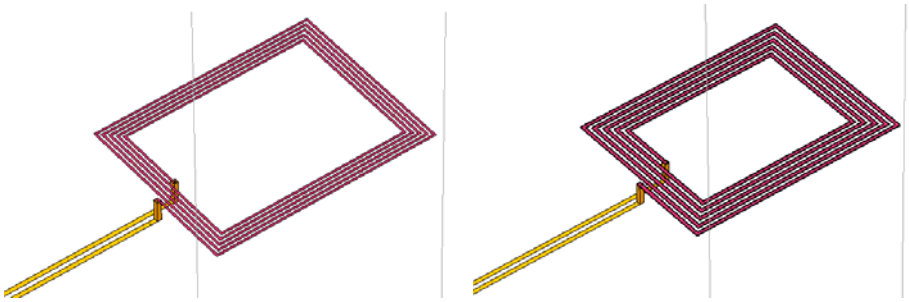

**Figure 20 Analyse à l'aide du logiciel *SonnetLite* des caractéristiques de deux type antennes.
Largeur et espacement des conducteurs :
gauche : 0,1mm, droite : 0,8mm**

La largeur et l'espace entre les conducteurs sont de 0,1mm pour l'antenne de gauche et 0,8mm pour celle de droite. Le paramètre observé est le facteur de qualité des antennes. Il a été calculé par le logiciel. Ils sont présentés dans les graphiques de la **figure 21**.

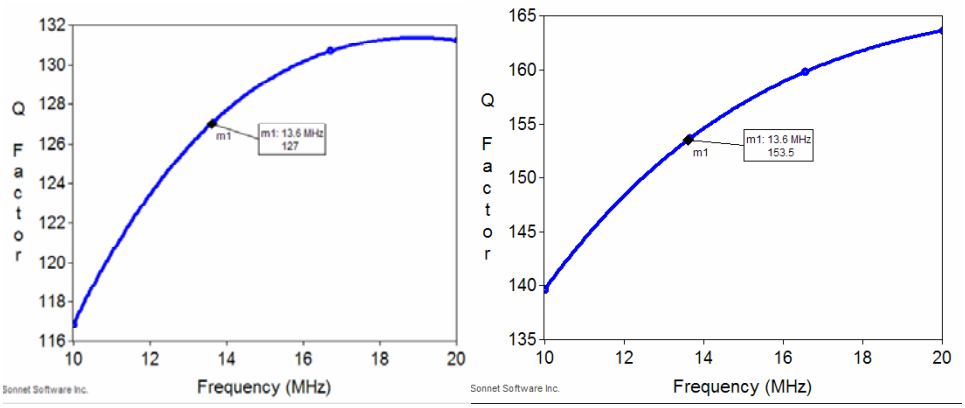

**Figure 21 Calculs du facteur de qualité des deux types d'antennes :
Gauche : Q=127, Droite : Q=154**

Le facteur de qualité de l'antenne avec le plus large conducteur est plus grand. Il est de 154 contre 127 pour celui avec le conducteur plus fin. Ce résultat s'explique facilement avec la formule du facteur de qualité d'une inductance :

$$Q = \frac{\omega L}{R_S}$$

La longueur des conducteurs restant la même, l'inductance L ne varie pas. Par contre, un conducteur plus large opposera moins de résistance au courant qui veut le traverser. La résistance série du conducteur plus large est donc plus petite, ce qui résulte en un facteur de qualité plus élevé.

Pour un circuit parallèle, plus le facteur de qualité est élevé, plus la tension à ses bornes sera grande lorsqu'il entre en résonance. Pour un circuit série, c'est le courant qui sera à son maximum. Dans notre cas, c'est la tension qui est recherchée : le tag est un circuit parallèle. C'est pourquoi l'antenne de droite a été choisie : son facteur de qualité est plus élevé.

Le logiciel *SonnetLite* est capable de fournir le modèle électrique de l'antenne simulée. Il donne la valeur de l'inductance de la spirale, sa résistance série ainsi que la capacité parasite qui représente le couplage qu'il existe entre les conducteurs. Il fourni ces données dans format PSPICE directement dans un fichier .lib. Pour la fréquence qui nous intéresse, ces valeurs sont :

```
*  Analysis frequencies: 13.3, 14.65 MHz
.subckt tag_spiral_3 1 GND
C_C1 1 GND 1.5161776405435pf
L_L1 1 2 1871.4719212329nh
R_RL1 2 GND 0.665842717189
.ends tag_spiral_3
```

Figure 22 Modèle PSPICE de l'antenne de droite

Connaissant la capacité interne de la puce RF, on est capable maintenant de concevoir complètement l'interface RF du tag. Pour facilité les calculs, comme pour l'antenne du lecteur, on transforme le modèle série de la spirale en modèle parallèle :

$$R_L = 0{,}7\Omega \cdot (1+154^2) \approx 17\,k\Omega$$

La résistance totale de l'antenne, pour avoir un facteur de qualité Q=16, doit être :

$$Q = \frac{R_P}{\omega L} \Rightarrow R_P = 16 \cdot 2\pi 13{,}56MHz \cdot 1870nH = 2{,}55k\Omega$$

Pour baisser la résistance de l'antenne, on rajoute une autre résistance en parallèle d'une valeur de :

$$2{,}55\,k\Omega = 17k\Omega \,//\, R_{(Q\,ajust)} \Rightarrow R_{(Q\,ajust)} \approx 3\,k\Omega$$

Pour avoir une antenne qui résonne, la capacité totale du circuit doit être de :

$$13{,}56MHz = \frac{1}{2\pi \cdot \sqrt{1870nH \cdot C_{tot}}} \Rightarrow C_{tot} = 73{,}2\,pF$$

Il faut donc rajouter une capacité en parallèle qui s'additionne aux autres capacités :

$$C_{tot} = C_{IC} + C_L + C_{(f\,ajust)} \Rightarrow C_{(f\,ajust)} = 73{,}2\,pF - 25\,pF - 1{,}5\,pF = 46{,}7\,pF$$

Le circuit complet est présenté à la **figure 23** :

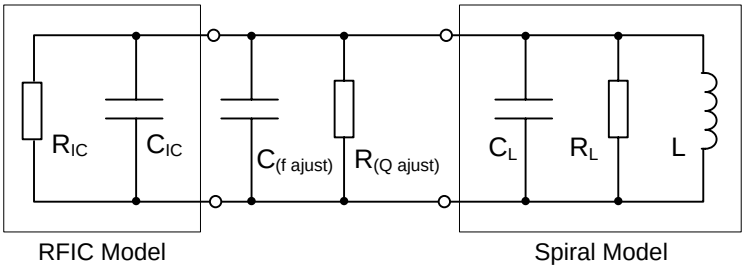

Figure 23 Circuit d'adaptation du tag: Capacité en parallèle pour régler la fréquence de résonance, résistance en parallèle pour régler le facteur de qualité

5.2.2 Le microcontrôleur

Le microcontrôleur sur le tag est utilisé pour interfacer les capteurs et transmettre les valeurs au lecteur. Il utilise les modules SPI et I^2C.

Connexion du microcontrôleur avec la puce RF

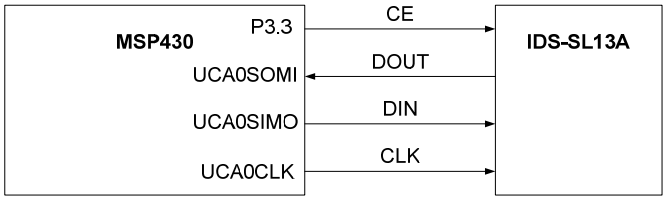

Figure 24 Connexion du microcontrôleur avec la puce IDS-SL13A

Le MCU communique avec le RFIC à travers une interface SPI. Cette interface est utilisée uniquement pour lire et écrire dans la mémoire interne de la puce *IDS-SL13A*.

Les trois signaux **DOUT**, **DIN** et **CLK** font partie de l'interface SPI. Le signal DOUT transporte les données qui proviennent du RFIC. Il est connecté sur l'entré SOMI (Slave Output Master Input) du module UCA0. Le signal DIN transporte les données en destination du RFIC. Il est branché sur la sortie SIMO (Slave Input Master Output) du MCU. Le MCU sera toujours le maître sur le bus. C'est toujours lui qui initie les conversations. C'est donc toujours lui qui génère l'horloge sur le signal CLK.

La mémoire EEPROM du RFIC est allumé/éteinte grâce au signal **CE**.

Connexion du microcontrôleur avec le convertisseur capacitif

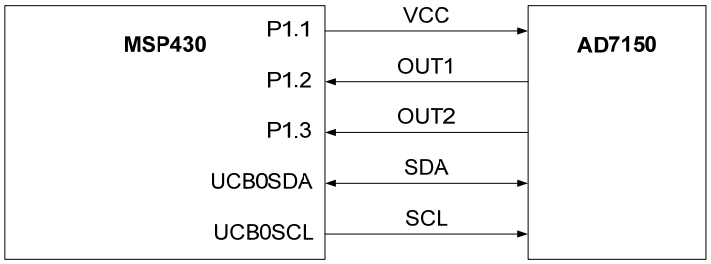

Figure 25 Connexion du microcontrôleur avec le convertisseur AD7150

La consommation du AD7150 n'étant pas supérieur à 150 µA et sa tension minimale de 2,7V, il est possible de l'alimenter directement depuis le MCU. Il sera ainsi allumé uniquement le temps de la conversion. Des économies d'énergie seront réalisées.

Les sorties **OUT1** et **OUT2** sont les signaux transmis par le convertisseur pour avertir que les seuils, qui ont été au préalable configurés, ont été dépassées par les valeurs des capacités qui ont été mesurées. Ces signaux sont amenés sur le ports 1 pour permettre les interruptions au cas ou.

La communication entre les deux éléments se fait à travers une interface I^2C. Le signal **SDA** transporte les données et le signal **SCL** l'horloge. Le bus sert à configurer les registres du convertisseur et à récupérer les valeurs digitalisées des capacités. De plus amples informations sur le convertisseur sont données dans la partie sur l'interfaçage des capteurs.

5.2.3 Circuit d'alimentation

Dans le mode semi-actif, le tag est alimenté par une pile bouton de 3V. Elle alimente en même temps la puce *IDS-SL13A* et le MSP430. La tension régulée à 3,4V qui est fournie par le RFIC sur la pin 3 est disponible uniquement lorsque le champ RF est présent. Le MCU ayant besoin de la tension de la pile pour faire les logs des capteurs, cette dernière est amenée par la diode D1 sur la pin 30. Cette diode empêche le courant du champ RF de remonter dans la pile lorsque le lecteur est présent. La diode D2 réalise le même travail pour ce qui est du courant de la pile. Les diodes sont du type Schottky pour éviter une trop grande chute de tension.

La résistance de 5,6MΩ a été ajoutée pour recueillir le léger courant inverse de la diode s'il s'avère nécessaire. C2 est la capacité de stockage qui est nécessaire au RFIC pour pouvoir écrire sur sa mémoire si la pile n'est pas présente. Elle sert également à stocker le courant pour le reste du circuit.

Si la pile est présente, la tension régulée (pin 3) du champ RF n'est plus nécessaire au MCU. Par contre, pour que celui-ci puisse voir que le lecteur vient interroger le tag, le niveau de cette tension est amené sur une entré du MCU qui est capable d'interruption (pin 32). La **figure 26** montre le circuit d'alimentation du tag.

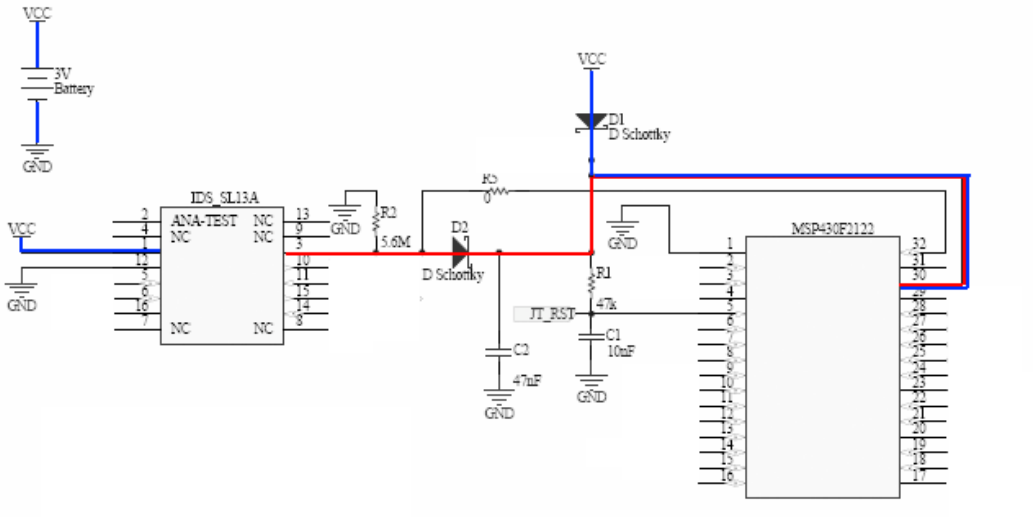

Figure 26 Circuit d'alimentations du tag

5.2.4 Interface de programmation JTAG

Le branchement de la prise JTAG sur le microcontrôleur du tag est abordé car une remarque importante doit être faite.

Le dessin de la **figure 27** représente le schéma général utilisé pour connecter la prise JTAG sur un MSP430. On trouve parfois, sur des schémas utilisant des MSP430, que la connexion TEST/VPP (pin 8 du connecter) n'est pas connectée. Celle-ci peut rester libre car la programmation du microcontrôleur peut très bien se faire sans elle. Mais ici, le MSP430 qui est utilisé est particulier. En effet, il fait partie de la famille x2xx. Les microcontrôleurs faisant partie de cette famille ont la possibilité d'être programmé non plus avec les 4 fils habituelles (TDO, TDI, TMS, TCK) mais simplement avec deux (TDO, TCK). Cette technique s'appelle le « Spy-Bi-Wire » et a été inventée pour gagner des connexions pour les applications qui utilisent beaucoup de pins du microcontrôleur. Ce mode n'a pas été utilisé pour le tag, mais il a été constaté, puisque le *MSP430F2122* permet cette fonctionnalité, que la connexion TEST/VPP est en réalité nécessaire même si le mode 4 fils est utilisé. Il en est de même pour la connexion RST qui n'est pas obligatoire pour les MSP430 qui ne sont capable que du mode 4 fils mais qui l'est pour les composants capable du mode 2 fils même si le mode 4 fils est utilisé.

Une autre remarque importante concerne la capacité pull-down (C1) du circuit. Il est précisé dans la documentation de TI [6] que sa valeur ne doit pas dépasser 2,2nF si le mode 4 fils est utilisé lorsque le MSP430 est capable du mode 2 fils : ce qui est donc le cas pour le tag.

Le connecteur JTAG contient deux prises pour les alimentations : VCC TOOL doit être connectée si l'appareil ne possède pas sa propre source : dans ce cas la c'est le debugger qui va fournir le courant pour programmer le MSP430. Dans le cas contraire, VCC TARGET peut être utilisé. Grâce à cette connexion, le debugger pourra lire la tension qui s'y trouve et adapter les signaux en fonction de la valeur de cette tension.

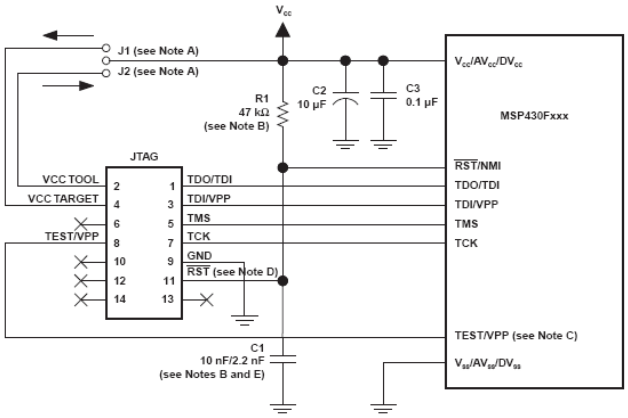

Figure 27 Schéma général de la connexion JTAG sur un MSP430

5.2.5 Interfaçage des capteurs

Cette partie aborde les explications sur la façon dont les capteurs sont connectés sur le tag. L'architecture du tag présentée au début et les connexions au microcontrôleur ci-dessus ont déjà introduit l'utilité du composant AD7150 sur lequel sont branchés les

capteurs. Cette partie donne quant à elle des explications plus approfondies sur le fonctionnement de ce composant ainsi que des données sur les capteurs utilisés.

Tous les capteurs qui sont utilisés ne sont pas capacitifs. Le dessin de la **figure 28** montre une vue rapide sur la façon dont ces derniers sont branchés sur le tag.

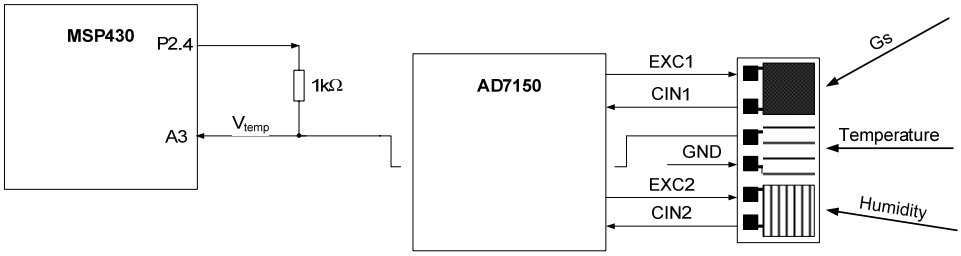

Figure 28 Connexion des trois capteurs : VOC, température et humidité

Seul les capteurs aux extrémités agissent comme une capacité. Le capteur au centre fonctionne comme une thermistance. La résistance augmente linéairement avec la température. Elle est branchée au moyen d'un pont diviseur sur une entrée ADC du microcontrôleur. La tension du pont est donnée depuis un port de ce dernier (P2.4). La connexion du capteur de température ne requiert pas beaucoup d'explication si ce n'est la configuration du module ADC au sein du microcontrôleur qui est décrite dans le chapitre consacré à l'implémentation des logiciels. Par contre, le convertisseur a un fonctionnement particulier qui est expliqué ici.

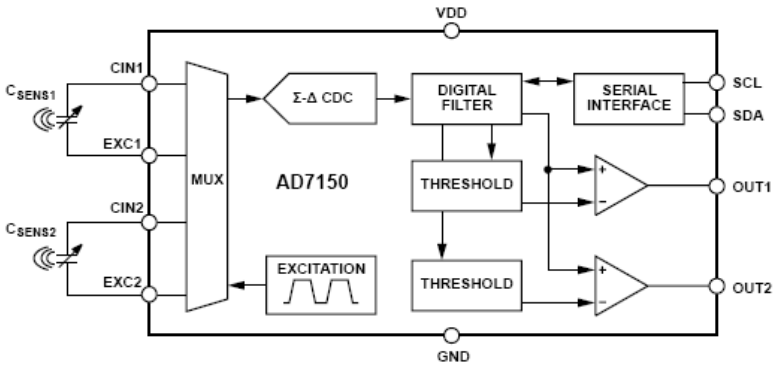

Figure 29 Bloc diagramme fonctionnel du AD7150

Le bloc diagramme de la **figure 29** donne une très bonne idée du fonctionnement du convertisseur. Le nom technique donnée au composant est CDC (capacitance-to-digital converter). Il convertie la capacité de la façon suivante : le signal d'excitation est appliqué à la capacité par la sortie EXC, les valeurs de capacités qui en résultent sont récupérées par l'entrée CIN qui sont échantillonnées en continue par le modulateur sigma-delta. Le flux de bits qui en sort passe par un filtre avant d'être sauvegardé sur 12 bits dans un registre. Les sorties OUT des comparateurs sont activées si les seuils sont dépassés. La valeur digitalisée peut également être récupérée à travers l'interface I^2C.

La gamme des capacités qui peuvent être interfacée est de 0-4pF. Néanmoins, des capacités plus grandes peuvent être acceptées grâce à un offset de 10pF qui peut être configurée à l'intérieur.

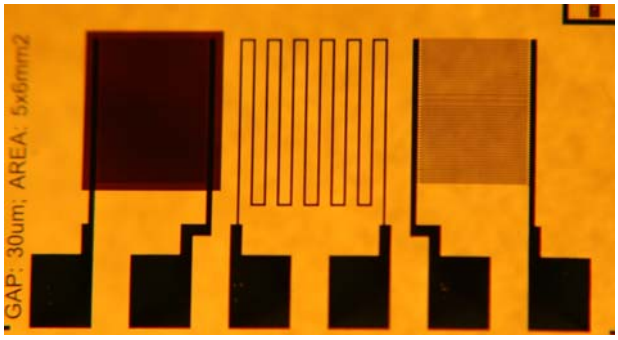

Figure 30 Image réel des capteurs

La schématique du tag se trouve en **annexe 2**.

6 ALGORITHMES ET PROGRAMMATION

Ce chapitre décrit la réalisation des logiciels du lecteur et du tag. Ils sont écrit en C et programmés à l'aide du logiciel ***Code Composer Essential 3***. C'est un programme gratuit fourni par *Texas Instrument* qui a spécialement été conçu (interface de *Eclipse*) pour programmer les microcontrôleurs MSP430. Les codes sont envoyés sur les cibles avec le debugger ***MSP-FET430UIF***. Il est branché sur le port USB du PC et programme les microcontrôleurs avec son interface JTAG.

Cette partie tente de décrire le plus clairement possible la programmation des logiciels qui tournent sur les microcontrôleurs. Les codes sources étant jugés suffisamment commentés, il serait superflu d'expliquer ici la programmation dans ces détails. Pour comprendre le fonctionnement des modules utilisés sur les microcontrôleurs comme le SPI, le I^2C ou encore les Timers, le lecteur est invité à se référer à la documentation qui donne des explications bien plus claires que ce qui pourrait se faire ici. Les documents qui sont utiles à posséder pour comprendre les codes sont les suivants :

- MSP430x1xx Family User's Guide
- MSP430x2xx Family User's Guide
- MSP430x15x Mixed Signal Microcontroller
- MSP430F21x2 Mixed Signal Microcontroller
- Datasheet *IDS-R13MP* (confidentiel) (alternative: *TRF7960*)
- Datasheet *IDS-SL13A* (confidentiel)

Un conseil pour ceux qui désirent appréhender la configuration des microcontrôleurs assez rapidement c'est de commencer par jeter un coup d'oeil à la description des registres qui se situent chaque fois à la fin des chapitres. C'est un bon moyen pour prendre connaissance des alternatives possibles sans avoir besoin de lire le fonctionnement du module au complet.

6.1 *Présentation générale*

Avant d'attaquer les explications sur ce qui a été mis en œuvre, il est coutume d'introduire une vision générale de ce qui doit être implémenté et quels sont les moyens pour le faire. L'objectif principal du système RFID est de récupérer au moyen du lecteur les valeurs des capteurs qui se trouvent le tag et de les afficher sur l'écran LCD. Le schéma bloc de la **figure 31** présente les flux des données (flèches) et les interfaces avec lesquelles cela sera réalisé.

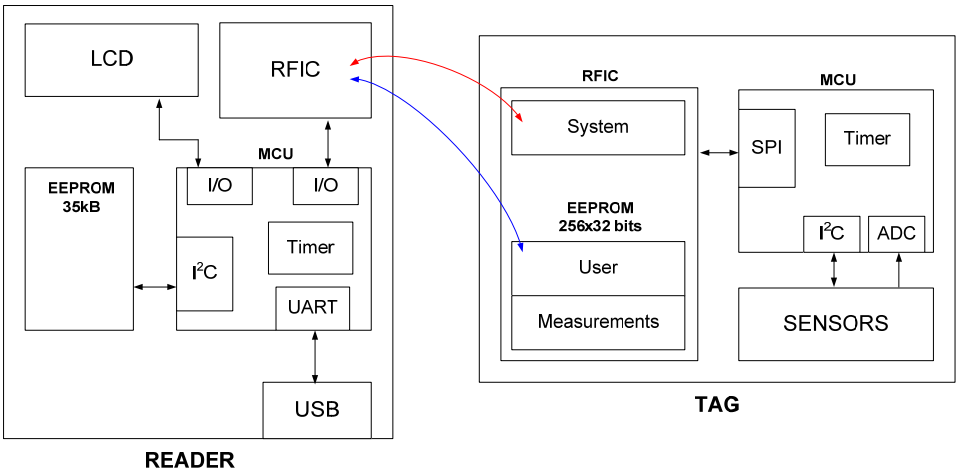

Figure 31 Flux des données des interfaces des microcontrôleurs

Le schéma est assez explicite pour ne pas re-décrire les interfaces qui sont utilisées (elles ont été introduites dans un précédent chapitre). L'idée du dessin est de se représenter de quelle façon les valeurs des capteurs sont acheminées jusqu'à l'écran et la mémoire de stockage du lecteur.

Le lecteur RFID ne peut communiquer avec le tag (c-à-d le microcontrôleur) que par l'intermédiaire de la mémoire EEPROM qui se trouve à l'intérieur de la puce RF. Cette mémoire c'est en faite séparée en deux mémoires différentes comme on peut le voir sur la figure. Il y a une mémoire « System » qui ne peut être accédée depuis le lecteur qu'avec des commandes spécifiques (pour la configuration et les actions particulières). L'autre mémoire, appelée « User & Measurements », d'une taille de 256 blocks peut quant à elle être lue et écrite par le lecteur comme une mémoire normale.

La taille de la section « User » peut être configurée avec une commande depuis le lecteur qui va enregistrer cette information dans la mémoire « System ». Le reste de la mémoire sera utilisé par la section « Measurements ». Cette option permet d'informer la puce RF à quel endroit elle doit commencer à enregistrer, lors du 'data logging', les mesures de son capteur interne. Mais ce mode n'est pas activé puisque le capteur interne n'est pas utilisé. Par contre, les données envoyées par le lecteur pour ce mode dans la mémoire « System » comme le délai entre chaque mesure et l'heure du début pourra être récupéré par le microcontrôleur. En effet, ce dernier, grâce à l'interface SPI, a un accès complet et sans restriction sur les deux mémoires. Il pourra ainsi réaliser des logs des capteurs externes en bénéficiant des commandes spécifiques envoyées par le lecteur. Mais le fonctionnement du tag ne se limite pas à cela. Il doit en priorité être capable de fournir au lecteur les valeurs instantanées des capteurs.

La puce RF possède une commande spécifique (non ISO15693) pour récupérer, depuis le lecteur, la valeur du capteur de température interne. Il s'agit de GetTemperature. Il faut pouvoir faire de même pour les capteurs externes. L'interface SPI du microcontrôleur n'étant faite que pour lire et écrire sur la mémoire EEPROM (et uniquement pour cela), la solution c'est d'utiliser cette mémoire comme tampon pour enregistrer les valeurs des capteurs et attendre que le lecteur vienne les récupérer. Puisque la mémoire « System » est complètement réservée, il faut utiliser la section « User ». Elle jouera ainsi très bien son rôle. La **figure 32** ci-dessous montre la structure de la mémoire pour enregistrer les valeurs d'un ou plusieurs capteurs.

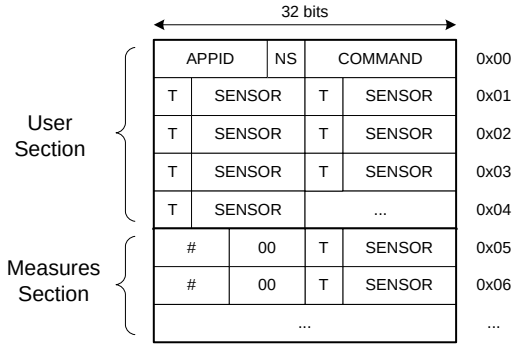

Figure 32 Structure de la mémoire du tag pour enregistrer les valeurs des capteurs et pour recevoir des commandes

La section « User » est donc utilisée pour enregistrer les valeurs instantanées des capteurs. Elles sont lues par le lecteur et affichées sur l'écran. La section « Measurements » est utilisée pour enregistrer périodiquement les valeurs des capteurs pendant que le lecteur n'est pas là. Le tableau ci-dessous décrit la structure de la mémoire.

APPID (Application Identifiant)	Identifiant de l'application (12 bits). Cette valeur est lue par le lecteur avant chaque opération sur le tag. Elle est utilisée, en plus du AFI, pour différentier le tag des autres applications potentielles. En effet, s'il n'était pas présent, un quelconque tag disposant du même AFI peut très bien être accepté et lu par lecteur. Si un tel cas se produit, le lecteur peut afficher des informations erronées sur l'écran.
NS (Number of Sensors)	Nombre de capteurs sur le tag (4 bits). Cette information est utilisé par le lecteur pour savoir combien de lignes il doit parcourir dans la section « User » pour récupérer les valeurs instantanés des capteurs.
COMMAND	Commande envoyée au tag (16 bits). Le tag reçoit des commandes par l'intermédiaire de ce champ. Mais le lecteur peut également lire ce champ pour savoir si le tag à répondu.
T (Type)	Type de capteurs (4 bits). Ce champ est enregistré à coté de chaque mesure pour indiquer au lecteur à quel capteur correspond la valeur.
SENSOR (Value of sensor)	Valeur du capteur (12 bits). Les valeurs des capteurs disposent de 12 bits pour être enregistrées dans la mémoire.
# (Number)	Numéro de l'enregistrement (8 bits). Le numéro est utilisé lors du 'data logging'. Il s'incrémente à chaque fois que le microcontrôleur enregistre les valeurs des capteurs. Il sera utilisé par le logiciel du PC pour retrouver, avec l'heure le départ (Start time) et le délai (Delay time) à quel moment les mesures ont été prises.
00	L'espace non utilisé est rempli avec des 0.

Les sections suivantes décrivent l'utilisation des mémoires du lecteur et du tag pour récupérer les valeurs instantanées et les valeurs enregistrées périodiquement.

6.1.1 Récupération des valeurs instantanées

Le processus de lecture par le lecteur RFID des capteurs sur le tag est décrit ici. Le dessin de la **figure 33** présente les différentes étapes qui ont été implémentées pour récupérer les valeurs instantanées.

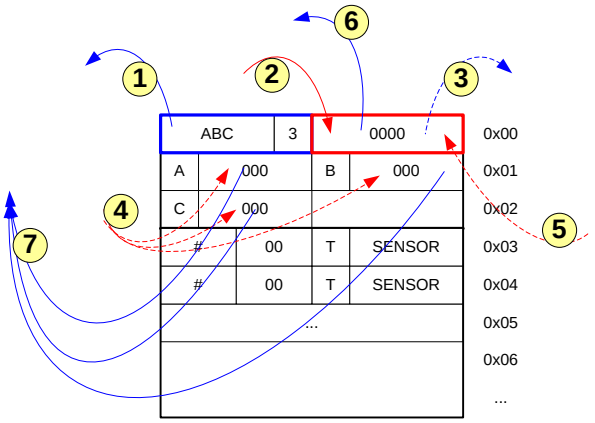

Figure 33 Interrogation du tag pour calculer les valeurs des capteurs, les enregistrer dans la mémoire et les récupérer depuis le lecteur

1. Le lecteur RFID récupère la première ligne de la mémoire, vérifie l'identifiant de l'application et récupère le nombre de capteurs.

2. Pour connaître les valeurs des capteurs, il envoie une commande en réécrivant le premier block de la mémoire dont il a modifié le champ de la commande (met un bit à 1).

3. Le microcontrôleur du tag voit la modification et comprend qu'il doit réaliser la lecture des différents capteurs.

4. Une fois qu'il a toutes les valeurs, il les écrits immédiatement à la suite du premier block.

5. Il remet le bit modifié précédemment à 0 pour dire qu'il a fini d'enregistrer les valeurs.

6. Le lecteur voit que le tag a remis à 0 le bit.

7. Il récupère les valeurs block après block.

A la fin de ces opérations le lecteur n'a plus qu'à afficher une liste avec les valeurs des capteurs. Mais en plus de les afficher il les enregistre temporairement dans une partie de la mémoire du lecteur qui est réservée pour cela. Ces valeurs ne seront ainsi pas perdues et l'utilisateur final pourra de cette manière les récupérée s'il le désire avec le logiciel sur le PC. Les valeurs instantanées des capteurs sont sauvegardées sur la mémoire de la façon suivante (**figure 34**) :

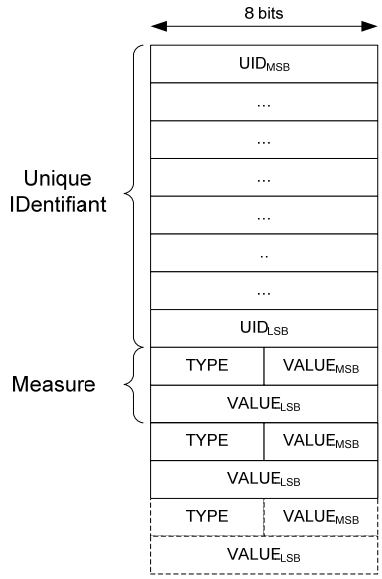

Figure 34 Stockage des valeurs instantanées des capteurs sur la mémoire du lecteur

Le lecteur enregistre pour chaque tag qu'il interroge son identifiant (8 bytes) suivit immédiatement des valeurs des capteurs (12 bits) et leur type (4 bits). La structure complète de la mémoire du lecteur est présentée dans la partie suivante.

6.1.2 Récupération des valeurs enregistrées périodiquement

Lorsque l'utilisateur désir récupérer les valeurs du 'data logging', le lecteur RFID ira simplement lire les blocs de la mémoire des mesures les un après les autres pour les stocker sur propre mémoire. Le lecteur RFID informe ensuite le tag par une commande de l'opération pour que celui-ci réinitialise son pointeur et qu'il recommence les enregistrements au début de la mémoire si le mode 'data logging' n'est pas désactivé. Les valeurs sont enregistrées sur la mémoire du lecteur de la façon suivante (**figure 35**) :

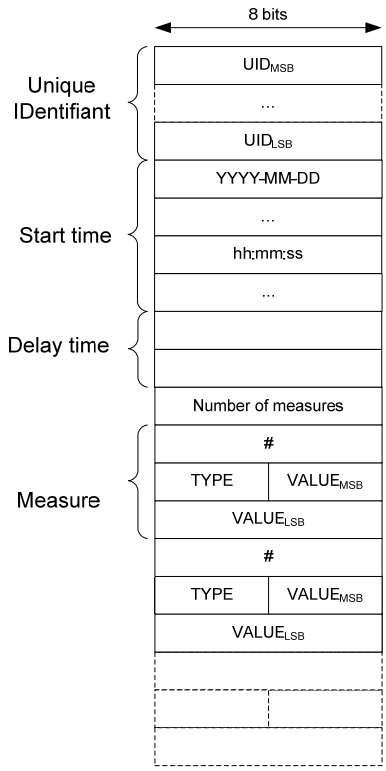

Figure 35 Stockage des enregistrements du 'data logging' sur le lecteur

Lorsque le lecteur récupère les données du ‘data logging’, il enregistre pour chaque tag son identifiant (8 bytes), l’heure du début du ‘data logging’ (4 bytes), le délai entre chaque mesure (2 bytes), le nombre de mesures effectuées (1 byte) et pour finir toutes les valeurs des capteurs avec leur numéro (3 bytes par mesure). Le temps, le délai et le nombre de mesures sont récupérés depuis la mémoire « System ». Ses données sont suffisantes pour afficher sur l’écran de l’ordinateur un graphique avec l’axe du temps et les valeurs de chaque capteur. La structure générale de la mémoire du lecteur est présentée sur la **figure 36**.

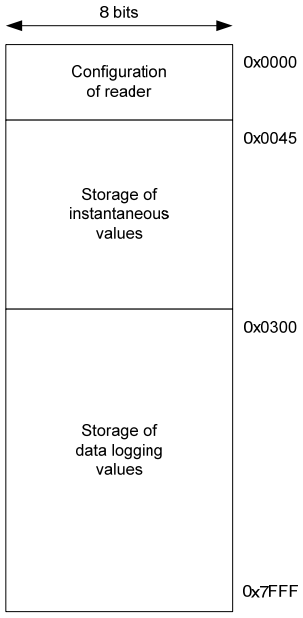

Figure 36 Structure de la mémoire du lecteur

La mémoire du lecteur est comme on peut le voir sur la figure séparée en trois parties. Le début de la mémoire est réservé pour la configuration du lecteur. Les quelques blocs de mémoire qui suivent sont prévus pour sauvegarder les valeurs instantanées des capteurs. Tout le reste de la mémoire est utilisée pour stocker les enregistrements du 'data logging'.

Avec cette configuration le lecteur est capable de stocker les valeurs du 'data logging' d'environ 40 tags différents (250 enregistrements chacun) et environ 30 tags de 8 capteurs sur la mémoire instantanées.

Les explications sur l'utilisation des mémoires s'arrêtent ici. Les sections suivantes donnes les explications techniques sur la façon dont les interfaces sont mises en œuvre et présente de façon générale les algorithmes des deux programmes.

Remarque : Toutes les fonctionnalités prévues au départ non pas été implémentées dans les logiciels à cause des problèmes rencontrés qui ont ralenti la programmation. Seul la récupération des valeurs instantanées a été mise en œuvre. De plus, bien que ce mode ne devrait pas utiliser de batterie, le logiciel sur le tag nécessite toute fois sa présence. Des problèmes subsistaient lorsque la pile n'était pas là pour la conversion des capacités. Il a été jugé plus sûr d'utiliser la batterie et de terminer le travail avec un produit qui fonctionne.

6.2 Firmware du lecteur

L'utilisation du lecteur a été réfléchie pour qu'elle soit la plus simple possible tout en incorporant les deux grandes fonctionnalités que doit avoir le lecteur : lire les valeurs instantanées des capteurs et récupérer les données enregistrés sur la mémoire. Le logiciel du lecteur est conçu pour que l'utilisateur puisse réalisé assez rapidement ces deux tâches.

Puisqu'il l'utilisateur n'aura le choix que entre deux tâches, et pour éviter qu'il ai à se déplacer dans des menus avec le joystick pour réaliser son choix, un système dans lequel il n'aura qu'à appuyer au bon moment sur le joystick pour exécuter la tâche a été élaboré. Le déroulement de ce système est représenté sur la **figure 37**.

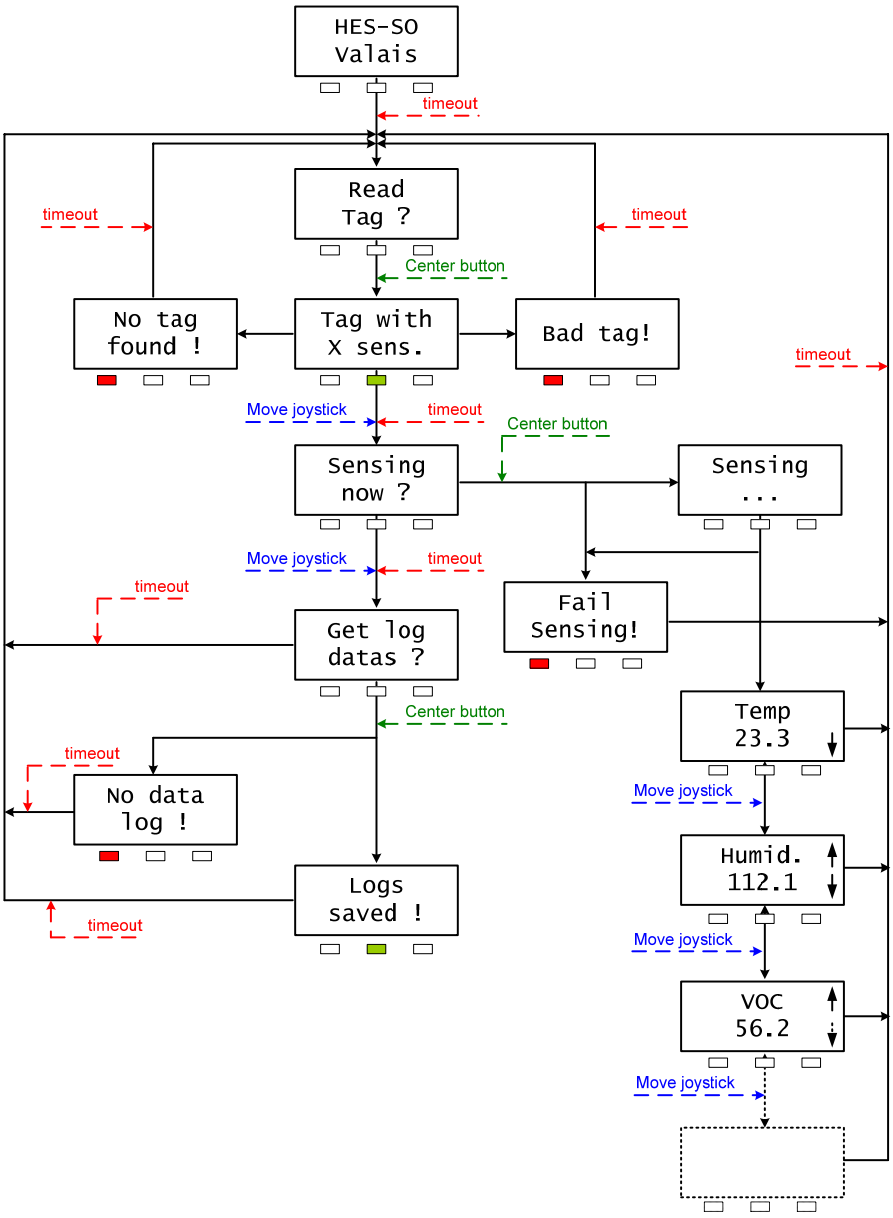

Figure 37 Déroulement des différents affichages de l'écran

Au démarrage de l'appareil il y aura toujours un premier écran qui s'affiche avant de passer, après ~2 secondes à l'écran suivant qui demande à l'utilisateur s'il veut lire un tag. L'utilisateur répond oui en appuyant sur le bouton central du joystick. Si aucun tag n'est présent ou que celui-ci n'est pas conçu pour l'application, un message d'erreur apparaît et disparaît après quelques secondes pour retourner à l'écran précédent.

L'utilisateur répète l'opération avec cette fois-ci un tag devant le lecteur. Un écran avec le nombre de capteurs qui se trouve sur le tag s'affiche. En déplaçant le joystick ou après ~1 seconde, le lecteur fera défiler sur l'écran les questions pour exécuter les deux tâches principales. L'utilisateur n'aura qu'à appuyer sur le joystick au moment ou la question apparaît pour exécuter la tâche correspondante. Le reste du déroulement est identique et se comprend facilement avec le dessin de la figure 37.

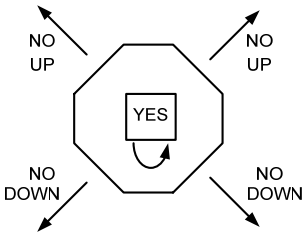

Figure 38 Signification des mouvements du joystick en cours de programme

6.2.1 Description des fichiers

En plus du fichier **main.c** qui implémente le comportement général du lecteur, le logiciel est écrit à travers 7 autres fichiers. Parmi ces 7 fichiers, il y en a 6 qui implémentent les fonctions de bas niveau et un seul (**reader.c**) dans lequel ces fonctions sont utilisées pour réaliser les fonctionnalités spécifiques du lecteur. La description des 6 fichiers de bas niveau est donnée dans le **tableau 4** ci-dessous.

Tableau 4 Fichiers qui implémentent les fonctions de bas niveau du lecteur

anticollision.c	Ce fichier a été récupéré tel quel (avec de légères modifications) du kit de démonstration fourni par IDS. Il contient le processus d'anticollision décrit par le standard ISO15693. Ce dernier est implémenté dans la fonction *InventoryRequest()*. Cette fonction est utilisée pour récupérer les identifiants des tags qui se trouvent dans le champ du lecteur. La fonction *RequestCommand()* implémentée dans ce fichier est utilisé quant à elle pour envoyer une commande vers le tag. La réponse du tag est également récupérée au sein de cette fonction.
spi.c	Ce fichier est aussi récupéré tel quel du kit de développement. Quelques légères modifications ont été apportées à la routine d'interruption qui se trouve à la fin du fichier. Le fichier implémente les fonctions pour communiquer avec la puce RF : c'est-à-dire à lire et à écrire dans les registres de ce dernier et envoyer les données au tag.
lcd.c	Contient les fonctions pour écrire sur l'écran LCD.
eeprom.c	Contient les fonctions pour communiquer avec le mémoire EEPROM. Il a été téléchargé depuis le site de *Texas Instrument*. Il n'a subit aucune modification.
hardware.c	Le fichier contient les fonctions pour initialiser le MSP430 et les modules qu'il utilise.
global.h	Ce fichier est utilisé pour rendre globales des variables qui sont utilisé dans plusieurs fichiers.

6.2.2 Communication avec la puce RF

Le microcontrôleur a besoin à certains moments de communiquer avec la puce RF. Qu'il s'agisse d'écrire dans les registres pour configurer la puce ou transmettre des données au tag, le MCU emploie l'interface parallèle (un port au complet) pour transmettre les informations. Il y a trois modes de communication possible (**figure 39**) :

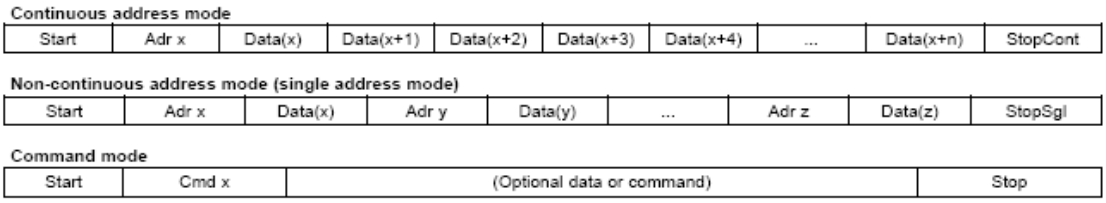

Figure 39 Exemple de communication entre le microcontrôleur et la puce RF

Pour écrire simplement dans un registre on utilise le mode « Single » alors que pour transmettre des données au tag, on utilise le mode « Continuous ». Mais le mode continue peut très bien être utilisé pour écrire dans des registres à la suite car l'adresse s'incrémentera alors automatiquement. Le mode « Single » se réalise avec le fonction *WriteSingle()* et *ReadSingle().* Les données sont transmis en continue au tag avec la fonction *RAWwrite()*.

La puce peut également recevoir des commandes spécifiques comme passer en mode « Idle », faire un « Reset », activé/désactiver des options, … La fonction *DirectCommand()* a été conçue pour envoyer ces commandes.

Le premier octet transmis après le Start indique s'il s'agit d'une adresse ou d'une commande :

Tableau 1 Octet Address/Command

Bit	Description	Bit Function	Address	Command
Bit 7	Command control bit	0 = address, 1 = command	0	1
Bit 6	Read/Write	1 = read, 0 = write	R/W	0
Bit 5	Continuous address mode	1 = Cont. mode	R/W	0
Bit 4	Address/Command bit 4		Adr 4	Cmd 4
Bit 3	Address/Command bit 3		Adr 3	Cmd 3
Bit 2	Address/Command bit 2		Adr 2	Cmd 2
Bit 1	Address/Command bit 1		Adr 1	Cmd 1
Bit 0	Address/Command bit 0		Adr 0	Cmd 0

Au démarrage du lecteur RFID, la puce RF est initialisé avec les paramètres suivant :

- AGC on
- Multiplexage des entrées RF_IN1 et RF_IN2
- Full power
- RF off
- CL_SYS à 6,78 MHz
- Modulation OOK

Les paramètres ci-dessus se configurent avec les registres « *Chip Status Control (00h) »* et « *Modulator and SYS_CLK Control (09h) ».*

6.2.3 Configuration de l'écran

Ecrire sur l'écran est un jeu d'enfant une fois qu'il est correctement paramétré. Pour ce faire il faut respecter les temps qu'il impose pour être configuré. Le datasheet de l'écran indique les temps qu'il met pour exécuter chaque commande. On doit attendre que ce temps soit écoulé avant d'envoyer la prochaine commande.

L'écran dispose de deux mémoire interne : une pour la configuration et l'autre pour les données. Pour afficher les caractères, l'écran ira simplement lire la mémoire des données et afficher le contenu des cases mémoires.

Lors de la lecture ou l'écriture, la mémoire est sélectionnée avec le signal RS. Le signal R/W indique si l'on veut lire ou écrire. Une fois les données sur le port, elles sont validées avec une impulsion sur le signal E.

6.2.4 Communication avec la mémoire

Le lecteur lit et écrit sur la mémoire avec l'interface I^2C. On peut trouver sur le site de *Texas Instrument* (www.ti.com) un exemple de code pour écrire sur les mémoires EEPROM qui utilise le I^2C. Le fichier est utilisé tel quel dans le projet.

La fonction *InitI2C()* initialise le module du microcontrôleur. Après cela, on écrit sur la mémoire avec la fonction *EEPROM_ByteWrite().* On attend que l'écriture soit fini avec la fonction *EEPROM_AckPolling(). P*our lire la mémoire on utilise la fonction *EEPROM_RandomRead().*

La taille de la mémoire utilisée est de 32kB. La dernière adresse est 0x7FFF.

6.2.5 Communication avec le pc

Le lecteur RFID communique avec le PC à travers l'interface USB. Le composant FTDI traduit la ligne série du MSP430 (UART) en signal différentiel (D-, D+) utilisé par le standard USB. En branchant le câble USB, le PC est en communication direct avec le composant FTDI.

FTDI fourni deux alternatives d'interfaces logiciels sur le PC pour communiquer avec ses composants : une interface créer un port COM virtuel (VCP). Celui-ci apparaît aux yeux du système comme n'importe quel autre port COM. La seconde interface, D2XX, est une connexion directe. Elle est fournie via une DLL propriétaire (FTD2XX.DLL). Elle a l'avantage d'offrir des fonctions spéciales qui ne sont pas disponibles par les APIs des ports COM standards comme écrire des données sur la mémoire EEPROM.

Pour une installation sur Windows, les pilotes D2XX et VCP sont distribués dans le même paquet d'installation (CDM – Combined Driver Model) sur le site de FTDI (www.ftdichip.com). Mais un seul peut être utilisé à la fois. Pour des soucis de simplicité le port COM est à préférer.

La puce dispose d'une mémoire EEPROM sur laquelle sont mémorisées les données comme le « Vendor ID » et le « Product ID » ainsi que la description du produit qui apparaît sur le bureau Windows lorsque l'on branche l'appareil. Pour programmer la puce, FTDI fourni le logiciel MProg (**figure 40**).

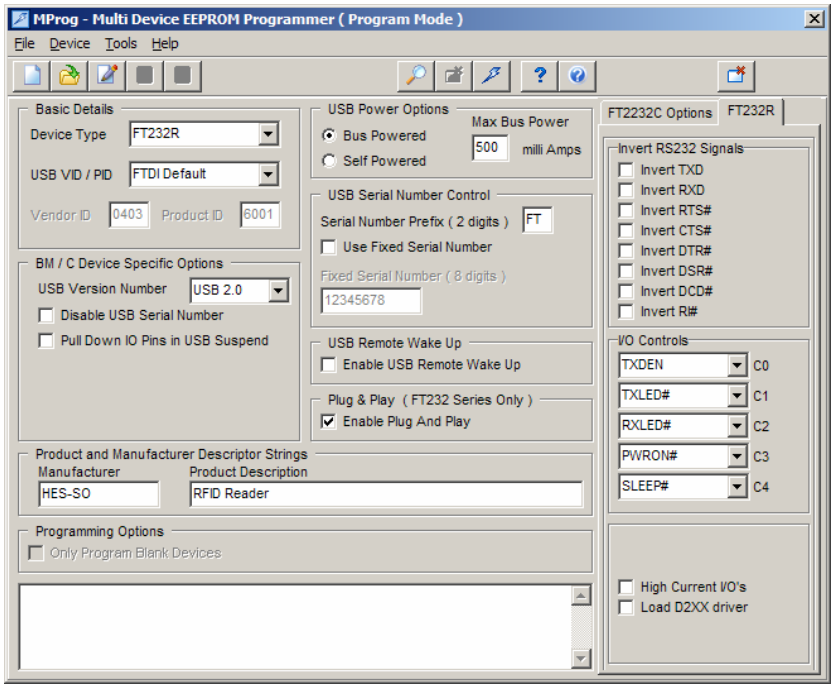

Figure 40 MProg est utilisé pour programmer le composant FTDI qui s'occupe de l'interface USB

Le paramètre important à configurer c'est le courant maximum utilisé par l'appareil. Il faut le configurer à 500mA pour que le chargeur de batterie puisse disposer de ce courant.

Le microcontrôleur n'a plus qu'à lire et écrire dans ses tampons RX/TX de son interface UART pour communiquer avec l'ordinateur.

6.3 Firmware du tag

La tâche principale du microcontrôleur est d'enregistrer les valeurs des capteurs dans la mémoire de la puce RF. Ces valeurs sont soit enregistrées lorsque le lecteur demande les valeurs instantanées des capteurs soit pendant le log lorsque le délai a été passé. Le reste du temps les composants doivent être endormis pour consommer le moins de courant possible.

6.3.1 Communication avec la puce RF

Le microcontrôleur sur le tag communique avec la puce RF avec une interface SPI. L'interface peut uniquement être utilisée pour lire et écrire sur la mémoire EEPROM de la puce RF. Cette dernière exige des trames particulières pour communiquer avec sa mémoire. Ces trames sont décrites ci-dessous.

Opération d'écriture

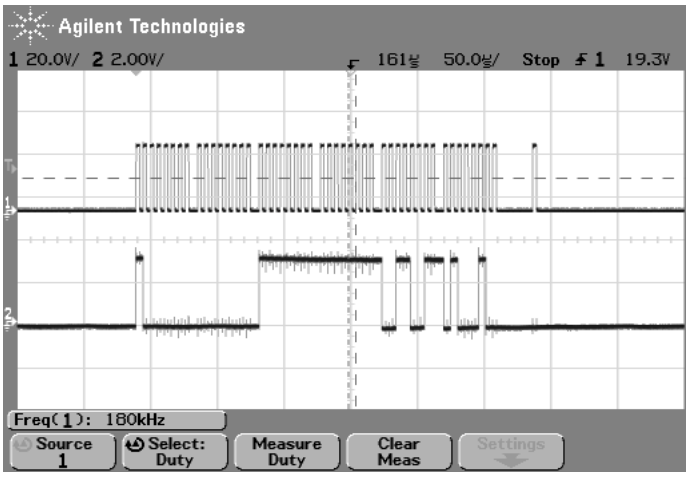

Figure 41 Opération d'écriture sur l'interface SPI de la puce RF

La **figure 41** montre une opération d'écrire sur la mémoire par le SPI. Chaque trame d'écriture est composée de 6 bytes plus une impulsion d'exécution. Le premier byte c'est la commande : 0x80 (0x81 pour écrire sur la mémoire « System »). Le byte suivant c'est l'adresse. Les 4 bytes qui suivent sont les données. Une dernière impulsion sur l'horloge exécutera la commande.

Il est important de respecter le temps de mise en route de la mémoire après le signal CE qui est de 120µs. L'écriture sur la mémoire demande 8ms. Il faut par conséquent également respecter ce délai avant la prochaine opération sur la mémoire.

Pour créer l'impulsion d'exécution, il faut désactiver le module SPI de façon à libérer la pin du microcontrôleur et l'utiliser comme simple entrée/sortie. Un niveau bas, suivit d'un niveau haut puis encore d'un niveau bas suffira à produire l'impulsion demandée.

Opération de lecture

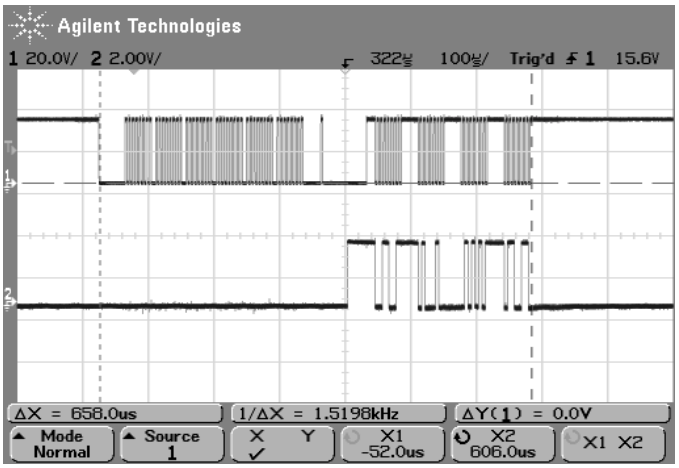

Figure 42 Opération de lecture sur l'interface SPI de la puce RF

La **figure 42** présente une opération de lecture sur la mémoire par le SPI. Comme on peut le voir sur la figure, la commande est plus longue que celle pour l'opération d'écriture. Pour lire un bloc de la mémoire, il faut envoyer une trame de 6 bytes comme précédemment avec cette fois-ci la commande 0x40 (0x41 pour lire la mémoire « System ») suivit de l'adresse qu'on veut lire. Les 6 premiers bytes sont donc identiques que pour l'opération d'écriture à part que les 4 bytes avant l'impulsion d'exécution sont obligatoires mais ne sont pas utilisés.

Une fois l'impulsion d'exécution traitée, la puce RF attend 32 coups d'horloge de la part du maître pour fournir les 4 bytes de données de l'adresse demandée. Pour générer ces coups d'horloge, le maître, c'est-à-dire le microcontrôleur, doit simplement envoyer des données au hasard sur le signal de transmission DIN. Il récupère en même temps les données sur le signal DOUT. La polarité du signal d'horloge a du être inversée pour lire correctement les valeurs.

6.3.2 Communication avec le convertisseur

Le microcontrôleur communique avec le convertisseur capacitif via une interface I^2C. Le convertisseur exige des trames particulières pour lire et écrire dans ses registres. Elles sont décrites ici.

Pour écrire dans un registre du convertisseur, il faut envoyer après la condition S (Start) un premier byte qui contient l'adresse du composant (7 bits) et un bit pour dire si c'est une opération d'écriture ou de lecture (R/W_). L'adresse du composant c'est 0x48. Pour écrire il faut donc envoyer 0x90 et pour lire il faut envoyer 0x91. Le byte suivant c'est l'adresse à laquelle on désire écrire. Ensuite viennent les données. On écrit tant qu'il n'y pas de condition P (Stop). Le pointeur interne s'incrémente automatiquement.

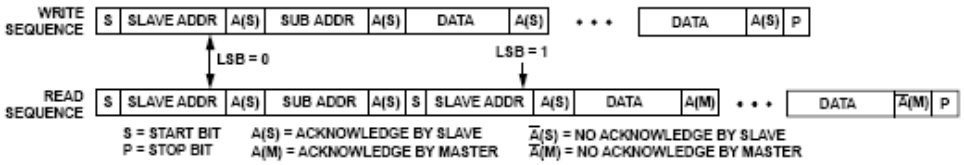

Figure 43 Séquence de lecture/écriture avec le convertisseur capacitif

Pour lire un registre, la séquence démarre de la même manière que pour l'opération d'écriture : on envoie l'adresse du composant suivit de l'adresse du registre que l'on désir lire. En envoyant une nouvelle condition S, on renvoie une nouvelle fois l'adresse du composant. Après ça les données contenues dans le registre sont transmises sur le bus par le convertisseur. Le composant incrémente ensuite automatiquement son pointeur interne et fournit les données des registres à la suite sans s'arrêter tant qu'il n'y a pas de condition P sur le bus. La communication réelle lors de la récupération des valeurs des capteurs a été enregistrée sur l'oscilloscope (**figure 44**).

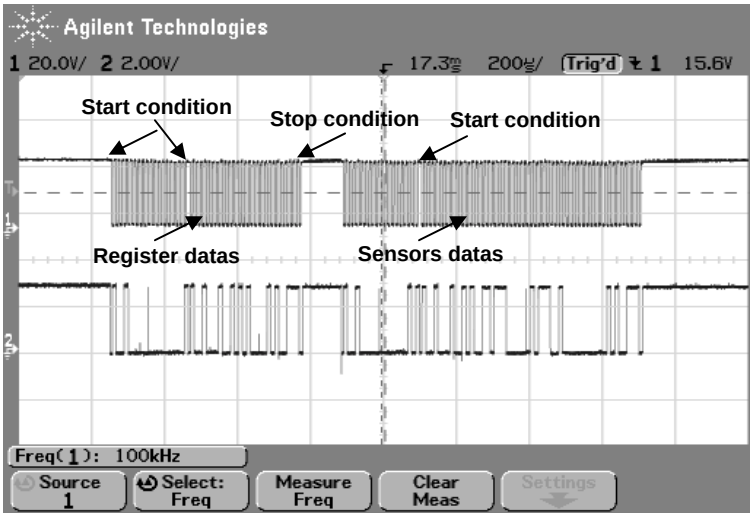

Figure 44 Lecture des registres du convertisseur pour connaître son état et pour récupérer les valeurs des capteurs

L'image de l'oscilloscope montre la lecture des registres par le microcontrôleur après la conversion des capacités. La première partie (trame de gauche) est une opération de lecture avec laquelle on récupère l'état du convertisseur. La seconde partie récupère les 4 registres qui contiennent les données des conversions.

Le microcontrôleur s'endorme dans le mode LPM0 en attendant que la transmission ou la réception se termine.

Pour que la communication marche, il est important de contrôler que les temps de monter et descente, ainsi que les durées des impulsions soit corrects. Le datasheet du convertisseur capacitif fournit cette spécification.

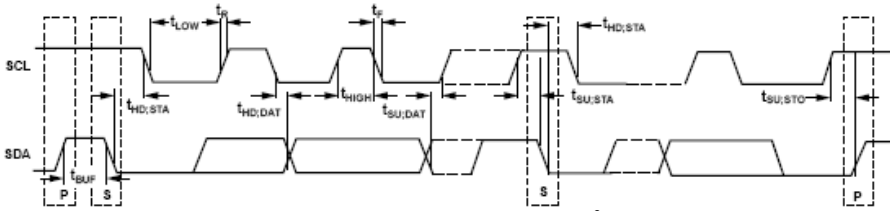

Figure 45 Temps à respecter pour la communication I^2C avec le convertisseur

S'il s'avère que les temps de monter par exemple ne sont pas corrects, il faut ajuster les résistances pull-up du bus. Pour connaître les résistances admissibles il faut se référer à la spécification du bus I^2C [7].

6.3.3 Configuration du module ADC

Le module ADC du microcontrôleur est utilisé pour convertir la tension du capteur de température. Pour mettre en œuvre le module, il suffit de configurer le registre ADC10CLT0. On lance la conversion également avec ce registre.

Avant la conversion le microcontrôleur entre dans le mode LPM0. Dès que la conversion est terminée, ce dernier se fait réveiller par le vecteur d'interruption du module ADC. Le résultat de la conversion sur 10 bits s'enregistre dans le registre ADC10MEM.

6.3.4 Structogramme du logiciel sur le tag

Le fonctionnement du programme sur le tag est décrit à l'aide du structogramme de la **figure 46**. Il ne décrit que la lecture instantanée des capteurs.

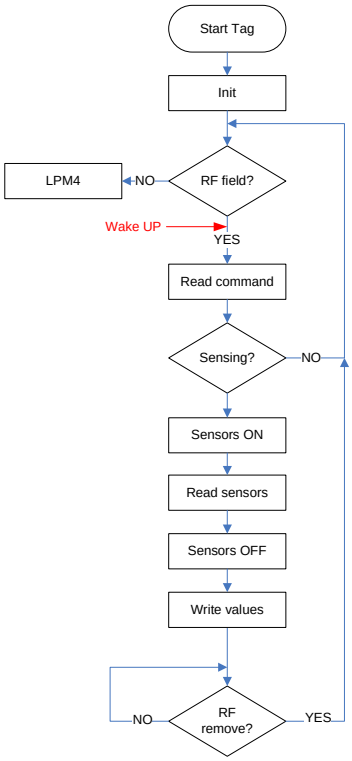

Figure 46 Structogramme du logiciel du tag

La batterie étant présente sur le tag, la microcontrôleur démarre, s'initialise et attend le champ RF. S'il n'est pas là, il entre dans le mode LMP4. Il se fait réveiller par une interruption lorsque le champ est là. Dès qu'il se fait réveiller, il lit le premier bloc de la mémoire EEPROM jusqu'à ce qu'une commande soit envoyée. Si le champ disparaît sans qu'aucune commande n'ait été envoyée, le microcontrôleur s'endort à nouveau.

Lorsque la commande pour lire les capteurs est envoyée, le microcontrôleur allume les capteurs (convertisseur capacitif et pont diviseur), attend que les conversions soient terminées et éteint les capteurs. Il écrit ensuite les valeurs dans la mémoire.

Le microcontrôleur ne fait ensuite plus rien tant que le champ RF n'est pas enlevé. En effet, s'il recommence à lire en continue le premier bloc de la mémoire, il risque d'y avoir une erreur avec la mémoire puisqu'à ce moment là le lecteur va aussi aller lire les blocs de la mémoire. Le tag doit s'endormir une fois avant de pouvoir lire à nouveau les capteurs.

7 CONSOMMATIONS

Ce chapitre présente la consommation électrique du lecteur et du tag. La durée de vie des batteries est également calculée pour les deux appareils.

7.1 Consommation du lecteur

Le lecteur peut se retrouver dans 4 états différents. Dans chacun des états il consomme une certaine quantité de courant. Les 4 états et leur consommation sont représentés sur la **figure 47**.

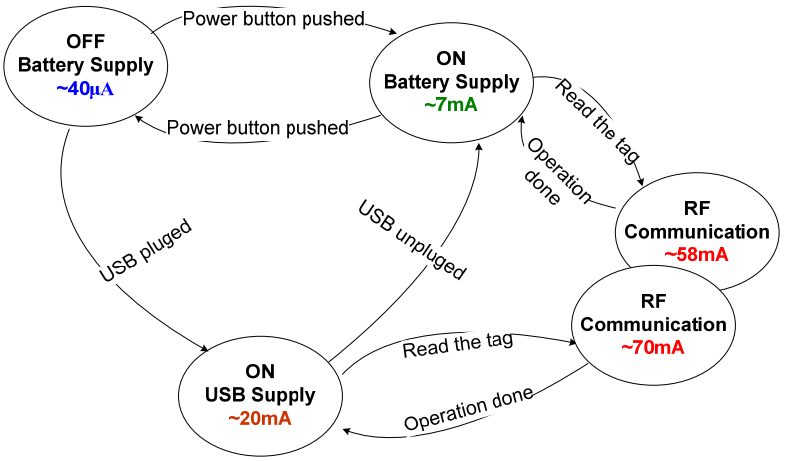

Figure 47 Diagramme des états de consommation effective du lecteur

Lorsque le lecteur est éteint, une batterie d'une capacité de 660mAh peut durer environ 16500 heures et 94 heures lorsqu'il est allumé. La lecture d'un tag dure environ 6 secondes. Si par exemple un tag est lu chaque dix minutes, le nombre de mesures qu'il est possible de faire se calcule comme suit :

$$\frac{660mA \cdot 3600}{6s \cdot 58mA + 600s \cdot 7mA} \approx 522$$

7.2 Consommation du tag

Le tag peut se trouver dans trois états différents. Ils sont décrits avec leur consommation sur la **figure 48**.

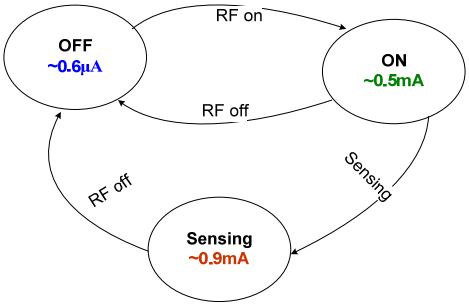

Figure 48 Diagramme des états de consommation effective du tag

Dans l'état OFF le microcontrôleur est dans la mode LMP4. Lorsque le champ RF se présente le microcontrôleur se réveille (état ON) et consomme 500µA. Lorsque le lecteur demande une lecture des capteurs, la consommation s'élève à 900µA. Cette augmentation est sans doute due à l'écriture sur la mémoire. En effet, les capteurs, une fois la lecture terminée, sont éteints par le microcontrôleur. La consommation reste quant même à 900µA tant que le champ RF n'est pas enlevé.

La capacité de la pile bouton utilisé sur le tag est de 170mAh. Le tag peut rester dans l'état OFF environ 32 années (11800 jours). Si le tag est interrogé chaque 10 minutes par un lecteur, le nombre de mesure peut se calculer comme suit :

$$\frac{170mA \cdot 3600}{3s \cdot 0{,}5mA + 3s \cdot 0{,}9mA + 600s \cdot 0{,}6\mu A} \approx 134000$$

8 DESCRIPTION HARDWARE

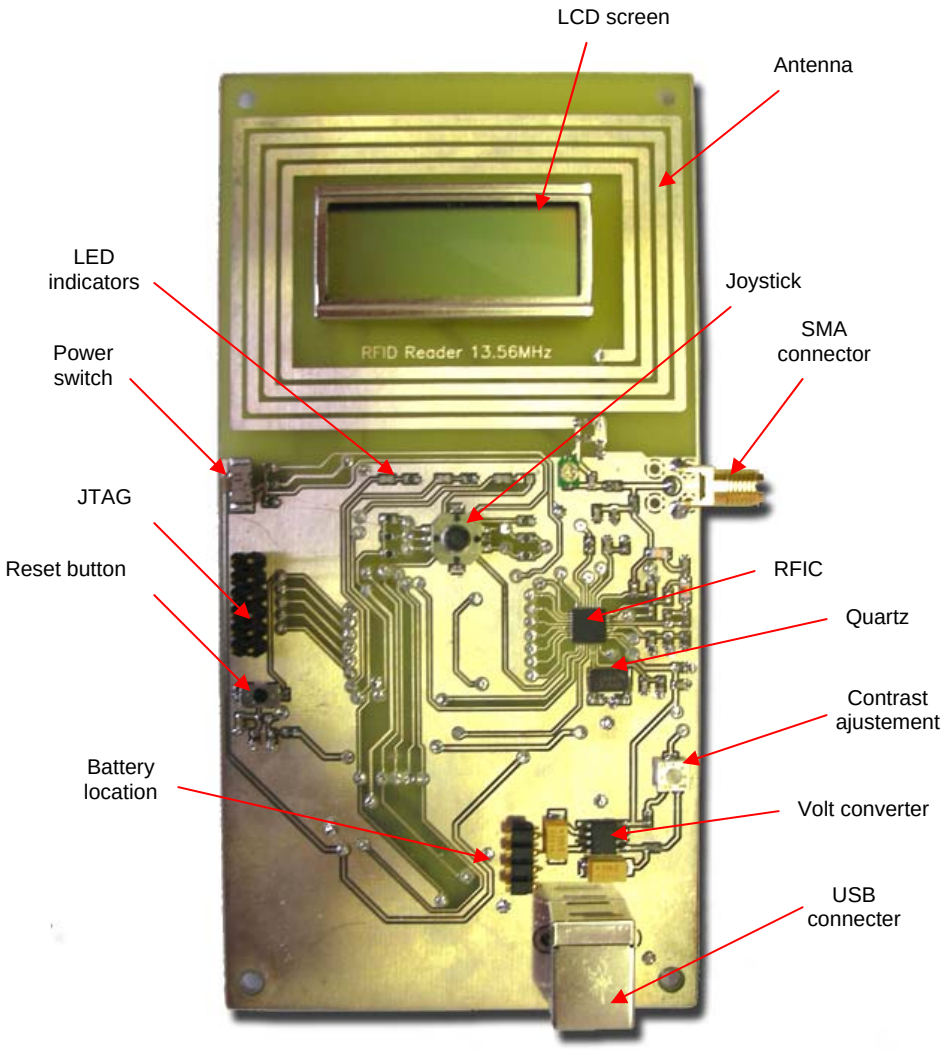

Figure 49 Image de la façade avant du lecteur

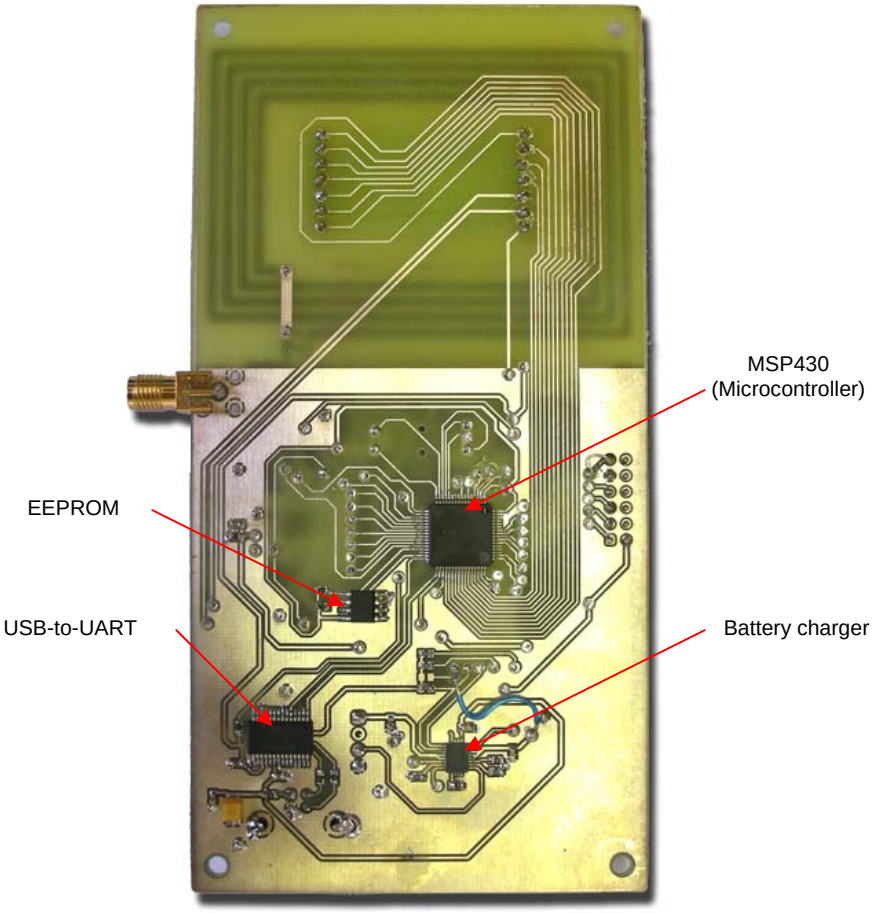

Figure 50 Image de la façade arrière du lecteur

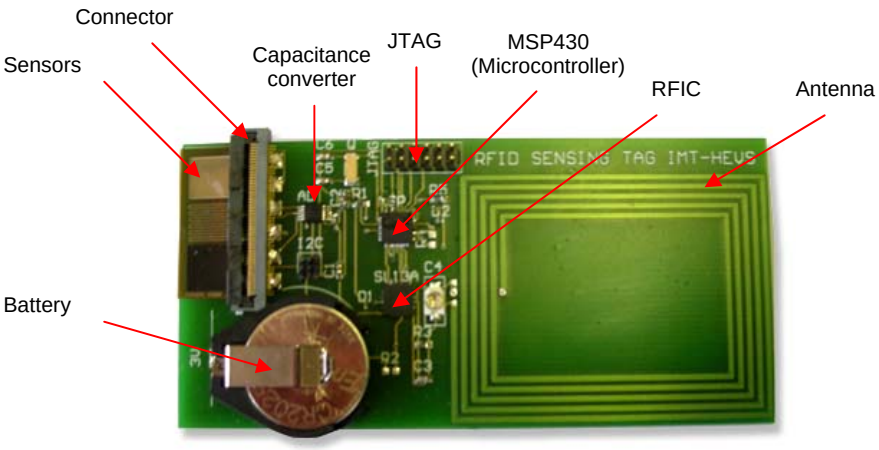

Figure 51 Image du tag

9 AMELIORATIONS

Les problèmes rencontrés lors de la programmation ont ralenti le développement des logiciels. Plusieurs fonctionnalités n'ont pas pu être implémentées. Elles sont décrites dans cette section. Les résultats obtenus avec les antennes des tags se sont révélés contradictoire avec ce qui a été compris de la théorie. Ce chapitre explique également de quelle façon l'antenne du tag peut être améliorée.

9.1 Logiciel du lecteur

Pour obtenir un lecteur complet, les fonctionnalités suivantes devront être implémentées :

- Enregistrement sur la mémoire des valeurs instantanées selon la figure 34.
- Processus de récupération des mesures du 'data logging' et enregistrement sur la mémoire selon la figure 35.
- Implémentation de l'interface UART pour la communication avec le logiciel sur le PC.
- Le chargeur de batterie n'a pas fonctionné correctement. Il faudrait contrôler qu'avec un nouveau chargeur de batterie la recharge fonctionne.
- Trouver pourquoi le champ RF ne n'est pas désactivé lorsque le câble USB est branché.

9.2 Logiciel du tag

Pour obtenir un tag complet, les fonctionnalités suivantes devront être implémentées :

- Mode 'data logging'
- Rendre possible la lecture des capteurs sans la pile
- Améliorer la consommation électrique en essayant de configurer une horloge plus petite sur le microcontrôleur.
- Comprendre pourquoi après une lecture des capteurs la consommation ne redescend pas à 0,5mA et essayer de la résoudre.

9.3 Logiciel sur le PC

Le logiciel sur le PC est à programmer depuis le début ou presque. Le kit de démonstration de *IDS Microchip* contient le code source du logiciel Windows utilisé pour communiquer avec le lecteur. Il a été programmé en C++ et utilise le port COM virtuel. Le nouveau logiciel peut s'inspirer de ce code pour communiquer avec le lecteur. Mais puisque le programme de *IDS Microchip* est assez fourni, il est préférable de commencer la programmation du nouveau logiciel depuis le début.

Un exemple de code qui créer une interface et qui ouvre le port COM est fourni dans les sources du projet. Il a été créé avec *Microsoft Visual Studio 2005* et peut être utilisé comme base pour débuter la programmation (**figure 52**).

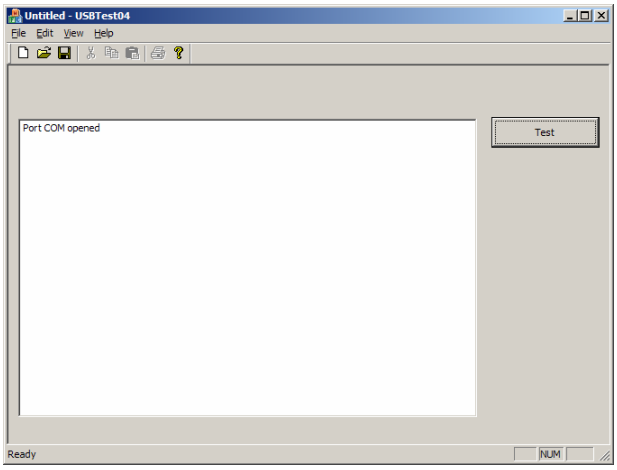

Figure 52 Interface du programme qui peut être utilisé comme base pour le développement du logiciel sur le PC

9.4 Schématique du lecteur

Il n'était pas venu à l'esprit au moment de la réalisation de la schématique du lecteur qu'il serait intéressant d'utiliser un composant step-up qui élèverait la tension de la batterie de 3,7V à 5V pour alimenter la puce RF de manière à avoir une plus grande distance de lecture.

Ce même step-up pourrait être utilisé pour alimenter l'écran LCD à 5V au lieu de +3,7 et -3,7V car les grandes capacités que le convertisseur de tension utilise prennent de la place.

9.5 Schématique du tag

La schématique du tag peut également être améliorée. Tout à bord l'oscillateur externe n'est nécessaire. La programmation et les communications SPI et I^2C fonctionnent très bien avec le DCO interne.

Il faut rajouter la connexion TEST/VPP de l'interface JTAG sur le microcontrôleur.

Les résistances pull-up internes du microcontrôleur ne sont pas suffisantes pour le bus I^2C. Il faut par conséquent rajouter deux résistances pull-up sur les deux signaux du bus. La tension sur les résistances pull-up doit provenir du microcontrôleur.

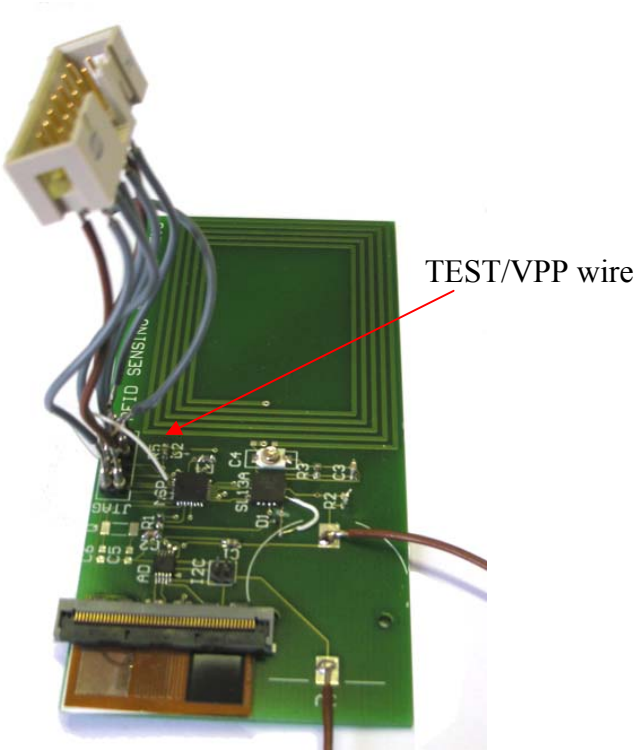

Figure 53 JTAG patch

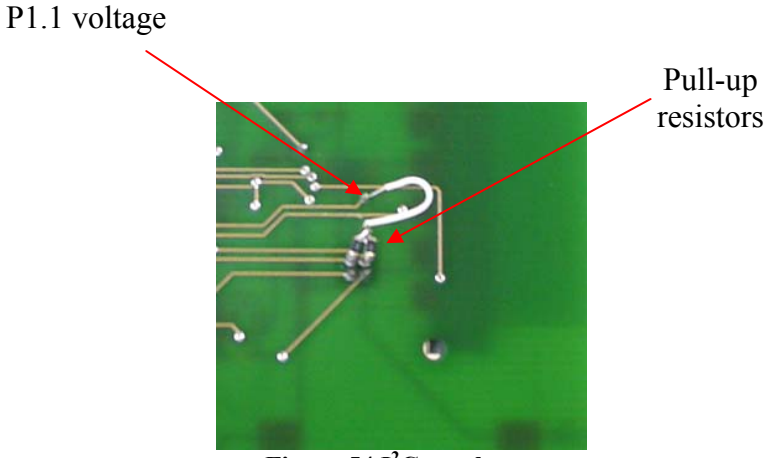

Figure 54 I²C patch

9.6 Antenne du tag

Généralement le paramètre recherché pour concevoir une antenne est le facteur de qualité. Les simulations réalisées à la section 5.2.1 ont montré que l'antenne de droite sur la figure 20 est meilleure de ce point de vue là que celle de gauche. Mais les expériences tendent à prouver que l'antenne de gauche serait en réalité un meilleur choix. En effet, les essais menés avec le lecteur sur le tag de *IDS Microchip* et le tag du projet ont montré que celui de IDS avait une plus grande distance de lecture. Certes la spirale de celui-ci est plus grande, mais il semblerait que celle-ci soit plus performante grâce à sa plus grande surface interne.

Le facteur de qualité reste toute fois important, mais une spirale qui a une plus grande surface interne englobe plus de champ magnétique. Le tag gagne de ce fait en distance de lecture. Puisque de toute façon le facteur de qualité de l'antenne est diminué avec une résistance en parallèle, l'antenne de gauche sur la figure x est à préférer pour le développement d'un futur tag.

9.7 Calibration des capteurs

Les valeurs des capteurs affichées sur l'écran du lecteur ne sont pas vraies. L'inconvénient des capteurs capacitifs c'est qu'ils ont une très faible variation (**annexe 10**). La valeur de la capacité est de plus un peu plus grande que ce que le convertisseur peut accepter sur ses entrées. Mais ce dernier semble tout de même vouloir convertir les valeurs. Une étude approfondie de la configuration du convertisseur capacitif est nécessaire.

Le programme du lecteur affiche des valeurs fausses mais il a été ajusté pour qu'il affiche de grandes variations même si les capteurs varient très légèrement. La calibration exacte des valeurs qui sont affichées sur l'écran est laissée pour un futur développement.

10 REMERCIEMENTS

Remerciement à M. Riad Kanan pour le coaching tout au long du projet, à Steve Gallay et à Olivier Walpen de l'atelier électronique, à Pascal Sartoretti pour les commandes des pièces. Merci également à Jérôme Courbat pour sa collaboration à la réalisation du tag RFID.

11 CONCLUSION

Le système RFID développé dans ce projet a eu pour objectif de démontrer les faisabilités d'un lecteur RFID portable ainsi que la mise en œuvre de capteurs en polyimide fabriqués par l'institut de microtechnique (IMT) de l'université de Neuchâtel.

Un kit de développement avec lecteur, tag et logiciel sur Windows a été commandé chez *IDS Microchip*. Ce kit a servit de base pour le développement du projet. Les composants principaux du lecteur ont été repris (*IDS-R13MP*, *MSP430F156*) et plusieurs autres éléments ont été rajoutés pour en faire un lecteur portable. Le nouveau lecteur RFID est muni d'une batterie rechargeable, d'un écran lcd, d'un joystick et de leds. Un chargeur de batterie s'occupe de recharger proprement la batterie de 3,7V. Une interface USB est utilisée pour recharger la batterie et communiquer avec un ordinateur. L'antenne a été mesurée et calibrée à l'aide d'un analyseur réseau. L'interface RF de l'appareil a été simulée avec *Ansoft Designer* avant d'être réalisée.

Le tag a également été réalisé à l'aide de la puce RF de *IDS Microchip* spécialement conçu pour les tags (*IDS-SL13A*). Le nouveau tag est muni d'un microcontrôleur et d'un convertisseur capacitif pour interfacer les capteurs. L'antenne a été simulée avec *SonnetLite* avant sa réalisation. La communication entre le lecteur et le tag se fait sur la base du standard ISO 15693 à 13,56MHz.

Les logiciels du lecteur et du tag ont été programmés en C. Toutes les fonctionnalités prévues au départ n'ont pas été implémentées. Le lecteur est pour le moment uniquement capable de lire en temps réel les valeurs des capteurs. L'activation de l'enregistrement périodique des capteurs ainsi que le logiciel sur Windows qui doit récupérer les valeurs stockées sur le lecteur n'ont pas étaient mis en œuvre.

Darko Petrovic

12 LISTE DES ANNEXES

1. Schématique du lecteur
2. Schématique du tag
3. PCB du lecteur
4. Image du 'print' du tag
5. Vue centrale du 'print' du tag
6. Liste des composants du lecteur
7. Liste des composants du tag
8. Code source du logiciel sur le lecteur
9. Code source du logiciel sur le tag
10. Données pour la calibration des capteurs

13 REFERENCES

[1] Identification cards – Contactless integrated circuit(s) cards – Vicinity cards

[2] TI, HF Antenna Cookbook Technical Application Report, 11-08-26-001 Jan 2004

[3] Melexis, 13.56 MHz RFID systems and antennas design guide, Rev. 001 Oct.2004

[4] RFID Handbook, Second Edition, p.70

[5] AN710 - Microchip - Antenna Circuit Design for RFID Applications

[6] MSP-FET430 Flash Emulation Tool (FET) (for Use With Code Composer Essentials for MSP430 Version 3)

[7] The I^2C-Bus specification, Version 2.1, January 2000

14 CD-ROM

Le CD-ROM en annexe contient les dossiers suivants :

1_Schematics Il contient la schématique du lecteur. Le fichier .sch peut être ouvert avec le logiciel PCAD.

2_PCB Il contient le 'print' pour le lecteur. Le fichier .pcb peut être ouvert avec le logiciel PCAD. Le fichier gerber du tag se trouve également dans le dossier.

3_Composants Il contient les datasheets des composants utilisés sur le lecteur et le tag, ainsi que les liste des composants (BOM).

4_Documentation	Il contient de la littérature générale pour comprendre le fonctionnement des composants ainsi que la documentation de quelques logiciels utilisés dans le projet.
5_Software	Le dossier contient les codes sources du lecteur et du tag, ainsi que le début de l'interface pour le logiciel Windows.
6_Install	Contient les exécutables des programmes qui ont été utilisés durant le projet.
7_Simulation	Fichiers de simulation réalisée avec *Ansoft Designer* et *SonnetLite*.