Expansions Bash

M. Amine ROSTANE

À Propos du Tilde



Table des matières

1 Introduction

Le shell Bash (**B**ourne **A**gain **SH**ell) est l'un des interpréteurs de commandes les plus utilisés dans les environnements Unix et Linux. Bien que sa syntaxe et ses fonctionnalités soient bien documentées, certaines de ses caractéristiques, telles que les extensions de chaînes de caractères, demeurent moins explorées. Cette recherche a été initiée suite à une interaction académique avec mon professeur, M. Aubin, qui a souligné l'importance de comprendre ces aspects moins connus mais cruciaux de Bash. Le but de cette étude est d'examiner en détail le mécanisme des extensions dans Bash, avec un focus particulier sur le rôle et le fonctionnement du tilde (~).

2 Types d'expansions

Dans un script Bash, l'expansion est un mécanisme par lequel le shell transforme une commande en une ou plusieurs chaînes de caractères, qui sont ensuite exécutées. Ce processus est réalisé en plusieurs étapes et types d'expansion, chacune avec ses propres règles et syntaxes.

2.1 Pourquoi les expansions sont-elles importantes ?

Les expansions permettent d'écrire des scripts plus flexibles et puissants. Elles donnent la possibilité de manipuler des variables, de faire des calculs arithmétiques simples, et même de substituer la sortie d'une commande en tant qu'argument pour une autre. Sans les expansions, les scripts Bash seraient beaucoup moins dynamiques et fonctionnels.

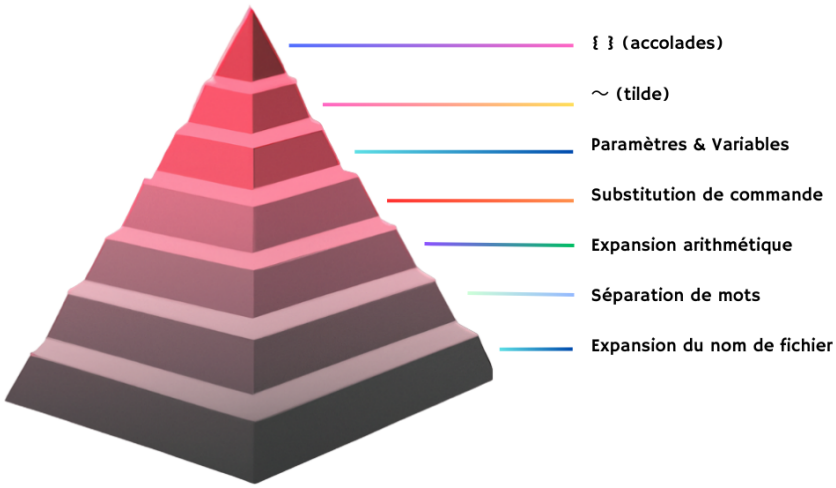
2.2 Types d'expansions

Bash effectue plusieurs types d'expansions, y compris mais sans s'y limiter :

— **Expansion des accolades** (Brace expansion)
— **Expansion de tilde** (Tilde expansion)
— **Expansion de paramètre et de variable** (Parameter and variable expansion)
— **Substitution de commande** (Command substitution)
— **Expansion arithmétique** (Arithmetic expansion)
— **Séparation de mots** (Word splitting)
— **Expansion du nom de fichier** (File name expansion)

Chaque type d'expansion a ses propres règles et est effectué dans un ordre spécifique, ce qui affecte le résultat final.

2.3 Ordre d'expansion



L'ordre dans lequel Bash effectue ces expansions est crucial pour comprendre comment une ligne de commande sera finalement interprétée. L'ordre est le suivant :

1. Expansion des accolades
2. Expansion de tilde
3. Expansion de paramètre et de variable
4. Substitution de commande
5. Expansion arithmétique
6. Séparation de mots
7. Expansion du nom de fichier

Après toutes les expansions, la suppression des guillemets est effectuée.

3 Utilisation pratique des expansions

Dans cette section, nous allons examiner plusieurs types d'expansions et voir comment les combiner dans un exemple pratique. Chaque étape ajoutera un nouveau type d'expansion ou une nouvelle fonctionnalité.

3.1 Étape 1 : Créer des fichiers simples

Nous commencerons par créer un fichier simple pour comprendre les bases. La commande 'touch' permet de créer un fichier vide.

```
# Cree un fichier vide nomme file.txt

$ touch file.txt
```

3.2 Étape 2 : Brace Expansion pour créer plusieurs fichiers

Dans cette étape, nous utilisons l'expansion des accolades pour créer plusieurs fichiers d'un coup. Les accolades fonctionnent comme des séquences génératrices.

```
# Cree trois fichiers : file1.txt, file2.txt, et file3.txt

$ touch file{1,2,3}.txt
```

3.3 Étape 3 : Introduction aux variables

Une variable est une façon de stocker une valeur pour l'utiliser plus tard. Dans cet exemple, nous stockons une chaîne de caractères dans une variable.

```
# Declare une variable et lui affecte la valeur "Hello, world!"

$ MY_VARIABLE="Hello, world!"
```

3.4 Étape 4 : Afficher le contenu d'une variable

Nous utilisons la commande 'echo' pour afficher le contenu de la variable. Le symbole '$' est utilisé pour récupérer la valeur stockée.

```
# Affiche la valeur de la variable MY_VARIABLE

$ echo $MY_VARIABLE
```

3.5 Étape 5 : Utilisation de la Variable dans une Commande

Ici, nous utilisons une variable pour stocker le nom du répertoire de destination pour les fichiers que nous allons copier.

```
# Declare une variable avec le nom du repertoire de destination

$ DEST="backup"
```

3.6 Étape 6 : Créer un répertoire en utilisant la variable

Nous utilisons la valeur de la variable '$DEST' pour créer un répertoire avec ce nom.

```
# Cree un repertoire avec le nom stocke dans la variable DEST

$ mkdir $DEST
```

3.7 Étape 7 : Copier les fichiers vers le répertoire de sauvegarde

Nous utilisons maintenant la commande 'cp' pour copier les fichiers dans le répertoire de sauvegarde.

```
# Copie les fichiers file1.txt, file2.txt, et file3.txt dans le
# repertoire de destination

$ cp file{1,2,3}.txt $DEST/
```

3.8 Étape 8 : Introduction à la Substitution de Commande

La substitution de commande vous permet d'utiliser la sortie d'une commande comme argument pour une autre commande. Ici, nous listons les fichiers dans le répertoire 'backup'. Nous utiliserons le résultat de cette commande dans l'étape suivante.

```
# Liste tous les fichiers .txt dans le repertoire stocke dans la
# variable DEST

$ ls $DEST/*.txt
```

3.9 Étape 9 : Créer une archive en utilisant la Substitution de Commande

Nous utilisons la commande 'tar' pour créer une archive compressée des fichiers de sauvegarde.

```
# Cree une archive tar.gz contenant tous les fichiers .txt dans le
# repertoire backup

$ tar -czf archive.tar.gz $(ls $DEST/*.txt)
```

3.10 Étape 10 : Introduction à l'Expansion Arithmétique

L'expansion arithmétique permet de faire des calculs simples directement dans le shell.

```
# Calcule la somme de 5 + 3 et stocke le resultat dans la variable
# NUMBER

$ NUMBER=$(( 5 + 3 ))
$ echo $NUMBER
```

3.11 Étape 11 : Utiliser l'Expansion Arithmétique dans une Commande

Nous intégrons maintenant l'expansion arithmétique dans une commande plus complexe, en utilisant la date pour calculer une valeur.

```
# Recupere le jour du mois et multiplie par 2

$ DAY=$(date +%d)
$ NUM=$(( DAY * 2 ))
```

3.12 Étape 12 : Exemple Complet

Finalement, nous combinons tous les éléments d'expansion que nous avons appris dans un exemple complet.

```
# Cree une archive avec un nom specifique, en utilisant les variables,
# la substitution de commande, et l'expansion arithmetique

$ tar -czf $DEST/archive$(date +\%Y\%m\%d)-$(( $(date +\%d) * 2 )).tar.gz
        $(ls $DEST/*.txt)
```

4 Exploration du code source de Bash : Focus sur l'Expansion

Dans cette section, nous explorons la structure du code source du shell Bash, en mettant particulièrement l'accent sur les mécanismes d'expansion.

4.1 Overview des Répertoires

- **builtins** : Ce répertoire contient le code source des commandes intégrées et des fonctionnalités du shell bash.
- **lib** : Un répertoire complet qui contient diverses bibliothèques utilisées dans le projet. Les sous-répertoires incluent :
  - **glob** : Fonctions pour la correspondance de motifs de noms de fichiers.
  - **intl** : Support de l'internationalisation et de la localisation.
  - **malloc** : Routines d'allocation de mémoire.
  - **readline** : Bibliothèque GNU Readline pour les fonctionnalités d'édition de ligne.
  - **sh** : Fonctions utilitaires du shell.
  - **termcap** : Base de données des capacités du terminal.
  - **tilde** : Utilitaires pour l'expansion du tilde.
- **tests** : Contient des scripts de test et des fichiers associés pour valider les fonctionnalités du shell bash.
- **include** : Contient des fichiers d'en-tête qui sont utilisés dans plusieurs parties du projet.
- **parser-built** : Ce répertoire contient des fichiers liés à l'analyseur syntaxique du shell bash.
- **po** : Contient des fichiers liés à la localisation, sous forme de fichiers `.po` et `.mo` pour différentes langues.
- **support** : Contient des scripts de support ou des utilitaires qui aident à la construction, aux tests, ou à l'exécution du shell bash.

4.2 Répertoires et Scripts Clés pour l'Étude de l'Expansion en Bash

Pour une étude approfondie des mécanismes d'expansion dans Bash, les répertoires et scripts suivants sont particulièrement pertinents :

1. **builtins** : Ce répertoire contient des informations sur les commandes intégrées qui utilisent ou affectent les mécanismes d'expansion. En utilisant la commande `compgen -b` nous pouvons lister toutes les commandes builtins.

```
bash-3.2$ compgen -b
```

2. **lib/tilde** : Ce sous-répertoire est spécifiquement dédié à l'expansion du tilde (`~`), qui est un type d'expansion en Bash.
3. **subst.c et subst.h** : Ces fichiers sont au cœur de la logique d'expansion en Bash. Ils contiennent des fonctions et des routines pour gérer divers types d'expansions comme l'expansion de paramètres, l'expansion arithmétique, etc.
4. **parse.y** : Ce fichier contient la grammaire Yacc pour le shell Bash et serait essentiel pour comprendre comment les expressions sont analysées, y compris les mécanismes d'expansion.
5. **tests** : Pour voir des exemples pratiques de comment les mécanismes d'expansion sont testés, ce répertoire serait utile.

4.3 Analyse détaillée de `subst.h`

Le fichier `subst.h` est un fichier d'en-tête dans le code source de Bash. Il sert de "contrat" entre différentes parties du code, définissant ce que le fichier `subst.c` est censé faire. En d'autres termes, il énumère les fonctions et les constantes qui sont utilisées pour gérer les expansions et les substitutions dans le shell Bash.

— **Constantes de gestion des guillemets et des backslashes** : Ces constantes, comme `Q_DOUBLE_QUOTES` et `Q_HERE_DOCUMENT`, définissent comment les guillemets et les backslashes doivent être traités pendant l'expansion. Par exemple, `Q_DOUBLE_QUOTES` serait utilisé pour indiquer que les backslashes dans une chaîne entre guillemets doubles doivent être traités d'une manière spécifique.

— **Fonctions d'expansion et de substitution** : Le fichier déclare des fonctions de type `expand_string*` et `do_assignment*`. Les fonctions `expand_string*` sont utilisées pour effectuer l'expansion de variables et d'expressions arithmétiques dans une chaîne, tandis que `do_assignment*` sont utilisées pour effectuer des assignations de variables.

— **Fonctions pour la gestion des listes de mots et des chaînes** : Des fonctions comme `string_list` et `word_list_remove_quoted_nulls` sont déclarées pour manipuler des listes de mots et des chaînes de caractères. Par exemple, `string_list` pourrait être utilisée pour convertir une liste de mots en une seule chaîne de caractères.

4.4 Analyse de la fonction `expand_string_if_necessary`

```
1 /* If there are any characters in STRING that require full expansion,
2    then call FUNC to expand STRING; otherwise just perform quote
3    removal if necessary.  This returns a new string. */
4 static char *
5 expand_string_if_necessary (string, quoted, func)
6      char *string;
7      int quoted;
8      EXPFUNC *func;
```

La fonction `expand_string_if_necessary` dans le fichier `subst.c` joue un rôle clé dans le processus d'expansion de Bash. Elle prend en entrée une chaîne de caractères, un indicateur pour savoir si la chaîne est entre guillemets, et une fonction à appeler pour effectuer l'expansion.

4.4.1 Vérification des Caractères Spéciaux

```
i = saw_quote = 0;
while (string[i])
  {
    if (EXP_CHAR (string[i]))
break;
    else if (string[i] == '\'' || string[i] == '\\' || string[i] == '"')
saw_quote = 1;
    ADVANCE_CHAR (string, slen, i);
  }
```

La fonction parcourt la chaîne de caractères pour vérifier la présence de caractères qui nécessitent une expansion (comme le signe du dollar $).

4.4.2 Appel à la Fonction d'Expansion

```
if (string[i])
  {
    list = (*func) (string, quoted);
    if (list)
{
  ret = string_list (list);
  dispose_words (list);
}
    else
ret = (char *)NULL;
  }
```

Si un tel caractère est trouvé, la fonction d'expansion passée en argument (`func`) est appelée pour effectuer l'expansion nécessaire.

4.4.3 Suppression des Guillemets

```
else if (saw_quote && ((quoted & (Q_HERE_DOCUMENT|Q_DOUBLE_QUOTES)) == 0))
  ret = string_quote_removal (string, quoted);
```

Si aucun caractère nécessitant une expansion n'est trouvé mais que la chaîne contient des guillemets, ceux-ci sont retirés.

4.4.4 Retour de la Chaîne Modifiée

```
else
  ret = savestring (string);
return ret;
```

La fonction retourne la chaîne de caractères modifiée, soit après expansion, soit après suppression des guillemets, soit telle quelle si aucune modification n'est nécessaire.

Cette fonction sert de "gardien", décidant si une chaîne de caractères donnée nécessite une expansion complète ou simplement la suppression des guillemets. Elle appelle ensuite la fonction appropriée pour effectuer cette tâche.

5 Le Tilde

Le Tilde a été introduit comme un raccourci pour le répertoire personnel dans les premiers systèmes Unix. Le concept de "répertoire personnel" est fondamental dans les systèmes Unix et Unix-like, car il offre à chaque utilisateur un espace privé pour stocker des fichiers, des scripts, des configurations, etc.

Dans les premiers jours de Unix, où les ressources système étaient limitées et les interfaces utilisateur étaient principalement des lignes de commande, l'efficacité et la simplicité étaient cruciales. Le tilde offrait un moyen simple et efficace de naviguer vers le répertoire personnel sans avoir à taper le chemin complet.

5.1 Analyse du Header `lib/tilde.h`

Ce fichier contient des déclarations de fonctions et de variables qui sont utilisées pour la gestion de l'expansion du tilde (`~`) dans les chaînes de caractères.

5.1.1 Points Pertinents

1. **Type de Fonction pour les Hooks** : Le type `tilde_hook_func_t` est défini comme un pointeur vers une fonction qui prend une chaîne de caractères et retourne une nouvelle chaîne de caractères. Ce type est utilisé pour les hooks d'expansion du tilde.
2. **Hooks d'Expansion du Tilde** : Deux hooks sont définis pour personnaliser le comportement de l'expansion du tilde :
   — `tilde_expansion_preexpansion_hook` : Appelé avant d'essayer les expansions standard du tilde.
   — `tilde_expansion_failure_hook` : Appelé si l'expansion standard du tilde échoue.
3. **Préfixes et Suffixes Additionnels** : Les variables `tilde_additional_prefixes` et `tilde_additional_suffixes` permettent de spécifier des chaînes supplémentaires qui peuvent être considérées comme des préfixes ou des suffixes lors de l'expansion du tilde (**e.g., le "/path/" dans "~/path/" peut être considéré comme un suffixe au tilde**).

4. **Fonctions d'Expansion du Tilde** :
   — `tilde_expand` : Effectue l'expansion du tilde sur une chaîne donnée.
   — `tilde_expand_word` : Semblable à `tilde_expand`, mais spécifique à un mot.
   — `tilde_find_word` : Trouve la portion de la chaîne qui commence par un tilde et qui doit être étendue.

5.2 Analyse de la fonction `tilde_expand`

La fonction `tilde_expand` est responsable de l'expansion des tildes (~) dans une chaîne de caractères donnée. Elle retourne une nouvelle chaîne avec les tildes correctement étendus.

```
char *tilde_expand (const char *string)
{
  char *result;
  int result_size, result_index;
  ...
}
```

5.2.1 Initialisation des Variables

```
  result_index = result_size = 0;
  if (result = strchr (string, '~'))
    result = (char *)xmalloc (result_size = (strlen (string) + 16));
  else
    result = (char *)xmalloc (result_size = (strlen (string) + 1));
```

La fonction commence par initialiser les variables nécessaires. Elle alloue de la mémoire pour la chaîne résultante en fonction de la présence ou non d'un tilde dans la chaîne d'entrée.

5.2.2 Boucle d'Expansion du Tilde

```
  while (1)
  {
    ...
    /* Make START point to the tilde which starts the expansion. */
    start = tilde_find_prefix (string, &len);
    ...
    /* Make END be the index of one after the last character of the username. */
    end = tilde_find_suffix (string);
    ...
  }
```

La fonction entre dans une boucle infinie où elle cherche des tildes à étendre. Elle utilise les fonctions ‘tilde_find_prefix‘ et ‘tilde_find_suffix‘ pour identifier les portions de la chaîne à étendre.

5.2.3 Expansion et Copie

```
  expansion = tilde_expand_word (tilde_word);
  ...
  strcpy (result + result_index, expansion);
  result_index += len;
```

La fonction utilise ‘tilde_expand_word‘ pour effectuer l'expansion du tilde. Le résultat est ensuite copié dans la chaîne résultante.

5.2.4 Terminaison

```
  result[result_index] = '\0';
  return (result);
```

Enfin, la chaîne résultante est terminée par un caractère nul et retournée.

5.3 Spécificité Cygwin dans `tilde_expand`

```
#ifdef __CYGWIN__
  /* Fix for Cygwin to prevent ~user/xxx from expanding to //xxx when
     $HOME for 'user' is /.  On cygwin, // denotes a network drive. */
  if (len > 1 || *expansion != '/' || *string != '/')
#endif
```

Cette partie du code est spécifique à l'environnement Cygwin sous Windows. Dans Cygwin, le chemin d'accès `//` est utilisé pour désigner un lecteur réseau. Le code cherche à éviter une situation où l'expansion du tilde (`~`) pourrait aboutir à un chemin d'accès commençant par `//`.

Le code utilise une directive de préprocesseur `#ifdef __CYGWIN__` pour s'assurer que ce bloc de code ne s'exécute que dans un environnement Cygwin.

La condition `if (len > 1 || *expansion != '/' || *string != '/')` vérifie trois choses :

1. `len > 1` : Vérifie si la longueur de l'expansion est supérieure à 1, ce qui signifie que l'expansion n'est pas simplement un `/`.
2. `*expansion != '/'` : Vérifie si le premier caractère de l'expansion n'est pas un `/`.
3. `*string != '/'` : Vérifie si le premier caractère de la chaîne d'origine n'est pas un `/`.

Si l'une de ces conditions est vraie, le code continue normalement. Sinon, il évite de copier l'expansion, empêchant ainsi la création d'un chemin d'accès commençant par `//`.

5.4 Pourquoi le Tilde est-il Spécial ?

Le tilde (`~`) est un caractère qui a acquis une importance particulière dans les systèmes Unix et Unix-like. Il sert de raccourci pour le répertoire personnel de l'utilisateur, une fonctionnalité qui, bien que simple en apparence, a des implications profondes pour l'interaction utilisateur-système. Explorons les raisons de cette importance :

- **Origines Historiques :** L'usage du tilde pour représenter le répertoire personnel trouve ses origines dans les premiers jours des systèmes Unix. À cette époque, les ressources système étaient limitées et les interfaces utilisateur étaient principalement textuelles. Le tilde offrait une manière simple et efficace de se référer au répertoire personnel, ce qui était particulièrement utile pour la navigation rapide et l'administration du système.
- **Efficacité et Simplicité :** Dans une interface en ligne de commande où chaque caractère compte, l'efficacité est primordiale. Le tilde permet d'accéder rapidement au répertoire personnel sans avoir à taper le chemin d'accès complet, ce qui accélère considérablement les opérations de gestion de fichiers.
- **Portabilité :** Le tilde est interprété par le shell lui-même, ce qui signifie que son comportement est cohérent sur différents systèmes Unix et Unix-like. Cette cohérence est cruciale pour la portabilité des scripts et des commandes, permettant aux utilisateurs de passer d'un système à l'autre sans avoir à réécrire ou à ajuster leurs scripts.

— **Clarté et Lisibilité :** L'utilisation du tilde dans les scripts et les commandes contribue à la lisibilité en remplaçant un chemin d'accès potentiellement long et complexe par un seul caractère. Cela rend le code plus facile à comprendre, à maintenir et à partager avec d'autres.

6 Conclusion

Cette recherche a entrepris une analyse exhaustive du mécanisme des expansions dans Bash, en se concentrant spécifiquement sur le rôle et la fonctionnalité du tilde (~). En explorant le code source de Bash, nous avons pu identifier les composants clés responsables de l'expansion du tilde et d'autres types d'expansions.

Nous avons également examiné l'histoire et la raison d'être du tilde dans les systèmes Unix, mettant en lumière son importance pour l'efficacité et la portabilité des scripts Bash. L'isolation du code relatif au tilde dans un répertoire séparé dans le code source de Bash témoigne de son importance et de sa complexité.

L'étude a révélé que, bien que le tilde soit un simple caractère, son traitement dans Bash est le résultat d'une conception minutieuse. Il sert non seulement à simplifier la navigation dans le système de fichiers, mais aussi à rendre les scripts plus lisibles et maintenables.

Les connaissances acquises au cours de cette recherche ont des implications significatives pour la compréhension des fonctionnalités avancées de Bash. Elles sont non seulement utiles pour les utilisateurs qui cherchent à maîtriser Bash, mais aussi pour les développeurs qui souhaitent contribuer au projet Bash ou développer des fonctionnalités similaires dans d'autres shells.

Références

[1] GNU Project Documentation, “3.5.1 Brace Expansion" *Docs*.

[2] GNU Project Documentation, “3.5.2 Shell Tilde Expansion" *Docs*.

[3] GNU Project Documentation, “3.5.3 Shell Parameter Expansion" *Docs*.

[4] GNU Project Documentation, “3.5.3 Shell Parameter Expansion" *Docs*.

[5] GNU Project Documentation, “3.5.4 Shell Command Substitution" *Docs*.

[6] GNU Project Documentation, “3.5.5 Shell Arithmetic Expansion" *Docs*.

[7] GNU Project Documentation, “3.5.7 Word Splitting Expansion" *Docs*.

[8] GNU Project Documentation, “3.5.8 Filename Expansion" *Docs*.

[9] GNU Project Documentation, “6.8.1 Directory Stack Builtins" *Docs*.

[10] Unofficial Mirror of Bash Repository - Updated Daily *Git Repository*.

[11] DataCademia, “Bash - Tilde ( ) Expansion“ *Docs*.

[12] Ants Are Everywhere, “Let's read the Bash source code“ *Youtube*.

[13] Kris Jordan, “Shell - 5 - Bash Shell Expansions“ *Youtube*.

[14] Superuser Forum, @daveslab, “How to go to remote directory in Cygwin“ *Superuser Forum*.

[15] Cygwin Documentation, “Cygwin User's Guide“ *Docs*.

Challenges Techniques

M. Amine ROSTANE

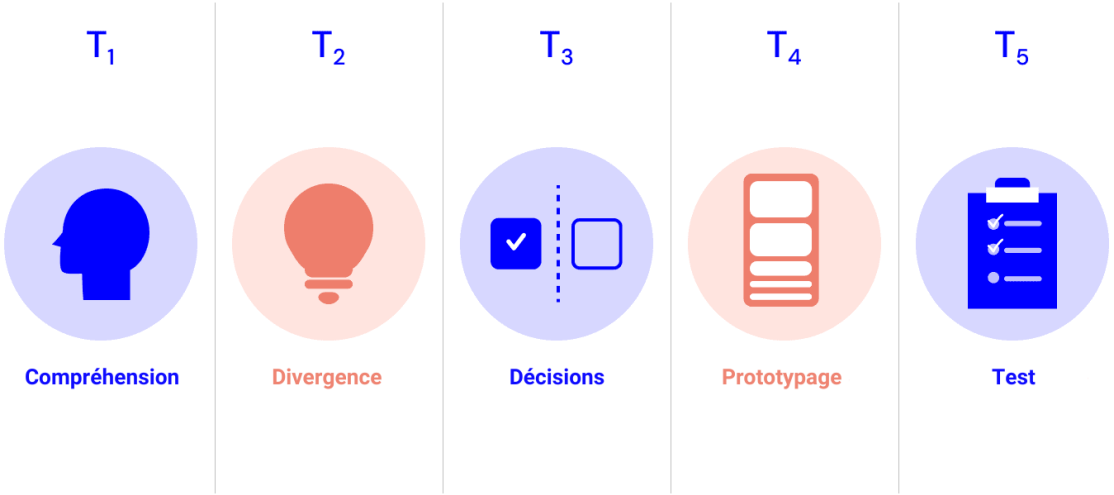
Le Design Sprint

Table des matières

1 Introduction

Dans un paysage technologique en constante évolution, l'efficacité et la rapidité de développement sont cruciales pour les ingénieurs. Le Design Sprint, une méthodologie développée initialement chez Google Ventures, se présente comme un outil révolutionnaire pour transformer rapidement des idées en solutions concrètes. Ce rapport vise à explorer le concept du Design Sprint et à souligner ses avantages pour les ingénieurs, en se concentrant sur la manière dont cette méthode accélère la conception, le prototypage et le test d'idées avec les utilisateurs. En tant qu'étudiants en ingénierie, l'approfondissement de méthodologies innovantes telles que le Design Sprint est essentiel pour notre développement professionnel, nous permettant de rester à la pointe de la technologie et de la gestion de projet.

2 Compréhension du Design Sprint



2.1 Définition du Design Sprint

Le Design Sprint est une méthodologie de conception itérative et rapide qui vise à résoudre des problèmes complexes et à tester de nouvelles idées en un temps record. Cette approche, qui combine stratégie, innovation, comportement des utilisateurs et design, permet aux équipes de développer et de valider des concepts rapidement grâce à des cycles de conception, de prototypage, et de tests utilisateurs.
Le Design Sprint se distingue par son cadre structuré et son orientation vers l'action. Contrairement aux méthodes traditionnelles de développement de produits, qui peuvent s'étendre sur de longues périodes, le Design Sprint concentre les efforts de l'équipe sur une semaine intensive. Cette concentration temporelle permet non seulement de réduire les coûts et le temps consacrés au développement, mais encourage également une prise de décision rapide et efficace.

2.2 Histoire et origines du Design Sprint

Le Design Sprint a été développé chez Google Ventures, la branche de capital-risque de Google. Jake Knapp, alors designer chez Google, est considéré comme le principal créateur de cette méthodologie. Knapp a commencé à élaborer le concept de Design Sprint en 2010, en intégrant des éléments du Design Thinking, une méthode populaire dans les années 2000, et de l'agilité, une approche de gestion de projet issue du développement logiciel. Les premiers sprints chez Google ont aidé à lancer des produits comme Gmail et Google Hangouts. En 2012, Knapp a rejoint Google Ventures, où il a affiné la méthode avec Braden Kowitz et John Zeratsky, en l'appliquant à diverses startups du portefeuille de Google Ventures. Le processus a été formalisé et popularisé à travers leur livre "Sprint : How to Solve Big Problems and Test New Ideas in Just Five Days", publié en 2016.

2.3 Les principales étapes du Design Sprint

Un Design Sprint typique se déroule sur cinq jours, chacun consacré à une phase spécifique du processus de conception.

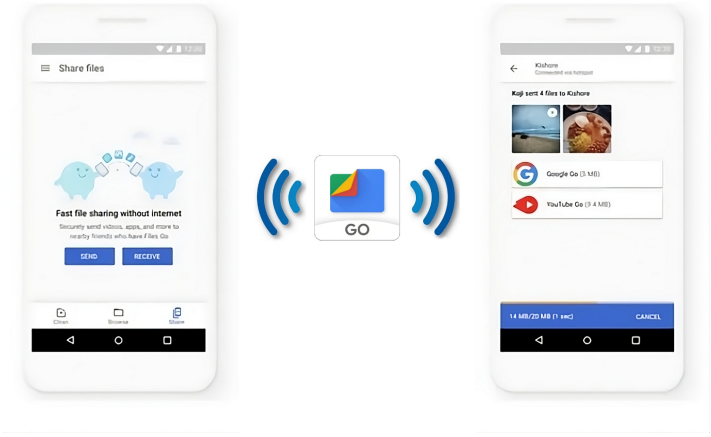
1. **Jour 1 : Comprendre** - L'équipe s'immerge dans le problème, en examinant l'expertise existante et en définissant les objectifs du sprint. Cette phase comprend souvent des discussions avec des experts, une cartographie des processus et l'identification des défis clés.
2. **Jour 2 : Diverger** - Les participants explorent individuellement différentes solutions, souvent à travers des sessions de brainstorming et de sketching. L'objectif est de générer un large éventail d'idées et de perspectives.
3. **Jour 3 : Décider** - L'équipe examine les idées générées, engage des discussions critiques et vote pour les solutions les plus prometteuses. Une "solution cible" est choisie pour le prototype, souvent en combinant les éléments les plus forts de plusieurs propositions.
4. **Jour 4 : Prototyper** - Un prototype réaliste mais simplifié est construit. L'objectif n'est pas de créer un produit fini, mais un artefact suffisamment élaboré pour tester les hypothèses clés.
5. **Jour 5 : Tester** - Le prototype est testé avec de vrais utilisateurs. Les retours sont collectés pour comprendre les réactions, les compréhensions et les comportements des utilisateurs face au prototype. Ces informations sont utilisées pour valider ou réfuter les hypothèses du sprint.

Chaque phase est structurée pour maximiser l'efficacité et la créativité, permettant aux équipes de surmonter les obstacles traditionnels du développement de projet et d'atteindre des résultats tangibles en un temps réduit.

3 Avantages du Design Sprint pour les ingénieurs

3.1 Facilitation de la résolution de problèmes complexes

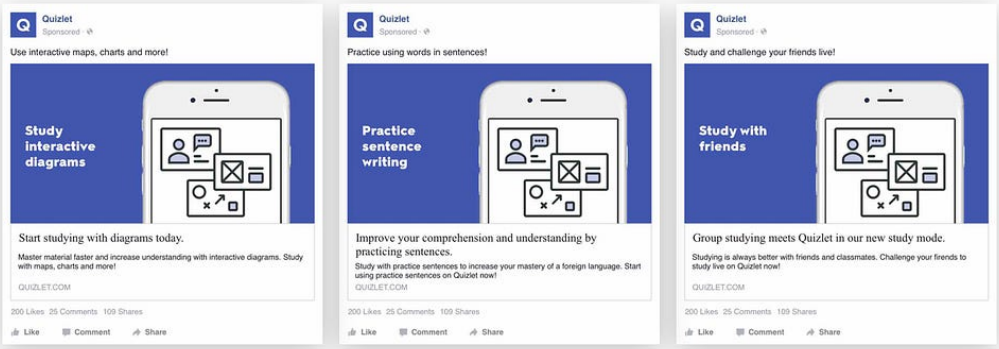
Le Design Sprint facilite la résolution de problèmes complexes en permettant une compréhension rapide et une réponse coordonnée aux défis. Un exemple concret est celui de Google avec leur application Files Go. Pour répondre aux besoins de stockage des utilisateurs de smartphones dans les pays en développement, l'équipe de Google a utilisé un Design Sprint pour développer la deuxième version de leur application. Ce processus a conduit à la création de prototypes testés auprès des utilisateurs d'Android Go, permettant une adaptation rapide aux besoins réels du marché.



3.2 Promotion de l'innovation et de la créativité

Les Design Sprints jouent un rôle clé dans la promotion de l'innovation et de la créativité, en offrant un environnement structuré qui encourage la génération d'idées nouvelles et originales. L'exemple de Quizlet, une plateforme d'apprentissage en ligne, illustre parfaitement comment le Design Sprint peut transformer le processus de développement de produits.

Quizlet a utilisé le Design Sprint pour identifier et développer de nouvelles fonctionnalités qui répondent directement aux besoins des étudiants. Au cours du sprint, l'équipe de Quizlet a d'abord défini clairement le problème à résoudre : comment rendre l'apprentissage plus efficace et engageant pour les étudiants ? Ensuite, au cours de la phase de divergence, l'équipe a généré une multitude d'idées innovantes, allant de nouvelles méthodes d'apprentissage interactif à des fonctionnalités améliorées pour l'organisation des cours.



En impliquant activement les utilisateurs dès le début du processus, notamment à travers des interviews et des sessions de tests, Quizlet a pu obtenir des retours précieux directement de la part des étudiants. Ces retours ont joué un rôle crucial dans la phase de décision, où l'équipe a dû choisir quelles idées méritaient d'être développées en prototypes.

Les prototypes ont ensuite été testés et itérés, en se basant sur les réactions des utilisateurs. Ce processus itératif a permis à Quizlet de peaufiner ses idées initiales en solutions réellement adaptées aux besoins des étudiants, stimulant ainsi une innovation véritablement centrée sur l'utilisateur. Le Design Sprint a donc non seulement facilité la création de nouvelles fonctionnalités, mais a également assuré qu'elles soient pertinentes et efficaces, reflétant une véritable compréhension des besoins des étudiants.

3.3 Réduction des délais de développement

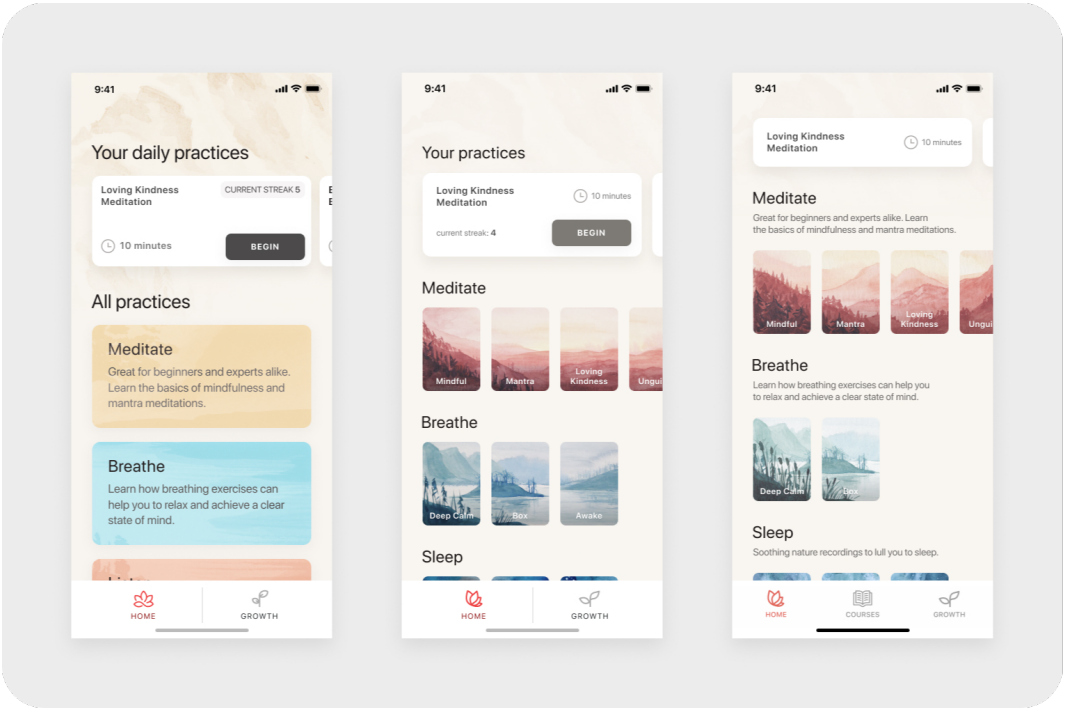
Le Design Sprint est particulièrement efficace pour réduire les délais de développement, un avantage crucial dans le secteur de l'ingénierie où la rapidité de mise sur le marché peut être un facteur clé de succès. Cette méthode permet de valider rapidement des idées avant d'investir dans des prototypes de haute fidélité et dans le développement à grande échelle.

Un exemple illustratif de cette efficacité est le cas de l'application de méditation Oak. Confrontée au défi de rafraîchir et d'améliorer son expérience utilisateur, l'équipe d'Oak a opté pour un Design Sprint. Cette démarche a permis de cerner rapidement les aspects de l'application nécessitant une amélioration et de définir des priorités claires pour le développement.

Durant le sprint, l'équipe a travaillé de manière intensive sur des cycles courts de prototypage et de tests. En quelques jours seulement, ils ont pu créer et tester des prototypes fonctionnels, recueillir

des retours d'utilisateurs et affiner leur approche. Ce processus itératif a permis de développer des solutions innovantes et adaptées, tout en évitant les longues périodes de développement traditionnelles.

Le Design Sprint a ainsi conduit à une version simplifiée et plus conviviale de l'application Oak, en un temps record. Cet exemple démontre comment le Design Sprint peut aider les équipes d'ingénierie à accélérer le processus de développement, en garantissant que les solutions sont à la fois innovantes et alignées sur les besoins des utilisateurs.



4 Ressources éducatives sur le design sprint

4.1 Introduction aux MOOCs

Les MOOCs (Cours en Ligne Ouverts et Massifs) offrent un accès flexible et accessible à des enseignements de qualité sur divers sujets, y compris le Design Sprint. Ces cours en ligne facilitent l'apprentissage autonome, permettant aux utilisateurs d'acquérir de nouvelles compétences selon leur rythme et leurs intérêts.

Lors du choix d'un MOOC, il est important de considérer la qualité et l'actualité du contenu, l'expertise des instructeurs, ainsi que la structure et la clarté du cours. De plus, il est utile de prendre en compte les retours et évaluations des participants précédents pour évaluer l'efficacité du cours et son adéquation avec vos objectifs d'apprentissage.

Ces critères aident à sélectionner les MOOCs les plus pertinents et bénéfiques pour l'apprentissage du Design Sprint, en fonction des besoins individuels et des objectifs professionnels.

4.2 Tableau comparatif des MOOCs gratuits disponibles

Voici un tableau comparatif de quelques MOOCs sur le Design Sprint :

Nom du cours	Plateforme	Durée	Contenu abordé	Niveau requis	Avis
UX Design Strategy and Application : Customer Profiling and Design Sprints	FutureLearn	4 semaines	Stratégies UX, Profilage Client, Design Sprints	Débutant	Non spécifié
Product Design	Udacity	Auto-rythmé	Validation de produit, Pratiques UI/UX, Design Sprint	Débutant	Non spécifié
Google : Foundations of User Experience (UX) Design	Coursera	Auto-rythmé	Concepts de base en UX, Design centré sur l'utilisateur, Accessibilité, Design Sprint	Débutant	4,8 étoiles

TABLE 1 – Comparaison des MOOCs sur le Design Sprint

5 Pertinence de l'enseignement du Design Sprint en 3ème Année d'ingénierie

5.1 Analyse des besoins en compétences pour les ingénieurs

L'intégration du Design Sprint dans le cursus d'ingénierie répond à un besoin croissant de compétences interdisciplinaires et de pensée innovante. Les ingénieurs d'aujourd'hui doivent être capables de résoudre des problèmes complexes, de travailler en équipe et de développer rapidement des solutions viables. Le Design Sprint, en tant que méthode agile, encourage le développement de ces compétences essentielles, en mettant l'accent sur la collaboration, l'innovation rapide et la résolution de problèmes orientée utilisateur.

5.2 Avantages pédagogiques du design sprint pour les étudiants en ingénierie

Enseigner le Design Sprint en 3ème année, une période consacrée aux fondamentaux, présente plusieurs avantages pédagogiques. Premièrement, cela permet aux étudiants de s'initier tôt à des méthodologies de travail collaboratif et de pensée créative, essentielles dans de nombreux domaines de l'ingénierie. Pour les étudiants envisageant une carrière dans l'entrepreneuriat ou dans des entreprises créatives, cette compétence est particulièrement bénéfique. Elle les prépare à aborder les défis futurs avec une approche innovante et centrée sur l'utilisateur, un atout majeur dans le contexte professionnel actuel.

5.3 Défis et considérations pour l'intégration dans le cursus

L'intégration du Design Sprint dans le programme d'études d'ingénierie n'est pas sans défis. Il est crucial de trouver un équilibre entre les fondamentaux théoriques et l'apprentissage pratique de cette méthodologie. De plus, la mise en place de projets de Design Sprint nécessite des ressources adéquates, notamment des formateurs expérimentés et des outils adaptés. Il est également important de contextualiser le Design Sprint dans le cadre plus large des compétences d'ingénierie, en s'assurant que les étudiants comprennent sa place et sa valeur dans le processus global de développement et de conception.

6 Conclusion

La réalisation de ce rapport a été une expérience enrichissante et éducative. L'analyse approfondie des différentes facettes du Design Sprint, depuis sa définition et son histoire jusqu'à ses applications pratiques, m'a permis de gagner une compréhension holistique de cette méthodologie. Cette exploration a non seulement renforcé mes connaissances théoriques, mais m'a également offert des perspectives pratiques applicables dans le domaine de l'ingénierie.

L'étude des avantages du Design Sprint pour les ingénieurs a mis en évidence l'importance de méthodes agiles et innovantes dans le processus de développement moderne. La revue des MOOCs disponibles sur le sujet a été particulièrement bénéfique. Elle a non seulement élargi mon horizon d'apprentissage mais m'a aussi fourni des ressources précieuses pour poursuivre mon éducation autonome dans ce domaine.

Enfin, la réflexion sur la pertinence de l'enseignement du Design Sprint en 3ème année d'ingénierie m'a permis de comprendre son rôle crucial dans la formation d'ingénieurs polyvalents et innovants. Cela a renforcé ma perspective sur l'importance de l'apprentissage pratique et centré sur l'utilisateur dans l'ingénierie.

Références

[1] QU'EST-CE QUE LE DESIGN SPRINT ? *Article*.

[2] The Design Sprint *Article*.

[3] 2 minutes pour comprendre le Design Sprint *Youtube*.

[4] What is a Design Sprint ? | Google UX Design Certificate *Youtube*.

[5] Explore the Future of Files Go | Google Design Sprint Case Study *Article*.

[6] Quizlet : How we moved beyond definitions and into diagrams *Article*.

[7] From Idea to App Store : A Design Sprint Case Study *Article*.

[8] UX Design Strategy and Application : Customer Profiling and Design Sprints *MOOC*.

[9] Product Design | Udacity *MOOC*.

[10] Foundations of User Experience (UX) Design | Google *MOOC*.

Rapport de TP

Amine ROSTANE
Adam BEREKSI

Traitement Numérique du Signal

Table des matières

1 Introduction

Dans le cadre de ce TP de TNS, nous nous engageons à approfondir la logique du traitement numérique du signal en utilisant un DSP de Texas Instruments. Notre objectif est de comprendre et de mettre en œuvre les algorithmes de traitement de signal, de coder des filtres numériques courants et de concevoir des filtres personnalisés. Nous nous concentrons sur l'apprentissage de la théorie sous-jacente et sur l'application de cette théorie à travers l'utilisation de MATLAB et Code Composer Studio pour effectuer des simulations et programmer le DSP. Ce processus nous permettra d'acquérir une compréhension approfondie des calculs et des principes qui régissent la création de filtres efficaces et la manipulation des signaux numériques.

2 Analyse des schémas

Nous avons commencé par analyser les schémas structurels et techniques pour établir un parallèle entre les éléments conceptuels et physiques de notre kit de traitement de signal. Cette analyse vise à comprendre comment les signaux sont physiquement traités par les composants du kit.

2.1 Schémas structurel et technique

Le schéma structurel illustre le flux général de traitement du signal, indiquant le passage du signal original à travers différents composants jusqu'à la sortie du signal traité. En comparaison, le schéma technique fournit une représentation détaillée des composants réels sur le DSP.

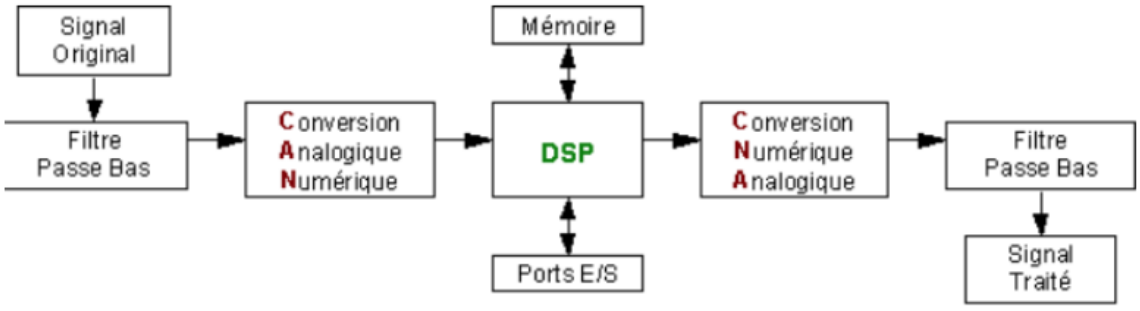


FIGURE 1 – Schéma structurel

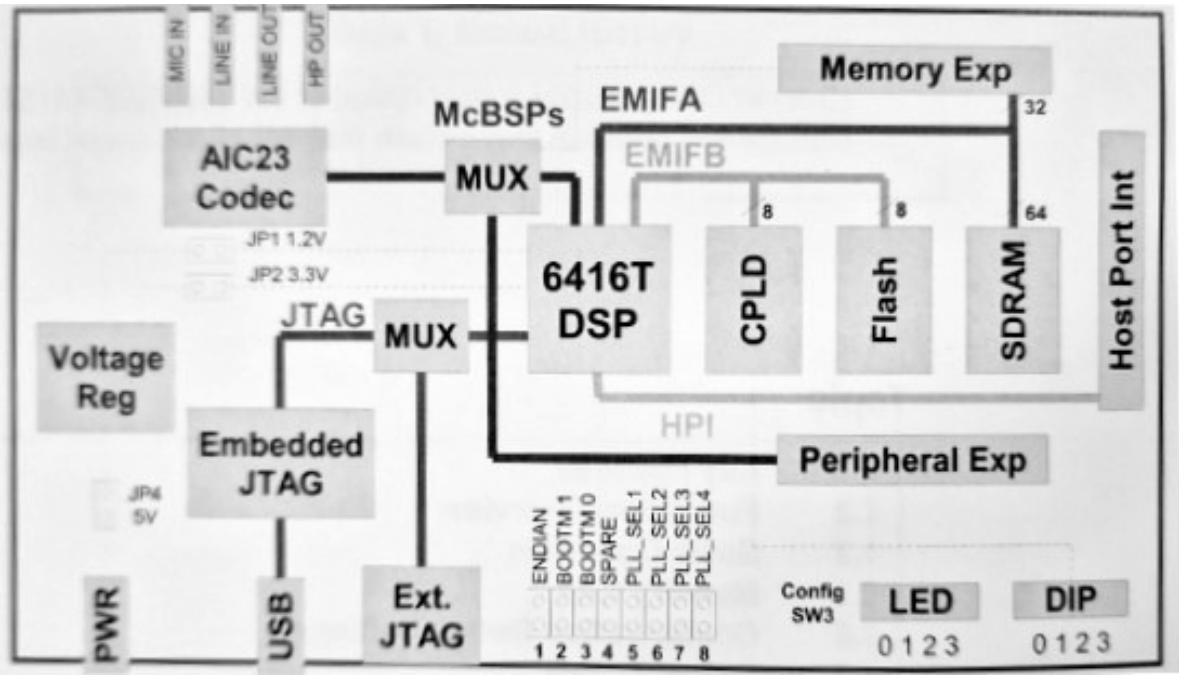


FIGURE 2 – Schéma technique extrait de la documentation.

2.2 Correspondance entre les éléments des schémas

À travers l'analyse des schémas, nous avons identifié les correspondances suivantes entre les éléments conceptuels et leurs contreparties physiques dans le kit DSP :

- Le **connecteur Jack** (MIC IN) correspond à l'entrée du signal original.
- Le **filtre passe-bas** est implémenté via le codec **AIC23**.
- La **conversion analogique-numérique** (CAN) est gérée par le codec **AIC23**.
- Le cœur du traitement, le **DSP**, est le **6416T DSP**.
- La **mémoire** est représentée par **Flash** et **SDRAM**.
- Les **ports E/S** sont gérés par le **MUX**.
- La **conversion numérique-analogique** est aussi réalisée par le codec **AIC23**.
- Enfin, le **connecteur Jack** (HP OUT) sert de sortie pour le signal traité.

3 Gestion des entrées avec les interrupteurs DIP

Cette section du rapport se concentre sur l'utilisation des interrupteurs DIP (Dual Inline Package) pour contrôler le comportement du DSP. Nous avons configuré les interrupteurs pour modifier la sortie audio et les indicateurs LED en fonction de l'état des interrupteurs.

3.1 Code source

Le code suivant illustre comment les entrées des interrupteurs DIP influencent la sortie audio et l'état des LEDs sur le DSP.

```
// Global scope
union {
    Uint32 uint;
    short channel[2];
```

```
} AIC_my_data;

// Inside c_int11()
// Son mono gauche
if (DSK6416_DIP_get(0) == 1) {
    DSK6416_LED_off(0);
} else if (DSK6416_DIP_get(0) == 0 && DSK6416_DIP_get(1) == 1) {
    output_left_sample(AIC_my_data.channel[0]);
    DSK6416_LED_on(0);
}

// Son mono droit
if (DSK6416_DIP_get(1) == 1) {
    DSK6416_LED_off(1);
} else if (DSK6416_DIP_get(1) == 0 && (DSK6416_DIP_get(0) == 1)) {
    output_right_sample(AIC_my_data.channel[1]);
    DSK6416_LED_on(1);
}

// Son stereo
if (DSK6416_DIP_get(0) == 0 && DSK6416_DIP_get(1) == 0) {
    output_sample(AIC_my_data.uint);
    DSK6416_LED_on(0);
    DSK6416_LED_on(1);
}

// TODO: Volume control to be implemented
if (DSK6416_DIP_get(2) == 1) {
    DSK6416_LED_off(2);
} else {
    DSK6416_LED_on(2);
}

// Coupure du son
if (DSK6416_DIP_get(3) == 1) {
    output_sample(0);
    DSK6416_LED_off(3);
} else {
    DSK6416_LED_on(3);
}
```

Listing 1 – Gestion des interrupteurs DIP sur le DSP.

3.2 Explications du code

Le code est structuré autour d'une union nommée `AIC_my_data`, qui permet l'accès aux données audio sous forme de deux canaux ou d'une valeur entière de 32 bits. L'interruption `c_int11()` est configurée pour répondre aux changements d'état des interrupteurs DIP.

Chaque bloc conditionnel correspond à un interrupteur DIP spécifique et contrôle soit l'état d'une LED, soit la manière dont l'audio est traité et sorti du DSP. Par exemple, lorsque le premier interrupteur DIP est activé (`DSK6416_DIP_get(0) == 1`), la LED 0 est éteinte et aucun son n'est sorti. Lorsque les interrupteurs sont configurés pour le mode stéréo (`DSK6416_DIP_get(0) == 0 & DSK6416_DIP_get(1) == 0`), les deux LEDs sont allumées et le son est sorti en stéréo.

La fonctionnalité de contrôle du volume reste à implémenter (`TODO`), où l'état de la LED 2 reflétera le mode de modification du volume. De même, la coupure du son est gérée par le quatrième interrupteur, avec l'état de la LED 3 indiquant si le son est coupé.

4 Calculs en format Q15

Le calcul en format Q15 est essentiel dans la représentation des nombres à virgule fixe en traitement de signal. Nous détaillons ici les étapes de conversion et de calcul pour l'équation linéaire $y = ax + b$ avec des coefficients en format Q15.

4.1 Conversion et calcul de y

Avec les valeurs données $a = 0.15$, $b = -0.3$, et $x = 0.25$, nous calculons y en base 10 :

$$y = 0.25 \times 0.15 - 0.3 = -0.2625$$

Ensuite, nous convertissons y en format Q15 :

$$y_{Q15} = -0.2625 \times (2^{15}) = -8601.6$$

Et en hexadécimal :

$$y_{Q15(10)} = -8601.6 \approx -2200_{(16)}$$

4.2 Calculs des coefficients en format Q15

Pour chaque coefficient, nous procédons à la conversion en format Q15 et à sa représentation hexadécimale.

Pour a :

$$a_{Q15} = 0.15 \times (2^{15}) = 4915.2_{(10)} \approx 1332_{(16)}$$

Pour b :

$$b_{Q15} = -0.3 \times (2^{15}) = -9830.4_{(10)} \approx -2666_{(16)}$$

Pour x :

$$x_{Q15} = 0.25 \times (2^{15}) = 8192_{(10)} = 2000_{(16)}$$

4.3 Calcul final de y_{Q15}

En utilisant les valeurs converties, nous obtenons :

$$y = a_{Q15} \times x_{Q15} + b_{Q15}$$
$$y = 4915.2 \times 8192 + (-9830.4)$$
$$y_{Q15(10)} = 9\,827\,333.4$$

5 Réalisation d'un oscillateur numérique

L'objectif est de générer un signal sinusoïdal à une fréquence déterminée. Pour ce faire, nous avons utilisé un filtre RII instable, dont l'équation caractéristique est donnée par :

$$y(n) = Ay(n-1) + By(n-2) + Cx(n-1)$$

où $A = 2\cos(\theta)$, $B = -1$, $C = \sin(\theta)$, et $\theta = 2\pi \frac{F_{dsire}}{F_{chantillonnage}}$. Ce filtre est une équation différentielle

d'ordre deux avec une réponse impulsionnelle infinie (RII), idéale pour notre oscillateur numérique.

5.1 Initialisation de l'oscillateur

Pour démarrer l'oscillateur, une impulsion est appliquée sur l'entrée à l'instant $t = T_1$. Les valeurs initiales sont calculées comme suit :

— Pour $n = 0$, $y(0) = A \times y(-1) + B \times y(-2) + C \times 0$
— Pour $n = 1$, $y(1) = A \times y(0) + B \times y(-1) + C \times 1$
— Et ainsi de suite pour les valeurs suivantes.

Nous avons réalisé un oscillateur de 2000 Hz et vérifié la précision de la fréquence obtenue.

5.2 Code pour l'oscillateur numérique

Le code suivant a été implémenté pour générer l'oscillateur sinusoïdal :

```
// Coefficients du filtre RII convertis en format Q14
#define A 23170
#define B -16384
#define C 11585

int32_t y_tab[2] = {0, C}; // Initialisation des valeurs y(n-1) et y(n-2)

interrupt void c_int11() {
    int32_t current_yn;

    // Calcul de y(n) en utilisant les valeurs precedentes et decalage pour division
    current_yn = (A * y_tab[1] + B * y_tab[0]) >> 14;

    // Mise a jour du tableau avec la nouvelle valeur calcul e
    y_tab[0] = y_tab[1];
    y_tab[1] = current_yn;

    // Sortie du signal calcule
    output_sample(current_yn);
}
```

Listing 2 – Code pour l'oscillateur numérique

Ce code est exécuté à chaque interruption pour calculer la valeur actuelle de $y(n)$ et générer le signal sinusoïdal. Les coefficients A, B, et C sont définis conformément à l'équation du filtre RII et sont ajustés au format Q14 pour une représentation à virgule fixe. Le décalage de 14 bits vers la droite ($>> 14$) est utilisé pour la division par 2^{14}, nécessaire en raison du format Q14.

5.3 Remarques

Des précautions particulières ont été prises pour les points suivants :

— Commencer le programme avec $n = 2$ pour éviter les références à des indices négatifs.
— Attention aux calculs pour les angles, étant donné que nous utilisons un format à virgule fixe.
— Le choix d'une fréquence d'échantillonnage de 8 kHz nous offre suffisamment de points pour une période précise du signal sinusoïdal.
— Utilisation de décalages pour les multiplications ou divisions par des puissances de 2, ce qui est essentiel en virgule fixe.

6 Implémentation d'un filtre passe-bas numérique

L'objectif de cette section est de décrire la mise en place d'un filtre passe-bas numérique avec une fréquence de coupure de 1kHz et une fréquence d'échantillonnage de 8kHz.

6.1 Théorie du filtre passe-bas

Un filtre passe-bas analogique est généralement défini par la fonction de transfert suivante :

$$H(p) = \frac{1}{1 + \tau p}$$

où τ représente la constante de temps du filtre et p est la variable de Laplace. La fréquence de coupure F_c est définie comme $F_c = \frac{1}{2\pi\tau}$.

Pour convertir ce filtre analogique en un filtre numérique, nous utilisons la transformation bilinéaire qui relie la variable continue p à la variable discrète z :

$$p \to \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}}$$

Le filtre numérique est alors décrit par l'équation aux différences :

$$y(n) = Ax(n) + Bx(n-1) + Cy(n-1)$$

où $y(n)$ est la sortie du filtre à l'instant n, et $x(n)$ est l'entrée.

6.2 Code d'implémentation

Le filtre passe-bas a été implémenté dans le cadre d'une routine d'interruption comme suit :

```
// Coefficients du filtre passe-bas en format Q14
#define A_LOW_PASS 4619
#define B_LOW_PASS 4619
#define C_LOW_PASS 7144

// Initialisation des valeurs precedentes de x et y
short x_n_1 = 0;
short y_n_1 = 0;

interrupt void c_int11() {
    short x_n;
    short y_n;
    // Lecture de l'echantillon d'entree
    x_n = input_left_sample();
    // Application de l'equation du filtre passe-bas
    y_n = (((A_LOW_PASS * x_n) + (B_LOW_PASS * x_n_1) + (C_LOW_PASS * y_n_1)) >> 14);

    // Sortie de l'echantillon filtre
    output_left_sample(y_n);

    // Mise a jour des valeurs precedentes de x et y
    y_n_1 = y_n;
    x_n_1 = x_n;
}
```

Listing 3 – Implémentation du filtre passe-bas numérique

6.3 Vérification et tests

Le filtre a été testé en alimentant le DSP avec un signal sinusoïdal de fréquence variable. L'entrée et la sortie du filtre ont été observées à l'aide d'un oscilloscope pour vérifier le bon fonctionnement du filtre et la conformité de la fréquence de coupure.

Ces tests ont confirmé que le filtre passe-bas numérique fonctionne comme prévu, atténuant les fréquences au-dessus de 1kHz et laissant passer celles en dessous de la fréquence de coupure.

7 Conception et implémentation de filtres RIF

Dans cette partie du TP, nous avons abordé la conception de filtres RIF (Réponse Impulsionnelle Finie) pour différents types de réponses fréquentielles : passe-bas, passe-haut et passe-bande.

7.1 Principe des filtres RIF

Les filtres RIF sont caractérisés par des équations aux différences de la forme :

$$y(n) = h_1 x(n) + h_2 x(n-1) + \ldots + h_k x(n-k+1)$$

Ces équations sont représentées dans le diagramme fonctionnel comme une série de retards (Z^{-1}), de multiplications et d'additions, constituant ainsi un filtre numérique.

7.2 Méthodologie de conception

Pour réaliser ces filtres, il est nécessaire de calculer les coefficients h_i selon les paramètres du filtre tels que l'ordre, la fréquence d'échantillonnage et la fréquence de coupure. Nous avons utilisé MATLAB pour déterminer ces coefficients à l'aide de la fonction FIR2.

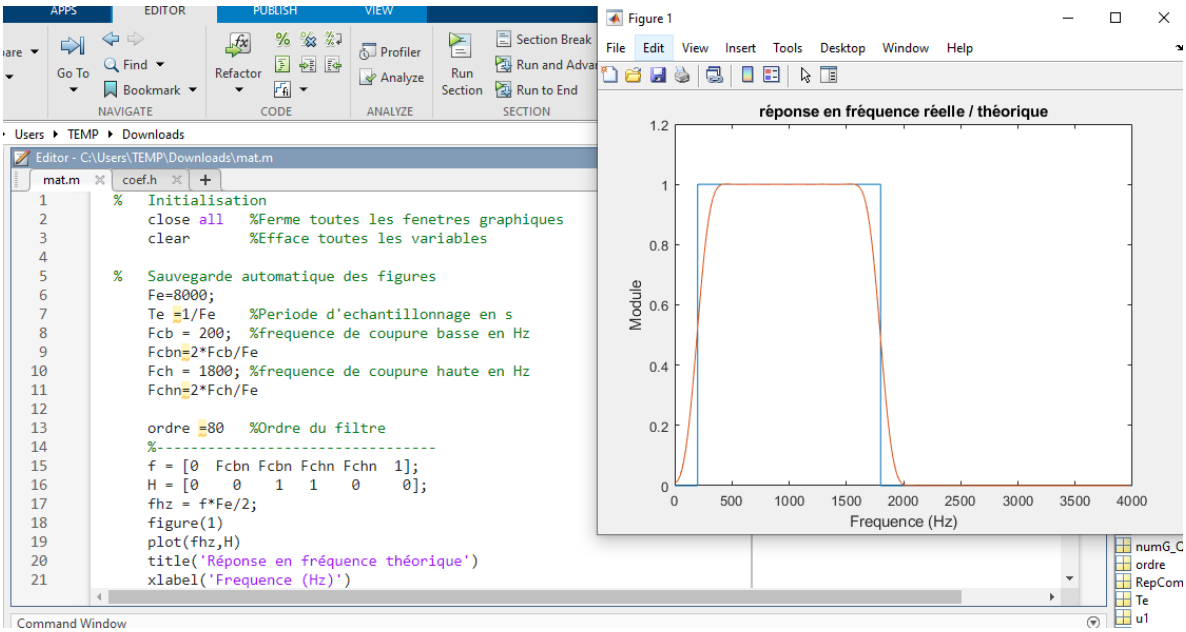


FIGURE 3 – Passe-bande

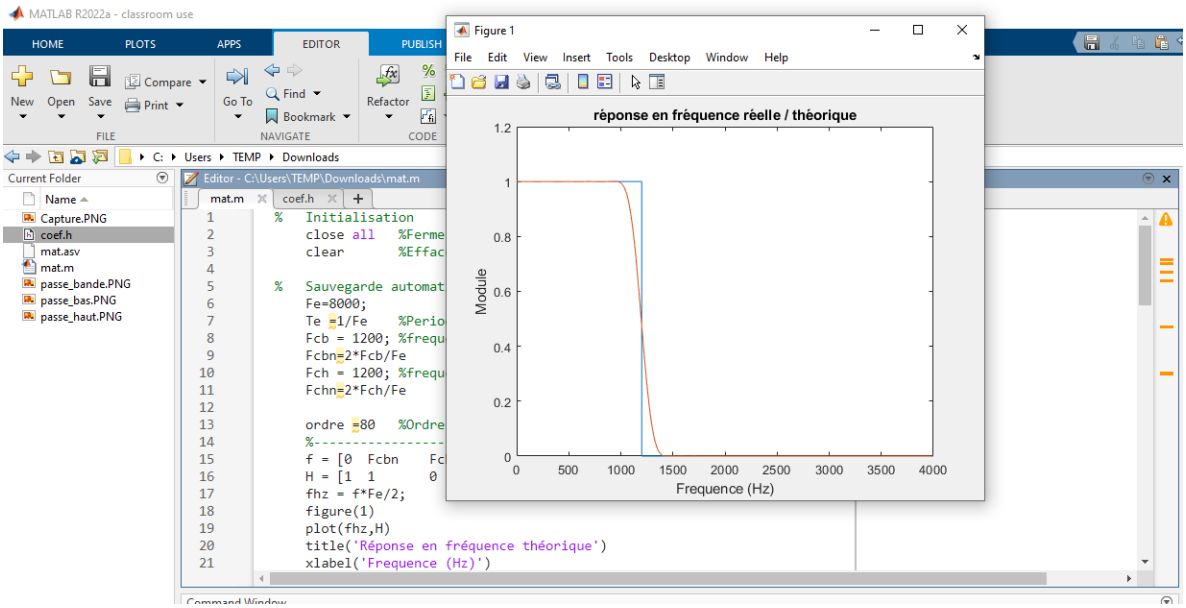


FIGURE 4 – Passe-bas

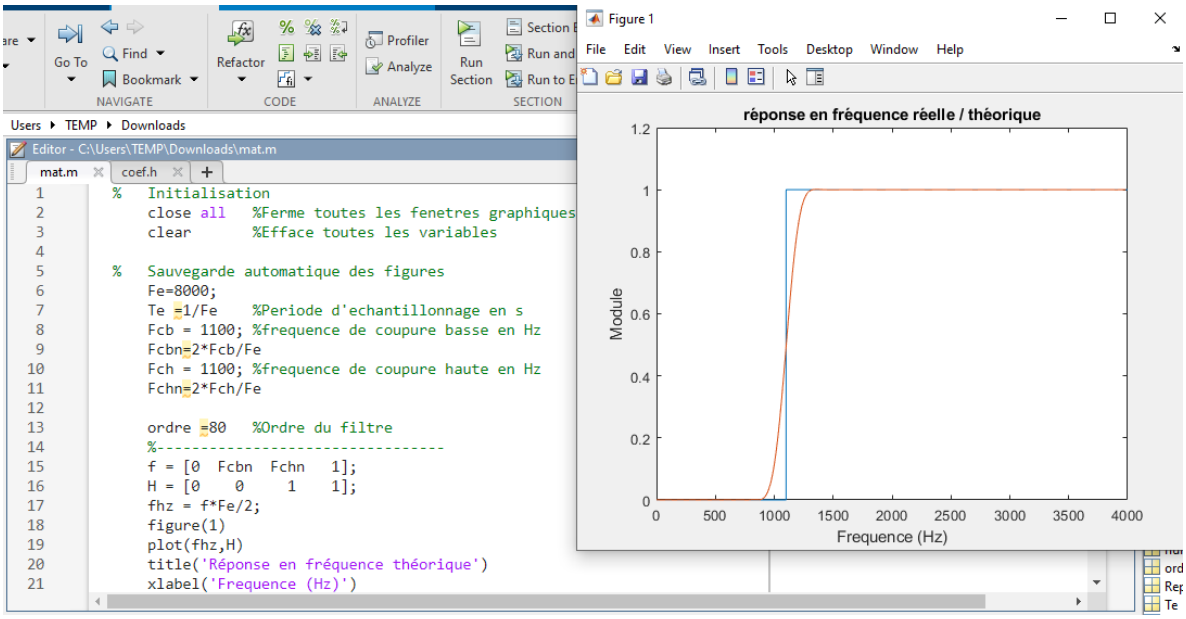


FIGURE 5 – Passe-haut

7.3 Implémentation des filtres sur le DSP

Les coefficients calculés ont été intégrés dans notre code DSP pour la création des filtres RIF. Voici le code d'implémentation :

```
// Coefficients des filtres RIF
short coef_bande[81] = { /* Coefficients du passe-bande */ };
short coef_haut[81] = { /* Coefficients du passe-haut */ };
short coef_bas[81] = { /* Coefficients du passe-bas */ };

short filterStatus;
short x_n_tab[81] = {0};
short x_n;

interrupt void c_int11() {
    int y_n = 0;
    int i;

    x_n = input_left_sample();

    // D calage des   chantillons
    for (i = 80; i >= 0; i--) {
        x_n_tab[i] = x_n_tab[i-1];
    }
    x_n_tab[0] = x_n;

    // Application des coefficients selon le type de filtre selectionne
    switch(filterStatus) {
        case 0: // Passe-bande
            for (i = 0; i < 81; i++) {
                y_n += coef_bande[i] * x_n_tab[i];
            }
            break;
        case 1: // Passe-haut
            for (i = 0; i < 81; i++) {
                y_n += coef_haut[i] * x_n_tab[i];
            }
            break;
        case 2: // Passe-bas
            for (i = 0; i < 81; i++) {
                y_n += coef_bas[i] * x_n_tab[i];
            }
            break;
    }

    y_n = y_n >> 15; // Ajustement de la sortie
    output_left_sample(y_n);
}
```

Listing 4 – Code pour les filtres RIF sur le DSP

8 Suppression de fréquences parasites

8.1 Filtre coupe-bande pour le traitement audio

Nous avons été confrontés à un fichier audio corrompu par des fréquences parasites. Notre objectif était d'identifier ces fréquences et de concevoir un filtre coupe-bande pour les éliminer et récupérer le signal audio original.

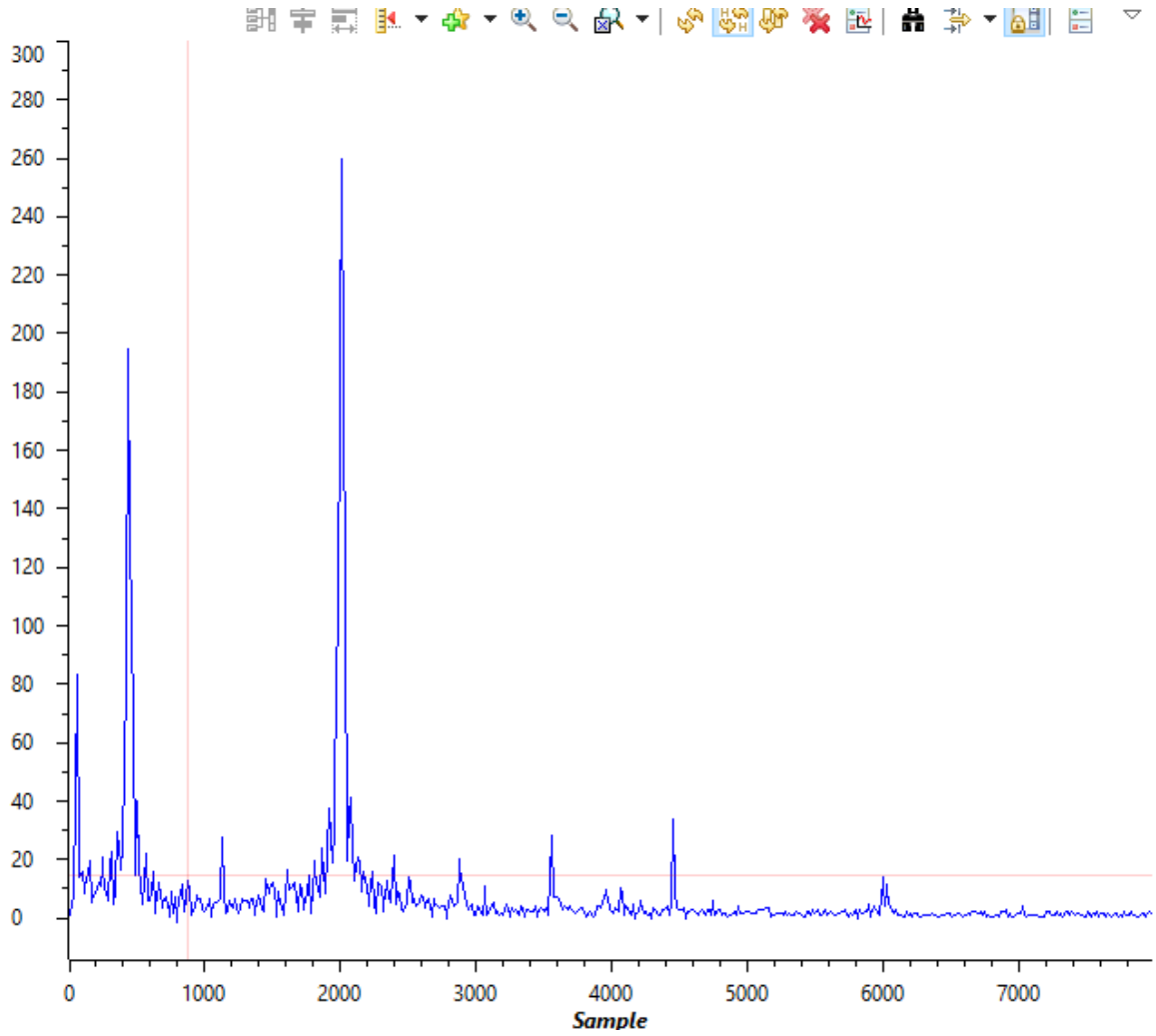


FIGURE 6 – Signal pollué

8.1.1 Identification des fréquences parasites

Le fichier audio "sound_pb.mp3" a été analysé en utilisant l'outil FFT (Fast Fourier Transform) du logiciel. Cela a permis de détecter les fréquences indésirables présentes dans le signal.

8.1.2 Conception du filtre avec MATLAB

Nous avons utilisé MATLAB pour concevoir un filtre coupe-bande approprié. Les fréquences de coupure ont été déterminées à partir de l'analyse FFT et les coefficients du filtre ont été calculés comme suit :

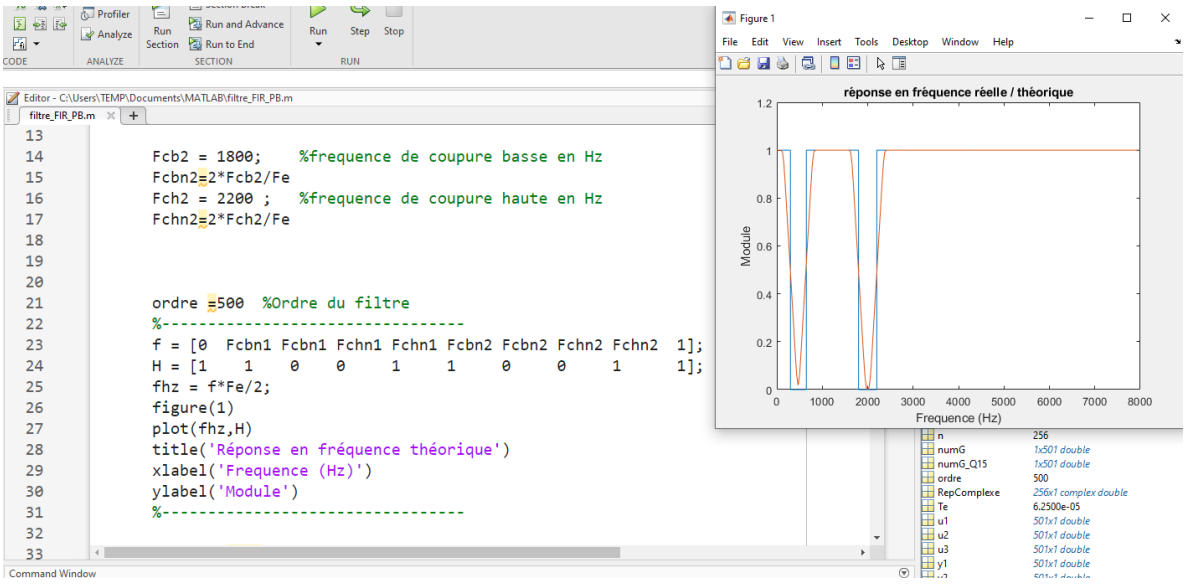


FIGURE 7 – Double filtre coupe-bande

8.2 Implémentation du filtre sur le DSP

Une fois les coefficients du filtre établis, nous avons procédé à leur implémentation dans le DSP pour filtrer le signal audio corrompu. Voici le code C utilisé :

```c
// Coefficients du filtre coupe-bande
short coef_double_bande[501] = { /* Coefficients generes par MATLAB */ };
short x_n_tab[501] = {0};
interrupt void c_int11() {
    int y_n = 0;
    int i;


    // Lecture de l'echantillon d'entree et misea  jour du buffer
    x_n = input_left_sample();
    for (i = 500; i >= 0; i--) {
        x_n_tab[i] = x_n_tab[i-1];
    }
    x_n_tab[0] = x_n;

    // Application du filtre coupe-bande
    for (i = 0; i < 500; i++) {
        y_n += coef_double_bande[i] * x_n_tab[i];
    }
    y_n = y_n >> 15; // Normalisation du r sultat

    // Sortie de l'echantillon filtre
    output_left_sample(y_n);
}
```

Listing 5 – Code C pour le filtre coupe-bande

Le filtre a été testé pour s'assurer qu'il supprimait efficacement les fréquences parasites et restaurait le signal audio à son état original. Les résultats ont confirmé la réussite de la suppression des fréquences indésirables.

9 Conclusion

Le TP réalisé a été une exploration approfondie des principes fondamentaux du traitement du signal numérique. Nous avons non seulement renforcé notre compréhension théorique des filtres numériques mais aussi acquis une expérience pratique précieuse en concevant et en implémentant plusieurs types de filtres numériques. Cette démarche nous a permis de comprendre l'impact direct de la théorie du traitement du signal sur des applications concrètes, comme l'amélioration de la qualité d'un fichier audio. Ces acquis forment une base solide qui nous équipera pour aborder avec confiance des projets d'ingénierie plus complexes dans le futur.