# Conception et réalisation d'un démonstrateur embarqué de traitement du signal avec réseau de capteurs sans fils

Nathalie Teppe

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

---

**MEMOIRE**

**présenté en vue d'obtenir**

**le DIPLOME D'INGENIEUR CNAM**

**SPECIALITE : ELECTRONIQUE – UE UA5M25**

**par**


## Nathalie TEPPE


---

# Conception et réalisation d'un démonstrateur embarqué de traitement du signal avec réseau de capteurs sans fils

---

Soutenu le 29 Mai 2013, à Grenoble

**JURY**

| | | |
|---|---|---|
| **PRESIDENTE :** | **Mme Catherine ALGANI,** | **Professeur au CNAM de Paris** |
| **MEMBRES :** | **M. Claude DELPUECH,** | **Docteur Ingénieur à l'INSERM** |
| | **M. Thierry FALQUE,** | **Directeur Développement Analogique à STMicroelectronics** |
| | **M. Pierre GRANJON,** | **Enseignant-chercheur à Grenoble-INP et au GIPSA-Lab** |
| | **M. Alain COUPAT,** | **Directeur du développement produits à OROS S.A** |

CONSERVATOIRE NATIONAL DES ARTS ET METIERS

CENTRE REGIONAL RHÔNE-ALPES

CENTRE D'ENSEIGNEMENT DE GRENOBLE

---

**MEMOIRE**

**présenté en vue d'obtenir**

**le DIPLOME D'INGENIEUR CNAM**

**SPECIALITE : ELECTRONIQUE – UE UA5M25**

**par**

**Nathalie TEPPE**

---

# Conception et réalisation d'un démonstrateur embarqué de traitement du signal avec réseau de capteurs sans fils

---

Soutenu le 29 Mai 2013, à Grenoble

Les travaux relatifs à ce mémoire ont été effectués au Laboratoire GIPSA-Lab,
situé au 11 rue des Mathématiques – BP n° 46 – 38402 Saint Martin d'Hères – France,
du 5 Septembre 2011 au 22 Juin 2012,
sous la direction de Nadine MARTIN, Directrice de Recherche au CNRS.

# Remerciements

Je voudrais remercier mes collègues du laboratoire Gipsa-Lab, et plus particulièrement Mme Nadine Martin, ma tutrice de stage, Mme Michelle Vieira et M. Pierre Granjon pour leur précieuse collaboration tout au long de ce projet et M. Julien Minet, ingénieur en informatique embarquée pour ses judicieux conseils.

J'aimerais également remercier mes collègues de la société OROS, notamment M. Daniel Degryse, mon ancien directeur technique, pour leur soutien et leur encouragement tout au long de ces années que j'ai passées au CNAM à préparer ce diplôme.

Un grand merci également à tous mes relecteurs (collègues, amis, famille) pour leur précieuse aide à la correction et à l'amélioration de ce mémoire.

Enfin, je voudrais rendre hommage à toute l'équipe administrative et éducative du CNAM de Grenoble, conduite par M. André Plisson et Mme Véronique Marie Olive, sans qui toutes ces années de travail et d'études n'auraient pas été possibles.

# Sommaire

# Liste des Figures



# Liste des Tableaux

# Acronymes

| | |
|---|---|
| CAN | Convertisseur Analogique Numérique |
| CBM | Condition-Based Monitoring |
| ADC | Analog to Digital Converter |
| DMA | Direct Memory Access |
| GNU | GNU's Not Unix |
| $I^2C$ | Inter Integrated Circuit |
| IDE | Integrated Development Environment |
| IEPE | Integrated Electronics Piezo Electric |
| IHM | Interface Homme Machine |
| MEMS | Micro Electro Mechanical System |
| MFC | Microsoft Fondation Class |
| SPI | Serial Peripheral Interface |
| WSN | Wireless Sensor Network |
| WPAN | Wireless Personnal Area Network |

# Introduction

L'équipe de recherche « Signal et Automatique pour la surveIllance, le diaGnostic et la biomécAnique » (SAIGA) fait partie du laboratoire GIPSA-lab. SAIGA s'appuie sur une activité dans le domaine de l'analyse de signaux. Dans ce contexte, un des besoins importants de l'équipe SAIGA est un démonstrateur embarqué lui permettant de prouver sur des signaux expérimentaux l'efficacité des méthodes et traitements qu'elle a développé ces dernières années.
Les domaines d'application sont nombreux, citons par exemple la surveillance et le diagnostic de machines pour l'industrie, le suivi de paramètres physiologiques pour la santé ou la prévention des risques sismiques pour la géologie.

Le projet consistera à proposer et réaliser un démonstrateur embarqué, basé sur un réseau de capteurs sans fil, afin de permettre l'acquisition et l'analyse des signaux.
Il devra être portable afin de réaliser des démonstrations lors de salons, d'expositions ou de présentation de travaux à des partenaires.

Le système complet se compose :

- d'un réseau de capteurs analogiques et numériques
- d'un module d'acquisition des signaux fournis par ces capteurs
- d'un ordinateur avec un logiciel pour enregistrer et traiter les signaux

On pourra s'appuyer sur les méthodes et les produits existants afin de bâtir une solution adaptée. Le travail consistera à intégrer ces éléments et à réaliser les études complémentaires pour satisfaire les critères qui seront spécifiés lors de l'élaboration du cahier des charges.
Une des principales difficultés sera de trouver une solution permettant de satisfaire à la fois les contraintes d'acquisition temps réel et synchrone des capteurs, tout en utilisant une communication radio robuste adapté au haut débit.

Le travail sera décomposé en 3 étapes :

Analyse du besoin
Une analyse du besoin sera effectuée avec les futurs utilisateurs, afin de déterminer le plus précisément possible la forme que devra prendre le système voulu.
Une étude de marché pourra être réalisée à ce stade afin de comparer les produits du commerce qui pourraient éventuellement répondre à des sous-parties du besoin exprimé.
Les choix techniques seront faits, selon les contraintes et spécifications définies.
Un cahier des charges sera réalisé, qui servira de fil conducteur à la réalisation du projet.

Réalisation
La réalisation du démonstrateur embarqué suivra le cahier des charges précédemment établi, en deux temps :

- création d'une première maquette pour valider le concept
- amélioration de cette maquette (ajout de capteurs, fiabilisation)

Les réalisations électroniques seront réalisées dans les ateliers du laboratoire.
Une documentation complète sera fournie aux utilisateurs pour se servir du démonstrateur.

Bilan et perspectives
Le prototype livré à la fin du stage devra être opérationnel et utilisable.
On décrira les parties à perfectionner et les évolutions futures possibles.

# 1 UN PROJET TECHNIQUE POUR UN LABORATOIRE DE RECHERCHE SCIENTIFIQUE

## 1.1 Environnement de travail : Gipsa-Lab

### 1.1.1 Un laboratoire de recherche

Le choix d'effectuer mon stage dans ce laboratoire n'est pas dû au hasard.
Il résulte d'une réflexion visant à réaliser mon mémoire dans un contexte différent de celui d'une PME industrielle, tout en restant dans un domaine scientifique proche de mon expérience professionnelle en traitement du signal. Le Gipsa-Lab (*Grenoble Images Parole Signal Automatique*) [1] répondait parfaitement à ces deux critères.
C'est un laboratoire de recherche publique grenoblois, internationalement réputé dans ses domaines de compétences, qui regroupe depuis 2007 trois départements :

- Automatique (anciennement LAG : Laboratoire d'Automatique de Grenoble)
- Images et Signal (anciennement LIS : Laboratoire des Images et des Signaux)
- Parole et Cognition (anciennement ICP : Institut de la Communication Parlée)

Chaque département est divisé en 4 à 5 équipes de recherche, soit une douzaine au total.
J'ai rejoint l'équipe SAIGA (*Signal et Automatique pour la surveillance, le diagnostic et la biomécanique*) qui a la particularité d'être commune aux départements « Automatique » et « Images et Signal ».
Le Gipsa-Lab traite de thématiques transversales et pluridisciplinaires sur les signaux et systèmes complexes, sous l'égide de plusieurs tutelles (CNRS, Grenoble INP, Université Joseph Fourier, Université Stendhal).

Ce ne sont pas moins de 300 personnes (160 permanents et 140 doctorants/stagiaires) qui travaillent de concert, dont de nombreux doctorants étrangers venant de pays aussi variés que le Brésil, le Pakistan, l'Iran, le Viêt-Nam…
C'est dans ce contexte multiculturel, que j'ai eu la chance de pouvoir assister à des présentations de thèses, de séminaires et de rencontrer des sommités dans leurs matières. J'ai ainsi eu l'opportunité de rencontrer William Gardner, spécialiste de la cyclostationnarité [2] et Jean-Louis Lacoume, spécialiste en traitement du signal [3] lors d'une journée dédiée à la cyclostationnarité et à ses applications possibles dans les domaines de recherches du laboratoire.

Cette expérience m'a donné l'opportunité de suivre l'évolution du contexte institutionnel de l'université et de la recherche avec, par exemple, la création des Pôles de Recherche et d'Enseignement Supérieur (PRES), la coloration par des « labels » pour les doctorants, l'introduction d'outils de gestion de la performance…
J'ai ainsi découvert de l'intérieur une autre facette du triptyque Université-Recherche-Industrie qui fait tant la renommée de Grenoble, dans un environnement de travail très différent de celui d'une PME industrielle.

### 1.1.2 Recherche & industrie : valoriser l'innovation

La recherche et l'industrie, deux mondes que tout oppose et qui auraient tant à gagner à se rapprocher. Une meilleure synergie serait bénéfique au devenir économique et industriel de la France. Grenoble fait mieux que la moyenne nationale grâce à son pôle de compétitivité Minalogic, ses incubateurs et autres structures comme l'Alliance Université Entreprise de Grenoble (AUEG) pour favoriser l'innovation.
De nombreux résultats de recherches restent inexploités. Les laboratoires recèlent de véritables trésors qui mériteraient d'être valorisés pour offrir des produits et/ou des solutions innovantes. Et cela concerne non seulement les sciences dites « dures », mais aussi les Sciences Humaines et Sociales (SHS) essentielles pour comprendre les enjeux sociétaux comme par exemple le bien-être dans le monde du travail : en évaluant et en diffusant les bonnes pratiques.
En ce sens, la présentation de la journée des doctorants en début d'année montre toutes les possibilités d'applications pratiques de leurs travaux. On peut même créer sa propre entreprise.

On constate un frein culturel à l'emploi des doctorants dans l'industrie en opposant la recherche publique « noble », non vénale, à la course aux profits des sociétés privées. Il ne faut pas en avoir honte, c'est une exception culturelle bien française de mépriser les activités rémunératrices du secteur marchand. Et pourtant cette création de richesse se retrouve forcément à la fois dans l'économie, par la consommation supplémentaire qu'elle engendre, et dans le social, par la redistribution des sommes collectées par les impôts. Sans oublier notre balance commerciale déficitaire sur laquelle un partenariat recherche/industrie apporterait un effet de levier non négligeable.
On oppose également recherche fondamentale et recherche appliquée : la première nourrit la deuxième. Il faut déterminer des concepts scientifiques fondamentaux avant d'extrapoler et d'en tirer la recherche appliquée qui pourra à son tour nourrir l'industrie.
Ce qui compte, c'est la passion. On peut être tout autant passionné de son métier et s'épanouir dans une entreprise privée.
Cependant des outils comme le crédit impôt recherche, les conventions de thèses Cifre, les partenariats public-privé tendent à rapprocher ces deux mondes, même si les lourdeurs administratives se font encore sentir [4].

Heureusement, l'environnement grenoblois propose un cadre propice aux transferts de technologies. Ceux-ci étant surtout orientés vers les grands groupes comme ST ou Orange Labs, mais qui gagnerait aussi à miser plus régulièrement sur les PME à taille humaine pour en faire des « gazelles » comme le préconise judicieusement Patrick Arthus et Marie-Paule Virard dans « La France sans ses usines » [5].
Des collaborations voient également le jour au niveau européen.
Ainsi les travaux de Nadine Martin, directrice de Recherche, pour concevoir TetrAS [6], un « super analyseur de spectres » basé sur une méthode innovante de traitement et d'interprétation du signal, a naturellement trouvé un débouché dans les énergies renouvelables (éoliennes) au sein d'un consortium associant partenaires industriels et laboratoires (KAStrion [7]).

On note depuis quelques années des difficultés de la recherche française à séduire les étudiants français (situation identique dans l'industrie) dues à une désaffection des filières scientifiques[8].
J'ai eu la chance de rencontrer des doctorants étrangers issus des cinq continents.
C'est d'autant plus courageux de leur part, qu'ils font bien souvent face à de nombreuses difficultés administratives [9].

### 1.1.3 Comprendre les besoins des chercheurs

Venant de l'industrie, où le marketing transcrit les besoins des clients pour établir un cahier des charges qui servira de fil conducteur pour le bureau d'étude, j'ai dû m'adapter à un client un peu différent : le chercheur.
Sur le fond, le travail est identique, mais sur la forme, un effort et une attention toute particulière doivent être portés sur le vocabulaire et la compréhension des uns et des autres.
Pour cela, il faut également comprendre le contexte : le travail des chercheurs consiste en l'exploitation mathématique des signaux, sans forcément savoir comment ils ont été obtenus techniquement dans le détail. D'où un décalage que je me suis efforcée de combler en m'intéressant de près à l'articulation des travaux en cours entre les différentes équipes, afin de m'approprier un langage commun pour appuyer mes propositions.
La démonstration que j'ai pu faire de mon projet lors d'une réunion d'équipe vers la fin de mon stage m'a conforté dans la réussite de cet objectif. La compréhension de mon exposé n'a pas soulevé de problème particulier et les questions posées ont révélé tout l'intérêt et le potentiel que mon projet peut apporter dans le cadre des recherches des membres de l'équipe.

### 1.1.4 Outils à disposition

L'intérêt de travailler dans un laboratoire de recherche sur un campus universitaire est l'accès à une base de connaissances remarquables.
En effet, en tant que membre de ce laboratoire, j'avais accès non seulement à la bibliothèque spécialisée du laboratoire mais aussi à tous les fonds documentaires des bibliothèques universitaires. Les livres que j'ai empruntés m'ont beaucoup aidé à la compréhension des concepts physiques et techniques nécessaires à la réalisation de mon projet.
Les thèses, les bases de données et les magazines spécialisés ont naturellement complété ma base de travail, comme en atteste la riche bibliographie jointe à la fin de ce rapport.

Les nombreuses plates-formes du laboratoire m'ont donné une idée du type de manipulations et des outils employés, tant au niveau des capteurs que de la mise en œuvre de ceux-ci, afin que je puisse orienter mon travail en ayant à l'esprit ce qu'utilise les chercheurs au quotidien.
Le laboratoire comporte également un atelier de mécanique et d'électronique, chapeauté par une équipe technique, où j'ai passé une bonne partie de mon projet pour réaliser les équipements électroniques dont il se compose.

De même, au niveau informatique, j'ai bénéficié à la fois du support informatique pour installer et utiliser les outils logiciels dont j'avais besoin et des compétences des ingénieurs en programmation (embarqué, C++) pour m'épauler dans les phases délicates de codage.

Enfin, grâce au service achat, j'ai pu commander toutes les fournitures électroniques nécessaires à la conception de mon projet.

### 1.1.5 Relation de travail

Le projet que j'ai mené dans le cadre de ce stage est un outillage destiné aux chercheurs.
Dans ce cadre, le rôle d'un développeur électronique est d'assurer une fonction de support et d'accompagnement des chercheurs en leur fournissant des outils adaptés à leurs besoins spécifiques.
Le rythme de travail adopté s'inspire des méthodes agiles : un point tous les quinze jours pour faire le bilan de l'itération passée et planifier l'itération suivante.
Le « reste à faire » pouvant évoluer selon l'état d'avancement, les difficultés rencontrées et les alternatives possibles, sans trahir les bases essentielles du cahier des charges initial.

### 1.1.6 Contexte

C'est dans ce contexte que j'ai dû mener un projet à la fois académique, en faisant des recherches sur les solutions existantes et à la fois très pratique, en réalisant une maquette. Au fur et à mesure de mes recherches, j'ai bâti un cahier des charges répondant aux contraintes données.

L'équipe SAIGA gère des programmes de recherches dont la clé est le traitement du signal.
Les algorithmes développés sont testés sur de vrais signaux issus de l'acquisition de données.
Ces données peuvent être fournies par les industriels eux-mêmes, dans le cadre d'un partenariat ou peuvent être obtenus en interne à partir des plates-formes simulant des systèmes.
En effet, le Gipsa-Lab dispose de nombreuses plates-formes servant de support physique pour exploiter les fruits des recherches menées au sein du laboratoire. Elles servent également d'outils pour les travaux pratiques des étudiants des écoles d'ingénieur de Grenoble INP.
Citant par exemple la plate-forme GOTIX [10], un banc de mesure d'un moteur à courant triphasé dont le système d'acquisition se trouve être un OR38, matériel fabriqué par l'entreprise OROS [11] où je travaille habituellement.

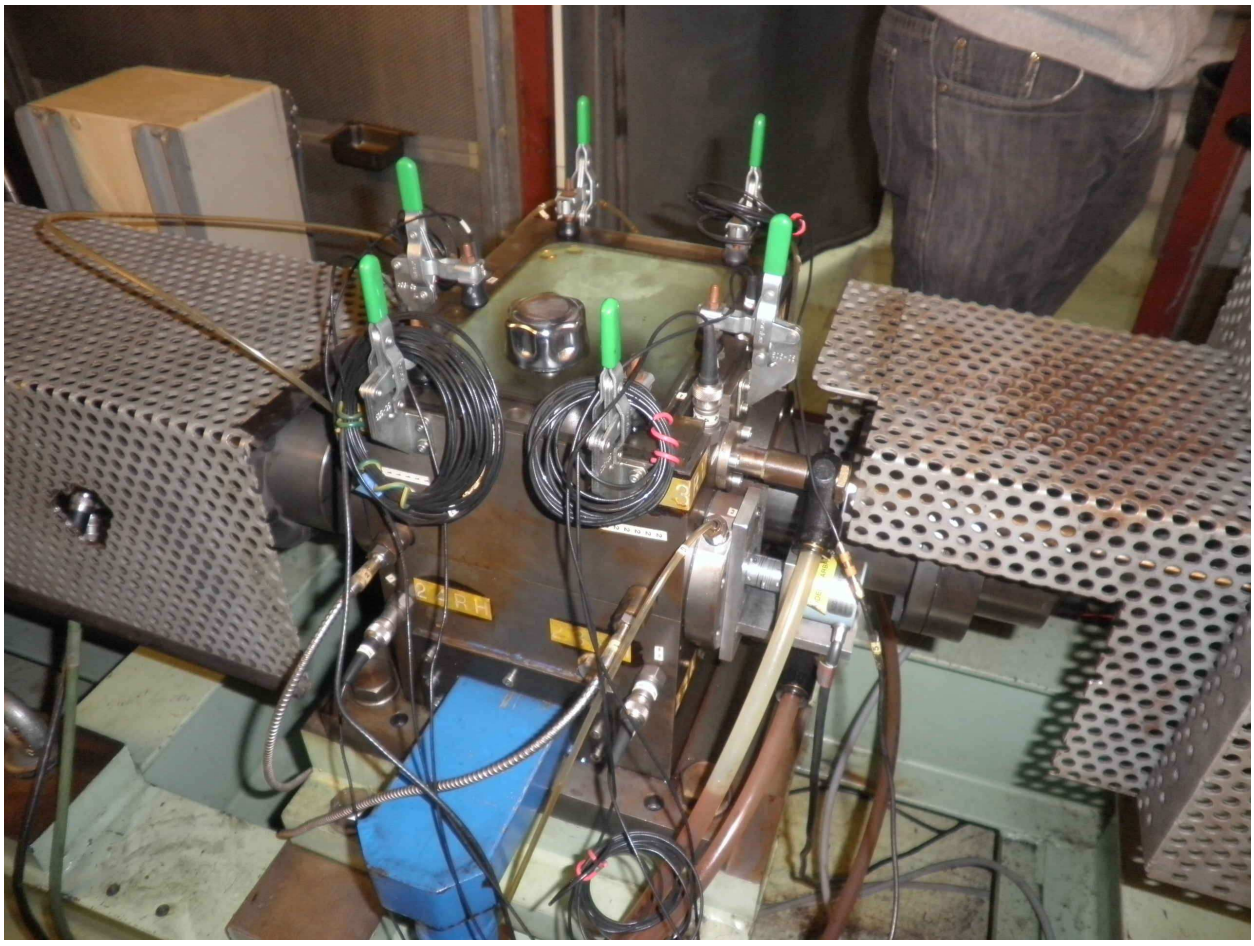


*Figure 1 : Détail du moteur de la plate-forme Gotix*

Le chapitre suivant, après quelques rappels techniques pour restituer le contexte, montrera le cheminement intellectuel qui a mené au choix des technologies pour réaliser une maquette sur mesure. Celle-ci sera dédiée à l'acquisition de données via sur un système embarqué de capteurs utilisant une technologie sans fil.

## 1.2 Réseaux de capteurs sans fil : Etat de l'art

Les réseaux de capteurs sans fil font partis d'un secteur de l'électronique plus vaste des systèmes d'acquisition de données physiques et de traitement du signal.
Pour comprendre l'analyse du besoin, je vais restituer le contexte en décrivant la chaîne d'acquisition du signal et les technologies employées.

### 1.2.1 Acquisition : Chaîne de mesure

Le schéma ci-dessous décrit une chaîne générique de mesure qui permet d'enregistrer sous forme numérique l'image d'une mesure analogique d'un signal physique. Toutes les liaisons sont dites « filaires » car chaque étape est reliée physiquement à l'étape suivante.

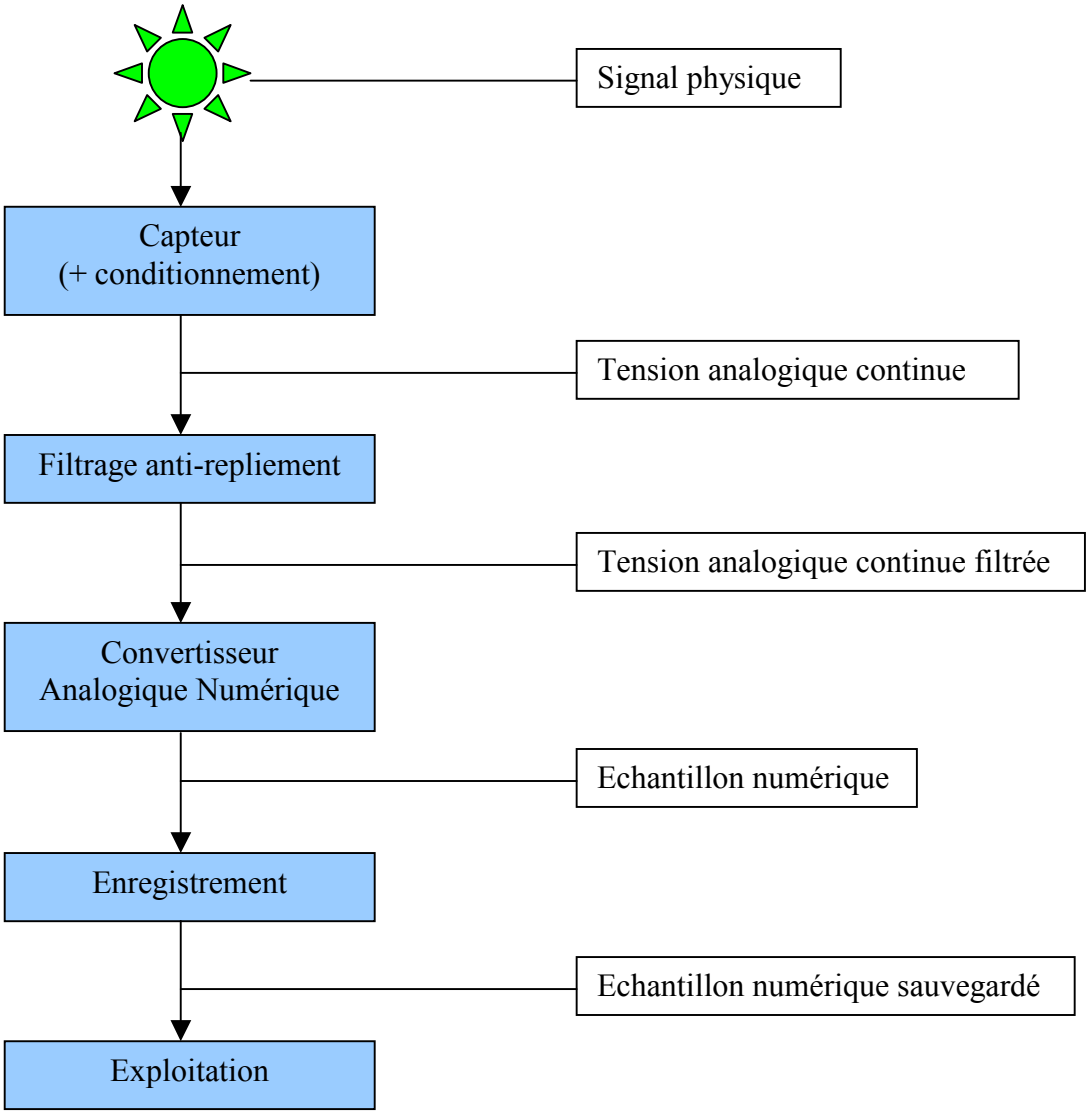



*Figure 2 : Principe général d'une chaîne de mesure*

Les échantillons enregistrés pourront être exploités par des techniques de traitement de signal (moyenne, FFT, etc.…) et les résultats enregistrés et/ou affichés sur un ordinateur.

### 1.2.2 Introduction du « sans-fil »

Dans le cas des capteurs sans fil, se pose la question : où introduire la rupture physique ?
Deux grandes solutions sont envisageables :

- soit traiter une tension analogique
- soit traiter un échantillon numérique

**Au niveau de la tension analogique**

Si l'on veut introduire la liaison sans fil au niveau de la tension analogique, celle-ci peut se placer soit directement après le capteur, soit après le filtrage.
Cette liaison consistera alors à moduler en amplitude un signal radio de fréquence fixe (AM).
La mise en œuvre sera délicate car elle nécessitera autant de fréquence radio qu'il y a de capteurs à exploiter.
Il faudra alors réaliser autant de paires d'émetteur/récepteur. L'intégration avec un ordinateur pourra se révéler compliquée si on veut gérer la réception simultanée de plusieurs capteurs.
C'est comme si on voulait écouter plusieurs fréquences radio en même temps !
Or le principe des liaisons radio analogiques consiste plutôt à sélectionner à l'aide d'un tuner une seule fréquence à la fois.
De plus, cette solution rend la restitution de la donnée sensible à l'environnement électromagnétique et radioélectrique qui peuvent perturber le signal.
L'amplitude **A** de la tension analogique qui module le signal radio en émission sera démodulé en réception pour donner une tension **B**, image de **A** modifiée par les perturbations environnantes.
Si celles-ci sont trop importantes, le signal sera fortement corrompu.

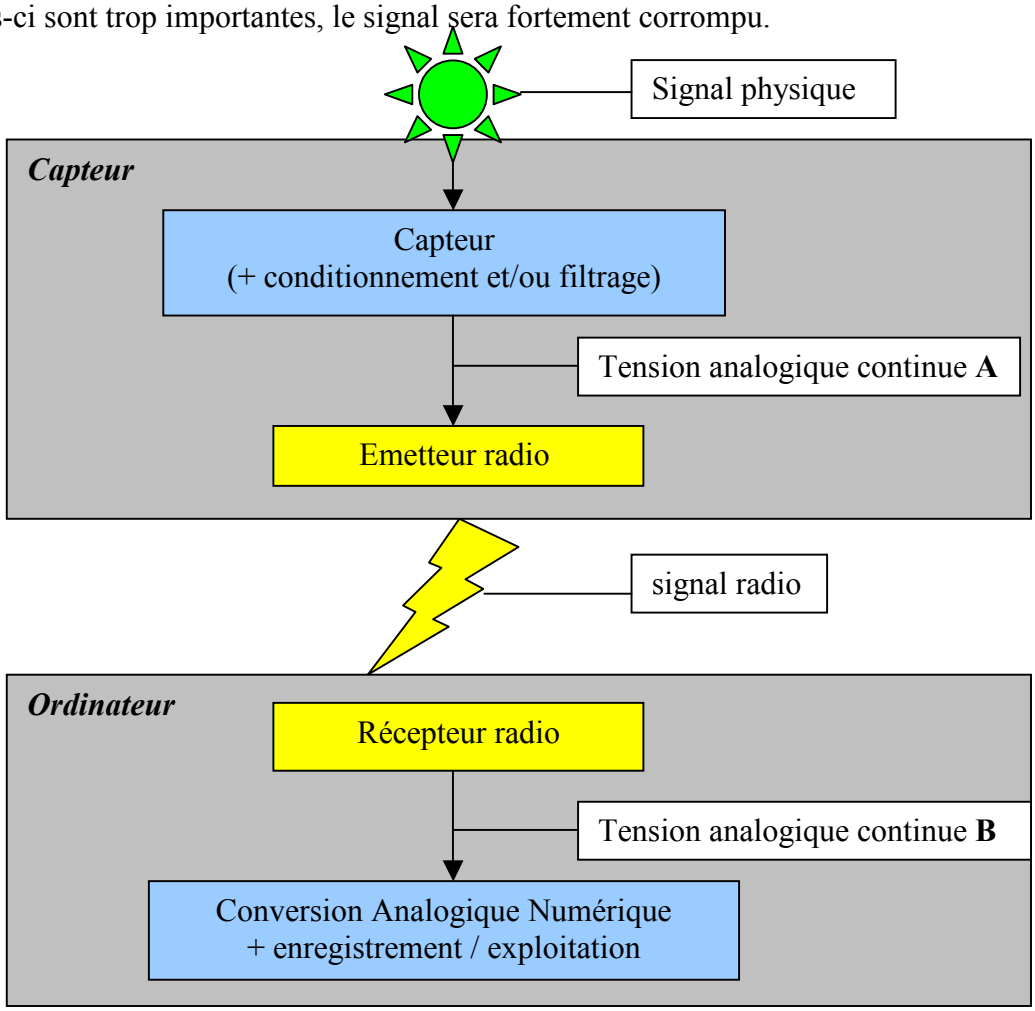



*Figure 3 : Liaison analogique sans fil*

**Au niveau de l'échantillon numérique**

La solution consistant à introduire une liaison sans fil, une fois le signal physique converti en une donnée numérique, paraît plus judicieuse.

En effet, la valeur analogique sera traitée au plus près de son acquisition physique sans subir de perturbation au cours de sa transmission, ou du moins celle-ci pourra être mieux contrôlée grâce aux techniques radios numériques qui intègrent des algorithmes de sécurisation de l'intégrité du signal.

De plus, cette solution a l'avantage de pouvoir introduire les étapes de traitement numérique au niveau du capteur comme la sélection de la fréquence d'échantillonnage ou le nombre de capteurs que l'on veut surveiller.

Nous verrons par la suite que ces opérations seront facilement réalisables avec un microcontrôleur.

Notons également, que les communications numériques peuvent être cryptées, un critère qui peut être important selon le niveau de confidentialité attendu.

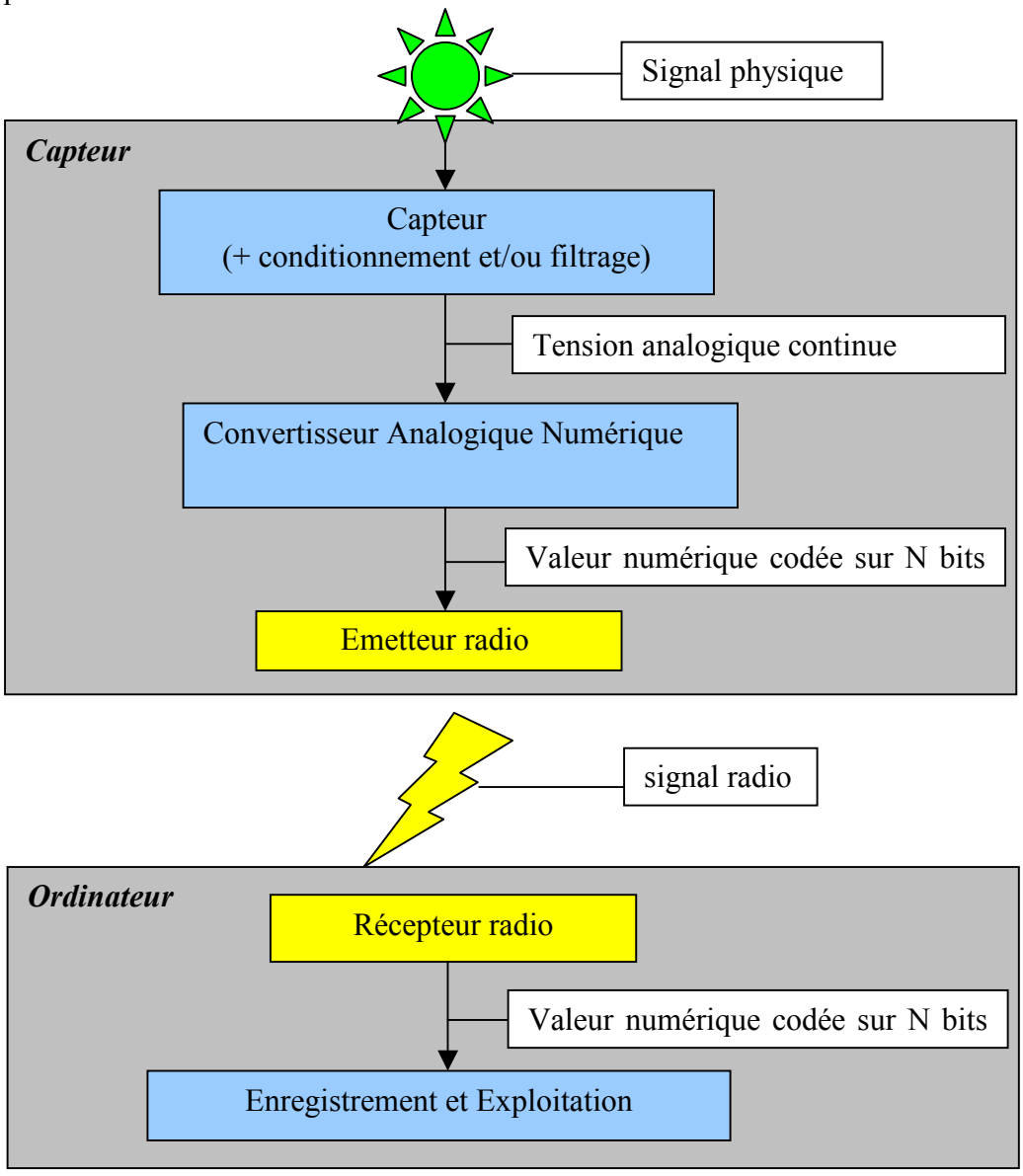



*Figure 4 : Liaison numérique sans fil*

**Signal Physique**

Par signal physique, on entend n'importe quel phénomène mesurable.
Ce peut être un phénomène ondulatoire comme la lumière ou le bruit, un phénomène mécanique comme les vibrations ou les déplacements, un phénomène électrique comme une tension ou un courant, ou un phénomène biologique comme les pulsations cardiaques.
Selon le type de phénomène étudié, il faut pouvoir récupérer les données à une cadence plus ou moins élevée selon la rapidité des variations.

**Bande passante**

La bande passante est conditionnée par :

- la fréquence d'échantillonnage (de quelques centaines de hertz à quelques dizaines de kilohertz)
- le nombre de capteurs (de quelques unités à plusieurs centaines)

Par exemple, certaines applications de géo-localisation nécessitent plusieurs centaines de capteurs avec une fréquence d'échantillonnage de 100 ou 200 Hz, alors que des applications « temps réel » d'analyse industrielle n'ont besoin que d'une dizaine de capteurs mais échantillonnés à 20 kHz.

**Analyse vibratoire**

Notre principale contrainte est la fréquence d'échantillonnage, conditionnée par la bande passante.
On désire une bande passante, équivalente à celle de l'acoustique, soit 20 kHz, ce qui impose d'échantillonner à 40 kHz au minimum (Théorème de Shannon-Nyquist).
Or la plupart des réseaux de capteurs sans fils décrits dans les références des fabricants, sont limités à 5 kHz.
En effet, le débit de données offert par la technologie radio utilisée conditionne et limite la bande passante (voir 1.2.4 « Standards pour les réseaux de capteurs sans fil »).
Si l'on veut récupérer le signal temporel issu d'un accéléromètre triaxes, échantillonné à 40 kHz, sur 16 bits, on aura besoin d'un débit de près de 2 Mbps ($3 \times 40000 \times 16 = 1.96 \times 10^6$)
Nous verrons que les principales technologies radio des capteurs sans fil sont basées sur le Zigbee limité à 250 kbps. Nous nous orienterons plus vers une technologie de type Wi-Fi autorisant des débits de 1 à 54 Mbps.

### 1.2.3 Types de Capteurs

Les principales grandeurs physiques utilisées dans les gammes de fréquence dites acoustiques (avec une fréquence d'échantillonnage de quelques centaines de hertz à quelques dizaines de kilohertz) sont mesurées à l'aide des capteurs de type :

- Accéléromètres
- Microphones
- Jauges de contraintes
- Mesure de vitesse angulaire
- Capteur de pression acoustique

Elles peuvent être complétés par des mesures nécessitant une fréquence d'échantillonnage plus faible, car les grandeurs physiques varient lentement :

- Tension, courant
- Température, humidité

**Contraintes d'alimentation**

Si on souhaite un système autonome alimenté par pile (autour de 3V), on évitera des capteurs type IEPE (Integrated Electronics Piezo Electric) car ils sont alimentés par une source de courant de quelques mA obtenue avec une tension de 9 à 12V. Il faudrait alors développer un étage électronique supplémentaire pour intégrer un convertisseur DC-DC. Les accéléromètres de la plate-forme Gotix sont de ce type (B&K 4391 [12]).
En règle générale, les accéléromètres possédant une large bande passante sont lourds et encombrants. Plus ils sont réduits, plus la bande passante est faible.
Néanmoins, la technologie des MEMS (Micro Electro Mechanical System) change la donne… il est enfin possible d'avoir des capteurs miniatures avec une bande passante de quelques kilohertz (Colybris [13], PCB [14], Analog Devices [15]). Toutefois, les accéléromètres ne sont que mono-axe, il faudra en positionner trois, avec une très grande précision géométrique, pour obtenir une fonction tri-axiale. Ils sont également conçus pour fonctionner sous une alimentation de 3 à 6V. Aujourd'hui, les performances des capteurs MEMS dépassent celles des capteurs classiques et rentrent dans la conception des objets grand public (airbag de voiture, smartphone, console de jeu) [16].

**Capteurs analogiques**

Ils délivrent une tension proportionnelle à la grandeur physique mesurée.
Pour obtenir une valeur numérique, il faut l'interfacer sur un Convertisseur Analogique Numérique (CAN).
Ils nécessitent une électronique associée (pont diviseur pour adapter le gain, offset pour adapter le zéro, filtre passe-bas selon la fréquence d'échantillonnage choisie…)

**Capteurs numériques**

Ils délivrent directement une valeur numérique que l'on peut lire dans un registre par une liaison numérique. Les deux protocoles de communication les plus utilisés sont l'$I^2C$ et le SPI.

Pour se familiariser avec leur fonctionnement, nous allons choisir de mettre un œuvre un capteur de type analogique (voir 2.6 « Mise en œuvre d'un capteur analogique : Microphone ») et un capteur de type numérique (voir 2.7 « Mise en œuvre d'un capteur numérique : accéléromètre »).

### 1.2.4 Standards pour les réseaux de capteurs sans fil

Nous limiterons volontairement notre recherche aux standards radio-fréquences.
Il existe également des technologies optiques, comme les Infra Rouge (IrDA : Infra Red Data Association) mais l'inconvénient majeur est qu'il ne faut aucun obstacle entre les deux éléments qui communiquent. De plus, la portée est généralement limitée à quelques mètres. Notons néanmoins que cette solution a été étudiée dans une thèse [17] .

**Zigbee / 802.15.4**

Historiquement, le standard 802.15.4 (sur lequel est basée la technologie Zigbee) est le standard de référence pour les capteurs sans fil. La bande passante du signal analysé est limitée par un débit maximal de 250 kbps, ce qui en fait un standard de choix pour des applications qui analysent des signaux échantillonnés à basses fréquences (par exemple en sismique, à quelques centaines de hertz).
L'avantage est que les capteurs sans fils consomment peu. Ils ne nécessitent que quelques volts pour fonctionner (typiquement des piles bâtons de 1,5V ou boutons de 3V).
De nombreuses solutions « open source » existent en Zigbee (Mote, sensor node) largement utilisées dans les milieux universitaires.
L'établissement des protocoles de communication « Wireless Hart » [18] et « ISA 100.11a Wireless Systems for Industrial Automation : Process Control and Related Applications » ont été conçus pour faciliter l'interopérabilité des capteurs sans fils dédiés aux applications industrielles.
Ce sont deux standards concurrents, basés sur la norme 802.15.4 (et donc limité à 250 kbps) qui s'affrontent… Aucun ne s'est encore imposé et les utilisateurs potentiels sont dans l'expectative [19].

**Bluetooth**

Très déployé dans les périphériques informatiques et nomades (clavier, oreillette…) le Bluetooth consomme trop et sa portée est faible (quelques mètres).
Néanmoins, de nouveaux produits Bluetooth Low Energy (initié par le Wibree de Nokia [20]) sont apparus sur le marché fin 2011 et offrent un débit pouvant atteindre 1Mbps pour une consommation 10 fois plus faible que le Bluetooth [21]. Le magazine Electronique Pratique propose d'ailleurs un montage d'acquisition de quatre voies analogiques en Bluetooth, avec une bande passante utile de 70 Hz et un échantillonnage de 1 kHz sur 10 bits [22].

**Radio**

Dans la bande des 400 ou 900 MHz, la bande passante est faible mais la propagation efficace vis à vis des obstacles, autorisant des portées de plusieurs centaines de mètres, voir kilomètres.
Par ailleurs, les contraintes réglementaires françaises limitent les bandes de fréquence utilisables sans autorisation et/ou limite la puissance. Cette technique est très déployée aux Etats Unis.

**Téléphonie**

Les téléphones portables sont très intéressants au niveau des débits, comme le prouve l'engouement des smartphones et ses applications Web.
Mais pour utiliser ce moyen de communication, il faut forcément passer par un opérateur téléphonique et gérer un abonnement (type carte SIM).

**UWB**

Cette technologie est en cours de déploiement mais difficile à mettre en œuvre et plutôt utilisée pour de la transmission d'images/vidéo. En effet, contrairement aux autres technologies radios, celle-ci utilise une technique de modulation par impulsion, qui nécessite une plus grande bande passante au niveau du spectre (> 500 MHz) pour obtenir une résolution temporelle de 2 nanosecondes. La synchronisation de la réception est délicate mais les travaux en cours sont

encourageants [23]. Néanmoins, au vu des débits promis par cette technologie, elle pourra constituer une solution intéressante dans le futur, citant par exemple la société DecaWave qui se positionne sur ce marché [24] suivant la norme 802.15.4a qui pourrait offrir un débit de 27 Mbps.

**Wi-Fi**

La tendance actuelle pour gérer des débits importants de données s'oriente vers le Wi-Fi 802.11.
Historiquement, cette technologie requiert une source d'alimentation importante, mais la norme « low power » offre un fonctionnement plus adapté aux réseaux de capteurs.
Les solutions Wi-Fi Low Power actuelles sont limitées à 1 ou 2 Mpbs, alimentées entre 3 et 5V (piles AAA, piles boutons, batteries plates lithium-ion).
Pour obtenir un débit de 54 Mbps, il faudra utiliser une solution plus exigeante en énergie (batterie 12V typiquement).
Nous reparlerons plus en détail du Wi-Fi lors de sa mise en œuvre dans le chapitre 2.8 « Mise en œuvre de la liaison Wi-Fi ».

| Nom | Norme IEEE | Débit Max | Fréquence | Portée (sans obstacle) |
|---|---|---|---|---|
| Zigbee | 802.15.4c | 250 kbps | 2,4 GHz | 50 à 100 m |
| Bluetooth | 802.15.1 | 1 Mbps | 2,4 GHz | 10 m à 30 m |
| Radio | *(propriétaire)* | 150 kbps | 400 – 900 MHz | 100 m à 1 km |
| UWB | 802.15.3a | 100 Mbps | 3,1 à 10,6 GHz | 30 m |
| Wi-Fi | 802.11 | 54 Mbps | 2,4 GHz | 100 à 200 m |

*Tableau 1 : Récapitulatif des normes radio-fréquences*

### 1.2.5 Exemples d'applications

L'apport du sans-fil trouve des débouchés dans de nombreux domaines d'applications.
Les paramètres principaux à prendre en compte sont :

- le nombre de capteurs
- la portée / distance qui les sépare
- le débit des données, la fréquence à laquelle on les récupère
- la consommation qui conditionne l'autonomie, le mode d'alimentation et de recharge

Pour simplifier la présentation des exemples, nous utiliserons la convention suivante :

| | * | ** | *** |
|---|---|---|---|
| **Nombre de capteurs** | Quelques unités | Quelques dizaines | Quelques centaines |
| **Portée** | < 10m | de 10 à 100 m | > 100m |
| **Fréquence** | < 100 Hz | 100 à 1 kHz | > 1 kHz |
| **Consommation** | faible | moyenne | importante |

**Dans la santé**

Dans le domaine de la surveillance médicale, une liaison sans-fil donne une plus grande liberté de mouvements au patient par la tolérance d'un dispositif moins contraignant.
Ces dispositifs sont particulièrement adaptés pour le maintien d'un patient à domicile, les données physiologiques étant enregistrées et transmises selon le programme défini par l'équipe médicale, suffisamment réactive pour réagir en cas d'alerte (chute, rythme cardiaque…).
L'essor des *smartphones*, avec leurs capteurs intégrés (accéléromètres, gyroscopes) autorisent le développement d'applications destinées au bien-être (surveillance du sommeil, podomètre …).
A l'intérieur même des structures médicales, le sans-fil supprime les câbles gênants lors d'une opération améliorant le confort de la pratique du chirurgien et de l'anesthésiste [25].
Dans le domaine de l'étude du corps humain, en mesurant les vibrations subies pour adapter l'environnement de travail et diminuer les troubles musculo-squelettiques [26].

| Nombre de capteurs | Portée | Fréquence | Consommation |
|---|---|---|---|
| * | * | * à ** | * |

**Dans le bâtiment**

Typiquement, dans les applications de domotiques pour optimiser les actions (ouverture de volets, chauffage) et adapter la consommation d'énergie en fonction de la température, du taux d'humidité…

| Nombre de capteurs | Portée | Fréquence | Consommation |
|---|---|---|---|
| * | ** | * | * |

**Dans l'énergie**

Les réseaux de capteurs sans fil sont déployés dans les « Smart Grids » pour optimiser la production, la distribution et l'utilisation de l'énergie électrique.
Les fournisseurs d'électricité peuvent également effectuer des télé-relevés de la consommation en temps réel.

| Nombre de capteurs | Portée | Fréquence | Consommation |
|---|---|---|---|
| ** | *** | * | * |

**Dans la géologie**

Pour détecter les mouvements du sol et prévoir les séismes, on dissémine des capteurs à grande échelle sur des milliers de kilomètres carrés. La difficulté réside alors dans la communication avec tous ces capteurs sur de telles distances, qu'on surmonte à l'aide d'une topologie en arborescence.

On peut également déployer des capteurs pour surveiller des phénomènes géologiques tels que le déplacement des glaciers, comme dans le projet PermaSense [27].

| Nombre de capteurs | Portée | Fréquence | Consommation |
|---|---|---|---|
| ** | *** | * | * |

**Dans l'environnement**

Pour mesurer des données météorologiques comme la température, la pression atmosphérique, la pluviométrie à l'aide d'une station météo sans fil qui envoie périodiquement les données.

On peut également surveiller le débit et le niveau de cours d'eau ou de cuves contenant des liquides pour déclencher des alarmes.

| Nombre de capteurs | Portée | Fréquence | Consommation |
|---|---|---|---|
| * | ** | * | * |

**Dans la construction**

Pour la surveillance des vibrations et l'analyse de structure de bâtiments et d'ouvrages d'art. On instrumente avec des accéléromètres dont la bande passante est de plusieurs centaines de hertz et la fréquence d'échantillonnage de plusieurs kilohertz. On constate que l'utilisation du sans-fil pour restituer avec fidélité ce type de données n'est pas simple [28].

| Nombre de capteurs | Portée | Fréquence | Consommation |
|---|---|---|---|
| * | ** | *** | * |

**Dans les usines**

Pour la surveillance des machines (CBM : Condition-Based Maintenance [29]) instrumentées avec des capteurs. L'intérêt du sans-fil est que l'on peut facilement ajouter un capteur sur une nouvelle machine à surveiller ou pour contrôler une autre grandeur physique, sans avoir un tas de fils qui traversent l'usine.

A la différence des autres domaines d'application présentés, celui-ci requiert une fréquence d'échantillonnage bien plus élevée si l'on veut suivre les variations du signal temporel des grandeurs physiques qui changent rapidement comme les bruits et les vibrations d'un moteur ou d'une machine tournante.

Notre projet sera destiné à ce type d'application.

| Nombre de capteurs | Portée | Fréquence | Consommation |
|---|---|---|---|
| * | * | *** | * |

### 1.2.6 Cahier des Charges

Comme nous venons de le voir, les réseaux de capteurs sans fils répondent à un grand nombre d'applications. Il faut donc se poser les bonnes questions et y répondre pour peaufiner le cahier des charges.
A ce sujet, National Instruments décrit une bonne synthèse des paramètres à étudier [30] [31] :

- Quel est le débit nécessaire ?
- Quel est le type de capteurs, leur nombre, leur topologie ?
- Quelle doit être la portée ?
- Quelle doit être l'autonomie ?
- Quelles fonctionnalités doit apporter le sans-fil ?

Les paragraphes précédents nous ont aidé à cerner le type d'application visé et à déterminer les principes suivants :

**Sans fil**
C'est la fonction principale de notre système.
Il faut choisir une technologie radio qui remplace la liaison filaire habituelle de ce type de matériel, sans sacrifier la robustesse et la fiabilité.
Sa bande passante dépend de la vitesse d'acquisition et du nombre de voies.

**Portabilité**
Le facteur de forme est important : on doit arriver à miniaturiser le plus possible avec les contraintes de calcul et d'alimentation.
Le système doit être peu encombrant pour pouvoir être transporté facilement (de l'ordre de quelques centimètres cubes).

**Autonome**
Concernant le type d'application envisagée, on peut considérer que l'alimentation n'est pas un problème et peut être fournie sur place (machine tournante dans un atelier).
Dans ce cas, une solution sans fil se rapproche plus d'une solution filaire : l'acquisition du signal et sa digitalisation peuvent se faire au plus près de la mesure physique, avec une alimentation similaire (secteur ou batterie). Seul le canal de transmission des données entre le capteur et l'interface chargée d'analyser les signaux, typiquement un ordinateur, change : il sera sans fil.
Mais pour le critère de portabilité ci-dessus, avoir un système qui se passe complètement de branchement est un atout.
On envisagera alors une pile rechargeable (de quelques volts, typiquement 3 ou 5 V) qui donnera une autonomie de l'ordre de quelques heures, de façon à pouvoir faire une démonstration en cours ou une mesure sur site. On peut changer la pile ou recharger la batterie régulièrement, le matériel n'étant pas dans un endroit distant ou inaccessible.

**Déploiement**
Le système doit être facile à déployer, tant en terme d'installation que d'utilisation.
Les capteurs doivent se manipuler et se positionner facilement, sans être invasifs.
Le logiciel d'acquisition doit être intuitif et paramétrable par une interface utilisateur et ne pas dépendre du système d'exploitation de l'ordinateur (Windows, Unix).

**Portée**
La distance entre les capteurs et l'ordinateur doit être de plusieurs mètres, voir dizaines de mètres, ce qui correspond à un réseau dit Wireless Personnal Area Network (WPAN)

**Convertisseur Analogique Numérique**

Pour les applications visées, qui requièrent une bonne précision dans les fréquences acoustiques, il faudra choisir un Convertisseur Analogique Numérique (CAN) avec :

- Précision = 12 bits minimum
- Fréquence d'échantillonnage = 20 kHz minimum

Théoriquement, une fréquence d'échantillonnage de 20 kHz permet d'observer des signaux avec une bande passante de 10 kHz (Théorème de Shannon-Nyquist). En pratique, la bande passante sera plutôt de 5 kHz.

**Débit**

Le nombre de capteurs et les paramètres du CAN conditionnent le débit.
Attention, pour le débit des données, il faut compter 30% du débit théorique de la technologie radio utilisée.

**Capteurs**

On doit prévoir d'interfacer plusieurs types de capteurs pour être polyvalent.
On imagine ainsi placer des connecteurs sur lesquels on branchera les capteurs désirés.
Le système sera évolutif et ouvert pour développer d'autres fonctionnalités dans le futur.
Les capteurs devant mesurer les vibrations (accéléromètre triaxe), les grandeurs électriques (capteurs de courant), les phénomènes acoustiques (microphone), les données environnementales (température, pression).
Pour que l'alimentation des capteurs soit compatible avec notre solution autonome sur pile de quelques volts, ils ne devront pas être de type IEPE.

**Synchronisation**

On doit être capable de récupérer de façon synchrone les données issues de plusieurs modules, plusieurs capteurs physiques pouvant être connectés sur chaque module.
Il faut donc à la fois prévoir l'acquisition simultanée par le CAN sur chaque module de toutes les voies connectées (fonction « bloqueur »), mais aussi une périodicité synchrone d'agrégation des données issues de chaque module.

**Temps réel**

La transmission doit s'effectuer en temps réel, sans mise en mémoire tampon ou compression de données.
On peut admettre une latence de quelques millisecondes.
Dans les applications actuelles, la fréquence d'échantillonnage dépasse rarement 5 kHz afin de permettre un débit suffisant pour l'envoi des données en temps réel.
On avait envisagé d'enregistrer les données avant de les envoyer, sur une SDCard par exemple, pour enregistrer un nombre de données suffisant.
La compression des données (cf. MP3 pour l'audio) aurait pu diminuer le débit également, mais il faut prendre en compte le temps et la puissance de calcul pour l'appliquer.

**Topologie**

Chaque module devra être capable de gérer 4 ou 6 voies analogiques.
Le système complet devra gérer 3 modules connectés selon la topologie dite « en étoile ».
Dans la terminologie réseau, un module est un nœud.
L'ordinateur fera office de chef d'orchestre ou nœud central.

**Budget**

Le coût doit être faible, car c'est un démonstrateur pour des étudiants ou des scientifiques. Il n'a pas vocation à être vendu commercialement.
Le système complet avec les capteurs doit revenir à quelques centaines d'euros.

**Calcul du débit**

$$Débit = Noeud \times Capteurs \times \frac{Echantillon}{seconde} \times \frac{Bits}{Echantillon}$$

Avec notre cahier des charges :

- Nœud = 3
- Capteurs = 6
- Echantillons/seconde = 20 000
- Bits/Echantillon = 12

Pour 1 nœud :

Débit = $1 \times 6 \times 20k \times 12 = 1{,}44$ Mbps

Pour 3 nœuds :

Débit = $3 \times 6 \times 20k \times 12 = 4{,}32$ Mbps

Ce qui conforte notre choix pour le Wi-Fi.
Le Zigbee n'aurait clairement pas été suffisant avec ces 250 kbps.

**Objet de notre application**

Plutôt qu'un « Wireless Sensor Network » (WSN), on veut plutôt construire un « Remote Data Acquisition » (acquisition de données à distance).
Mon projet colle au plus près de l'actualité et se nourrit des nouveautés au fur et à mesure de leur sortie (voir les solutions industrielles ci-dessous), en suivant les informations délivrées par les sites spécialisés, comme « Sensors Magazine » [32].
Un exemple type d'application serait un système d'acquisition synchrone multi-capteurs (multi-triaxes par exemple).
La maquette se borne simplement à réaliser l'acquisition. Le traitement du signal se fera en post-traitement sur les fichiers enregistrés sur l'ordinateur.
L'énergie nécessaire à l'acquisition et à la transmission des signaux est importante, notre maquette s'apparentant à un oscilloscope sans fil…
Pour des raisons de coût, une solution « Open Source » sera préférée, basée sur des modules faibles coût, basse consommation.
La consommation de courant augmentant avec le débit des données, il faudra réussir à concilier ces apparents contraires.

**Récapitulatif du cahier des charges**

| Paramètre | Valeur |
|---|---|
| Alimentation | par pile, inférieure à 5Volts |
| Autonomie | quelques heures |
| Dimensions | quelques centimètres cube |
| Portée | 10 à 100 m |
| CAN | 12 bits minimum<br>Fe = 20 kHz minimum |
| Caractéristiques des capteurs | bande passante : 5 kHz (Fe/4)<br>non invasifs<br>miniatures (type MEMS) |
| Types de Capteurs | accéléromètres<br>microphones<br>capteurs de courant<br>capteur de température |
| Sans-Fil | technologie radiofréquence<br>débit > 1 Mbps<br>topologie Star |
| Coût | quelques centaines d'euros |

*Tableau 2 : Récapitulatif du cahier des charges*

### 1.2.7 Solutions existantes

Sans prétendre être exhaustifs, voici quelques solutions existantes se rapprochant de ce que l'on veut faire. On trouve sur le marché du matériel industriel, performant mais onéreux, et des solutions « open-source », plus abordables mais dont une partie du développement est à la charge de l'utilisateur.

**Matériel industriel**

Nous nous limiterons aux fabricants proposant des solutions « Condition-Based Monitoring » (surveillance) avec des bandes passantes élevées (typiquement 20 kHz, échantillonnés à 51,2 kHz) et utilisant des technologies radio ou Wi-Fi.

InCheck Technologies

Son « InBox™ Wireless Data Acquisition Modules » [33] dédié au diagnostic des machines tournantes, échantillonne jusqu'à 16 voies à 51,2 kHz sur 16 bits, mais n'enregistre que 10 secondes de signal transmis par Wi-Fi … avant de répéter l'opération.



SKF

Cette société spécialisée dans l'instrumentation des procédés rotatifs (roulement, palier, mécatronique) propose le Multilog On-line System WMx [34] qui collecte les voies échantillonnées jusqu'à 102,4 kHz sur 16 ou 24 bits mais qui envoie les données par Wi-Fi en différé.

InteSens

Cette PME française propose un kit de diagnostic « InteLog » [35] basé sur une technologie radio (bande 866-915 MHz) et un débit de 100 kbps.
L'acquisition se fait à 51,2 kHz sur 18 bits mais sans transmission des données temporelles. Seules les valeurs scalaires moyennées sur une bande de fréquence (AVG et AVN) sont envoyées, complétées par un système d'alarmes.

National Instrument

Ce fabricant propose du matériel haut de gamme dans les réseaux de capteurs sans fil [36].
Par exemple, le modèle WLS-9234 [37] échantillonne 4 voies à 51,2 kHz sur 24 bits et transmet en Wi-Fi le signal temporel en continu.
Mais le coût total de cette solution (car il faut également se procurer le logiciel d'acquisition associé LabView ou un Toolkit) est de plusieurs milliers d'euros…

Toutes ces solutions sont robustes mais ne satisfont pas les critères que nous nous sommes fixés :

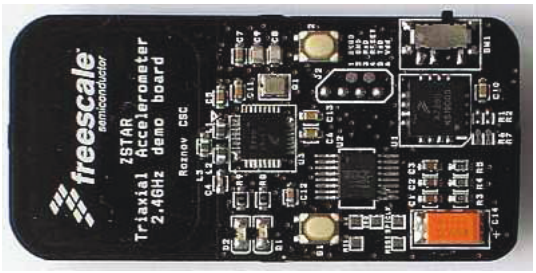
- éviter les capteurs IEPE
- transmettre en continu le signal temporel
- pour un coût « raisonnable » de quelques centaines d'euros

**Solutions « Open Source »**

Les solutions dites « Open Source » proposent des briques technologiques pré-assemblées dont les schémas électroniques et les codes sources sont disponibles et utilisables pour un coût modique. La philosophie est celle d'une communauté qui partage librement les applications développées et les améliorations apportées.
La technologie sans-fil de référence pour les capteurs est sans conteste le ZigBee.
En effet, nous le retrouvons dans des solutions éprouvées comme TelosB [38], TMote Sky [39] ou MICAz [40], largement décrit dans la littérature.

FreeScale

Le Gipsa-Lab a fait l'acquisition d'un module ZStar de FreeScale [41] avec un accéléromètre triaxiale et un microcontrôleur.
Le débit brut est de [*3 axes × 16 capteurs parallélisables × CAN 10 bits × 120 Hz d'échantillonnage*] 57,6 kbits/s compatible avec le débit du Zigbee de 250 kbits/s. La limite de 120 Hz est un compromis entre la vitesse et le nombre de capteurs… trop faible pour notre application. Mais il démontre qu'il est possible de faire fonctionner un capteur sans fil avec une simple pile bouton de 3V sur une platine de quelques centimètres carrés.




Arduino

**Arduino** [42] propose une plate-forme de développement logiciel et matériel, pour construire rapidement des cartes électroniques prototypes. L'originalité d'Arduino, c'est son approche « Orienté Objet » pour interfacer des briques ayant chacune une fonctionnalité spécifique (par exemple le capteur physique, la transmission des données, l'enregistrement des données).
Historiquement, les platines cibles étaient à base de microcontrôleur 8 bits (Atmel ***ATM168*** et ***ATM328***) comme l'Arduino Uno [43]. En y adjoignant un accéléromètre et un module XBee® (Zigbee), on peut réaliser un accéléromètre commandé à distance [44]. Si le principe correspond à l'idée de notre projet, ses caractéristiques techniques sont limitées : le convertisseur n'a que 10 bits et on ne contrôle pas la fréquence d'échantillonnage.


La plupart des projets sans fils utilisent des composants ***XBee®*** avec la norme 802.15.4/Zigbee comme la réalisation de télécommande domotique [45][46]. La platine Arduino Funnel I/O a été spécialement adaptée pour recevoir un composant XBee® [47].
Ce module XBee® a évolué et se décline désormais en modules radio et Wi-Fi.
La société espagnole Libellium s'en est fait une spécialité et propose des plate-formes de capteurs sans fils [48].

OpenPicus

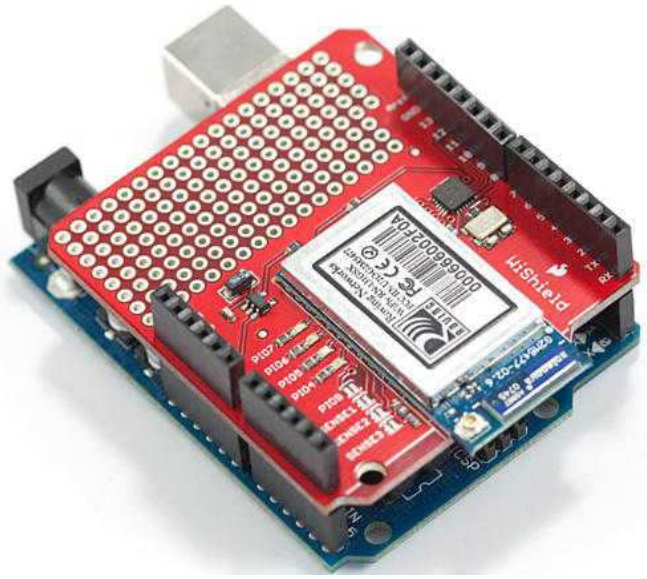
Cette société propose un module FlyPort [49] intégrant un processeur PIC 16 bits et une interface Wi-Fi 802.11b de chez Microchip.
L'avantage est que son intégration très poussée en fait un candidat parfait en terme de taille, comme le montre la photo ci-contre.
L'inconvénient est que le CAN ne possède que 4 entrées 10 bits alors que l'on souhaite 12 bits minimum.


STM32W

ST propose un kit de développement [50] basé sur son composant STM32W intégrant un processeur ARM® Cortex™ 32bits et un module radio 2,4 GHz 802.15.4 (Zigbee).
L'avantage du processeur, c'est qu'il propose un CAN pour 6 entrées 12 bits.
Toutefois, comme nous l'avons vu auparavant, le Zigbee n'est pas suffisant pour le débit dont on a besoin.

Maple & WiFly

LeafLabs a développé un module *Maple* [51] sur le format Arduino mais en intégrant un microcontrôleur 32 bits en lieu et place du microcontrôleur 8 bits d'origine. Ce microcontrôleur 32 bits est un STM32F103, de la même famille que celui de la solution décrite ci-avant pour le STM32W.
Cette solution a donc le double avantage de posséder un processeur adapté à nos besoins en terme de performances pour les entrées analogiques (CAN 6 entrées 12 bits) et de pouvoir s'interfacer avec les platines additionnelles Arduino. Il existe une platine Wi-Fi compatible *pin* à *pin*, appelée *WiFly* [52].
Les 2 platines s'interfacent en les superposant l'une sur l'autre à l'aide de connecteurs.



### 1.2.8 Solution choisie

Le choix technologique devra porter sur un système dit « ouvert », ce qui signifie qu'il doit être :
- gratuit, avec un code source accessible et une licence ouverte (type open GPL)
- évolutif et adaptable à nos besoins
- collaboratif pour échanger avec des pairs

Au vu des contraintes de coût et de délais, j'ai privilégié une solution de type Arduino.
En effet, cette philosophie consistant à acheter des briques électroniques à juxtaposer est très intéressante pour effectuer du prototypage rapide.
La programmation « clés en main » avec une interface et un langage intuitif accélère également la mise au point des fonctions que l'on souhaite réaliser.

C'est donc la dernière solution décrite dans le paragraphe précédent (*Maple & WiFly*) qui a été retenue.

Mon choix du processeur STM32 a été validé en interne par un ingénieur en informatique embarqué qui l'utilise déjà dans plusieurs applications. Le support technique en sera facilité.
La société Hikob OpenLab [53] en a également fait la base de ses capteurs sans fil Zigbee.
Ce microcontrôleur a l'avantage d'être proposé sous la forme d'une brique compatible Arduino par LeafLabs dénommée « *MAPLE* ».
Grâce à cette compatibilité, on peut y adjoindre une brique Wi-Fi, dénommée « *WiFly* ».

Le module *WiFly* contient un composant Wi-Fi de chez Roving Network, le RN131C.
Grâce à sa « faible » consommation (moins que le Wi-Fi classique, mais plus que du Zigbee ou du Bluetooth), il peut être alimenté en 3,3V. Sa portée (100m sans obstacle) et son débit (1Mb au moins) correspondent au cahier des charges.
De plus, la platine *WiFly* possède une plage libre pour souder des composants. Elle sera très utile pour réaliser l'interfaçage avec nos capteurs sans nécessiter de développement électronique supplémentaire.
Le Wi-Fi est très largement diffusé, ce qui simplifiera son déploiement et son accès aux ordinateurs par clé USB.

Il ne reste plus qu'à ajouter des capteurs pour traiter des signaux réels.
Pour évaluer le potentiel de cette solution, on mettra en œuvre un capteur analogique de type microphone et un capteur numérique de type accéléromètre.

L'autonomie de la maquette sera assurée par une pile plate Lithium-Polymère de 3,7V.

# 2 Mise en œuvre d'une première maquette

## 2.1 Introduction

L'objectif de la réalisation de cette première maquette est de tester la faisabilité de la solution retenue en mettant en œuvre :

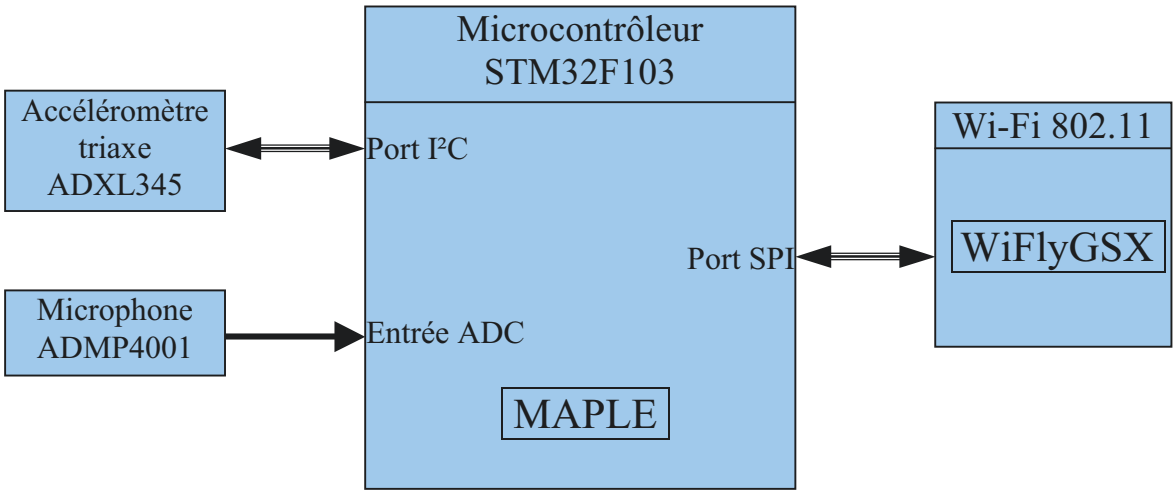
- un microcontrôleur 32 bits
- un capteur numérique
- un capteur analogique
- une liaison Wi-Fi

En utilisant, la philosophie Arduino, on va baser le travail sur un prototype rapide à l'aide de platines électroniques déjà montées que l'on assemblera, un peu comme des Lego®.

## 2.2 Architecture

On assemblera :

- une platine « *Maple* » à base d'un microcontrôleur 32 bits STM32F103
- une platine « *WiFly* » pour la connexion Wi-Fi
- une platine « Accéléromètre triaxe » numérique pour enregistrer les vibrations
- une platine « Microphone » analogique pour enregistrer le son
- une batterie rechargeable lithium-polymère de 3,7V, régulée à 3,3V pour alimenter tous les composants




| Capteur | Type | Bande passante | Nb bits conversion |
|---|---|---|---|
| Accéléromètre triaxe ADXL345 | Numérique (CAN intégré) | Paramétrable, de 3,125 à 1600 Hz | Paramétrable, de 10 à 13 |
| Microphone ADMP4001 | Analogique (CAN par le µC) | 100 Hz à 15 kHz | 12 |

## 2.3 Platine Maple

C'est le cœur du système, incluant le microcontrôleur STM32, la gestion de l'alimentation et les connecteurs pour les périphériques d'entrées/sorties.

**Références :**
Carte : Maple
Fabricant: Leaflabs
Composant principal: microcontrôleur ARM Cortex M3 **32 bits** STM32F103RB.

**Caractéristiques techniques [54] :**
- Microcontrôleur : 32 bits STM32F103RB (ARM Cortex M3)
- Vitesse d'horloge : 72 MHz
- Tensions d'alimentation : 3,3V
- Tension d'entrée : 3,0V-12V
- Nombre d'entrées/sorties : 39 (GPIO)
- 16 entrées analogiques
- Convertisseur ADC 12-bit ((Analog-to-Digital Converter)
- 15 ports PWM résolution 16-bit
- 1 port USB dédie à la programmation
- Interface JTAG externe (USB)
- Mémoire Flash : 128 K
- Mémoire SRAM : 20 K
- SPI (Serial Peripheral Interface) et $I^2C$ (Inter Integrated Circuit) intégrés
- 7 canaux DMA (Direc Memory Access
- 4 *timers*

**Description succincte de la platine :**
La sélection du mode d'alimentation se fait par cavalier sur un connecteur monté sur la carte.
On peut sélectionner les 3 modes d'alimentation :
- batterie (pile connectée sur le connecteur blanc)
- USB (cordon relié à l'ordinateur)
- alimentation externe (connecteur noir)

Les connecteurs d'entrées/sorties (E/S) numériques peuvent être configurés en :
- liaison série
- liaison $I^2C$
- liaison SPI
- sortie PWM
- sortie tout ou rien logique

Un connecteur reprend 6 des 16 entrées analogiques disponibles.

Dans les paragraphes qui suivent nous allons utiliser :
- une entrée analogique pour le microphone
- un port $I^2C$ pour le capteur numérique
- un port SPI pour communiquer avec la platine Wi-Fi
- un *timer*, qui servira de base à la fréquence d'échantillonnage

Le « RM0008 Reference Manual » [55] sera un support appréciable pour les fonctions avancées de programmation dans les chapitres qui suivent.

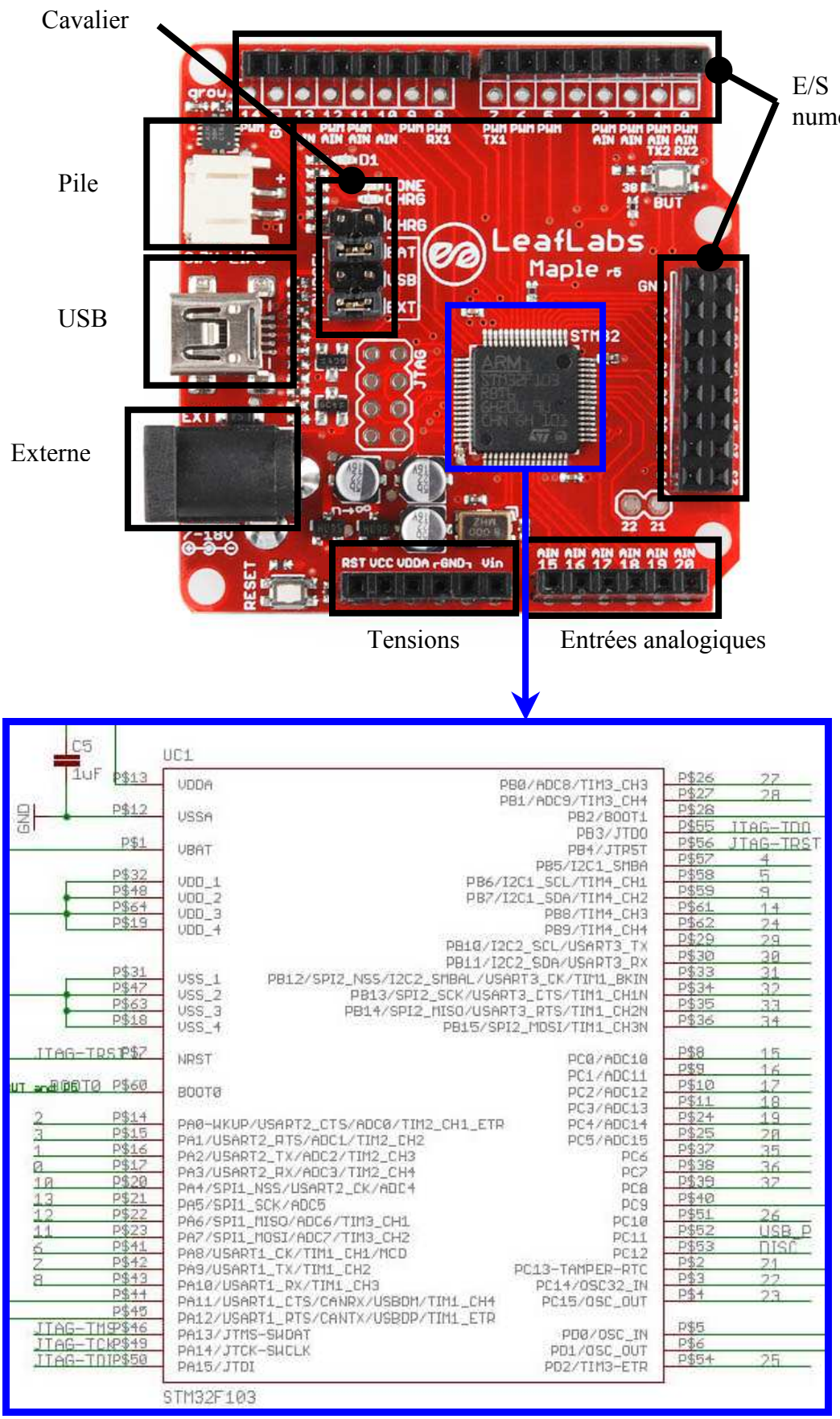



*Figure 5 : Platine Maple (sources [56])*

## 2.4 Fonctions à programmer

Ce système nécessitera deux programmes distincts à coder, l'un destiné à être embarqué sur le microcontrôleur, l'autre à être déployé sur l'ordinateur.
On implémente un automate sur le système embarqué pour interpréter les commandes envoyées par l'ordinateur et effectuer l'action demandée.
Cet automate comprendra des commandes de paramétrage (fréquence d'échantillonnage, paramètres des capteurs numériques…) et de conduite de l'acquisition elle-même (démarrage et arrêt).

Les grandes étapes de codage seront les suivantes :

***Côté « embarqué » :***

- gestion des paramètres du capteur numérique ADXL345 par $I^2C$ ou SPI
- gestion du microphone (CAN, Fréquence d'échantillonnage)
- concaténation des données des différents capteurs
- connexion à la brique *WiFly*
- envoie des échantillons par *WiFly*
- mécanisme de synchronisation et/ou de datation

***Côté « ordinateur » :***

- choix du/des capteurs à observer
- choix des paramètres du capteur numérique ADXL345
- sauvegarde dans un fichier texte (avec en-tête = date, SF, données capteurs, etc...)
- affichage des courbes
- connexion Ad-Hoc Wi-Fi (paramétrage simplifié pour l'utilisateur)

## 2.5 Prise en main des interfaces de développement

Avant toute programmation, il faut se familiariser avec les interfaces de développement (ou IDE : Integrated Development Environment).
Le plus simple étant d'arriver à afficher un message de type « Hello World ».
Toujours dans la philosophie Arduino, de nombreux ouvrages facilitent la compréhension des concepts électroniques et les bases de la programmation. Pour ce projet, le livre « Making Things Talk » est un excellent tutorial pour débuter [57].

### 2.5.1 Maple IDE

**Interface**
Côté microcontrôleur, l'interface de développement *Maple* et le pilote sont téléchargeables gratuitement sur le site du fabricant [58].
La platine électronique se connecte sur un port USB de l'ordinateur.
Une fois branchée et mise sous tension, les pilotes sont automatiquement installés.
Il suffit de noté le numéro du port COM correspondant à la liaison série (simulée par l'USB) pour paramétrer l'interface de développement.
Elle ressemble, du point de vue graphique, à celle de l'Arduino.
L'intérêt de ces interfaces intuitives réside dans leur simplicité d'utilisation.
Le langage, basé sur le C, est très simple et l'application permet à la fois de compiler et de télécharger le code sur la cible par un câble USB en une seule opération. Il n'est pas nécessaire d'avoir des connaissances en architecture logiciel ou en chaîne de compilation. Tout est transparent.

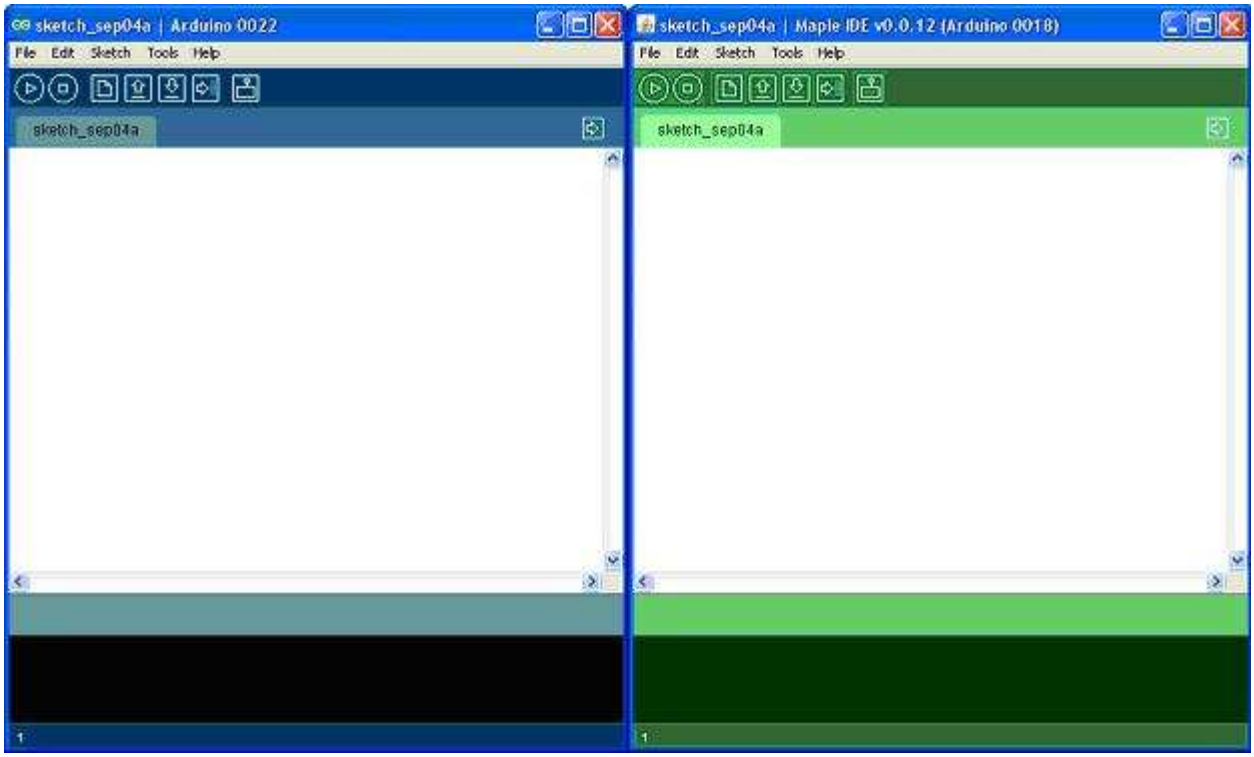



*Figure 6 : Interfaces de développement « Arduino » et « Maple »*

La barre d'outils se compose de quelques icônes nécessaires et suffisantes au fonctionnement de l'application (gestion des sketchs, compilation, téléchargement sur la cible, exécution du programme)

**Sketch**

Les programmes se présentent sous forme de « sketch », un simple fichier en langage C.
A la différence des principaux outils de développement en C/C++, il n'est pas nécessaire de déclarer les fonctions. Le compilateur s'en charge automatiquement.
Le « sketch » principal doit comprendre au minimum les 2 fonctions suivantes :

`void setup()` : on initialise toutes les variables du programme

`void loop()` : le programme tourne en boucle et exécute les instructions

Toutes les fonctions sont encapsulées dans des librairies, le programmeur n'ayant plus qu'à manipuler des objets avec des méthodes simples.
Ainsi, pour communiquer avec l'ordinateur, on utilise l'objet `SerialUSB`, avec les méthodes `read()` pour lire ce qui vient de l'ordinateur et `write()` pour écrire à l'ordinateur (ou `writeln()` pour envoyer un caractère de fin de ligne).

**HelloWorld**

Pour écrire un programme de type « Hello World », rien de plus simple !
Le code suivant suffit :

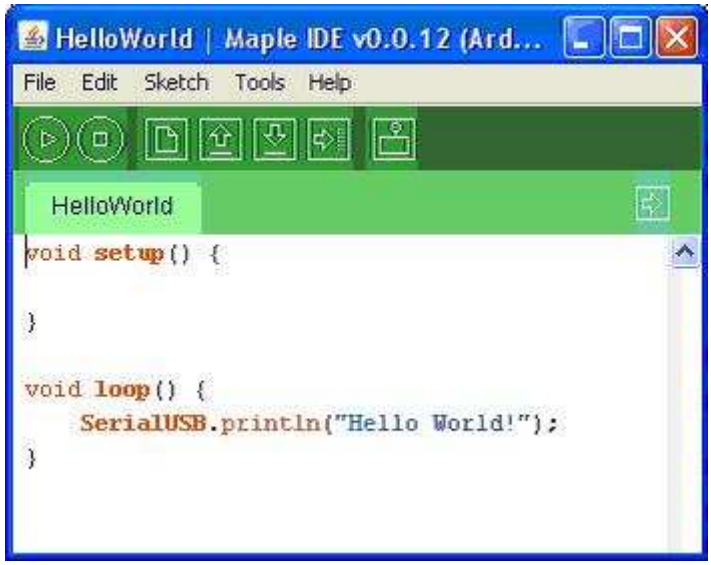



*Figure 7 : Code « Hello World! »*

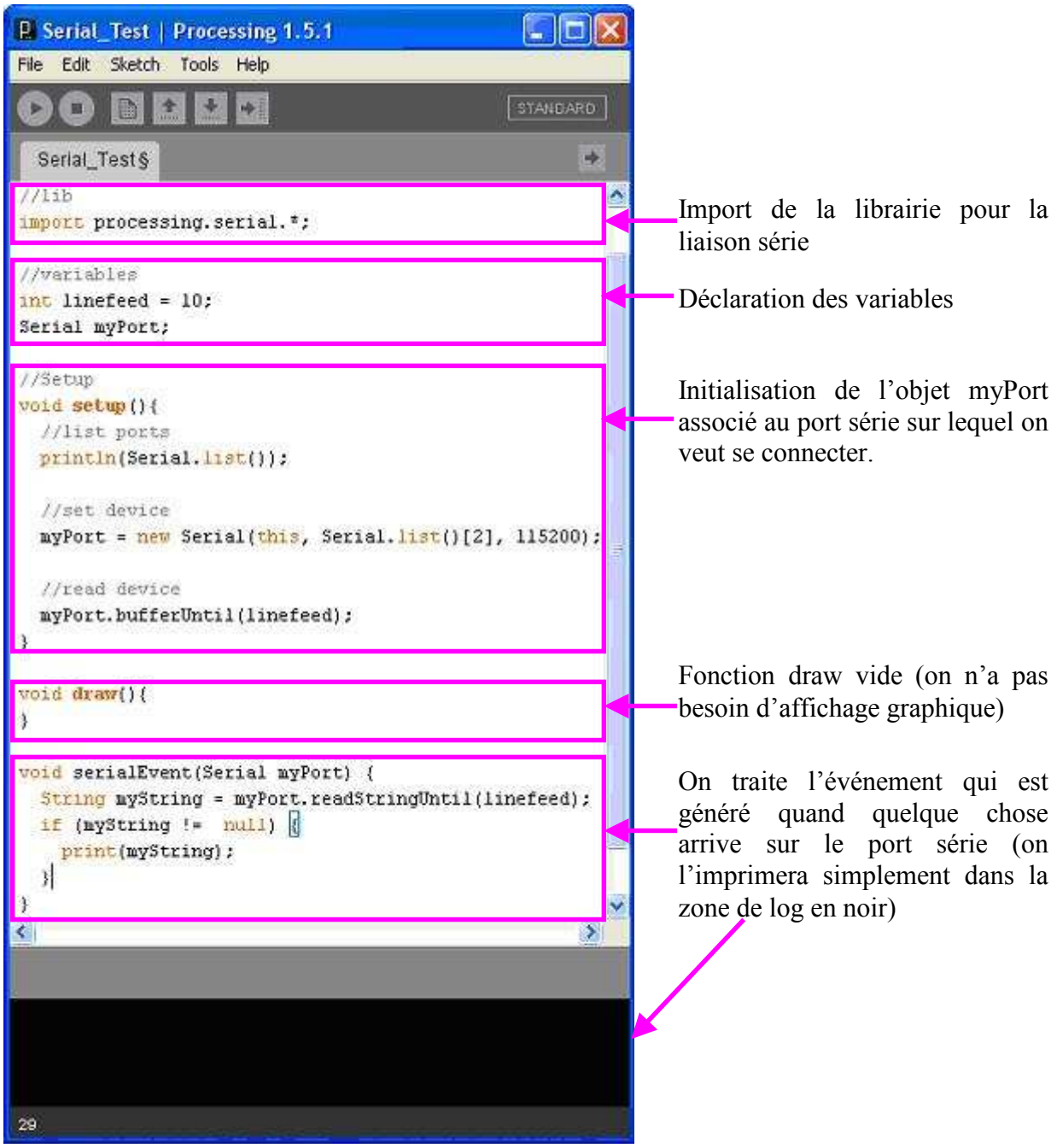
Un simple clic sur  compile le code et le charge sur la platine. Celui-ci s'exécute immédiatement. Il suffit alors de cliquer sur  pour basculer en mode espion sur le port série et voir défiler « Hello World !» en boucle.

Moins de 10 minutes suffisent pour installer et faire fonctionner la platine !

### 2.5.2 Processing

Dans notre application, la communication doit s'effectuer par Wi-Fi, donc il faudra trouver un autre moyen de communication que le port série, au moyen d'une socket.
Pour cela, une interface côté ordinateur est disponible gratuitement : Processing [59].
A la différence de l'application précédente, celle-ci est basée sur un langage Java, langage que je ne maîtrise absolument pas. Mais une fois trouvée la syntaxe des quelques éléments de base de la programmation dont j'avais besoin, cette interface s'est révélée aussi intuitive que la précédente.

Là aussi les programmes sont des « sketchs » très facile à coder, utilisant le même type d'interface que celle décrite précédemment, à la différence près que la fonction `loop()` s'appelle ici `draw()` et redessine constamment la fenêtre sur l'ordinateur.

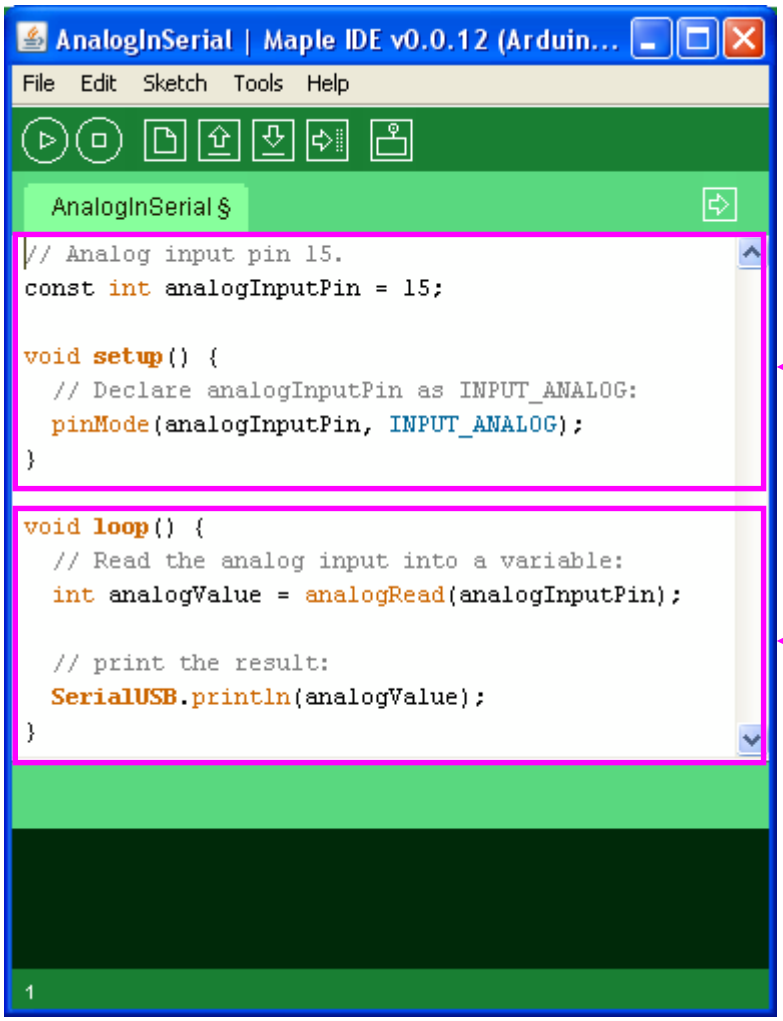



Là aussi, un simple clic sur  compile et lance l'application.

## 2.6 Mise en œuvre d'un capteur analogique : Microphone

### 2.6.1 Lecture d'une valeur analogique

Grâce à cet outil, la mise en œuvre d'un capteur analogique est très simple, tout du moins pour un fonctionnement basique.
En effet, il suffit de déclarer une Entrée/Sortie de la platine comme étant analogique et de lire la valeur de cette entrée, le convertisseur 12 bits donnant une valeur comprise entre 0 et 4096, proportionnel à une tension d'entrée de 0V à 3,3 V (tension d'alimentation).




On déclare l'entrée/sortie 15 comme entrée analogique

On boucle sur la lecture de la valeur de l'entrée analogique et de son écriture sur la liaison USB

Côté ordinateur, il suffit de lire le port série et d'enregistrer les données dans un fichier en utilisant par exemple l'HyperTerminal (utilitaire standard de Windows), configuré pour lire les données du port série où est branché la platine (port COM).

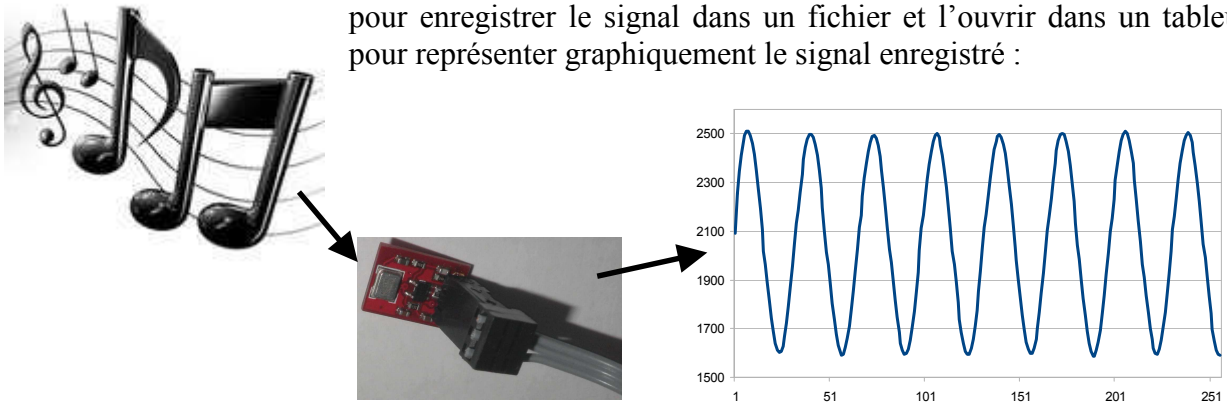
### 2.6.2 Câblage du microphone

La platine utilisée [60] est un capteur MEMS ADMP401 intégrant un microphone subminiature amplifié et ne nécessitant que 3 fils (Vcc, masse et sortie signal) que l'on reliera à la platine *Maple* en connectant la sortie signal sur une des entrées analogiques.

*Caractéristiques*:
- Alimentation: 1,5 à 3,3 Vcc
- Bande Passante : 100 Hz –15 kHz
- Sortie max.: 40 mW
- SNR: -62 dBA

### 2.6.3 Test préliminaire

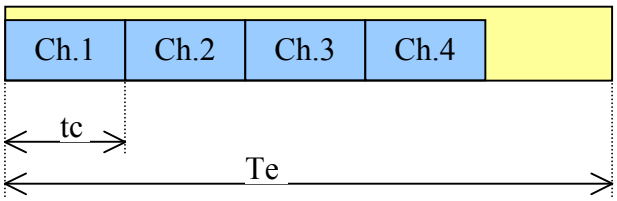
Il suffit alors de siffler dans le microphone pendant quelques secondes pour enregistrer le signal dans un fichier et l'ouvrir dans un tableur pour représenter graphiquement le signal enregistré :




Etant donné que le convertisseur donne une donnée numérique brute comprise entre 0 et 4096, il suffit de retrancher la valeur moyenne 2048 pour obtenir une reproduction plus fidèle du signal sonore, 2048 étant le « zéro ».

### 2.6.4 Fréquence d'échantillonnage

Le test effectué ci-dessus avec une lecture en boucle ne permet pas de connaître et surtout de régler l'intervalle « T » entre les lectures.
Or, nous avons besoin de pouvoir régler la fréquence d'échantillonnage « f » de ce capteur, inverse de cet intervalle (f = 1/T).
Le convertisseur analogique numérique du STM32 permet de régler le temps de conversion, communément appelé « Echantillonneur/Bloqueur » dans la littérature.
La fréquence d'échantillonnage sera réalisée à l'aide d'une base de temps (ou « *timer* »).
Bien que ce microcontrôleur procède à une acquisition séquentielle des voies d'entrées analogiques, en choisissant un temps de conversion (tc) relativement faible devant la période d'échantillonnage (Te) on peut estimer que l'acquisition se fait quasi simultanément :




Toutefois, l'ajout d'un étage bloqueur externe pourrait assurer la simultanéité.
La notion de filtrage du signal en entrée sera abordée plus tard (cf. Chapitre 3.1.5 « Filtrage »).

```
// Microphone ADMP4001 - Analog input pin 15
const int analogInputPin = 15;

//Timer
// Use timer 1 for analogical microphone acquisition
HardwareTimer timerAcq(1);
const int MAX_FIFO = 2048;
int nbMicroseconds = 500;//2khz
int index_fifo = 0;
int data_micro[MAX_FIFO];

//*******************************************************
//                    SETUP
//*******************************************************
void setup()
{
  //********** MICROPHONE ********************//
  // Declare analogInputPin as INPUT_ANALOG:
  pinMode(analogInputPin, INPUT_ANALOG);

  //Use this to change internal sample and hold analogical capacity
  //    ADC_SMPR_1_5,               /**< 1.5 ADC cycles */
  //    ADC_SMPR_7_5,               /**< 7.5 ADC cycles */
  //    ADC_SMPR_13_5,              /**< 13.5 ADC cycles */
  //    ADC_SMPR_28_5,              /**< 28.5 ADC cycles */
  //    ADC_SMPR_41_5,              /**< 41.5 ADC cycles */
  //    ADC_SMPR_55_5,              /**< 55.5 ADC cycles */ ==> DEFAULT
  //    ADC_SMPR_71_5,              /**< 71.5 ADC cycles */
  //    ADC_SMPR_239_5              /**< 239.5 ADC cycles */
  adc_set_sample_rate(ADC1, ADC_SMPR_55_5);

  //********** TIMER INTERRUPT FOR ACQUISITION ***************//
  // Pause the timer while we're configuring it
  timerAcq.pause();
  // Have the timer repeat every nbMicroseconds
  timerAcq.setPeriod(nbMicroseconds);
  // Set up an interrupt on channel 1
  timerAcq.setChannel1Mode(TIMER_OUTPUT_COMPARE);
  timerAcq.attachInterrupt(TIMER_CH1, fill_micro_fifo);
  timerAcq.refresh();
}
//*******************************************************
//function called periodically by the timer
//*******************************************************
void fill_micro_fifo(void){

    //get micro value
    data_micro[index_fifo] = analogRead(analogInputPin);

   //increment fifo index
    index_fifo++;

    //restart fifo
    if (index_fifo == MAX_FIFO){
      index_fifo = 0;
    }
}
```

On déclare un objet « HardwareTimer »

Te = nbMicroseconds
(Te = 500 ms ⇔ Fe = 2kHz)

On règle la durée de l'échantillonneur/bloqueur

On paramètre le timer.

La méthode « SetPeriod » équivaut à affecter la période d'échantillonnage « Te ».

On associe la fonction « fill_micro_fifo » à ce timer.

La fonction « fill_micro_fifo » est appelé périodiquement et rempli un tableau de MAX_FIFO échantillons avec celui lu sur l'entrée analogique.

## 2.7 Mise en œuvre d'un capteur numérique : accéléromètre

### 2.7.1 Câblage de l'accéléromètre ADXL345 en I²C

La platine ADXL345 [61] peut être interfacé en I²C ou en SPI.
Dans un premier temps, on testera en I²C en suivant les exemples fournis.
Nous verrons par la suite, que l'I²C limite le débit et que le mode SPI est préconisé si l'on veut récupérer les données à la fréquence d'échantillonnage maximale du composant (1,6 kHz)

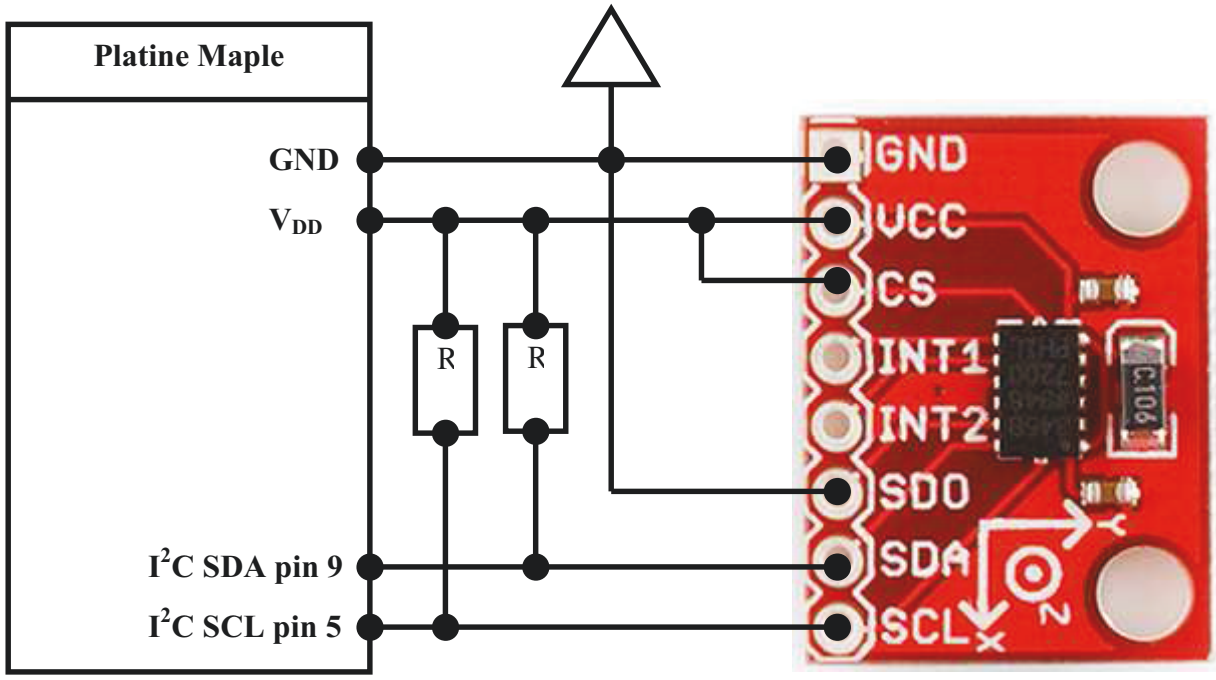



*Figure 8 : Câblage ADXL345 en mode I²C*

SDA : Serial Data = ligne de données
SCL : Serial Clock = horloge

Pour sélectionner le mode I²C, la ligne SDO (Serial Data Out) est mise à la masse et la ligne CS (Chip Select) est connectée à $V_{DD}$.

Les lignes INT1 et INT2 sont des interruptions programmables. On utilisera l'interruption INT1 pour générer une interruption lorsqu'un nouvel échantillon est disponible.

**Avantages d'un capteur numérique :**

- pas de composant électronique de filtrage
- systèmes de registres pour paramétrer son fonctionnement
- lecture directe des valeurs dans les registres
- possibilité de gestion d'interruption (dans notre cas, pour dire que de nouvelles données sont prêtes)

*Caractéristiques*:
- Alimentation: 2 à 3,6 Vcc
- Gamme : ± 2, 4, 8 ou 16g (programmable)
- Bande Passante : 0,1 à 1600 Hz (programmable)

### 2.7.2 Librairie Wire

Pour communiquer avec le composant ADXL345, il suffit d'utiliser la librairie Wire, spécifique à l'I²C comme suit :

```
#include <Wire.h>

//I2C ADXL345
int sda = 9;
int scl = 5;

//Acquisition
uint16 xval = 0;
uint16 yval = 0;
uint16 zval = 0;
#define AXE_X    0
#define AXE_Y    1
#define AXE_Z    2

//************************************************************
//                    SETUP
//************************************************************
void setup()
{
  //initializa variables
  xval = 0;
  yval = 0;
  zval = 0;

  //Setup I2C + ADXL
  Wire.begin(sda, scl);
 }
```

On inclut la librairie Wire

On déclare les lignes SDA et SCL pour l'I²C

Variables pour stocker les valeurs lues sur les axes X, Y et Z

Initialisation de la liaison I²C

Ensuite, il suffit d'écrire dans des registres pour configurer la gamme et la bande passante (voir la datasheet [62]) et de lire les registres où se trouvent les échantillons des 3 axes.
La datasheet présente également les diagrammes de communication du port I²C.

Une liaison I²C ne nécessite que deux lignes de communication, l'une pour les données (SDA), l'autre pour l'horloge (SCL).
L'avantage évident est sa facilité d'utilisation puisqu'il suffit de gérer deux *pins* numériques, mais le revers est que le débit est limité par la vitesse de 400 kHz.
Le protocole de communication nécessite d'indiquer dans quel sens on accède au registre (en lecture ou en écriture), simplement par l'état du premier bit de l'octet envoyé (0 ou 1).

```
//Write data to ADXL register
void WriteADXLRegister(int adr, int data)
{
    Wire.beginTransmission(ADXL_ADDR);
    Wire.send(adr);
    Wire.send(data);
    Wire.endTransmission();
    delay(5);
}
```

Ecriture dans un registre par I²C

```
//Read data from ADXL register
int ReadADXLRegister(int adr)
{
    int data = 0;
    Wire.beginTransmission(ADXL_ADDR);
    Wire.send(adr);
    Wire.endTransmission();
    Wire.requestFrom(ADXL_ADDR, 1);
    if (Wire.available()) {
      data = Wire.receive();
    }
    return data;
}
```

Lecture dans un registre par I²C

```
//Read data from 6 axis registers as multibyte (2 byte * 3-axis)
void ReadAllAxis()
{
    const int MAX_DATA = 6;
    int i = 0;
    int data[MAX_DATA] = {0, 0, 0, 0, 0, 0};
    Wire.beginTransmission(ADXL_ADDR);
    Wire.send(DATAX0);
    Wire.endTransmission();
    Wire.requestFrom(ADXL_ADDR, MAX_DATA);
    if (Wire.available() == MAX_DATA) {
      for (i = 0; i < MAX_DATA; i++){
        data[i] = Wire.receive();
      }
    }
    xval = data[0] + (data[1] << 8);
    yval = data[2] + (data[3] << 8);
    zval = data[4] + (data[5] << 8);
}
```

Lecture de 6 registres contigus par I²C pour récupérer les 3 échantillons des 3 axes stockés sur 2 octets chacun (à partir de l'adresse du registre DATAX0, premier octet de l'axe X)

### 2.7.3 Fréquence d'échantillonnage

La bande passante est programmable de 0,1 Hz à 1600 Hz, en doublant à chaque fois la fréquence (0,1 Hz, 0,2 Hz… 800 Hz, 1600 Hz).
Pour notre application, nous testerons les plus hautes (25, Hz, 50 Hz, 100 Hz, 200 Hz, 400 Hz, 800 Hz et 1600 Hz).

Pour récupérer les échantillons, on peut utiliser un *timer*, comme nous l'avons vu précédemment avec le microphone (cf. 2.6.4 « Fréquence d'échantillonnage ») ou par interruption, quand un nouvel échantillon est prêt et a remplacé l'ancienne valeur.
Cette méthode a l'avantage de n'être appelée que quand de nouvelles valeurs sont disponibles, évitant ainsi de relire les mêmes valeurs plusieurs fois de suite.
Pour cela, il faut programmer les registres de sorte que l'interruption soit générée par le DATA_ READY et câbler l'interruption (ligne INT1) sur une entrée numérique de la platine *Maple*.

Le code comprendra les éléments suivants :

```
const int ADXL_INTERRUPT_PIN = D35;

void setup()
{
  //interrupt on pin35 (= ADXL INT1)
   pinMode(ADXL_INTERRUPT_PIN, INPUT);
   //Acquire by interruption
   attachInterrupt(ADXL_INTERRUPT_PIN, DoReadAxisOnInterrupt, RISING);
}

//Update data from 6 axis registers as multibyte (2 byte * 3-axis)
void DoReadAxisOnInterrupt()
{
    //Read axis
    Adxl345Serial.ReadAllAxis(nAcceleroDataArray);
}
```

Numéro de l'entrée physique d'interruption

Déclaration du type entrée numérique et association de la fonction d'interruption

Fonction appelée par l'interruption qui va lire les registres contenant les valeurs des échantillons mis à jour

## 2.8 Mise en œuvre de la liaison Wi-Fi : WiFly

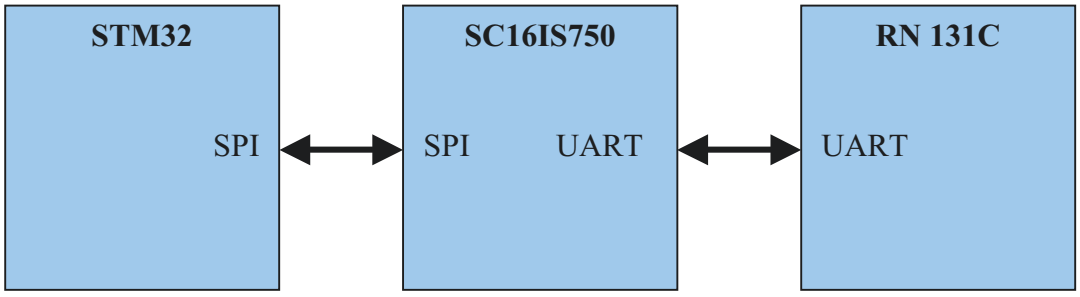
C'est la partie de mon stage qui a été le plus difficile à mettre en œuvre. Elle a nécessité une modification de la librairie d'origine et je l'utilise au maximum de ses capacités de transfert.
La communication entre le microcontrôleur STM32 et le composant Wi-Fi RN131C n'est pas directe mais passe par un adaptateur SPI/UART (SC16IS750) qui autorise des vitesses de transfert supérieures à 115200 bauds.

*Caractéristiques Radio du RN131C*

| | |
|---|---|
| Frequences | 2402 ~ 2480MHz |
| Modulation 802.11b | DSSS |
| Modulation 802.11g | OFDM |
| Intervalle entre les canaux | 5 MHz |
| Canaux | 1 - 14 |
| Débit de transmission 802.11b | 1 – 11Mbps |
| Débit de transmission 802.11g | 6 – 54Mbps |
| Sensibilité en réception | -85dBm typ. |
| Niveau de sortie (Classe 1) | +18dBm (63 mW) |

### 2.8.1 Architecture générale

Pour mettre en œuvre la maquette Wi-Fi, je me suis inspirée du tutoriel proposé par SparkFun [63]. La librairie Arduino fournie avec le *WiFly* est destinée à être utilisée avec une carte Arduino basée sur un microcontrôleur Atmega328. Il est donc nécessaire de l'adapter pour la rendre compatible avec le *Maple* basé sur un microcontrôleur STM32.




La librairie contient les couches successives pour mettre en œuvre le Wi-Fi :

- Fonctions bas niveau pour contrôler le port SPI
- Fonctions de configuration de la passerelle SPI/UART [64]
- Fonctions de configuration du composant Wi-Fi RN131C [65]
- Fonctions pour créer un client TCP
- Fonctions pour administrer des pages Web ou des fichiers FTP

Ce fonctionnement devra être modifié car on n'utilise pas le serveur (Web), mais juste la couche physique de liaison radio.
J'ai donc également créé couche applicative propre pour concaténer et envoyer les octets correspondant aux échantillons et aux éléments de contrôle du temps réel.

### 2.8.2 Adaptation des fonctions bas niveau

Les fichiers cités dans ce chapitre sont fournis dans l'Annexe C « Sources programmes Maple ».
Les fonctions bas niveau communiquent avec le composant Wi-Fi via un convertisseur SPI/UART. Voici les modifications apportées aux fichiers « _spi.c » et « _spi.h »

⇒ déclaration des *pins* du SPI1

| Définition SPI | Pins Arduino | Pins Maple |
|---|---|---|
| Master Out/ Slave In | MOSI | BOARD_SPI1_MOSI_PIN |
| Master In / Slave Out | MISO | BOARD_SPI1_MISO_PIN |
| SPI Clock | SCK | BOARD_SPI1_SCK_PIN |
| Note Slave Select | NSS | BOARD_SPI1_NSS_PIN |

⇒ lecture/écriture des adresses de registres du SPI1

| Définition SPI | Registre/bit Arduino | Registre/bit Maple |
|---|---|---|
| Control Register | SPCR | CR1 et CR2 |
| Status Register | SPSR | SR |
| Data Register | SPDR | DR |
| Bit TX done | | SPI_SR_TXE |
| Bit RX done | | SPI_SR_RXNE |
| Bit End Of Transmission | SPIF | SPI_SR_BUSY |

Pour la déclaration des *pins*, il a été nécessaire de définir un nouveau type d'Output **`OUTPUT_AF_PP`** dans « io.cpp » afin d'utiliser le mode Alternate Function Push-Pull dans la fonction '`pinMode`' du fichier « wirish_digital.cpp » sur la platine Maple :

```
void pinMode(uint8 pin, WiringPinMode mode)
{  ...
   switch(mode)
   {  ...
   case OUTPUT_AF_PP:
            outputMode = GPIO_AF_OUTPUT_PP;
            pwm = false;
            break;
   ...  }
   gpio_set_mode(PIN_MAP[pin].gpio_device, PIN_MAP[pin].gpio_bit, outputMode);
   if (PIN_MAP[pin].timer_device != NULL) {
   // Enable/disable timer channels if we're switching into or out of PWM.
   timer_set_mode(PIN_MAP[pin].timer_device,
                          PIN_MAP[pin].timer_channel,
                          pwm ? TIMER_PWM : TIMER_DISABLED);
}
```

Le manuel de référence du STM32F102 décrit le fonctionnement du SPI ([55] « *25.3 SPI functional description* » page 675 et suivantes).
Le port de communication SPI est connecté via 4 *pins* :

- *MISO = Master In / Slave O*ut : transmet des données en mode « Esclave » et reçoit des données en mode « Maître ».
- *MOSI = Master Out / Slave In Out* : transmet des données en mode « Maître » et reçoit des données en mode « Esclave ».
- *SCK = Serial Clock* : horloge de sortie en mode « Maître » et horloge d'entrée en mode « Esclave »
- *NSS = Slave Select* : correspond à un « *Chip Select* » pour choisir un « Esclave » et communiquer avec lui.

Il faut configurer les registres de telle sorte que l'on puisse commander par soft la *pin* NSS.
On utilisera le mode *Full Duplex*, décrit page 683.

⇒ *Principales fonctions bas niveau modifiées :*

| Fonction | Arduino | Maple |
|---|---|---|
| _initPins | pinMode(MOSI, OUTPUT);<br>pinMode(MISO, INPUT);<br>pinMode(SCK, OUTPUT);<br>pinMode(_selectPin, OUTPUT); | pinMode(BOARD_SPI1_MOSI_PIN, OUTPUT_AF_PP);<br>pinMode(BOARD_SPI1_MISO_PIN, INPUT);<br>pinMode(BOARD_SPI1_SCK_PIN, OUTPUT_AF_PP);<br>pinMode(_selectPin, OUTPUT); |
| _initSpi | SPCR = (1<<SPE)\|(1<<MSTR)\|(0<<SPR1)\|(0<<SPR0);<br><br>SPSR = SPSR \| (1 << SPI2X);<br><br>char clr = 0;<br>clr=SPSR;<br>clr=SPDR;<br><br>//SPE = SPI enable<br>//MSTR = Master<br>//MODE0 = Clock 0 & Phase 0 | rcc_clk_enable(SPI1->clk_id);<br>rcc_reset_dev(SPI1->clk_id);<br><br>//Disable<br>spi_irq_disable(SPI1, SPI_INTERRUPTS_ALL);<br>spi_peripheral_disable(SPI1);<br><br>//Mode hardware<br>uint32 flags = SPI_FRAME_MSB \| SPI_DFF_8_BIT ;<br>SPI1->regs->CR1 = SPI_CR1_SPE \| SPI_BAUD_PCLK_DIV_32 \| flags \| SPI_CR1_MSTR \| SPI_MODE_0;<br>SPI1->regs->CR2 = SPI_CR2_SSOE;<br><br>//enable<br>spi_peripheral_enable(SPI1);<br><br>//clear registers by reading them<br>int clr = 0;<br>clr = SPI1->regs->SR;<br>clr = SPI1->regs->DR;<br>clr = SPI1->regs->CR1;<br>clr = SPI1->regs->CR2; |
| byte transfer (volatile byte data) | // Start the transmission<br>SPDR = data;<br>// Wait for the end of the transmission<br>while (!(SPSR & (1<<SPIF))) {};<br>// Return the received byte<br>return SPDR; | // Start the transmission<br>SPI1->regs->DR = data;<br>// Wait for the end of the transmission<br>while((SPI1->regs->SR & SPI_SR_TXE) != SPI_SR_TXE);<br>// Wait for the end of the reception<br>while((SPI1->regs->SR & SPI_SR_RXNE) != SPI_SR_RXNE);<br>// Return the received byte<br>return (SPI1->regs->DR); |

### 2.8.3 Paramétrage du réseau Wi-Fi

On va créer une liaison en mode « Ad-hoc » entre l'ordinateur et la maquette. Comme on n'a pas besoin d'avoir accès à Internet, ce réseau privé direct entre le capteur et l'ordinateur suffira.
Dans ce mode, les deux objets doivent communiquer avec le même SSID sur le même canal, chacun avec une adresse IP fixe connue de l'autre, un peu comme pour une liaison téléphonique.
On n'a pas besoin d'un niveau de sécurité/confidentialité élevé (i.e. clé WES ou WAP), ce qui permet de gagner de la bande passante. De plus, lorsqu'une liaison est en mode ad-hoc, le SSID est transmis en clair par tous les acteurs et le réseau ne peut être fermé [66].

**Configuration Wi-Fi côté ordinateur**

Côté ordinateur, la liaison Wi-Fi est gérée par une clé USB (NetGear N150 Wireless USB Adapter) car il n'y a pas de carte Wi-Fi intégrée.
La documentation du composant Wi-Fi RN131C donne l'adresse IP « 169.254.1.1 » et le SSID « Wifly-GSX-32 » par défaut pour un réseau Ad-Hoc.
Il faudra choisir celle de l'ordinateur dans le même domaine, en configurant le réseau sans fil :


En mode administrateur, on paramètre le protocole Internet (TCP/IP) avec:

**`Adresse = 169.254.1.11`**
`Métrique : 2`

(Le métrique doit être modifié, sinon Windows re-route arbitrairement les données vers la connexion réseau locale filaire).

Dans la liste des réseaux sans fil, on voit bien apparaître l'item Wifly-GSX-32 :


**Configuration Wi-Fi côté maquette**

Côté *Maple*, on charge un sketch ("*SpiUartTerminal_Maple.pde*", disponible dans l'Annexe C « Sources programmes Maple ») permettant uniquement de simuler une communication entre la liaison USB du *Maple* et la liaison Wi-Fi, avec :

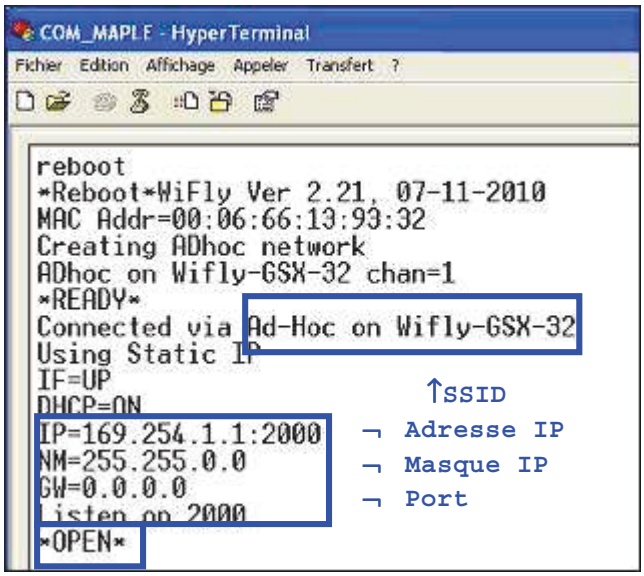
| | |
|---|---|
| Mode : | Ad-Hoc |
| SSID : | Wifly-GSX-32 |
| Adresse IP : | 169.254.1.1 |
| Masque IP : | 255.255.0.0 |
| Vitesse UART : | 9600 bauds |
| Port : | 2000 |

## Connexion

Le sketch s'exécute sur la maquette *Maple* et établit une connexion avec l'ordinateur.

D'un côté, on observe ce qui transite sur la liaison USB du *Maple* en ouvrant une fenêtre *HyperTerminal* sur le port COM correspondant (voir 0 « **Interface** »).

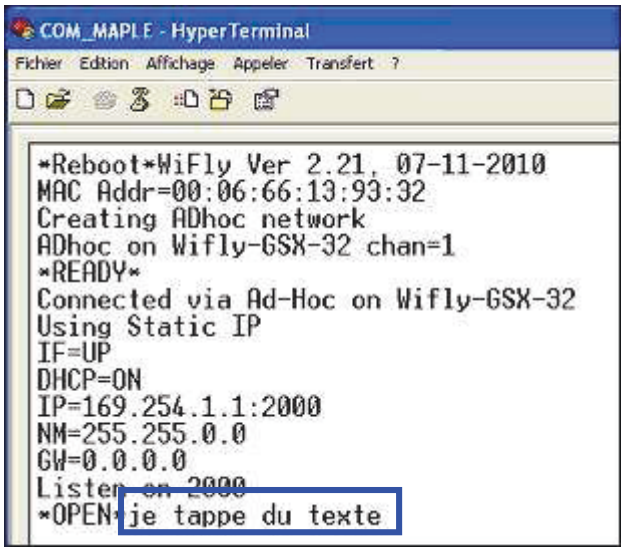
La liaison est établie par *OPEN*




De l'autre côté, on observe ce qui transite sur la liaison Wi-Fi en ouvrant une session *Telnet* en ligne de commande (telnet 169.254.1.1 2000).

La liaison est établie par *HELLO*


Si on écrit du texte dans l'une…




… il apparaît dans l'autre

**Echange de données par liaison serveur/client**

Pour tester l'échange de données entre le programme exécuté sur la maquette et celui exécuté sur l'ordinateur, on rajoute un « automate » permettant d'envoyer une commande à partir de l'ordinateur pour que *Maple* effectue des actions.

| Commande | Action | Affichage |
|---|---|---|
| m | Retourne la valeur du CAN de la voie connectée au microphone | *microphone 1925* |
| x | Retourne une chaîne de caractères prédéfinie | *ceci est du texte* |
| t | Renvoie 1024 fois le caractère '0' et 'x' pour finir, afin de mesurer le débit côté ordinateur | *0000....(1024 fois)....0000000x* |

*Tableau 3 : Commandes de l'automate pour tester le Wi-Fi*

Dans un premier temps, on ne se préoccupe pas de la vitesse, fixée par défaut à 9600 bauds pour la liaison SPI/UART entre le microcontrôleur et le composant Wi-Fi.

Le code comprendra les éléments suivants :

```
#include "WiFly.h"
char ssid[] = "Wifly-GSX-32";
int  ip_id = 1;// = 169.254.1.ip_id, with ip_id = sensor id
int  channel = 3; //Pour éviter le channel 1 très utilisé ...
const int analogInputPin = 15;

Server server(80);

void setup() {

  // Declare analogInputPin as INPUT_ANALOG:
  pinMode(analogInputPin, INPUT_ANALOG);

  //Set Ad Hoc Configuration
  WiFly.beginAdHoc();
  WiFly.setConfigurationAdHoc(ssid, ip_id, channel) ;

  //Reboot: automatically start in AdHoc with previous parameters
  if (!WiFly.RebootAdHoc()) {
       while (1) {
      // Hang on failure.
    }
  }
 // WiFly.configure(WIFLY_BAUD, 2400);
  server.begin();
}
```

On inclut la librairie WiFly

On déclare les paramètres

On crée un objet Serveur sur le port 80

On initialise une liaison en mode Ad-Hoc et on reboot

On démarre le Serveur

```
void loop() {

  Client client = server.available();
  if (client) {
    while (client.connected()) {
      if (client.available()) {
        char c = client.read();
        switch(c){
          case 'm':
            client.print("microphone ");
            client.println(analogRead(analogInputPin));
            break;
          case 'x':
            client.println("ceci est du texte");
            break;
          case 't':
            for (int i = 0; i < 1024; i++){
              client.print(0);
            }
            client.print("x");
            client.println();
            break;
        }//switch
      }//client.available
    }//client
```

On crée un Client qui se connecte à un serveur dés qu'il en trouve un (i.e. l'ordinateur)

On scrute s'il y a des données à traiter (les commandes envoyées par l'ordinateur)…

… que l'on décrypte pour exécuter l'action correspondante

## 2.9 Programme côté ordinateur

Côté utilisateur, il faut développer un programme qui intègre les fonctions :
- de communication avec la maquette
- de traitement des données

Pour la communication Wi-Fi avec le RN131C, le protocole conseillé est le TCP (Transmission Control Protocol) qui assure l'intégrité des données transmises.
Mais nous verrons que le protocole UDP (User Data Protocol) est plus adapté à notre application (cf. 2.10.3 « Test du débit du Wi-Fi ») en augmentant le débit théorique possible.

Le paramétrage de la maquette sera assuré par un automate qui traitera les commandes transmises par l'utilisateur.
En mode acquisition, la lecture des échantillons se fera en continu. Il faudra s'assurer que l'on récupère bien tous les échantillons avec un algorithme de contrôle périodique du flux.

Les données décryptées pour chaque voie analysée seront affichées sous forme de courbes temporelles, pour contrôler visuellement ce qui se passe.
En parallèle, ces données seront sauvegardées dans un fichier texte pour être analysées ultérieurement par les chercheurs.

L'interface de développement que j'ai choisi est « Processing », recommandé par la communauté Arduino. Il est facile de prise en main et, comme pour l'électronique, permet le prototypage rapide de solution.
Les commandes sont envoyées par le clavier car il n'y a pas Interface Homme Machine (IHM) pour dessiner les boutons de commandes et les choix de réglages.
Mais il montre vite ses limites en capacité de traitement temps réel : l'affichage se fige si la fréquence d'échantillonnage est supérieure à 5 kHz.
Théoriquement, on devrait atteindre une fréquence d'échantillonnage de 20 kHz.

Le langage « Processing » est orienté Java. L'application qui en résulte est un JavaScript qu'il suffit de recopier sur l'ordinateur où on veut l'utiliser.

### 2.9.1 Protocole

En « Processing », le **client TCP** est utilisable en natif, en incluant la librairie « processing.net ».
Voici une brève description du code de ce Client :

| | |
|---|---|
| `import processing.net.*;` | Chargement de la librairie |
| `Client myClient;` | Déclaration du Client TCP |
| `myClient = new Client(this, "169.254.1.1", 8001);` | Initialisation du Client TCP :<br>Adresse IP + n° port |
| `myClient.write(dataOut);` | Données à émettre vers le Wi-Fi |
| `count = myClient.available();`<br>`for (int i = 0; i < count; i++)`<br>`  dataIn = myClient.read();` | Données à recevoir du Wi-Fi |
| `void ReceiveData()` | Evènement déclenché par l'arrivée de données |

Pour tester le **protocole UDP**, il faut inclure une librairie téléchargeable sur le site Processing [67].

| | |
|---|---|
| `import hypermedia.net.*;` | Chargement de la librairie |
| `UDP myClient;` | Déclaration du Client UDP |
| `myClient = new UDP( this, 8001, "169.254.1.1");`<br>`myClient.listen(true);` | Initialisation du Client UDP :<br>n° port + Adresse IP<br>A l'écoute… |
| `myClient.send(dataOut, szIPSensor, nPortSensor );` | Données à émettre vers le Wi-Fi |
| `count = data.length;`<br>`for (int i = 0; i < count; i++)`<br>`dataIn = data[i];` | Données à recevoir du Wi-Fi |
| `void receive( byte[] data, String ip, int port )` | Evènement déclenché par l'arrivée de données du capteur avec l'adresse IP indiquée |

Quelque soit le protocole, la mise en œuvre est isofonctionnelle :
- déclaration
- lecture par gestion d'événement « il y a de nouvelles données disponibles »
- écriture à la demande par « SendKey »

### 2.9.2 Paramètres et commandes de l'automate

Les paramètres sont partagés entre le programme côté ordinateur et le système d'acquisition déporté. Les réglages sont transmis par le Wi-Fi (ordinateur ⇒ *WiFly*).
Un automate côté WiFly traite la demande (valeur valide) et renvoie une réponse à l'ordinateur par Wi-Fi (WiFly ⇒ ordinateur).
Dans le programme développé en Processing, les commandes seront envoyées par le clavier avec une fonction de type « SendKey ».

| Paramètre | Description | Commande automate | Par défaut |
|---|---|---|---|
| **CAN (voies analogiques)** | | | |
| CAN - NbChannels | Nombre de voies analogiques (1 à 6) | m0 : aucune<br>m1 à m6 : de 1 à 6 | 0 (inactive) |
| CAN – SF | Fréquence d'échantillonnage | fxxx : xxx en Hz | 2000 (Hz) |
| CAN – SH | Durée du « *Sample and Hold* » en nombre de cycles<br><br>C'est la durée de la fonction Échantillonneur/Bloqueur du microcontrôleur | c0 : 1,5<br>c1 : 7,5<br>c2 : 13,5<br>c3 : 28,5<br>c4 : 41,5<br>c5 : 55,5<br>c6 : 71,5<br>c7 : 239,5 | 1 (7,5 cycles) |
| **Accéléromètre numérique triaxe** | | | |
| Accel – ON/OFF | Activation de l'acquisition | x0 : OFF<br>x1 : ON | 0 (inactive) |

| | | | |
|---|---|---|---|
| Accel - Range | Gamme de l'accéléro | r2 : +/- 2g sur 10 bits<br>r4 : +/- 4g sur 11 bits<br>r8 : +/- 8g sur 12 bits<br><br>r20 : +/- 2g sur 10 bits<br>r40 : +/- 4g sur 10 bits<br>r80 : +/- 8g sur 10 bits<br>r160 : +/- 16g sur 10 bits | 8 (8g) |
| Accel - Bandwidth | Bande passante de l'accéléro | bxx : xx en Hz<br>(3, 6, 12, 25, 50, 100, 200, 400, 800, 1600) | 800 (Hz) |
| Accel – Identifiant | Récupère l'identifiant de l'accéléro | i | "ADXL345" |
| **Analyse** | | | |
| Analyse – Durée | Acquisition Continue ou sur une durée définie | t0 : continue<br>txxx : xxx secondes | 0 |
| Analyse – Nb Max d'échantillons | Nombre d'échantillons au bout duquel on envoie le *TimeStamp* pour vérifier la fréquence d'échantillonnage | nxxx : xxx samples<br>(default : 2048) | 2048 |

*Tableau 4 : Commandes de l'automate avec pramètres*

| Fonction | Description | Commande automate | Renvoie |
|---|---|---|---|
| Acquisition ON | Démarrage de l'acquisition | a | idem 'g' puis *START* |
| Acquisition OFF | Stoppe l'acquisition | s | *STOP* |
| Help | Aide en ligne | h | liste des commandes disponibles |
| Hello | Vérifie la connexion en renvoyant vérifiant une chaîne de caractère simple | H | *HELLO* |
| Get current parameters | Récupère les paramètres actuels du CAN, de l'accéléro et de l'analyse | g | valeurs des paramètres (cf. Tableau 4) |

*Tableau 5 : Commandes de l'automate sans paramètres*

### 2.9.3 Structure des échantillons

Un échantillon est une représentation numérique de la valeur analogique mesurée.
Comme nous utilisons un convertisseur 12 bits, il sera compris entre 0 et 4095.
La fonction brute qui envoie une donnée numérique par Wi-Fi est en ASCII.
Ainsi la valeur [1419] sera envoyée avec la suite des 4 caractères « 1 » « 4 » « 1 » « 9 », soit 4 octets (32 bits).
Or la valeur initiale tient sur 12 bits, on devrait pouvoir trouver un moyen de densifier l'information. La solution proposée est la suivante :
Pour garder la compatibilité avec un nombre entier d'octets, on va coder sur 16 bits soit 2 octets.
Il reste alors 4 bits pouvant être utilisés comme bits de contrôle pour sécuriser la transmission.
Un découpage possible consiste en 2 octets contenant chacun 6 bits de l'échantillon et 2 bits de contrôle.
Dans notre exemple, [1419] donne en binaire : 0101.1000.1011

| | | |
|---|---|---|
| donnée 12 bits | 0101.1000.1011 | |
| découpée en… | 0101.10 ✂ 00.1011 | |
| … 2 fois 6 bits | 010110 | 001011 |
| bits de contrôle à ajouter | $b_1b_2$ | $b_3b_4$ |
| chaque octet devient | $b_1b_2$010110 | $b_3b_4$001011 |
| que nous appellerons | MSB | LSB |

Ce découpage représente un double avantage :
- un facteur 2 sur la vitesse de transmission des données
- une transmission fiable en contrôlant chaque octet transmis

Reste à choisir une méthode qui soit facile à implémenter sans réclamer trop de temps de calcul qui pourrait obérer le facteur de vitesse gagné.

Un principe très simple consiste à marquer les octets de poids fort (MSB) et faible (LSB) :

| Position de l'octet | Valeur du bit |
|---|---|
| MSB | 1 |
| LSB | 0 |

Cette méthode ne nécessite aucun calcul car on peut la coder en dur lors du découpage en deux octets. Dans le code de réception, il suffira de faire un masque sur ce premier bit pour s'assurer qu'on ne manque pas un octet. Mais la limite est qu'on ne sait pas combien d'octets n'ont pas été transmis. Nous verrons plus loin comment compter périodiquement les échantillons envoyés.

Un deuxième principe expérimenté pour utiliser le deuxième bit de contrôle disponible pour chaque octet est de calculer sa parité :

| Somme des 6 bits de données | Valeur du bit de parité |
|---|---|
| paire | 0 |
| impaire | 1 |

Ce bit permet de s'assurer que le contenu de l'octet a été transmis sans erreur.
Après l'avoir testé, nous avons constaté que la transmission était suffisamment fiable de ce côté mais que ce calcul prenait environ 3 kHz de notre précieuse bande passante.

Le format finalement choisi pour transmettre un échantillon est donc :

| MSB | LSB |
|---|---|
| **1**.X.$b_{12}$. $b_{11}$. $b_{10}$. $b_9$. $b_8$. $b_7$ | **0**.X.$b_6$. $b_5$. $b_4$. $b_3$. $b_2$. $b_1$ |

X : sans importance
$b_{12}$ … $b_1$ : les 12 bits de données

On a illustré la philosophie et l'intérêt de contrôler les flux. Les méthodes de théorie de l'information étudiées en cours utilisent des algorithmes de contrôle plus sophistiqués.

Dans le cas idéal, on pourrait admettre que la transmission est totalement fiable. On pourrait alors imaginer accoler les échantillons de toutes les voies les unes aux autres. Ainsi 4 voies échantillonnées à 12 bits (soit 48 bits de données) tiendraient sur 6 octets seulement.
Avec l'utilisation du code ASCII, nous aurions eu besoin de 16 octets.
On gagnerait un facteur de 2,7 sur la vitesse de transmission des données (i.e 16/6).
C'est le facteur de vitesse maximum que l'on pourrait obtenir sans compression des données et sans contrôle du flux.

Le chapitre 4.4 « Acquisition : description du format de transmission des données » décrit en détail les différents niveaux de contrôle de transmission des données.

## 2.10 Résultats & Performances

Une fois toutes les briques assemblées, nous allons pouvoir mesurer les performances globales de la maquette, principalement en termes de vitesse (qui conditionnera la fréquence d'échantillonnage) et de robustesse de la transmission.

### 2.10.1 Chronométrage

On peut instrumenter le code embarqué pour mesurer les temps d'exécution des différentes fonctions :

- soit avec la fonction « `micros()` » qui retourne le nombre de microsecondes écoulées depuis le début de l'exécution du programme (en comparant les temps d'entrée et de sortie de la fonction),
- soit en utilisant une sortie numérique que l'on peut observer à l'oscilloscope (que l'on met au niveau haut lorsque l'on rentre dans la fonction et au niveau bas quand on sort).

### 2.10.2 Mesures en mode filaire par l'USB

Dans un premier temps, on se limitera à une transmission « filaire » par le câble USB.
C'est le mode de fonctionnement par défaut de la platine.
Le débit est d'environ 115 kbps maximum.
On teste l'acquisition de deux types de capteurs:

- un accéléromètre tri-axe à sortie numérique (ADXL345)
- un microphone à sortie analogique (ADMP401)

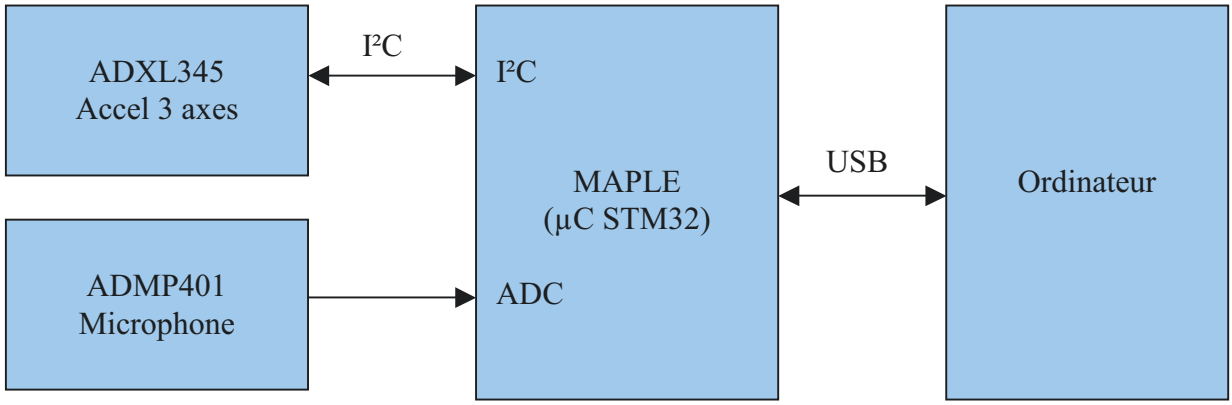



*Figure 9 : Architecture pour la mesure filaire*

**Premières mesures brutes**

On teste simplement en boucle une séquence de la forme :

- acquisition d'une valeur analogique du microphone
- acquisition de 3 valeurs numériques de l'accéléromètre tri-axes
- transmission à l'ordinateur par la liaison USB

| Fonction | Durée |
|---|---|
| Acquisition microphone | 1 ms |
| Acquisition accéléromètre | 4 ms |
| Transmission | 17 ms |
| Total | 22 ms |

On constate que ce qui prend du temps, c'est surtout la transmission des données par l'USB !
On va donc tester séparément selon le principe :

- acquisition de X valeurs
- transmission de X valeurs

**Acquisition des données de l'accéléromètre seul**

L'accéléromètre est à sortie numérique.
On peut communiquer avec deux types de protocoles :

- I²C en mode 100 kHz ou 400 kHz
- SPI jusqu'à 5 MHz

Pour des raisons de confort, en réutilisant du code disponible dans des exemples, j'ai d'abord testé en câblant en I²C.
Pour avoir accès aux données correctement mises à jour, il faut lire le bit « Data_Ready ».
On fait varier le réglage de la bande passante et on mesure le temps nécessaire pour acquérir 512 échantillons, en lisant les 6 octets de données à la fois (2 registres par axe).
En théorie, le débit des données de sortie (ODR, Output Data Rate) est égal à 2 fois celui de la bande passante (BP).

| **Bande Passante (Hz)** | **Temps d'acquisition de 512 échantillons (µs)** | **Débit de données correspondant (Hz)** | **A comparer à 2 × BP = ODR** |
|---|---|---|---|
| 1600 | 345 378 | 1484 | < 2 × 1600 = 3200 |
| 800 | 357 607 | 1433 | < 2 × 800 = 1600 |
| 400 | 646 518 | 792 | = 2 × 400 = 800 |
| 200 | 1 291 571 | 397 | = 2 × 200 = 400 |
| 100 | 2 587 027 | 198 | = 2 × 100 = 200 |
| 50 | 5 169 002 | 99 | = 2 × 50 = 100 |
| 25 | 10 337 960 | 50 | = 2 × 25 = 50 |

*Tableau 6 : Mesure débit ADXL345 en mode I²C*

Ce qui est cohérent avec la documentation ([62], page 15) :

| Pour des débits de données (ODR) de : | Il est recommandé d'utiliser : |
|---|---|
| 1600 Hz, 3200 Hz | SPI avec fréquence > 2 MHz |
| 800 Hz, 400 Hz | I²C @ 400 kHz |
| 200 Hz et en dessous | I²C @ 100 kHz |

Pour l'I²C, la vitesse de communication optimale doit être de ODR x 500.
Dans notre programme la liaison I²C est en configuration 400 kHz.

**➔ Avec l'I²C, la bande passante maximum que l'on peut mesurer est de 400 Hz**
**Si on veut utiliser les réglages supérieurs, il faudra câbler et programmer en mode SPI (cf. 3.2.1 « ADXL345 par SPI »).**

**Acquisition des données du microphone seul**

Le microphone est à sortie analogique.
Il est connecté à une entrée du Convertisseur Analogique Numérique du microcontrôleur.
La bande passante de ce microphone est [100 Hz – 15 kHz]
On rajoute un simple filtre RC du premier ordre (R = 182 kOhm, C = 47 pF, Fc = 18,6 kHz) pour éliminer d'éventuels parasites et éviter des phénomènes de repliement.

Procédure :

- Acquérir 512 échantillons :
  - attendre T microsecondes
  - récupérer 1 échantillon
- Transmettre les 512 échantillons

| T (µs) | Temps d'acquisition de 512 échantillons (µs) | Débit de données (kHz) |
|---|---|---|
| 100 | 55064 | 9,928 |
| 10 | 8836 | 57,944 |
| 1 | 4402 | 116,310 |
| /* sans */ | 3636 | 140,812 |

*Tableau 7 : Mesure débit ADMP401*

Ce qui est cohérent avec la documentation du CAN du STM32 ([55], chapitre 11.« Analog-to-Digital Converter (ADC) »).
Avec les réglages du convertisseur par défaut, on doit avoir une période d'échantillonnage de 6 à 7µs (ici 3636/512 = 7,1 µs).
On peut régler le temps de conversion au plus bas dans les registres de l'ADC.
J'ai ainsi obtenu une fréquence d'échantillonnage maximale de 500 kHz.
En pratique, il faudra jouer à la fois sur le réglage du temps d'acquisition de l'ADC et sur son déclenchement (par un *timer* ou une interruption).

**Selon la période et le temps de conversion, la fréquence d'échantillonnage peut être réglée de quelques dizaines à quelques centaines de kHz.**

### 2.10.3 Test du débit du Wi-Fi

Par défaut, la vitesse de la liaison entre le microcontrôleur et le composant Wi-Fi RN131C est de 9600 bauds, alors que le débit Wi-Fi peut aller jusqu'à 54 Mbps. Il est possible de modifier la vitesse de l'UART, en respectant un ordre pour ne pas perdre la liaison :
1 – modifier la vitesse du composant Wi-Fi (commande « set uart instant 9600 »)
2 – modifier la vitesse du composant SPI/UART (modifier les registre de division de l'horloge)

**Choix du protocole : TCP ou UDP ?**
Les différences et les cas d'utilisation de ces deux protocoles sont très bien documentés dans les chapitres 13 et 14 du livre « *Implementing 802.11 with microcontrollers* » [68].

| Protocole TCP | Protocole UDP |
|---|---|
| **T**ransmission **C**ontrol **P**rotocol<br><br>C'est un protocole de type « *handshaking* » qui établit la communication en 3 étapes :<br>- A demande à B s'il est disponible<br>- B acquitte la demande à A<br>- A établit le lien avec B<br>Ces étapes synchronisent et sécurisent la connexion.<br>Ce mode s'assure également que les paquets numérotés sont reçus dans le bon ordre et les renvoie le cas échéant. | **U**ser **D**atagram **P**rotocol<br><br>C'est un protocole de type « *I don't care* », c'est à dire que l'on envoie et on reçoit des paquets sans se soucier de savoir s'ils sont bien arrivés à destination.<br>Il est inutile d'établir une connexion.<br>L'UDP envoie les messages à la vitesse du microcontrôleur et de l'application qui l'utilise. |

Pour améliorer le débit, il est recommandé d'utiliser le protocole UDP, moins fiable, mais qui ne répète pas les paquets déjà envoyés.
Il nécessite :
- Un numéro de Port pour la Source
- Un numéro de Port pour la Destination
- Une adresse IP pour la Source
- Une adresse IP pour la Destination

Ainsi, chaque Socket est défini par un couple [Port + adresse IP] et rend le multiplexage possible.

Le composant RN131C supporte les deux protocoles TCP et UDP. On va donc adapter le programme pour le configurer en UDP.

On rajoute dans l'automate une commande permettant de régler la vitesse de l'UART de 2400 à 921600 bauds.
Côté ordinateur, on développe une application pour mesurer le débit en réceptionnant un tableau de 1024 fois la valeur « 0 ».

| Commande | Vitesse théorique | Vitesse mesurée TCP | Ratio TCP | Vitesse mesurée UDP | Ratio UDP |
|---|---|---|---|---|---|
| 0 | 2400 | 1916 | 79,8 | 1920 | 80 |
| 1 | 4800 | 3800 | 79,2 | 3842 | 80 |
| 2 | 9600 | 7563 | 78,8 | 7640 | 79,6 |
| 3 | 19200 | 14965 | 77,9 | 15292 | 79,6 |
| 4 | 38400 | 29930 | 77,9 | 29930 | 78 |
| 5 | 57600 | 42355 | 73,5 | 44896 | 78 |
| 6 | 115200 | 79602 | 69 | 89792 | 78 |
| 7 | 230400 | 143897 | 62,4 | 181032 | 78,6 |
| 8 | 460800 | 244000 | 52,9 | 362064 | 78,6 |
| 9 | 921600 | 244000 | 26,5 | 362064 | 39,3 |

*Tableau 8 : Comparaison des mesures de vitesse en TCP et UDP*

En mode TCP, de 2400 à 57600 bauds, le transfert des données utilisateurs est effectué à 75-80% de la vitesse théorique puis baisse régulièrement pour tomber à 27 % à 921600 bauds.
En mode UDP, le transfert des données utilisateurs reste stable jusqu'à 460800 bauds, mais tombe à 40% pour le réglage maximum de 921600 bauds.
Cette limitation est causée par le réglage de la vitesse du SPI. En effet, la vitesse du SPI est fixée en divisant l'horloge du microcontrôleur qui est de 72 MHz. Les valeurs possibles en divisant par 32 et 16 sont 2,25 MHz et 4,5 MHz Comme la passerelle SC16IS750 admet une fréquence maximum de 4 MHz, j'ai du choisir la valeur inférieure de 2,25 MHz.

**Ces résultats peuvent laisser espérer une fréquence d'échantillonnage maximale de :**
**15 kHz en mode TCP (i.e 244 kb / 16bits par échantillon)**
**22,6 kHz en mode UDP (i.e 362 kb / 16bits par échantillon)**

Théoriquement à 921600 bauds, si on atteint un ratio de 78,6 % (soit 921600 × 0,786 = 724378) on aurait pu espérer une fréquence d'échantillonnage de 724378/16 = 45,3 kHz.

### 2.10.4 Robustesse & fiabilité

Pour vérifier l'intégrité des données lors de la transmission en Wi-Fi, on va envoyer des valeurs connues pour simuler plusieurs voies analogiques :

Voie 1 = 600
Voie 2 = 1200
Voie 3 = 1800
Voie 4 = 2400

On s'assure alors, côté réception, que l'ordinateur enregistre bien dans un fichier texte les ***N*** échantillons reçus dans l'ordre :

| Echantillon n° | Voie 1 | Voie 2 | Voie 3 | Voie 4 |
|---|---|---|---|---|
| 1 | 600 | 1200 | 1800 | 2400 |
| 2 | 600 | 1200 | 1800 | 2400 |
| … | 600 | 1200 | 1800 | 2400 |
| N-1 | 600 | 1200 | 1800 | 2400 |
| N | 600 | 1200 | 1800 | 2400 |

Ce procédé a permis de mettre en évidence des décrochages si la période d'échantillonnage choisie était trop élevée ou si les paramètres de la liaison Wi-Fi n'étaient pas cohérents.

Ensuite, pour vérifier la fréquence des signaux enregistrés, j'utilise un générateur de fréquence pour injecter une sinusoïde qui simule un capteur analogique en entrée du CAN et sur une entrée d'oscilloscope.
Dans le programme côté ordinateur, je calcule la fréquence du sinus reçu (période entre deux détections des passages par zéro dans le même sens).
Je m'assure alors que cette valeur et celle de l'oscilloscope sont identiques.
Cette opération vérifie le théorème de Shannon-Nyquist : la fréquence d'échantillonnage doit être au moins le double de la fréquence du sinus.
Le vrai facteur limitant n'est pas le CAN du microcontrôleur, mais la vitesse de transmission du Wi-Fi, comme nous l'avons constaté dans 2.10.3 « Test du débit du Wi-Fi ».

**Alors que la fréquence d'échantillonnage du CAN peut facilement attendre plusieurs centaines de kilohertz, la transmission du Wi-Fi limitera celle-ci à une vingtaine de kilohertz pour une voie mesurée.**

Des mesures plus détaillées seront présentées dans le chapitre 4.8 « Résultats & Performances ».

### 2.10.5 Observation des signaux temporels

Nous pouvons contrôler visuellement à l'écran les signaux temporels issus du microphone et de l'accéléromètre triaxes. On voit bien le signal du microphone (en noir) qui varie si on siffle et ceux de l'accéléromètre (rouge = X, vert = Y et bleu = Z) qui varient quand on le tamponne sur la table.

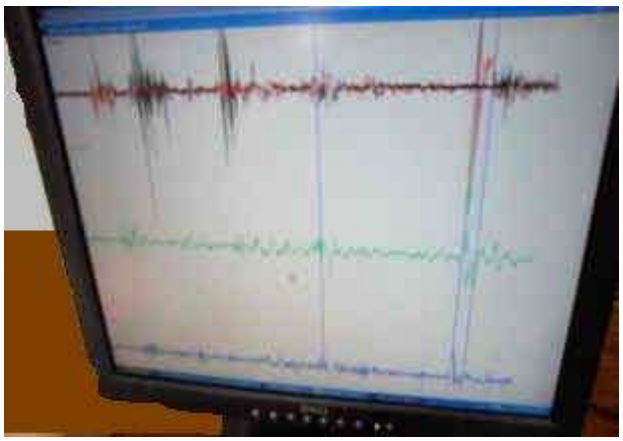


*Figure 10 : Capture d'écran Monitoring de la première maquette*

L'image ci-dessous représente tous les éléments électroniques assemblés sur la première maquette.

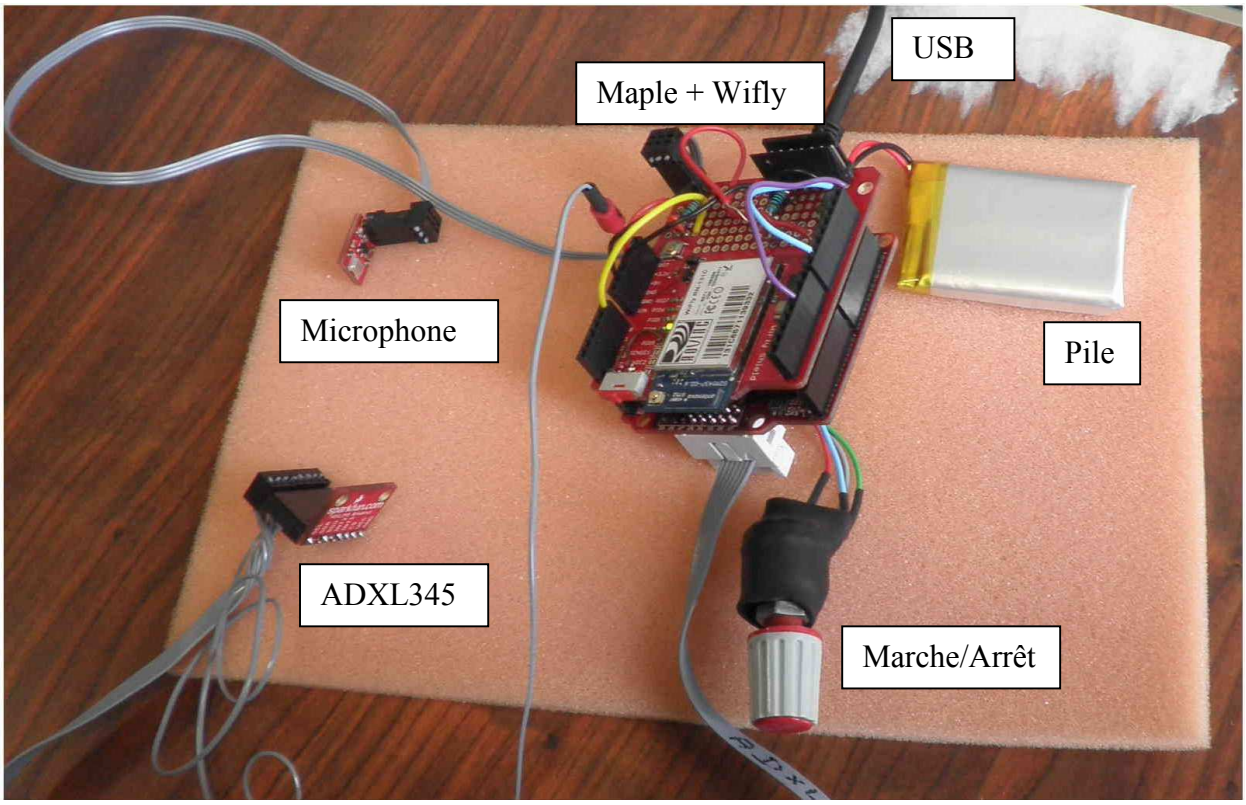



*Figure 11 : Photo de la première maquette*

## 2.11 Evolutions envisagées de la maquette

Cette première maquette répond en partie à nos exigences. Cette première étude de faisabilité démontre que la solution choisie est réalisable et que l'on peut transmettre par Wi-Fi des échantillons issus à la fois d'un capteur analogique et d'un capteur numérique.

Nous pouvons maintenant envisager d'améliorer son fonctionnement par :

- l'adjonction d'autres types de capteurs
- le développement d'une interface logicielle plus conviviale
- l'augmentation du débit de transmission des données, par programmation
- la sauvegarde des données

# 3 Evolutions électroniques : diversifier les capteurs

Dans la littérature, les capteurs industriels sont classés par famille de mesure [69][70] :

- Température, humidité
- Longueurs et déplacements
- Forces et accélérations
- Capteurs magnétiques à effet Hall
- Vitesses et positions
- Débits et niveaux
- Capteurs optiques

Pour diversifier les types de capteurs et rendre la maquette plus polyvalente, nous allons mettre en œuvre des capteurs analogiques de vibration et de courant.
Leur bande passante doit être supérieure à 5 kHz sous une alimentation de 3V.
Pour respecter le cahier des charges qui spécifie des capteurs non invasifs et amovibles, on fixera les capteurs avec de la Patafix UHU® ou des bandes Velcro®.
Le document « *Procedure_de_cablage.pdf* » fournit dans l'Annexe B. « Documents » contient toutes les informations pour réaliser les développements électroniques de ce chapitre.

## *3.1 Capteurs analogiques*

### 3.1.1 Capteur de courant

Une des applications possibles au sein du laboratoire consiste à mesurer le courant triphasé alimentant un moteur. La difficulté réside dans le choix d'un capteur qui ne soit pas invasif. En effet, la plupart des capteurs de courant doivent encercler le câble où l'on veut mesurer le courant.
On trouve dans la littérature des descriptions détaillées sur les principes physiques, la géométrie des capteurs, les composantes électromagnétiques, la galvano-magnétique, les matériaux…[71]
On retiendra qu'un bon capteur à effet Hall doit être réalisé dans un matériau :

- possédant une faible concentration de porteurs
- ces porteurs sont très mobiles,
- avec une large bande de gap (c'est à dire une faible densité de porteurs intrinsèque)
- Le coefficient Hall $R_H$ et le coefficient de mobilité Hall $\mu_H$ doivent être grands

Le choix s'est porté sur le composant MLX91205 de chez Melexis Inc [72], un astucieux circuit intégré adapté à un montage non invasif. On l'utilise comme capteur du champ magnétique (en Tesla), qui est l'image instantanée du courant à mesurer (en Ampère). L'induction magnétique B est proportionnelle au courant I. De fait, il suffira de le poser sur le fil où l'on veut mesurer le courant et de le faire tenir avec du Velcro®.

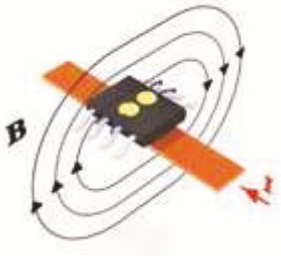



*Figure 12 : Principe de fonctionnement du MLX91205*

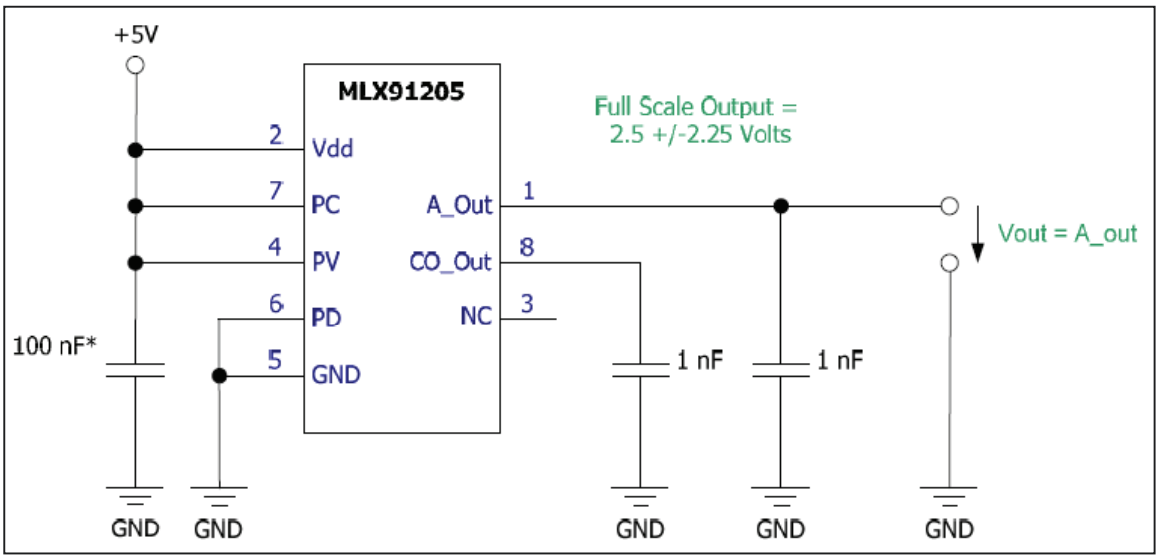
***Caractéristiques***

| | |
|---|---|
| Alimentation | 4,5 à 5,5 Vcc |
| Courant | 10 à 16 mA |
| Sensibilité | 280 mV/mT |
| Gamme | ±10 mT (Low Field) |
| Bande passante | DC – 100 kHz |

Ce capteur nécessite une alimentation de 5V. On ajoutera une pompe de courant pour produire une tension de 5V à partir du 3,3V (voir 3.4 « Gestion des alimentations »).

***Montage***

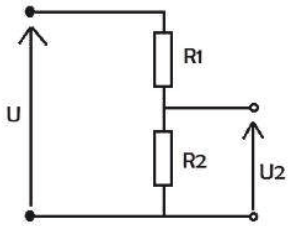
Le schéma de montage est relativement simple puisqu'il ne requiert que 3 capacités.
Comme nous désirons une sortie *Vout* proportionnelle à 3,3V (et non pas 5V), nous rajouterons simplement un pont diviseur en sortie. Trois capteurs de courant identiques ont été fabriqués.




Le composant doit être alimenté en +5V.
Comme l'entrée de notre CAN est limité à +3,3V, il faut un pont diviseur.

En choisissant R1 = 10 kΩ et R2 = 18 kΩ, on a :

$$U2 = (U \times R2) / (R1 + R2)$$
$$U2 = (5 \times 18) / (10 + 18)$$
$$U2 = 3{,}21$$




Le montage tient sur une simple platine CMS d'environ 3 × 2 cm que l'on percera sous le composant principal. Il suffit de l'envelopper de gaine thermo-rectractable pour l'isoler et de percer également cette gaine (Figure 13). Le capteur est prêt à l'emploi : on le pose sur le câble où l'on veut mesurer le courant et on le fixe avec une bande Velcro® (Figure 14).

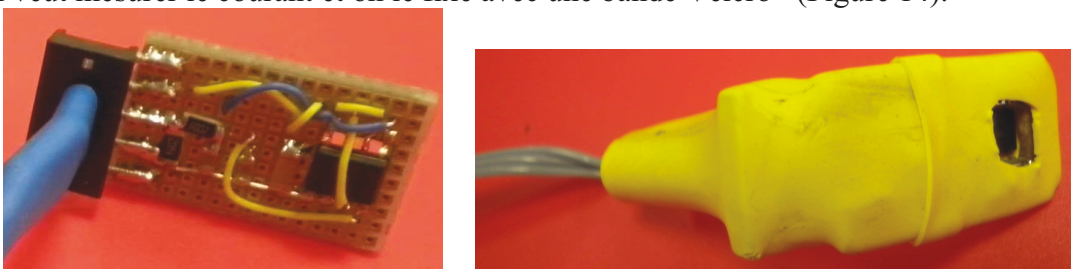


*Figure 13 : Montage capteur de courant (vue de dessus et dessous)*

***Exemple d'utilisation***

Dans l'armoire d'alimentation de la plate-forme GOTIX [10], se trouve l'armoire de commande du courant d'alimentation triphasé. Les trois câbles sont instrumentés par des pinces ampèremétriques installées à demeure (en rouge).

Nous instrumentons un des câbles d'alimentation avec :

- une pince ampèremétrique de laboratoire (en noire), modèle
- le capteur MLX91205 fixé avec du Velcro® (en vert)

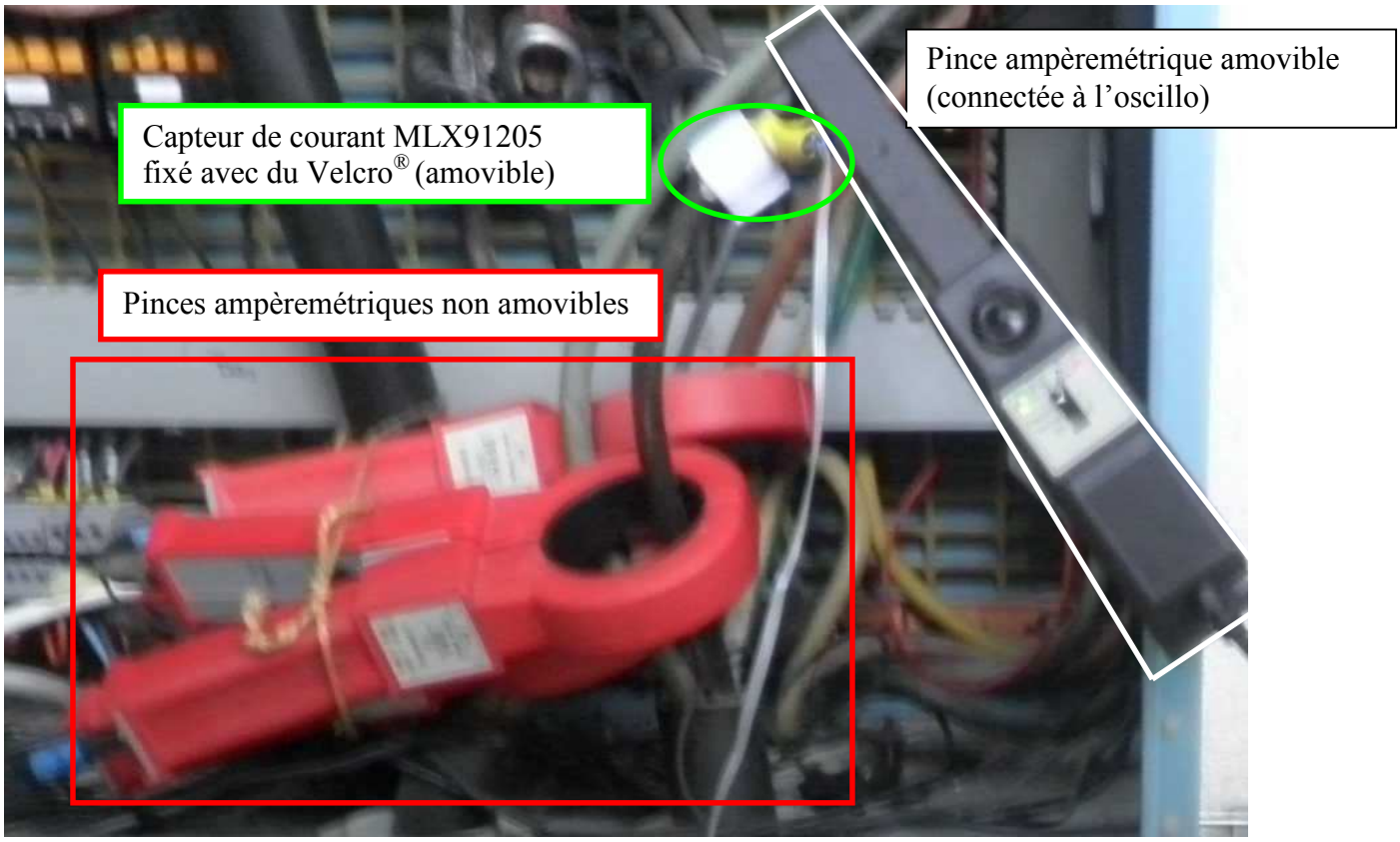



*Figure 14 : Exemple d'utilisation de capteurs de courant*

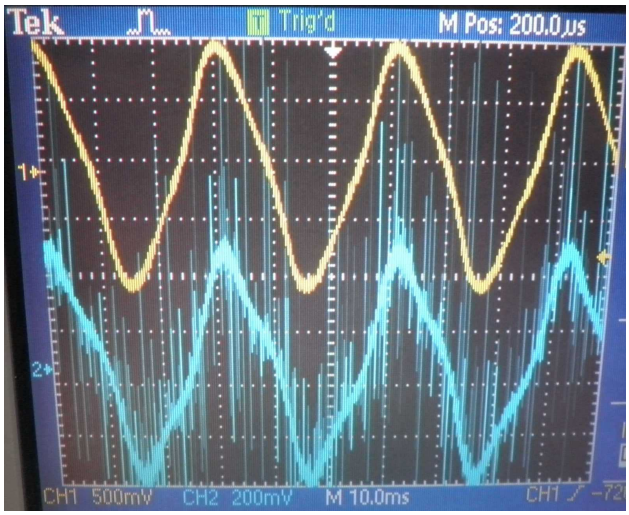
Caractéristiques des deux instruments :

| | **Pince ampèremétrique** | **Capteur de courant** |
|---|---|---|
| Modèle | Tektronix A622 | MLX91205 Low Field |
| Gamme | 100 A | +/- 10 mT |
| Sensibilité | 10 mV/A | 280 mV/mT |
| Calibré en courant | Oui | Non |
| Bande Passante | DC –100 kHz | DC – 100 kHz |
| Alimentation | Pile 9V | 5V (à partir du 3,3V) |
| Prix | 577 euros HT | 4,20 euros HT |

**Mesures**

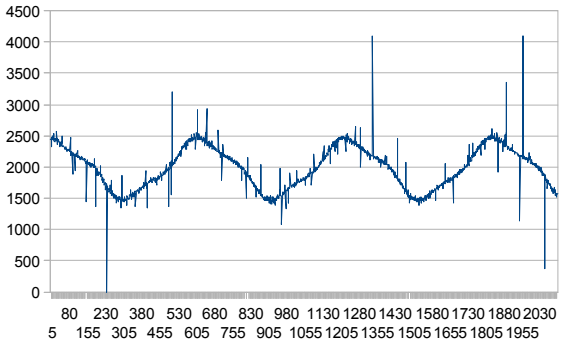
Sur la capture d'écran de l'oscilloscope, nous avons l'image analogique du courant mesuré par :

- la pince ampèremétrique (en orange)
- le capteur de courant MLX91205 (en bleu)




Voici les données transmises par Wi-Fi par notre maquette visualisé sur l'écran de l'ordinateur :

Et voici un extrait des données enregistrées dans un fichier texte lu dans un tableur :




La fréquence d'échantillonnage de la maquette est de 20 kHz.

Nous constatons que l'allure du courant enregistrée par l'ordinateur est similaire à celle de référence enregistrée par la sonde ampèremétrique. Pour un capteur qui ne coûte que quelques euros, on atteint des performances dignes d'une sonde de plusieurs centaines d'euros.

Les résultats du capteur ne sont pas filtrés et comportent des parasites. Ceux-ci devraient disparaître avec l'utilisation des filtres RC développés plus loin (voir 3.1.5 « Filtrage »).

Le fabricant préconise également que le montage soit blindé dans une boite en métal.

### 3.1.2 Capteur de force

Dans le laboratoire, il y a une maquette d'une plate-forme de forage pétrolière.
Celle-ci est équipée de capteurs de force correspondant exactement à nos critères : 3 fils, alimentée en 3,3V. Il suffit de rajouter le connecteur permettant de le brancher sur notre maquette, sans aucune électronique supplémentaire.


*Figure 15 : Capteur de force*

***Caractéristiques***

| | |
|---|---|
| Modèle | FT2431-0000-0100-L (MSI Sensors) |
| Alimentation | 3 à 5Vdc |
| Gamme | ± 100 lbf |

La documentation technique ne précise pas la bande passante que l'on peut estimer à quelques centaines de hertz au vu de l'utilisation de ce type de composant.

***Mesures***

Aucune mesure n'a été effectuée en situation réelle avec ce capteur.
Nous nous sommes bornés à vérifier que le fait de tirer manuellement les extrémités induisait bien une variation du signal à l'écran.

### 3.1.3 Accéléromètre large bande - ADXL001

C'est un accéléromètre MEMS monté sur une platine possédant déjà une électronique de conditionnement pour faciliter son évaluation [73].
Il est très difficile de trouver des capteurs accélérométriques triaxiaux analogiques en technologie MEMS ayant une large bande passante. Il existe des capteurs « trois axes » avec une bande passante de 1 à 3 kHz. Pour des bandes passantes plus élevées, il faudra se contenter d'un capteur mono-axial.
Ce type de capteur est indispensable pour le contrôle vibratoire des machines en milieu industriel. Il permet de surveiller la bonne santé de la machine et d'intervenir en cas de besoin.

***Caractéristiques***

| | |
|---|---|
| Alimentation | 3,3 Vcc |
| Gamme | ± 70g |
| Bande passante | DC – 22 kHz |

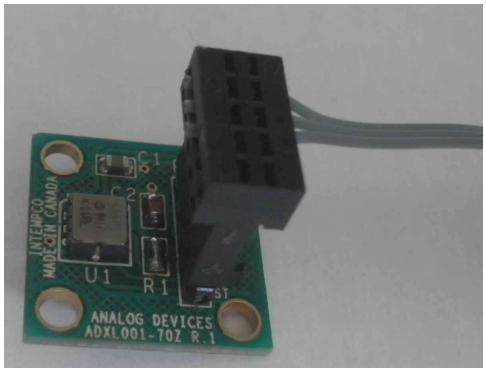
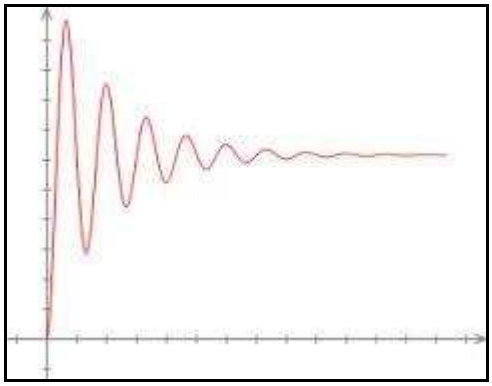



*Figure 16 : Accéléromètre ADXL001*

**Montage**
La platine possède des plages pour souder un réseau RC de filtrage. Celui-ci peut être court-circuité si on utilise un autre filtrage, ce qui est notre cas (voir 3.1.5 « Filtrage »).
Nous court-circuitons donc R1 et ajoutons une capacité de 22 nF sur C1.
Comme les autres capteurs analogiques, nous montons un connecteur pour le relier à la maquette.

**Mesures**
Aucune mesure n'a été effectuée en situation réelle avec ce capteur.
Nous nous sommes bornés à vérifier qu'en heurtant le capteur sur la table, on obtenait bien un signal transitoire de ce style :



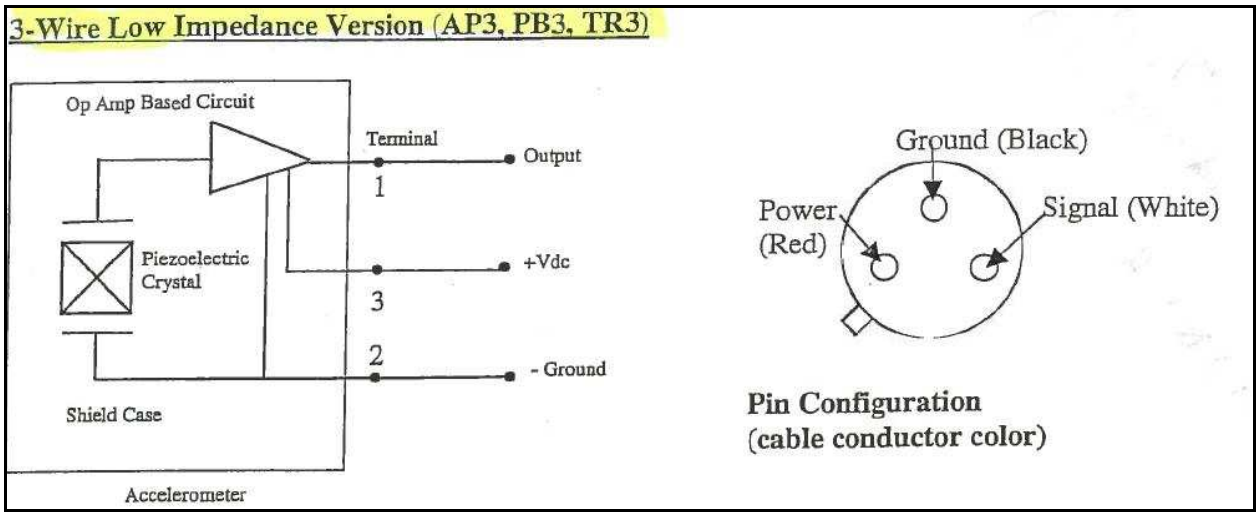
### 3.1.4 Accéléromètre large bande - PCB 3 fils

Ce capteur a été gracieusement offert comme échantillon par un commercial de PCB PiezoTronics France [74]. Qu'il en soit chaleureusement remercié.

***Caractéristiques***

| | |
|---|---|
| Modèle | Capteur PCB 660series-0802<br>Low Profile TO-5<br>3-Wire, Low-Power Configuration |
| Alimentation | 3 à 5Vdc |
| Gamme | ± 200g |
| Bande passante | 32 Hz – 10 kHz |

***Schéma***




***Montage***

Comme c'est un capteur « 3 fils », il suffit de rajouter un connecteur pour l'interfacer avec la maquette. Le filtrage sera réalisé par un réseau RC (voir 3.1.5 « Filtrage ») sur la maquette.
Le câblage c'est révélé plus difficile que prévu et le capteur a été cassé lors du montage (patte d'alimentation *+Vdc*).
C'est d'autant plus dommageable qu'il était exactement ce que nous recherchions pour les mesures de vibrations. On aurait pu comparer les mesures avec celles réalisées avec l'autre capteur large bande ADXL001 décrit auparavant.

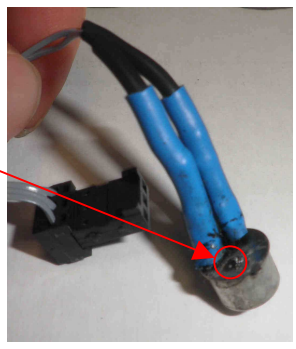


*Figure 17 : Accéléromètre PCB*

### 3.1.5 Filtrage

Le filtrage des entrées analogiques s'effectue par un banc de filtres RC montés sur un connecteur tulipe. Ce filtre du premier ordre n'est pas très performant, mais il est suffisant pour notre démonstrateur. Le filtrage est identique pour les 6 voies analogiques.

Voici les 5 bancs de filtres réalisés :

| Module n° | Fréquence Coupure | Capacité | Résistance |
|---|---|---|---|
| 0 | aucune | / | fil AWG24 |
| 1 | 16 kHz | 100 pF | 100 kΩ |
| 2 | 10,6 kHz | 100 pF | 150 kΩ |
| 3 | 4,8 kHz | 100 pF | 330 kΩ |
| 4 | 2,34 kHz | 100 pF | 680 kΩ |

*Tableau 9 : Valeurs des filtres RC*

Il suffit de sélectionner le banc désiré selon la fréquence de coupure et de l'insérer sur l'embase tulipe prévu à cet effet.
Le filtrage sera effectué simultanément sur les 6 capteurs analogiques qui seront connectés.

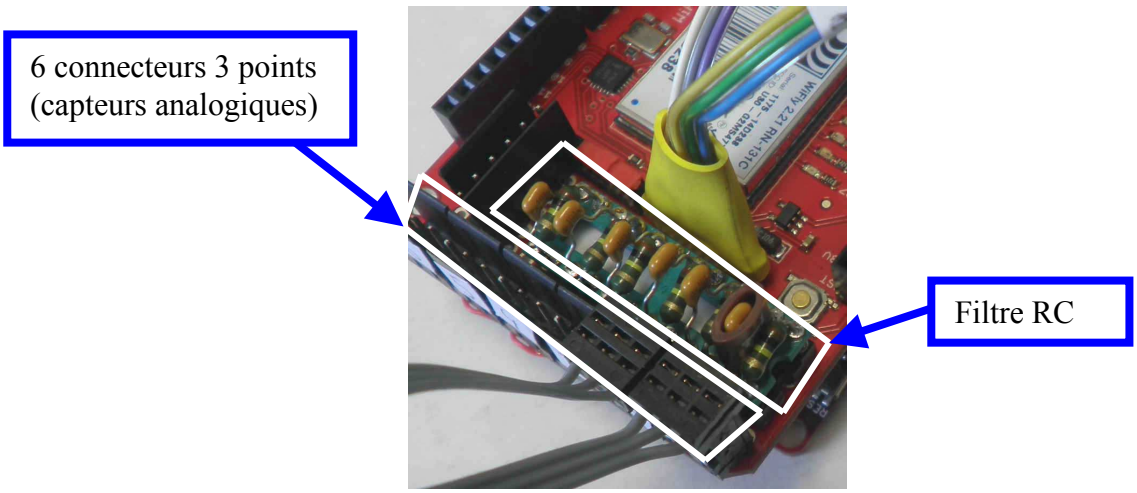



*Figure 18 : Exemple d'utilisation du filtre RC*

## 3.2 Capteurs numériques

### 3.2.1 ADXL345 par SPI

L'accéléromètre numérique ADXL345 vu au chapitre 2.7 « Mise en œuvre d'un capteur numérique : accéléromètre » supporte deux modes de communication : I²C et SPI.
L'utilisation du mode I²C est déconseillée si on veut utiliser les bandes passantes supérieures à 400 Hz (voir 2.10 « Résultats & Performances »). La datasheet recommande d'utiliser le mode SPI à 2 MHz minimum pour assurer un bon fonctionnement dans toute la gamme.
Nous avons vu comment gérer un port SPI pour communiquer avec la platine *WiFly*.
Le microcontrôleur STM32 peut gérer un deuxième port SPI.
Comme nous avons déjà développé les fonctions bas niveau dans une librairie pour la partie Wi-Fi (voir 2.8.2 « Adaptation des fonctions bas niveau »), il suffira de dupliquer ce code avec les numéros de broche du deuxième port SPI.
Les fonctions d'écriture/lecture des registres seront adaptées à la communication SPI.

Par ailleurs, le bit *DATA_READY* indiquant que des nouveaux échantillons sont disponibles, peut être programmé comme interruption. On s'en servira comme signal *IRQ* pour déclencher la lecture des échantillons des 3 axes.
Sur la platine *Maple*, ce port SPI (*pins* 31, 32, 33, 34) et des *pins* configurables en entrée numérique sont accessibles sur un connecteur, dont la 35.
J'ai enlevé ce connecteur pour le remplacer par un autre dédié au capteur ADXL 345, de sorte que toutes les *pins* nécessaires soient au même endroit. Seul le 3,3V sera prélevé ailleurs sur la platine.

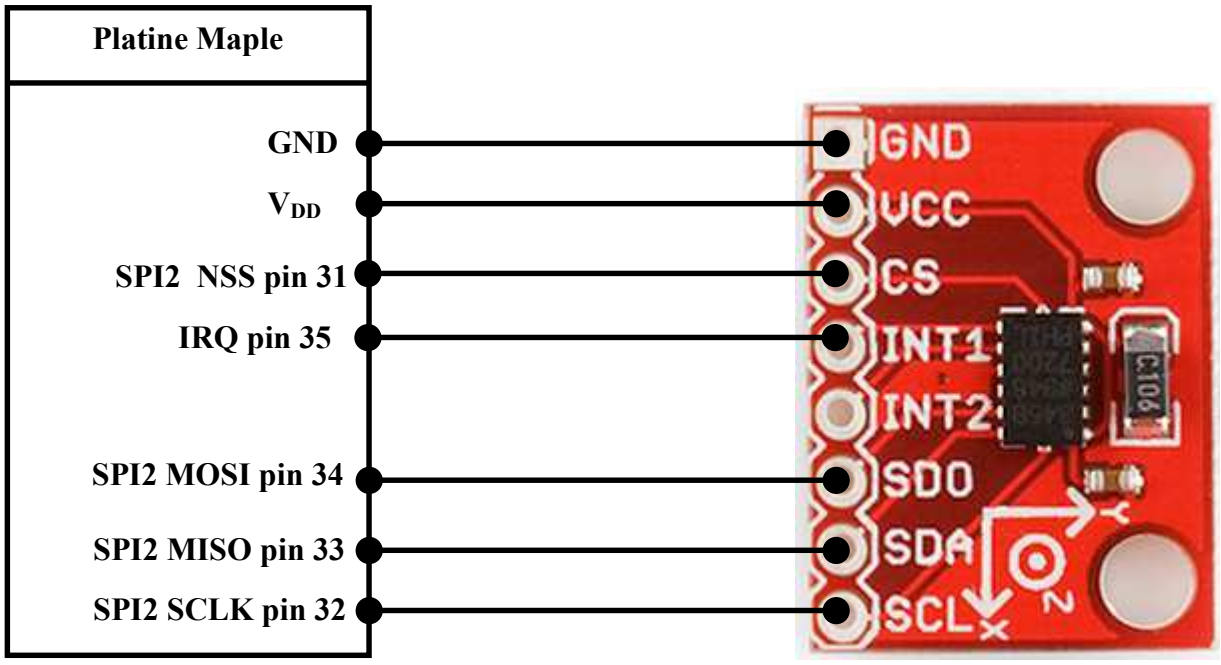



*Figure 19 : Câblage ADXL345 en mode « 4-wire SPI »*

On s'assurera du bon fonctionnement de la sélection de la bande passante en mode SPI dans 4.8.1 « Chronométrage ».

### 3.2.2 I²C pour IMU

Dans notre stratégie de diversification des capteurs, nous avons envisagé d'implémenter une centrale inertielle appelé IMU (Inertial Motion Unit).
L'IMU (Fusion Board - ADXL345 & IMU3000 [75]) choisi est un capteur numérique regroupant un accéléromètre 3 axes et un gyroscope 3 axes, soit 6 degrés de liberté mesurés.
On gère la récupération des données sur un port I²C par interruption sur un signal *IRQ*.

***Caractéristiques***

| | |
|---|---|
| Alimentation | 3,3V |
| Gamme accéléro | ADXL 345 : ±2, 4, 8 ou 16g |
| Gamme gyro | IMU 3000 : 250 à 2000°/s |

***Montage***
On monte un connecteur dédié sur la platine *WiFly* pour interfacer ce capteur sur un port I²C.

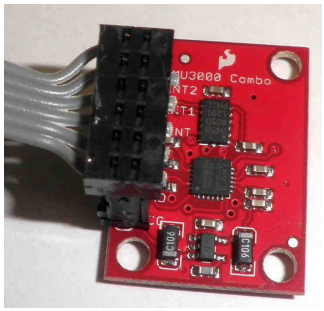



*Figure 20 : Capteur IMU*

***Remarque***
La gestion de l'IMU n'est pas implémentée dans le programme fourni à la date de la fin du stage.
Pour la partie programmation, on pourra s'inspirer de l'exemple développé pour gérer le capteur ADXL345 en mode I²C (voir 2.7 « Mise en œuvre d'un capteur numérique : accéléromètre »).

### 3.3 WiFly : câblage de l'IRQ

Pour voir si des commandes sont réceptionnées sur le Wi-Fi, on scrutait régulièrement la ligne de réception. Cette opération prend du temps et interrompt l'acquisition périodiquement.
Il existe un autre moyen pour être informé de la présence d'une commande.
En effet, la platine prévoit le contrôle d'une ligne d'interruption de type *IRQ* sur la *pin* 7 du microcontrôleur (voir schéma ci-dessous).

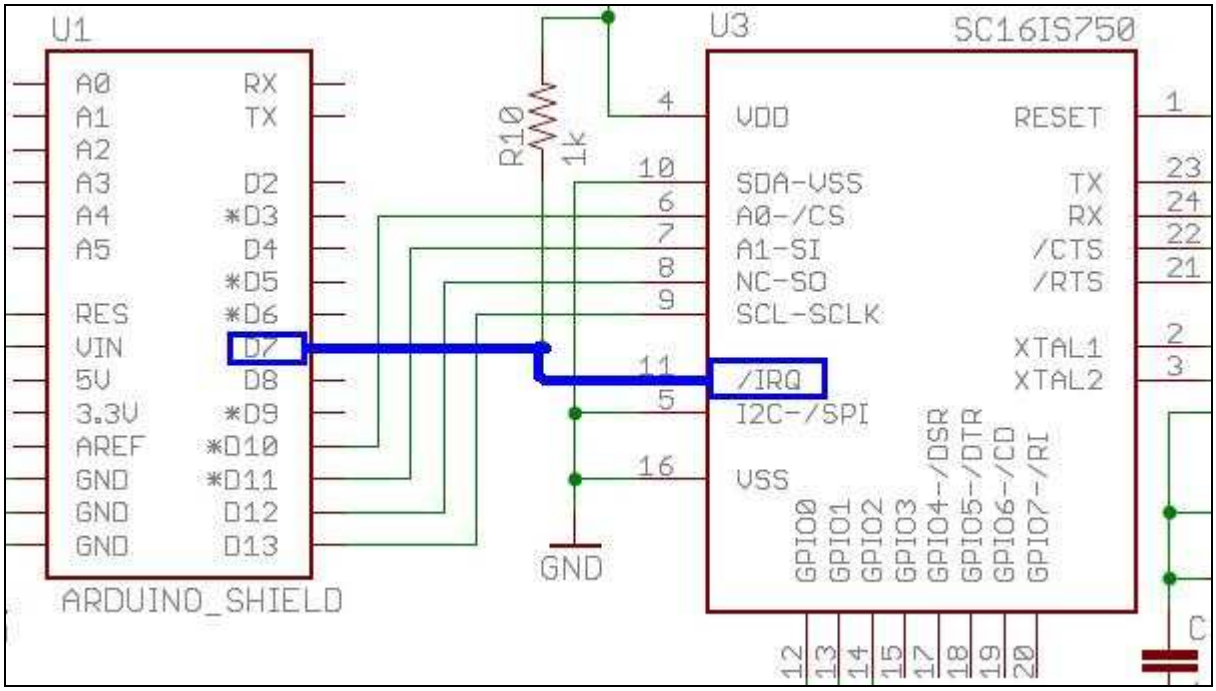



*Figure 21 : Détail du Schéma de la carte WiFly (IRQ)*

Il suffit alors de configurer cette ligne pour indiquer qu'un caractère a été reçu.
Pour cela, on modifie la méthode d'initialisation du SPI/UART dans « `SpiUart.cpp` » :

```
//Configure IRQ on receive data (RHR),
//will put pin7 low when receiving data from PC
writeRegister(IER, IRQ_RHR); //0x1
```

## 3.4 Gestion des alimentations

### 3.4.1 Sélection de l'alimentation

A la livraison, la sélection du mode d'alimentation se fait par cavalier sur un connecteur monté sur la platine « *Maple* ». On peut sélectionner les 3 modes d'alimentation :

- Batterie
- USB
- alimentation externe (non utilisé)

On remplace ces cavaliers par un commutateur à glissière pour choisir d'alimenter soit par l'USB soit par la batterie.
On ajoute également un switch pour recharger la pile à partir de l'USB. Cette fonction est intégrée sur la platine *Maple* par un circuit intégré « chargeur de batterie LiPo ».

### 3.4.2 Pompe de charge 5V

Certains capteurs nécessitent d'être alimentés en 5V (cf. 3.1.1 « Capteur de courant »).
Or la tension régulée sur les platines *Maple/WiFly* est de 3,3V.
Il faut donc rajouter une pompe de charge pour obtenir une alimentation 5V.
Le MAX1595 permet de fabriquer du 5V à partir du 3,3V avec quelques composants additionnels (voir Figure 22).

Le platine électronique de ce montage tient sur une platine d'environ 1 cm², qui sera fixé au dos de la carte Wi-Fi avec du scotch double face.

Nous devons alimenter de 1 à 6 voies analogiques. Pour faciliter le câblage, l'alimentation sera :

- 3,3 V pour les voies 1 à 3
- sélectionnable à 3,3V ou 5V, par un interrupteur, pour les voies 4 à 6

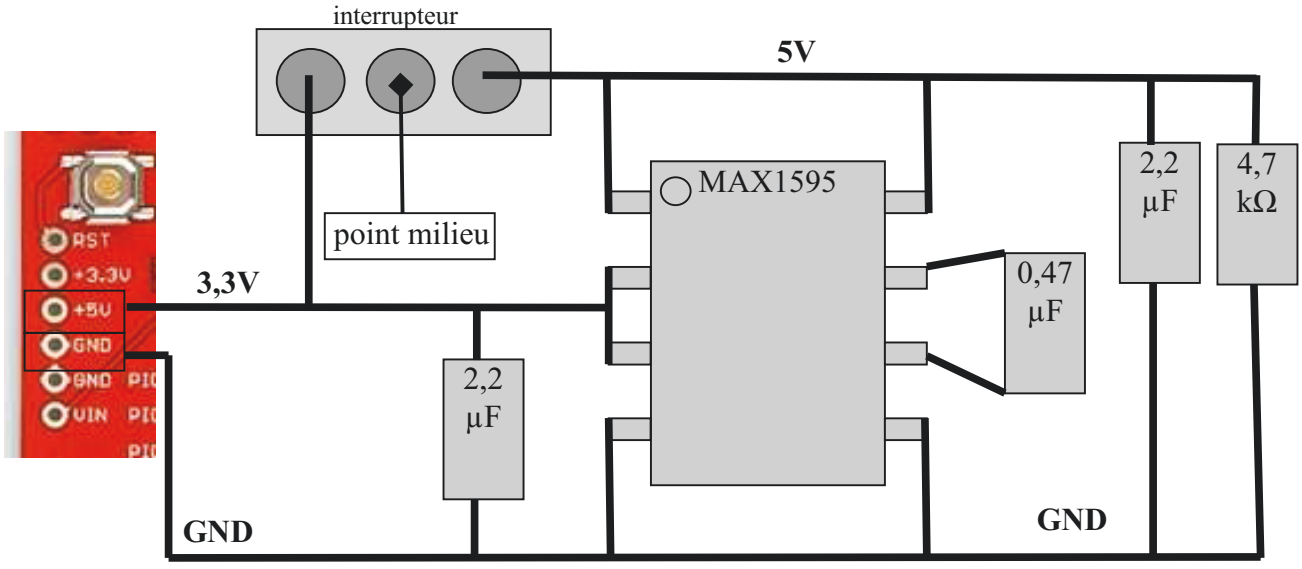



*Figure 22 : Pompe de charge 5V*

Cette pompe de charge peut fournir jusqu'à 125 mA, ce qui devrait être largement suffisant pour des capteurs de type MEMS (le capteur de courant MLX91205 requiert 10 mA).

### 3.4.3 Mesure de courant

Pour se faire une idée de la consommation de la maquette, j'ai placé une résistance de shunt de 0,47 ohm en alimentant par l'USB (4,9V) afin d'avoir une tension stable et continue.

Si on démarre juste la platine *Maple* (en initialisant les fonctions dont nous avons besoin), elle consomme 40 mA, ce qui est conforme à la documentation (de 32,8 mA au repos à 50 mA quand tout est activé).
En ajoutant la platine *WiFly*, on consomme environ 100 mA en moyenne.
Le fait d'activer 1 à 6 voies analogiques de CAN augmente peu la consommation (de l'ordre de 2-3 mA par voie) avec une fréquence d'échantillonnage de 2 kHz.
La fréquence d'échantillonnage influe également légèrement la consommation (compter 10 mA de plus à 20 kHz).

En fonctionnement (échantillonnage et transmission) la consommation observée à l'oscilloscope oscille entre 2 valeurs selon que l'on est en train d'émettre ou pas.

| *(shunt 0,47 ohm)* | **Consommation en mA** | **Puissance (en W)** |
|---|---|---|
| Sans transmission (Avg) | 102 | 0.34 |
| Avec transmission (Peak-Avg) | 268 | 0.88 |
| Total | 370 | 1.22 |

Sans transmission, on a une consommation fixe d'environ 100 mA. Il faut ensuite ajouter une consommation proportionnelle à la fréquence d'échantillonnage.
La puissance maximale consommée est d'environ 1,2 W (370 mA sous 3,3 V).

Pour le RN131C (composant principal du *WiFly*), la documentation donne 40 mA en réception et 210 mA en transmission.

Dans le détail, on note qu'une durée de l'impulsion varie de 0,88 à 1,26 ms selon la fréquence d'échantillonnage : plus on échantillonne vite, plus la fréquence des impulsions de consommations augmente.
La fréquence de l'impulsion au repos est de 10 Hz (ce qui correspond au réglage par défaut du beacon à 100 ms).

En définitif, on a une valeur moyenne de 100 mA (50mA pour le STM32 et 50 mA pour le *WiFly* en mode réception). Une trame de beacon toutes les 100 ms consomme 210 mA en transmission.
Plus on échantillonne vite, plus on envoie des données souvent, plus les trains d'impulsions de 210 mA sont rapprochés.

### 3.4.4 Batterie Lithium Ion Polymère

La maquette sera alimentée par une batterie Lithium Ion Polymère (LiPo) de 3,7V / 1000 mAh.
Cette pile plate est rechargeable par la platine *Maple* à partir de la tension de la prise USB.
Ce type de batterie n'est distribué en France que par quelques revendeurs, comme EVOLA [76].

## 3.5 Maquette finale

Visuellement, la maquette finale se présente comme un module où l'on peut connecter les capteurs, la pile et le cordon USB.

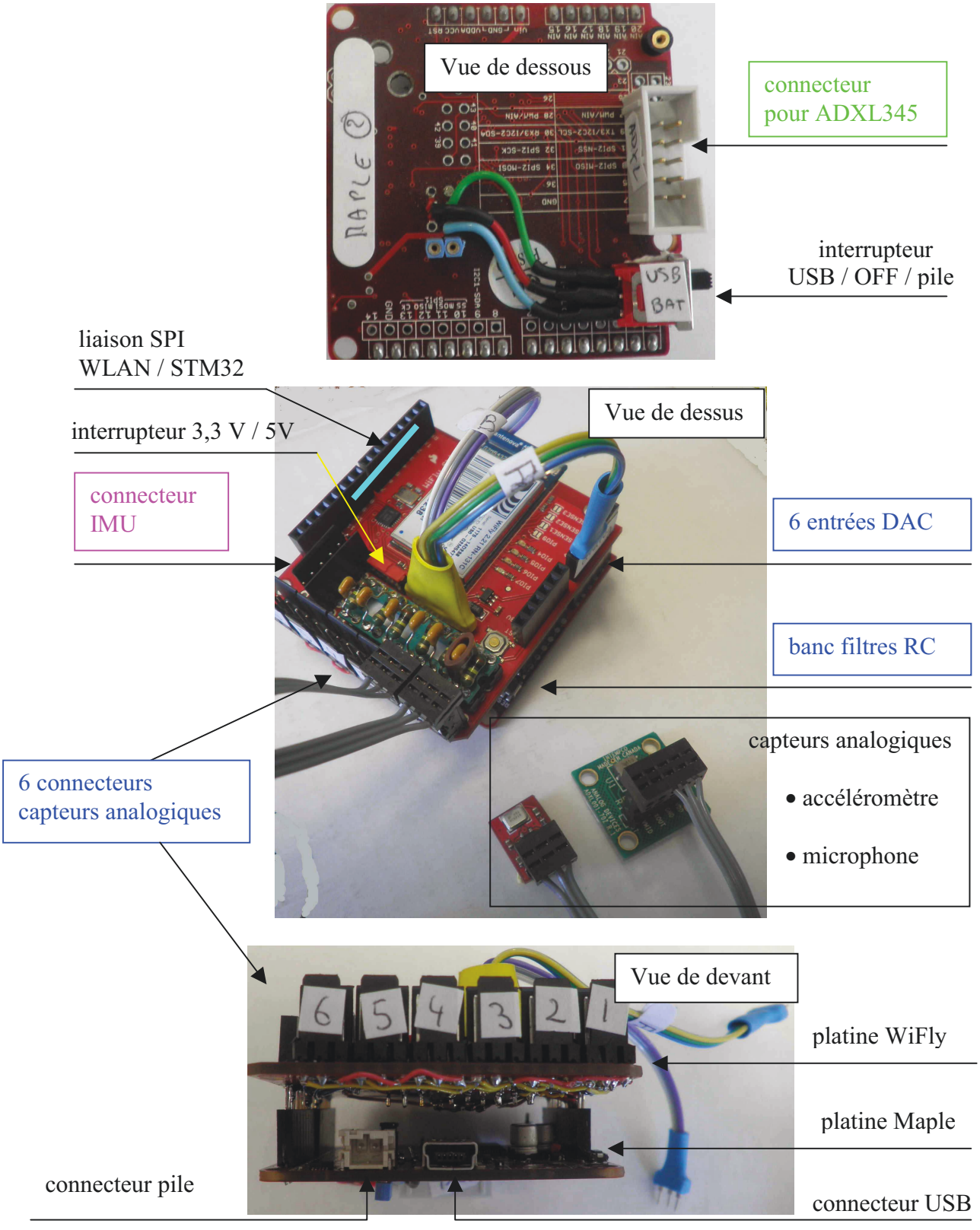



*Figure 23 : Vues de la maquette finale*

Au niveau du microcontrôleur STM32F103, on utilise les E/S du schéma :

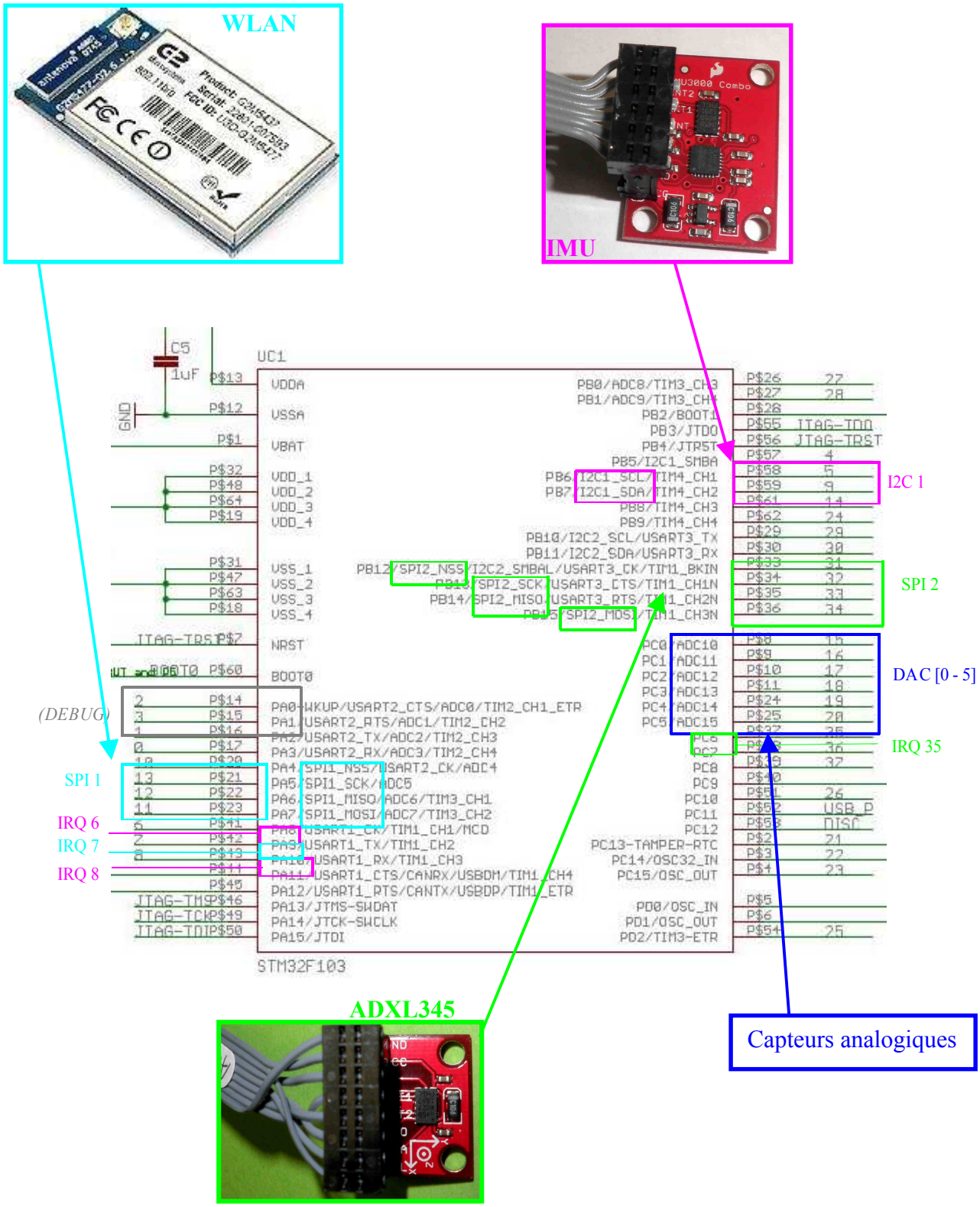



*Figure 24 : STM32 implantation des E/S*

## 3.6 Coût

Les contraintes budgétaires de ce projet sont très strictes et ne doivent pas dépasser quelques centaines d'euros.

Les principaux composants et platines utilisés sont listés dans le tableau ci-dessous.

| Composant | Description | Fabricant | Ref. Achat | Prix (HT) |
|---|---|---|---|---|
| Maple | Platine de test pour STM32 | LeafLabs | LEXTRONIC DEV-10664 | 49 |
| ADXL345 | Accéléromètre triaxe numérique | Analog Devices | LEXTRONIC SEN-09836 | 21 |
| ADMP4001 | Microphone MEMS avec ampli | Analog Devices | LEXTRONIC BOB-09868 | 9,9 |
| WiFly | Platine WLAN pour Arduino | Sparkfun | LEXTRONIC WRL-09954 | 72 |
| MLX91205 | 3 Capteurs de courant | Melexis | DIGIKEY MLX91205KDC-LB-ND | 12,6 |
| IMU Centrale inertielle | Module accéléromètre/gyroscope 6 axes | Sparkfun | LEXTRONIC SEN-10252 | 45,9 |
| MAX1595 | Pompe de charge 5V | Maxim | FARNELL MAX1595EUA501673185 | 4,7 |
| ADXL001 | Carte d'évaluation MEMS accéléromètre monoaxe 70g | Analog Devices | FARNELL EVAL-ADXL001-70Z | 77,5 |
| Batterie | Lithium Polymère plate 3,7V 1A | | EVOLA | 7,5 |
| ***Total*** | | | | ***300*** |

Ce qui nous amène à un total de 300 euros hors taxe environ (et hors frais de port).
Auquel il faut rajouter les fournitures de câblages, les interrupteurs et les composants passifs (résistances, capacités).

Le coût de revient de ce démonstrateur revient à moins de 400 euros TTC, ce qui rentre parfaitement dans le cahier des charges.

# 4 Evolutions informatiques : IHM en C++

L'interface de développement *Processing* a montré ses limites. Il n'y a pas de contrôles pour interagir avec l'utilisateur. La gestion de la mémoire entre le traitement des données et l'affichage présente des performances limitées qui « bloquent » le programme.
Nous avons donc décidé de migrer l'application en C++, avec Visual Studio 2008 pour pouvoir utiliser les classes MFC (Microsoft Fondation Class) pour dessiner des boutons, des zones de textes et de paramétrages dans une IHM (Interface Homme Machine) conviviale et intuitive.
La fonction critique de réception des données par Wi-Fi doit être fluide. Elle sera réalisée grâce à une socket asynchrone UDP par gestion d'évènements.
L'avantage, c'est que c'est un environnement de développement que je maîtrise de par mon expérience professionnelle et qui présente une excellente rapidité d'exécution du code. Il est également utilisé par l'ingénieur informatique du laboratoire, ce qui simplifiera la maintenabilité du code.
L'inconvénient est que c'est un outil de développement payant (70 euros) et que les applications ne sont exécutables qu'avec des systèmes d'exploitation Windows. Du coup, la portabilité n'est pas assurée sur des systèmes Macintosh ou Linux, limitant notre réponse au cahier des charges.
Mais au vu des algorithmes détaillés ci-dessous, n'importe quel informaticien pourra porter ce code dans un autre environnement de développement qui puisse satisfaire ce critère, comme par exemple Qt, une bibliothèque de classe C++ sous licence GNU.

Le document « *Description programmes .pdf* » fournit dans l'Annexe B. « Documents » contient toutes les informations pour prendre la relève.

Les sources des programmes décrits dans ce chapitre sont disponibles en Annexes :

- ⇒ Annexe C « Sources programmes Maple », pour la partie embarquée
- ⇒ Annexe D « Sources programmes Ordinateur », pour la partie IHM

## 4.1 Architecture

Voici une brève description des objets utilisés et des relations entre eux.

| Objet | Description |
|---|---|
| Socket Wi-Fi (UDP) | Reçoit les données et les infos du *WiFly*<br>Envoie les commandes au *WiFly* |
| Fichier Log | Fichier contenant infos de debug |
| Fichier Acquisition | Fichier contenant les échantillons, en colonnes pour chaque capteur.<br>Pourra être exploité ultérieurement par une application en C++ ou Matlab |
| Échantillons | Données numériques des échantillons |
| Décryptage des données d'acquisition | Décrypte et converti les octets reçus en valeur numériques d'échantillons<br>Décrypte le *TimeStamp* et vérifie |
| Décryptage des chaînes de caractères | Décrypte les données autres que celles d'acquisition (principalement 'Start/Stop' et les paramètres du tableau de la page suivante) |
| Affichage graphique *(Optionnel)* | Affiche sous forme de courbe temporelle les échantillons |

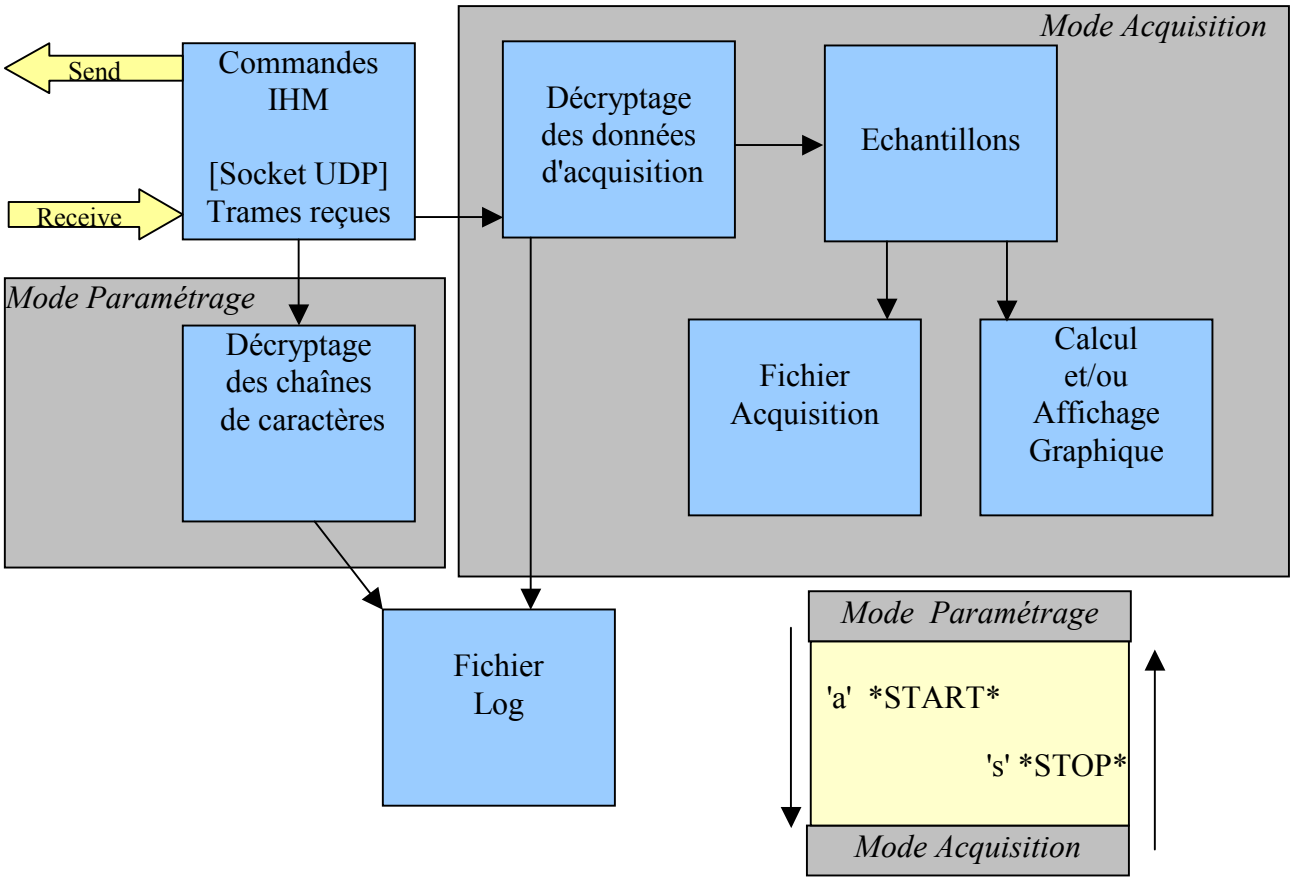



*Figure 25 : Architecture du programme en C++*

## 4.2 Description du programme côté microcontrôleur

On conserve ici l'environnement de développement dédié ( 2.5.1 « Maple IDE »).
Le fichier « *sketch_Wifly_Select_Sensor_ContinuousStreaming.pde* » que l'on peut extraire de l'Annexe C « Sources programmes Maple » est le listing principal pour générer le code embarqué sur le microcontrôleur STM32.

Il contient :

- l'initialisation des fonctions électroniques
- la gestion des interruptions et des *timers*
- le paramétrage des fonctions électroniques (automate)
- l'acquisition des données analogiques et numériques
- l'envoi des données par Wi-Fi

Le répertoire « Libraries » permet d'encapsuler les accès aux composants périphériques :

- ADXL345SPI : accès à l'ADXL en mode SPI (port #2)
- WiFly : accès au Wi-Fi en mode SPI (port #1)

### 4.2.1 Mode paramétrage

Un automate décrypte chaque commande, composée d'une lettre '*command*' et de 0 à 3 chiffres '*param*', pour effectuer les réglages électroniques (voir la description dans 2.9.2 « Paramètres et commandes de l'automate »).
Ces commandes reflètent celles utilisées dans l'interface en C++ (voir 4.3 « Paramétrage : fonctions de l'automate »).

```
void ProcessCommand() = Switch case (command) :
```

| Cmd | Param assign to | Fonction appelée | Description |
|---|---|---|---|
| 'H' | | DoSendHello(); | Send *HELLO* (just to check connection) |
| 'h' | | DoDisplayHelp(); | Help |
| 't' | lAcquisitionDuration | DoSelectAcquisitionMode(); | Continous or start to time mode |
| 'd' | bDoDisplayGraph | DoIsGraph(); | Enable/Disable Graphic Display |
| 'n' | nMaxSample | DoChangeMaxSample(); | Max number of samples |
| 'f' | nbMicroseconds = NB_MICRO_IN_ONE_SECOND/param | DoChangeMicroPeriod(); | Analog Sampling Frequency |
| 'm' | NbChannels | DoSelectAnalogChannels(); | Nb channels (0 to 6) |
| 'c' | nHoldRateIndex | DoChangeADCHoldTime(); | Modify ADC hold time |
| 'r' | nAccelRange | DoSetAccelRange(); | Accel Range |
| 'b' | nAccelBandwidth | DoSetAccelBandwidth(); | Accel Bandwidth |
| 'x' | bDoAccelAcq | DoSelectAccel(); | Accel ON OFF |
| 'i' | | DoGetAccelID(); | Get ADXL ID |
| 'g' | | DisplayParameters(); | Get All Current Parameters |

*Tableau 10 : Automate `ProcessCommand()`*

Voici le code des principales fonctions qui modifient les propriétés matérielles :

```
DoChangeMicroPeriod()
      //Change Timer period (Sampling Frequency = 1/nbMicroseconds)
      timerAcq.setPeriod(nbMicroseconds);
      timerAcq.refresh();

DoSelectAnalogChannels()
      //Activate channels as Input Analog
      // nAnalogInputPin[MAX_CHANNELS] = {15, 16, 17, 18, 19, 20};
      for (int i = 0; i < NbChannels; i++)
      {
            pinMode(nAnalogInputPin[i], INPUT_ANALOG);
      }

DoChangeADCHoldTime()
      //Change internal sample and hold analogical capacity :
      // ADC_SMPR[8] =
      // ADC_SMPR_1_5, ADC_SMPR_7_5, ADC_SMPR_13_5,
      // ADC_SMPR_28_5,ADC_SMPR_41_5,ADC_SMPR_55_5,
      // ADC_SMPR_71_5, ADC_SMPR_239_5
      eADCSampleRate = ADC_SMPR[nHoldRateIndex]
      adc_set_sample_rate(ADC1, eADCSampleRate);

DoSetAccelRange()
      //Set Range
      Adxl345Serial.SetADXLRange(nAccelRange);

DoSetAccelBandwidth()
      //Set Bandwitdth
      Adxl345Serial.SetADXLBandwidth(nAccelBandwidth);

DoGetAccelID()
      //Get Accel ID (must be 0xE5)
      Adxl345Serial.GetADXLDeviceID();
```

Ces dernières fonctions '`Adxl345Serial`' appellent des méthodes dans la librairie ADXL345SPI qui accèdent aux registres du capteur numérique.

### 4.2.2 Mode acquisition

Pour passer en mode acquisition, il suffit d'envoyer la commande '`a`'.
Pour sortir du mode acquisition, il faut attendre la fin de la durée choisie si elle est différente de zéro ou envoyer la commande '`s`'.

Pour optimiser la vitesse d'exécution, l'acquisition est contrôlée par des interruptions :

- un *Timer* appelé périodiquement pour échantillonner les voies analogiques et envoyer les données par Wi-Fi
- un *IRQ* pour récupérer les données du capteur numérique ADXL345 quand de nouvelles valeurs sont disponibles
- un *IRQ* pour réceptionner une commande reçue par Wi-Fi

Ces interruptions sont associées à des fonctions appelées par déclenchement.

## Interruption par Timer

Déclaration :
```
// Use timer 1 for analogical microphone acquisition
HardwareTimer timerAcq(1);
```

Initialisation :
```
// Pause the timer while we're configuring it
timerAcq.pause();
// Have the timer repeat every nbMicroseconds
timerAcq.setPeriod(nbMicroseconds);
// Set up an interrupt on channel 1
timerAcq.setChannel1Mode(TIMER_OUTPUT_COMPARE);
```

Association :
```
timerAcq.attachInterrupt(TIMER_CH1, DoSampleAnalogData);
timerAcq.refresh();
```

`DoSampleAnalogData()`

- Désactive les interruptions
- Lit les valeurs analogiques échantillonnées par le CAN
- Vérifie la place requise pour envoyer les données (en fonction du nombre de voies et du *TimeStamp*)
- Transmet les octets en continu par le Wi-Fi (voir 4.4 « Acquisition : description du format de transmission des données » )
  - Les échantillons analogiques, s'il y en a
  - Les échantillons numériques, s'il y en a
  - Le *TimeStamp* (tous les N échantillons)
- Incrémente le compteur d'échantillon du bloc (de 1 à N)
- Incrémente le compteur d'échantillons total
- Ré-active les interruptions

## Interruption par IRQ : ADXL345

Déclaration :
```
//Indicate Data_Ready at ADXL bandwidth rate
const int ADXL_INTERRUPT_PIN = D35;
```

Initialisation :
```
//interrupt on pin35 (= ADXL INT1)
pinMode(ADXL_INTERRUPT, INPUT);
detachInterrupt(ADXL_INTERRUPT);

//Set DATA_READY on INT1
Adxl345Serial.writeRegister(INT_MAP, 0x7F);
```

Association :
```
attachInterrupt(ADXL_INTERRUPT, DoReadAxisOnInterrupt, RISING);
```

`DoReadAxisOnInterrupt()`

- Met à jour le tableau des échantillons numériques en lisant les 6 registres de données de l'ADXL345 (2 octets × 3 axes)
- Si aucune voie analogique n'est sélectionnée, appelle `DoSampleAnalogData()` pour envoyer les échantillons numériques seuls

**Interruption par IRQ : RX-WIFI**

Déclaration :
```
//IRQ on pin 7 to get the stop command during acquisition
const int RX_INTERRUPT = 7;
```

Initialisation :
```
//interrupt on pin7 (= RX on IRQ_RHR)
pinMode(RX_INTERRUPT, INPUT_PULLUP);
detachInterrupt(RX_INTERRUPT);
```

Association :
```
attachInterrupt(RX_INTERRUPT, WaitForStop, FALLING);
```

`WaitForStop()`

- Si un caractère est reçu sur la liaison Wi-Fi, met le *Timer* en pause
- Lit le caractère
  - Si c'est '`s`' : dé-associe les interruptions et repasse en mode paramétrage
  - Si ce n'est pas '`s`' : le *Timer* reprend et on continu l'acquisition

**Déroulement de l'acquisition**

Lorsque la commande '`a`' est envoyée, on procède à l'acquisition :

- Affiche tous les paramètres avant de commencer (on envoie tous les réglages à l'ordinateur par Wi-Fi en format texte)
- Calcule le nombre d'octets pour l'UART selon les voies sélectionnées (2 octets par voie analogique + 6 octets pour le capteur numérique)
- Envoie ***START*** à l'ordinateur pour dire qu'on enverra désormais les données en octets et non plus en texte
- Initialise les compteurs
- Associe l'interruption de l'accéléromètre numérique ADXL345, s'il est sélectionné
- Démarre le chronomètre
- Associe l'interruption de la réception d'un caractère sur le Wi-Fi
- Démarre le *Timer* si au moins une voie analogique est sélectionnée
- Initialise le *TimeStamp*

On laisse faire jusqu'à ce qu'on reçoive la commande '`s`' et/ou on chronomètre pour arrêter l'acquisition à la fin de la durée sélectionnée.

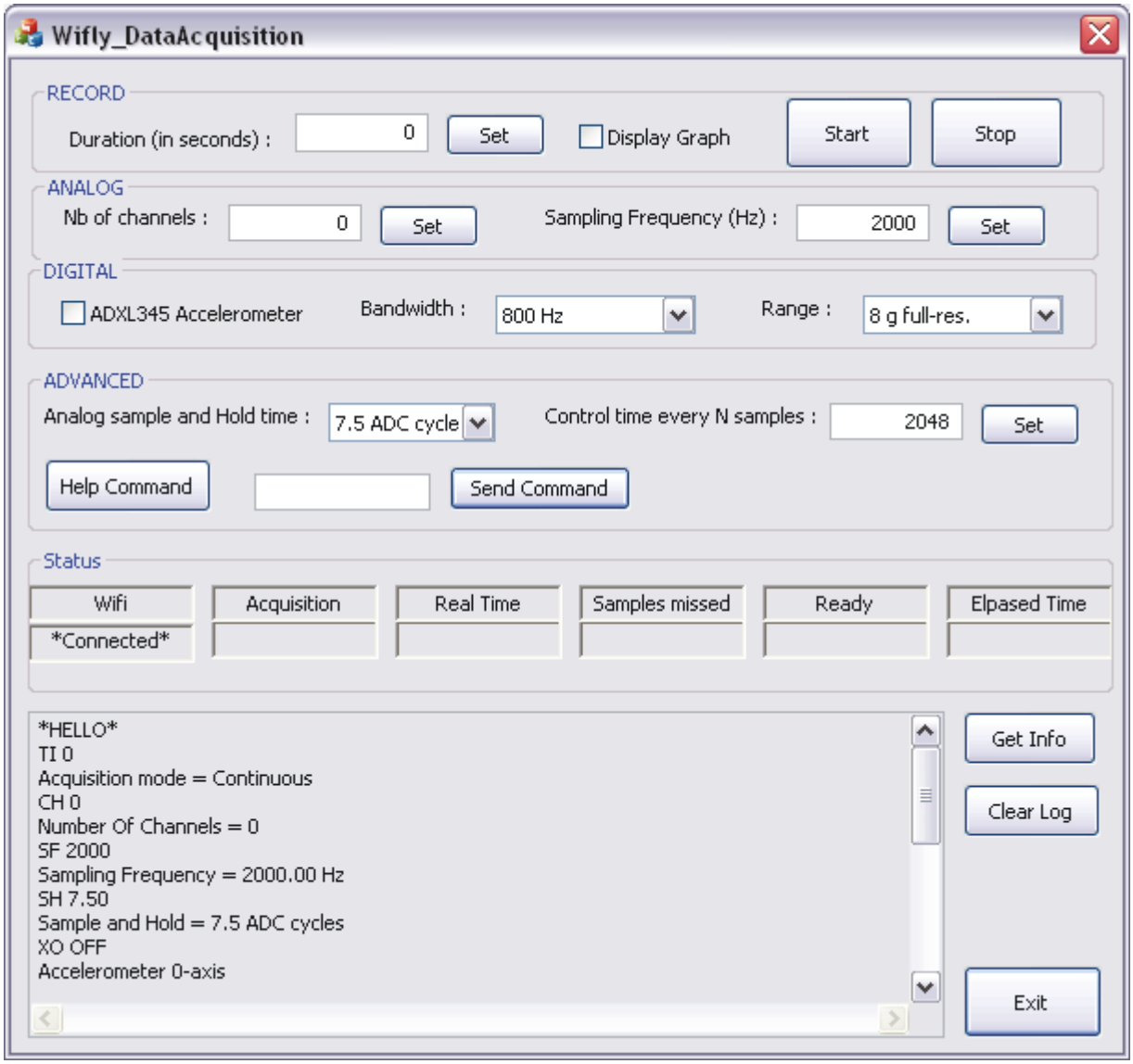
A l'arrêt de l'acquisition :

- Détache les interruptions
- Arrête le *Timer*
- Envoie ***STOP*** à l'ordinateur pour dire qu'on repasse en mode texte
- Envoie le nombre total d'échantillons qui sont censés avoir été transmis pour que le programme côté ordinateur puisse s'assurer que tout a été reçu

Le programme se remet en mode « paramétrage » pour attendre une nouvelle commande à traiter par l'automate.

### 4.3 Paramétrage : fonctions de l'automate

Cette IHM remplace avantageusement la saisie des commandes par le clavier en langage Processing (voir 2.9.2 « Paramètres et commandes de l'automate »).




Cette interface permet :

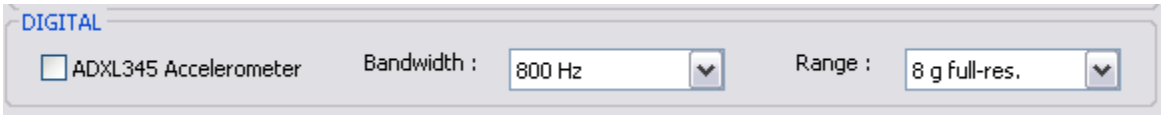
- de paramétrer les capteurs et les données d'enregistrement
- de lancer/arrêter l'acquisition
- de visualiser le signal
- de contrôler le bon déroulement des opérations
- d'enregistrer les échantillons dans un fichier

Le projet Visual Studio « *Wifly_DataAcquisition* » que l'on peut trouver de l'Annexe D « Sources programmes Ordinateur » contient les sources complètes pour générer l'exécutable côté ordinateur.

### 4.3.1 Choix des paramètres


➔ Sélectionner le nombre de capteurs analogiques connectés (de 1 à 6)
➔ Sélectionner la fréquence d'échantillonnage, en tenant compte des limitations :

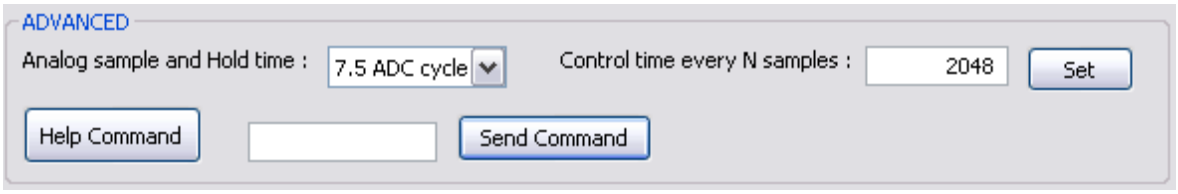
| Nombre de capteurs analogiques | Fréquence d'échantillonnage Max (en kHz) |
|---|---|
| 1 | 23 |
| 2 | 12 |
| 3 | 8 |
| 4 | 6 |
| 5 | 4,8 |
| 6 | 4 |




➔ Cocher la case pour utiliser l'accéléromètre triaxes numérique
➔ Sélectionner sa bande passante
➔ Sélectionner sa gamme

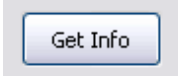
Ce capteur compte pour 3 voies, en tenir compte pour régler la fréquence d'échantillonnage des capteurs analogiques dans le tableau ci-avant.


➔ Indiquer la durée d'enregistrement de l'acquisition (laisser 0 pour un enregistrement continu)
➔ Cocher la case pour afficher une fenêtre de surveillance des signaux *(uniquement la première voie analogique pour le moment)*




Ces réglages sont des réglages avancés, à laisser de préférence par défaut.

➔ Choisir la durée de l'échantillonneur/bloqueur du microcontrôleur
➔ Choisir le nombre d'échantillons utilisés pour le contrôle du temps réel (le programme vérifie tous les N échantillons que la durée d'acquisition du bloc correspond à la fréquence d'échantillonnage sélectionnée)
➔ Possibilité de régler manuellement les paramètres en ligne de commande




permet de récupérer tous les paramètres de la maquette

### 4.3.2 Exécution de l'acquisition

➔ Cliquer sur 

pour lancer la mesure

➔ Cliquer sur 

pour arrêter la mesure

➔ L'enregistrement s'arrête automatiquement au bout de la durée (si différente de 0)

La fenêtre de log affiche brièvement des informations pour récupérer les paramètres et détecter la chaîne de caractère ***START*** envoyé par la maquette pour indiquer que la transmission des échantillons commence.

Puis, durant l'acquisition :

- l'interface affiche des informations de statuts
- la fenêtre de log indique si des échantillons n'ont pas été décryptés correctement
- les échantillons sont enregistrés dans un fichier texte (voir 4.6 « Enregistrement et traitement des données »)
- si la case « Display Graph » a été cochée, les échantillons sont affichés en temps réel *(limité à la première voie sur 2000 points pour l'instant)*

L'acquisition s'arrête lorsque la chaîne de caractère ***STOP*** envoyé par la maquette :

- l'interface affiche un résumé pour vérifier que tout c'est bien passé
- le fichier texte est sauvegardé et contient toutes les informations utiles pour une exploitation ultérieure (par Matlab par exemple)

Pour faciliter le debug, les informations de la fenêtre log sont retranscrites dans un fichier texte. Ce fichier est sauvegardé à la fermeture de l'application et contient donc les différents changements de réglages que l'utilisateur a fait, les échantillons qui ont été manqués pendant une acquisition, les problèmes de synchronisation...

Pendant l'acquisition, la **barre de statuts** indique :

| | | |
|---|---|---|
| Acquisition = | *Measuring* | La mesure est en cours |
| Real Time = | *OK*<br>*X* blocks missed | Tous les blocs d'échantillons sont correctement récupérés.<br>*X* blocs d'échantillons n'ont pas pu être récupérés. |
| Samples missed = | 0<br>> 0 | On a pu décrypter chaque échantillon.<br>On n'a pas pu décrypter le nombre indiqué d'échantillons |
| Ready = | *Not Ready* | L'acquisition étant en cours, on ne peut pas modifier les réglages. |
| Elapsed Time = | | Temps écoulé depuis le début de l'acquisition (en secondes) |

## 4.4 Acquisition : description du format de transmission des données

Notre application est critique au point de vue débit. Elle requiert un contrôle du temps réel pour s'assurer que l'on reçoit bien toutes les données en continu.
Pour cela j'introduis un algorithme qui va contrôler périodiquement le débit des données, à l'aide d'une étiquette temporelle appelé « *TimeStamp* ».

Les données seront transmises dans cet ordre :

- N échantillons
- *TimeStamp*

Pour contrôler que l'on est bien temps réel :

- Tous les N échantillons, le système embarqué envoie la durée écoulée T en ms
- Côté ordinateur, on calcule la fréquence d'échantillonnage correspondante $FE(tx)=\frac{N\times 1e_6}{T}$
- Et on compare *FE(tx)* à la fréquence d'échantillonnage choisie par l'utilisateur
- A la fin de l'acquisition, on compare le nombre d'échantillons envoyés par le système embarqué avec celui traité par l'ordinateur

Le décryptage des données d'acquisition se compose de :

- N échantillons de M voies sur 2 octets chacun
- Suivi d'un intervalle de contrôle du temps écoulé entre N échantillons, sur 6 octets <ABCD>, appelé *TimeStamp*.

Exemple (N = 1024 & M = 2) :

| **N / M** | **Ch_1** | **Ch_2** |
|---|---|---|
| **1** | Val_1 | Val_2 |
| **2** | Val_3 | Val_4 |
| **3** | Val_5 | Val_6 |
| **...** | ... | ... |
| **1024** | Val_2047 | Val_2048 |
| **Control** | TimeStamp | |

On reçoit tout les octets à la suite :
[Val_1][Val_2][Val_3] ... [Val_2048] [TimeStamp]

Chaque valeur 'Val' est codée sur 16 bits.
Le convertisseur étant 12 bits, il reste 4 bits inutilisés.
Pour effectuer un minimum de contrôle, on transmet les 12 bits en 2 octets (MSB et LSB) contenant 6 bits d'information + 1 bit de contrôle qui indique si c'est le MSB (1) ou le LSB (0),

Ainsi :

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| D12 | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 | D3 | D2 | D1 |

Devient :

| **MSB** | | | | | | | | **LSB** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***1*** | *X* | D12 | D11 | D10 | D9 | D8 | D7 | ***0*** | *X* | D6 | D5 | D4 | D3 | D2 | D1 |

De cette façon, on a un contrôle certes basique mais qui permet au moins de vérifier que l'on reçoit 2 octets par valeur.

Le *TimeStamp* est décomposé comme suit :

| | | | | | |
|---|---|---|---|---|---|
| < | A | B | C | D | > |

Que l'on reconstitue pour avoir une valeur *Long* de 32 bits.
On peut contrôler que l'on est toujours temps réel en vérifiant la cohérence entre cette valeur de *TimeStamp* et le nombre de d'octets reçus entre 2 *TimeStamps*.
Le *TimeStamp* est en fait l'intervalle de temps écoulé 'L' en millisecondes.
On calcule *[L / N]* et on vérifie que ça correspond à la fréquence d'échantillonnage sélectionnée (avec une marge d'erreur à paramétrer, par défaut 5%).

Les erreurs trouvées lors de la réception pourront être analysées lors d'une relecture du fichier des résultats avant son exploitation finale.
Il manque un algorithme pour balayer et réparer les « trous ». Si quelques octets manquent, on peut les remplacer par interpolation entre les valeurs encadrantes. S'il en manque trop, il est plus prudent de mettre des zéros et d'indiquer que ce morceau est inexploitable.

## 4.5 Liaison Wi-Fi

En premier lieu, j'ai implémenté une communication dans le mode TCP, avec une excellente sécurité quant à la transmission des données. En effet, le mode TCP s'assure que les paquets ont bien été transmis par un acquittement et les renvoie le cas échéant. Mais le revers de la médaille est qu'il en limite le débit.
On préfèrera le mode UDP qui ralentit moins la connexion, sans rétrocontrôle, pour communiquer un flux continu avec un débit théoriquement plus élevé.

On utilise un réseau « Ad-Hoc », plus facile à mettre en oeuvre qu'une liaison type « Access Point » et ne nécessitant pas de mot de passe. La gestion par mot de passe est indispensable pour des raisons de sécurité, lorsqu'on ne souhaite pas qu'une communication soit interceptée.
Ici, le niveau critique de nos données est très faible puisqu'il s'agit de signaux temporels et non pas de texte en clair à crypter.
De plus, contrairement à la plupart des applications Wi-Fi qui transmettent des données en format HTML par exemple, la nôtre code directement des échantillons les uns à la suite des autres dans un format binaire. Même si la communication est interceptée, il faut pouvoir reconstruire les signaux.

Ainsi la méthode décrite ci-avant (voir 4.4 « Acquisition : description du format de transmission des données ») surmonte ces deux écueils par :

- un contrôle à posteriori du temps réel
- le cryptage des données sur 2 octets, au lieu du code ASCII

Le paramétrage, tant du côté embarqué que du côté ordinateur ne nécessite alors que quelques fonctions pour que l'un et l'autre se reconnaissent :

| Paramètres Wi-Fi (mode UDP) | Description |
|---|---|
| Wifly-GSX-32 | Nom attribué au réseau Ad-Hoc |
| IP Computer = 169.254.1.11 | Adresse IP de l'ordinateur (paramètre de la clé USB Wi-Fi) |
| Port Computer = 2048 | Port de l'ordinateur associé à la liaison Wi-Fi |
| IP WiFly = 169.254.1.W | Adresse IP du module capteur WiFly<br>W = numéro de capteur<br>(pour l'instant un seul : W = 1, IP = 169.254.1.1) |
| Port Computer = 8001 | Port du module capteur WiFly |

En théorie, on est capable de récupérer l'information permettant de savoir quelle IP et/ou port a écrit et donc de différencier les données provenant de plusieurs capteurs.
Mais je n'ai pas eu l'occasion de vérifier ce mode de fonctionnement car je n'ai pu réaliser qu'une seule maquette à l'issue de mon stage. Toutefois, j'ai consigné ce qui pouvait être envisagé pour des évolutions futures (voir 5.4 « Synchronisation de plusieurs modules »).

En choisissant de migrer l'application en C++, j'ai pu bénéficier d'un composant « Socket UDP » natif pour recevoir les données par gestion d'évènements. Cela évite de scruter si on reçoit des données. C'est l'arrivée de celles-ci qui déclenche par interruption leur propre traitement.

La vitesse de transmission des données est principalement conditionnée par le goulot d'étranglement dû à la passerelle SPI/UART entre le microcontrôleur et le Wi-Fi (voir 2.8 « Mise en œuvre de la liaison Wi-Fi »).
De plus, la vitesse instantanée a du être réglée à 460800 au lieu de 921600 qui est censé être le maximum autorisé (sinon on ne sort pas du mode commande sur le composant RN131C, c'est une limitation du un a bug dans le *firmware*).
On a fait le choix de régler la vitesse instantanée à chaque démarrage après l'avoir initialisée à 9600 par défaut afin de ne pas perdre la communication.
En effet, si on éteint la platine sans savoir à quelle vitesse elle est réglée, le microcontrôleur ne saura plus comment communiquer avec le composant Wi-Fi.
Cette opération est délicate car on doit régler à la fois la vitesse du Wi-Fi et celle de la passerelle pour qu'elles coïncident.

## 4.6 Enregistrement et traitement des données

Le fichier texte enregistré pourra être post-traité par un tableur ou Matlab.
Matlab est un outil de calcul très puissant utilisé par les chercheurs.
Un des objectifs de ce démonstrateur était de pouvoir appliquer les algorithmes de traitement du signal aux signaux. Cet objectif est atteint grâce à ce formalisme compatible avec Matlab.

Les données récupérées précédemment sont enregistrées dans un **fichier texte** dont le nom est :

Acq_*C*_channels_*F*_Hz_Accel_*R*_g_*B*_Hz_*H_M_S*.txt

*C* : nombre de voies analogiques
*F* : fréquence d'échantillonnage des voies analogiques
*R* : gamme de l'accéléromètre numérique
*B* : bande passante de l'accéléromètre numérique
*H_M_S* : heure de début du fichier (heure_minute_seconde)

**L'en-tête du fichier** contient les informations suivantes:

| | |
|---|---|
| Start date = 10 h 59 m 25 s | La date de début de l'enregistrement |
| Acquisition duration = Continous | Sa durée en secondes ou continu |
| Channels = 2 SF = 20000 | Le nombre de voies analogiques et la fréquence d'échantillonnage (si on en a sélectionné) |
| Accelerometer Range = 8 BW = 800 | La gamme et la bande passante de l'accéléromètre numérique (s'il est sélectionné) |

Puis au fur et à mesure de la réception des échantillons, ceux-ci sont sauvegardés dans le fichier texte, séparé par des tabulations pour être facilement exploités dans Excel ou Matlab par la suite.

- *Sample* : le numéro de l'échantillon. Il suffira de le multiplier par l'inverse de la fréquence d'échantillonnage pour obtenir l'indice temporel correspondant.
- Channel 1 à Channel 6 : les valeurs numérisées des voies analogiques, s'il y en a (entre 0 et 4095 car on récupère la valeur brute d'un CAN 12 bits).
- X-axis, Y-axis, Z-Axis : si l'accéléromètre numérique a été sélectionné, les valeurs numérisés des axes X, Y et Z (entre -2048 et + 2048, correspondant à +/- la gamme sélectionnée).

| Sample | Channel 1 | Channel 2 | X-axis | Y-axis | Z-axis |
|---|---|---|---|---|---|
| 1 | 2039 | 2007 | 268 | -56 | -12 |
| 2 | 2036 | 2041 | 270 | -54 | -34 |
| 3 | 2040 | 2026 | 256 | -52 | -36 |
| 4 | 2045 | 2007 | 262 | -48 | -28 |
| ... | ... | ... | ... | ... | ... |

*Traitement des erreurs de réception :*
J'ai prévu un mécanisme de traitement des erreurs afin de ne pas arrêter l'acquisition si la réception s'est temporairement mal passée.
Les échantillons qui n'ont pas pu être reconstitués sont remplacés par *** dans le fichier texte (voir 4.4 « Acquisition : description du format de transmission des données » ).

On contrôle la réception des échantillons tous les N échantillons en vérifiant qu'on reçoit bien une date de contrôle de la forme <*ABCD*> (4 octets entourés entre flèches).
Si ce n'est pas le cas, on écrit "Error Not Time Interval at sample *X*" dans le fichier, avec *X* comme étant le dernier échantillon connu récupéré.
En général, cette erreur est suivie de "Error getting *N* samples : *M* Missing : *N-M*" qui indique le nombre de « trous » *N-M* estimés.

Si les octets récupérés ne semblent correspondre ni à une date de contrôle, ni à un échantillon, ils sont indiqués dans le fichier par "Error *B* bytes not processed after sample *X*", *B* étant le nombre d'octets non interprétés et *X* étant le dernier échantillon connu récupéré.
On s'assure également que le nombre total d'échantillons reçus par l'ordinateur est identique à celui envoyé à la fin de l'acquisition par la maquette.

A la fin de l'analyse, le **fichier texte** est complété (avant d'être sauvegardé et fermé) avec :

| | |
|---|---|
| Elapsed Time = 13.594 s compared to number of samples over sampling frequency = 12,666 s<br>Elapsed Time between Start and First Sample = 0,219 s<br>Start Elapsed Time = 13.375 s compared to number of samples = 12,666 s (difference = 0,709 s) | (Voir la description ci dessus) |
| Stop date = 10 h 59 m 38 s | La date de fin de l'acquisition |
| Total samples acquired on PC = 253328<br>Total samples sent by Maple = 253328 | (Voir la description ci dessus) |
| Byte error conformity = 0<br>Blocks Missing = 0<br>Error bytes not processed = 0 | Les éventuels erreurs de réception des échantillons |

## 4.7 Visualisation graphique

Le Control Chart trouvé sur le web [77] sous licence ouverte m'a évité de réinventer la roue pour l'affichage. Celle-ci se présente sous forme d'une grille pour visualiser le signal temporel en continu.
L'objet est très facile à manipuler, puisqu'il suffit de dimensionner la grille avec les méthodes prévues à cet effet puis d'ajouter les points qui seront automatiquement reliés entre eux.
Mais je n'ai pas eu le temps de dériver la classe pour plusieurs voies.
Par conséquent, seule la voie 1 est visualisable graphiquement à la fin de mon stage.
La figure ci-dessus montre une copie d'écran de ce que l'on peut observer sur l'ordinateur.

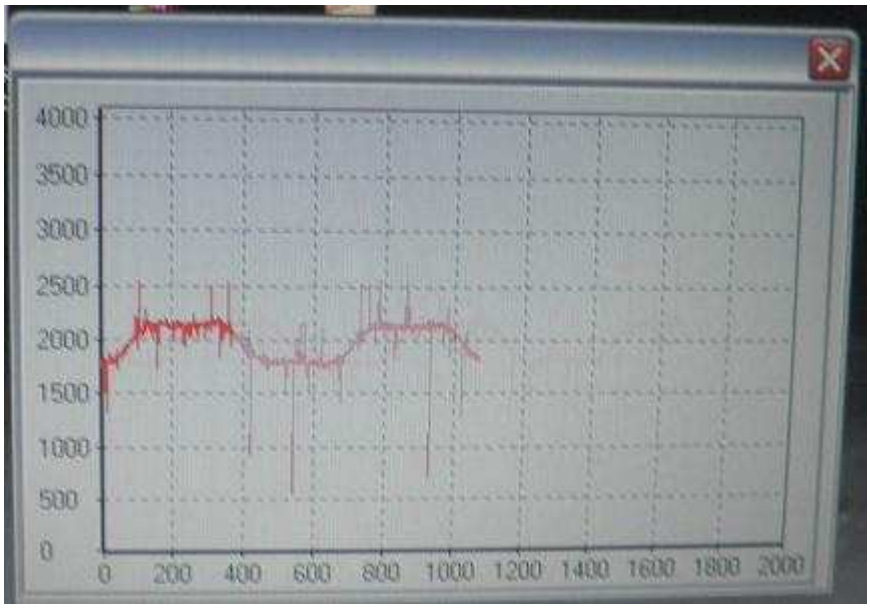



*Figure 26 : Exemple de visualisation graphique*

## 4.8 Résultats & Performances

### 4.8.1 Chronométrage

On instrumente le code pour mesurer les différents temps d'exécution et le bon fonctionnement des interruptions.

| Fonction | Interruption |
|---|---|
| CAN | *Timer* (soit l'inverse de la fréquence d'échantillonnage FE) |
| ADXL345 | IRQ sur la présence de nouvelles données |
| Wi-Fi | IRQ sur la réception d'un caractère |

Le composant *WiFly* utilisé est en Version 2.23, réglé à une vitesse de 460800 bauds.

**Code de l'interruption *Timer* :**
Cette fonction est appelée périodiquement (la période est l'inverse de la fréquence d'échantillonnage) :

```
void DoSampleAnalogData(void)
{
   //*** FILL IN ANALOG VALUES FROM DAC ***//
   //*** WIFI START SENDING ***//
   //*** SEND SENSORS VALUES ***//
   //*** COMPUTE AND SEND DATE STAMP ***//
   //*** WIFI END SENDING ***//
}
```

On effectue les mesures à l'oscilloscope en utilisant une sortie numérique que l'on fait basculer entre les niveaux haut et bas.

/*** FILL IN ANALOG VALUES FROM DAC ***//

A fréquence d'échantillonnage fixe (2 kHz) on chronomètre le temps (en µs) nécessaire pour faire l'acquisition de 1024 échantillons en faisant varier :

- Le temps de conversion du microcontrôleur (SH pour *Sample&Hold*) de 1,5 à 239,5 µs
- Le nombre de voies de 1 à 6

On en déduit la période pour initialiser le convertisseur et par voie.

| Nb / SH | 1,5 | 7,5 | 13,5 | 28,5 | 41,5 | 55,5 | 71,5 | 239,5 |
|---|---|---|---|---|---|---|---|---|
| **1** | 3,6 | 3,9 | 4,4 | 5,7 | 6,8 | 7,8 | 9,6 | 24 |
| **2** | 6,4 | 7,2 | 8,3 | 10,8 | 13 | 15,2 | 18,4 | 46 |
| **3** | 9,3 | 10,5 | 12 | 15,8 | 19,2 | 22,6 | 26,8 | 69 |
| **4** | 12,2 | 13,9 | 15,8 | 20,8 | 25,4 | 29,8 | 35,6 | 91 |
| **5** | 15,1 | 17,2 | 19,6 | 25,8 | 31,6 | 37,2 | 44,4 | 114 |
| **6** | 18,1 | 20,5 | 23,2 | 30,8 | 38 | 44,6 | 53,2 | 137 |
| **TinitSH** | 0,8 | 0,6 | 0,5 | 0,6 | 0,6 | 0,4 | 0,8 | 2 |
| **TchSH** | 2,9 | 3,3 | 3,8 | 5 | 6,2 | 7,4 | 8,8 | 23 |

*Tableau 11 : Période du CAN en fonction du nombre de voies et du SH*

//*** SEND SENSORS VALUES ***//

A fréquence d'échantillonnage fixe (1 kHz) on chronomètre le temps (en µs) nécessaire pour faire envoyer 1024 échantillons en faisant varier le nombre de voies.
On en déduit la période pour initialiser la transmission et par voie.

| Nb voies | T |
|---|---|
| **1** | 6,52 |
| **2** | 11,4 |
| **3** | 16,3 |
| **4** | 21,2 |
| **5** | 26 |
| **6** | 31 |
| **TinitWifi** | 1,64 |
| **TchWifi** | 4,9 |

*Tableau 12 : Période du Wi-Fi en fonction du nombre de voies*

L'envoi des données pour les 3 axes de l'accéléromètre ADXL345 est de 16,4 µs (à peu près identique à 3 voies analogiques).

//*** WIFI START SENDING ***//
//*** COMPUTE AND SEND DATE STAMP ***//
//*** WIFI END SENDING ***//

Le temps passé dans ces fonctions est fixe et indépendant des réglages du CAN (nombre de voies, SH, FE).

WIFI START SENDING = 10,5 µs
COMPUTE AND SEND DATE STAMP = 15,2 µs
WIFI END SENDING = 1,18 µs

Calculs

On en déduit le temps incompressible dû au protocole de transmission :

Tprotocol = 10,5 + 15,2 + 1,2 = 26,9 µs soit FE = 37 kHz

On calcule la durée maximum en fonction des paramètres du CAN :

T = Tprotocol + TinitSH + (N×TchSH) + TinitWifi + N×TchWifi
Et on trouve la fréquence d'échantillonnage FE = 1/T correspondante

| Nb / SH | 1,5 | 7,5 | 13,5 | 28,5 | 41,5 | 55,5 | 71,5 | 239,5 |
|---|---|---|---|---|---|---|---|---|
| **1** | 26,6 | 26,5 | 26,2 | 25,3 | 24,6 | 24 | 23 | 17 |
| **2** | 22 | 21,7 | 21,3 | 20,2 | 19,3 | 18,5 | 17,5 | 11,5 |
| **3** | 18,8 | 18,5 | 18 | 16,9 | 15,9 | 15,1 | 14,1 | 8,7 |
| **4** | 16,4 | 16 | 15,6 | 14,5 | 13,5 | 12,7 | 11,8 | 7 |
| **5** | 14,5 | 14,2 | 13,7 | 12,6 | 11,8 | 11 | 10,2 | 5,9 |
| **6** | 13,1 | 12,7 | 12,2 | 11,2 | 10,4 | 9,7 | 8,9 | 5 |

*Tableau 13 : Fréquence d'échantillonnage maximum théorique (en kHz)*

En fixant SH à 7,5 µs, on vérifie en pratique selon le nombre de voies :

| Nb Voies | FE Théorique (en kHz) | FE Pratique (en kHz) |
|---|---|---|
| **1** | 26,5 | 24 |
| **2** | 21,7 | 11,1 |
| **3** | 18,5 | 8 |
| **4** | 16 | 5 |
| **5** | 14,2 | 4 |
| **6** | 12,7 | 4 |

*Tableau 14 : Comparaison des fréquences d'échantillonnage théorique et pratique*

En pratique, on constate que l'on perd beaucoup de la capacité théorique et que le phénomène s'accentue avec le nombre de voies !
On instrumente plus finement le code pour comprendre ce qui se passe...
On remarque que le goulot d'étranglement qui conditionne la fréquence d'échantillonnage maximum est **TinitWifi** qui démarre l'écriture à la passerelle Wi-Fi.
Dans cette fonction, on vérifie qu'on a bien assez d'espace pour transmettre les échantillons (soit 2 octets par voie, plus éventuellement 6 octets de *TimeStamp* tous les *N* échantillons de contrôle).
Bizarrement, la durée de cette fonction est « en accordéon » lorsque l'on se rapproche de la fréquence d'échantillonnage maximum alors qu'elle est constante en-dessous (le décrochage est flagrant).

| **Nb voies -FE** | 1 - 23 kHz | 2 - 12 kHz | 3 - 8 kHz | 4 - 6 kHz | 5 - 4,8 kHz | 6 - 4 kHz |
|---|---|---|---|---|---|---|
| **TinitWifi (µs)** | 10,5 à 30 | 10,6 à 63 | 10,6 à 102 | 10,8 à 108 | 10,8 à 174 | 10,8 à 186 ! |

Peut-être faut-il chercher du côté de la gestion des trames en dimensionnant différemment le FIFO du SC16IS750 en regroupant les octets à envoyer ? Je n'ai pas eu le temps de vérifier cette hypothèse.
Il faut peut-être mettre à jour le firmware du composant RN131C. La version utilisée avec la deuxième maquette est la V2.23 qui s'est trouvé être déjà bien plus stable que la V2.21 de la première maquette, tombée en panne. En effet lors de l'utilisation de cette première maquette, j'ai constaté des blocages et un manque de robustesse dans la transmission.

**Temps d'acquisition ADXL345**

Le temps passé dans l'interruption pour récupérer les données de l'accéléromètre numérique est de 35 µs.

La période entre deux appels à cette interruption est fonction de la bande passante (BP) sélectionnée.

| Bande Passante (Hz) | Période entre deux acquisitions | Débit de données correspondant (Hz) | A comparer à 2 × BP = ODR |
|---|---|---|---|
| 1600 | 320 µs | 3125 Hz | = 2 × 1600 = 3200 |
| 800 | 636 µs | 1572 Hz | = 2 × 800 = 1600 |
| 400 | 1,28 ms | 781,2 Hz | = 2 × 400 = 800 |
| 200 | 2,56 ms | 390,6 Hz | = 2 × 200 = 400 |
| 100 | 5,1 ms | 196,1 Hz | = 2 × 100 = 200 |
| 50 | 10,2 ms | 98,04 Hz | = 2 × 50 = 100 |
| 25 | 20,4 ms | 49,02 Hz | = 2 × 25 = 50 |

*Tableau 15 : Période de l'IRQ ADXL345 en fonction de la bande passante*

On constate que, contrairement au mode I²C (voir 2.10.2 « Mesures en mode filaire par l'USB »), le mode SPI permet bien d'attendre la fréquence d'échantillonnage maximale du composant ADXL345.

### 4.8.2 Fréquence d'échantillonnage maximale de la fonction acquisition de voies analogiques

En conditions réelles, c'est à dire en connectant des capteurs analogiques sur les voies 1 à 6 du CAN, on a pu mesurer les performances suivantes.

En plaçant la maquette et ses capteurs dans une autre pièce à une distance d'une dizaine de mètres, on a pu récupérer en temps réel, sans perte d'échantillons, les données issues de *Nb* capteurs avec une fréquence d'échantillonnage maximum selon le tableau ci-dessous :

| Nombre de capteurs analogiques | Fréquence d'échantillonnage maximale |
|---|---|
| 1 | 24 kHz |
| 2 | 12 kHz |
| 3 | 8 kHz |
| 4 | 6 kHz |
| 5 | 4,8 kHz |
| 6 | 4 kHz |

*Tableau 16 : Performances de la maquette*

Ces résultats démontrent qu'il est possible d'échantillonner au moins une voie à plus de 20 kHz et d'envoyer les échantillons par Wi-Fi pour les visualiser en temps réel sur un ordinateur.

Rappelons que la maquette est alimentée par une simple pile LiPo de 3,3V.

On notera que le débit équivalent estimé (*Nb × Fe Max × 12 bits*) est de 288 kbps, ce qui équivaut à 30% de la capacité de la liaison, limitée à 1 Mbps ici par la passerelle SPI/UART.

Il y a donc une marge de progression possible juste en réussissant à contourner cet écueil technique. De l'autre côté, le composant Wi-Fi peut atteindre 54 Mbps.

# 5 Bilan et Perspectives

## 5.1 Bilan

Ce projet très complet, à la croisée de plusieurs matières, m'a permis de mettre en application tout un panel de compétences et connaissances.

Au niveau technique/scientifique :

- veille technologique (moyens) et scientifique (applications)
- compréhension de documents techniques (datasheet, technologies…)
- chaîne d'acquisition de traitement du signal
- mesure, métrologie
- capteurs analogiques et numériques
- prototypage électronique rapide
- programmation Java, C++
- programmation embarquée microcontrôleur : entrées/sorties, $I^2C$, SPI, CAN, Timer…
- technologies radiofréquence et plus particulièrement le Wi-Fi

Au niveau humain/social :

- compréhension/interprétation des besoins des chercheurs
- environnement international et multiculturel
- environnement de très haut niveau scientifique (laboratoire de recherche)
- savoir trouver les personnes et plateformes ressources
- savoir trouver et utiliser les forums internet interactifs

A la fin de mon stage, j'ai remis à l'équipe un équipement complet fonctionnel comprenant :

- la maquette électronique
- les programmes informatiques
- le manuel utilisateur
- les possibilités d'évolution et d'amélioration

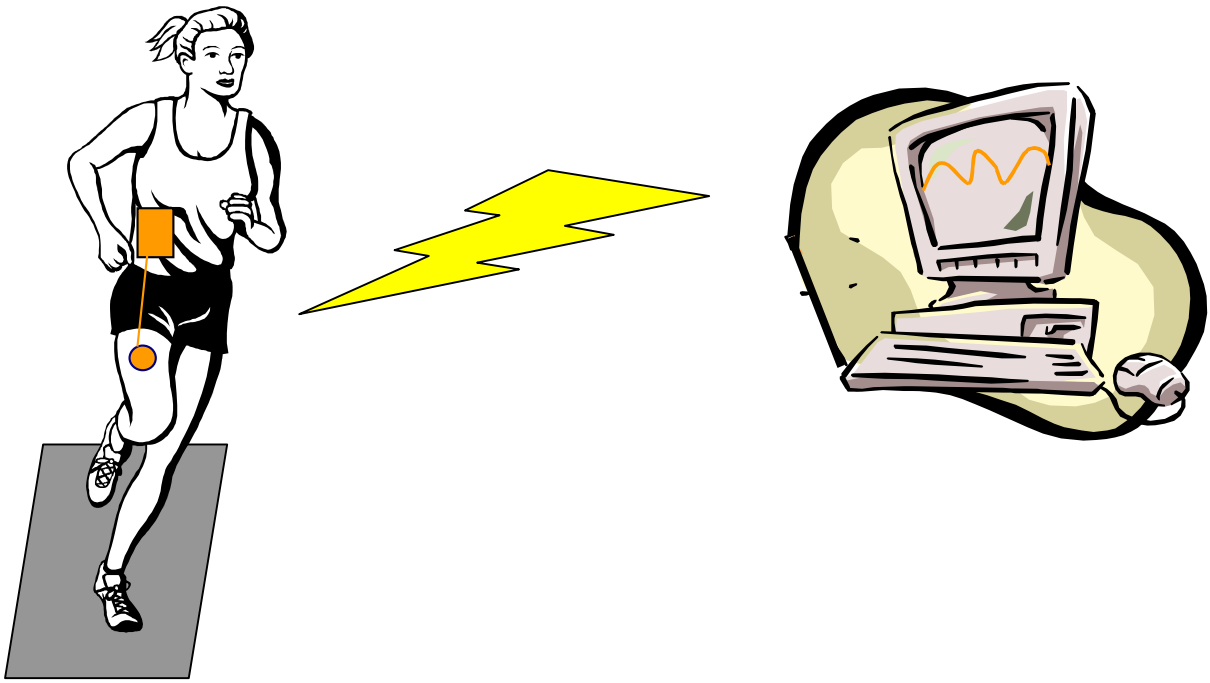
L'équipement respecte en grande partie le cahier des charges défini dans le « Tableau 2 : Récapitulatif du cahier des charges ».
En l'état, le matériel livré permet de récupérer par Wi-Fi les données de 1 à 6 capteurs analogiques et d'un accéléromètre numérique triaxial.
Le logiciel déployé sur l'ordinateur récupère ces données pour les enregistrer dans un fichier texte et les visualiser graphiquement à l'écran.

En quelques mois de stage, les possibilités de ce démonstrateur embarqué ont montré une partie de son potentiel.
Les sources des développements électronique et informatique associés fournis en annexe constituent une bonne base pour envisager d'autres fonctionnalités.

Le document « *Description programmes .pdf* » fournit dans l'Annexe B. « Documents » comprend un chapitre « 4. PISTES D'AMELIORATIONS » avec les indications pour qu'un programmeur puisse les exploiter.

## 5.2 Diversification des capteurs

Les capteurs intégrés à ma maquette sont principalement destinés à des applications vibro-acoustiques.
Néanmoins, des variantes sont possibles, l'architecture étant suffisamment ouverte pour accueillir d'autres types de capteurs.
Par exemple, une autre équipe du laboratoire m'a suggéré des applications biométriques. Ils imaginent équiper une personne avec un matériel léger qui puisse mesurer des mouvements et/ou des paramètres biologiques.
L'avantage évident est que l'on puisse fixer cet équipement sur le sujet à observer en le laissant libre de ses mouvements. Certes il reste quelques fils reliant le(s) capteur(s) au boîtier portable mais son alimentation par une pile et la transmission par Wi-Fi fait qu'il n'a besoin d'aucun support physique pour le relier au monde extérieur. Toutefois la technologie utilisée limite l'utilisation à l'intérieur d'un bâtiment à une dizaine de mètres ce qui suffisant pour effectuer une opération avec l'observateur dans une pièce et le sujet à observer dans une autre.



Pour mesurer des mouvements, il existe ce que l'on appelle une centrale IMU (Inertial Measurement Unit) qui combine les données d'accélération et gyroscopiques. En transmettant les données temporelles brutes, on laisse à l'utilisateur le soin de développer les algorithmes de calcul à la réception sur l'ordinateur pour traduire le mouvement.
Dans le domaine de la santé, des électrodes peuvent surveiller le rythme cardiaque, des ondes cérébrales, des mouvements musculaires ou tout autre paramètre biologique

On a acheté un capteur IMU numérique accessible sur le port I²C du *Maple*.
Seul la partie câblage a été décrite et effectuée (voir 3.2.2 « $I^2C$ pour IMU »).
Pour la partie programmation, on pourra s'inspirer de l'exemple développé pour gérer le capteur ADXL345 en mode I²C (voir 2.7 « Mise en œuvre d'un capteur numérique : accéléromètre »).

On a vu au paragraphe 3.4.2 « Pompe de charge 5V » qu'il était possible de fabriquer une tension supérieure à 3,3 Volts. Sur ce même principe ou avec un convertisseur DC-DC, on peut alimenter un capteur de type IEPE fonctionnant sous 9 à 12 Volts.

## 5.3 Perspectives sur les débits

### 5.3.1 Evolutions matérielles

Le plus gros goulet d'étranglement de notre maquette est le débit qui est limité par les technologies radio-fréquence actuelles.
Mais les évolutions sont encourageantes dans ce domaine.
Le Wi-Fi, que nous avons utilisé dans ce projet, devrait voir son débit multiplié par quatre avec la norme IEEE802.11ac (aussi appelé Wi-Fi 5G pour « Cinquième Génération ») et atteindre 7 Gbps [78][79]. Plutôt destiné aux contenus multimédias, pour fluidifier les flux vidéo haute définition, ce débit sera confortable pour transmettre simultanément les données issues de plusieurs capteurs échantillonnés à quelques dizaines de kilohertz.

L'UWB, une autre technologie sans fil, promet des débits très importants (pour des contenus multimédia) et est à suivre comme nous l'avons évoqué dans le paragraphe 1.2.4 « Standards pour les réseaux de capteurs sans fil ».

A plus brève échéance, des modules Wi-Fi basse puissance sortent régulièrement sur le marché.
A l'instar de notre *WiFly*, le module *XBee Wi-Fi* de chez Digi (sorti été 2011) [80] promet un débit de 54 Mbps, alimenté sous 3,3V. Mais la communication avec le microcontrôleur est possible directement par un SPI de 3,5Mbps. Ce débit permet théoriquement d'échantillonner 4 voies 12 bits à 40 kHz ou 51,2 kHz.

### 5.3.2 Evolutions logicielles

Côté programmation, il faut regarder si une implémentation du canal DMA (Direct Memory Access) est possible entre le CAN (échantillonnage des voies analogiques) et le SPI1 (transmission des échantillons au Wi-Fi). On peut espérer optimiser le temps de traitement des données par ce biais.

Pour sécuriser la transmission, on a volontairement scindé un échantillon codé sur 12 bits en 2 mots de 8 bits comportant 6 bits d'infos utiles et 1 bit de LSB/MSB.
Pour accélérer la transmission, on peut envisager de concaténer les échantillons des voies.
Ce qui donnerait par exemple pour 3 voies :

$3 \times 12 = 36$ bits à transmettre.

| | | |
|---|---|---|
| Actuellement : | 2 octets par voie ($2 \times 3$) | = 6 octets |
| En concaténant : | 36/8 | = 4 octets |

Soit un gain de 2 octets... mais en perdant en robustesse.

Par ailleurs, il faudrait envisager des méthodes de compression de données sans perte.
Par exemple, en codant uniquement la variation entre deux échantillons successifs et en contrôlant périodiquement par l'envoi de la vraie valeur.
De ce principe découlent des méthodes plus évoluées comme le « *Continuously variable slope delta modulation* » [81].

## 5.4 Synchronisation de plusieurs modules

On peut envisager de synchroniser plusieurs modules pour fusionner des données physiques récoltées à divers endroits. Par exemple, on peut mesurer les vibrations engendrées par le passage d'un tramway en espaçant les modules le long de la voie.
Le principe consiste à mesurer les délais de transmission entre les modules et de les corriger.

Au moins deux pistes seraient à évaluer :

- le protocole de précision temporelle IEEE1588 [82]

L'IEEE1588 pourrait être implémenté en utilisant la nouvelle génération de STM32 qui supporte cette fonction en natif.

- le GPS (Global Positioning System)

Le GPS permet non seulement la géo-localisation mais il fournit également une méthode pour partager des signaux de synchronisation. Un module GPS compatible « Arduino » peut être ajouté à notre maquette.

Il était prévu de monter une deuxième maquette identique à la première pour réaliser cette fonction. Malheureusement, la première étant tombée en panne, la construction de la première a juste suffit à remettre en état les développements réalisés jusqu'alors.

Avec l'architecture Wi-Fi, chaque maquette aura une adresse IP différente pour les reconnaître et les adresser.

## 5.5 Alimentation et autonomie

Pour évaluer l'autonomie, il faudrait faire des mesures en conditions réelles et mesurer la consommation. Celle-ci varie en fonction du nombre de voies et de la fréquence d'échantillonnage.
Pour l'instant, une pile LiPo de 1 Ampère, rechargeable par un câble USB, assure le fonctionnement.
Mais on peut développer une solution de récupération de l'énergie ambiante pour recharger une pile et rendre le système parfaitement autonome. J'ai étudié ces possibilités pour mon épreuve T.E.S.T en électronique (« Nouvelles sources d'énergie électrochimiques pour la microélectronique » - 2010).

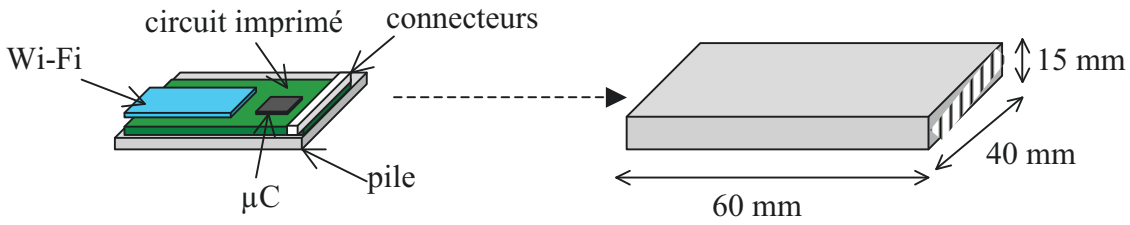
Les microcontrôleurs et module de radiocommunication Wi-Fi ont des fonctions d'économie d'énergie. Ils peuvent être programmés pour prolonger l'autonomie de la maquette.

Il serait aussi intéressant de pouvoir contrôler l'état de la batterie avec un indicateur pour signaler que celle-ci doit être rechargée.

Enfin, en cas de perte de communication, il faut prévoir un mécanisme de réinitialisation et de reconnection automatique du module embarqué.

## 5.6 Ergonomie

Ce projet a été réalisé en assemblant des modules déjà montés, un peu à la manière de Lego®.
Pour le protéger, on peut le mettre dans un boîtier en prévoyant des trous pour connecter les capteurs et la pile sur les embases.
La partie électronique peut être miniaturisée. Les modules « Arduino » sont très pratiques pour faire des prototypes. Mais on peut obtenir un système plus compact en implémentant les composants « utiles » sur un seul circuit imprimé multicouches et ne prévoir que des connecteurs pour les capteurs. La plupart des modules sont « Open Source » et leurs schémas disponibles.
Les deux composants les plus volumineux sont le RN131C Wi-Fi (37×20×4 mm) et la pile (51×34×6 mm). Le tout devrait tenir dans un volume de 60×40×15 mm.




## 5.7 Traitement des données informatiques

### 5.7.1 Traiter les échantillons non reçus

Les chapitres 4.4 « Acquisition : description du format de transmission des données » et 4.6 « Enregistrement et traitement des données » expliquent comment sont traités et enregistrés les échantillons, avec un système de contrôle du temps réel.
Pour finaliser ce fonctionnement, à la fin de l'acquisition, il faut relire le fichier texte généré et remplacer les échantillons non traités :

- interpoler les échantillons manqués 'singleton' (ceux notés ***)
- rajouter des zéros (ou autre chose, comme la valeur moyenne des données encadrantes par exemple) là où on a manqué une série d'échantillons, en décalant les numéros d'échantillons d'autant (première colonne du fichier texte)

### 5.7.2 Limitation de la Fréquence d'Échantillonnage

J'ai mesuré à l'oscilloscope de façon empirique les limites constatées de la fréquence d'échantillonnage en fonction du nombre de voies analogiques.
Il faudrait fixer en dur ces limites dans les choix des paramètres de l'IHM.

| Nombre de capteurs analogiques | Fréquence d'échantillonnage Max (en kHz) |
|---|---|
| 1 | 23 |
| 2 | 12 |
| 3 | 8 |
| 4 | 6 |
| 5 | 4,8 |
| 6 | 4 |

### 5.7.3 Acquisition simultanée par le CAN

Etant donné que le CAN du STM32 effectue une acquisition successive voie par voie, il peut être utile d'ajouter un étage bloqueur en amont pour obtenir la récupération simultanée de toutes les voies analogiques.

### 5.7.4 Calibration des entrées

Les échantillons des entrées sont récupérés tels quels en sortie du convertisseur, c'est à dire sous une forme numérique de 12 bits (0 à 4095).
En ajoutant une IHM de calibration des entrées (gain, offset, unité), on pourra visualiser sur l'axe Y du composant graphique et enregistrer dans le fichier texte dans la bonne unité.

Par exemple, pour le capteur de courant qui n'est pas conditionné en 3,3V et malgré le pont diviseur, il faut pouvoir dire quelle valeur correspond au zéro (offset) et quelle est la précision en Ampère (sachant que le composant donne en principe la pleine échelle pour +/- 10 milliTesla).

### 5.7.5 Affichage temporel

L'affichage temporel n'a été implémenté que pour 1 voie analogique sur 2000 points.
Il faut modifier le composant graphique pour :

- Afficher les graphes des N voies analogiques (N de 1 à 6)
- Afficher les 3 axes de l'accéléromètre
- Pouvoir sélectionner la durée de l'axe X (en fonction du nombre d'échantillons et de la fréquence d'échantillonnage : N échantillons ⇔ T ms)

### 5.7.6 Module de calcul

On peut prévoir un module de calcul pour implémenter des opérations de traitement du signal (moyenne, FFT, etc...) et les afficher.

Par exemple, un premier calcul simple est d'afficher la fréquence d'une sinusoïde.
J'avais développé un algorithme lors de la première maquette réalisée avec Processing.
C'est un calcul tout simple qui calcule la fréquence d'un signal par détection de seuil (passages par zéro sur front montant) en moyennant par rapport à la fréquence d'échantillonnage.
Ce calcul est surtout utile pour démontrer que l'on récupère bien la fréquence d'une sinusoïde pure injectée avec un générateur de signaux sur une des entrées analogiques (à la place d'un capteur).
Il faut prévoir d'afficher les résultats dans l'interface utilisateur.

Ensuite, on peut développer sur le même modèle des outils de calcul plus évolué (FFT, moyenne sur une bande de fréquence, Skew, Kurtosis...)

# Conclusion

L'acquisition de données sans fil est un véritable défi pour les ingénieurs.
Il doit résoudre la difficile équation de transmettre le maximum d'informations avec un minimum d'énergie.
La plupart des solutions existantes sont, soit trop gourmandes en énergie mais autorise des débits importants, soit faible consommation au détriment de la bande passante fortement réduite.
Ce mémoire illustre que l'on peut, à partir de composants peu onéreux du commerce, bâtir une solution efficace et robuste par une programmation astucieuse, permettant d'obtenir un débit de données substantiel avec une alimentation de quelques volts.
Cette architecture originale s'articule autour de briques logicielles et matérielles, basée sur la philosophie Arduino, en « Open Source ».
Le principe est de revenir aux fondamentaux : ne transmettre que l'information utile en continu par une liaison radio.

Ce projet démontre techniquement et pratiquement qu'il est possible de surveiller en temps réel, les données temporelles issues de capteurs par une liaison sans fil dans la gamme acoustique, avec une alimentation adaptée à un système embarqué de faible consommation.
Nous avons atteint la récupération sans défaut d'une voie échantillonnée à 20 kHz sur 12 bits pendant plusieurs heures, avec une pile plate de 3,7 V et une liaison Wi-Fi de 1 Mbps.
Une fréquence d'échantillonnage de 5 kHz permet de suivre 6 voies simultanément.
Les capteurs mis en œuvre sont analogiques (microphone, accéléromètre mono-axe, capteur de courant) et numérique (accéléromètre triaxe).

Une mallette contenant tout le nécessaire pour utiliser ce système d'acquisition sans fil a été livrée à l'équipe du laboratoire avec son manuel d'utilisation.
C'est un prototype fonctionnel utilisable par les enseignants-chercheurs du GipsaLab pour réaliser des mesures et des démonstrations.

Le projet pourra évoluer vers un système parallélisable de plusieurs nœuds synchronisés et s'ouvrir à d'autres domaines applicatifs, avec des capteurs dédiés.

# Annexes

Etant donné la taille et la nature des annexes, celles-ci sont fournies à part sur un support numérique joint à ce rapport.

## Annexe A. Image contenue clé USB

Contient tous les éléments finaux pour l'utilisation de la maquette :
- drivers
- programme à installer sur l'ordinateur
- sources nécessaires pour programmer la platine
- manuel d'utilisation

## Annexe B. Documents

Contient les fichiers **PDF** suivants :
- ***Description programmes*** : document destiné aux programmeurs pour l'évolution des logiciels
- ***Manuel Utilisateur*** : Manuel d'installation et d'utilisation de la maquette
- ***Procedure_de_cablage*** : document destiné aux électroniciens pour construire les platines électroniques de la maquette
- ***Couverture boite*** : l'image qui illustre la boîte où se trouve la maquette et ses accessoires

## Annexe C. Sources programmes Maple

Fichier zip qui contient les sources du code Maple pour la partie embarquée.

## Annexe D. Sources programmes Ordinateur

Contient les solutions Visual Studio pour les programmes côté ordinateur.

## Annexe E. Datasheets

Contient les datasheets et les manuels de référence des composants utilisés dans la maquette.

# Bibliographie

*(Tous les sites Internet ont été consultés pour vérification le 8 Mars 2013)*


[1] Gipsa- Lab : http://www.gipsa-lab.grenoble-inp.fr/le-laboratoire/presentation.php

[2] William Gardner. *Cyclostationarity in Communications and Signal Processing*, IEEE Press, 1994, ISBN 0-7803-1023-3

[3] Jacques Max, Jean-Louis Lacoume, *Méthodes et Techniques de traitement du signal, 5ème édition*, Editions Dunod, 2004, ISBN 2-10-048331-5

[4] « *Le maillon faible de la recherche, c'est le privé – cinq passionnés d'innovation issus du public et du privé racontent*», Usine Nouvelle n°3266 du 15 décembre 2011, pages 44-46

[5] Artus Patrick, Virard Marie-Paule, *La France sans ses usines*, Editions Fayard, Octobre 2011, ISBN 978-2-213-66597-9

[6] Projet TetrAS http://www.gipsa-lab.fr/~nadine.martin/TetrAS%20--%29%20AStrion.html

[7] Projet KAStrion http://www.gipsa-lab.fr/projet/KASTRION/

[8] Olivier Las Vergnas , *L'institutionnalisation de la « culture scientifique et technique », un fait social français (1970-2010),* Savoirs 2011, 27 (2012) 7-100

[9] Galères administratives des doctorants étrangers http://www.rue89.com/2012/04/23/carnets-de-galeres-administratives-de-chercheurs-etrangers-en-france-230357

[10] Gotix http://www.gipsa-lab.grenoble-inp.fr/projet/gotix/experimental_bench.html

[11] OROS France http://www.orosfrance.fr/

[12] B&K 4391, Accéléromètre industriel http://www.bksv.com/products/transducers/vibration/accelerometers/accelerometers/4391.aspx

[13] Colybris (capteur MEMS) http://www.colibrys.com

[14] PCB (capteur MEMS) http://www.pcb.com/Accelerometers/MEMS.asp

[15] Analog Devices (capteur MEMS) http://www.analog.com/en/mems-sensors/products/index.html

[16] *Le second souffle des capteurs MEMS*, Usine Nouvelle n°3176, 22 Janvier 2010

[17] Kalyanramu Vemishetty, *Embedded Wireless Data Acquisition System, Thèse 2005, http://scholar.lib.vt.edu/theses/available/etd-12202005-005049/unrestricted/Kalyan_Grad_Thesis_Final.pdf*

[18] Wireless Hart http://www.hartcomm.org/protocol/wihart/wireless_technology.html

[19] « What's Delaying Wireless Adoption? », 4 novembre 2011, Tom Kevan http://www.sensorsmag.com/sensors-mag/what-s-delaying-wireless-adoption-9110

[20] Wibree Nokia http://research.nokia.com/news/254

[21] Bluetooth Low Energy http://www.bluetooth.com/Pages/Low-Energy.aspx

[22] P. Morin, *Acquisition de quatre voies analogiques via une liaison Bluetooth*, Electronique Pratique n°369, Mars 2012

[23] Séminaire « *Détection non cohérente et résolution temporelle : application à la radio impulsionnelle* » http://www.gipsa-lab.grenoble-inp.fr/animation/seminaires.php?id_sem=346

[24] DecaWave – UWB Sensor http://www.decawave.com/scensor.html

[25] Jacques Marouani, *Kaptalia Monitoring : La surveillance médicale sans fil*, Magazine Electroniques n°30 Septembre 2012

[26] Diogo Koenig, Marilda S. Chiaramonte, Alexandre Balbinot, *Wireless Network for Measurement of Whole-Body Vibration*, ISSN 1424-8220, 2008, http://www.scribd.com/doc/49682235/Wireless-Network-for-Measurement-of-Whole-Body-Vibration

[27] Projet PermaSense http://www.permasense.ch/home/about.html

[28] Sukun Kim, Structure Monitoring using Wireless Sensor Networks , http://www.eecs.berkeley.edu/~binetude/course/cs294_1/paper_word.pdf

[29] Andrew K.S, Daming Lin, Dragan Banjevic, *A review on machinery diagnostics and prognostics implementing condition-based maintenance*, Octobre 2006, http://www.sciencedirect.com/science/article/pii/S0888327005001512

[30] National Instrument, *Selecting the Right Wireless Technology* , http://www.ni.com/white-paper/8939/en

[31] Charlie Stiernberg, *Five Things Every Scientist and Engineer Should Know about Wireless Sensor Measurements*, in LaboratoryEquipement, Septembre 2009, http://www.laboratoryequipment.com/articles/2009/03/five-things-every-scientist-and-engineer-should-know-about-wireless-sensor-measurements

[32] Sensors Magazine – Sensor technology news and real-world sensor application http://www.sensorsmag.com/

[33] InCheck Technologies - InBox™ Wireless Data Acquisition Modules http://www.incheckteh.com/InCheck/inbox.htm

[34] SKF Multilog On-line System WMx http://www.skf.com/group/products/condition-monitoring/surveillance-systems/on-line-systems/wireless-systems/skf-multilog-on-line-system-wmx/index.html

[35] Intesens : kit de diagnostic InteLog http://www.intesens.com/dms/index.php?option=com_content&view=article&id=31&Itemid=42&lang=fr

[36] National Instrument : Réseaux de capteurs sans fil http://www.ni.com/wsn/f/

[37] National Instrument: WLS-9234 http://sine.ni.com/nips/cds/view/p/lang/fr/nid/205714

[38] Telos B DataSheet http://www.willow.co.uk/TelosB_Datasheet.pdf

[39] TMote Sky DataSheet http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf

[40] MICAz DataSheet http://www.willow.co.uk/html/mpr2400-_micaz_zigbee.html

[41] FreeScale ZStar http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=RD3152MMA7260Q

[42] Introduction au Système Arduino http://arduino.cc/fr/Main/DebuterIntroduction

[43] Arduino Uno http://arduino.cc/en/Main/ArduinoBoardUno

[44] Remote Accelerometer http://tinkerlog.com/2010/02/07/remote-accelerometer/

[45] Digi XBee®, http://www.digi.com/xbee/

[46] *Modules XBee et télécommande*, Electronique Pratique n°343 Novembre 2009

[47] Funnel I/O et XBee : http://www.seeedstudio.com/blog/2009/01/10/funnel-io-remixed-by-seeedstudio/

[48] Libellium http://www.libelium.com/

[49] OpenPicus Flyport http://store.openpicus.com/openpicus/prodotti.aspx?cprod=015350

[50] STM32W-RFCKIT http://www.st.com/internet/evalboard/product/251361.jsp

[51] LeafLabs Maple http://leaflabs.com/devices/maple/

[52] WiFly Shield https://www.sparkfun.com/products/9954

[53] Hikob Open Lab http://www.hikob.com/

[54] Platine Maple (réf. Française) http://www.lextronic.fr/P22507-platine-maple.html

[55] STM32F103 Reference Manual http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference_manual/CD00171190.pdf

[56] Platine Maple (réf. Anglaise) https://www.sparkfun.com/products/10664?

[57] Tom Igoe, *Making Things Talk 1st Edition*, Editions O'REILLY, September 2007, ISBN 978-0-596-51051-0

[58] Maple IDE http://leaflabs.com/docs/ide.html

[59] Processing http://www.processing.org/

[60] Microphone MEMS ADMP401 https://www.sparkfun.com/products/9868

[61] ADXL345 Triple Axis Accelerometer Breakout https://www.sparkfun.com/products/9836

[62] ADXL345 Datasheet http://www.analog.com/static/imported-files/data_sheets/ADXL345.pdf

[63] C. Taylor, *WiFly Wireless Talking SpeakJet Server*, 4 Mars 2010, http://www.sparkfun.com/tutorials/158

[64] SPI/UART SC16IS750 datasheet http://www.sparkfun.com/datasheets/Components/SMD/SC16IS740_750_760.pdf

[65] RN131C User Manual http://www.rovingnetworks.com/resources/download/93/WiFly_User_Manual

[66] Davor Males et Guy Pujolle, *Wi-fi par la pratique, Deuxième Edition*, Editions Eyrolles, 2004, ISBN: 2-212-114009-5

[67] Client UDP pour Langage Processing http://ubaa.net/shared/processing/udp/documentation/udp/udp_class_udp.htm

[68] Fred Eady, *Implementing 802.11 with Microcontrollers - Wireless Neworking for Embedded Systems Designer*, Editions Elsevier Newnes, 2005, ISBN: 0-7506-7865-8

[69] Alain Boulenger, Christian Pachaud, *Aide-mémoire « Surveillance des machines par analyse des vibrations »*, Dunod - Usine Nouvelle – 2009, ISBN 978-2-10-051780-0

[70] Michel Lavabre, Fabrice Baudoin, *Capteurs: Principes et utilisation - Cours et exercices résolus*, Casteilla, 2007, ISBN 978-2-7135-2749-4

[71] RS Popovic, Hall Effect Devices - *Magnetic Sensors and characterization of semiconductors*, Adam Hilger, 1991, ISBN 0-7503-0096-5

[72] MLX91205 Datasheet http://www.melexis.com/Asset/Datasheet-MLX91205-DownloadLink-5505.aspx

[73] ADXL001 Datasheet http://www.analog.com/static/imported-files/eval_boards/EVAL-ADXL001.pdf

[74] PCB 660 series DataSheet http://www.pcbpiezotronics.fr/docs/produits/IMI_Sensors_serie_6601.pdf

[75] IMU https://www.sparkfun.com/products/10252?

[76] EVOLA DIY http://www.evola.fr/

[77] Cedric Moonen, *High-speed Charting Control,* http://www.codeproject.com/Articles/14075/High-speed-Charting-Control

[78] *Sans Fil, Wi-Fi Cinquième Génération*, Usine Nouvelle n°3270, 2 Février 2012

[79] Pierrick Arlot, *Les normes Wi-fi pulvérisent le mur du gigabit par seconde*, Electroniques n°24 Février 2012

[80] Xbee Wi-Fi Digi http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-wi-fi#overview

[81] Continuously variable slope delta modulation http://en.wikipedia.org/wiki/Continuously_variable_slope_delta_modulation

[82] Standard IEEE1588 http://www.nist.gov/el/isd/ieee/ieee1588.cfm

**C.N.A.M : Mémoire Ingénieur en ELECTRONIQUE – UE UA5M25**

# Conception et réalisation d'un démonstrateur embarqué de traitement du signal, avec réseau de capteurs sans fils

**Nathalie TEPPE** **Grenoble, le 29 Mai 2013**

---

**Résumé**

L'acquisition de données à distance dans l'industrie pour la surveillance et le diagnostic de machines, est un secteur en plein essor. Dans cette optique, un démonstrateur embarqué de traitement du signal doit permettre de transmettre des données temporelles représentant fidèlement le signal enregistré. Il doit être à la fois fiable, robuste, haut débit et temps réel.
Ce démonstrateur se décompose en une maquette électronique embarquée, de type « *Arduino* », à base d'un microcontrôleur, d'une liaison Wi-Fi basse consommation et de capteurs MEMS, accompagné d'un logiciel utilisateur pour paramétrer et enregistrer les résultats. Les données, issues des échantillons, sont transmises en continu par la liaison radio, contrôlée périodiquement.
Les performances finales permettent d'échantillonner un signal à 20 kHz en temps réel sous une alimentation de 3.3V.

**Mots-clés :** acquisition de données à distance, Wi-Fi, temps réel, surveillance, embarqué

---

**Abstract**

Remote data acquisition dedicated to industrial applications such as machinery monitoring and diagnosis, is a growing market. Considering this opportunity, an embedded demonstrator board for signal processing must transmit time data mirroring exactly the recorded signal. It must be reliable, robust, high-speed and real-time.
This demonstrator is built upon an embedded electronic « Arduino-based » model, with a microcontroller, a low-power Wi-Fi connection and MEMS sensors, along with an user software to set-up and record results. Sample data are sent though radio on a continuous streaming basis, periodically controlled by a timestamp.
The final performance reaches to sample a signal at 20 kHz rate in real time under a 3.3V supply.

**Keywords :** Remote data acquisition, Wi-Fi, Real Time, Monitoring, Embedded