République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

ECOLE NATIONALE POLYTECHNIQUE

DÉPARTEMENT D'ÉLECTRONIQUE

Projet de fin d'études

En vue de l'obtention du diplôme d'Ingénieur d'Etat en Electronique

Thème :

Conception d'un capteur réseau dédié à l'estimation des paramètres de mouvement d'une source acoustique

Encadré par :

Dr. Rabah SADOUN

Pr. Adel BELOUCHRANI

Réalisé par :

M. Sami BENABIDALLAH

M. Sidali DIOUANI

Promotion : Juin 2015

Dédicaces

Je dédie ce modeste travail à :
mes très chers parents
mes frères et sœurs,
tous ceux qui ont contribué de près ou de loin à mes études.

Sami

Je dédie ce modeste travail à :
mes très chers parents
mon grand-père,
mes frères et sœurs,
tous ceux qui ont contribué de près ou de loin à mes études.

Sidali

Remerciements

En premier lieu et avant toute chose, Louange et Action de Grace à Allah le tout puissant pour toutes ses grâces et ses faveurs, pour nous avoir accordé la chance d'étudier à l'École Nationale Polytechnique et nous avoir donné le courage et la force d'accomplir ce modeste travail.

Ce travail a été effectué au sein du département d'électronique, au niveau du laboratoire de signal et communication de l'École Nationale Polytechnique (Alger). On tient avant tout à remercier notre encadreur de projet Monsieur Rabah Sadoun. Ses conseils et son optimisme nous ont été d'une aide précieuse tout au long de ce projet. On le remercie pour la liberté qu'il nous a laissée dans notre travail et pour tout le temps qu'il nous a consacré lors de l'écriture de ce mémoire.

Merci également à Monsieur Adel BELOUCHRANI de nous avoir aidé, ses conseils lors de nos entrevues ont toujours été fructueux. On tiens également à remercier les membres de notre jury, Madame Mhania GUERTI pour l'honneur qu'elle nous fait de le présider, Monsieur Mourad ADNANE et Monsieur Ahmed BOUZID pour l'intérêt qu'ils ont porté à notre travail en acceptant d'être examinateurs, leurs remarques et suggestions nous ont permis d'améliorer la qualité de notre travail.

On tiens à remercier également Mlle Hakima TIMELET pour l'assistance, l'aide et la sympathie qu'elle nous a témoignée.

On est très reconnaissant envers Salah BOUHOUN, Hacene BOUAROUA et Smail BECHICHE de nous avoir aidé. Remerciements à tous ceux qui ont contribués de près ou de loin à l'élaboration de ce travail.

ملخص

العمل المقدم في هذا التقرير يتناول تنفيذ خوارزمية تحديد موقع مصدر الصوت. قدم للقارئ غير الملم جميع القواعد الأساسية لفهم هذا الموضوع.

أولا، قدمنا مفاهيم تحديد موقع الصوت و حالة الفن في هذا المجال .ثانيا، قدمنا منهجيات التصميم المختلفة و بعد دراسة هذه المنهجيآت، اخترنآ المنهجية التي تحقق الشروط المحددة في (Model–Based Design) (Rapid Prototyping) بعد ذلك قمنا بتنفيذ المنهجية المختارة في سياق هذا التطبيق. أخيرا، قمنا باختبار التطبيق في اوساط مختلفة فتحصلنا على نتائج واعدة.

الكلمات المفتاحية : خوارزمية, موقع مصدر الصوت. منهجيات التصميم

Résumé

Le travail présenté dans ce mémoire traite l'implémentation d'un algorithme de localisation de source acoustique. Cette thématique a été traitée, dans son ensemble, tout au long de ce manuscrit, de façon à fournir au lecteur non initié l'ensemble des bases indispensables à la bonne compréhension de ce problème de traitement de signal et d'implémentation sur cible matérielle. Dans un premier temps, nous avons présenté des notions de localisation acoustique, ainsi que l'état de l'art dans ce domaine. Nous avons mis en contraste l'algorithme à implémenter ainsi que les contributions qu'il apporte par rapport aux anciens algorithmes de localisation. Dans un deuxième temps, nous avons exposé différentes méthodologies de conception. Après une étude de ces méthodologies, nous en avons choisi une satisfaisant les conditions du cahier de charges (Prototypage rapide). Par la suite, nous avons mis en œuvre la méthodologie choisie dans le contexte de notre application. Enfin, nous avons testé notre application dans différents environnement, cela nous a retourné des résultats prometteurs.

Mots clés :
Localisation acoustique, Prototypage rapide, Implémentation, Capteur réseau.

Abstract

The work presented in this thesis deals with the implementation of a sound source localization algorithm. This theme was treated as a whole, throughout this manuscript, so provide the uninitiated reader all the essential bases for the understanding of this signal processing problem and implementation of hardware target. First, we introduced the concepts of acoustic localization and state of the art in this field. We contrasted the algorithm to be implemented and the contributions it brings compared to the old location algorithms. Secondly, we presented different design methodologies. After a study of these methodologies, we have chosen one which satisfies the conditions of the specifications (Rapid Prototyping). Subsequently, we have implemented the methodology chosen in the context of our application. Finally, we tested our application in different environment, it turned us promising results.

Keywords :
Acoustic localization , Rapid prototyping, Implementation, Network sensor.

Table des matières

Table des figures

Liste des tableaux

Introduction Générale

Ce rapport décrit le travail effectué au niveau du laboratoire de signal et communication de l'École Nationale Polytechnique (Alger) pour l'obtention du diplôme d'ingénieur d'état en Électronique.

Ce travail s'inscrit dans la continuité des travaux effectués aux seins des laboratoires signal et communication de l'ENP et Instrumentation de l'USTHB couvrant le thème de localisation acoustique, il est question cette fois ci de l'implémentation d'un algorithme de localisation acoustique et détermination de paramètres d'un mobile. Un des aspects de l'implémentation sera l'étude et la mise en œuvre de la méthodologie dite "Model based design", il s'agit de projeter une modélisation haut niveau de l'algorithme sur une cible matérielle, pour cela des outils de prototypage rapide seront exploités. Nous utiliserons principalement l'environnement Mathworks pour la modélisation et la validation de l'algorithme puis de son implémentation, les environnements propres au matériel nous permettrons de traiter les questions d'interfaçage.

Afin de valider le fonctionnement de notre système dans des conditions réelles, une plateforme expérimentale a été mise en place, émulant le passage d'une source sonore.

Le présent mémoire est structuré en cinq chapitres. Après une introduction qui englobera les motivations de ce travail, le premier chapitre abordera la problématique de la localisation de sources sonores ainsi qu'une rétrospective sur l'état de l'art de la localisation acoustique, une première étape sera la détermination du modèle mathématique (Pré-traitement et traitement du signal) découlant du phénomène physique. Un second modèle sera présenté, il s'agira de l'algorithme à implémenter sur la cible matérielle. Nous décrirons notamment dans ce chapitre le travail expérimental réalisé ainsi que la chaine d'instrumentation utilisée.

Le second chapitre exposera principalement les différentes méthodologies de conception existantes, il s'en suivra la présentation de l'approche utilisée dans notre travail.

Le troisième chapitre s'articulera sur la mise en œuvre du modèle de traitement suivant la méthodologie de travail choisie précédemment.

le quatrième chapitre concernera d'abord la présentation des différentes cibles technologiques, puis l'implémentation et la génération de code. Une extension de ce chapitre proposera une expansion du modèle pour la transmission des résultats vers un réseau.

Le cinquième chapitre est consacré à la présentation des résultats d'estimations des paramètres par simulation et par pratique en utilisant la plateforme mécanique en champ libre. Nous terminerons ce travail par une conclusion énumérant les contributions apportées.

Problématique et objectifs

La problématique étant la localisation acoustique, plus précisément, l'estimation de paramètres de mouvement d'une source sonore, à savoir, la vitesse, la hauteur et la fréquence de la source acoustique.

Notre objectif à terme est de réaliser un capteur réseau capable d'estimer les paramètres de mouvement d'une source acoustique et de transmettre ses résultats vers un réseau. Dans ce but, nous pouvons citer principalement quatre objectifs à ce travail :

— Faire le point sur les méthodes de localisation déjà existantes.

— Proposer une solution de localisation.

— Faire le point sur les méthodologies de conception et en choisir une satisfaisant nos contraintes et cahier de charges.

— Réalisation d'une plateforme expérimentale.

— Réalisation du capteur réseau proprement dit.

Chapitre 1

Localisation acoustique

Sommaire

La localisation de sources sonores est un domaine de recherche en plein essor. En effet, on voit de nos jours l'apparition de nouvelles techniques de localisation de plus en plus performantes et pointues. Ces techniques, qui autre fois étaient rudimentaires, ont pu voir le jour grâce au développement d'algorithmes de localisation qui ne cessent d'évoluer ainsi qu'à leur déploiement sur des systèmes embarqués. La localisation acoustique de sources sonores en mouvement repose sur le traitement du son émis par l'objet en mouvement, l'analyse et l'étude du signal permettent la détermination des paramètres caractérisants l'objet en mouvement. Les premières techniques de localisation acoustique remontent au début du 19 ème siècle, cependant, avec l'apparition du RADAR lors de la seconde guerre mondiale, les techniques de localisation acoustique se sont vues remplacer. De nos jours, on assiste à un regain d'intérêt pour les systèmes de localisation acoustique pour les avantages qu'ils apportent (passifs, détection d'objets à basse altitude, dispositifs peu coûteux, etc).

1.1 Localisation acoustique et effet doppler

1.1.1 Localisation acoustique

La localisation acoustique, précédemment appelée positionnement acoustique (acoustic location) était utilisée avant même l'avènement de l'avion. En effet, le premier qui fut à

concevoir un système de détection en acoustique passive fut Léonard de Vinci qui écrivit en 1490 : " si vous arrêtez votre bateau et placez l'extrémité d'un tube creux dans l'eau et l'autre extrémité dans votre oreille, vous entendrez les autres bateaux qui bougent à de grandes distances de vous." En fait, deux tubes, un dans chaque oreille, ont été utilisés jusqu'à la première guerre mondiale pour détecter et estimer la direction de bruits sous-marins.Au début du XXème siècle les marins utilisaient un topophone [1] formé de deux cornets (Figure 1.1), un pour chaque oreille. Ce système repose sur le principe de différence de temps d'arrivé du son entre les deux oreilles, celles ci jouent le rôle de capteurs, lorsque le déphasage est nul on repère la direction de la source. On peut augmenter la précision en augmentant la taille du topophone i.e. en espaçant les écouteurs, cependant si l'écart entre les deux extrémités de l'appareil est supérieur à une longueur d'onde du signal provenant de la source on peut voir apparaître des ambiguïtés pouvant induire en erreur l'opérateur.

FIGURE 1.1: Topophone

Par la suite, ce topophone a beaucoup évolué, notamment vers les années trente où furent fabriqués des dispositifs d'écoute géants, toujours avec des cornets, allant de l'appareil acoustique à des architectures dédiées à la localisation d'avions ennemis pour parer à des bombardements surprises [2].

FIGURE 1.2: Topophones à la deuxième guerre mondiale

Ces techniques rudimentaires, certes limitées par les capacités de l'oreille humaine, permettent d'illustrer simplement le principe d'une goniométrie effectuée grâce à un réseau de capteurs, qui sont dans ces exemples des cornets ou des pavillons acoustiques. Le maximum d'intensité sonore rayonnée par la source sonore correspond à la direction d'arrivée de l'onde acoustique.

D'une manière générale, les algorithmes développés pour la localisation acoustique exploitent les signaux issus d'un capteur ou d'un réseau de capteurs et sont une transposition mathématique du principe physique qu'est la goniométrie acoustique.

Cependant, avec l'apparition du RADAR (RAdio Detection And Ranging) en 1934 [3], tous les travaux sur les systèmes de détection acoustique étaient arrêtés, et tous le financement redirigé vers la recherche dans les systèmes RADAR, en raison de sa robustesse et de sa précision. Néanmoins, l'intérêt pour les systèmes de détection acoustique a été relancé, lors des dernières décennies, car contrairement au RADAR, ces systèmes offre une détection en basse altitude, ils ne présentent pas de problèmes d'interférences électromagnétiques, qui dans le cas du RADAR peuvent induire l'opérateur en erreur, et du fait que la localisation acoustique soit passive, le système ne peut être détecté.

Dans cette optique, plusieurs approches ont été proposées , on s'intéressera à la technique de localisation acoustique basée sur l'effet Doppler, pour estimer la fréquence, la vitesse et l'altitude de la source sonore.

1.1.2 Effet doppler

Christian Doppler (1803-1853) mathématicien et physicien autrichien s'est rendu célèbre dans l'une de ces publications "Sur la lumière colorée des étoiles doubles et d'autres étoiles du ciel" pour la découverte de cet effet [4].

FIGURE 1.3: christian Doppler

L'effet Doppler se traduit par le décalage de fréquence d'une onde (dans notre cas acoustique), entre la mesure à l'émission et celle a la réception lorsque la distance entre les deux

varie au cours du temps.

L'effet a été testé pour la première fois pour une onde sonore par Christoph Buys-Ballot en 1845, en faisant jouer une note par un groupe de musicien sur un train en marche à Amsterdam.

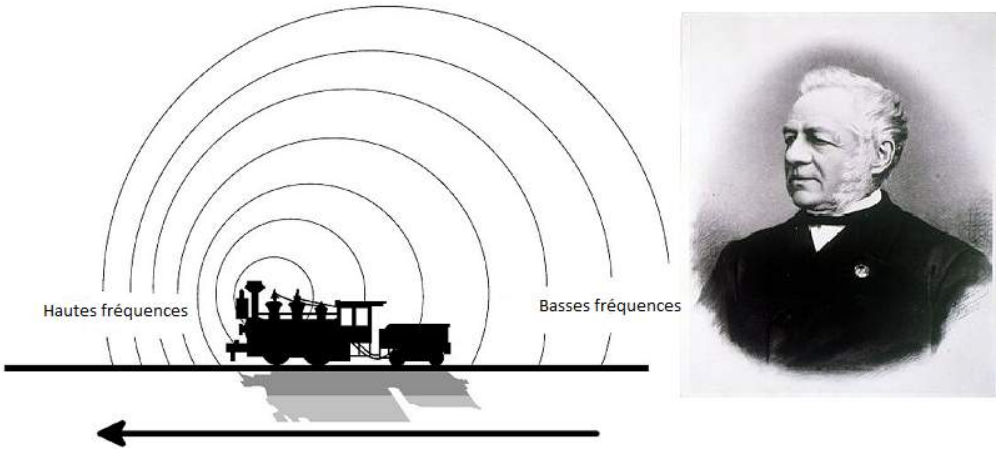


FIGURE 1.4: Effet Doppler

Différents cas de figure peuvent se présenter. Nous allons surtout nous intéresser au cas d'une source acoustique mobile avec un observateur (Microphone) immobile.

Illustration :

Soit un émetteur qui se dirige vers l'observateur situé à une distance d et se meut à une vitesse constante v.
Au temps $t_1 = 0$, le premier front d'onde est émis. Celui-ci arrive à l'observateur à un temps $t'_1 = \frac{d}{c}$ où c est la vitesse de propagation de l'onde sonore dans le milieu.

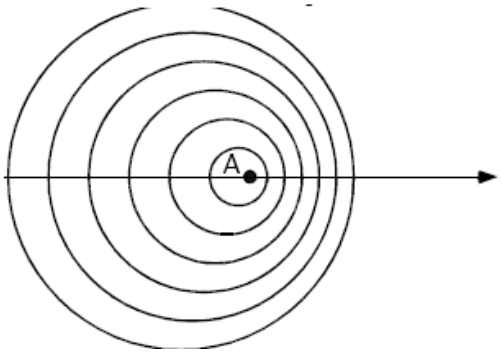


FIGURE 1.5: Front d'onde

A $t_2 = T$, un second front d'onde est émis, pendant ce temps l'émetteur a parcouru une distance $d' = v.T = \frac{v}{f}$ le deuxième front d'onde arrive à l'observateur au bout d'un temps $t'_2 = t_2 + \frac{d-d'}{c}$

Si l'on note T' la période du signal sonore perçu par l'observateur

$T' = t'_2 - t'_1 = t_2 + \frac{d-d'}{c} - \frac{d}{c}$

On aura ainsi :

$$f' = \frac{f}{(1 - \frac{v}{c})} \tag{1.1}$$

Un raisonnement analogue donne pour une source qui s'éloigne la relation :

$$f' = \frac{f}{(1 + \frac{v}{c})} \tag{1.2}$$

1.2 Formulation du problème de la localisation acoustique

La localisation acoustique est un ensemble de techniques, qui, à partir de signaux sonores permet la déduction de paramètres caractérisants un phénomène (déplacement, position, etc).
On s'intéresse dans ce cas à l'estimation de la vitesse, de la fréquence source ainsi que la distance d'un mobile émettant un signal sonore au tour d'une fréquence f_0 inconnue.

1.2.1 Modèle physique

Dans cette section nous illustrerons le modèle physique de la localisation acoustique, duquel sera tiré le formalisme mathématique.

Où :
h : Altitude.
d : Distance entre le mobile et le microphone (distance oblique).
r : Distance horizontale entre le mobile et le microphone.

d et r sont tout deux fonction du temps.

Le mobile se déplace à une vitesse constante v , et émet un signal sonore de fréquence constante, cependant le sigal perçu s'apparente à un signal modulé en fréquence FM (effet Doppler).
Pour résoudre un tel problème, on a besoin de déterminer l'expression, de la fréquence instantanée du signal sonore.

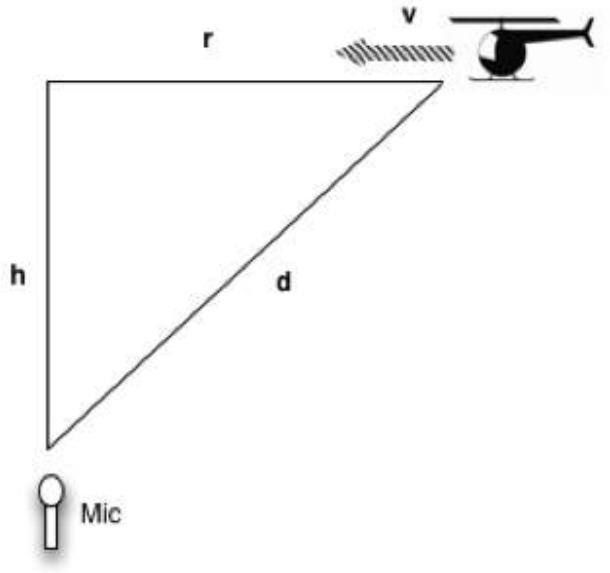


FIGURE 1.6: Diagramme schématisant le modèle physique

1.2.2 Modèle mathématique

La résolution du problème de localisation acoustique nécessite la transcription du modèle physique en un modèle mathématique qui régit les phénomènes étudiés.

On s'intéresse à la détermination puis l'étude de la formule de la fréquence instantanée du signal sonore perçue au niveau de l'observateur en fonction des paramètres (c, v, tc, h et v). Où :

c : La vitesse de propagation de l'onde sonore dans le milieu (340m/s dans l'air).

v : Vitesse du mobile.

t_c : Le temps écoulé pour que le mobile soit au dessus du microphone ($d(t_c) = h$).

d : Distance entre le mobile et le microphone

f_0 : Fréquence du signal sonore au repos.

t : Le temps au niveau du microphone (délai dû à la propagation).

τ : Indique le temps au niveau du mobile.

Le mobile est supposé à une altitude constante et se déplace à vitesse constante. Le modèle mathématique sera :

$$t = \tau + \frac{d}{c} \tag{1.3}$$

La distance entre le microphone et la cible peut s'écrire comme suit, en utilisant le théorème de Pythagore :

$$d = \sqrt{h^2 + r^2} = \sqrt{h^2 + v^2(\tau - t_c)^2} \tag{1.4}$$

Résolution de l'équation du second degré pour déterminer τ en fonction de t, t_c, h, et v.

$$(c^2 - v^2)\tau^2 + 2(v^2 t_c^2 - c^2 t)\tau + c^2 t^2 - v^2 t_c^2 - h^2 = 0 \tag{1.5}$$

cela mène à la solution suivante :

$$\tau = \frac{c^2 t - v^2 t_c - \sqrt{h^2(c^2 - v^2) + v^2 c^2 (t - t_c)^2}}{c^2 - v^2} \tag{1.6}$$

La phase du signal aperçue au temps t est définie par :

$$\theta(t) = \theta(\tau) + \theta_0 \tag{1.7}$$

Où : $\theta(\tau)$ est la phase du signal à l'instant τ et θ_0 la phase initiale du signal.

$$\theta(t) = 2\pi f_0 \tau + \theta_0 \tag{1.8}$$

La fréquence instantanée est définie comme suit :

$$f(t) = \frac{1}{2\pi} \frac{d\theta(t)}{dt} \tag{1.9}$$

$$f(t) = \frac{1}{2\pi} \frac{d}{dt}(2\pi f_0 \tau + \theta_0) \tag{1.11}$$

$$f(t) = f_0 \frac{d\tau}{dt} \tag{1.10}$$

En remplaçant τ par son expression, on obtient la formule de la fréquence instantanée :

$$f(t) = f_0\left(\frac{c^2}{c^2 - v^2} - \frac{c^2 v^2 (t - t_c)}{(c^2 - v^2)\sqrt{h^2(c^2 - v^2) + c^2 v^2 (t - t_c)^2}}\right) \tag{1.11}$$

Courbe de la fonction :

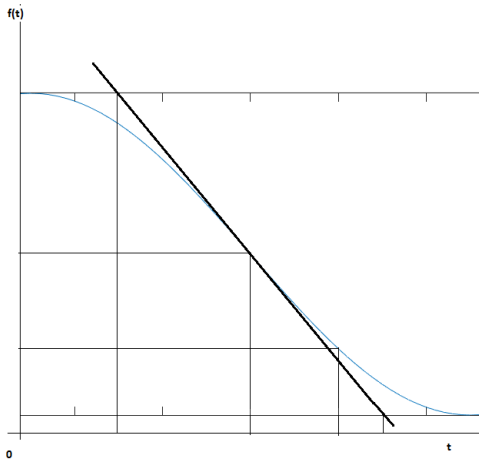


FIGURE 1.7: Courbe de la fréquence instantanée théorique

La fonction passe par un point d'inflexion représentant la fréquence centrale f_0 et le temps de passage, les tangentes horizontales représentent f_0 plus ou moins Δf_0 représentant le décalage dû au doppler.

1.3 État de l'art

1.3.1 Système de localisation acoustique utilisant un seul microphone

Dans le cas où le mobile possède un spectre de bruit tonal et se déplace à une vitesse et altitude constantes, il ya plusieurs méthodes pour obtenir ses paramètres de mouvement. Ces méthodes se basent sur l'effet Doppler relayé par un seul capteur acoustique. [5] et [6] utilisent un seul récepteur situé directement au-dessous du mobile pour obtenir la vitesse, l'altitude et la fréquence du signal source à partir de l'évolution temporelle de la fréquence instantanée.

1.3.2 Système de localisation acoustique utilisant un réseau de microphones

Les premières approches datent de la seconde guerre mondiale et elles sont basées sur la formation de voies conventionnelle (ou beamforming) dont le principe est de réaliser un balayage mécanique ou électronique de l'espace par le réseau de capteurs. Le balayage consiste simplement à effectuer une rotation de l'antenne autour de son axe, mécaniquement ou électroniquement exploitant la réponse des capteurs.

Une simple méthode de localisation acoustique est d'estimer la différence de temps d'arrivée d'un signal sonore entre deux microphones. La différence de temps d'arrivée permet d'estimer l'angle d'arrivée. En combinant les données reçues par les deux microphones et en faisant une triangulation, on calcule la distance de la source sonore par rapport aux microphones.

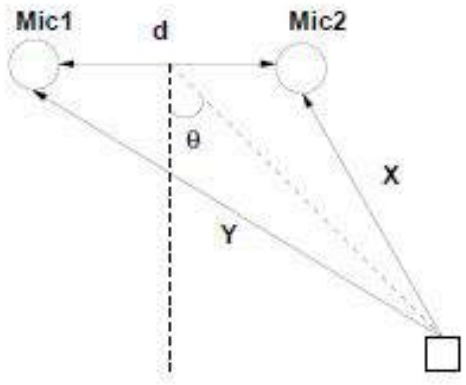


FIGURE 1.8: Détermination de l'angle d'arrivée

[7] évalue la performance d'un réseau sphérique de microphone pour obtenir les paramètres de mouvement d'hélicoptères, jet et d'avions à hélices qui voyagent le long d'une ligne droite à une vitesse et hauteur constantes

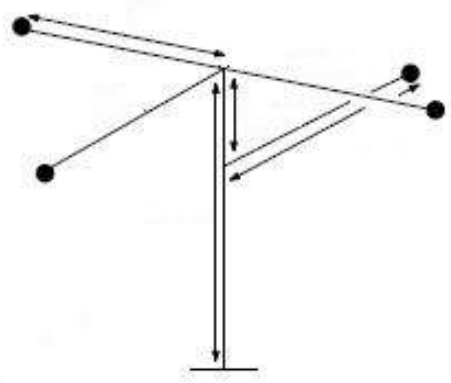


FIGURE 1.9: Détection acoustique utilisant un réseau sphérique de microphones

1.3.3 Détermination de la fréquence instantanée d'un signal sonore

L'effet Doppler a été significativement exploité dans l'estimation passive des paramètres de trajectoire : vitesse, fréquence et l'altitude d'un mobile en mouvement en ligne droite dont la vitesse est constante.
La base du traitement du signal est la transformée de fourrier, elle permet de décomposer un signal en une somme de ses composantes fréquentielles et donne la puissance de chacune d'entre elles, cependant,dans le cas où l'on a un effet Doppler, la transformée de Fourier standard ne permet pas d'accéder à l'information fréquence en fonction du temps. Par ailleurs, une alternative s'offre à nous, et qui est le passage vers l'analyse temps fréquence, constituant un outil adapté pour les signaux non stationnaires. On peut citer la transformée de Fourrier à court terme qui n'est autre que la transformée de Fourrier standard appliquée dans une fenêtre glissante dont la largeur vérifie la condition de stationnarité du signal dans cette même fenêtre. Cependant, cette approche ne convient pas pour notre application, car si l'on réduit le temps d'observation c'est à dire la largeur de la fenêtre, on perd en résolution fréquentielle, en effet, la résolution est déterminée par la largeur de la fenêtre d'analyse.
Cette résolution suit le principe d'incertitude de Heinsenberg :

$$\triangle f \triangle t \geq 4\pi \tag{1.12}$$

On peut aussi citer la distribution temps-fréquence, Wigner-Ville qui peut être utilisée pour estimer la fréquence instantanée du signal produit par le passage d'une cible acoustique. La méthode des distributions temps-fréquence a l'avantage d'être précise mais présente des inconvénients dans le cas d'une application temps réel. En effet, le traitement est lourd et demande par conséquent un temps de calcul plus long. On peut tout de même utiliser une méthode plus directe, qui est beaucoup plus rapide, cette méthode est appelée méthode de la différence de phase, elle consiste à dériver la phase du signal analytique. Le signal analytique est obtenu en transformant le signal aperçu, il définie d'une façon unique la phase du signal et ne présente pas d'artefacts (interaction entre fréquences positives et négatives).

1.3.3.1 Méthode de la différence de phase

C'est la méthode la plus simple, en effet, celle ci consiste à dériver la phase du signal sonore.

La représentation mathématique du signal est la suivante :

$$s(t) = a(t)cos(\theta(t)) \tag{1.13}$$

$\theta(t)$: phase du signal

$a(t)$: amplitude du signal

On passe à la transformée d'Hilbert pour avoir un signal analytique :

$$z(t) = s(t) + jH(s(t)) \tag{1.14}$$

Ainsi on réécrit :

$$z(t) = A(t)exp(\phi(t)) \tag{1.15}$$

Où $\phi(t)$ la phase et $A(t)$ l'amplitude du signal analytique.
Finalement la fréquence instantanée est déduite en dérivant l'argument de z(t).

$$f_i(t) = \frac{d\phi(t)}{dt} \tag{1.16}$$

Cette méthode offre l'avantage d'être rapide mais elle est très sensible aux bruits et elle peut ne pas être précise, sachant que cette étape peut influer sur l'exactitude des résultats de l'algorithme qui vient juste après pour l'estimation des paramètres.

1.3.3.2 Utilisation de représentations temps-fréquence

D'autres méthodes pour l'estimation de la fréquence instantanée d'un signal existent, on cite celles utilisées en cours d'analyse temps-fréquence, où la fréquence instantanée est déterminée par le moment d'ordre 1 de la distribution temps-fréquence.

$$W(t,f) = \frac{\int f W_z(t,f)\mathrm{d}f}{\int W_z(t,f)\mathrm{d}f} \tag{1.17}$$

Où W(t ; f) est la représentation de x(t) dans le domaine (t,f).

$$W(t,f) = \int x(t+\tau/2)\overline{x}(t-\tau/2)\mathrm{e}^{-2\pi f\tau}\,\mathrm{d}\tau \tag{1.18}$$

Cette procédure de calcul offre des résultats plus précis mais elle est bien plus gourmande en temps de calcul (complexité supérieure avec trois intégrales).

1.3.4 Estimation des paramètres de mouvement de la source acoustique

On a vu précédemment les méthodes d'estimation de la fréquence instantanée ; qui peuvent être utilisées dans la partie pré-traitement, la localisation en elle même conduit à l'estimation de paramètres de mouvement, on présente ici deux méthodes différentes.

1.3.4.1 Méthode d'optimisation ou algorithme de *Ferguson* [5]

Pour l'estimation des paramètres fréquence f0, la vitesse v et l'altitude h, on minimise la somme des carrés des écarts, entre l'expression de la fréquence instantanée f(t) estimée et la fréquence calculée en exploitant l'analyse temps-fréquence g(t).

Les paramètres estimés sont ceux qui minimisent l'expression suivante :

$$\sum_{j=1}^{N}(g_j(n) - f_j(n))^2 \tag{1.19}$$

$f_j(n)$ et $g_j(n)$ sont, respectivement, la fréquence instantanée provenant du modèle mathématique et la fréquence instantanée estimée par l'une des méthodes de calcul de la fréquence instantanée (méthode de la différence de phase ou méthode temps-fréquence).
On notera que cette méthode est une solution numérique et qui est lente.

1.3.4.2 Méthode analytique

La particularité de cet algorithme réside en son traitement. Contrairement a celui présenté dans la section précédente, qui fait appelle à des approximations et calculs numériques d'estimations avec la méthode des moindres carrés, cet algorithme se base sur des équations permettant la détermination de façon analytique des paramètres de mouvement. Cela offre une estimation directe, avec seulement quelques équations mathématiques [8].

Étapes de calcul de l'algorithme :

Étape 1 :
Le temps est estimé à partir de pic de la dérivée de la fréquence instantanée, ensuite la vitesse v est estimée par une équation mathématique. Pour avoir une bonne résolution, on calcule la moyenne des vitesses trouvées.

$$v = \sqrt{\frac{(c^2(t-t_c)f'(t_c))^2(f(t_c)-f(t))^2}{((t-t_c)f'(t_c))^2 - ((f(t_c)-f(t))^2)f(t_c)^2}} \tag{1.20}$$

Etape 2 :
La fréquence fondamentale du son est estimée par l'équation suivante :

$$f_0 = f(t_c)(1 - \frac{v^2}{c^2}) \tag{1.21}$$

Etape 3 :
L'altitude est estimée par l'équation suivante :

$$h = \frac{f(t_c)v^2}{(f'(t_c)(c^2 - v^2)^{1/2}} \tag{1.22}$$

Au final, on a une solution analytique au problème de localisation acoustique dès lors qu'on observe le phénomène Doppler exposé précédemment.

On se basera sur cette méthode pour l'élaboration puis l'implémentation du modèle de traitement.

1.4 Déduction du modèle structurel à partir de l'algorithme choisi

L'application se décompose en trois parties :

— Acquisition

— Pré-traitement (Calcul de la Fréquence instantanée exploitation de la méthode directe).

— Traitement (Détermination des paramètres, exploitation de la méthode analytique).

On déduit un schéma ou modèle structurel englobant les trois parties constituant l'application :

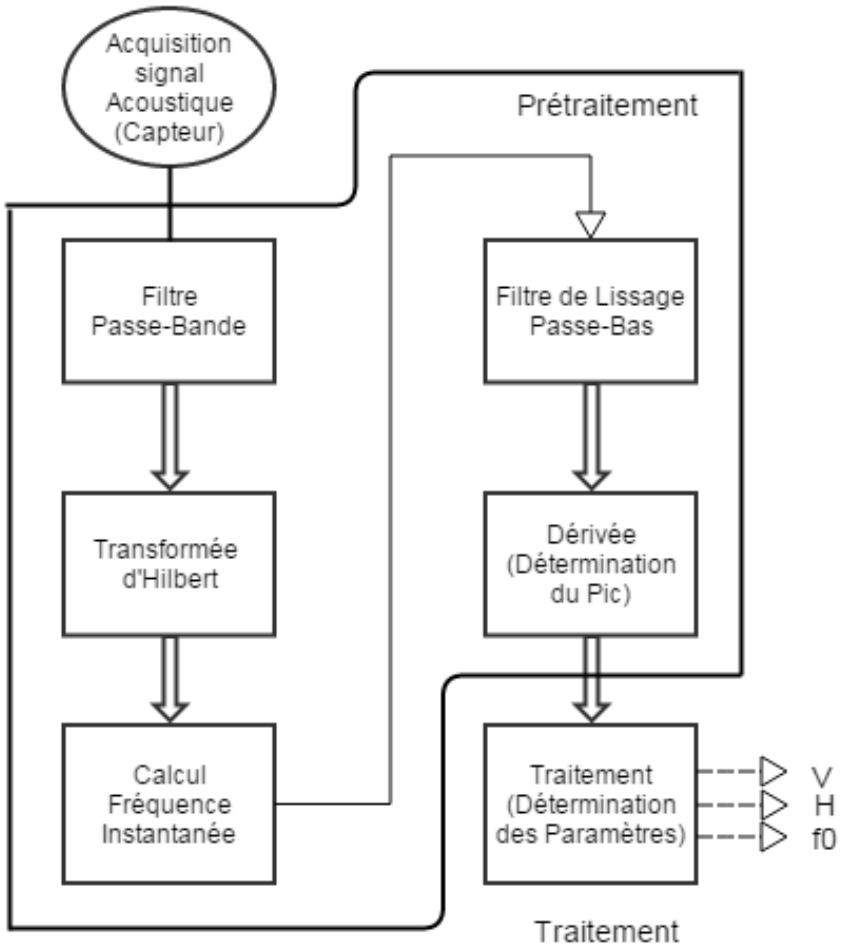


FIGURE 1.10: Schéma structurel de l'algorithme utilisé

- La partie transduction est réalisée par le moyen d'un capteur acoustique, ici, un microphone.

- Un bloc filtre passe bande pour filtrer les fréquences qui ne nous intéressent pas et travailler qu'avec la gamme de fréquence de la cible voulue.

- Un bloc pour estimer la fréquence instantanée.

- Un bloc filtre passe bas.

- Un bloc pour le calcul de la dérivée de la fréquence instantanée après filtrage.

- Un bloc pour estimer le pic et sa position.

- Le bloc final, estimation des paramètres : Vitesse, Altitude, Fréquence et temps Tc.

1.5 Plateforme d'expérimentation en champ libre

Dans le cadre de l'étude de l'application développée pour la localisation de sources sonores, une plateforme en champ libre a été conçue afin d'approcher le cas réel et ainsi de pouvoir comparer les résultats à ceux de l'analyse théorique. La structure mécanique à été montée au niveau de la terrasse du département d'électronique.(Annexe A)

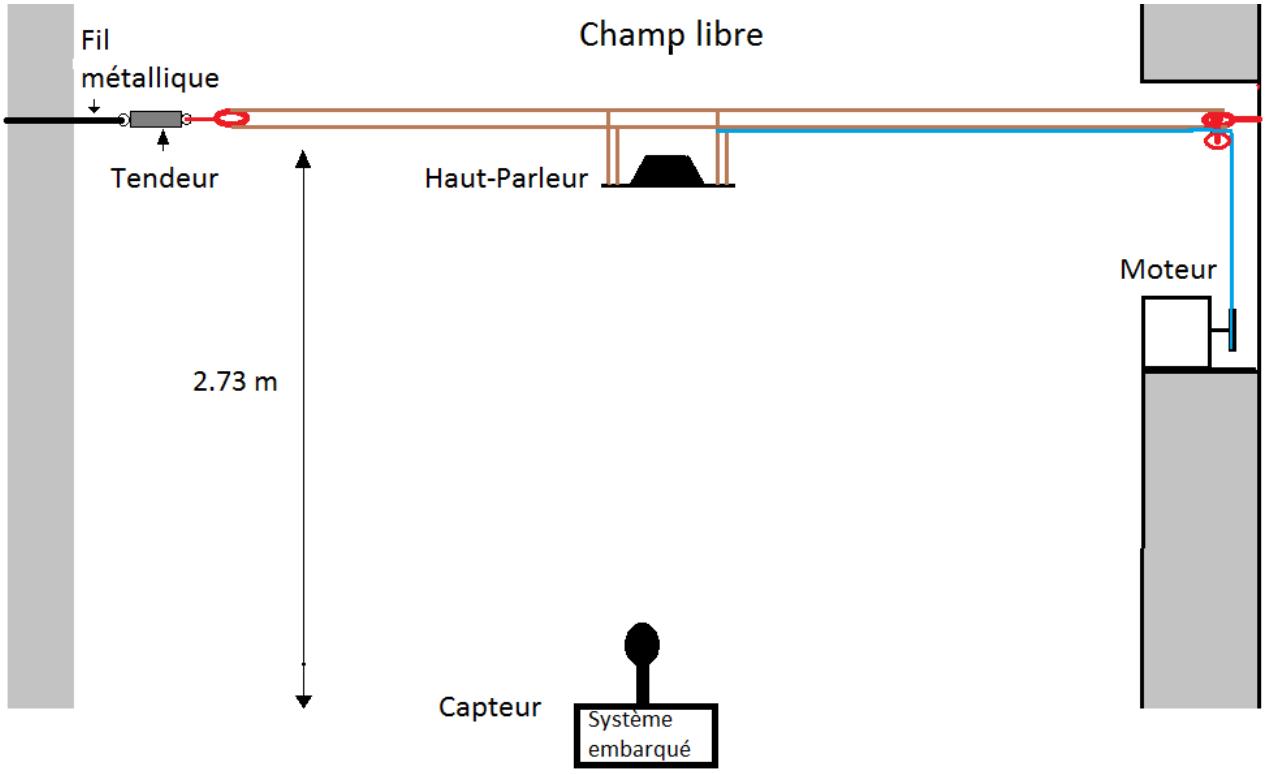


FIGURE 1.11: Structure de la plateforme d'expérimentation

1.6 Capteur réseau

L'idée générale est de concevoir un système dédié à l'estimation des paramètres de mouvement d'une source acoustique.

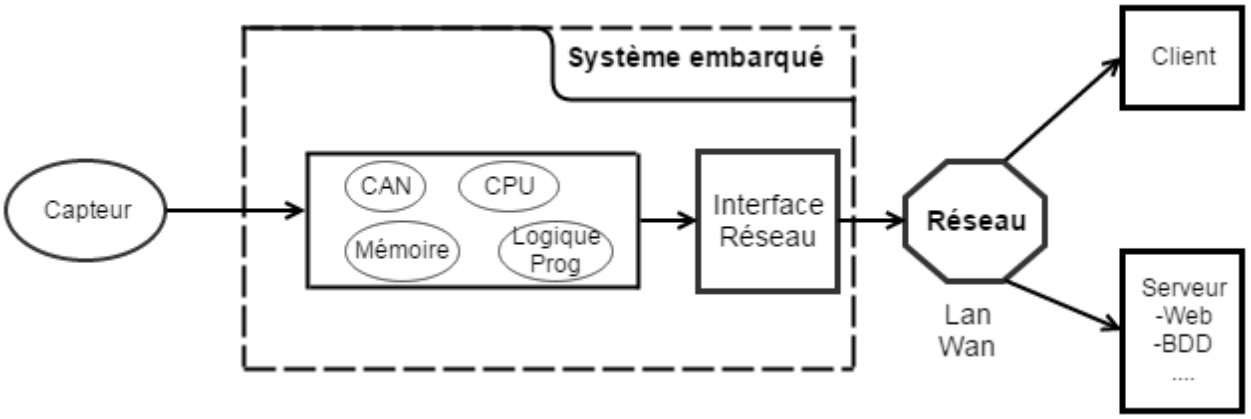


FIGURE 1.12: Architecture générale de l'application

L'application travaille avec en entrée un module capteur acoustique (Microphone) pour l'acquisition sonore, celui ci sera suivit par l'interface du cœur de l'application qui est un système embarqué, au sein duquel sera implémenté notre modèle de traitement (nous discuterons du choix de la carte "cible technologique" et de la méthodologie d'implémentation dans les chapitres qui suivent) en sortie diverses interfaces pourront aussi être exploitées pour l'accès réseau à des fins de transmission ou de sauvegarde des résultats.

1.7 Conclusion

On a introduit dans ce chapitre les notions globales concernant la localisation acoustique de sources sonores ainsi que l'algorithme utilisé dans notre application. On a également exposé la réalisation d'une plateforme d'expérimentation émulant le passage d'une source acoustique, dans le but de valider le prototypage dans des conditions réelles. Dans le chapitre suivant, nous présenterons les méthodologies de conception existantes et nous détaillerons l'approche de conception choisie de notre application.

Chapitre 2

Méthodologie de conception

Sommaire

Nous verrons dans ce chapitre les méthodes qui dictent et encadrent l'implémentation d'un algorithme, notre capteur sera considéré comme étant un système embarqué nous chercherons à aboutir à une méthodologie pour réaliser l'adéquation architecture algorithme (ou AAA).

2.1 Adéquation Algorithme Architecture (AAA)

Le développement croissant des systèmes embraqués (SoC), des outils d'implémentation et de conception ont eu un impact réel sur les méthodologies de conception.

Traditionnellement les étapes de conception-implémentation, validation logicielle et matérielle s'effectuaient de manière séparée (séparation entre algorithme et architecture). La tendance actuelle est la jonction de ces deux aspects afin d'offrir une approche globale et formalisée pour prendre en compte simultanément ces deux dernières. Ceci permet de faire intervenir les tests et validations très tôt dans le flot de conception. [9]

L'adéquation algorithme architecture (AAA) est l'étude des interactions algorithme architecture et la détermination des jalons à suivre pour aboutir à une implémentation optimale

(répond au mieux aux spécifications) et prenant en compte les contraintes(temps-réel, cout...).

Enfin, elle offre la possibilité d'améliorer les techniques de prototypage et d'aborder de manière plus claire la conception conjointe logicielle-matérielle (co-design).

2.2 Méthodologies en systèmes embarqués

2.2.1 Approche Top-down et Bottom-up

La conception des systèmes embarqués peut se faire par plusieurs approches. Parmi ces approches, nous citons les méthodes "Top-down et Bottom-up". La conception "Top-down" commence par une description très abstraite puis les détails seront enrichis durant le développement pour arriver enfin à des solutions spécifiques.

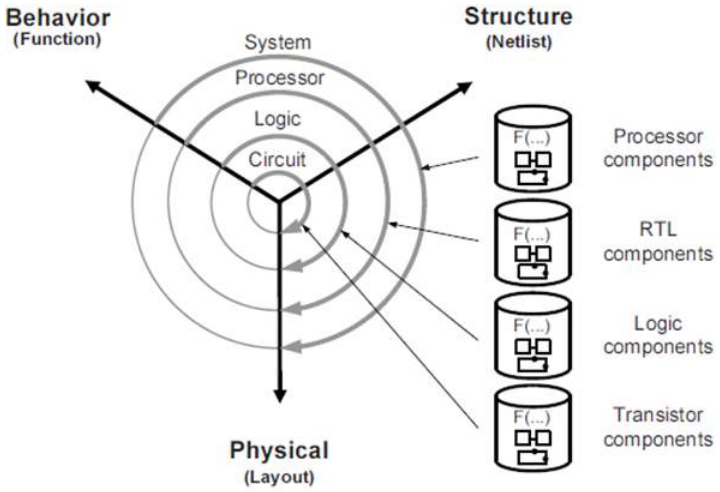


FIGURE 2.1: Approche Top-Down

La conception "Bottom-up" commence par un assemblage spécifique des petits composants pour obtenir un système complet.

Ce processus est basé à la fois sur l'expérience du designer qui doit intégrer les différentes contraintes, et sur des modèles équivalents qui vont rendre compte du comportement de l'équipement dans différents environnements. Il est possible de faire collaborer ces deux approches pour obtenir une médiane assemblage et une configuration spécifique.

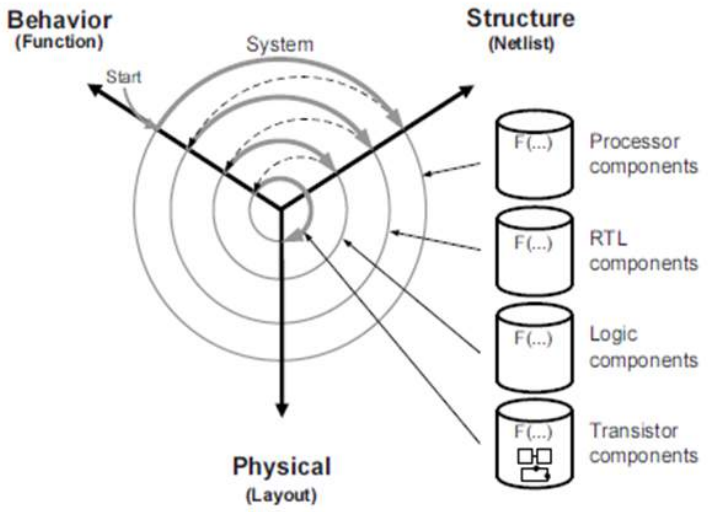


FIGURE 2.2: Approche Bottom-Up

2.2.2 Le co-design

Le cout et la performance étant des critères essentiels pour la mesure des exigences d'une conception, il est important d'étudier l'approche à adopter, une réalisation logicielle peut être préférée à une autre pour des raisons d'évolution et de cout, d'un autre coté les réalisations matérielles sont dédiées aux fonctionnalités nécessitant des circuits spécialisés ou des performances élevées.

Dans la conception traditionnelle la définition de l'architecture est suivie par le découpage des tâches entre la réalisation du matériel et du logiciel. La réalisation du matériel se traduit par la réalisation d'une description du système en utilisant un langage de description matériel tel que VHDL ou Verilog, puis par une réalisation de la synthèse et la génération des circuits intégrés en utilisant des outils de CAO. La réalisation du logiciel se traduit par l'écriture du code qui va être compilée et exécutée sur des processeurs d'usage général.

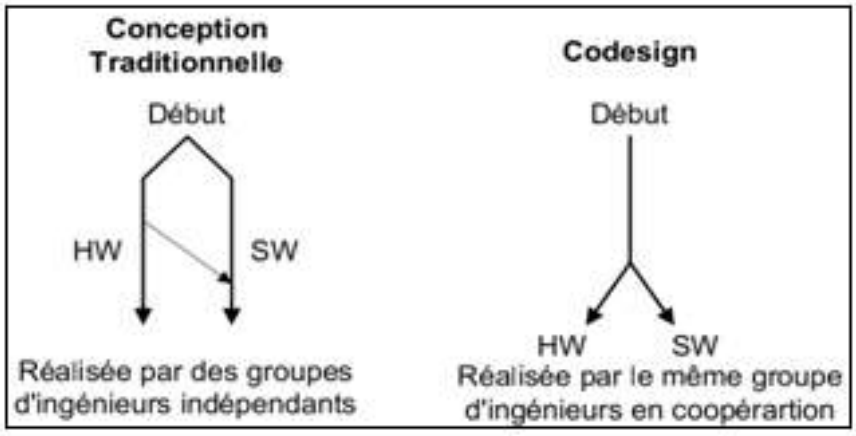


FIGURE 2.3: Le Co Design

On définit une conception mixte, matérielle et logicielle, dans laquelle la réalisation du logiciel et du matériel interagissent ensemble et à chaque étape de conception, elles réalisent l'intégration et le test des spécifications.

Le co-design permet de repousser loin dans la conception du système les choix matériels à faire contrairement à l'approche classique où les choix matériels sont faits en premier lieu [10].

2.2.3 Conception fonctionnelle

La méthode de conception fonctionnelle est une approche indépendante de la technologie. L'objectif est de décrire les fonctions par un algorithme séquentiel à partir des spécifications et non pas par assemblage de composants.

2.2.4 Conception architecturale

La conception architecturale est une approche dépendante de la technologie. L'objectif de cette étape est de choisir une architecture répondant aux exigences imposées par l'étape de spécification, cette conception sera vérifiée et validée par simulation.

On cite quelques types d'architectures de conception :

— Cricuit analogique.
— Système à base de CPU, uC.
— Circuit reprogrammable FPGA circuit spéficique ASIC, DSP.
— Système sur Puce (SoC).

Des deux paragraphes précedent on peut déduire une méthode globale de conception : " Méthode globale de conception "

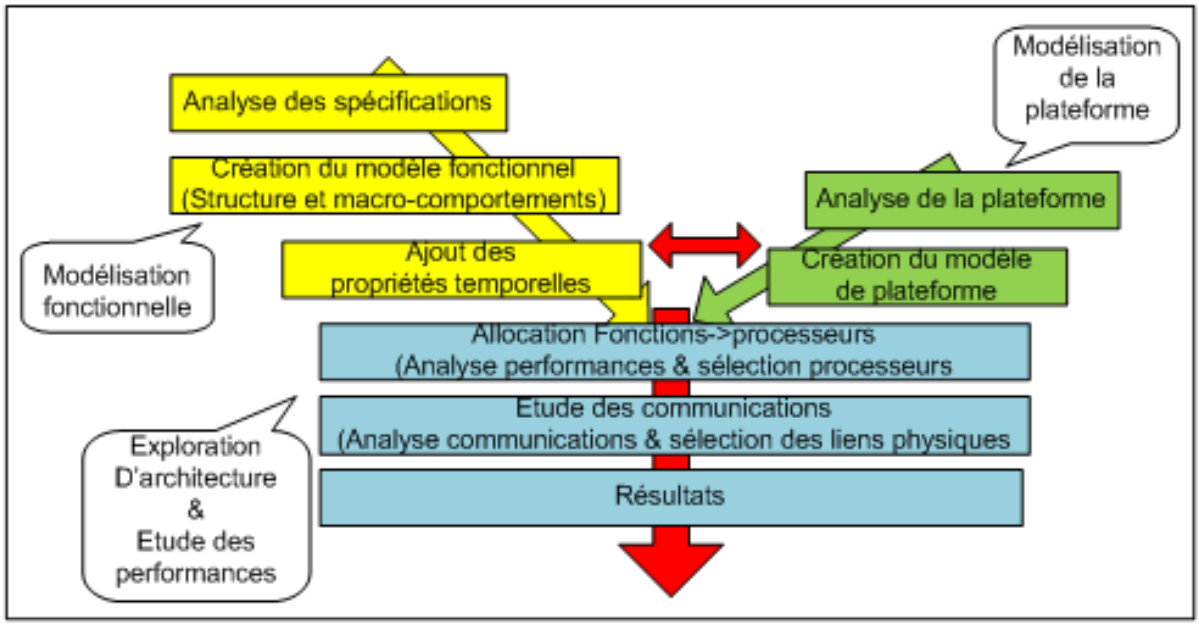


FIGURE 2.4: Méthode globale de conception

2.2.5 Prototypage rapide

De nos jours les outils de développement jouent un rôle essentiel à l'élaboration de nouveaux produits et leur mise sur le marché, le Prototypage Rapide est une technique qui tend à se généraliser et devenir incontournable quand il s'agit d'accélérer le market time, le temps de développement, la maintenance, la fiabilité et la sureté de fonctionnement de la conception, pour cause le prototypage rapide offre une transition simple depuis une conception système vers une implémentation sur cible.

L'élément clé du prototypage rapide est la génération automatique de code (soft exemple : C/C++ ou bien hard : HDL).
Les ingénieurs ont la possibilité de se concentrer uniquement sur le modèle de l'algorithme afin de l'améliorer.
Cette méthode offre l'avantage de générer du code optimisé pour la cible via l'utilisation d'un environnement de compilation croisée ainsi on s'éloigne de la programmation bas niveau et cela constitue un gain de temps certain.

2.3 Model-Based Design

Le "Model-Based Design" permet la planification et la mise en œuvre (fonctionnelle et architecturale) d'une conception, au lieu d'étudier isolément les étapes (de simulation, stynthèse ou encore la vérification) cette approche propose une combinaison en une méthodologie explicite s'inscrivant dans une conception basé sur le modèle, de l'abstraction a l'architecture et du concept à la réalisation ainsi le processus de conception gravite autour d'un modèle

systémique qui découle des spécifications et aboutit à l'implémentation.

On décompose le "Model-Based Design" en cinq étapes fondamentales qui constituent le flot du modèle.

Un autre avantage d'une telle approche est la vérification en temps-réel et le monitoring en boucle (software in the loop, hardware in the loop ...) [11].

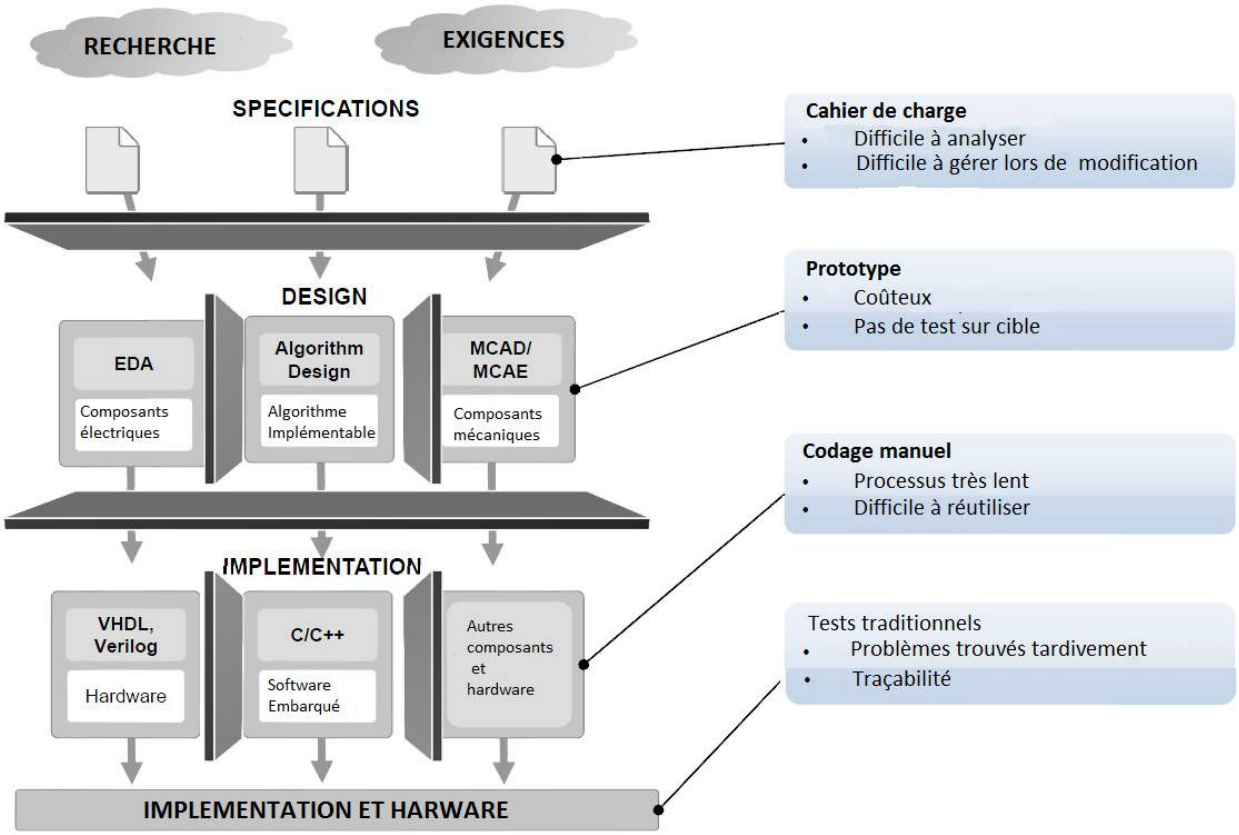


FIGURE 2.5: Méthode de développement traditionnelle

2.3.1 Flot du Model-Based Design

Les principales étapes dans l'approche Model-Based Design sont : [12]

Etape 1 : Formulation du problème

— Description du problème utilisant un langage simple et dénoué de terminologies techniques. Ceci constituera le modèle de départ pour le projet et représente une référence pour les développeurs et collaborateurs. A partir de cette formulation on tire les exigences de conception.

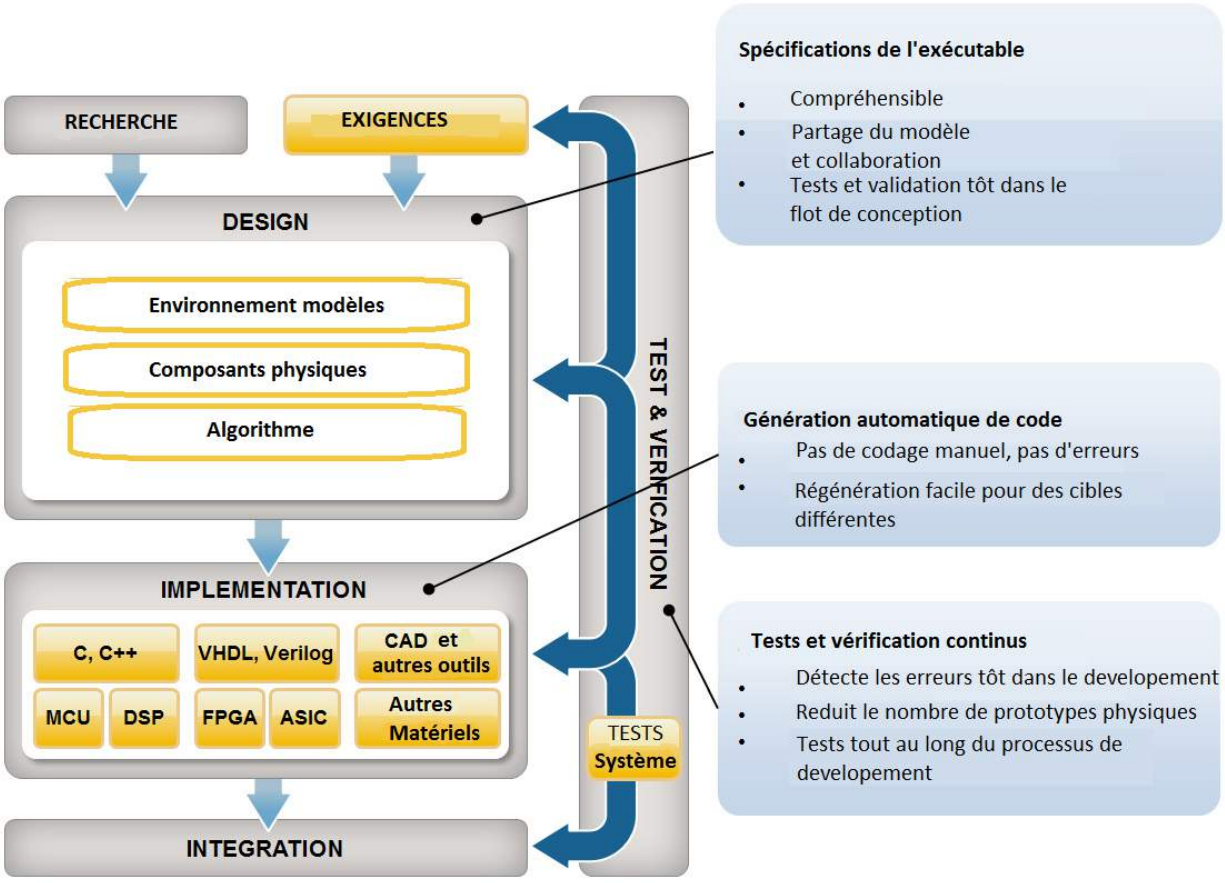


FIGURE 2.6: Développement Model-Based Design

Etape 2 : Modélisation du système

— La modélisation est axée sur des données ou sur la base de principes premiers de l'application. La modélisation basée sur les données utilise des techniques telles que l'identification système (utilise des méthodes statistiques pour construire un modèle mathématique de systèmes dynamique, à partir de données mesurées). Avec le système d'identification, le modèle est construit à partir de l'acquisition et le traitement de données brutes provenant du monde réel. Quant à la modélisation basée sur les premiers principes de l'application, elle s'identifie par la création de diagrammes modèles implémentant des équations régissant la dynamique dy système.

Etape 3 : Analyse et synthèse

— Le modèle mathématique conçu précédemment est utilisé pour identifier les caractéristiques dynamiques du système.
— Réglage et configuration du système.

Etape 4 : Test et validation

— La réponse du système est étudiée par une simulation dite off-line c'est à dire sur une machine hôte. La simulation permet de vérifier si les exigences sont vérifiées ainsi que de trouver immédiatement les sources de problèmes dans le cas où il y aurait des erreurs.
— Une simulation temps réel peut être faite par la génération automatique de code pour l'architecture cible, ce code peut être déployé sur la cible afin de tester son bon fonctionnement.

Etape 5 : Implémentation du code sur la cible [13]

— Génération automatique de code.
— Tests et vérifications.
— Reconfiguration du modèle.

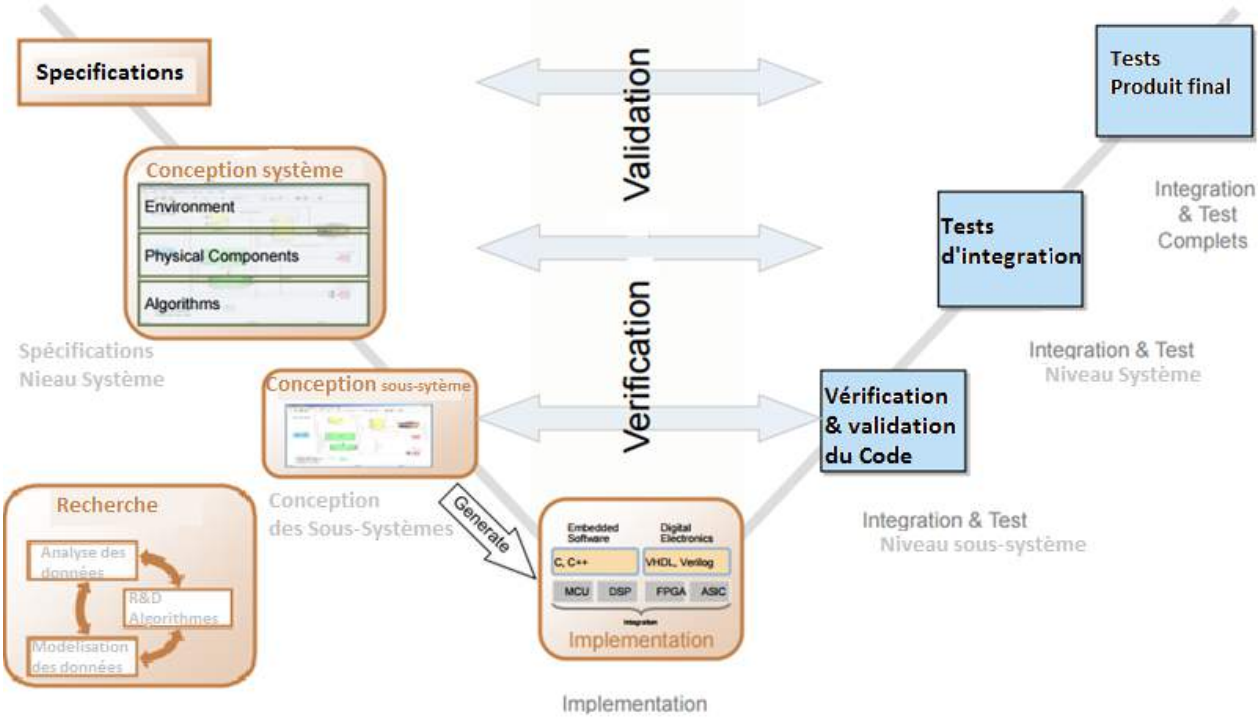


FIGURE 2.7: Diagramme en V du MBD

2.4 Motivations du choix de Model-Based Design

2.4.1 Approche classique confrontée au Model-Based Design

Dans un processus classique de conception LRU (Least Recently Used), les ingénieurs recueillent des spécifications à partir de plusieurs sources et les combinent afin d'élaborer un cahier des charges sur papier, qui aidera l'équipe de conception décentralisée à produire une conception détaillée.

Cette équipe analyse une série de concepts différents par le biais de prototypes de simulation ou matériels, elle contrôle la conformité à ces spécifications puis elle apporte les modifications appropriées. Après avoir mis au point une conception acceptable, elle transmet celle-ci pour implémentation à une autre équipe qui exécute les tests de vérification et de validation, puis de conformité si le produit doit respecter certaines normes spécifiques. Comme les tests se déroulent à la fin d'un processus en plusieurs étapes faisant appel à des équipes différentes, les erreurs introduites en phase de conception ne sont souvent détectées que bien plus tard. Leur correction onéreuse oblige la direction à prendre des décisions budgétaires difficiles. [14]

2.4.2 Détecter et corriger les problèmes dès le début du processus de conception

L'approche Model-Based Design débute par la création d'une spécification exécutable qui peut être liée aux spécifications initiales.

Cette spécification repose sur un modèle exécutable, utilisé et élaboré tout au long du processus.

Elle peut également intégrer des entrées et des sorties attendues, l'environnement de l'application et des objectifs de conception. Les ingénieurs peuvent vérifier que la conception respecte ces exigences tout au long du processus, et évaluer rapidement l'impact des modifications proposées.

Avec la génération de code automatique et les tests HIL (hardware-in-the-loop), les ingénieurs peuvent éliminer les erreurs introduites au cours de l'implémentation manuelle et accélérer la livraison du produit en générant le code qui servira aux tests, à la vérification et à la production finale.

2.4.3 Opérations de test et de vérification de manière continue

À l'aide d'un modèle, les ingénieurs peuvent tester et vérifier leur conception bien plus tôt qu'avec le développement traditionnel, pour lequel le lancement des tests nécessite une instanciation physique de la conception.

Ils peuvent ensuite élaborer ce même modèle tout au long du processus de conception, ce qui leur permet d'effectuer les opérations de test et de vérification de manière continue.

Avant de consacrer des ressources et des fonds à l'implémentation, à la conception ou aux tests physiques, les ingénieurs peuvent soumettre la conception à une analyse de couverture de modèle reposant sur des critères spécifiques afin de garantir la détection des erreurs dès le début du processus de conception.

Au cours des tests, les modèles ne sont pas limités à un environnement numérique ; ils peuvent également intervenir dans les tests portant sur les composants du système physique. Il peut s'agir par exemple des simulateurs de type (SIL, PIL, Processor-in-the-loop Software-In-The-Loop ...) développés en vu de tester l'interaction en temps réel des donnée avec les algorithmes de contrôle mis à jour, ou bien des tests HIL utilisés afin d'étudier les problèmes intermittents.

Après avoir déterminé la pertinence de leur conception algorithmique, les ingénieurs peuvent étudier les effets de l'exécution de ce même algorithme dans un environnement embarqué. Ils peuvent ajouter au modèle des détails d'implémentation.

Dans la mesure où la cible de l'implémentation est spécifiée dans les étapes ultérieures du processus, il est possible de réutiliser une grande partie du travail de conception lors de la sélection d'une nouvelle cible. Par ailleurs, puisque le processus d'implémentation est automatisé, il est reproductible et ne dépend pas de la disponibilité de compétences spécifiques.

2.4.4 Division par 10 des délais de mise au point et de développement global

Il est possible de réutiliser les cas de test développés au début du processus de conception pour tester l'implémentation finale. Les opérations de test et de vérification de la conception par rapport aux spécifications s'inscrivent donc sur l'ensemble du processus, et non dans ses dernières étapes.

Avec l'approche Model-Based Design, les ingénieurs n'ont plus à attendre la fin du processus de conception pour effectuer les tests et contrôler que les spécifications sont respectées. Cette approche signifie que les ingénieurs peuvent effectuer les itérations de conception dans un environnement de modélisation fiable et moins coûteux, de manière à obtenir une implémentation opérationnelle dès le premier essai, avec un cycle de développement accéléré et un meilleur niveau de qualité.

Enfin un exemple de système réalisée par logiciel, à l'aide de l'approche Model-Based Design a mis en évidence qu'il fallait en moyenne 645 heures à un ingénieur possédant de longues années d'expérience dans le codage VHDL pour coder une forme d'onde de SDR (Software Device Radio) totalement fonctionnelle dans le cadre du flot de conception classique.

Un autre ingénieur ne possédant qu'une expérience limitée a pu terminer le même projet en moins de 46 heures avec l'approche Model-Based Design. [15]

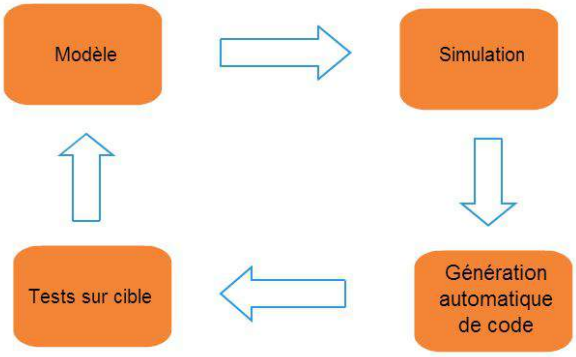


FIGURE 2.8: Flot design Model-Based

2.5 Conclusion

Dans ce chapitre on s'est intéressé aux bases théoriques que forment les méthodologies de développement d'un système embarqué et l'adéquation algorithme architecture, ceci nous a permis d'étudier puis d'élire le Model-Based Design comme méthode d'implémentation, la prochaine phase sera la mise en œuvre de cette dernière, ainsi dans le chapitre suivant nous verrons les outils qui permettent de concrétiser le Model-Based Design pour modéliser, valider, implémenter et vérifier le fonctionnement de notre application.

Chapitre 3

Conception du modèle de l'algorithme utilisé

Sommaire

Dans ce chapitre nous mettrons en œuvre la méthodologie choisie pour l'implémentation de l'application celle ci comporte trois caractéristiques majeurs qui sont le prototypage rapide et la conception basée sur le modèle ou "Model-Based Design" le tout sous un aspect de développement haut niveau, nous verrons ainsi comment utiliser les outils Mathworks pour la validation du modèle et la préparation au mieux pour l'étape d'implémentation et de déploiement sur cible.

3.1 Modèle Matlab

Comme cité précédemment dans le flot de développent la première étape de validation est la traduction du modèle fonctionnel vu en chapitre 1 vers un algorithme fonctionnel à partir duquel on se basera pour aller vers une implémentation sur cible toujours dans l'environnement Mathworks.

Matlab est un des outils les plus utilisés pour le calcul et ce, dans différents domaines techniques et scientifiques, il nous est permis d'exécuter des script, définir des fonctions et d'y faire appel.

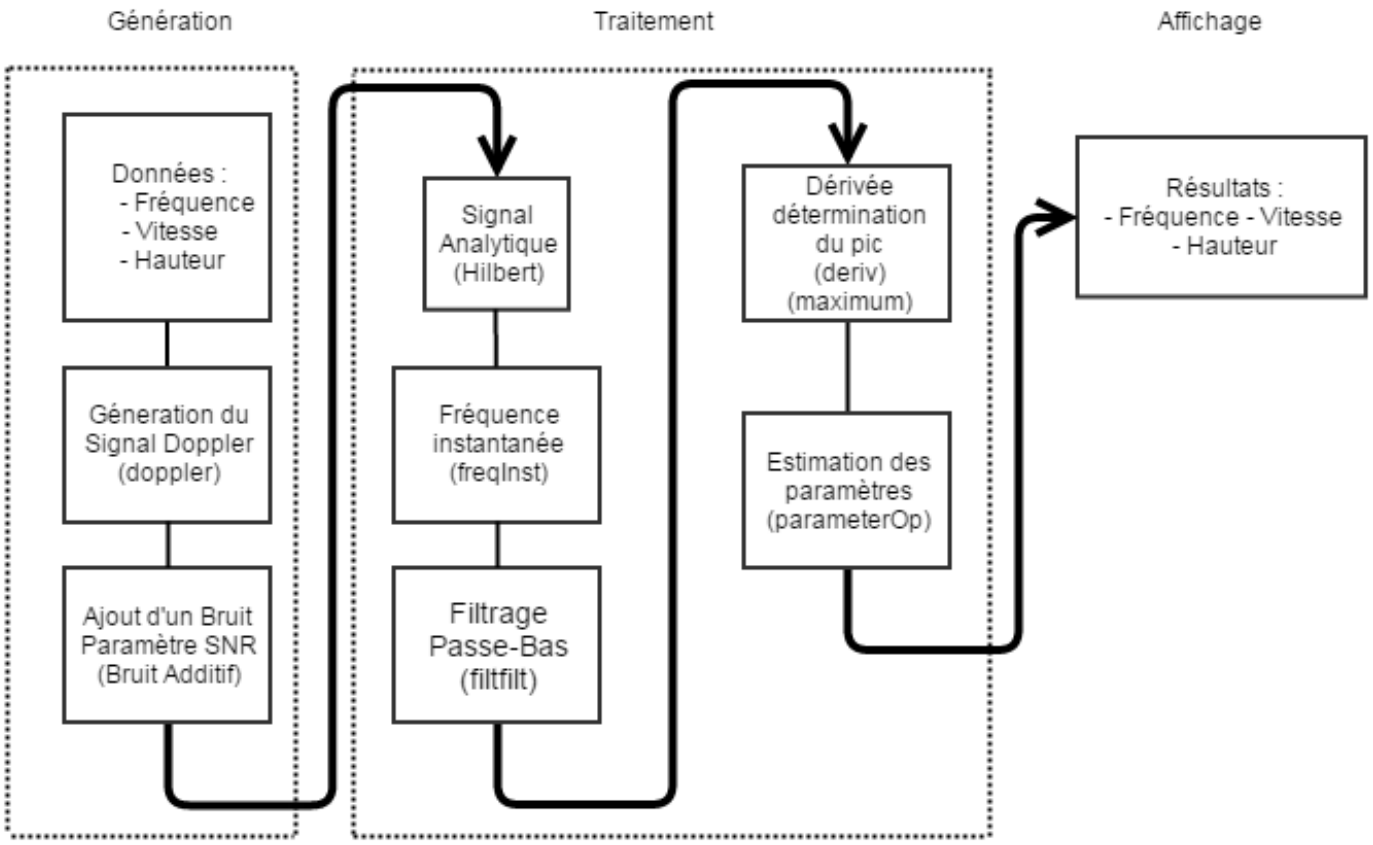


FIGURE 3.1: Organigramme

Ci dessus le modèle Matlab présenté sous forme d'organigramme, l'exécution est séquentielle, les blocs s'exécutent un à la fois jusqu'à avoir les résultats de l'estimation.

3.1.1 Validation sous Matlab

3.1.1.1 Validation du modèle avec un signal Doppler simulé

3.1.1.1.1 Generation du Doppler Tout d'abord on génére un signal Doppler avec la fonction *Doppler* [16] dont les paramètres sont : le rapport signal sur bruit SNR, la fréquence d'échantillonnage fs, la fréquence au repos f_0, la hauteur h et la vitesse v .

```
1    [fm,am,iflaw]=doppler(N,fs,f0,h,v);
```

La fonction Doppler donne en sortie f_m fréquence de modulation, a_m amplitude modulation et $iflaw$ la fréquence instantanée.

```
1    z = am.*fm + 10^(-SNR/20)*randn;
```

z est donc la simulation du signal reçu au niveau du capteur acoustique.

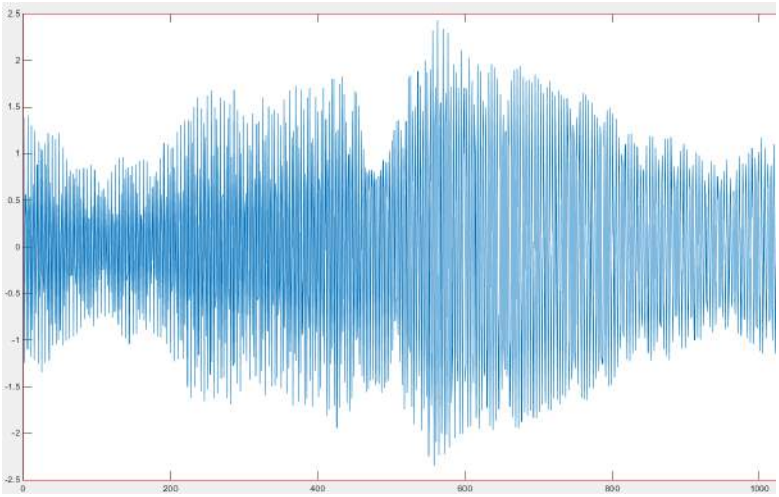


FIGURE 3.2: Illustration du signal simulé

3.1.1.1.2 Transformée d'Hilbert Toujours sous Matlab on passe à présent à la transformée d'Hilbert, afin d'obtenir le signal Analytique (On note que la fonction "hilbert" sous MATLAB correspond au calcul du signal analytique en lui même).

```
z=hilbert(z);
```

3.1.1.1.3 Estimation de la fréquence instantanée On utilise la fonction $instfreq$ [17] qui donne un vecteur contenant la FI normalisée et le vecteur temps qui lui est associé.

```
[fnormhat,t]=instfreq(z);
```

Où z est le signal analytique calculé précédemment.

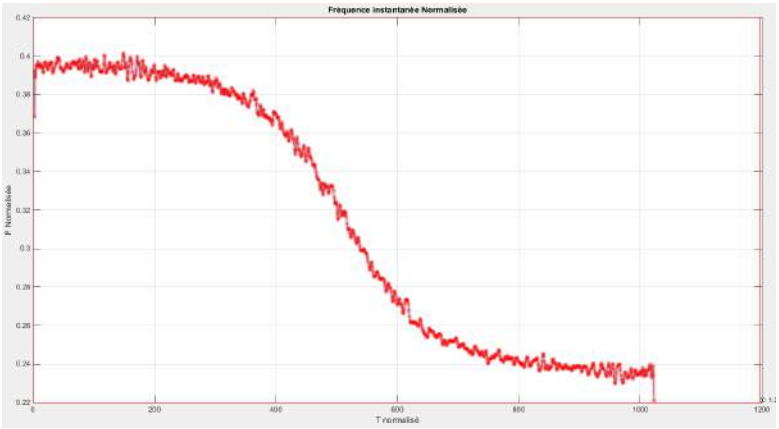


FIGURE 3.3: Fréquence instantanée illustration du Doppler

```
freq=fnormhat*fs;
t=((1:length(freq))./fs)';
```

On dénormalise

La courbe de la fréquence instantanée contient du bruit HF un filtrage passe bas va être effectuer.

3.1.1.1.4 Filtrage passe bas On utilise le filtre de Butterworth dont les paramètres sont

```
fc=0.8;
wn=[fc/(fs/2)];
[b,a] = butter(4,wn2,'low');
freqsmooth = filtfilt(b,a,freq(100:length(freq)));
```

Où *wn* est la bande passante, a et b les coefficients du filtre de Butterworth (a : le numérateur, b : dénominateur).

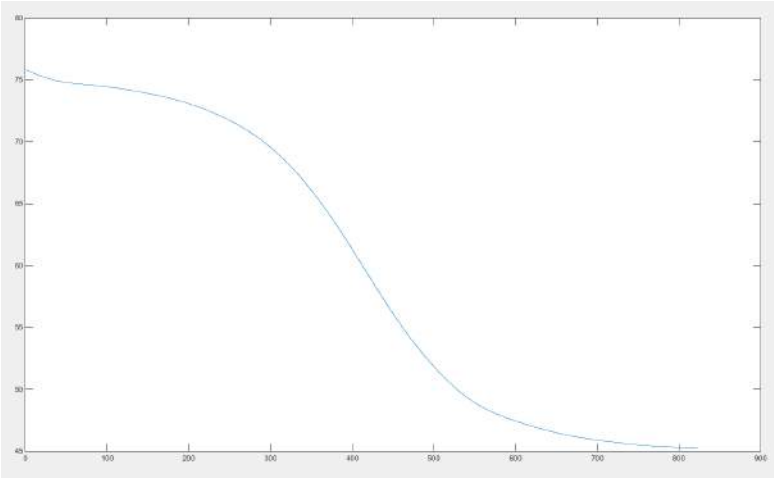


FIGURE 3.4: Fréquence instantanée filtrée

3.1.1.1.5 Détermination de $f0$ et tc Comme l'illustre la figure (freq(t)) vue au chapitre 1 $f(t)$ est une fonction décroissante ayant un point d'inflexion en (tc temps de passage, $f0$ fréquence au repos) ces paramètres seront exploité par la fonction Matlab *ParametreOp* pour le calcul de deux autres paramètres que sont (v, h).

On dérive donc la FI et on cherche le *pic*.

```
freqderiv=abs(deriv5(freqsmooth,fs));
```

On fait appel à la fonction deriv5, qui est un calcul de dérivée d'ordre 4.

Enfin, on obtient le pic (f0) et sa position (tc) à l'aide de la fonction *maximum*.

```
[peak,position]= maximum(freqderiv);
```

3.1.1.1.6 Estimation des paramètres de la source acoustique Pour l'estimation du h et v on utilise la fonction *parameterOp* celle ci nous donne un résultat direct. [18]

```
[v,f0,h,tc] = ParameterOp(freqsmooth,peak,position,t,fs);
```

3.1.1.1.7 Simulation validation sur Matlab On compare les résultats obtenu par calcul aux paramètres injectés dans la fonction *Doppler*.

```
Nombre de point N = 1024;
Frequence fondamentale f0 = 60;
Frequence d'echantillonnage Fs = 190;
Hauteur h = 70
Vitesse v = 90;
Rapport signal sur bruit SNR = 30 ;
```

Résultats :

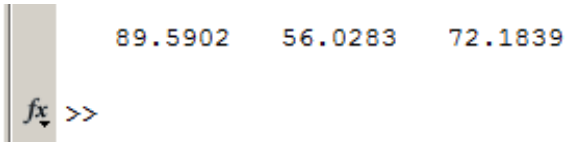


FIGURE 3.5: Validation sous Matlab

Les résultats sont correctes et très proches des vraies valeurs.
Erreurs relatives :

Paramètres	Valeurs expérimentales	Modèle Matlab	Estimation de l'erreur
Vitesse (m/s)	90	89.5902	0.4 %
Hauteur (m)	70	72.1839	3.12 %
Fréquence (Hz)	60	56.0283	6 %

TABLE 3.1: Calcul d'erreurs Modèle Matlab

On passe à présent à l'élaboration d'un modèle basé sur l'algorithme vu ci dessus afin d'avoir un modèle fonctionnel pour l'application.

3.2 Modèle Simulink

Dans cette partie nous traiterons les étapes qui permettent la projection de l'algorithme sur un modèle simulink. On fera la validation avec le signal *Doppler* simulé puis avec un signal réel.

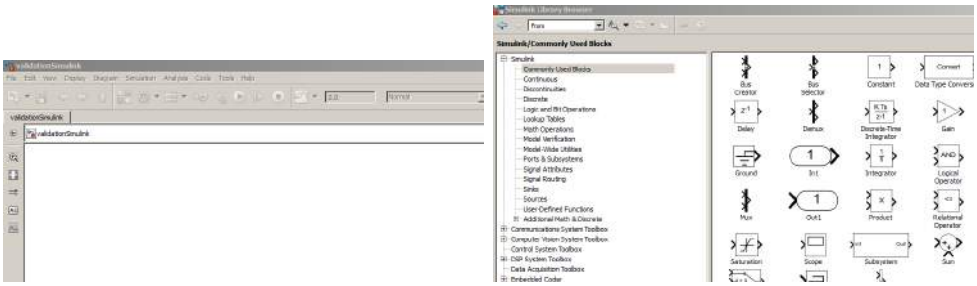


FIGURE 3.6: Simulink

3.2.1 Environnement Simulink

Simulink est une plate-forme de simulation et de modélisation de systèmes. Il fournit un environnement graphique et un ensemble de bibliothèques contenant des blocs de modélisation qui permettent le design précis, la simulation, l'implémentation et le contrôle de systèmes de communications et de traitement du signal. Simulink est intégré à Matlab, fournissant ainsi un accès immédiat aux nombreux outils de développement algorithmique, de visualisation et d'analyse de données de Matlab.

En se basant sur le schéma proposé à la fin du chapitre 1, nous allons pouvoir déployer un modèle fonctionnel sous Simulink à l'aide des blocs Simulink et de fonctions de l'algorithme.

3.2.2 Modèle sous Simulink

Le modèle sous Simulink est le suivant :

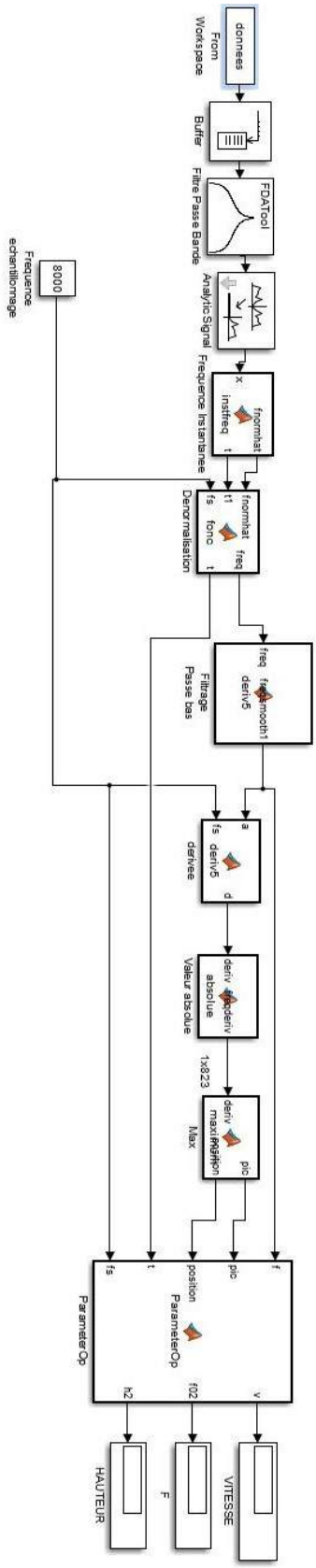


FIGURE 3.7: Modèle sous Simulink

Explication du comportement temporel du modèle :

Le bloc n(i=0) s'exécute à l'instant t=t(0) alors que les blocs n(i) pour i> 1 sont off. Le bloc n(i+1) s'exécute à l'instant t=t(2), n(i) continue de s'exécuter à l'instant t=t(2). Le bloc n(i+2) s'exécute à l'instant t=t(3), n(i) et n(i+1) continuent de s'exécuter à l'instant t=t(3) ainsi de suite.

3.2.2.1 Import de signaux

Ce paragraphe illustre l'intégration du signal à l'environnement Simulink.

3.2.2.1.1 Signal Doppler On génere le signal *Doppler*, par la suite on calcul sa transfomrée de Hilbert, on obtient le vecteur z sur le *workspace* de Matlab.

```
rr=[0 real(z)'];
ii=[0 imag(z)'];
```

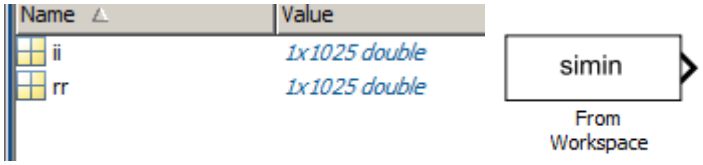


FIGURE 3.8: Workspace

On fait appel au bloc *From Workspace* pour avoir le vecteur contenant le signal sur Simulink.

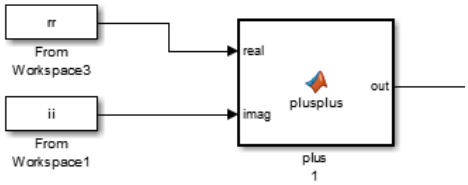


FIGURE 3.9: Acheminement du signal vers Simulink

Remarque : On prend séparément la partie réelle et imaginaire de z, un simple bloc qui fait l'addition et multiplie par l'imaginaire i nous permet de reconstituer le signal analytique.

3.2.2.2 Deploiement de l'algorithme sur Simulink (Signal Simulé)

3.2.2.2.1 Integration du code matlab sur Simulink Pour intégrer du code Matlab provenant de l'algorithme vu dans les parties précédentes, on utilise les blocs "user-Defined Functions" de la bibliothèque Simulink.

FIGURE 3.10: User defined function

3.2.2.2.2 Bloc Fréquence Instantanée puis Normalisation Ce bloc englobe deux sous systèmes, il a pour entrée le signal analytique et pour sorties la fréquence instantanée FI en plus du temps correspondant.

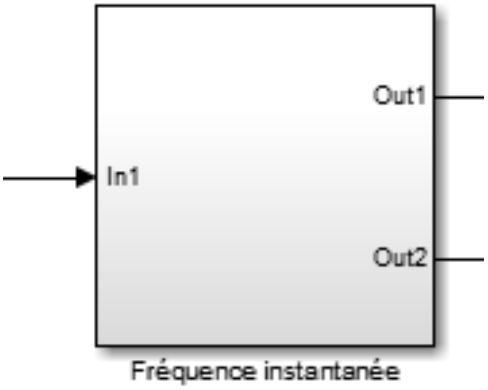


FIGURE 3.11: Fréquence instantanée

On ouvre le bloc.

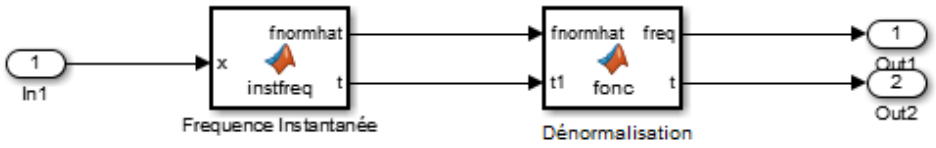


FIGURE 3.12: Sous système FI

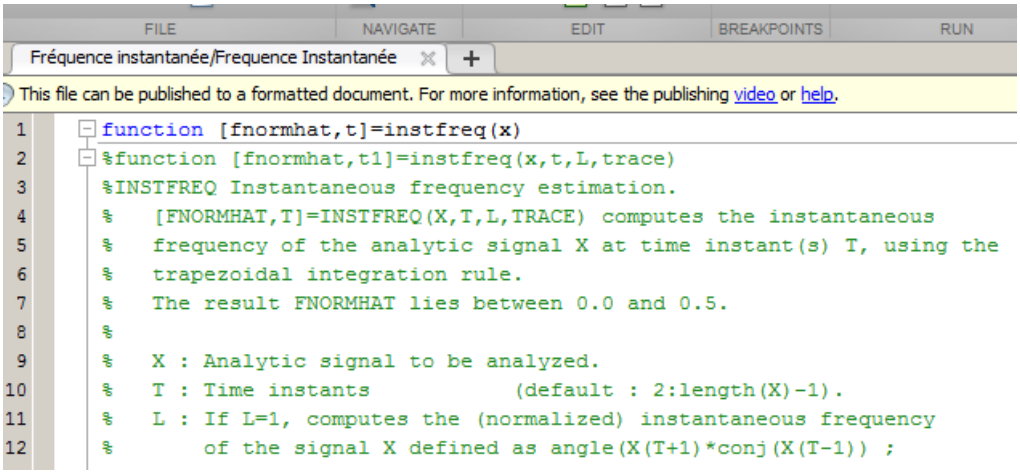


FIGURE 3.13: Fréquence Instantanée

La calcul de la FI est effectué avec le code de la fonction $instfreq$.

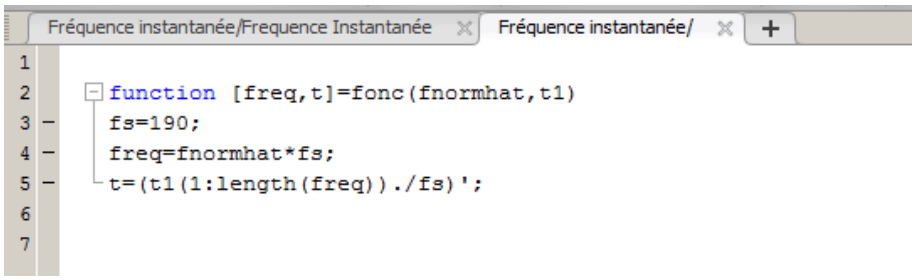


FIGURE 3.14: Dénormalisation

Le second bloc fait la dénormalisation.

3.2.2.2.3 Bloc Filtre Passe Bas Ce bloc effectue le lissage afin de réduire les fluctuations sur la courbe de la FI.

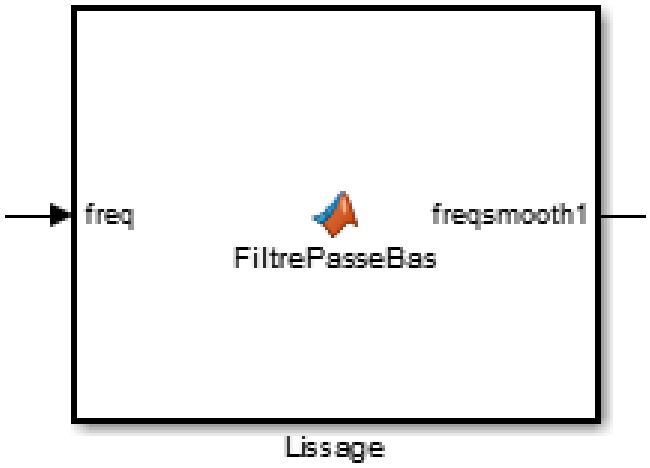


FIGURE 3.15: Filtre Passe Bas

```
function freqsmooth1=FiltrePasseBas(freq)
a=[1 -3.870384696880150 5.619488139332452 -3.627511720749895 0.878413957351279];
b=[3.549408554223699e-07 1.419763421689479e-06 2.129645132534219e-06 1.419763421689479e-06 3.549408554223699e-07];
freq=freq(100:length(freq)-100);
freqsmooth1=filtfilt(b,a,freq);
```

FIGURE 3.16: Code filtre passe bas

C'est un filtre du type *Butterworth*, On introduit les coefficients (calculés précédemment, à savoir, numérateur a et dénominateur b).

3.2.2.2.4 Bloc Dérivée et détermination du Pic ($f0$, tc) Le bloc suivant effectue deux fonctions : dérivation puis détermination du max et sa position.

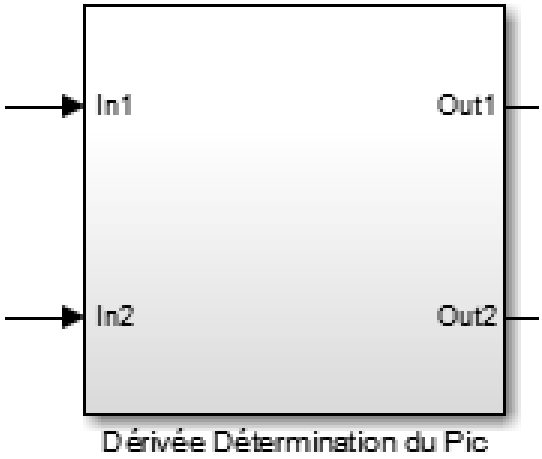


FIGURE 3.17: Enveloppe des blocs Dérivée et Détection du Pic

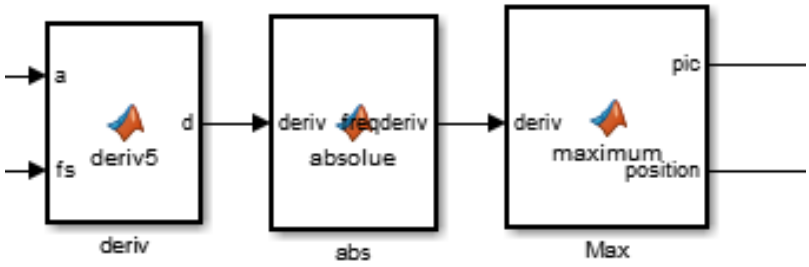


FIGURE 3.18: Blocs Dérivée et Détection du Pic

3.2.2.2.5 Bloc Calcul des paramètres Ce bloc constitue le dernier maillon de la chaine de traitement et se charge de l'estimation des paramètres.

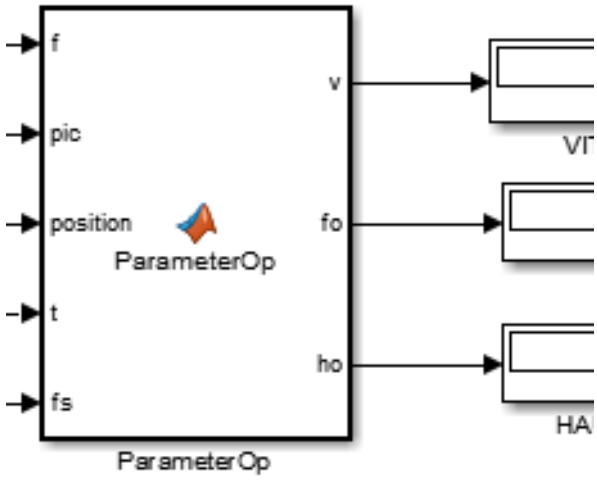
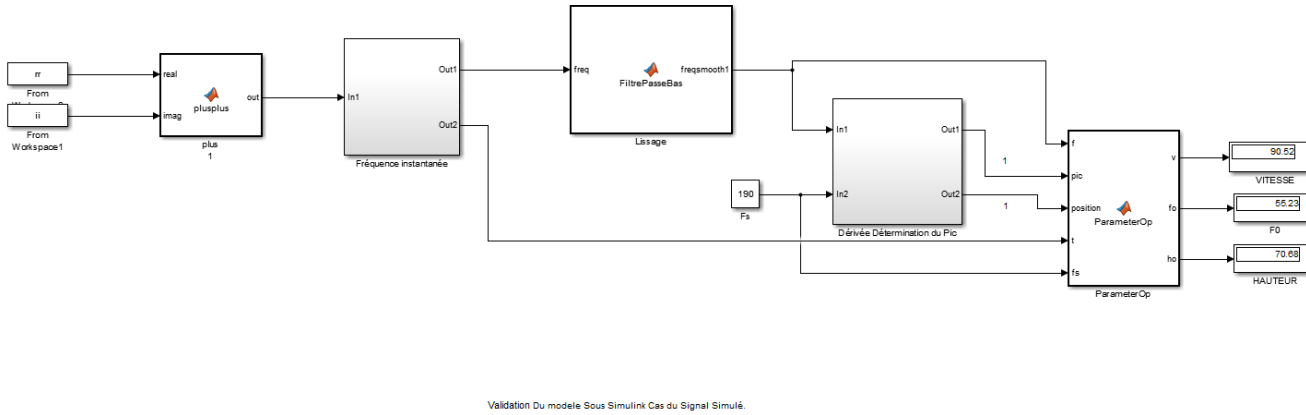


FIGURE 3.19: Bloc calcul des paramètres

3.2.3 Validation Simulink

On lance la simulation, et on compare par la suite les valeurs estimées avec les valeurs



réelles.

FIGURE 3.20: Modèle Simulink

- Comparaison :

Paramètres	Valeurs expérimentales	Modèle Simulink	Estimation de l'erreur
Vitesse (m/s)	90	90.52	0.5 %
Hauteur (m)	70	70.68	0.97 %
Fréquence (Hz)	60	55.23	7.95 %

TABLE 3.2: Calcul d'erreurs Modèle Simulink

On remarque qu'il y a une différence entre l'estimation d'erreurs de la validation Simulink et celle sous Matlab, cette différence vient du fait que la fonction transformée en signal analytique n'est identique à celle du Bloc Simulink.

3.3 Conclusion

Ce chapitre a permis la mise en place du modèle sous Simulink, les étapes de validation de l'algorithme et du modèle ont été effectuées afin de préparer le terrain pour l'implémentation via l'outil de génération de code et le ciblage matériel.

Chapitre 4

Implémentation

Sommaire

Dans ce chapitre nous décrirons l'implémentation de l'application à partir du modèle Simulink ou le passage vers le matériel, l'approche suivie visera à faire du multi-ciblage (implémentation de l' algorithme sur différentes cartes de développement).
On commencera par la présentation des cartes qui feront office de cible puis on traitera les moyens d'interfaçage avec le microphone, après cela, on s'intéressera aux moyens de transmission et de gestion des résultats, le tout dans le cadre du model-based design au sein de l'environnement Simulink.

4.1 Cibles embarquées

4.1.1 Système embarqué

Afin d'aboutir à une conception finale de l'application comprenant les différentes étapes de la chaine de traitement, (Acquisition du signal, estimation des paramètres et transmission du résultat), l'approche choisie vise à utiliser des cartes de prototypages ou systèmes embarqués, ayant un ou plusieurs processeurs avec ou non de la mémoire programmable, comme l'illustre l'image suivante.

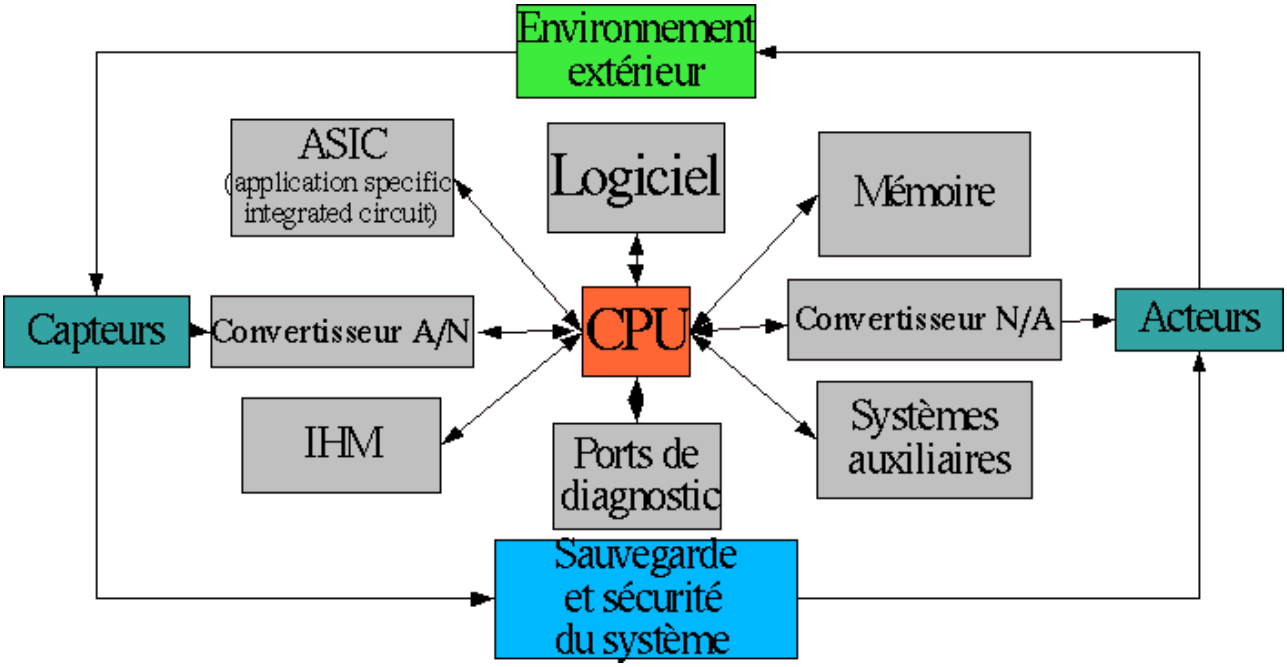


FIGURE 4.1: Système embarqué

Un système embarqué dispose généralement d'une unité de calcul des interfaces E/S de la Mémoire Volatile, Mémoire permanente, ou encore dans le cas FPGA de logique cablée reprogrammable (permet l'ajout de blocs d'accélération matériel par exemple), cela s'intégrer à un environnement à fortes contraintes (faible consommation, capacité mémoire réduite, temps réel, sécurité, robustesse).

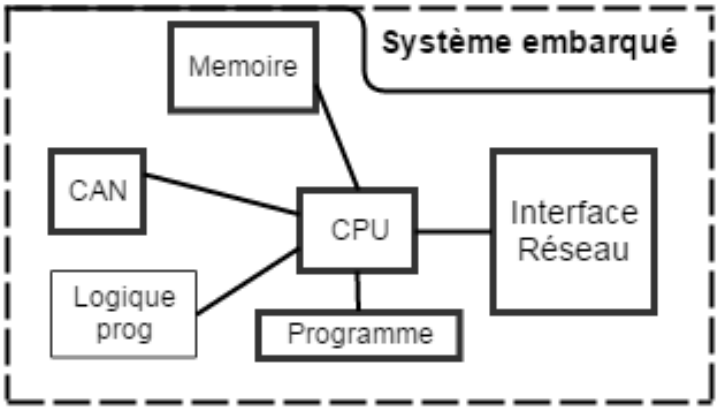


FIGURE 4.2: Structure interne du système embarqué

Les parties dont on aura besoin pour notre application sont : CPU, mémoire, convertisseur analogique-numérique CAN, interface réseau et la partie logicielles.

4.1.2 Cibles technologiques

Il existe une multitude de systèmes embarqués ou "cibles" potentiels pour l'implémentation de l'algorithme et la réalisation de l'application, on cite les cartes de prototypage à base de

Micro-controlleurs, des circuits spécifiques DSP, ASIC, circuits reprogrammables FPGA ou encore des SoC (Système sur une même puce).
Trois cibles ont été retenues pour l'implémentation :

Raspberry Pi, BeagleBone Black et ZedBoard, une des raisons qui ont motivé le choix de ces trois cartes est leur compatibilité avec l'outil de génération de code à partir de l'environnement Mathworks.

4.1.2.1 Raspberry Pi

Le Raspberry Pi contenant un processeur ARM Cortex-A7 et de multiples interfaces. [19]

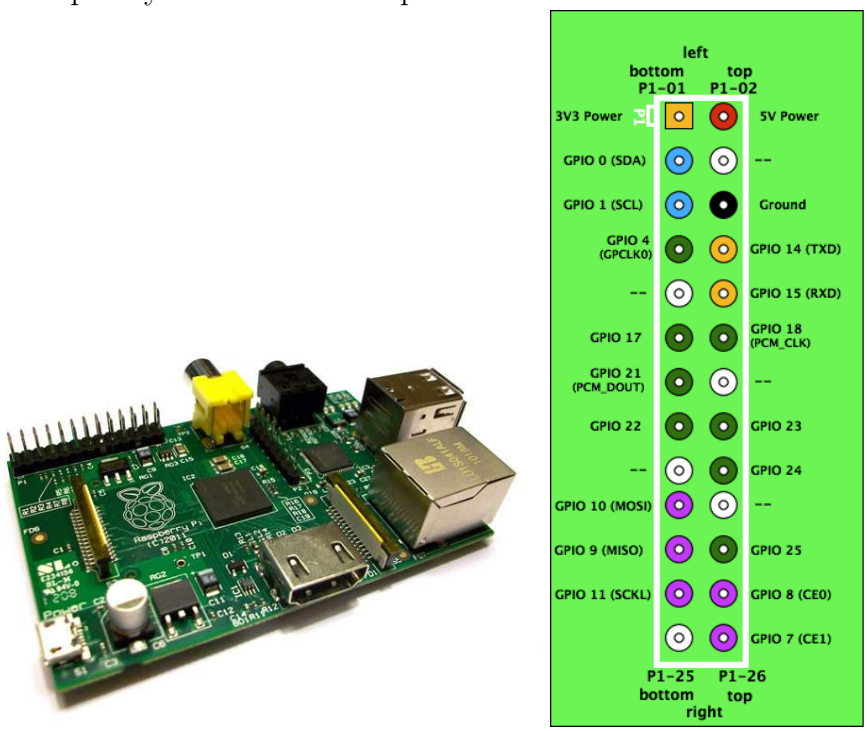


FIGURE 4.3: Raspberry Pi modèle B

Caractéristiques :

- Processeur : Cortex-A7 700 MHz
- Processeur graphique (GPU) Broadcom
- Mémoire Volatile (SDRAM) : 256 Mo
- Ports USB 2.0 x2
- Sortie vidéo : RCA Composite (PAL et NTSC) et HDMI
- Sortie audio : 3.5 mm jack, HDMI
- Port Fast Ethernet (10/100 Mbits/s)
- 8 x GPIO, UART, bus I2C, bus SPI

Le connecteur GPIO supporte des entrées/sorties binaires mais également les sorties PWM (Pulse Width Modulation) notamment des périphériques de communication tels que (I2C et SPI) le connecteur présente également des alimentations de 5 Volts et de 3.3 Volts.

4.1.2.2 BeagleBone Black

Le BeagleBone Black est une carte de prototypage embarquant un processeur Coretex-A8. [20]

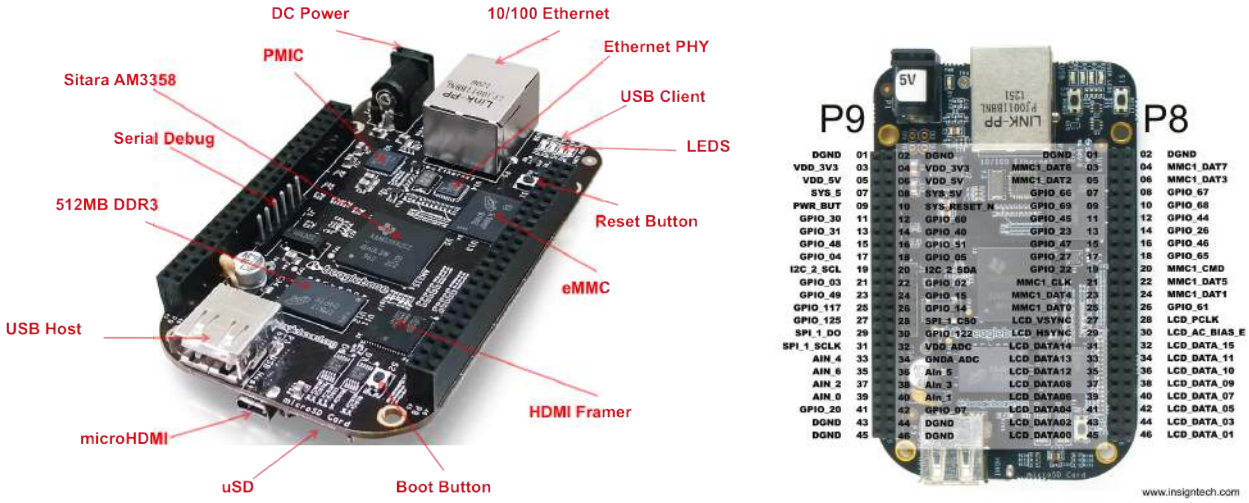


FIGURE 4.4: BeagleBone Black

Caractéristiques :

- Processeur Cortex-A8 1GHz
- Mémoire volatile 512MB DDR3
- Mémoire interne 4GB 8-bit eMMC
- Accélérateur graphique
- Accélérateur NEON
- Port USB
- Ethernet
- Micro HDMI
- 7 entrées analogiques
- 8 sorties PWM
- 5 ports série UART
- 65 entrées/sorties numériques
- Bus I2c, SPI, CAN

Remarque : Neon est un accélérateur processeur "SIMD" (Single Instruction Multiple Data) intégré dans une partie de l'ARM Cortex, permet l'utilisation du parallélisme.

4.1.2.3 ZedBoard

La carte de développement ZedBoard (de *Digilent*) Ref[] est basée sur un SoC *Xilinx* de type Zynq-7000 contenant un processeur ARM Dual-Core Cortex-A9 et couplé à un FPGA *Xilinx* Vertex 7. [21]

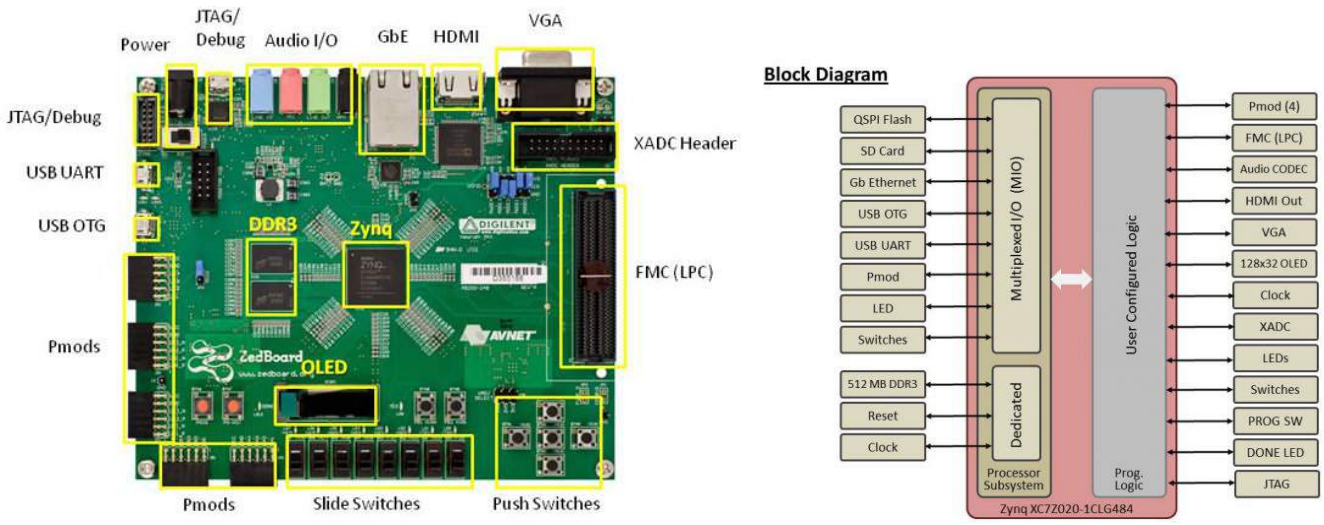


FIGURE 4.5: ZedBoard Revision D et ces Interfaces

La partie $Processor - Sub - system$ contient le dual-processeur en plus d'un Multiplexeur I/O, l'interfaçage avec la partie Programmable $Progammable - Logic$ s'effectue via un Bus dédié (Bus-AXI), enfin les interfaces de la carte sont connectés de deux façons différentes, a gauche les interfaces connectées directement au $Processor - Sub - System$, à droite les interfaces connectées à la partie programmable.

Du côté Processor Subsystem (PS) on trouve :

Processeur : Dual ARM Cortex-A9 MPCore 667 MHz
Extensions au processeur : NEON (Single/Double Precision Floating Point)
Fréquence maximum : 1Ghz.
L1 Cache : 32 KB Instruction, 32 KB Data par processeur
L2 Cache : 512 KB
On-Chip Memory : 256 KB
Mémoire volatile extérne supportée : DDR3, DDR2, LPDDR2
Mémoire statique éxterne supportée : Support(1) 2x Quad-SPI, NAND, NOR
DMA controlleur mémoire Channels : 8 (4 dédiés à la partie Programmable Logic).

Entrées sorties : 2x UART, 2x I2C, 2x SPI, 4x 32b GPIO.

Du côté logique programmable (PL) on trouve les entrées/sorties suivantes :

- GPIO : 8 LEDs, 5 Boutons poussoirs et 8 switch.
- Ecran OLED 128 x 32.
- VGA(8bits)
- HDMI.
- Controlleur et Entrées/sorties Audio.
- xADC (Convertisseur analogique numérique).

La carte Zedboard offre la possibilité de développer en $Co - Design$ software et hardware grâce aux deux entités Processing System (PS) et la Programmable Logic (PL).

4.1.2.4 Comparaison

Les systèmes embarqués cités ci-dessus contiennent tous des processeurs basé sur une architecture RISC (jeu d'instruction réduit) ARMv7 (32bits).

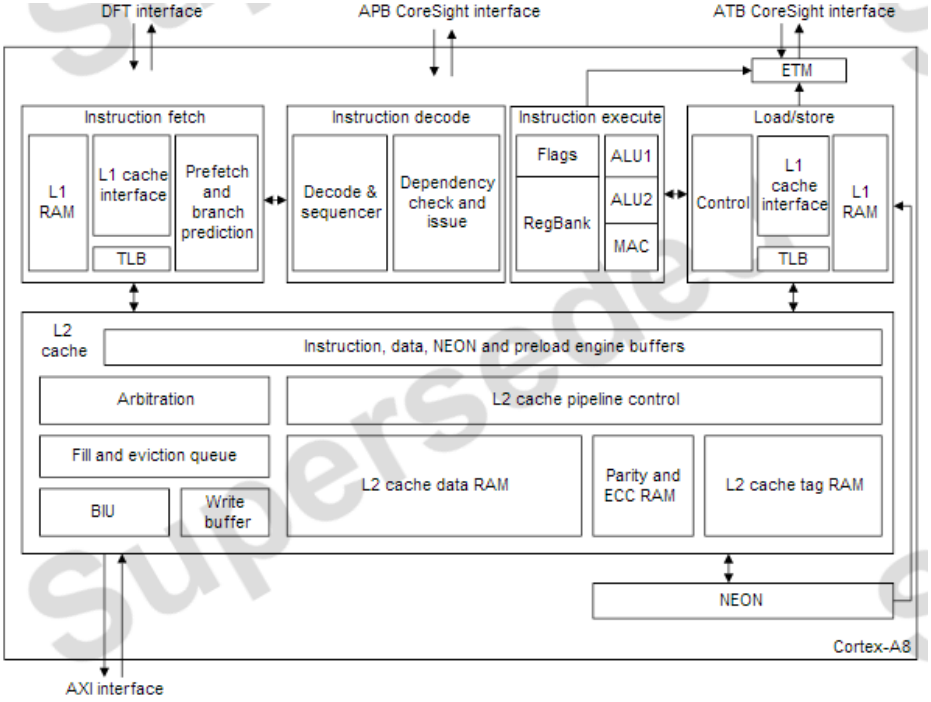


FIGURE 4.6: Architecture ARMv7

Le Cortex A8 est l'ancien haut de gamme d'ARM, le Cortex A9 est le haut de gamme actuel il permet le MPCore (multi-core) exploité justement sur ZedBoard, enfin le Cortex A7 Destiné à l'entrée de gamme, consomme beaucoup moins que le Cortex A8 et est plus petit.

4.2 Modèle et Génération de code

4.2.1 Simulink Coder et Cross-Compilation

(Annexe B)

4.2.1.1 Compilation Croisée

La compilation croisée est l'utilisation d'un compilateur spécifique pour créer un programme exécutable pour une autre machine. Ce compilateur doit tenir compte des spécificités du processeur cible, cette notion est indispensable quand il s'agit de travailler sur un système embarqué.
Lorsque l'on souhaite exporter un programme vers une architecture ou un système différent, il faut utiliser un compilateur spécialement créé pour la cible. Le générer de bout en bout peut s'avérer extrêmement fastidieux et présente un risque d'erreurs important. On utilise généralement une chaine d'outils de compilation (Toolchain) propre à chaque cible.

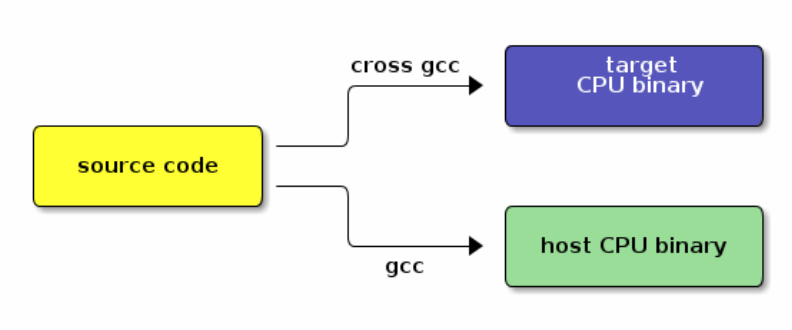


FIGURE 4.7: Compilation croisée

Une chaine de compilation croisée (Cross Toolchain) se compose des éléments suivants :

- L'en-tête de fichier d'un noyau Linux.
- Un ensemble de logiciels outils de développement : binutils (utilitaires binaires).
- Un compilateur : GCC.
- Les bibliothèques C nécessaires : glibc etc. (L'intégration de bibliothéques statique "extension .a" intégrée à l'exécutable lors de la compilation ou alors dynamique,en -.so (Sharing Object), est aussi possible.)

Une méthode traditionnelle pour la compilation destinée aux cibles *ARM* est l'utilisation de arm-linux-gnueabi-gcc.

Dans le contexte de la génération d'exécutable adapté à une cible depuis Simulink, tout d'abord le modèle est converti en code C (Simulink Coder décrit dans la section suivante), puis Simulink fait appel à la chaine de compilation propre à chaque cible (Compilation croisée), en effet, chaque matériel est automatiquement associé à la chaine de compilation adéquate pour la génération du binaire.

ZedBoard : Xilinx Software Développement Kit (SDK), BeagleBone Black : Linaro v4,8 , Raspberry Pi : gcc.

4.2.1.2 Simulink Coder

Simulink Coder est l'outil de génération de code C à partir de modèle Simulink. [22]

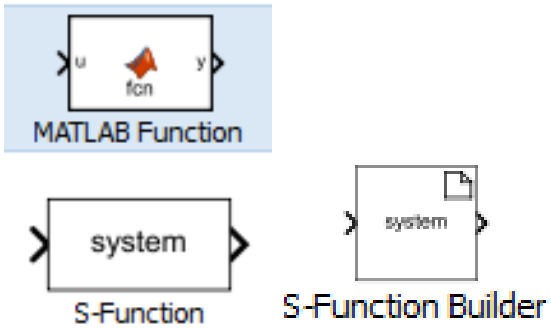


FIGURE 4.8: Blocs implémentables

Étapes de génération de code C à partir de modèles Simulink.

— Préparer le modèle : Le modèle doit être vérifié pour assurer que du code C peut être généré sinon des modifications (sur le modèle ou l'algorithme : fonctions non implémentables, variables propres au workspace entre autres) doivent être effectuées.
— Test et Vérification : Du code "MEX function" ou (Modèle Exécutable) pour une simulation sur la station de travail est faite, il est alors possible de vérifier le bon fonctionnement du modèle.
— Génération : Le code C est généré.

Simulink Coder, exécute automatiquement les étapes précédentes.

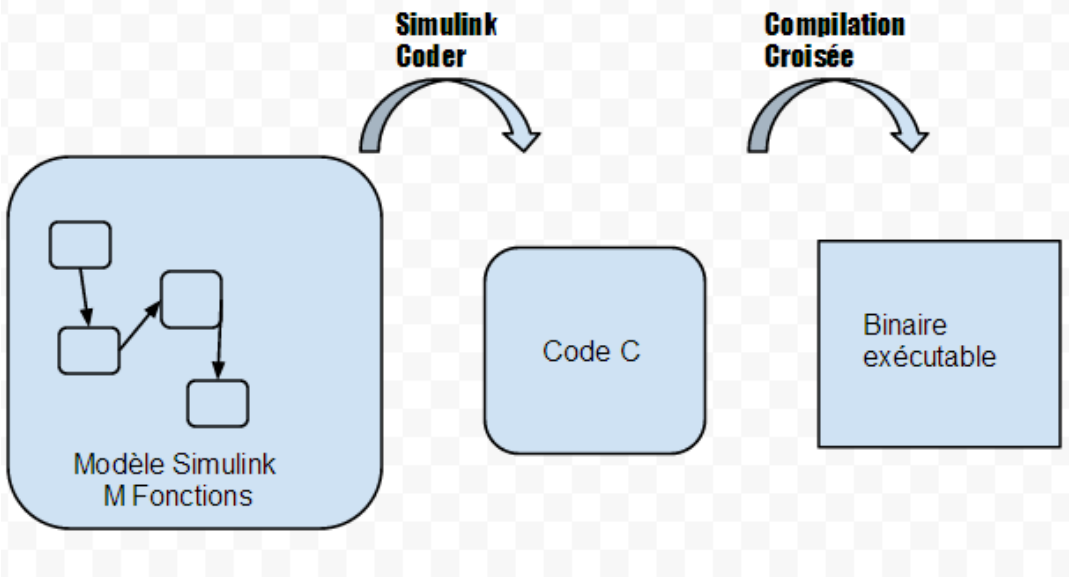


FIGURE 4.9: Relation Modèle-Simulink Coder-Compilation Croisée

Enfin l'illustration résume la relation entre "Modèle - Simulink Coder - Compilation Croisée".

4.3 Préparation du modèle pour l'implémentation

Le modèle vu précédemment doit être complété pour passer à l'implémentation, l'interfaçage et la transmission des résultats seront mis en place.

4.3.1 Interfaçage du capteur et acquisition du signal

4.3.1.1 Interfaçage pour Cibles

L'interfaçage entre le système embarqué et le capteur sonore est nécessaire. Sachant que la cible tourne sous un système d'exploitation Linux on peut à partir de là, penser à utiliser des pilotes pour la gestion du son. une solution est le pilote ALSA.

ALSA ou Advanced Linux Sound Architecture est un pilote du noyau Linux, destiné au support des cartes son. ALSA propose des outils permettant d'effectuer plusieurs tâches

Dans ce cas, on utilisera une carte son USB qu'on configurera comme carte son par défaut.

Simulink fournit un bloc de lecture audio à partir de la carte audio du système embarqué utilisant ALSA, ce bloc s'appelle ALSA audio capture.

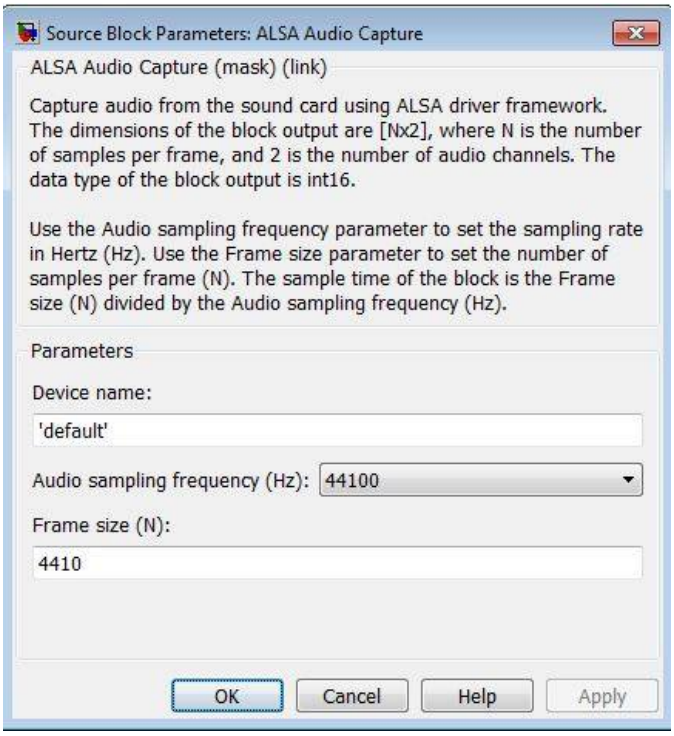


FIGURE 4.10: Bloc ALSA audio capture

Le bloc capture l'entrée droite et gauche du canal audio à partir de l'entrée audio de la carte son, en utilisant le pilote ALSA. Le bloc fournit en sortie une matrice de N lignes et 2 colonnes de valeurs réelles. Le nombre de colonne correspond au canal audio droit et gauche et le nombre N au nombre d'échantillons.

FIGURE 4.11: ALSA audio capture

— Afin d'utiliser la carte son par défaut, on laisse le paramètre Device name assigné à $'default'$.
— Fréquence d'échantillonnage (Hz)
— On peut entrer la fréquence d'échantillonnage voulue dans la case Audio samplig frequency, par défaut ce paramètre vaut 44.1 KHz.
— Nombre d'échantillons par bloc

— On peut entrer le nombre d'échantillons que ce bloc fournit en sortie vers le modèle, sa valeur par défaut est 4410 échantillons.

La configuration de la carte son sous linux est décrite en annexe.

4.3.1.2 Interfaçage pour la station de travail

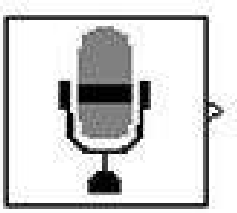


FIGURE 4.12: From Audio Device

On utilise le bloc from audio device, pour faire l'acquisition du signal du microphone vers le modéle Simulink lors des tests sur ordinateur.

4.3.2 Conception du Filtre Passe-Bande

4.3.2.1 Filtre en software

Contrairement aux signaux simulés (vus au chapitre 3), le bloc filtre passe bande s'impose et est incontournable lorsqu'il s'agira de faire les tests avec des signaux réels, sous Simulink l'outil *FdaTool* permet la conception de tout types de filtres.

La définition des paramètres et le choix du type de filtre se font comme suit :

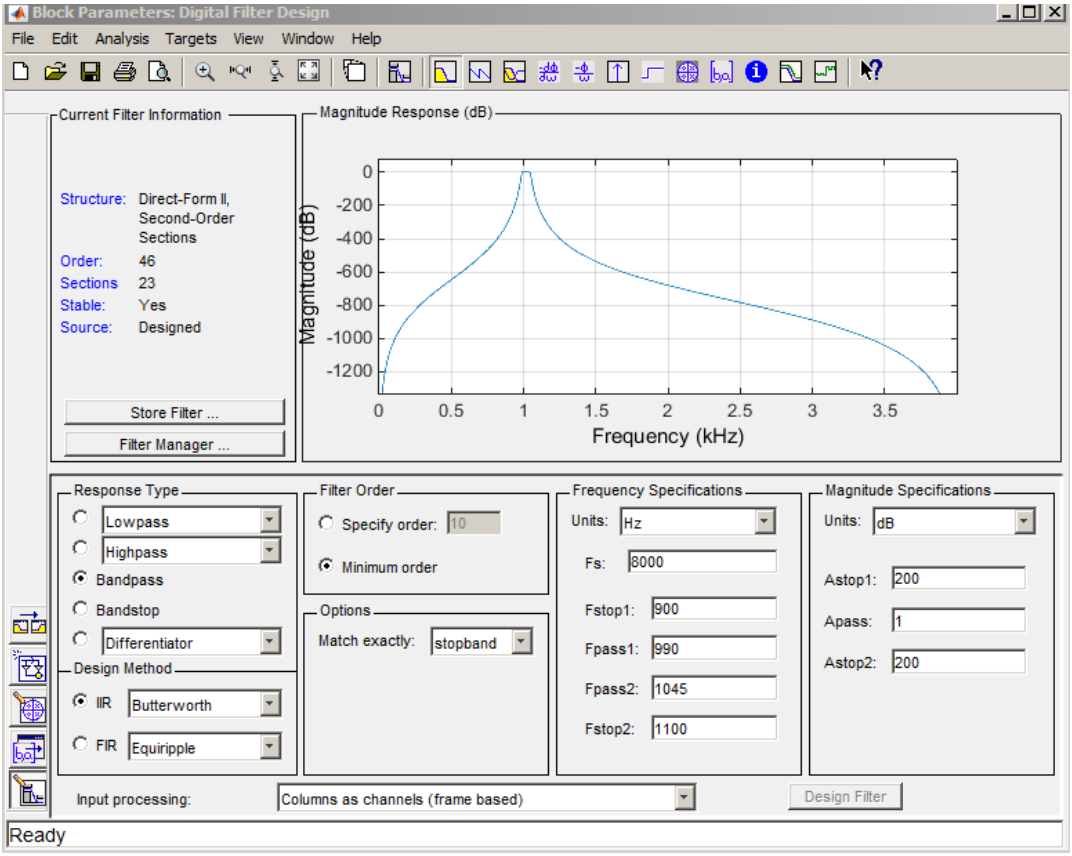


FIGURE 4.13: Réponse du filtre

Le gabarit du filtre :

- Fréquences de coupures : fstop1 = 900 hz, fstop2 = 1100 hz
- Bande passante : fpass1 = 990 hz, fpass2 = 1045
- L'atténuation : Astop1 = 200, Astop2 = 200.

Dans ce cas c'est un IIR Butterworth filtre passe bande utilisé dans notre modèle, visant à cibler une source acoustique émettant un signal acoustique d'une fréquence proche d'1khz .

D'autres cibles peuvent être localisées, suivant leur fréquences fondamentales moyennes, quelques cas sont répertoriés dans le tableau suivant [23] :

Aéronef	f_0 moyenne (Hz)
Van's Aircraft RV-7	91
Swearingen SX-300	135
Bushby Mustang II	86
Vought F4U Corsair	140

TABLE 4.1: Empreinte fréquentielle de quelques modèles d'aéronefs à hélices

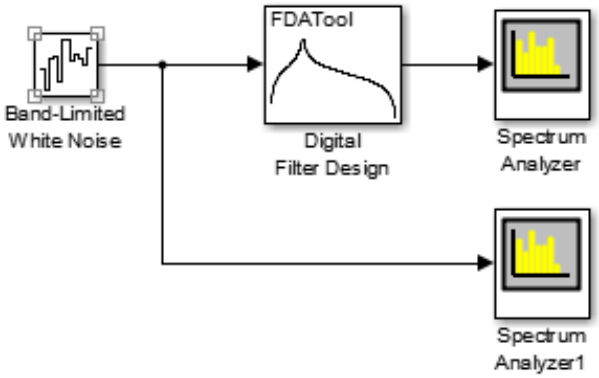


FIGURE 4.14: Fda Tool

Enfin une validation par simulation.

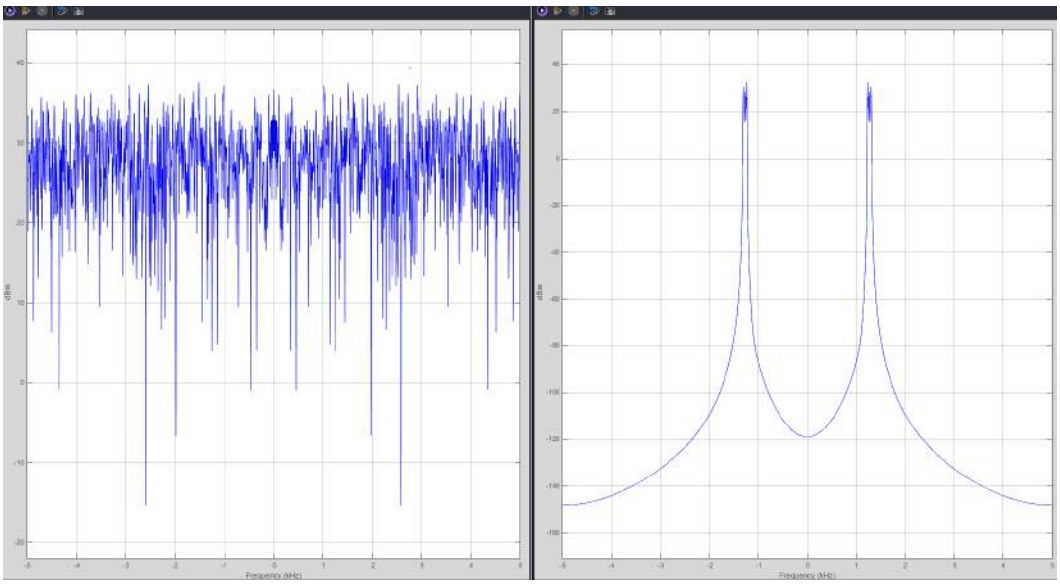


FIGURE 4.15: Avant/Après filtrage

4.3.3 Gestion des résultats du traitement

Le déploiement final de l'application doit inclure un moyen de sauvegarder et de consulter les résultats du traitement, pour cela deux nouveau bloc doivent être conçu, l'un consiste à écrire les résultats sur un fichier a l'intérieur de système, l'autre à communiquer le résultat vers une base de donnée externe.

4.3.3.1 Sauvegarde des résultats

Une partie sauvegarde doit être développer afin de gérer l'information (paramètres estimés), une approche consiste en l'écriture sur un fichier contenu dans le système embarqué.
Pour y parvenir, du code C est utilisé, on sélectionne le bloc $S - Function\ Builder$ sur Simulink.

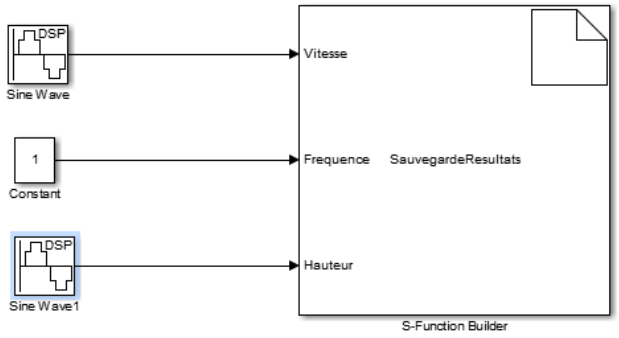


FIGURE 4.16: Modèle écriture

Le code pour l'écriture :

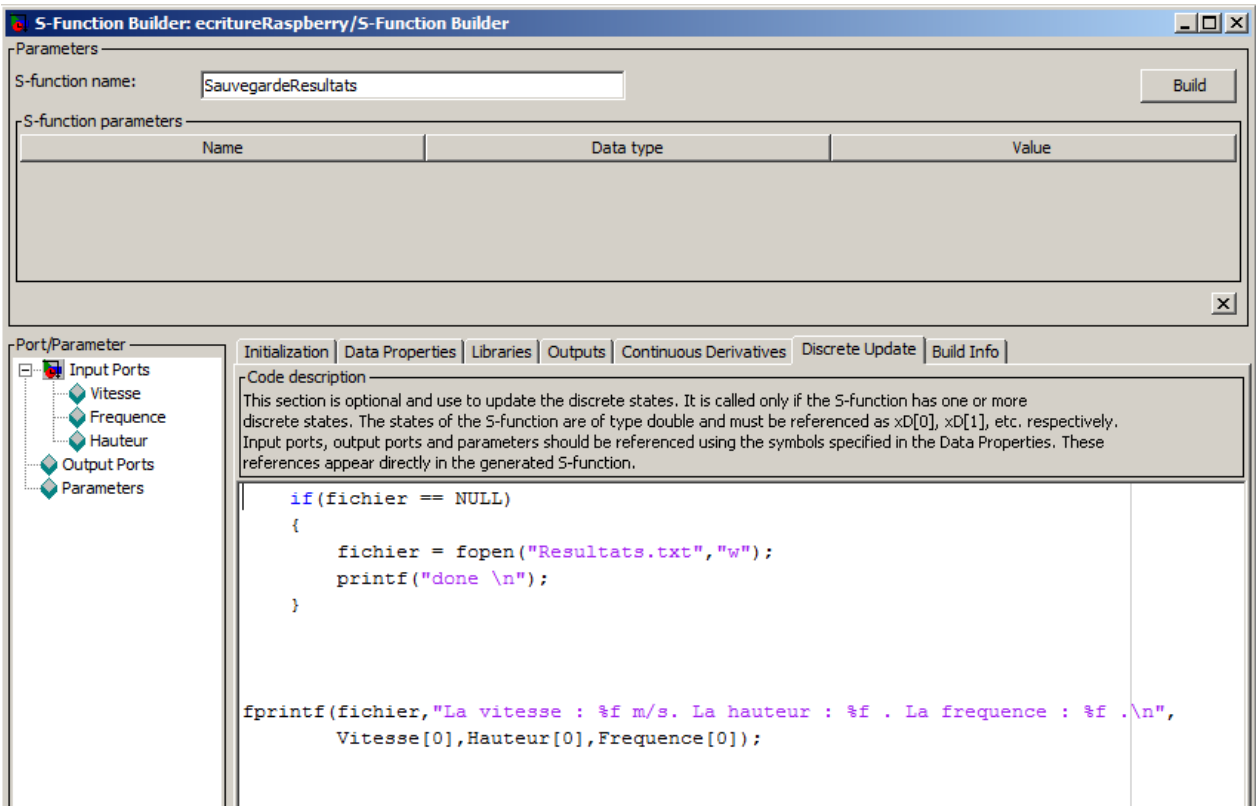


FIGURE 4.17: Code écriture

Cette partie du bloc permet de mettre à jour le fichier afin d'ajouter de nouvelles données.

Le fonctionnement du bloc *SBuilder* est décrit en annexe.

Validation

FIGURE 4.18: Vérification du fonctionnement

4.3.3.2 Transmission Vers le réseau internet

Plusieurs capteurs peuvent être connecter à un serveur, pour y transmettre les résultats de l'estimation des paramètres, une fois ces ressources récupérées, une application web peut très bien exploiter ces données à différentes fins (suivi, visualisation, information, archive, classification ...).

7	Couche Application
6	Couche Présentation
5	Couche Session
4	Couche Transport
3	Couche Réseau
2	Couche Liaison de données
1	Couche Physique

FIGURE 4.19: Modèle en couches

On projette pour cela le système embarqué ainsi que l'application dans le modèle en couches.

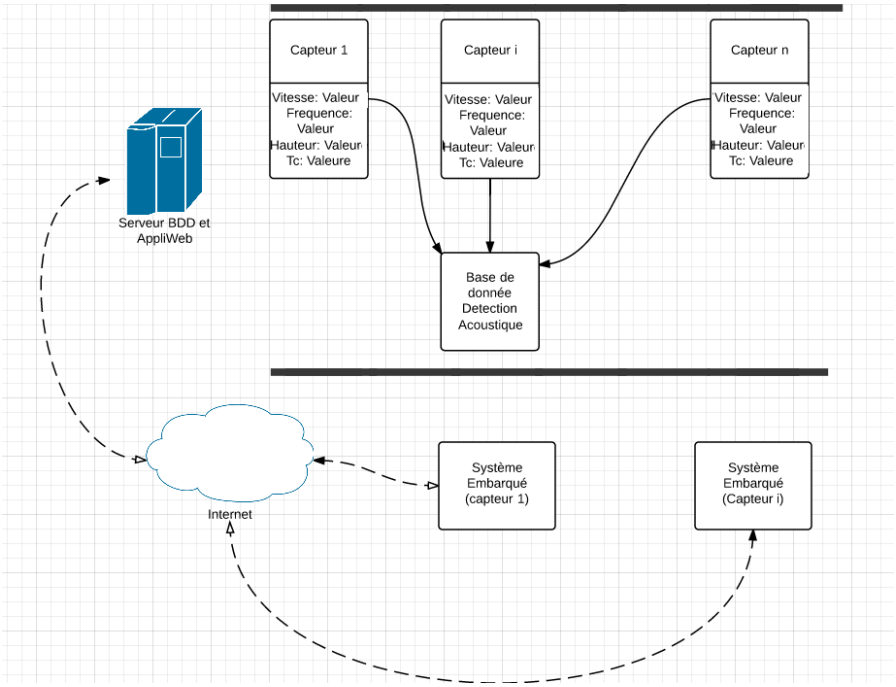


FIGURE 4.20: Schéma de principe

4.3.3.2.1 Couche Physique et Liaison de données Comme vu précédemment les trois systèmes contiennent une interface réseau avec un port Ethernet, celle ci sera exploitée pour la transmission des résultats d'estimation, d'autres solutions peuvent être envisagés tel que l'utilisation de modules sans fil $wifi$, $3g$ etc.

4.3.3.2.2 Couche Réseau Une solution est l'exploitation de l'interface Ethernet avec la configuration de l'interface, affectation d'une adresse *ip*, affectation du *dns* pour la résolution de l'url du serveur et configuration d'une passerelle.

- Configuration de l'adresse IP La configuration s'effectue sur le système d'exploitation de la cible $etc/networks/interfaces$ on y indiquera (dans le cas statique) l'adresse ip l'adresse du sous-reseau enfin l'adresse ip de la passerelle.
- Configuration du DNS
- Configuration de la passerelle

Autre exemple en utilisant un modem externe 3g ou clé wifi.

4.3.3.2.3 Couche applicative (Application-Présentation-Session-Transport) La transmission de données de la cible vers un serveur distant, exploite le modèle client-serveur. Deux applications l'une s'exécute sur la cible embraqué, l'autre s'occupe de la gestion et la visualisation des ressources sur le serveur.

Deux scénarios ont été envisagés :

- Des solutions tels que *mysqlclient odbc* ou encore *sqlite* sont possible pour gérer les requêtes *sql* entre le modèle et la base de donnée, enfin la visualisation pourrait se faire sur une application web.
- Une solution plus simple est l'exploitation de la plateforme *ThingsSpeak* via le bloc du même nom.

Les couches (Application-Présentation-Session) ont dans ce cas un couplage fort.

4.4 Implémentation

4.4.1 Station de travail

4.4.1.1 Mise en œuvre

Deux manipulations pourront etre effectuées :

- Traitement différé (le signal est chargé du Workspace matlab pour un traitement sur la cible)
- Traitement temps-réel (le signal sonore vient de l'interface externe pour avoir un fonctionnement normal de l'application)

Outre le chargement du signal depuis le workspace, le traitement via l'exploitation de l'interface microphone d'un ordinateur et l'utilisation du bloc audio acquisition est effectué.

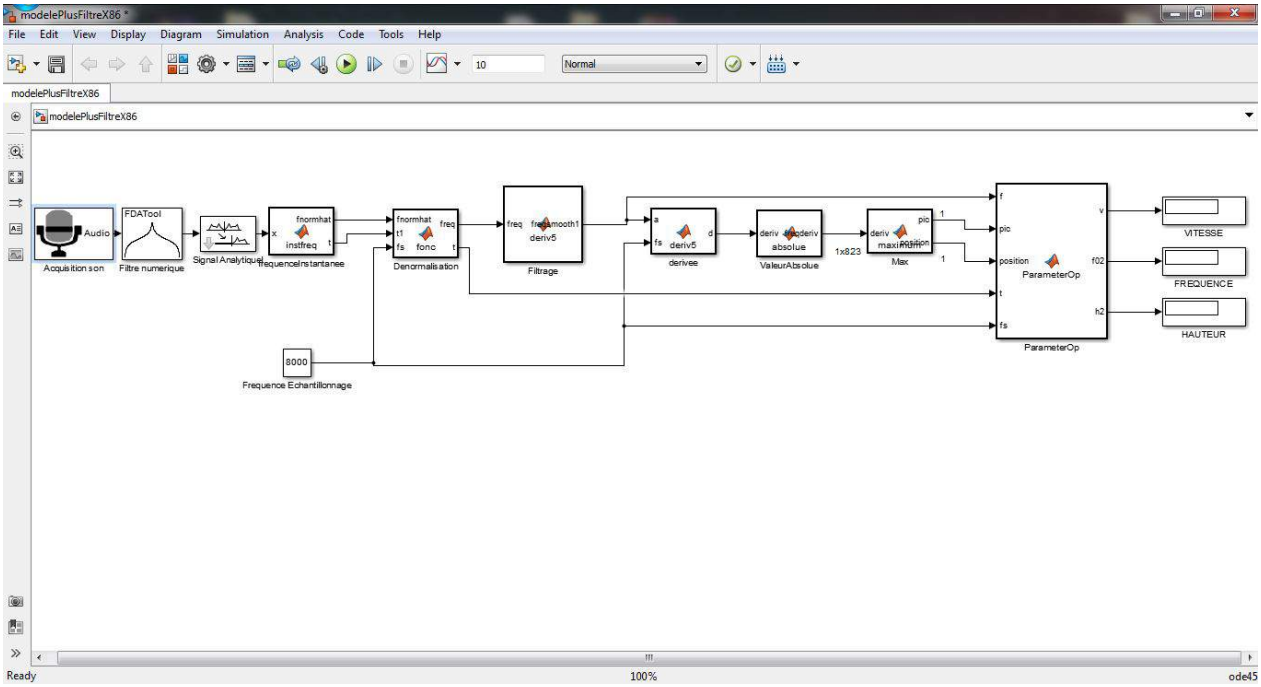


FIGURE 4.21: Modèle pour une station de travail

On l'ajoute donc au modèle de traitement, un buffer permet d'avoir le nombre d'échantillons suffisant afin de traiter le phénomène *doppler* en entier.
L'image ci dessus illustre le modèle prêt pour les tests, acquisition, prétraitement (fréquence instantanée), traitement (parameterOp) et enfin une visualisation via le *display*.

4.4.1.2 Mesure des performances

Dans le contexte de cibles contenant une partie FPGA, il est utile de savoir si oui ou non il y'a intéret de mettre une partie du traitment en logique cablée (accélération matérielle) aussi les cibles *ARM* vues précédemment contiennent un accélérateur *NEON* (utilisation du parallélisme).

Le Profiling ou mesure des performances consiste à analyser l'exécution d'un programme afin de connaitre son comportement à l'exécution et celui des différentes routines (fonctions et blocs de traitement) :

- la liste des fonctions appelées.
- l'utilisation processeur.
- le temps passé dans chacune des fonctions.

Le Profiling est utilisé pour identifier les parties qu'il faut optimiser selon le principe que l'on ne peut pas optimiser ce que l'on ne sait pas mesurer (adaptation du "Si vous ne pouvez pas le mesurer, vous ne pouvez pas le gérer" de Kaplan).

On lance la génération du rapport.

Summary | Function Details | Simulink Profiler Help | Clear Highlighted Blocks

Function List

Name	Time		Calls	Time/call	Self time		Location (must use MATL
simulate(modelePlusFiltreX86)	5.30403400	100.0%	1	5.30403400000	0.00000000	0.0%	modelePlusFiltreX86
simulationPhase	4.39922820	82.9%	1	4.39922820000	0.79560510	15.0%	modelePlusFiltreX86
modelePlusFiltreX86.Outputs.Major	3.60362310	67.9%	217	0.01660655806	0.03120020	0.6%	modelePlusFiltreX86
modelePlusFiltreX86/Filtrage / SFunction (StateflowChild.Outputs.Major)	1.76281130	33.2%	217	0.00812355438	1.76281130	33.2%	modelePlusFiltreX86/
modelePlusFiltreX86/Filtrage (AtomicMATLABFunctionSubSystem.Outputs.Major)	1.76281130	33.2%	217	0.00812355438	0.00000000	0.0%	modelePlusFiltreX86/
modelePlusFiltreX86/derivee / SFunction (StateflowChild.Outputs.Major)	0.85800550	16.2%	217	0.00395394240	0.85800550	16.2%	modelePlusFiltreX86/
modelePlusFiltreX86/derivee (AtomicMATLABFunctionSubSystem.Outputs.Major)	0.85800550	16.2%	217	0.00395394240	0.00000000	0.0%	modelePlusFiltreX86/
compileAndLinkPhase	0.73320470	13.8%	1	0.73320470000	0.73320470	13.8%	modelePlusFiltreX86
modelePlusFiltreX86/frequenceInstantanee/ SFunction (StateflowChild.Outputs.Major)	0.32760210	6.2%	216	0.00151667639	0.32760210	6.2%	modelePlusFiltreX86/ SFunction
modelePlusFiltreX86/frequenceInstantanee (AtomicMATLABFunctionSubSystem.Outputs.Major)	0.32760210	6.2%	216	0.00151667639	0.00000000	0.0%	modelePlusFiltreX86/
modelePlusFiltreX86/ParameterOp / SFunction (StateflowChild.Outputs.Major)	0.15600100	2.9%	217	0.00071889862	0.15600100	2.9%	modelePlusFiltreX86/
modelePlusFiltreX86/ParameterOp (AtomicMATLABFunctionSubSystem.Outputs.Major)	0.15600100	2.9%	217	0.00071889862	0.00000000	0.0%	modelePlusFiltreX86/
modelePlusFiltreX86/Acquisition son (S-Function.Outputs.Major)	0.14040090	2.6%	216	0.00065000417	0.14040090	2.6%	modelePlusFiltreX86/
terminationPhase	0.12480080	2.4%	1	0.12480080000	0.07800050	1.5%	modelePlusFiltreX86
modelePlusFiltreX86/ValeurAbsolue / SFunction (StateflowChild.Outputs.Major)	0.07800050	1.5%	217	0.00035944931	0.07800050	1.5%	modelePlusFiltreX86/
modelePlusFiltreX86/ValeurAbsolue (AtomicMATLABFunctionSubSystem.Outputs.Major)	0.07800050	1.5%	217	0.00035944931	0.00000000	0.0%	modelePlusFiltreX86/
modelePlusFiltreX86/Signal Analytiquel/Digital Filter (S-Function.Outputs.Major)	0.07800050	1.5%	216	0.00036111343	0.07800050	1.5%	modelePlusFiltreX86/ Filter
modelePlusFiltreX86/Denormalisation (AtomicMATLABFunctionSubSystem.Outputs.Major)	0.06240040	1.2%	217	0.00028755945	0.01560010	0.3%	modelePlusFiltreX86/
modelePlusFiltreX86/Filtre numerique/Digital Filter (S-Function.Outputs.Major)	0.06240040	1.2%	216	0.00028889074	0.06240040	1.2%	modelePlusFiltreX86/ Filter
modelePlusFiltreX86/Denormalisation/ SFunction (StateflowChild.Outputs.Major)	0.04680030	0.9%	217	0.00021566959	0.04680030	0.9%	modelePlusFiltreX86/
modelePlusFiltreX86.Terminate	0.04680030	0.9%	1	0.04680030000	0.04680030	0.9%	modelePlusFiltreX86
initializationPhase	0.04680030	0.9%	1	0.04680030000	0.01560010	0.3%	modelePlusFiltreX86

FIGURE 4.22: Rapport

Résultat :
On remarque que le bloc "estimation des paramètres" (parameterOp) ne consomme que $2,9\%$ du temps global de traitement on constate ainsi que la solution analytique au problème de localisation est performante, d'un autre coté le filtre de lissage $33,2\%$ constitue le bloc le plus lourd du modèle.

Pour conclure, l'incorporation d'une partie HDL (Sous Zynq) ou d'un traitement SIMD (via l'accélérateur NEON) pourrait être envisagé au niveau du filtre de lissage si besoin.

4.4.2 Implementation sur Cibles

En Model-Based design l'implementation se décompose en deux étapes :

- Mode External (ou Hardware-In-The-Loop HIL) Déploiement du code (binaire propre au modèle + un protocole de communication) ceci permet le contrôle et la visualisation des signaux depuis la station de travail afin de vérifier le comportement du modèle une fois implémenté.
- Mode Normal Déploiement du code (seul) sur la cible : Génération simple du binaire et exécution de celui ci sur la cible.

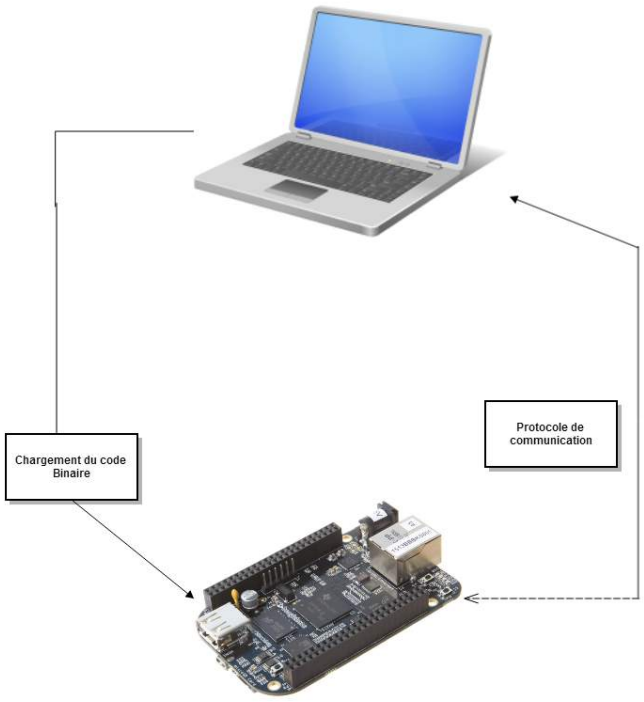


FIGURE 4.23: Schématisation du déploiement en mode "external" ou Hardware In the Loop

Dans le cadre du HIL on rencontre souvent l'exploitation d'une connexion série via le port "USB" ou encore SSH via le port Ethernet.

4.4.2.1 BeagleBone Black

Ayant le support de la carte BeagleBone installé sur l'environnement Simulink l'implémentation s'effectue comme suit :

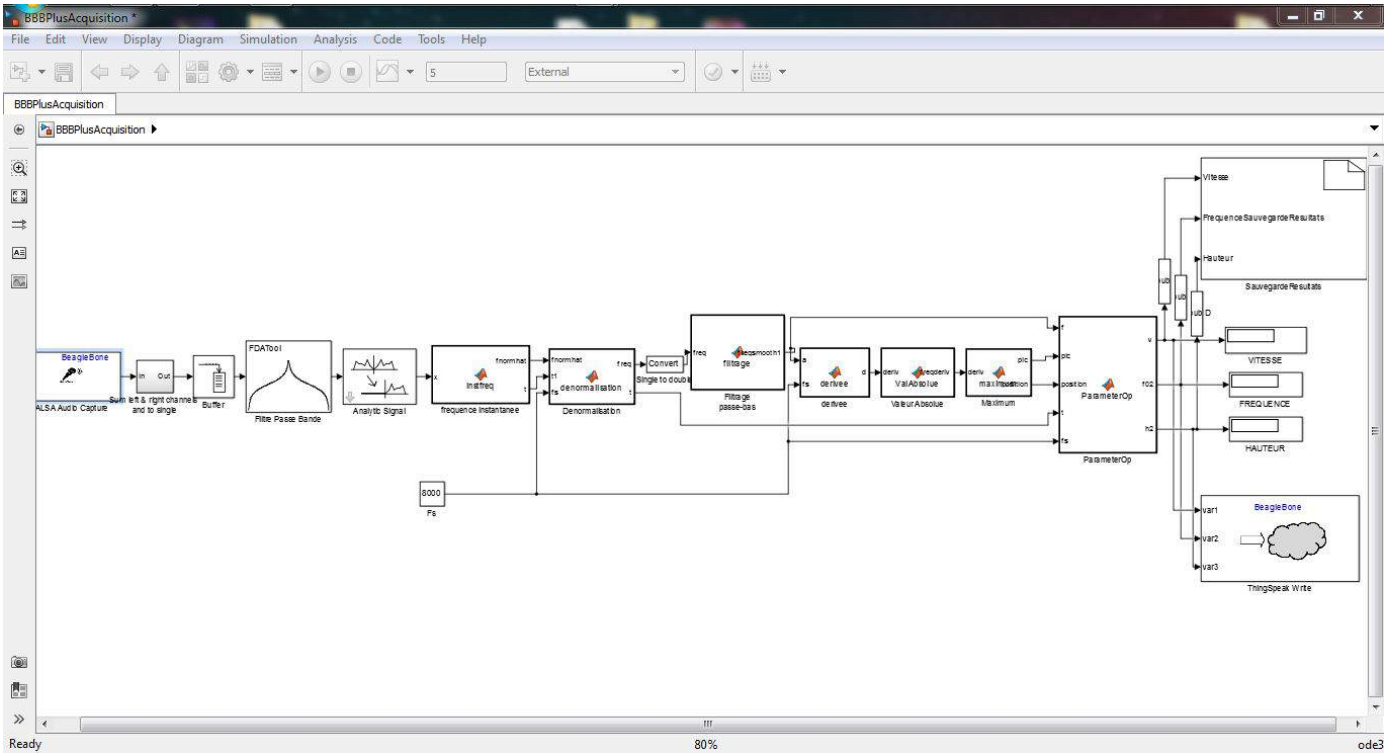


FIGURE 4.24: Modèle pour BeagleBone Black

On rajoute les parties acquisition en entrée de la chaine, les blocs transmission et sauvegarde en sortie du modéle.

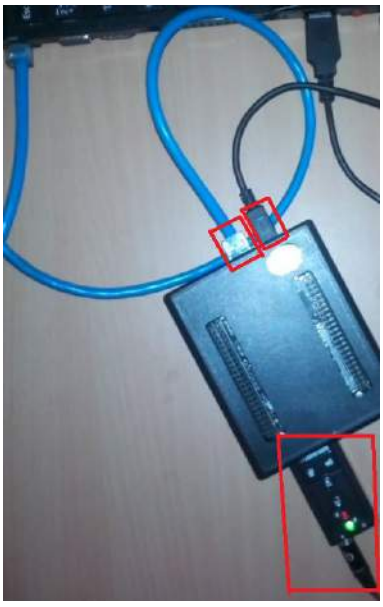


FIGURE 4.25: Branchement BeagleBone Black

— La carte son usb
— Interface série pour l'implémentation (envoi du binaire)
— Interface Ethernet pour la transmission des données lors du traitement.

4.4.2.2 ZedBoard

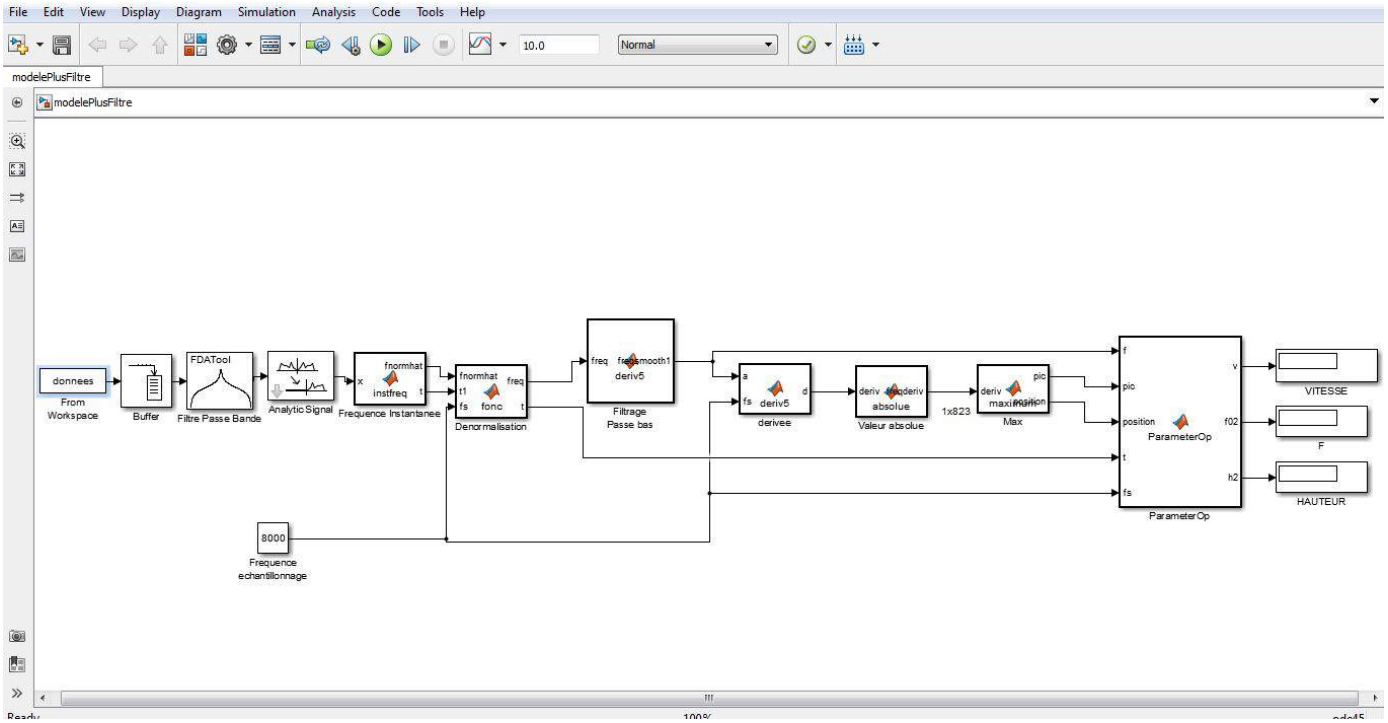


FIGURE 4.26: Modèle Simulink Zedboard

FIGURE 4.27: Branchement sur Zedboard

— L'interface Jtag illustré sur l'image ci dessus, permet de programmer la partie FPGA.
— Interface série pour l'envoi du binaire ou (soft).
— Interface Ethernet pour la transmission des données lors du traitement.

Remarque : Sur Zedboard on effectue un traitement différé.

4.4.2.3 Raspberry Pi

Le modèle pour Raspberry Pi est similaire à celui pour l'implémentation sur BeagleBone Black.

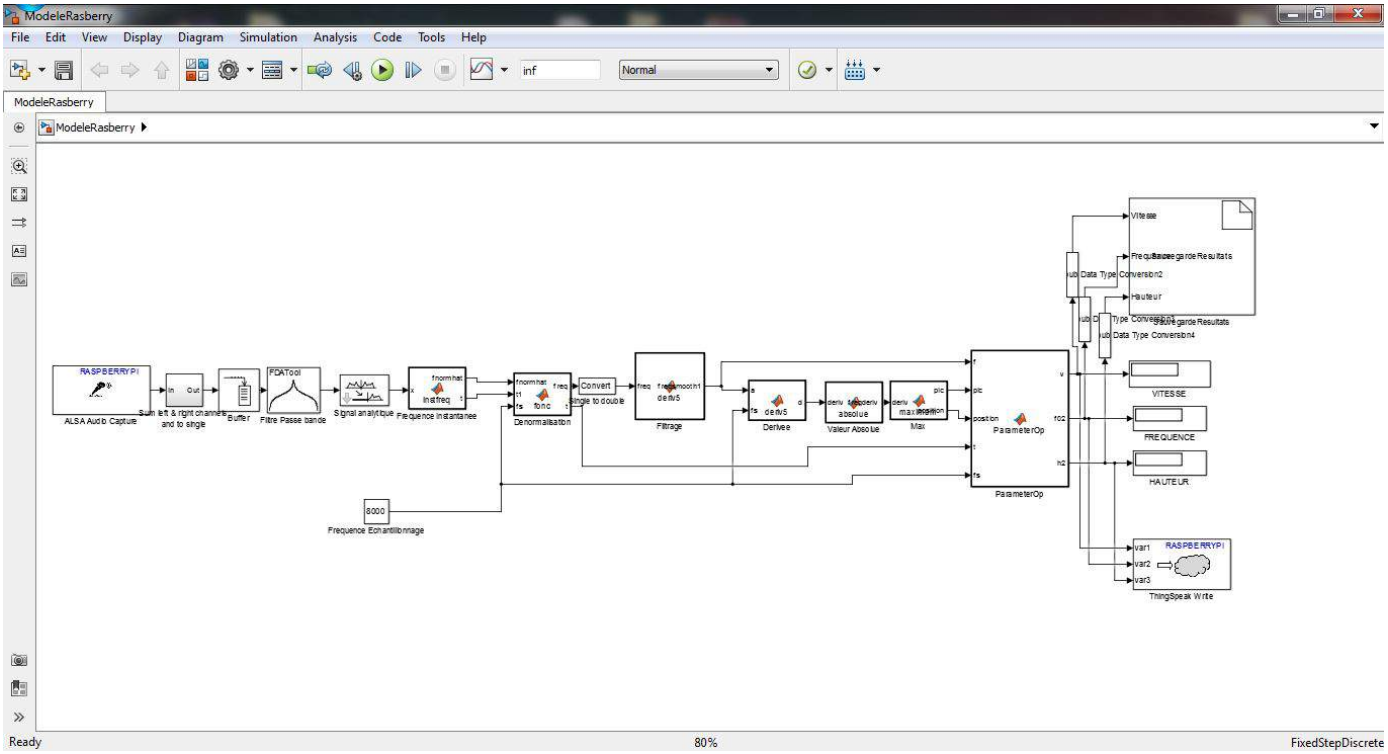


FIGURE 4.28: Modèle

On retrouve les mêmes interfaces que celles vues sur Beaglebone Black (Série du port USB et Ethernet).

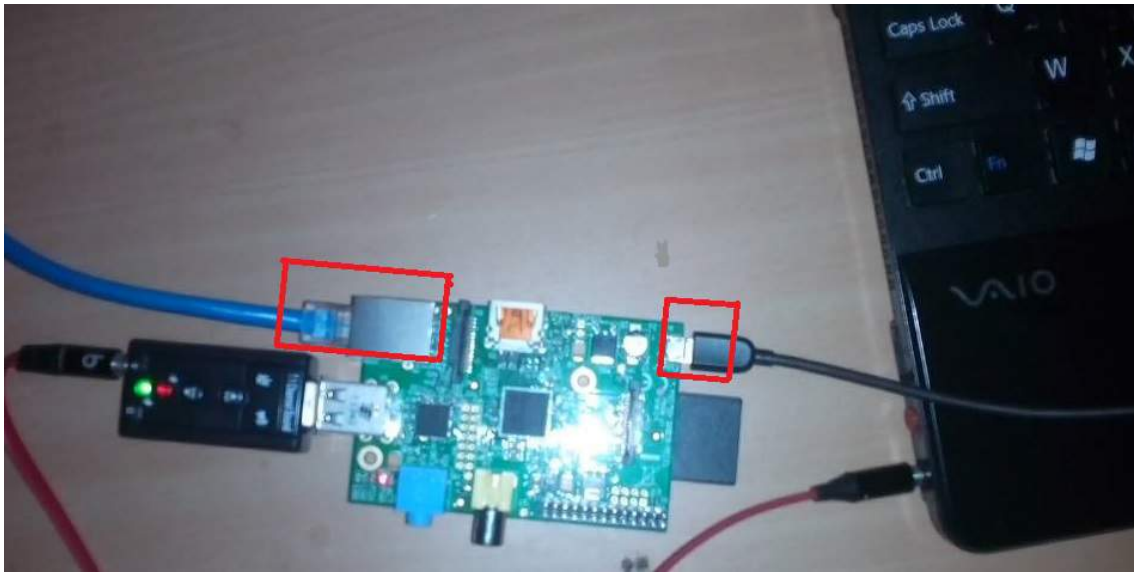


FIGURE 4.29: Implémentation sur Raspberry Pi

Le déploiement du modèle à partir de Simulink passe par la sélection de l'option "Normal" ou "External".

- Mode External (ou Hardware-In-The-Loop HIL) Déploiement du code (binaire propre au modèle + un protocole de communication) ceci permet le controle et visualisation des signaux depuis la station de travail afin de vérifier le comportement du modèle une fois implémenté.
- Mode Normal Déploiement du code (seul) sur la cible : Génération simple du binaire et exécution de celui ci sur la cible.

Un simple > "Delpoy to Hardware" fait la génération et l'envoi du binaire sur la cible.

4.5 Conclusion

L'implémentation et le déploiement d'un modèle fonctionnant sous Simulink a été effectuée sur plusieurs cibles tout comme la mise en évidence du passage assez simple d'un système embarqué à un autre, enfin la mesure des performances des différents blocs du modèle a montré la faible empreinte calcul du bloc estimation des paramètres.

Chapitre 5

Résultats de l'implémentation et discussions

Sommaire

La partie suivante sera consacrée à l'exposition des résultats de l'implémentation (sur les différentes cibles), tout d'abord avec le traitement des signaux simulés, puis les signaux réels (traitement différé puis temps-réel) en deux parties ceux issus de la chambre sourde et enfin ceux en champ libre. Les signaux simulés sont produits par une fonction MATLAB génératrice de Doppler, quant aux signaux réels, ils ont été acquis dans une chambre sourde ainsi qu'en extérieur issus de la plateforme d'expérimentation.

5.1 Signaux simulés

L'estimation des paramètres de mouvement est faite à partir d'un signal sonore simulé. Connaissant les vrais paramètres d'entrées, c'est à dire, vitesse, hauteur et fréquence, on valide l'implémentation en comparant les résultats aux vraies valeurs. Les signaux simulés ont été produit par une fonction MATLAB génératrice de signal Doppler. Plusieurs tests ont été effectué :

5.1.1 Test 1

5.1.1.1 Conditions

Fréquence d'émission de la source sonore simulée est de $f_0 = 60$ Hz.

Hauteur de la source sonore par rapport au microphone est de 70 mètres.

La fréquence d'échantillonnage est de $f_s = 190$ Hz.

La vitesse de la source sonore est de 90 m/s.

5.1.1.2 Résultats de l'implémentation

Les résultats de l'estimation des paramètres avec le signal simulé sont sauvegardés dans un fichier dans la carte cible ainsi que transmis sur une interface WEB.

5.1.1.3 Résultats sauvegardés sur la carte

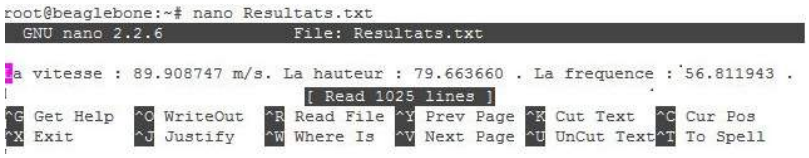


FIGURE 5.1: Résultats sauvegardés sur la carte Test

5.1.1.4 Résultats transmis à l'interface WEB

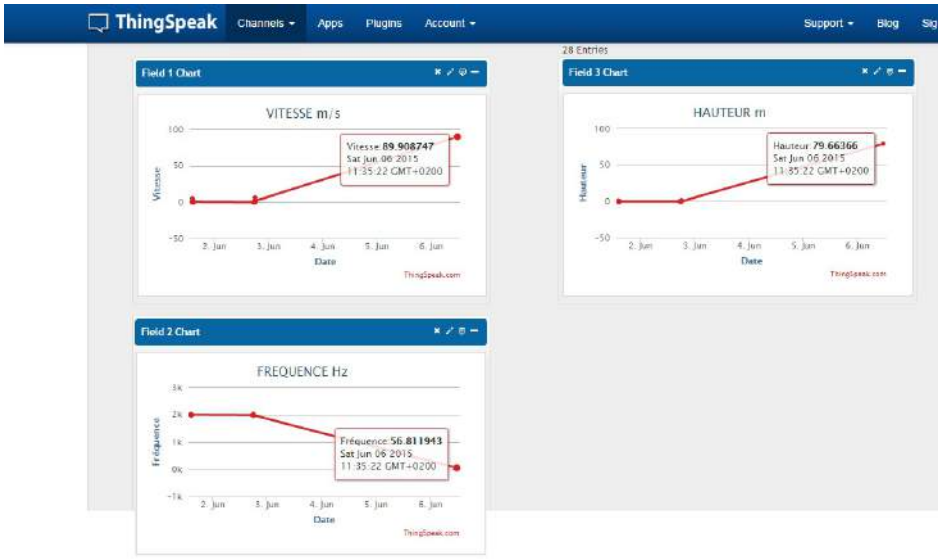


FIGURE 5.2: Résultats transmis à l'interface WEB Test 1

5.1.1.5 Calcul d'erreurs

Paramètres	Valeurs expérimentales	Notre approche	Erreur relative
Vitesse (m/s)	90	89.908	0.10 %
Hauteur (m)	70	79.663	13.80 %
Fréquence (Hz)	60	56.811	5.31 %

TABLE 5.1: Résultats du calcul des paramètres et du calcul d'erreur Test 1 Signal simulé

5.1.2 Test 2

5.1.2.1 Conditions

Fréquence d'émission de la source sonore simulée est de $f_0 = 200$ Hz.

Hauteur de la source sonore par rapport au microphone est de 70 mètres.

La fréquence d'échantillonnage est de $f_s = 8000$ Hz.

La vitesse de la source sonore est de 120 m/s.

5.1.2.2 Résultats de l'implémentation

5.1.2.3 Résultats sauvegardés sur la carte

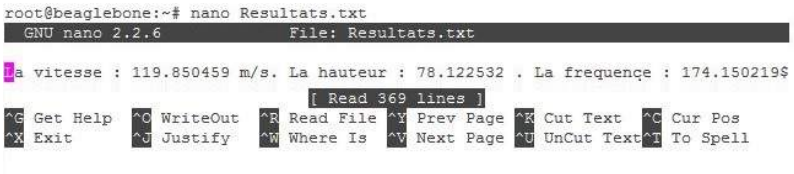


FIGURE 5.3: Résultats sauvegardés sur la carte Test

5.1.2.4 Résultats transmis à l'interface WEB

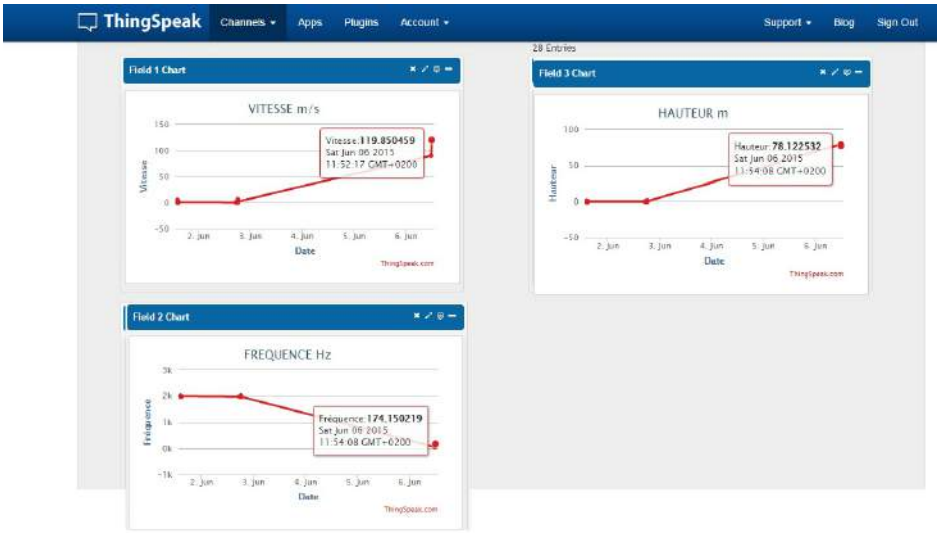


FIGURE 5.4: Résultats transmis à l'interface WEB Test 1

5.1.2.5 Calcul d'erreurs

Paramètres	Valeurs expérimentales	Notre approche	Erreur relative
Vitesse (m/s)	120	119.85	0.125 %
Hauteur (m)	70	78.12	11.6 %
Fréquence (Hz)	200	174.15	12.925%

TABLE 5.2: Résultats du calcul des paramètres et du calcul d'erreur Test 2 Signal simulé

5.1.3 Test 3

5.1.3.1 Conditions

Fréquence d'émission de la source sonore simulée est de $f_0 = 220$ Hz.

Hauteur de la source sonore par rapport au microphone est de 95 mètres.

La fréquence d'échantillonnage est de $f_s = 16000$ Hz.

La vitesse de la source sonore est de 110 m/s.

5.1.3.2 Résultats de l'implémentation

5.1.3.3 Résultats sauvegardés sur la carte

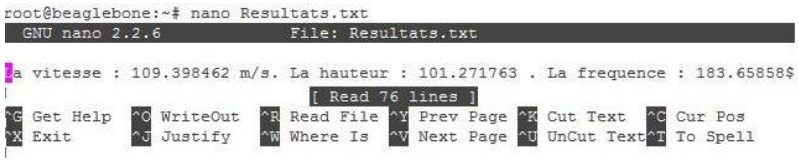


FIGURE 5.5: Résultats sauvegardés sur la carte Test 3

5.1.3.4 Résultats transmis à l'interface WEB

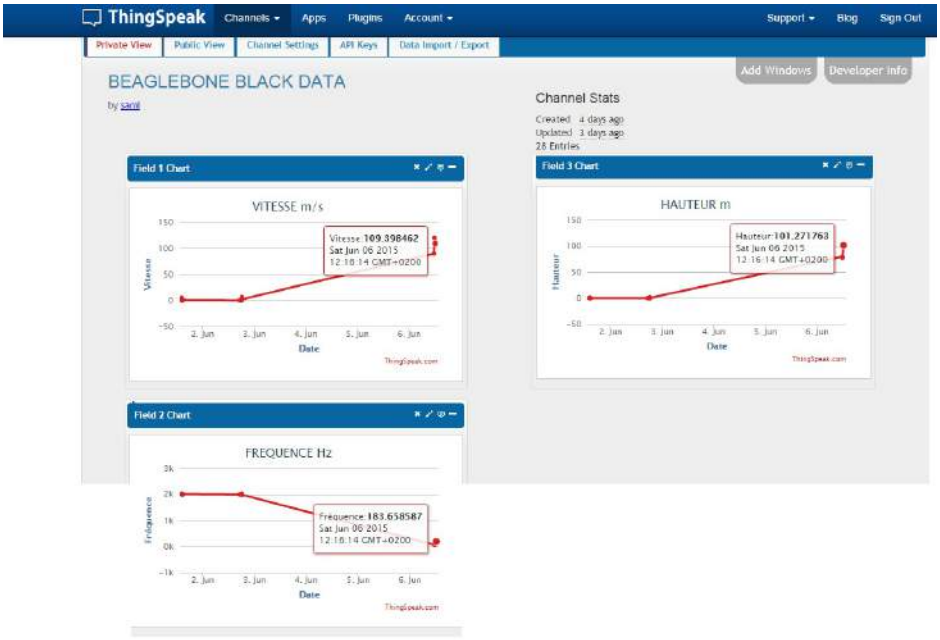


FIGURE 5.6: Résultats transmis à l'interface WEB Test 3 Signal simulé

5.1.3.5 Calcul d'erreurs

Paramètres	Valeurs expérimentales	Notre approche	Erreur relative
Vitesse (m/s)	110	109.398	0.547 %
Hauteur (m)	95	101.271	6.601 %
Fréquence (Hz)	220	183.658	16.519%

TABLE 5.3: Résultats du calcul des paramètres et du calcul d'erreur Test 3

5.2 Signaux réels

Dans cette partie, nous exposerons les résultats obtenus avec les deux plateformes (chambre sourde et champ libre).

5.2.1 Signaux issus de la chambre sourde

[24]

5.2.1.1 Test 1

5.2.1.2 Conditions

Fréquence d'émission de la source sonore simulée est de $f_0 = 2005$ Hz.

Hauteur de la source sonore par rapport au microphone est de 0.6 mètres.

La fréquence d'échantillonnage est de $f_s = 8000$ Hz.

La vitesse de la source sonore est de 1 m/s.

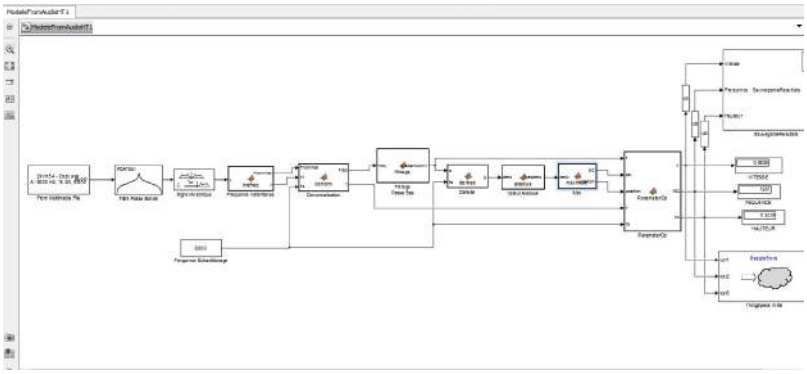


FIGURE 5.7: Modèle du traitement différé chambre sourde test 1

5.2.1.3 Résultats de l'implémentation du traitement différé

5.2.1.4 Résultats sauvegardés sur la carte

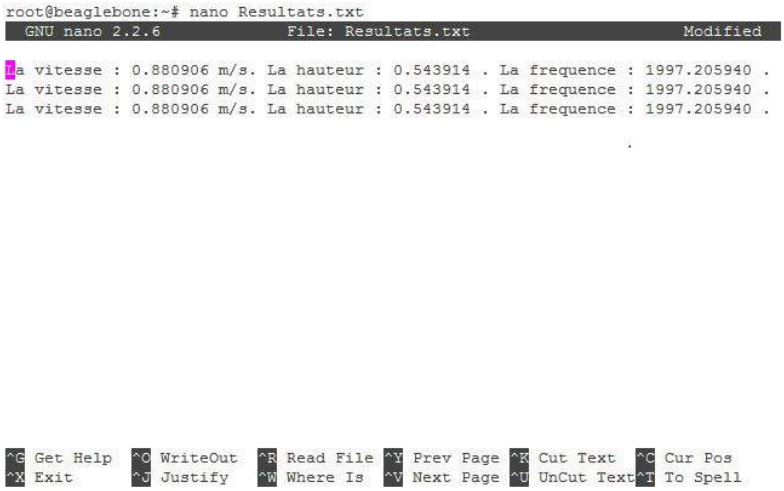


FIGURE 5.8: Résultats sauvegardés sur la carte traitement différé chambre sourde test 1

5.2.1.5 Résultats transmis à l'interface WEB

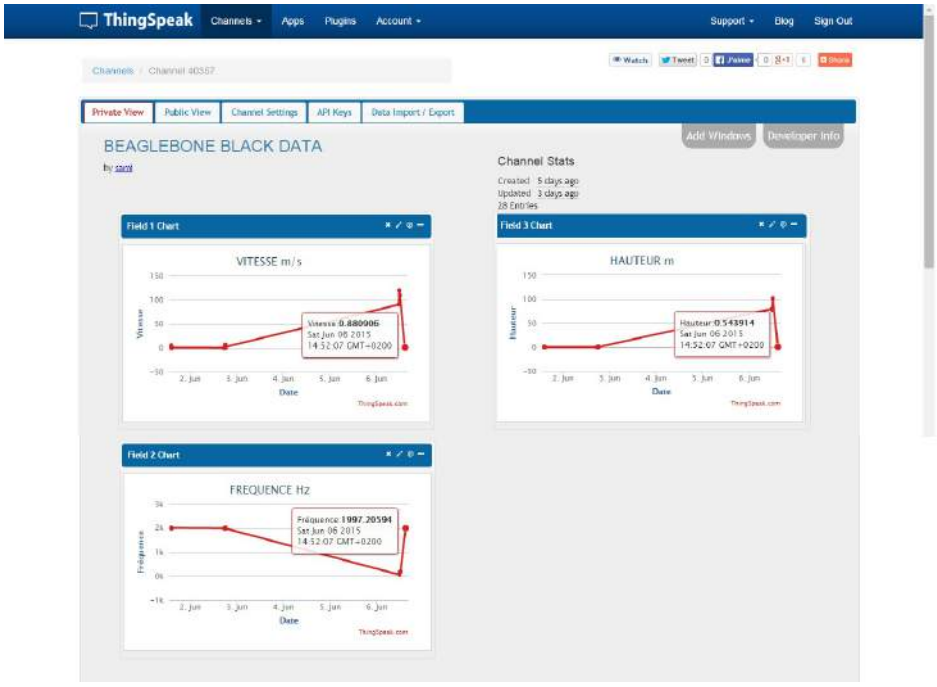


FIGURE 5.9: Résultats transmis à l'interface WEB traitement différé chambre sourde test 1

5.2.1.6 Calcul d'erreurs

Paramètres	Valeurs expérimentales	Notre approche	Erreur relative
Vitesse (m/s)	1	0.8809	11.91 %
Hauteur (m)	0.6	0.543	9.5 %
Fréquence (Hz)	2005	1997.205	0.388%

TABLE 5.4: Résultats du calcul des paramètres et du calcul d'erreur Test 1 Signal réel

5.2.1.7 Résultats de l'implémentation du traitement temps réel

La notion de temps réel est relative, dans notre application, il est impératif de percevoir le phénomène Doppler afin d'estimer les paramètres de mouvement de la source sonore. On définir donc un temps minimum d'acquisition, ce dernier est lié à la vitesse de déplacement de la source, plus elle est importante moins le temps de perception de l'effet Doppler est grand. Dans notre cas, étant restreint par les performances des plateformes expérimentales, on doit définir un temps d'acquisition au minimum égal à quatre secondes.

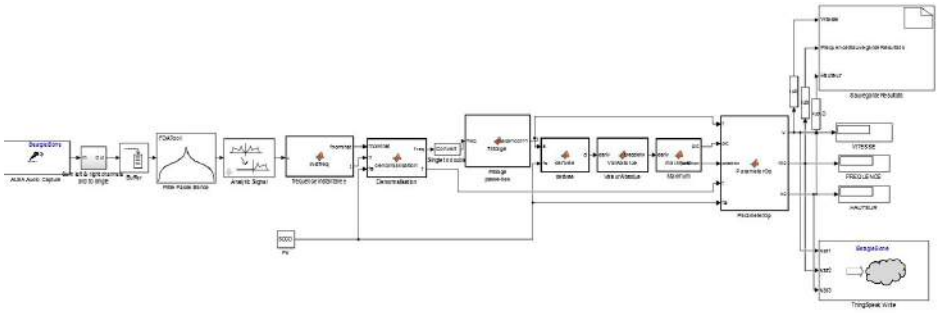


FIGURE 5.10: Modèle Simulink Temps réel

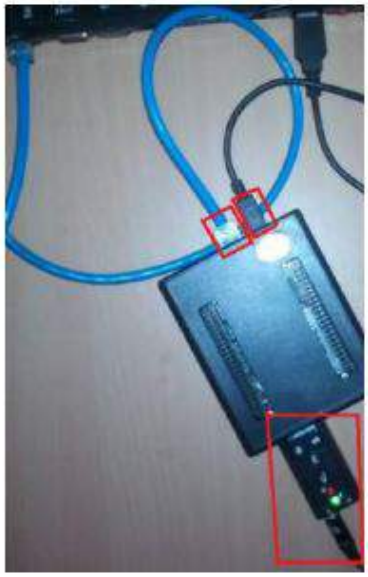


FIGURE 5.11: Branchement de la cible

5.2.1.8 Résultats de l'implémentation temps réel

On retrouve les mêmes résultats que dans le cas différé.

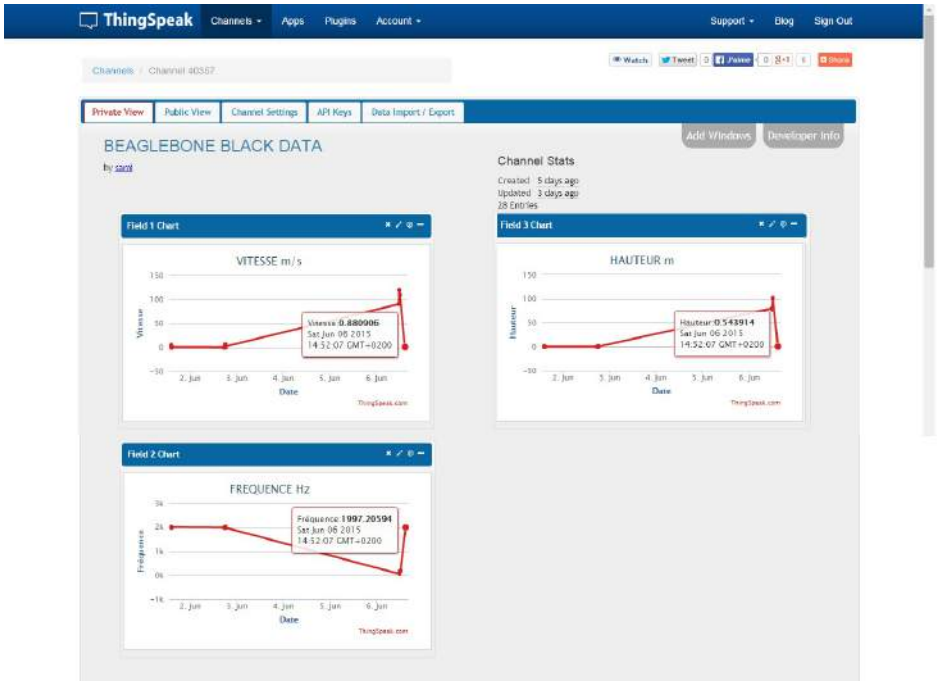


FIGURE 5.12: Résultats transmis à l'interface WEB traitement temps réel test 1

5.2.1.9 Observations

La figure ci dessous représente la fréquence instantanée estimée par la méthode directe et la fréquence instantanée prédite.

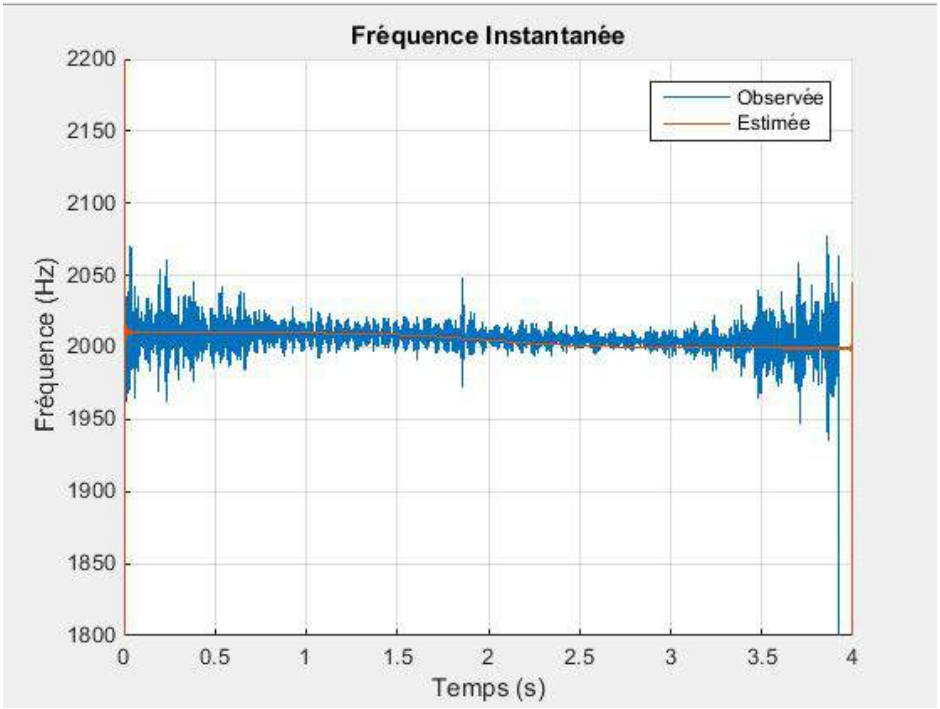


FIGURE 5.13: Courbes de la FI estimée et prédite Chambre sourde 1

5.2.1.10 Test 2

5.2.1.11 Conditions

Fréquence d'émission de la source sonore simulée est de $f_0 = 2005$ Hz.

Hauteur de la source sonore par rapport au microphone est de 0.6 mètres.

La fréquence d'échantillonnage est de $f_s = 8000$ Hz.

La vitesse de la source sonore est de 1.2 m/s.

5.2.1.12 Résultats de l'implémentation du traitement différé

5.2.1.13 Résultats sauvegardés sur la carte

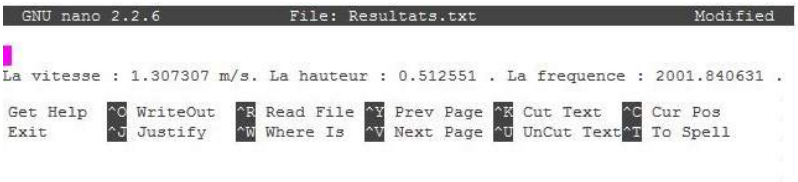


FIGURE 5.14: Résultats sauvegardés sur la carte traitement différé chambre sourde test 2

5.2.1.14 Résultats transmis à l'interface WEB

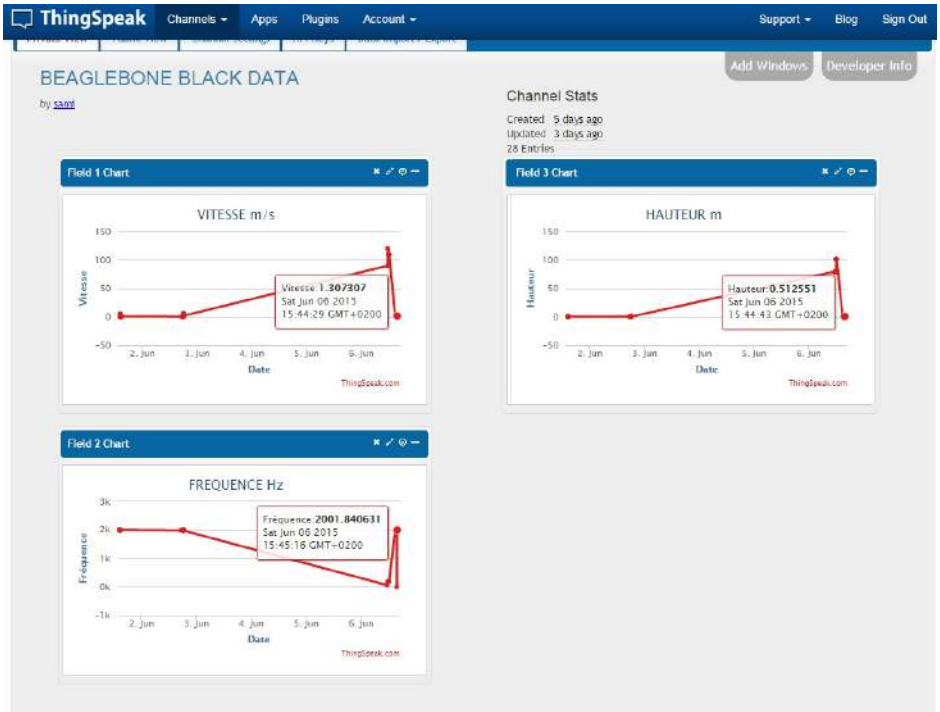


FIGURE 5.15: Résultats transmis à l'interface WEB traitement différé chambre sourde test 2

5.2.1.15 Calcul d'erreurs

Paramètres	Valeurs expérimentales	Notre approche	Erreur relative
Vitesse (m/s)	1.2	1.307	8.91 %
Hauteur (m)	0.6	0.5125	14.58 %
Fréquence (Hz)	2005	2001.84	0.1576%

TABLE 5.5: Résultats du calcul des paramètres et du calcul d'erreur Test 2 Signal réel

5.2.1.16 Résultats de l'implémentation du traitement temps réel

Le signal sonore est injecté dans l'entrée audio de la carte son. On obtient les mêmes valeurs que dans le cas du traitement différé.

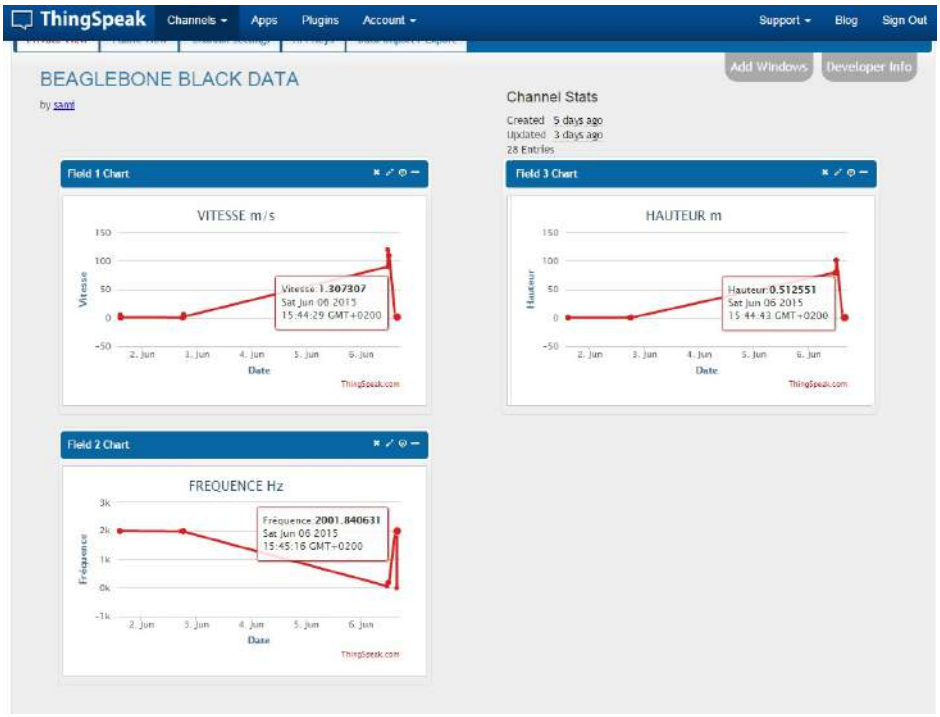


FIGURE 5.16: Résultats transmis à l'interface WEB traitement différé chambre sourde test 2

5.2.1.17 Observations

Les courbe de la fréquence instantanée observée et prédite sont illustrées dans la figure suivante :

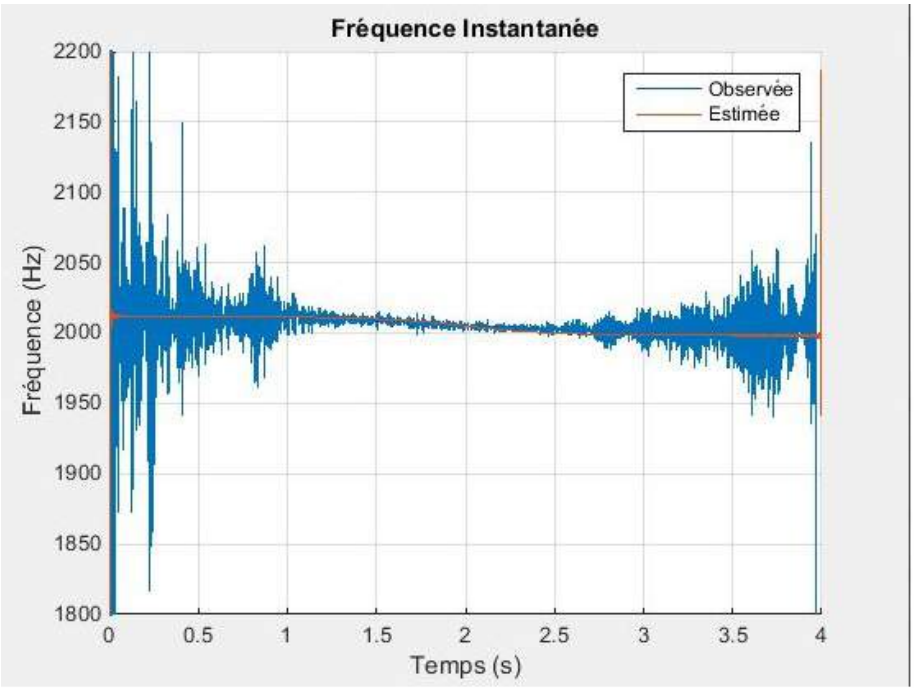


FIGURE 5.17: Courbes de la FI estimée et prédite Chambre sourde 2

5.2.2 Signal issu de la plateforme en champ libre

5.2.2.1 Traitement différé

5.2.2.2 Conditions

Fréquence d'émission de la source sonore simulée est de $f_0 = 3005$ Hz.

Hauteur de la source sonore par rapport au microphone est de 2.58 mètres.

La fréquence d'échantillonnage est de $f_s = 8000$ Hz.

La vitesse de la source sonore est de 3.5 m/s.

5.2.2.3 Résultats de l'implémentation du traitement différé

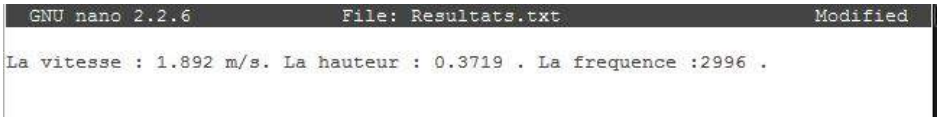


FIGURE 5.18: Modèle Simulink

5.2.2.4 Remarques et observations

On constate que les résultats de l'estimation sont très loin des valeurs réelles. Afin d'analyser ces erreurs, on trace la courbe de la fréquence instantanée prédite et celle de la FI observée.

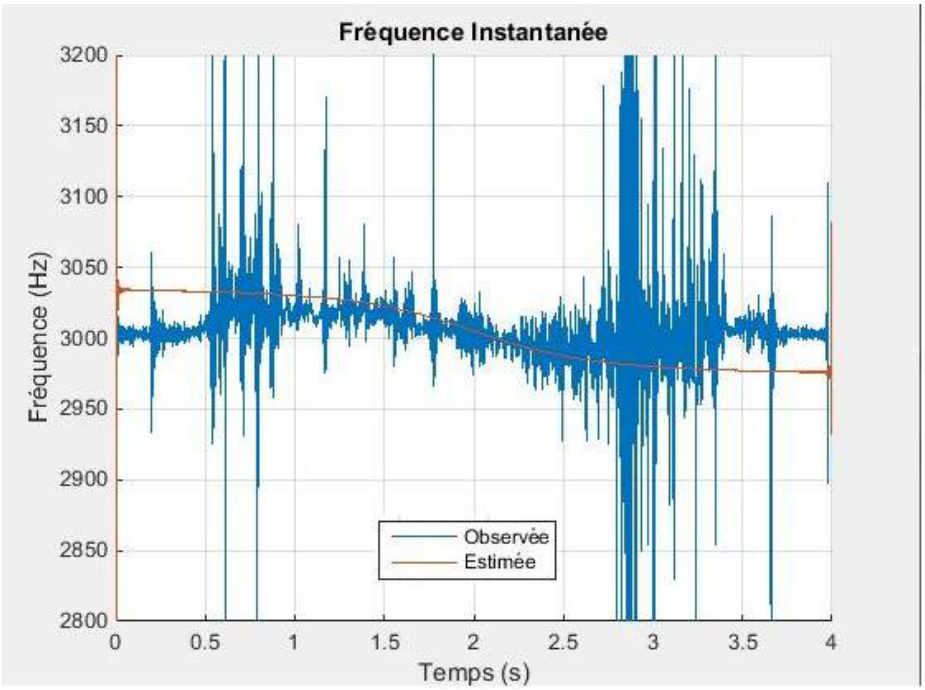


FIGURE 5.19: Courbes de la FI prédite et observée

On remarque que l'écart entre la courbe de la fréquence instantanée observée et celle de la FI prédite est très grand, cela s'explique par la forte présence de bruit ainsi que de réverbérations et réflexions sur le sol et sur les murs, en sachant que ceci sont très proche des rails.

5.3 Conclusion

Pour conclure on a montré dans ce chapitre le bon fondement de la démarche suivie pour l'implémentation de l'algorithme de localisation, les résultats de l'estimation de la vitesse hauteur et fréquence ont donné des valeurs correctes. L'estimation des paramètres en chambre libre a été critique en raison de la présence de bruits et de réverbérations dans la milieu.

Conclusion Générale

L'implémentation de l'algorithme de localisation acoustique sur système embarqué a été effectuée suivant la méthodologie "Model-Based Design", le modèle de l'algorithme de traitement a été présenté ainsi que le flot permettant le passage de l'algorithme au modèle de traitement et du modèle au code réalisant l'application.

L'environnement de développement ainsi que les cibles matérielles ont été présentées, les problématiques liés a l'interfaçage et l'acquisition du signal ainsi que la transmission des résultats ont été abordés enfin le passage du modèle fonctionnant sous Simulink vers les différents systèmes cible a été rendu possible par l'outil de génération automatique de code.

On a montré l'intérêt et l'avantage qu'offre l'approche basée sur la modélisation haut niveau couplée à la génération automatique de code d'autant plus lorsqu'il s'agit de faire du multi-ciblage (réduction du temps lié au développement, réduction des erreurs).

Les résultats obtenus pour l'estimation des paramètres sur système embarqué sont prometteurs et encouragent à poursuivre des initiatives dans le domaine de la localisation acoustique, les perspectives tels que l'utilisation de plusieurs capteurs pour le suivi de cible peuvent être envisagées.

Bibliographie

[1] J. H. Ku, "Alfred M. Mayer and Acoustics in Nineteenth-Century America," *Annals of Science*, vol. 70, no. 2, pp. 229–256, 2013.

[2] B. Kaushik, D. Nance, and K. K. Ahuja, "A Review of the Role of Acoustic Sensors in the Modern Battlefield+++," 2005.

[3] R. Morris, *The Origin of Radar*. Anchor Books, 1962.

[4] K. Toman, "Christian Doppler and the Doppler effect," *EOS Transactions*, vol. 65, no. 48, 1984.

[5] B. G. Ferguson and B. G. Quinn, "Application of the short-time Fourier transform and the Wigner–Ville distribution to the acoustic localization of aircraft," *The Journal of the Acoustical Society of America*, vol. 96, no. 2, pp. 821–827, 1994.

[6] B. G. Ferguson, "A ground-based narrow-band passive acoustic technique for estimating the altitude and speed of a propeller-driven aircraft," *The Journal of the Acoustical Society of America*, vol. 92, no. 3, pp. 1403–1407, 1992.

[7] F. Dommermuth and J. Schiller, "Estimating the trajectory of an accelerationless aircraft by means of a stationary acoustic sensor," *The Journal of the Acoustical Society of America*, vol. 76, no. 4, pp. 1114–1122, 1984.

[8] H. T. a. A. B. Y. Remram, "Design and implementation of an acoustic detection system of a moving sound source," *Article in International Metrology conference CAFMET*, 2012.

[9] M. Raulet, C. MOY, and M. A. à Supélec, "Optimisations mémoire dans la méthodologie aaa pour code embarqué sur architectures paralleles," *Electronique et traitement du signal, INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE RENNES*, 2006.

[10] D. D. Gajski, S. Abdi, A. Gerstlauer, and G. Schirner, *Embedded System Design : Modeling, Synthesis and Verification*. Springer Science & Business Media, 2009.

[11] J. Friedman, "L'approche Model-Based Design garantit qualité et réduction des délais et coûts," *The MathWorks*, Oct. 2010.

[12] J. C. Jensen, D. H. Chang, and E. A. Lee, "A model-based design methodology for cyber-physical systems," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pp. 1666–1671, IEEE, 2011.

[13] J. Lin, "Measuring return on investment of model-based design," *EE Times Design*, 2011.

[14] J. Jussel and C. Sullivan, "Software-Compiled System Design : A Methodology for Field-Programmable System-on-Chip Design," *Celoxica Ltd., Abingdon, Oxfordshire, UK*, 2003.

[15] J. C. Jensen, D. H. Chang, and E. A. Lee, "A model-based design methodology for cyber-physical systems," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, pp. 1666–1671, IEEE, 2011.

[16] "F. auger, july 94, august 95 - o. lemoine, october 95 | EE Times."

[17] "F. auger, march 1994, july 1995 | EE Times."

[18] "H timlelt and a belouchrani | EE Times."

[19] A. m. Microcomputers and R. Pi, "New records–27 November 2013,"

[20] G. Coley, "Beaglebone black system reference manual," *Texas Instruments, Dallas*, 2013.

[21] "HARDWARE USER 'S GUIDE - ZEDBOARD."

[22] "Automatic Code Generation - Simulink Coder."

[23] A. BOUZID, "Conception et réalisation d'une plateforme pour la localisation acoustique d'une source sonore avec un réseau de capteurs rf," 2013.

[24] "Y.remram and h.kelladi, construction of an inexpensive anechoic chamber for teaching aid, article in congress inter-noise, 2009.."

Annexe A

Plateforme expérimentale

Dans le cadre de l'étude des performances de l'application développée pour la localisation de sources sonores, une plateforme en champ libre a été conçue afin d'approcher le cas réel, simuler le passage d'une source sonore et ainsi de pouvoir vérifier et valider les résultats de l'analyse théorique. La structure mécanique à été montée au niveau de la terrasse du département d'électronique.

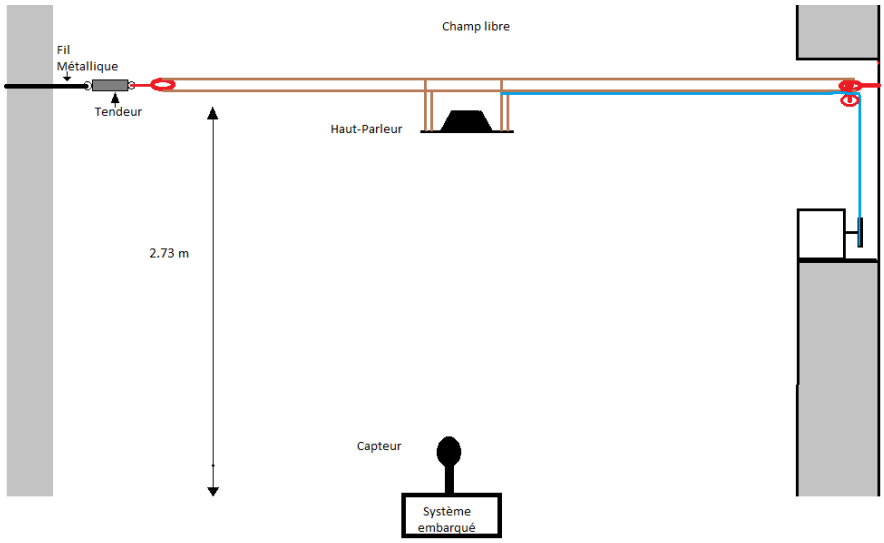


FIGURE A.1: Plateforme d'expérimentation

A.1 Structure mécanique

Le montage expérimental en champ libre repose sur un mécanisme simple composé d'une corde en chanvre suspendue à 2.73 mètres du sol, celle ci forme une boucle qui passe d'un côté par attache encrée dans le mur, et de l'autre par une poulie fixée à un poteau ,la poulie servant simplement d'attache. La corde formant ainsi un support pour le chariot portant le haut-parleur, et lui sert également de guide sur une distance de 13 mètres.

FIGURE A.2: Rails du montage

La longueur de la corde nous permet d'atteindre des vitesses élevées. Pour éviter les inflexions et maintenir une hauteur fixe le long du déplacement du chariot, un tendeur est utilisé (tire corde)

FIGURE A.3: Tendeur

L'expérimentation est un processus qui doit être effectué avec rigueur et délicatesse, pour cause de présence d'artefacts comme les réflexions et les bruits qui s'y propagent librement. Notamment, les valeurs mesurées dites exactes sont toujours imprécises : la distance du microphone à la corde, la vitesse linéique de la corde sont toujours des sources d'erreurs de mesure inévitables.

FIGURE A.4: Plateforme d'essais

A.2 Instrumentation

A.2.1 Haut-Parleur

Dans le but de simuler le passage d'une source sonore, on dispose d'un haut-parleur d'une puissance de 3 Watts, qui constitue la source sonore rayonnante à localiser, le haut-parleur a été fixé sur le chariot à l'aide de quatre vices. Le haut-parleur est alimenté par un générateur de basses fréquences.

FIGURE A.5: Haut-Parleur

Caractéristiques du haut-parleur :

Puissance moyenne	3 W
Puissance maximum	4 W
Impédance	4 Ohm
Réponse fréquentielle	-10dB 150-17000 Hz
Moyenne niveau d'intensité sonore	82dB
Déplacement maximun du cône	2mm
Poids	60g

TABLE A.1: Caractéristiques du haut-parleur

A.2.2 Générateur Basses Fréquences

L'alimentation utilisée pour le haut-parleur est un générateur de basses fréquences

FIGURE A.6: Générateur Basses Fréquences

A.2.3 Moteur

Le moteur utilisé est un moteur à courant-continu de marque AMETEK (Aiment permanent moteur DC 50 Vdc Ametek OIA E56617 110).

Tension nominale	50 VDC
Ampérage à vide	0.55 A
Diamètre	10 cm
Longueur	20 cm
Tours par minute	1700 tr/mn à 50 VDC
Poids	5.5 Kg

TABLE A.2: Caractéristiques du moteur

FIGURE A.7: Moteur à courant-continue

Caractéristiques du moteur :

A.2.4 Alimentation du moteur

L'alimentation utilisée est une alimentation stabilisée capable de délivrée une tension allant jusqu'à 30 VDC.

FIGURE A.8: Alimentation stabilisée

Bande Radio Fréquence	798-822 Mhz
Stabilité de la fréquence	0.005% à 25 C
Rapport Signal/Bruit	90 dB
Bande de Fréquence	50 Hz - 15 KHz
Sorties	Jack et XLR
Alimentation récepteur	DC 12 V / 500 mA
Puissance sortie RF	10 mW
Diagramme polaire	Cardioïde

TABLE A.3: Caractéristiques du microphone

A.2.5 Microphone

L'application détermine le choix des capteurs, c'est pour cela que dans notre conception de l'application, on a opté pour un microphone sans fil de type Proel RMW 10, afin d'éviter les désagréments des câblages.

Ce microphone travaille dans la bande UHF (Ultra High Frequency) et possède une portée de transmission RF allant jusqu'à 50 mètres.

Autre paramètre important du choix du microphone est sa directivité, le microphone Proel RMW 10 est un microphone cardioïde. Sachant que dans notre application la source sonore se déplace au-dessus du microphone, la directivité de ce dernier permet une bonne perception du signal et une minimisation considérable des réflexions provenant du sol.

Caractéristiques :

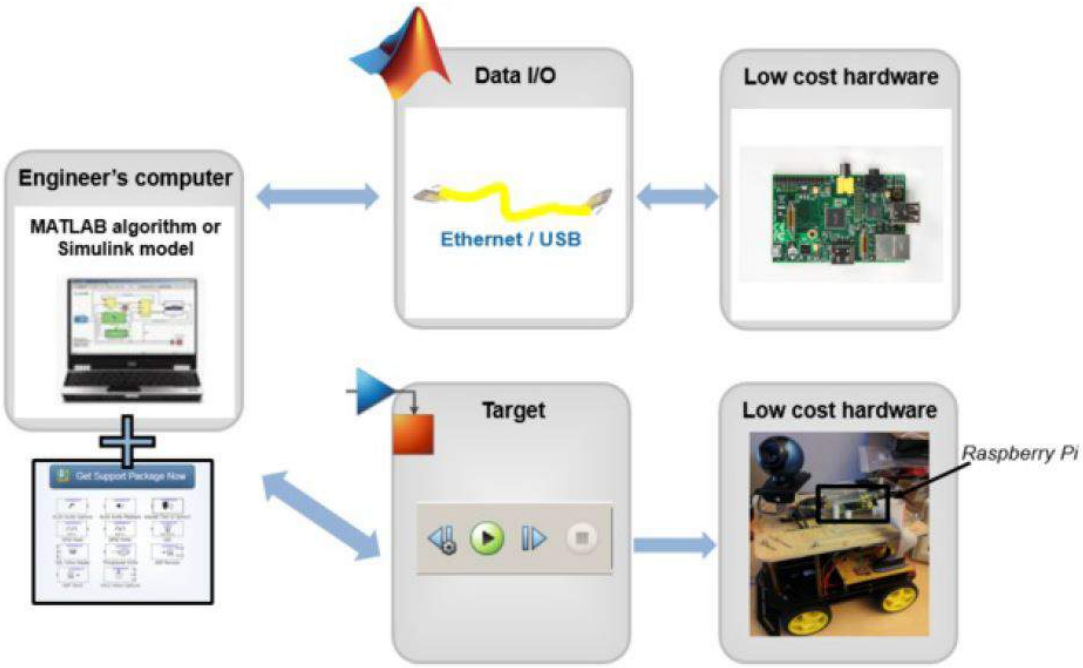
FIGURE A.9: Microphone

Annexe B

Installation des supports Simulink pour les cibles

B.1 Raspberry Pi

Vue d'ensemble :



Raspberry Pi peut être programmé pour exécuter modèles Simulink, en utilisant Simulink Support Package pour Raspberry Pi. Ces librairies génèrent du code à partir d'un modèle Simulink en un clique, qui sera ensuite exécuté sur Raspberry Pi.

Les éléments nécessaires :

Logiciel :

- Matlab et Simulink.

Matériels :

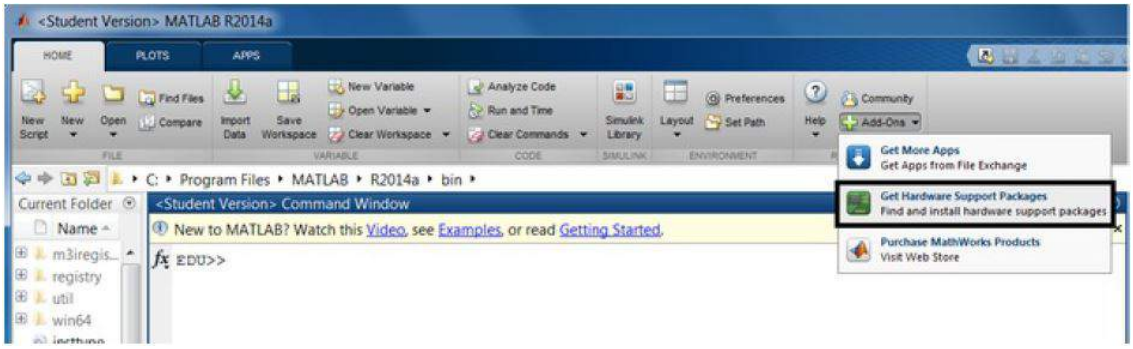
— Raspberry Pi.
— Cable USB.

— Cable Ethernet.

Installation du Support Package pour Raspberry Pi :

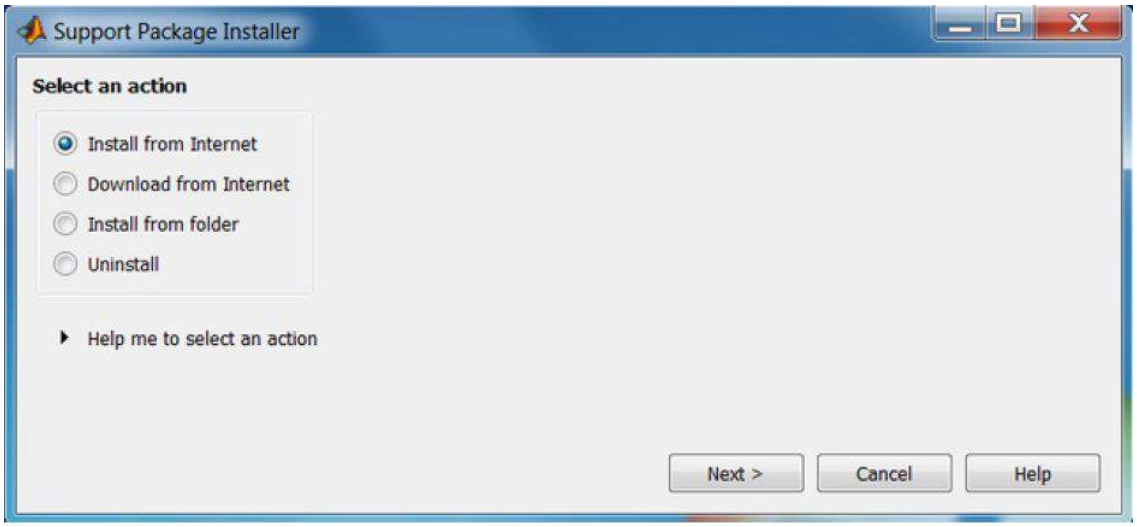
- Étape 1 :

- Ouvrir Matlab et lancer l'option *Get Hardware Support Package* sous l'onglet *Add-Ons*.



- Étape 2 :

- Le guide d'installation des support packages se lance. - Choisir *Install from Internet*



- Étape 3 :

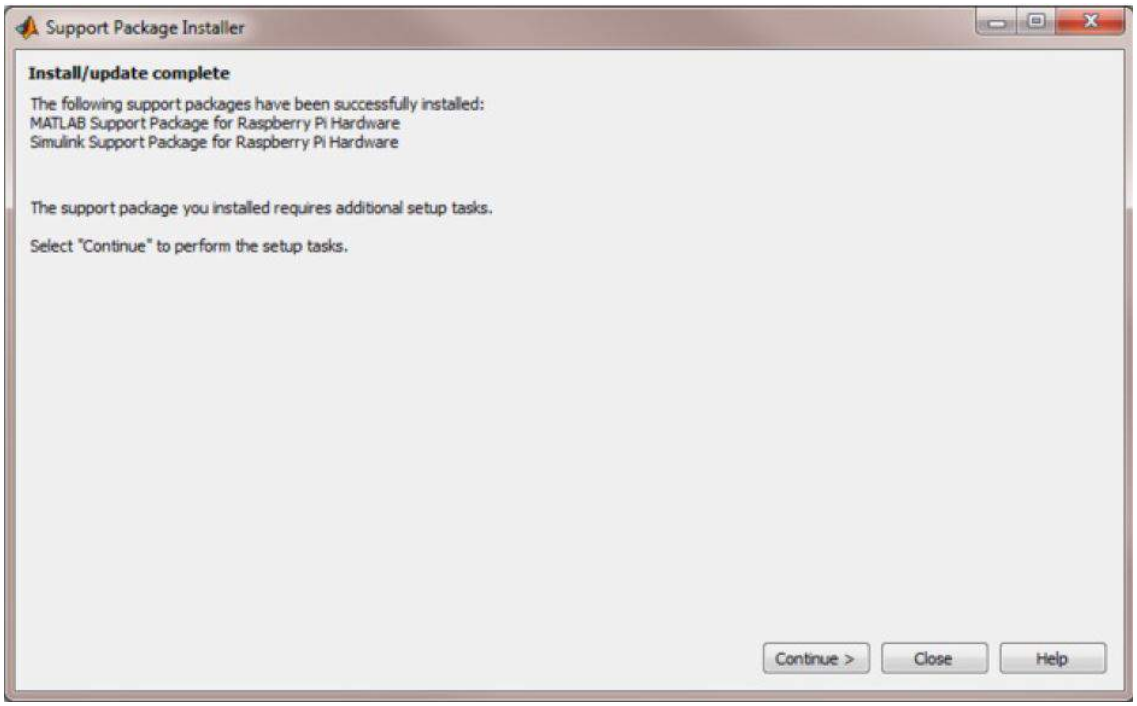
- Sélectionner Raspberry Pi support package de la liste des support packages, suivant.

- Étape 4 :

- Dans cette étape, on introduit les identifiants du compte Mathworks.

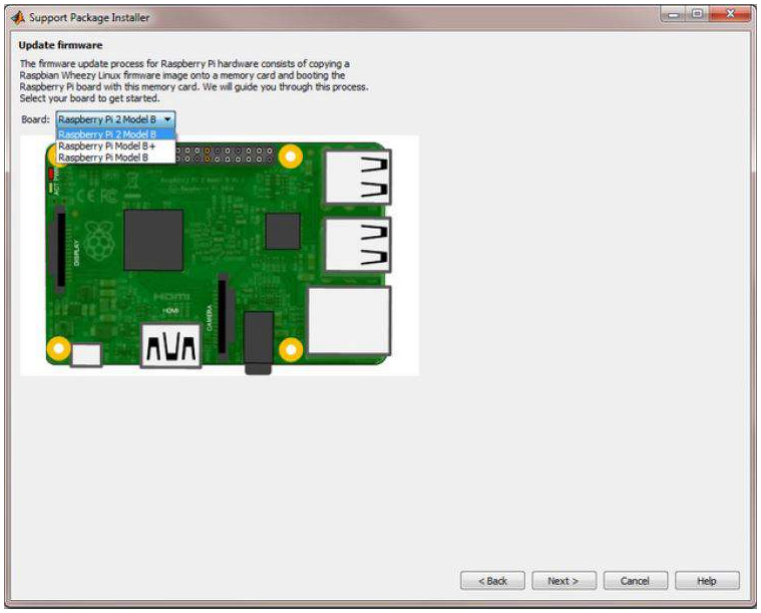
- Étape 5 :

- Accepter la licence dans l'écran suivant pour finaliser l'installation du Support Package.

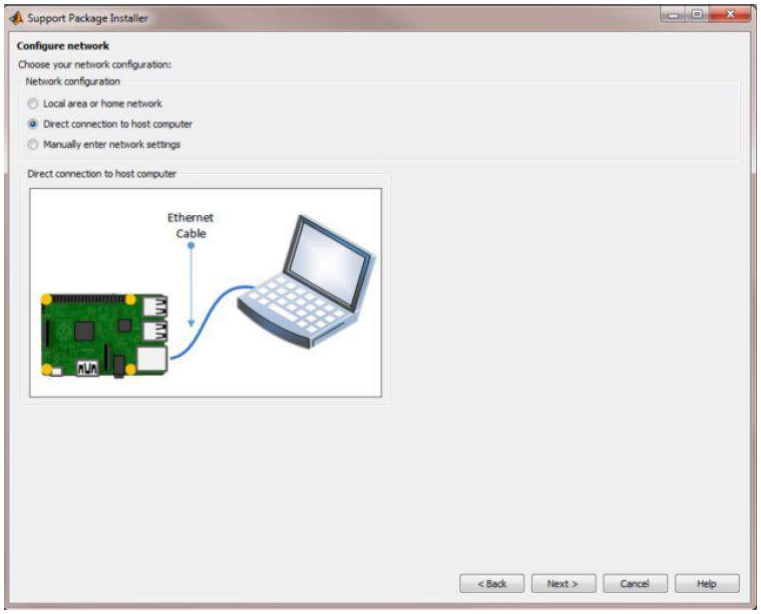


- Étape 6 :

- Sur l'écran suivant, sélectionner la carte utilisée (dans notre cas Raspberry Pi 2 Model B). Cette étape consiste à copier l'image Linux Raspbian Wheezy sur la carte mémoire de la Pi .

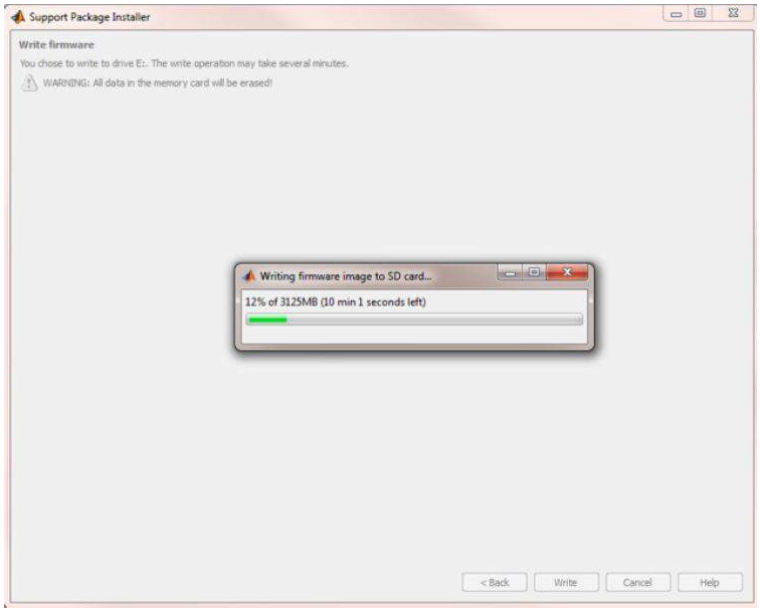


- Sur l'écran suivant, on configure la connexion réseau avec Raspberry Pi. Sélectionner *Direct connection to host computer*.



- A présent, Matlab doit effectuer l'écriture de l'image Linux sur la carte mémoire de la Pi. Pour celà on introduit la carte SD dans le lecteur de carte SD de la machine et on lance l'écriture.

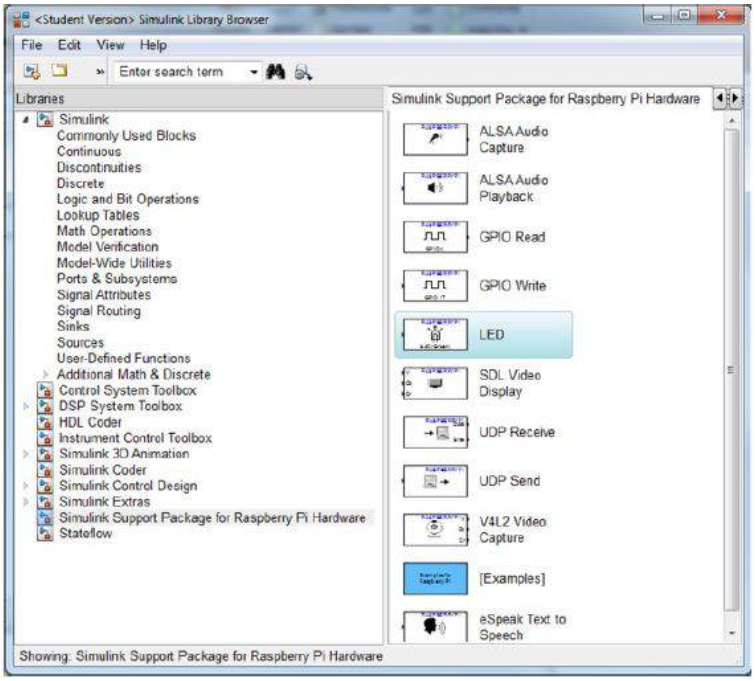
- Une fois que l'écriture de Raspbian sur la carte SD est terminée, on introduit la carte SD sur la Raspberry.



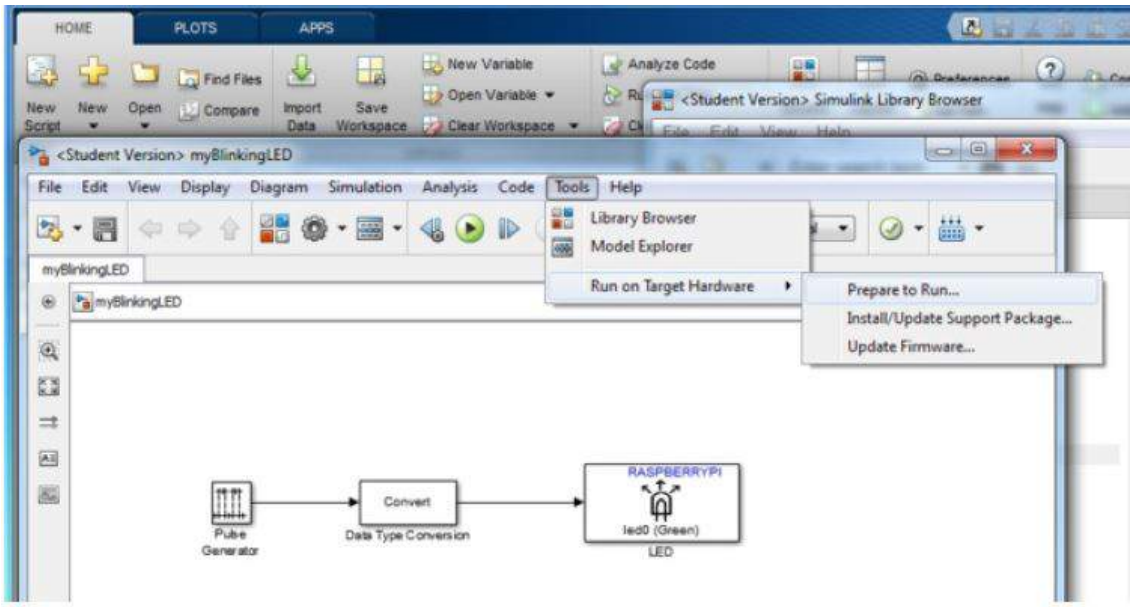
- Étape 7 :

L'utilisation du *Support Package Raspberry Pi* se fait de la manière suivante :

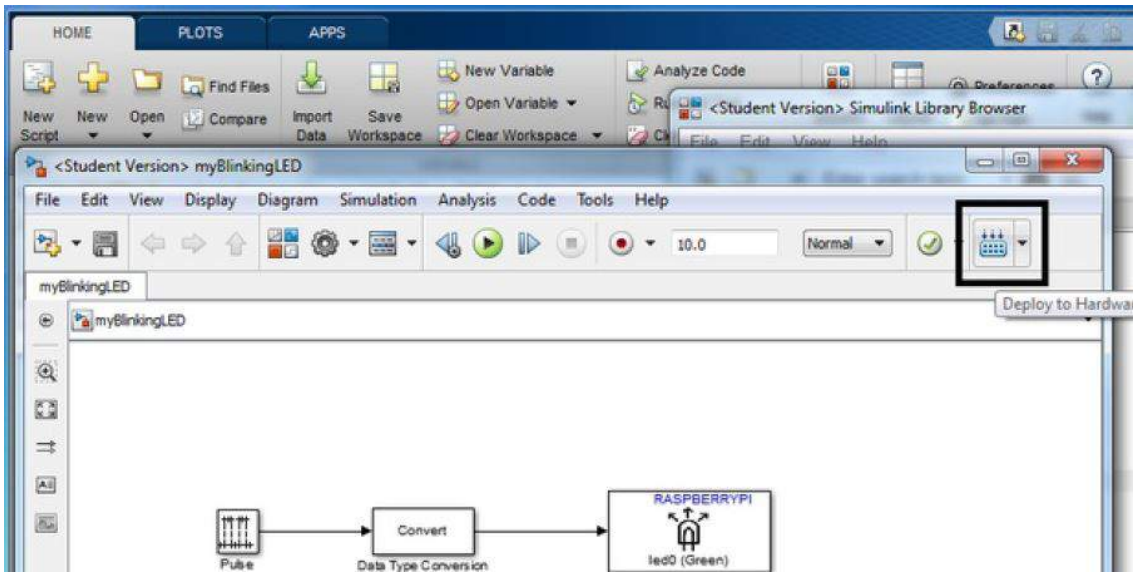
- Ouvrir Simulink et créer un nouveau modèle.
- Ouvrir la librairie *Support Package Raspberry Pi* et l'utiliser pour la création du modèle Simulink.



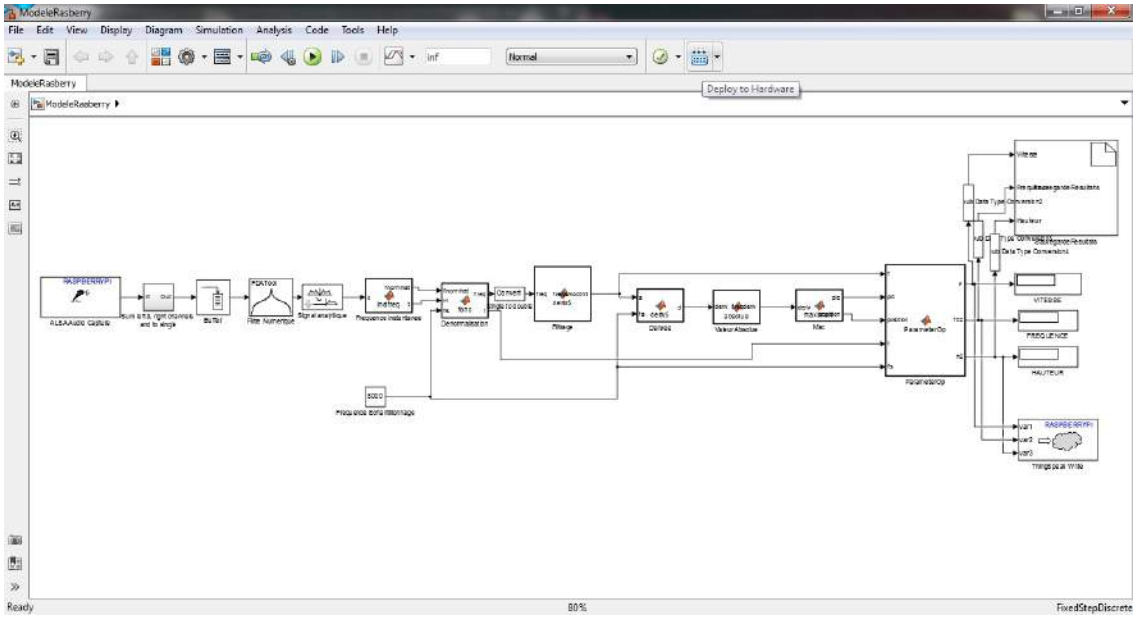
- Une fois que le modèle est prêt, on le configure pour l'exécution sur la Rasberry Pi. On ouvre le panneau de configuration *Run on Target Hardware* dans outils (Tools).



- Choisir Raspberry Pi dans *Target Hardware*.



- A présent le modèle est configuré pour Raspberry Pi, afin de le lancer sur la carte, on va sur *Deply To Hardware* présent dans la barre d'outils. Le code sera généré et déployé sur la Pi.



B.2 BeagleBone Black & ZedBoard

La même démarche vue précédemment est suivie, à la différence près de choisir les supports packages *BeagleBone Black* et *Xilinx Zynq 7000*.

Annexe C

Acquisition du signal via une carte son USB

La carte son USB utilisée est carte grand public, nommée 7.1 Channel USB External Sound Card Audio Adapter. Les raisons pour lesquelles cette carte a été choisi sont les suivantes :

FIGURE C.1: Carte son USB

— Peu couteuse

— Consommation réduite

— Taille restreinte

— Supportée par Linux

La carte son utilise un chipset C-Media audio, qui est supporté par ALSAproject sous Linux.

Configuration pour Raspberry Pi :

FIGURE C.2: Raspberry Pi Carte & carte son USB

Étape 1 :

- Brancher la carte son au port usb de Raspberry Pi. - Allumer Raspberry Pi. - Acceder à distance via Putty. - Vérifier que la carte son usb a été detecté, commande : lsusb.

```
pi@raspberrypi ~ $ lsusb
Bus 001 Device 002: ID 0424:9512 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp.
Bus 001 Device 004: ID 0d8c:000c C-Media Electronics, Inc. Audio Adapter
pi@raspberrypi ~ $
```

FIGURE C.3: Raspberry Pi Carte & carte son USB

La commande lsusb affiche les clés usb connectées, on peut voir la carte son C-Media Electronics, Inc. Audio Adapter.

Étape 2 :

- Afficher la carte son actuellement utilisée, en entrant la commande amixer.

```
pi@raspberrypi ~ $ amixer
Simple mixer control 'PCM',0
  Capabilities: pvolume pvolume-joined pswitch pswitch-joined penum
  Playback channels: Mono
  Limits: Playback -10239 - 400
  Mono: Playback -325 [93%] [-3.25dB] [on]
pi@raspberrypi ~ $
```

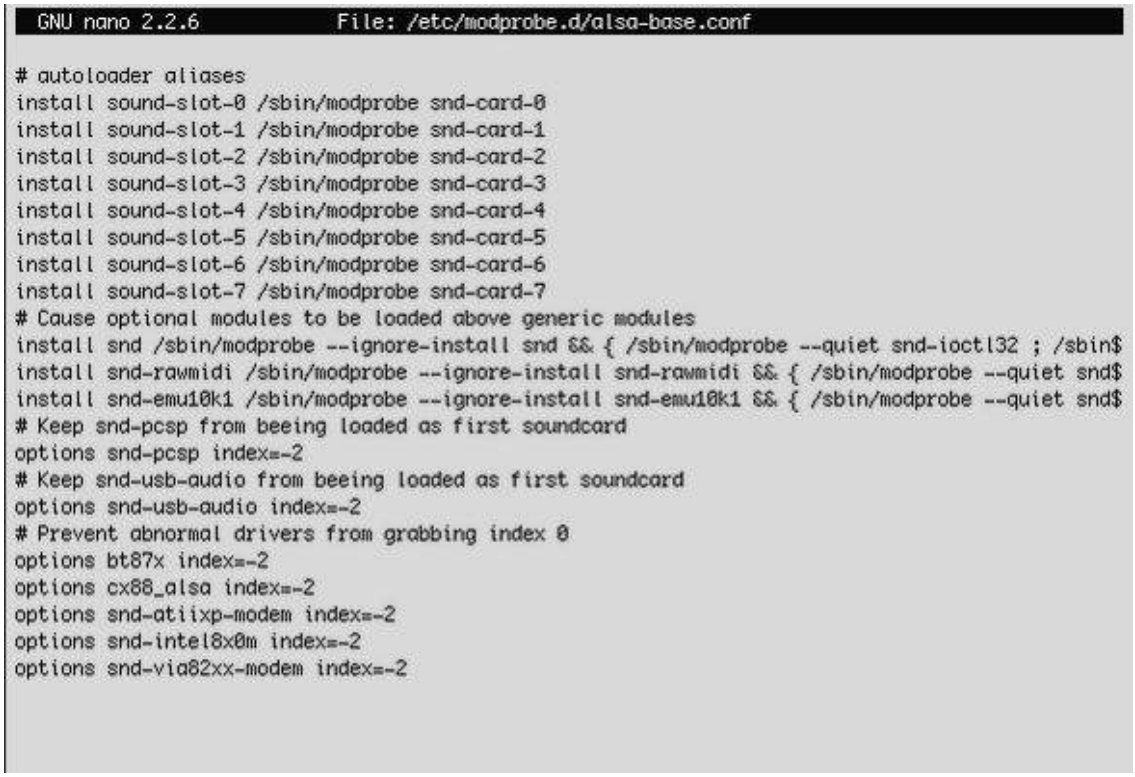
FIGURE C.4: Résultat de la commande amixer

Cette commande donne les informations concernant la carte audio par défaut du système.
Étape 3 :

- Configurer la carte son usb par défaut. Pour cela, une modification d'un fichier de configuration est nécessaire afin que la carte son usb soit utilisée par défaut et allumée au démarrage de Raspberry Pi.

- Le fichier à modifier est alsa-base.conf. Ce fichier contrôle la gestion des cartes audio connectées, ce fichier est modifié de telle sorte à définir la carte son usb la première à être utilisée. La commande à entrer est :

$ sudo nano /etc/modprobe.d/alsa-base.conf



options snd-usb-audio index=-2

```
  GNU nano 2.2.6          File: /etc/modprobe.d/alsa-base.conf

# autoloader aliases
install sound-slot-0 /sbin/modprobe snd-card-0
install sound-slot-1 /sbin/modprobe snd-card-1
install sound-slot-2 /sbin/modprobe snd-card-2
install sound-slot-3 /sbin/modprobe snd-card-3
install sound-slot-4 /sbin/modprobe snd-card-4
install sound-slot-5 /sbin/modprobe snd-card-5
install sound-slot-6 /sbin/modprobe snd-card-6
install sound-slot-7 /sbin/modprobe snd-card-7
# Cause optional modules to be loaded above generic modules
install snd /sbin/modprobe --ignore-install snd && { /sbin/modprobe --quiet snd-ioctl32 ; /sbin$
install snd-rawmidi /sbin/modprobe --ignore-install snd-rawmidi && { /sbin/modprobe --quiet snd$
install snd-emu10k1 /sbin/modprobe --ignore-install snd-emu10k1 && { /sbin/modprobe --quiet snd$
# Keep snd-pcsp from beeing loaded as first soundcard
options snd-pcsp index=-2
# Keep snd-usb-audio from beeing loaded as first soundcard
# options snd-usb-audio index=-2
# Prevent abnormal drivers from grabbing index 0
options bt87x index=-2
options cx88_alsa index=-2
options snd-atiixp-modem index=-2
options snd-intel8x0m index=-2
options snd-via82xx-modem index=-2
```

Étape 5 :

- Reboot du Raspberry Pi.
$ sudo reboot
Étape 6 :

- Lancer la commande amixer pour vérifier que la carte son usb est définie par défaut.

```
pi@raspberrypi ~ $ amixer
Simple mixer control 'Speaker',0
  Capabilities: pvolume pswitch pswitch-joined penum
  Playback channels: Front Left - Front Right
  Limits: Playback 0 - 151
  Mono:
  Front Left: Playback 96 [64%] [-10.38dB] [on]
  Front Right: Playback 96 [64%] [-10.38dB] [on]
Simple mixer control 'Mic',0
  Capabilities: pvolume pvolume-joined cvolume cvolume-joined pswitch pswitch-joined cswitch csw
itch-joined penum
  Playback channels: Mono
  Capture channels: Mono
  Limits: Playback 0 - 32 Capture 0 - 16
  Mono: Playback 23 [72%] [34.36dB] [off] Capture 0 [0%] [0.00dB] [on]
Simple mixer control 'Auto Gain Control',0
  Capabilities: pswitch pswitch-joined penum
  Playback channels: Mono
  Mono: Playback [on]
pi@raspberrypi ~ $
```

On remarque qu'on a bien la carte son usb, et notamment les informations concernant l'enregistrement audio.

Étape 7 :

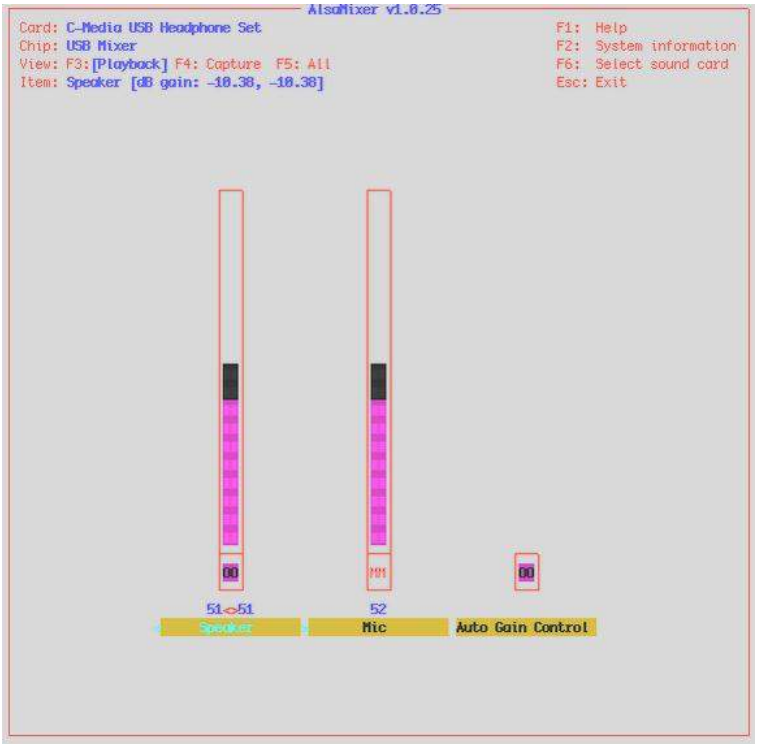
- Tester l'audio, en lançant la lecture d'un morceau sonore à l'aide de la commande suivante :

$ aplay /usr/share/scratch/Media/Sounds/Vocals/Singer1.wav

```
pi@raspberrypi ~ $ aplay /usr/share/scratch/Media/Sounds/Vocals/Singer1.wav
Playing WAVE '/usr/share/scratch/Media/Sounds/Vocals/Singer1.wav' : Signed 16 bit Little Endian,
 Rate 11025 Hz, Mono
pi@raspberrypi ~ $
```

On remarque que le niveau du son est faible, on ajuste le volume.

$ alsamixer



Configuration pour BeagleBone Black :

Étape 1 :

- Allumer BeagleBone Black et y accéder à l'aide de Putty.
- Lancer les commandes ci-dessous pour mettre à jour le système.

$ sudo apt-get update
$ sudo apt-get upgrade -y
$ cd /opt/scripts/tools/
$ git pull
$ sudo ./update_kernel.sh
$ sudo reboot

Étape 2 :

Désactiver la carte son HDMI :
- Lancer la commande suivante :
$ nano /boot/uEnv.txt
- Supprimer le caractère # de la ligne suivante :

#cape_disable=capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN

```
  GNU nano 2.2.6            File: /boot/uEnv.txt

#Docs: http://elinux.org/Beagleboard:U-boot_partitioning_layout_2.0

uname_r=3.8.13-bone71
#dtb=
cmdline=quiet init=/lib/systemd/systemd

##Example
#cape_disable=capemgr.disable_partno=
#cape_enable=capemgr.enable_partno=

##Disable HDMI/eMMC
#cape_disable=capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN,BB-BONE-EMM

##Disable HDMI
cape_disable=capemgr.disable_partno=BB-BONELT-HDMI,BB-BONELT-HDMIN

##Disable eMMC
#cape_disable=capemgr.disable_partno=BB-BONE-EMMC-2G

                          [ Read 28 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
```

- Redémarrer la carte.

Étape 3 :

- Vérifier que la carte son a été détecté, en lançant la commende :

$ aplay -l
- Le résultat figure ci-dessous :

**** List of PLAYBACK Hardware Devices **** card 1 : Headset [Logitech USB Headset], device 0 : USB Audio [USB Audio] Subdevices : 1/1 Subdevice #0 : subdevice #0
Étape 4 :

- Définir la carte son usb pour qu'elle soit prise en compte par défaut. Pour cela une modification du fichier alsa-base.conf est nécessaire.
- Commenter la ligne ci-après en ajoutant le caractère # en début de ligne.
options snd-usb-audio index=-2

- Redémarrer le pilote ALSA.

sudo alsa force-reload

```
root@beaglebone:/# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: Device [USB PnP Sound Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
root@beaglebone:/#
```

Étape 5 :

- Tester l'enregistrement.

$ arecord temp.wav

- Tester la sortie audio.

$ aplay temp.wav

Annexe D

Interfaçage pour l'application sous X86 (Station de travail) avec un microphone

Simulink propose un bloc qui permet de travailler et de traiter des signaux numériques, ce bloc s'appelle From Audio Device. Ce dernier lit les données audio à partir de la carte son du système en temps réel.

FIGURE D.1: From Audio Device

Le paramètre Device spécifie la carte son à partir de laquelle l'acquisition audio est faite. Ce paramètre pointe automatiquement la carte son par défaut du système, si un microphone est monté ou éjecté, la liste du matériel audio du système doit être mise à jour en entrant la commande clear mex dans le MATLAB Command.

Le paramètre Number of channels spécifie le nombre de canaux audio du signal, par exemple on entre le nombre 2 si la source audio est à deux canaux (stéréo).

La fréquence d'échantillonnage peut notamment être spécifier dans le paramètre Sample rate (hz).

Le bloc From Audio Device reçoit les données du matériel audio suivant le processus illustré par la figure suivante :

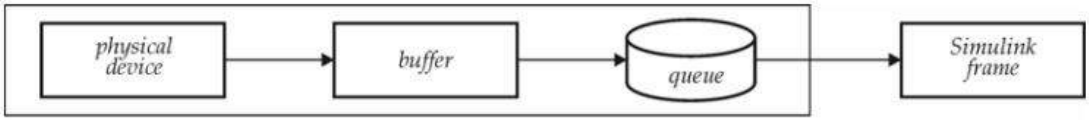


FIGURE D.2: Principales étapes régissant le fonctionnent du bloc From Audio Device

- Au lancement de l'exécution du modèle, la carte son commence à écrire les données entrantes dans une mémoire tampon (buffer). Ces données sont du type spécifié dans le

paramètre Device data type.

- Quand la mémoire tampon est remplie, le bloc From Audio Device écrit le contenu de la mémoire tampon dans la file d'attente (queue), sa taille est spécifiée par le paramètre Queue duration (secondes).

- La carte son ajoute les données audio au bas de la file d'attente, tandis que le bloc From Audio Device extrait les données de la partie supérieure de la file d'attente afin de les envoyer au modèle de traitement Simulink. Ces données sont du type spécifié par le paramètre Output data type.

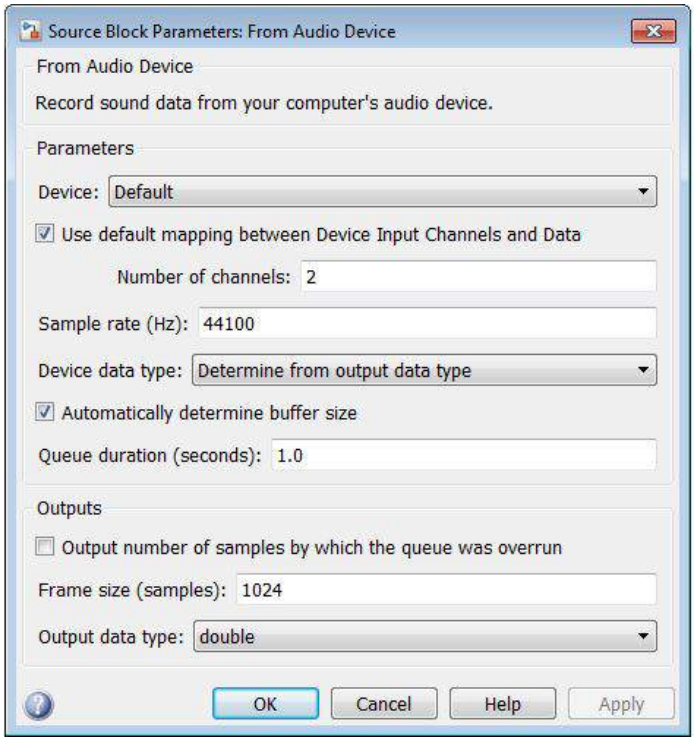


FIGURE D.3: Paramètre du bloc From Audio Device

Annexe E

Sauvegarde des résultats et utilisation du bloc S-Builder

E.1 Fonctionnement du S-Builder

Afin de pouvoir utiliser du code C/C++ dans un modèle Simulink on fait appel au bloc S-Builder, celui ci s'occupe de formater le code C et de le mettre en forme.

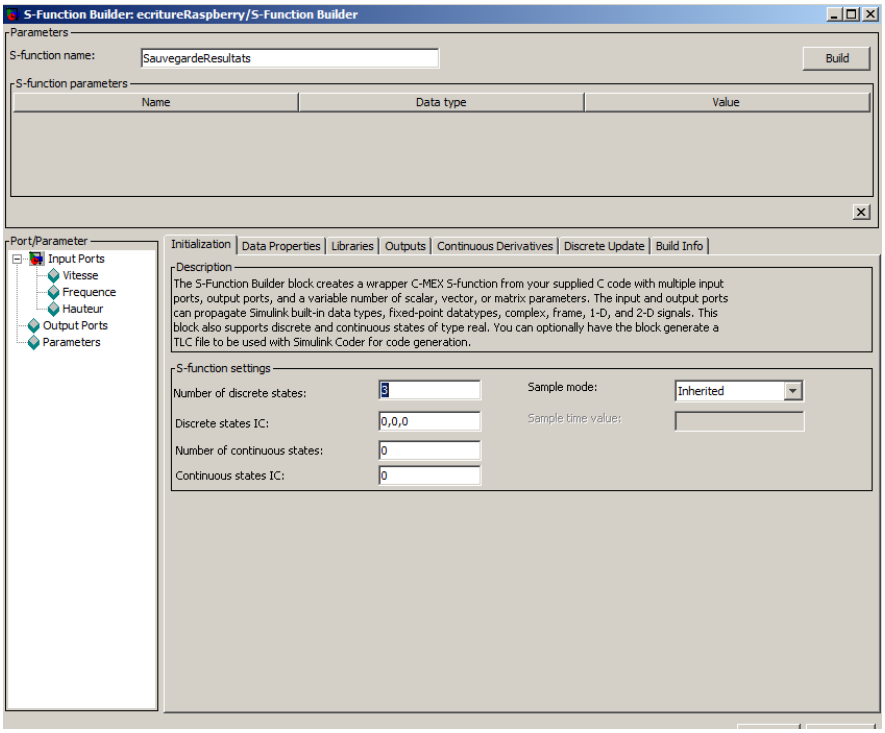


FIGURE E.1: Intérieur du S-Builder

On configure le nombre d'entrés sorties et le type de données ainsi que la fréquence d'échantillonnage du bloc.

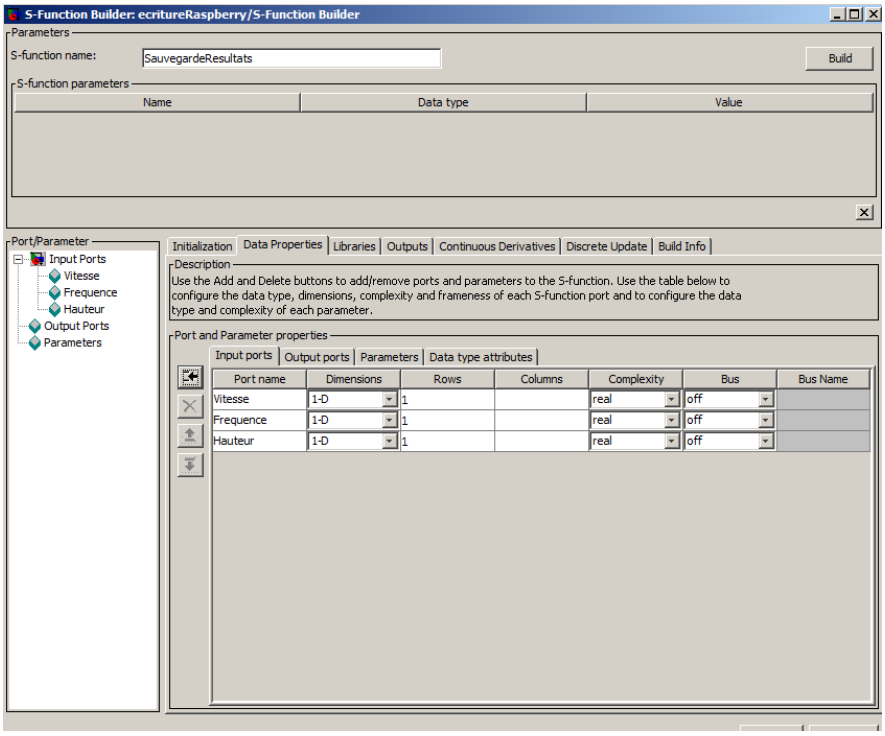


FIGURE E.2: Définition des signaux

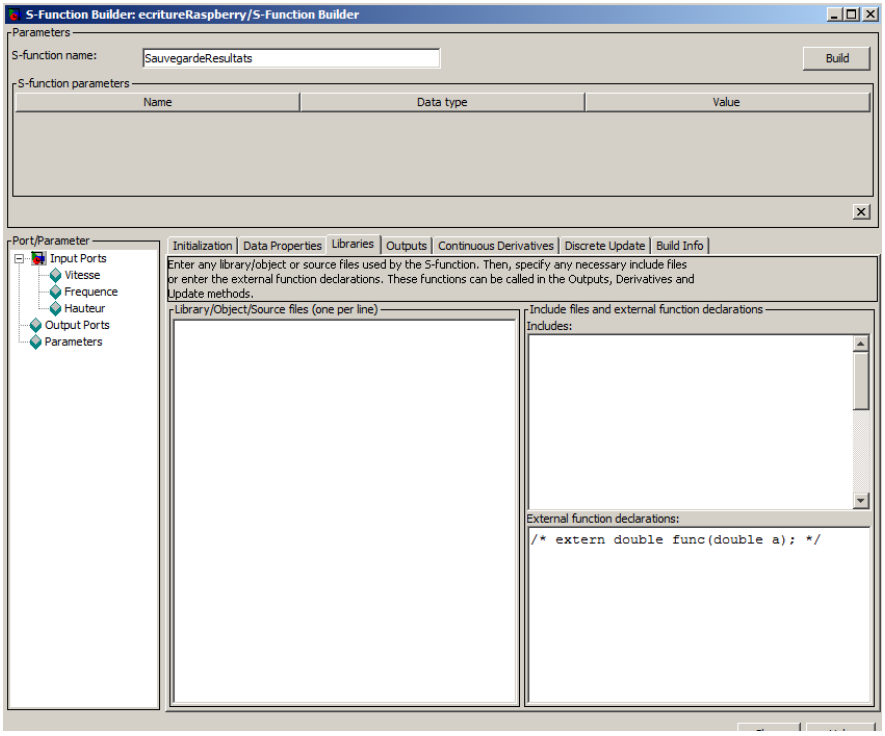


FIGURE E.3: Partie définition des bibliothèques

Consacrée aux entêtes et définition des variables du code C.

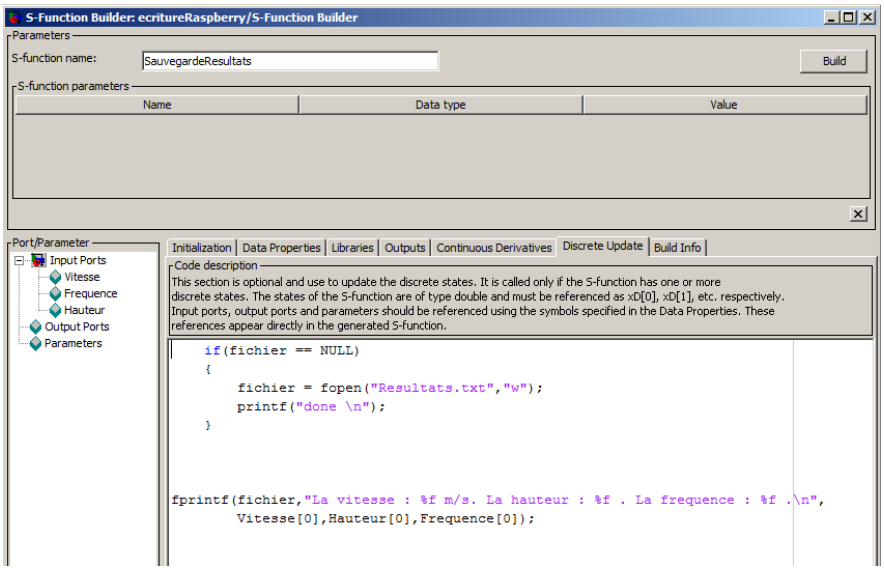


FIGURE E.4: Bloc de mise à jour

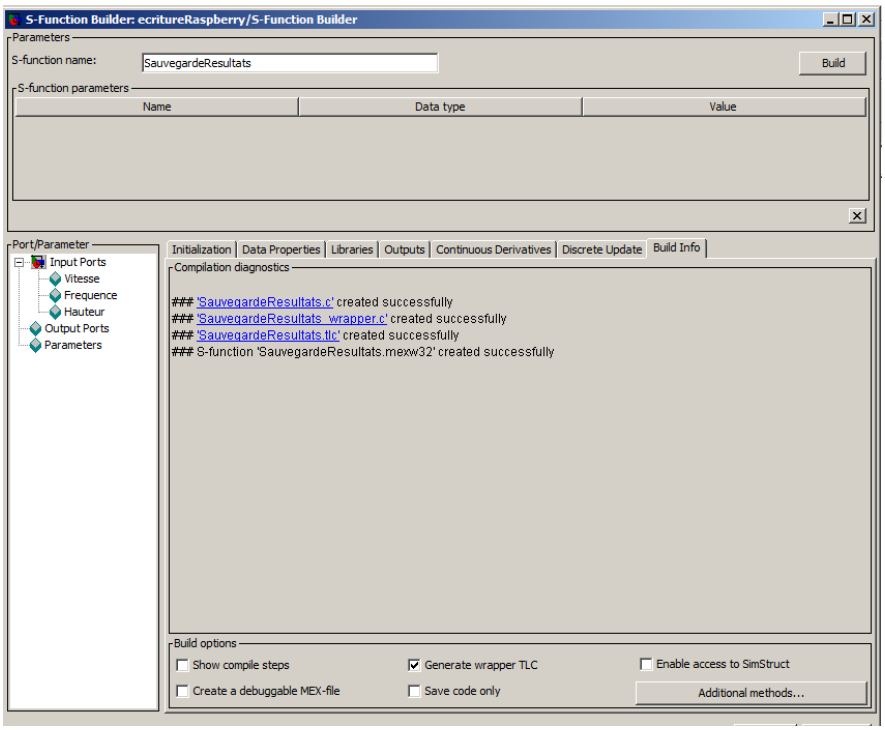


FIGURE E.5: Compilation

Enfin on compile avec Build.

E.2 Programme de sauvegarde

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
```

```
4 int i=0;
5   static  FILE* fichier = NULL;
```

On utilise le type *static* afin de pouvoir accéder à la variable fichier depuis la fonction de mise a jour du S-Builder.

```
1     if(fichier == NULL)
2     {
3         fichier = fopen("Resultats.txt","w");
4         printf("done \n");
5     }
6
7           fprintf(fichier,"La vitesse : %f m/s. La hauteur
                : %f . La frequence : %f .\n",Vitesse[0],
              Hauteur[0],Frequence[0]);
```

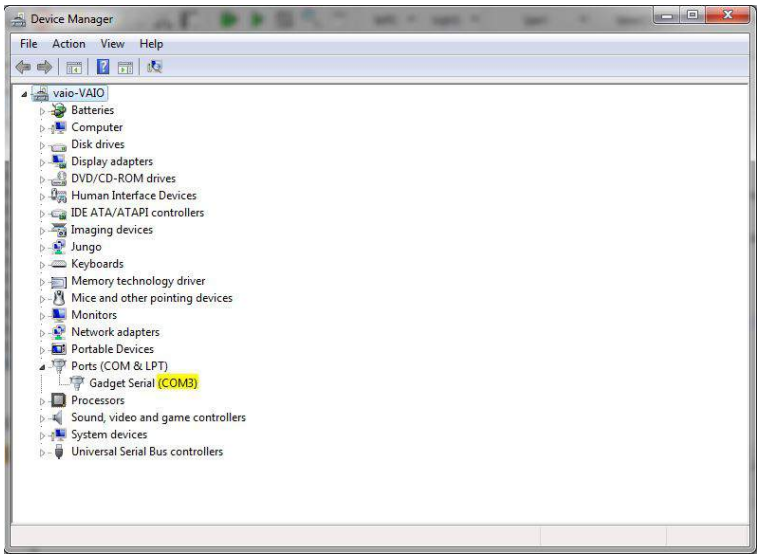
Ce code s'exécute en boucle à chaque période d'échantillonnage.

Annexe F

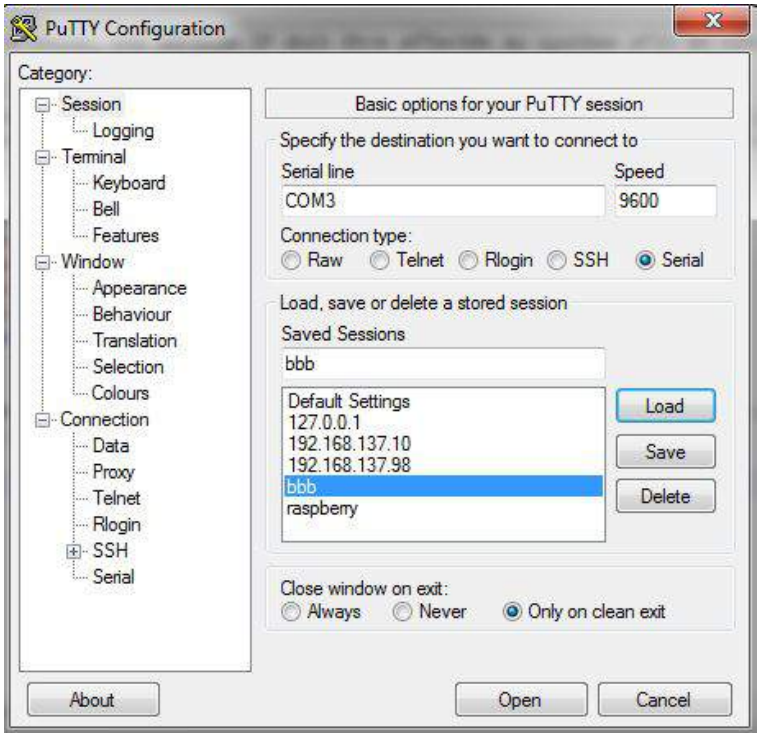
Configuration du système embarqué pour l'accès au réseau Internet

F.1 Affectation d'une adresse IP

Avant toute chose, une adresse IP statique (statique afin d'anticiper le cas ou il n'y aurait pas de serveur DHCP) doit être affectée au système afin de connecter ce dernier au réseau, pour cela, il faut suivre les étapes suivantes :
Étape 1 :
- Se connecter à la carte cible en utilisant une liaison série (cable USB) grâce à Putty. Il nous faut connaitre le port COM utilisé pour la liaison avec la carte, le gestionnaire des périphériques nous fournit cette information. Dans notre cas, c'est le COM3, comme le montre la figure suivante :



- A présent, dans Putty, on ouvre une session en entrant d'abord les paramètres nécessaire à la connexion série, à savoir COM3 dans le champ *Serial line* et 9600 dans le champ *vitesse* .



Étape 2 :

- Un fichier de configuration est à modifier, ce fichier se nomme Interfaces. Afin de le modifier on l'ouvre dans un éditeur de texte, dans ce qui suit, le fichier est ouvert avec nano.

$ sudo nano /etc/network/interfaces

- Dans ce fichier, il est nécessaire d'affecter un adresse IP, statique dans notre cas, le masque réseau, ainsi que la passerelle, dans notre cadre de travail, la passerelle était simplement le pc hôte, à travers lequel on dispose d'une connexion Internet qui est ensuite partagée avec le système embarqué.

```
  GNU nano 2.2.6         File: /etc/network/interfaces

# This file describes the network interfaces available on your system
# and how to activate them. For more information, see interfaces(5).

# The loopback network interface
auto lo
iface lo inet loopback
# The primary network interface
auto eth0
iface eth0 inet static
    address 192.168.137.2
    netmask 255.255.255.0
    network 192.168.137.0
    broadcast 192.168.137.255
    gateway 192.168.137.1

# Example to keep MAC address between reboots
#hwaddress ether DE:AD:BE:EF:CA:FE

# The secondary network interface
                        [ Read 36 lines ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

Étape 3 :

- Il est nécessaire, arrivé à cette étape, de redémarrer le service réseau afin de prendre en compte les modifications apportées.

$ sudo /etc/init.d/networking restart
- Une vérification peut être faite en lançant la commande

$ ifconfig

```
eth0      Link encap:Ethernet  HWaddr 1c:ba:8c:90:5b:9b
          inet addr:192.168.137.2  Bcast:192.168.137.255  Mask:255.255.255.0
          inet6 addr: fe80::1eba:8cff:fe90:5b9b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:814 errors:0 dropped:0 overruns:0 frame:0
          TX packets:269 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:82929 (80.9 KiB)  TX bytes:26708 (26.0 KiB)
          Interrupt:40
```

F.2 Définir un serveur DNS

Pour résoudre la correspondance entre une ressource identifiée par un nom de domaine et sa représentation logique (adresse ip), on doit définir un serveur DNS.
- Le fichier de configuration des serveurs DNS se nomme resolv.conf, on l'ouvre avec l'éditeur de texte nano pour y définir, par exemple ici, les serveurs DNS suivant :

$ nano /etc/resolv.conf

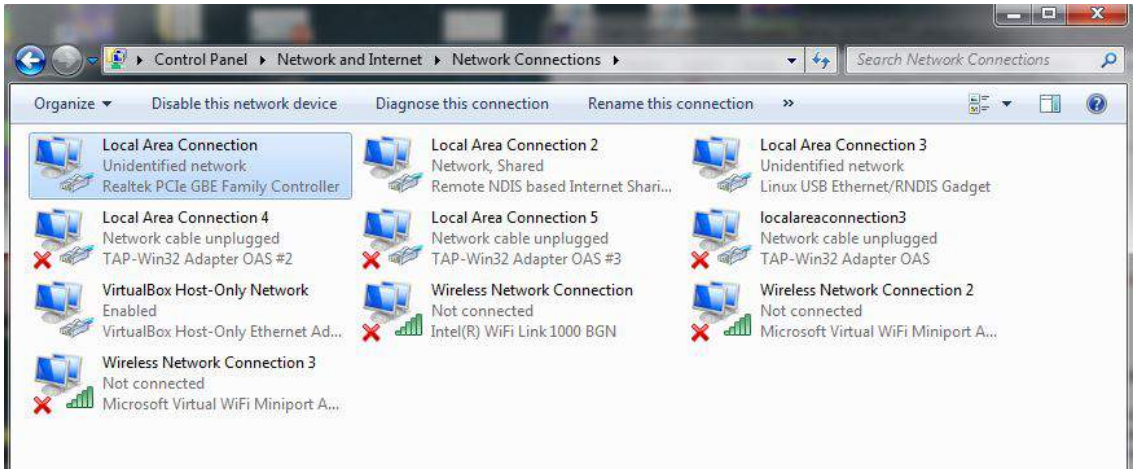
```
 GNU nano 2.2.6            File: /etc/resolv.conf

domain mshome.net
search mshome.net
nameserver 192.168.137.1
nameserver 8.8.8.8
nameserver 8.8.4.4
```

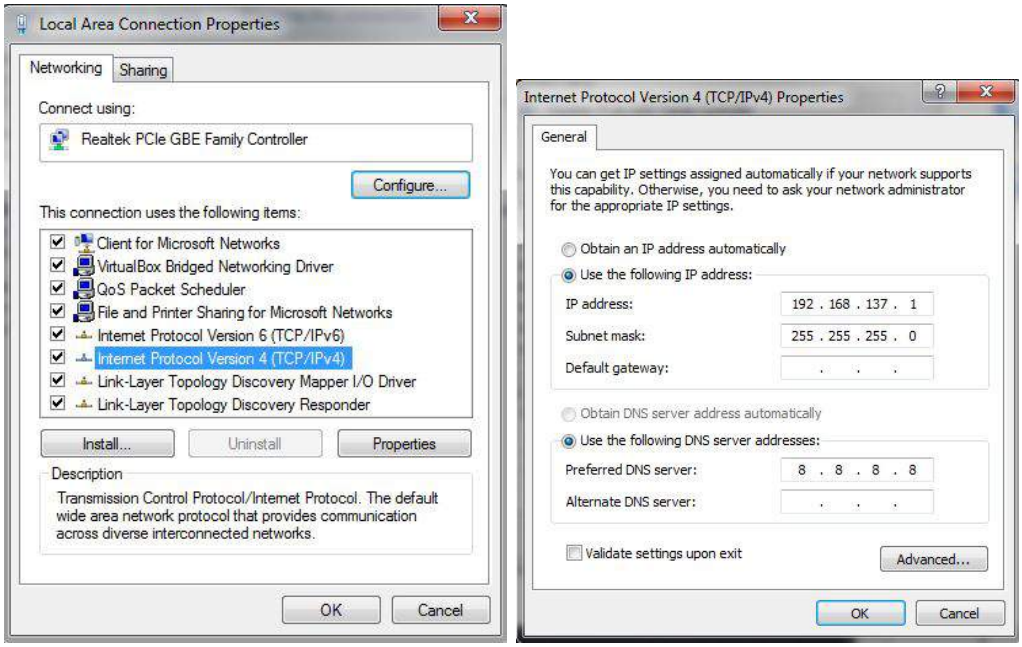
F.3 Partage réseau à partir de l'hôte

Il est nécessaire d'être sur le même le réseau, on se référer à l'adresse IP du système embarqué, introduite précédemment, pour définir une adresse IP pour l'hôte. Sachant qu'on a défini le hôte comme passerelle pour le carte cible, on doit donc reprendre la même adresse, à savoir dans notre cas, 192.168.137.1.

- Ouvrir le panneau de configuration des connexions réseau.

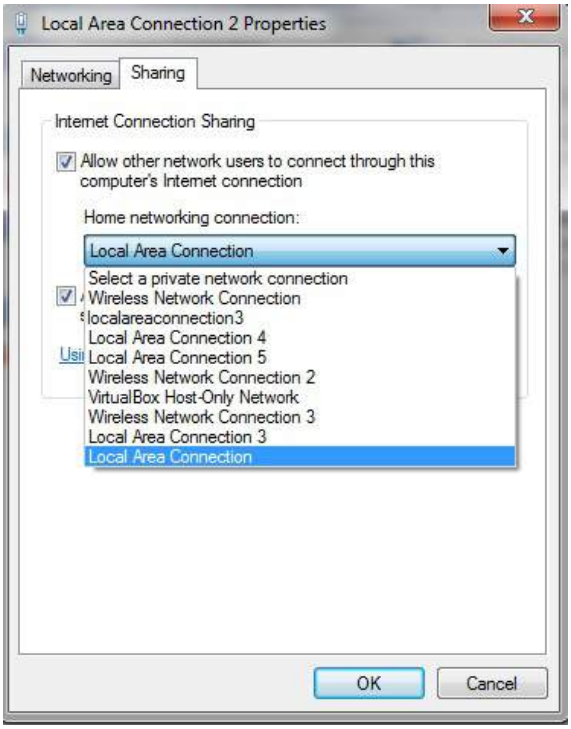


- Sélectionner la connexion LAN (correspondante à la liaison RJ45 avec la carte cible) et ouvrir Internet Prootocole Version 4.



- Introduire une adresse IP statique ainsi que le masque réseau, et en option une adresse d'un serveur DNS.

- Partager Internet avec la carte cible.



- Vérifier l'accès Internet. En lançant un ping sur www.google.com

```
ping root@beaglebone:~# ping www.google.com
PING www.google.com (41.201.164.59) 56(84) bytes of data.
64 bytes from 41.201.164.59: icmp_req=1 ttl=56 time=40.7 ms
64 bytes from 41.201.164.59: icmp_req=2 ttl=56 time=1623 ms
64 bytes from 41.201.164.59: icmp_req=3 ttl=56 time=1199 ms
64 bytes from 41.201.164.59: icmp_req=4 ttl=56 time=1175 ms
64 bytes from 41.201.164.59: icmp_req=5 ttl=56 time=814 ms
64 bytes from 41.201.164.59: icmp_req=6 ttl=56 time=91.2 ms
64 bytes from 41.201.164.59: icmp_req=7 ttl=56 time=443 ms
64 bytes from 41.201.164.59: icmp_req=10 ttl=56 time=51.5 ms
64 bytes from 41.201.164.59: icmp_req=11 ttl=56 time=396 ms
64 bytes from 41.201.164.59: icmp_req=12 ttl=56 time=75.6 ms
64 bytes from 41.201.164.59: icmp_req=13 ttl=56 time=483 ms
64 bytes from 41.201.164.59: icmp_req=14 ttl=56 time=536 ms
64 bytes from 41.201.164.59: icmp_req=15 ttl=56 time=1058 ms
64 bytes from 41.201.164.59: icmp_req=17 ttl=56 time=435 ms
```

Liaison établie.

Annexe G

Configuration d'une clée 3g, wifi

La configuration d'une clé 3G ou wifi sur un système embarqué s'effectue comme suit :

On vérfie que la clée est reconnue en utilisant la commande *lsusb*.
Pour le support du ce type de connectivité il faudra installer *modeswitch* avec un *sudoapt − getinstal usbmodeswitch*
$ifconfig$ indique la présence du modem $wwan0$.
Enfin on édite le fichier $interfaces$ comme suit.

```
1 allow-hotplug wwan0
2 iface wwan0 inet dhcp
```

Finalement l'accés sans-fil est fonctionnel.

FIGURE G.1: Clé 3g USB

Annexe H

Application Web Configuration via ThingSpeak

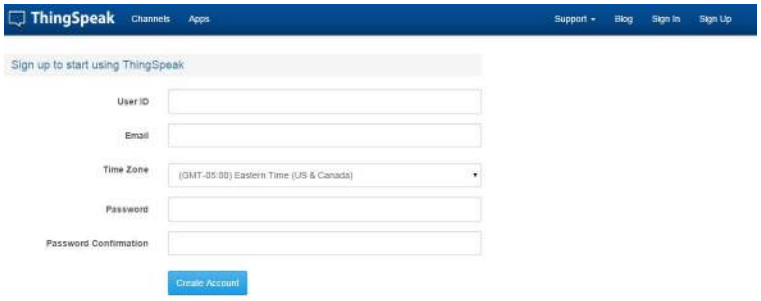
H.1 Configuration du modèle Simulink et ThingSpeak

- Configurer le modèle Simulink pour être exécuter sur la cible matérielle.

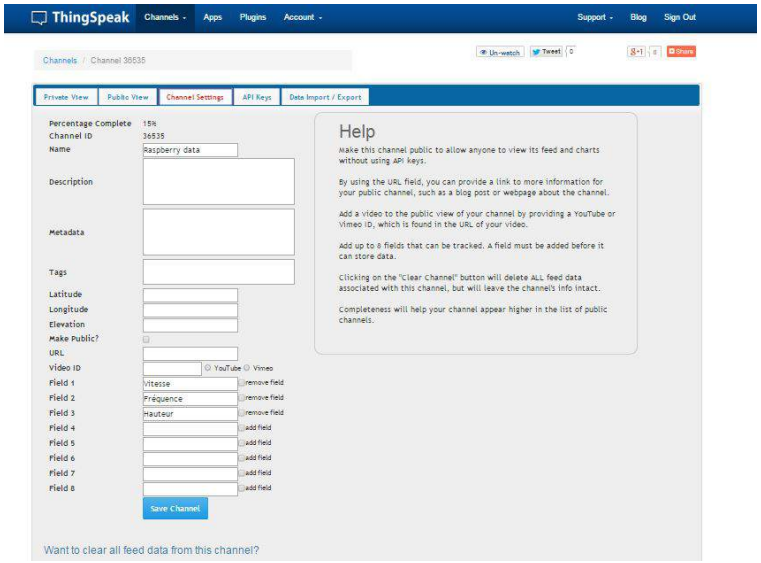
- Ajouter le bloc ThingSpeak au modèle.

- Connecter les signaux du modèle à transmettre, aux entrées du bloc ThingSpeak.

- Ouvrir un compte et s'identifier sur *https ://thingspeak.com/* .

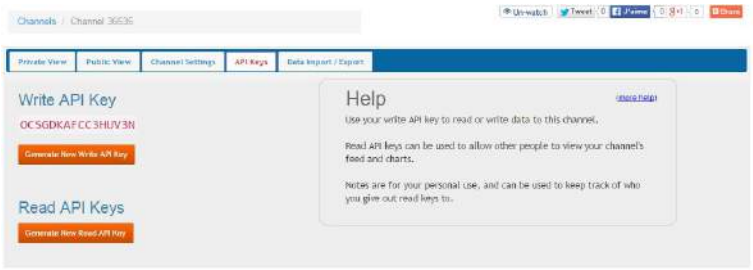


- Créer une nouvelle chaine et lui définir les noms de champ pour les données publiées.



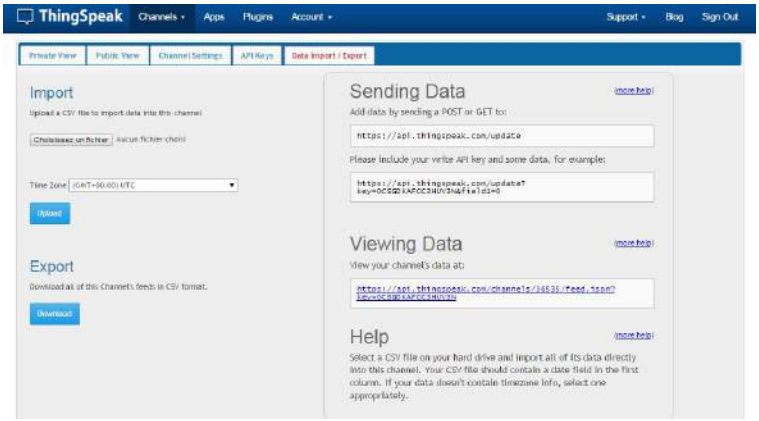
H.2 Déploiement du modèle

- Ouvrir l'onglet *API Keys* et copier la clé.



- Coller la clé dans le paramètre *Write API Key* du bloc ThingSpeak.

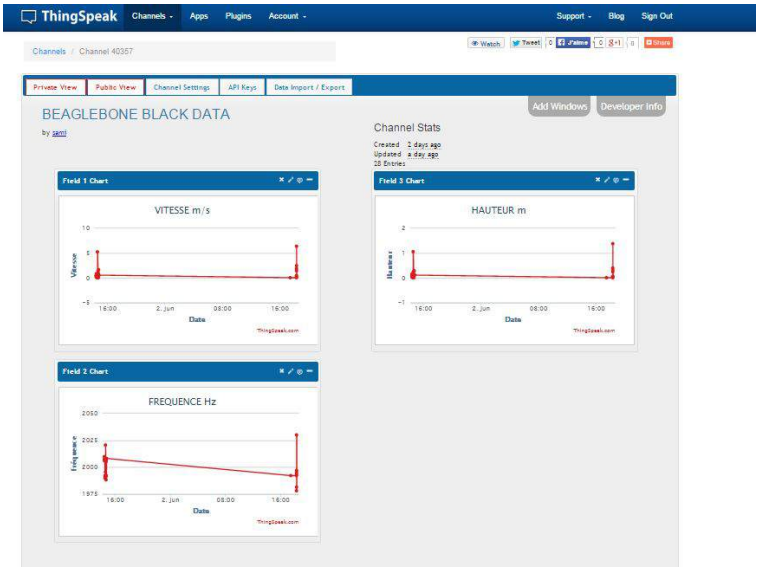
- Dans ThingSpeak.com, ouvrir l'onglet *Data Import/Export* et copier l'URL à partir de *Add data by sending a POST or Get to.*



- Dans le bloc ThingSpeak, coller l'URL dans le paramètre *Update URL* .

- Déployer le modèle sur la cible.

- Visualiser les données publiée dans ThingSpeak.com.


ThingSpeak
Channels
Apps
Plugins
Account
Support
Blog
Sign Out
Channels / Channel 40367
Private View
Public View
Channel Settings
API Keys
Data Import / Export
Add Windows
Developer Info
BEAGLEBONE BLACK DATA
by xavi
Channel Stats
Created 2 days ago
Updated a day ago
28 Entries
Field 1 Chart
VITESSE m/s
Field 3 Chart
HAUTEUR m
Field 2 Chart
FREQUENCE Hz
Date

Annexe I

Application Web classique

D'un coté on trouve une le programme de transmission des résultats vers une base de donnée (distante ou interne), de l'autre une application qui permet de visualiser le contenu de cette même bdd.

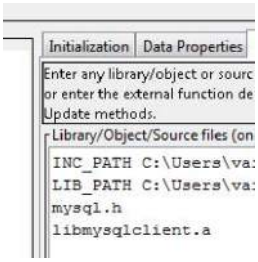


FIGURE I.1: Bibliothèques à inclure

Sur l'onglet *libraries* on ajoute les bibliothèques nécessaires (dynamiques ou statiques), les variables d'environnement INC_PATH, LIB_PATH et SRC_PATH respectivement pour les en-têtes, bibliothèques statiques ou dynamiques et enfin les sources.

— Avant de dialoguer avec une base de donnée, il faut s'y connecter. On a besoin de l'adresse IP de la machine où se trouve la base de donnée, du nom de la base de données ainsi que d'un login et d'un mot de passe.
— L'échange de données s'effectue via un langage permettant de faire des requétes précises (mettre a jour la base, lire, écrire), un exemple *sql*.

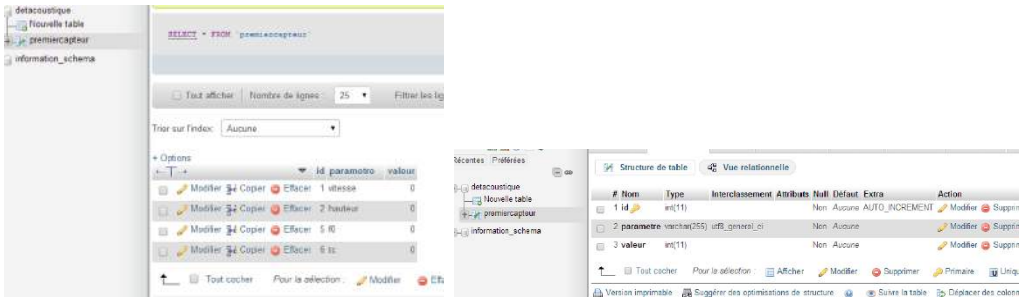


FIGURE I.2: Base de données via *phpmyadmin*

```php
<?php
    $bdds = new PDO('mysql:host=db4free.net;dbname=detacoustique','elnenp2015','eln2015');

    ?>

<head>
    <meta charset="utf-8">
    <title>Capteur détermination de paramètres de mobiles</title>

    <link rel="stylesheet" href="styleTempo.css">

    <link rel="shortcut icon" href="images/icon.png">
```

```php
<?php

$reponse = $bdds->query('SELECT * FROM premiercapteur') or die(print_r($bdds->errorInfo()));

while($var = $reponse->fetch())
{

echo '<p> La Valeur du paramètre ';
echo $var['parametre'];
echo ' est ';
echo $var['valeur'];
echo'</p>';

}

$reponse->closeCursor();
```

FIGURE I.3: Consultation de la base de données en *php*

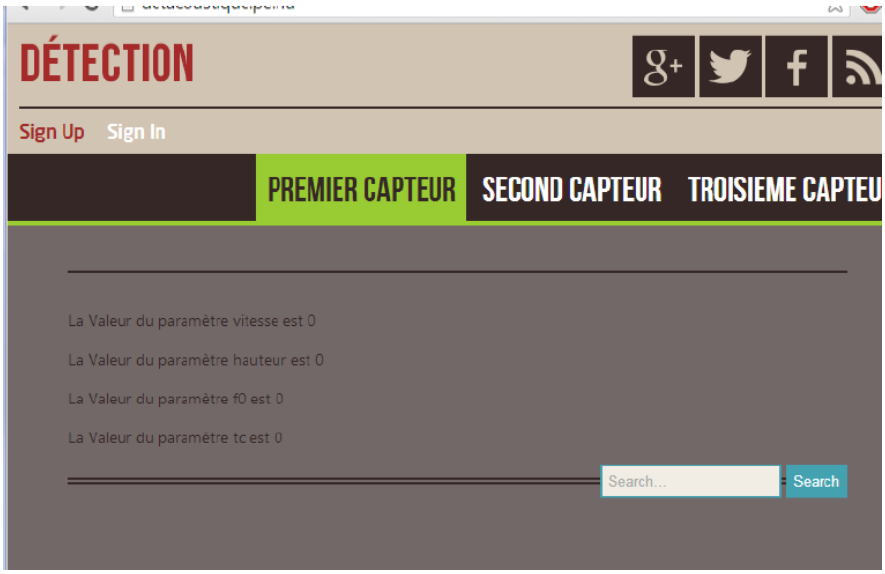


FIGURE I.4: Page Web prototype