RAPPORT DE STAGE

Préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

En vue de l'obtention du
Diplôme de Génie Technique de Télécommunications de l'Université Polytechnique de Catalogne

Spécialité
Systèmes électroniques

par
Josep CAUBET GOMÀ

Traitement du signal et circuit électronique associé au capteur de gaz SnO_2

Maître de stage: M. Philippe MENINI

Juin 2004

UNIVERSITE PAUL SABATIER TOULOUSE III

CNRS LAAS

UPC

RAPPORT DE STAGE

Préparée au
Laboratoire d'Analyse et d'Architecture des Systèmes du CNRS

En vue de l'obtention du
Diplôme de Génie Technique de Télécommunications de l'Université Polytechnique de Catalogne

Spécialité
Systèmes électroniques

par
Josep CAUBET GOMÀ

Traitement du signal et circuit électronique associé au capteur de gaz SnO_2

Maître de stage: M. Philippe MENINI

Juin 2004

À mes parents qui m'ont toujours soutenu,

Remerciements

Après trois mois de travail, je tenais à remercier tout d'abord les responsables du programme Erasmus pour m'avoir permis de réaliser ce séjour en France, à Toulouse.

Je tiens à remercier Monsieur Malik Ghallab, directeur du LAAS, pour m'avoir accueilli au sein du laboratoire.

Je remercie également Messieurs Philippe Ménini, maître de conférence à l'Université Paul Sabatier, Frédéric Parret, doctorant, qui m'ont proposé ce stage. Je tiens tout particulièrement à remercier Monsieur Stève Josse, post-doctorant, pour sa disponibilité, sa gentillesse et ses conseils, notamment pour la rédaction de ce rapport.

J'exprime aussi ma gratitude à toute l'équipe Technologie Micro et Nanostructures (TMN) sous la direction de Monsieur Gérard Sarrabayrouse.

Enfin, je remercie les responsables du département de l'Université Polytechnique de Catalogne pour leur disponibilité et les stagiaires partageant mon bureau pour leur gentillesse.

SOMMAIRE

I Introduction

Ce rapport est le résultat de mon stage que j'ai réalisé au LAAS-CNRS (Laboratoire d'Analyse et d'Architecture des Systèmes) dans le groupe TMN (Technologie de Micro et Nanostructures) pendant trois mois. Ces travaux ont été réalisés en collaboration avec Stève Josse, post-doctorant, Frédéric Parret, doctorant et Philippe Ménini, maître de conférences.

Durant ce stage, j'ai travaillé sur la partie électronique du traitement du signal de nouvelles générations de capteurs de gaz à base de couches sensibles nanoparticulières (SnO_2, WO_3, In_2O_3). Les principaux points forts de ces capteurs de gaz SnO_2 sont :
- Une durée de vie liée au vieillissement des composants électroniques, donc très supérieurs aux capteurs électrochimiques (≈ 10 ans).
- Une taille du module de détection réduite.
- Une mise en œuvre aussi simple que celle des capteurs électrochimiques.

Suivant ces avantages, les applications sont multiples et touchent plusieurs secteurs :
- Le secteur de l'automobile (le plus important).
- Le secteur domotique.
- Le secteur de la santé et l'environnement.

L'objectif de mon stage était d'étudier les différentes architectures possibles pour réaliser le circuit électronique et de choisir l'une d'elle dans l'optique de rendre le système complet portatif (capteur et circuit électronique). Ce système portatif rendrait les micro-capteurs de gaz encore plus attractifs par leurs faibles dimensions et leur faible consommation à oxydes métalliques.

Mon rapport se compose en trois parties :
- Une première partie qui introduit la description et le fonctionnement des capteurs de gaz étudies ainsi qu'une étude sur les méthodes mathématiques principales de discrimination et d'extraction de données utilisées dans les nez électroniques.

- Une seconde partie portant sur l'étude des différentes architectures possibles pour réaliser le circuit électronique de traitement du signal et sur l'architecture sélectionnée suivant notre cahier des charges.

- Une dernière partie qui concerne la programmation d'un cas concret de discrimination de 3 gaz dans le cadre d'une application automobile (NO_2, C_3H_8, CO). Cette application va aussi nous permettre d'obtenir des règles simples pour choisir le composant constituant l'architecture de notre circuit électronique.

II Capteur de gaz étudié et les méthodes mathématiques appliquées au traitement du signal

II.1 Capteur de gaz à SnO_2 nanoparticulaire

II.1.1 Généralités

Le rôle de notre capteur de gaz est de discriminer 3 types de gaz, deux gaz réducteurs, le monoxyde de carbone (CO) et le propane (C_3H_8) et un gaz oxydant, le dioxyde d'azote NO_2. En outre, le capteur doit être capable dans un premier pas de discriminer ces gaz avec ou sans mélange gazeux et dans un second temps de donner les concentrations pour chaque gaz.

Les résultats de discrimination et de concentration des gaz seront donnés par le circuit de traitement du signal à partir de la variation ohmique de la couche sensible du capteur, une couche mince de SnO_2. Cette couche sensible est commandée par un élément chauffant, appelé "heater", pour atteindre une température de fonctionnement donnée.

La *figure 2.1* représente le schéma fonctionnel du capteur de gaz et de son environnement, les circuits de traitement du signal et de commande du heater.

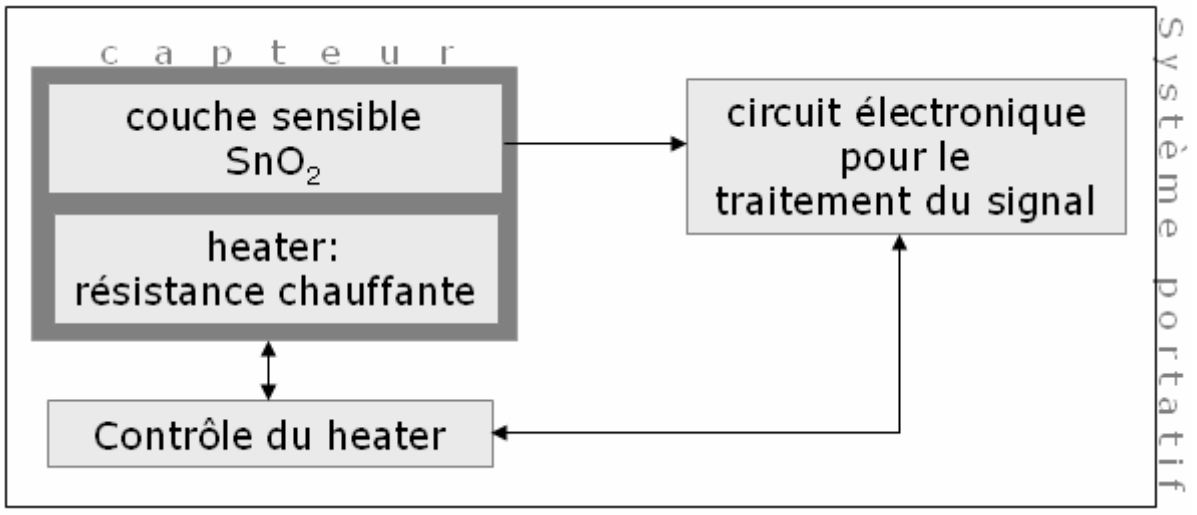


Figure 2.1 : Schéma synoptique du capteur et des circuits de traitement du signal et du contrôle du heater.

Afin de discriminer les gaz et de donner leurs concentrations, Les caractéristiques de la couche sensible sont :

- La sensibilité, définie par la variation relative de conductance :

$$S = \frac{G - G_0}{G_0} \tag{2.1}$$

Avec G_0 : la conductance sous concentration de référence par un gaz donné.
G : la conductance sous une concentration donnée.

- La sélectivité, pour distinguer différents gaz dans un mélange. Or la conductance de la couche SnO_2 est fonction de la température. Il faut donc trouver des conditions de fonctionnement spécifiques qui favorisent telle ou telle interaction.

- La réversibilité, pour caractériser le fait que la conductance retourne à sa valeur initiale après avoir été modifiée par la présence d'un gaz réducteur et que le capteur continue à bien fonctionner sans avoir été « empoisonné ».

- Le temps de réponse qui exprime le temps nécessaire pour stabiliser la valeur en sortie du capteur quand les conditions de mesures varient brutalement d'un état à un autre. Dans le cas des capteurs de gaz, cette valeur dépend essentiellement de la cinétique des réactions chimiques mises en jeu.

- La stabilité, si les performances évoluent dans le temps de façon irréversible, les réponses ne seront plus significatives des concentrations de polluant. Ces phénomènes d'évolution temporelle sont appelés « dérives ».

II.1.2 Description et fonctionnement du capteur étudié

Le capteur de gaz à oxyde métallique fonctionne selon la variation de la conductivité électrique (résistivité) de sa couche sensible. Celle-ci est chauffée à haute température en présence d'une atmosphère gazeuse pour discriminer le gaz désiré, de par le principe di0oxyde-réduction.

Lorsque l'oxygène se met en contact avec la couche sensible, l'adsorption du gaz produit une variation du nombre d'électrons entraînant un changement de la distribution de charge superficielle. Lorsque l'épaisseur de la couche est comparable au niveau de modification de charge, la conductivité de la couche est donc modulée par l'adsorption chimique. Puis pour que la conductivité reprenne sa valeur de référence ou une valeur plus élevée, il suffit qu'un agent réducteur s'applique sur la surface subissant une réduction avec l'oxygène, formant un composé neutre et libérant des électrons pour la conduction. Afin d'améliorer les caractéristiques, sélectivité et sensibilité, il est possible d'introduire dans la couche sensible des agents catalytiques (métalliques) tels que le platine (Pt), le palladium (Pd) ou l'argent (Ag), l'argent dopant est fonction du gaz ciblé et des interférents présents dans une application donnée. Enfin, pour favoriser la formation du composé neutre et s'affranchir des interférences dues aux molécules d'eau, le système doit être chauffé à une certaine température ou bien subir un certain cycle thermique.

a) Structure du capteur

C'est une plate-forme microélectronique composée de plusieurs éléments (*figure 2.2*) :
- Une couche sensible de SnO_2.
- Une résistance chauffante (heater).
- Des métallisations pour les contacts.
- Une membrane sur support silicium.

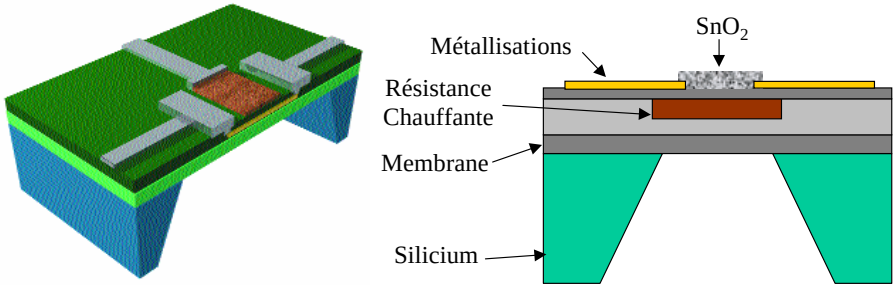


Figure 2.2 : Structure du capteur SnO_2 micro-usiné.

La couche sensible est la partie du capteur qui est sensible aux différents gaz. Les réactions qui déterminent la sensibilité du dioxyde d'étain sont essentiellement des réactions de surface. La technique mise en œuvre pour l'élaboration de la couche sensible ainsi que la température de recuit auront donc une importance sur sa microstructure. Pour augmenter la sensibilité de la couche SnO_2, nous introduisons dans cette couche, des dopants de types Platine (Pt), Palladium (Pd) à 2% et 4%.

La résistance chauffante à pour but de chauffer la couche sensible jusqu'à sa température de fonctionnement. Elle se situe sous la couche sensible et est isolée électriquement de celle-ci et de la membrane (dans le cas d'une membrane silicium) par une couche oxyde de silicium (SiO_2). Cette isolation est indispensable pour éviter des courts-circuits qui engendreraient un dysfonctionnement du capteur.
La résistance chauffante en polysilicium est un élément essentiel du dispositif de détection. En effet, celle-ci va conditionner l'homogénéité en température sur la couche sensible. Lors de la conception du capteur, une attention toute particulière devra donc être apportée à sa géométrie, car la température de la couche sensible a une influence à la fois sur la sélectivité et sur la sensibilité de la couche sensible [1].

La membrane est utilisée pour limiter les pertes thermiques par conduction. Celle-ci doit posséder de bonnes propriétés thermomécaniques pour supporter l'ensemble du dispositif mais aussi avoir une faible conductivité thermique. Elle est réalisée en Si_3N_4 et a pour dimensions 1000 μm sur 800 μm et 1,5 μm d'épaisseur [1].

Les plots de métallisations permettent les contacts électriques sur la couche sensible et la résistance chauffante. De ce fait, nous pourrons d'une part alimenter la résistance chauffante pour l'amener à la température désirée et de l'autre, mesurer la résistance de la couche sensible. Ces plots de métallisations supporteront des températures maximales de l'ordre de 300ºC et devrons éviter les phénomènes d'interdiffusion et d'électromigration. Pour cela, les contacts utilisés sont en Cr-Ti-Pt [2].

b) Mesure de la résistance de la couche sensible

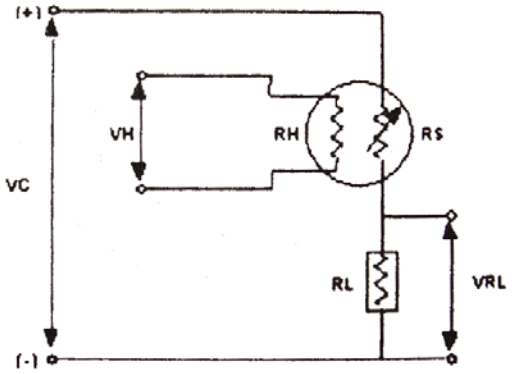
La méthode retenue pour mesurer la résistance de la couche sensible SnO_2 (R_S) est la lecture d'une tension (V_{RL}) sur la résistance de charge (R_L) placée en série avec la couche du capteur (*figure 2.3*) [3].

De la *figure 2.3*, l'équation du pont diviseur de tension est :

$$V_{RL} = \frac{V_c \cdot R_L}{R_S + R_{RL}} \tag{2.2}$$

Nous faisons l'acquisition de la tension V_{rl} et en utilisant l'équation précédente, nous calculons R_S :

$$R_s = \frac{V_c \cdot R_l}{V_{RL}} - R_{RL} \tag{2.3}$$



V_H : Tension d'alimentation du heater (élément chauffant).
V_C : Tension d'alimentation de la couche sensible (résistance R_s) et de la résistance de charge R_L.
V_{rl} : Tension lue aux bornes de la résistance de charge.
R_h : Résistance du heater (élément chauffant).
R_s : Résistance de la couche sensible SnO_2.
R_l : Résistance de charge.

Figure 2.3 : Principe de mesure de la résistance de la couche sensible du capteur.

Cette méthode présente l'inconvénient de donner une image électrique inversement proportionnelle à R_s. De ce fait, la précision est perdue pour les grandes valeurs de R_s. Ainsi, R_l sera choisie afin d'avoir un signal V_{rl} suffisamment grand, tout en conservant $R_s >> R_l$.

Remarque : Pour augmenter la sensibilité et la sélectivité des nez électroniques SnO_2, une matrice de capteurs avec différents dopants est généralement utilisée comme le représente la *figure 2.4* [3].

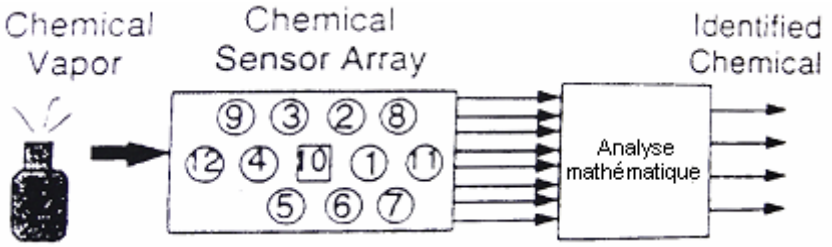


Figure 2.4 : Représentation schématique d'une matrice de capteur de gaz SnO_2.

II.2 Méthodes mathématiques de discrimination et d'extraction des données

II.2.1 Méthode ACP (Analyse en Composantes Principales)

La méthode ACP (Analyse en Composantes Principales) a été développée en 1901 par K.Pearson, puis adaptée à la statistique en 1933. Le procédé consiste à construire de nouveaux caractères synthétiques à partir de facteurs obtenus par combinaisons linéaires des variables initiales. Ces nouveaux caractères synthétiques sont appelés composantes principales et sont assimilés à des axes dans une représentation graphique. La problématique consiste alors à trouver les n composantes principales (dans un nombre de dimensions le plus réduit) qui expliquent au mieux la plus grande quantité d'informations originales. Ainsi, pour un tableau de données représenté dans un espace à deux dimensions, la première composante, l'axe des abscisses, expliquera la plus grande

partie de l'information d'origine. Puis, la seconde composante, l'axe des ordonnées, expliquera la plus grande partie de l'information restante.

La mise en œuvre mathématique de l'ACP peut être divisée en 6 étapes.

a) Etape 1 : Préparer les données pour le traitement

Les informations d'origines doivent d'abord être toutes stockées dans un tableau (matrice) selon une certaine disposition. On représente la matrice X origine, la matrice de type individus / variables, de la forme suivante :

$$X = \text{individus...}\ i \begin{array}{c} \\ 1 \\ \\ \\ n \end{array} \begin{array}{c} \text{variables} \\ 1 \quad j \quad p \\ \begin{pmatrix} x_1^1 & \dots & x_p^1 \\ & x_j^i & \\ x_1^n & \dots & x_p^n \end{pmatrix} \end{array}$$

- *p variables, représentées en colonnes.*
- *n individus, représentés en ligne.*
- *des valeurs prises par chaque variables, pour chaque individu, notées :*

$$\left(x_j^i\right)_{(1\le i\le n, 1\le j\le p)} avec \forall (i,j), x_j^i \in \Re.$$

Il faut tout d'abord traiter cette matrice de données originales de manière à ne pas privilégier des variables particulières. Pour cela, nous allons normaliser cette matrice pour éliminer les problèmes de données mesurées dans des unités différentes, soit une nouvelle matrice Xcr, une version centrée-réduite.

b) Etape 2 : Calculer la matrice des coefficients de corrélations des variables

Après avoir effectué la transformation de la matrice initiale, nous allons calculer à partir de Xcr, la matrice d'inertie V, encore appelée matrice de corrélation. Cette matrice représente l'ellipsoïde d'inertie pour laquelle nous allons rechercher les nouveaux axes de symétrie.
Cette matrice d'inertie est calculée à l'aide de la matrice de covariance et s'exprime de la manière suivante :

$$V_{(i,j)} = \frac{C_{(i,j)}}{\sqrt{C_{(i,i)}.C_{(j,j)}}} \quad \text{Avec} \quad C_{(i,j)} = \mathrm{E}((i - \mathrm{E}(i).(j - \mathrm{E}(j)) \qquad (2.4)$$

$C_{(i,j)}$: Covariance ; E(x) :Espérance mathématique.

c) Etape 3 : Extraire les valeurs propres et vecteurs propres de la matrice précédente

Nous désirons maintenant déterminer les axes factoriels. La formulation mathématique de l'ACP est alors la suivante :

- Trouver le sous-espace affine E_K de dimension K (k<p) tel que I_{Ek}, inertie du nuage N par rapport à l'espace E_K soit minimum.

Sachant que nous travaillons sur une matrice de données centrées - réduites, E_K est déjà un sous-espace vectoriel. Dans ce cas, l'inertie totale se décompose en une somme I_{Ek} + I_{Ek1}, où I_{Ek1} est l'inertie expliquée par l'orthogonal de E_K. La formulation mathématique peut maintenant s'expliquer de la manière suivante :

- Trouver le sous-espace affine E_K de dimension K (k<p) tel que I_{Ek}, inertie du nuage N par rapport à l'espace E_K soit minimum, c'est à dire, tel que l'inertie I_{Ek1} soit maximale.

En d'autre terme, afin de déterminer un plan factoriel, la méthode consiste à rechercher :
- L'axe Δu_1 maximisant l'inertie $I_{\Delta u1}$, moyenne des carrés des distances.
- L'axe Δu_2, perpendiculaire à Δu_1, maximisant l'inertie $I_{\Delta u2}$.
- (…).
- L'axe Δu_k, perpendiculaire à Δu_{k-1}, maximisant l'inertie $I_{\Delta uk.}$

Ainsi, en projetant l'individu *i* ayant pour coordonnées initiales x_{i1}, x_{i2},.., x_{ip}, les nouvelles coordonnées c_{i1}, c_{i2},…, c_{ip}, seront données par l'équation suivante :

$$CP^k = \boldsymbol{u_1^k}.\boldsymbol{x_1} + \boldsymbol{u_2^k}.\boldsymbol{x_2} + \ldots + \boldsymbol{u_p^k}.\boldsymbol{x_p} \qquad (2.5)$$

Où x_{i1}, x_{i2},.., x_{ip} sont les caractères mesurés sur les n individus et les coefficients $\boldsymbol{u_1^k}$, $\boldsymbol{u_2^k}$,…, $\boldsymbol{u_p^k}$ forment le k^{ième} facteur principal.

La solution est alors obtenue en utilisant les propriétés spectrales de la matrice d'inertie où les facteurs principaux sont les vecteurs propres de la matrice $V_{(i,j)}$. Ces vecteurs propres sont associés à des valeurs propres obtenues par diagonalisation de la matrice $V_{(i,j)}$ dont la relation est régie par l'expression suivante :

$$V \cdot T = T \cdot S \qquad (2.6)$$

Avec V, la matrice d'inertie, T, la matrice des vecteurs propres et S, la matrice diagonale des valeurs propres.

A chaque axes principaux k obtenus, est associée une valeur propre représentant le pourcentage d'inertie ou encore la variance expliquée par l'axe k.

d) Etape 4 : Classer les vecteurs propres dans l'ordre décroisant des valeurs propres associées

Nous disposons dans l'ordre décroissant, des facteurs principaux représentant les nouveaux axes et la part d'importance de chaque axe. Il suffit maintenant de décaler les colonnes de la matrice des vecteurs propres afin d'obtenir une nouvelle matrice appelée *u*, la matrice des k facteurs principaux dans l'ordre croissant de l'importance de chaque axe.

e) Etape 5 : Calculer la matrice des composantes principales

La matrice appelée matrice des composantes principales est celle qui contient les coordonnées des individus dans l'espace formé par les composantes principales. Pour cela, il suffit de multiplier la matrice centrée-réduite par la matrice des vecteurs propres.

f) Etape 6 : Fournir une ou plusieurs représentations graphiques

Comme nous l'avons expliqué auparavant, le but de la méthode ACP est de fournir un résumé des données originales et une analyse graphique la plus simplifiée et la plus lisible. Cette dernière étape est donc, à priori, la phase finale la plus importante de ce processus car elle permet d'avoir rapidement un aperçu des données originales dans son résumé.

L'ACP peut fournir deux types de représentations graphiques :

- **Les diagrammes de dispersion (*figure 2.5*) :**

Ce premier type de graphique permet de représenter les individus sous forme de nuages de points dans des plans ou espaces factoriels de 2 ou 3 composantes principales. Ce type de représentation montrent comment se situe chaque individu par rapport aux composantes principales, et donc de pouvoir identifier des cas isolés ou des regroupement entre individus. Dans cet espace, les proximités s'interprètent en termes de similitudes.

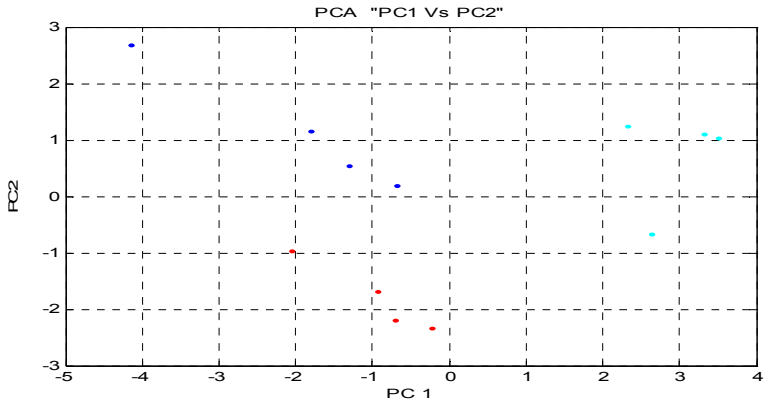


Figure 2.5 : représentation d'un diagramme de dispersion

- **Les cercles de corrélation (*figure 2.6*) :**

Il s'agit cette fois-ci de représenter les variables dans des systèmes d'axes factoriels à 2 composantes principales, munis d'un cercle de rayon 1 pour aider l'interprétation. Ce type de graphique permet de visualiser facilement les relations existantes entre les variables : la force de ces relations et quelles composantes principales les expliquent le mieux.

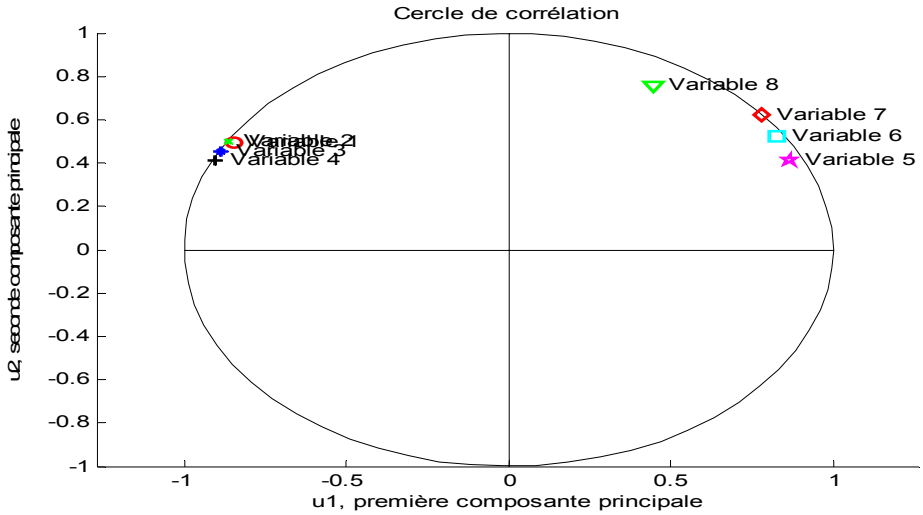


Figure 2.6 : Représentation d'un diagramme de cercle de corrélation

Il faut savoir maintenant exprimer les résultats obtenus graphiquement. Afin d'interpréter les résultats de la méthode ACP, des méthodes empiriques et des méthodes numériques sont utilisées.

Pour l'interprétation, nous utilisons en premier le graphique du cercle de corrélation, soit les variables. Les méthodes empiriques sont des règles basées seulement sur l'observation visuelle du graphique. Nous pouvons citer les deux règles principales qui sont :

- Plus une variable est proche de la périphérie du cercle de corrélation et d'un des axes (c'est à dire plus sa coordonnée sur cet axe est proche de 1 ou -1), plus elle est corrélée à cet axe (plus cette composante principale explique cette variable).
- Plus deux variables sont proches sur le graphe, plus leur lien est fort.

Dans cet exemple on distingue clairement la corrélation entre les variables 1, 2, 3 et 4 et une corrélation plus faible entre les variables 5 à 8.

Ces méthodes visuelles sont assez efficaces afin d'obtenir les premières conclusions autant en terme de corrélation de variables que pour l'importance de chaque axe pour les

variables. Mais ces méthodes comportent des contraintes lorsque les variables ne sont pas à la périphérie du cercle (car les règles ne sont plus applicables) ou lorsque nous utilisons un nombre de variables assez élevées. Pour cela, nous associons des méthodes numériques, principalement pour l'étude des individus et non plus des variables.

Les méthodes numériques consistent par des calculs mathématiques déterminés par la méthode ACP, de nous fournir des indicateurs mathématiques qui expriment des caractéristiques particulières du système axes/individus/variables.

Parmi les indicateurs numériques existants, nous pouvons citer les plus utilisés :
- Le pourcentage d'information initiale expliqué par chaque composante principale, ou axe. Si le pourcentage de contribution d'un axe est inférieur à 10%, on peut considérer qu'il est inintéressant.
- La qualité de la représentation des individus sur chaque axe (matrice appelée cos^2, car elle utilise le carré du cosinus de l'angle formé par le vecteur de l'individu et les axes). Si le facteur cos^2 est le plus proche de 1, alors l'individu est bien représenté par sa projection sur le plan. Ce facteur est utilisé pour interpréter les points centraux.
- La contribution de chaque individu à chaque axe ou par rapport à tout le nuage. Dans ce cas, nous calculons le centroïde de chaque groupe d'individus et on regarde la dispersion de chaque individu par rapport au centroïde.

En conclusion, nous pouvons dire que les avantages de cet outil sont avant tout :
- La simplicité de mise en œuvre d'un point de vue mathématique.
- La simplicité des résultats grâce aux graphiques qu'elle fournit.
- La puissance car cette méthode exprime au mieux un résumé des données initiales et une vue complète des relations existant entre variables et individus.
- La flexibilité car celle-ci traite un ensemble de données de contenu et de tailles quelconques.

L'ACP n'a pas réellement d'inconvénients. Par contre étant donné qu'il s'agit d'une méthode permettant de résumer les données, il est donc évident que des informations seront perdues afin de gagner en lisibilité. Il faut donc prendre des précautions sur l'utilisation et l'analyse qui nécessite d'être interprétée et critiquée.

II.2.2 Méthode AFD (Analyse Factorielle Discriminante)

L'Analyse Factorielle Discriminante (AFD) ou encore appelée Analyse Linéaire Discriminante (ALD) a été développée par Fisher en 1936. Cette méthode est une variante de la méthode ACP où le principe est de regrouper ou de classer un ensemble de données en un certain nombre de paquets séparés d'individus selon une dépendance prédéfinie à l'avance. Cette méthode garde tous les avantages de la méthode ACP soit :
- Le résumé des données initiales afin de les représenter sur deux ou trois axes principaux pour une analyse explicative.
- La possibilité d'établir des règles décisionnelles.

La méthode AFD est à la fois descriptive et prédictive. Dans un premier temps de façon descriptif, nous essayons de chercher les n composantes principales représentant au mieux les données originales (échantillon d'apprentissage) en fonction de q classes. Dans un second temps, d'un point de vue prédictif, nous cherchons à représenter des n' nouveaux individus exprimés par les mêmes variables explicatives et à connaître la classe d'affectation de ces nouvelles données.

La *figure 2.7* représente le principe :

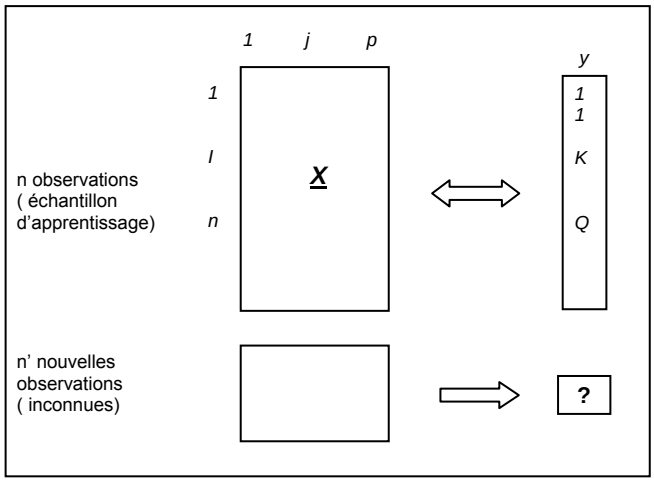


Figure 2.7 : schéma descriptif de la méthode AFD.

Nous allons maintenant aborder le principe mathématique de la méthode AFD se décomposant en 2 étapes.

a) <u>Etape 1 : Analyse factorielle discriminante d'ordre descriptif</u>

La première phase est effectuée sur un échantillon d'apprentissage expliqué par les *p* variables et affecté en *q* classes. La disposition des données originales dans la matrice *X* est identique à la méthode ACP. Nous faisons ensuite un centrage et une réduction de cette matrice *X* afin d'obtenir une nouvelle matrice X_{cr} négligeant les problèmes de données mesurées en unité différente et ne privilégiant pas de variables particulières. (Le calcul mathématique est exprimé dans le paragraphe II.2.1).

Comme dans l 'analyse ACP, nous allons maintenant calculer la matrice de variance totale *V* (encore appelé matrice d'inertie ou bien encore matrice de corrélation). De cette matrice, nous allons chercher les nouveaux axes de symétrie. La relation calculant cette matrice est exprimée dans le paragraphe II.2.1.b).

La matrice de variance totale *V* peut se décomposer de la manière suivante :

$$V = W + B = cste \tag{2.7}$$

Elle se décompose en une somme de 2 matrices, une matrice de variance intra-classe *W* (Within) et une matrice de variance inter-classe *B* (Between). La matrice de variance intra-classe représente la dispersion des individus appartenant à une même classe autour de leur centre de gravité. La matrice de variance inter-classe est liée à la dispersion des centres de gravité des classes autour de l'origine.

Il faut maintenant trouver l'axe factoriel qui discrimine au mieux l'ensemble des classes d'individus. Pour cela, nous cherchons à maximaliser la variance inter-classe et à minimaliser la variance intra-classe. Cette double contrainte peut être unifiée en maximalisant le rapport entre la variance intra-classe et la variance totale. La relation est la suivante :

$$\max\left(\frac{\boldsymbol{B}(\boldsymbol{a},\boldsymbol{a})}{\boldsymbol{V}(\boldsymbol{a},\boldsymbol{a})}\right) \tag{2.8}$$

avec $a = V^{-1}.u$

u : vecteur unitaire

Afin de réaliser cette condition, nous allons utiliser les propriétés spectrales de la matrice de variance totale et la solution a ($a = V^{-1}.u$) qui est donnée par les vecteurs propres de la matrice de variance totale associé à des valeurs propres respectives λ. Ces valeurs propres correspondent au pouvoir discriminant de chaque axe de symétrie et sont exprimée en pourcentage.
Afin de trouver la nouvelle matrice dans le nouveau plan factoriel, il suffit simplement, comme dans le cas de la méthode ACP, de multiplier la matrice originale (la matrice centrée-réduite) par la matrice des vecteurs propres.

Enfin, nous pouvons représenter le diagramme de dispersion des individus afin de visualiser les groupes d'individus pour différentes classes d'individus (*figure 2.8*).

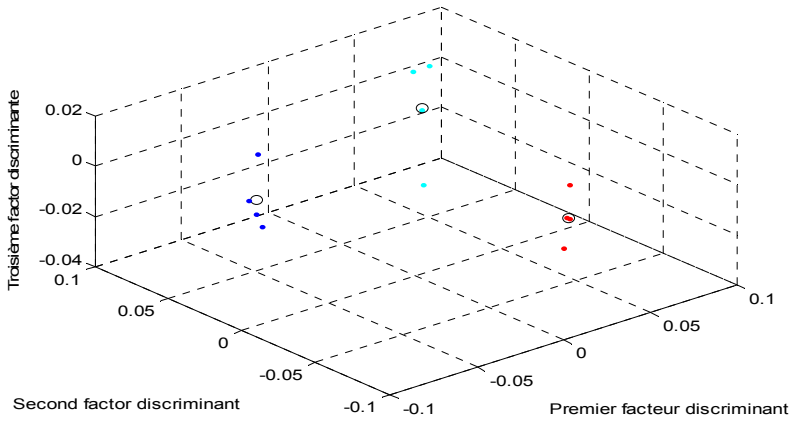


Figure 2.8 *:* Représentation sur trois axes de la méthode AFD.

b) <u>Etape 2 : Analyse factorielle discriminante d'ordre décisionnelle</u>

La seconde partie de la méthode AFD est d'ordre décisionnelle. Il faut maintenant introduire dans ce diagramme de dispersion de nouvelles observations inconnues et de les affecter à une classe d'individus. Afin de réaliser cette opération, il suffit de multiplier la nouvelle matrice par la matrice des vecteurs propres afin de les représenter dans le plan factoriel.

Afin d'affecter un nouvel individu à une classe où à un groupe défini dans la phase de l'apprentissage, nous pouvons citer deux méthodes de classification :
- La méthode de la distance de « Mahalanobis ».
- La méthode de « bayésien ».

La première méthode est basée sur la distance de « Mahalanobis ». Cette méthode est donc purement géométrique. Afin de l'appliquer, nous calculons le centre de gravité des groupes d'individus de la phase d'apprentissage (de chaque classe) et nous affectons le nouvel individu où la distance entre le centre de gravité et l'individu est la plus petite.

La seconde méthode est basée sur les probabilités et plus particulièrement sur le modèle « bayésien » d'affectation pour lequel nous calculons les probabilités d'appartenance à chacun des groupes. Puis nous affectons le nouvel individu au groupe où la probabilité est maximale.

En conclusion, dans le cadre de notre projet nous utiliserons dans un premier temps la méthode AFD plus adapté à la discrimination des différents gaz ou différentes concentrations de gaz.

III État de l'art sur les circuits électroniques associés au traitement des données

Dans ce second chapitre, nous nous sommes intéressés au circuit électronique associé au traitement des données transmises par le capteur de gaz SnO_2. Nous avons tout d'abord étudié les principales architectures de circuits dédiés à notre application. Puis de ces analyses, nous avons pu choisir une première architecture pour réaliser notre circuit de traitement du signal.

III.1 Architecture ASIC

Un circuit ASIC (*Application Specific Integrated Circuit*) a une grande souplesse de conception puisque celui-ci permet d'optimiser :
- La puissance.
- La fréquence.
- L'architecture et la taille de la puce.

Un autre avantage majeur du circuit ASIC est la possibilité d'implanter des circuits analogiques comme des amplificateurs, des convertisseurs, ….

Néanmoins, son temps de développement important nécessite de forts coûts. De plus, pour réaliser l'implantation physique du circuit, nous devons passer par un fondeur.

III.2 Architecture FPGA

Les circuits FPGA (*Field Programmable Gate Array*) se décomposent en 2 familles :
- Les FPGA de type RAM qui sont reprogrammables.
- Les FPGA de type anti-fusible (PROM) qui sont non reprogrammables.

Un circuit FPGA est constitué (*Figure 3.1*) [4]:
- De plusieurs cellules incorporant un bloc logique simple.
- De switchs qui réalisent les interconnexions entre les cellules.
- De blocs d'entrées / sorties.

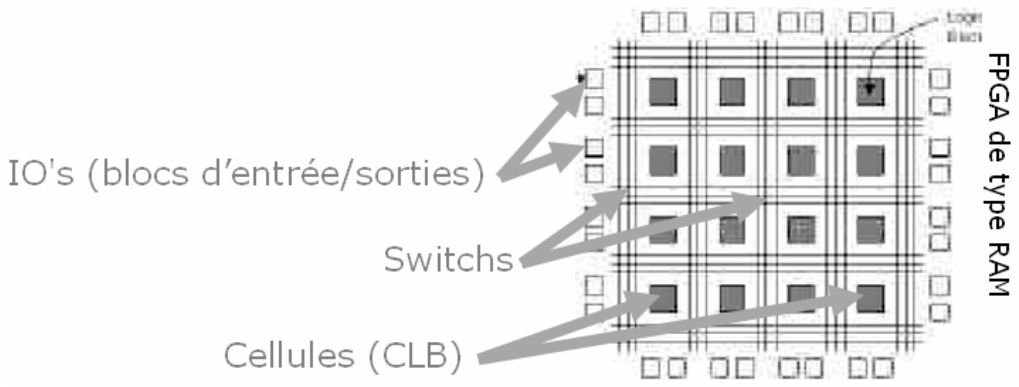


Figure 3.1 : Schéma fonctionnel d'un circuit FPGA de type RAM.

III.2.1 Blocs logiques configurables (CLB) : Cellules

Les cellules constituant un circuit FPGA sont principalement composées soient :

- De portes logiques configurables pour réaliser des fonctions logiques simples.
- De bascules pour réaliser des mémoires aléatoires d'un bit.

Pour implanter des fonctions combinatoires, ces fonctions logiques simples sont associées à différents composants : latchs, multiplexeurs, mémoires à faibles dimensions….

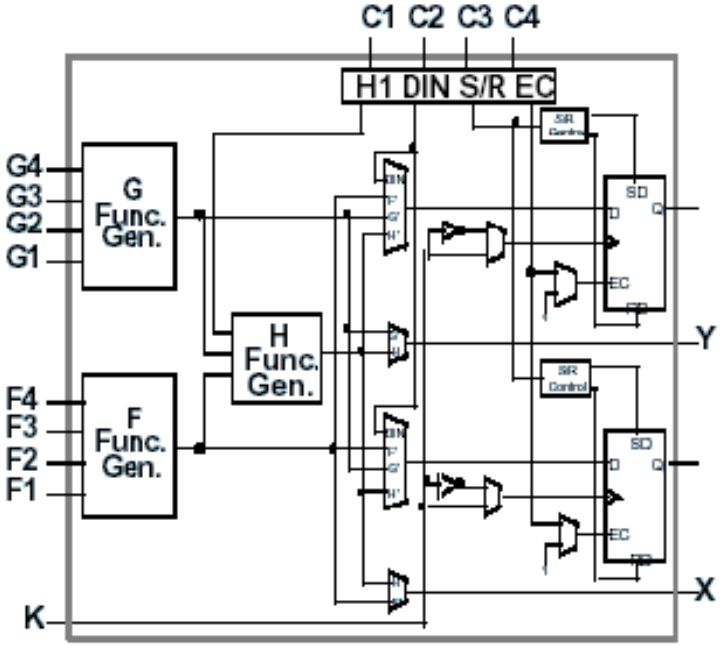


Figure 3.2 : Schéma fonctionnel d'une cellule (CLB).

III.2.2 Blocs d'entrées/sorties programmables

Les blocs d'entrées / sorties servent d'interface entre les pins (plots) du circuit et le cœur du FPGA (via le routage programmable). Il existe différents types d'entrées/sorties :
- Entrées/sorties adaptées suivant les signaux.
- Alimentations des cellules, des plots, … .
- Signaux d'horloges.
- Signaux de configuration: programmation du FPGA.
- Signaux de test.

III.2.3 Programmation du circuit FPGA

Une première étape dans la programmation d'un circuit FPGA consiste à connecter les blocs logiques entre eux et les entrées/sorties (routage du programme). Le routage d'un programme est présenté *figure 3.3*.

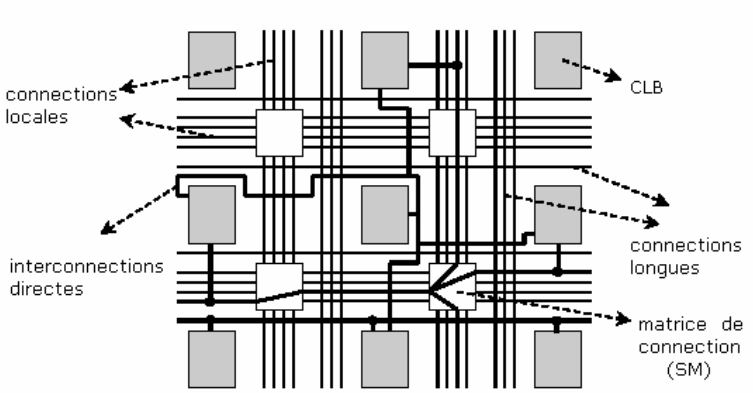


Figure 3.3 : Interconnections entre cellules (CLB)

Les circuits FPGA disposent de plusieurs ressources de routage. En général, les niveaux hiérarchiques disponibles sont :
- Connexions directes vers les voisins proches.
- Connexions générales à travers des matrices de routages et des canaux disposés suivant une topologie simple (grille, tore).
- Connexions à longue distance (avec driver pour haute sortance).
- Distribution d'horloge spécifique (sur des canaux dédiés).

Ensuite, la seconde étape de programmation des circuits FPGA est la définition des connexions entre les switchs et les blocs logiques. Cet outil de design traduit le langage introduit par l'utilisateur (VHCL, Handel-C, Verilog), puis réalise la validation (*figure 3.4*).

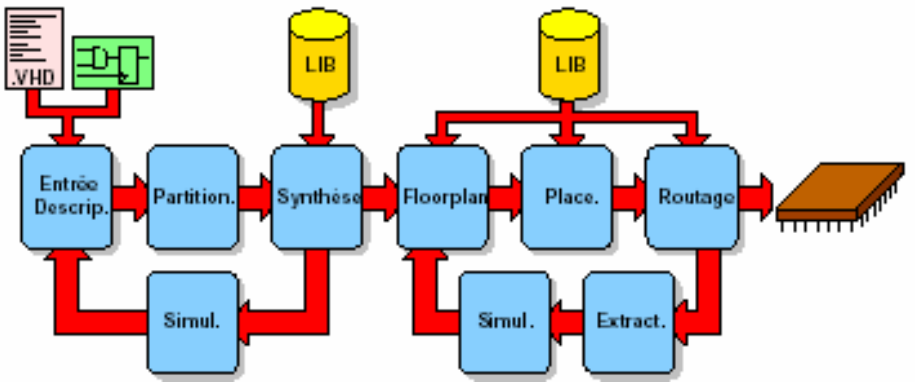


Figure 3.4 : Etapes de l'outil de design : du programme au routage.

III.2.4 Conclusions

Concernant notre application, nous pouvons constater que ces circuits FPGA sont adaptés pour réaliser des prototypes. En effet, ils permettent :
- De « designer » des systèmes complexes comme les microprocesseurs (capacité entre 5000 à 10 millions de portes).
- De pouvoir effectuer des changements de requêtes dans les étapes avancées du design.
- De re-programmer sous une forme dynamique rapide.

Néanmoins, Les FPGA présente des inconvénients qui sont :
- L'obligation de rajouter des composants extérieurs analogiques (CAN, …). Les FPGA ne contiennent que des cellules logiques.
- Un temps de développement important, car nous devons "programmer" des systèmes de mémoires.

III.3 Architecture DSP

Le circuit DSP (*Digital Signal Processors*) est la solution effective pour la plupart des applications de traitement du signal. En effet, les caractéristiques de ce circuit sont :
- La réalisation des opérations de multiplication et d'accumulation (MAC) en un seul cycle d'horloge.
- Une architecture qui supporte un flux d'information bidirectionnel à grande vitesse allant de l'unité de calcul à la mémoire (*Figure 3.5*).
- Une architecture harvard qui incorpore plusieurs bus et mémoires. Cette architecture permet alors une lecture ou une écriture de plusieurs cases mémoires en même temps. L'architecture Harvard classique utilisent deux mémoires, une pour les données et l'autre pour les instructions (*Figure 3.5*).

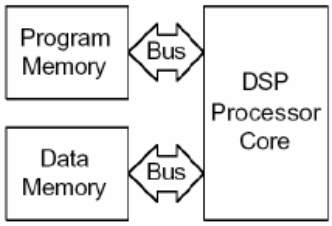


Figure 3.5 : Illustration architecture harvard

- L'utilisation d'unités DMA (Direct Memory Acess) et de générateurs bidirectionnels (DAG) qui travaillent en parallèle avec les autres éléments du circuit. Les générateurs DGA réalisent des calculs de directions qui permettent d'aller chercher deux données différentes en un seul cycle d'horloge.

- L'utilisation de plusieurs unités de calcul permettant de réaliser plusieurs opérations en parallèle. Ces unités de calcul réalisent les opérations de :
  - Arithmétique et logique ALU
  - Multiplication-addition avec ou sans accumulation interne (MAC)
  - Décalage.
  - Spécifiques (manipulation de bits (BMU) ; comptage de bits).

Par exemple, la *figure 3.6* représente une architecture classique d'un circuit DSP qui est constitué de :
- Trois unités de calcul travaillant en parallèle, l'unité MULT qui réalise les fonctions de multiplications et d'accumulations, l'unité BARREL SHIFTER qui réalise les fonctions spécifiques pour les bits et l'unité ALU qui réalise les opérations classiques.
- Deux unités de génération de directions duales, DAG1, DAG2.
- Des unités de mémoire.
- Plusieurs bus pour incrémenter les taux de transfert de la mémoire et éviter les conflits de directions, PMA, PMD, DMA, DMD.
- Deux unités DMA pour les adresses et les données.

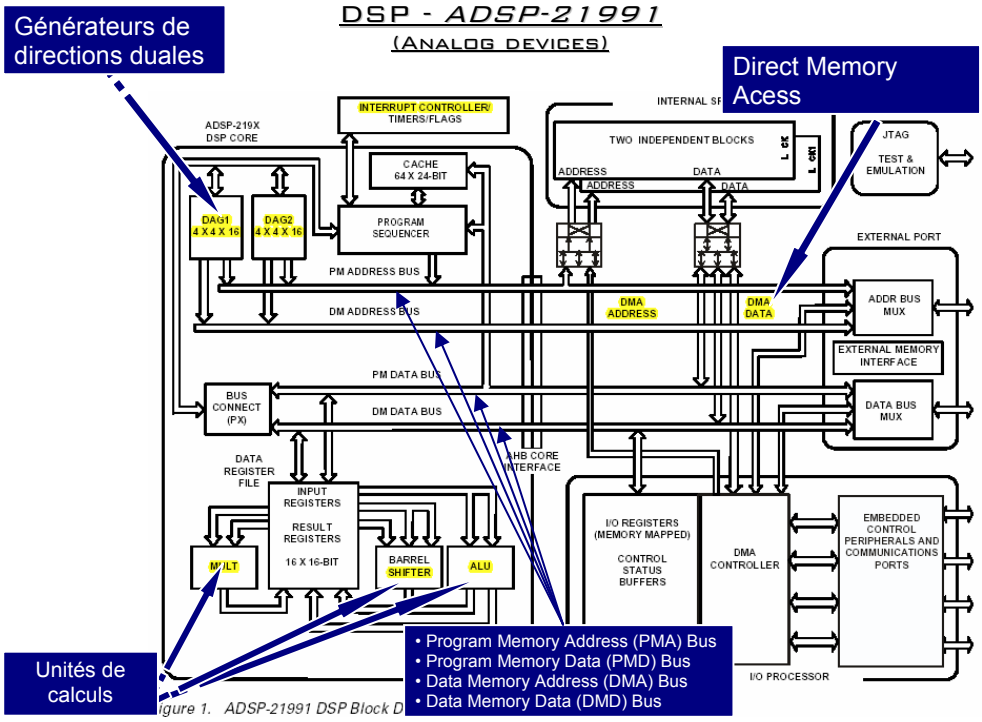


Figure 3.6 : Architecture harvard d'un DSP, avec les différents unités

III.3.1 Deux familles de circuit DSP

Les circuits DSP se regroupent en deux familles, les processeurs à virgule fixe et les processeurs à virgule flottante (*figure 3.7*) [5].

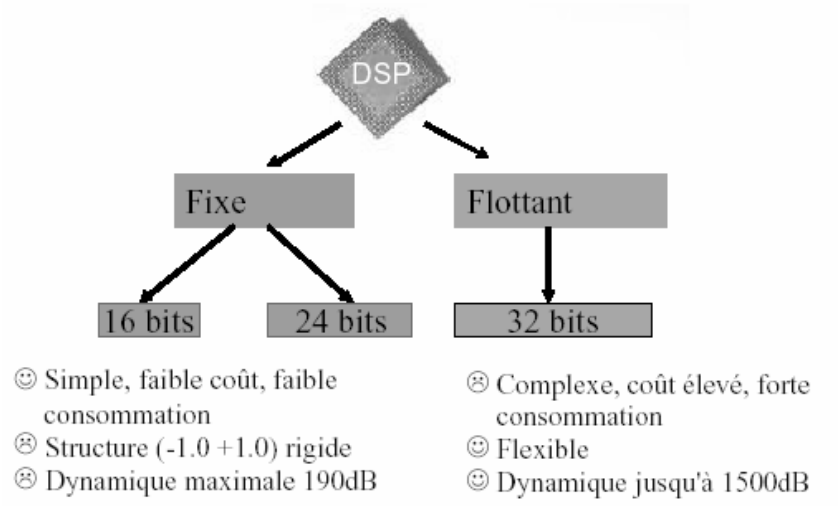


Figure 3.7 : Comparaison entre les deux familles de DSP.

a) Famille des circuits DSP à virgule fixe

La plupart des circuits DSP sont à virgule fixe car ceux-ci présentent une architecture plus simple donc un faible coût, une faible consommation et une plus grande rapidité.

Ces circuits DSP contiennent généralement des mots de 16 ou 24 bits. Néanmoins, il existe aussi des mots de 20 ou 32 bits. La *figure 3.8* montre la codification en virgule fixe, les 16 bits correspondent à des fractions en puissance négative de 2.

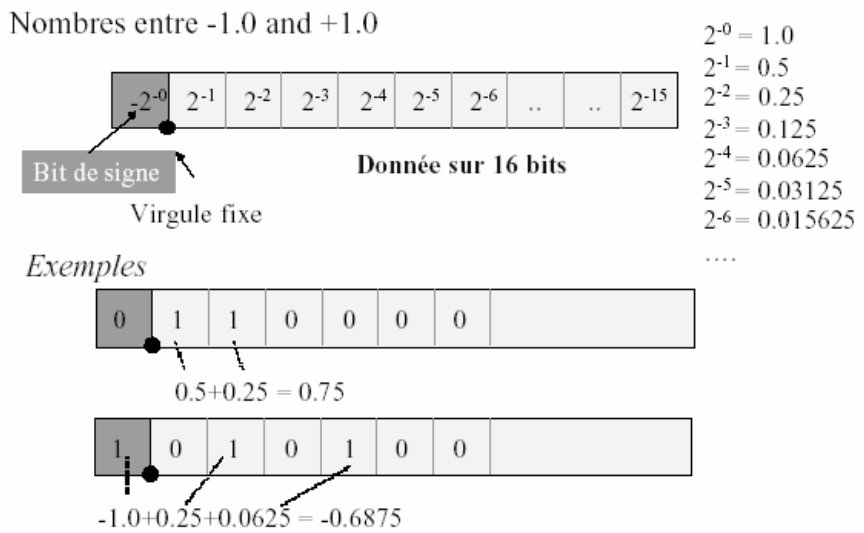


Figure 3.8 : Illustration des données sur 16 bits codées en virgule fixe.

b) Famille des circuits DSP à virgule flottante

Les processeurs à virgule flottante utilisent des mots de 16, 24 ou 32 bits [6]. Dans ce cas, les données sont codées sous la forme d'une mantisse, multipliée par un exposant (*figure 3.9*). Les processeurs à virgule flottante ne sont utilisés que dans les applications à grande puissance de calcul, et pour traiter des signaux de grande dynamique.

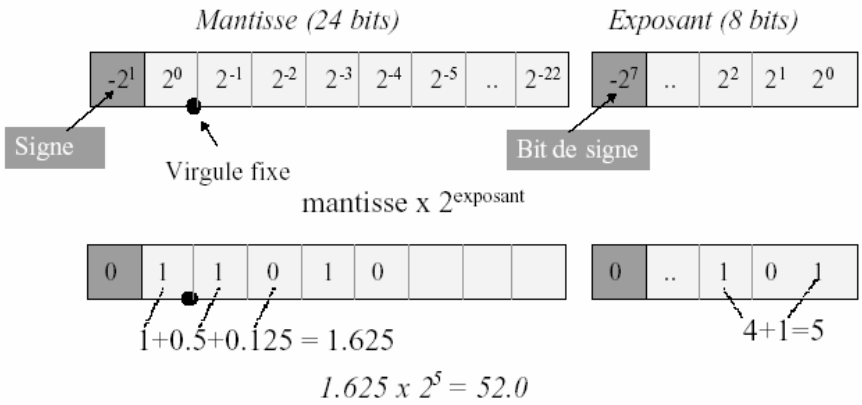


Figure 3.9 : Illustration des données codées en virgule flottante

III.3.2 Programmation des circuits DSP

La *figure 3.10* montre les différentes étapes de programmation des circuits DSP. Le programme du circuit peut être réalisé en langage C puis traduit en langage assembleur par un compilateur spécifique « DSP C code ». la fonction « Linker » va chercher les fonctions spécifiques utilisées dans le code assembleur puis la compilation du code est réalisée par la fonction « Debugger » pour chercher les erreurs éventuelles

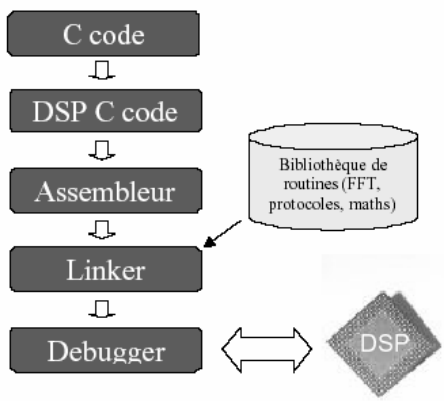


Figure 3.10 : Programmation du DSP

III.3.3 Caractéristiques des circuits DSP

Un circuit DSP sera choisi suivant :
- Famille : virgule fixe
- Bits : 16
- Puissance en MIPS (Millions d'instructions par secondes)
- La quantité de mémoire interne (DRAM/RAM/ROM/Flash...)
- Ses entrées/sorties (ports série, ports parallèles) et leurs vitesses respectives
- Son architecture interne, avec la présence ou non de canaux DMA (Direct Memory Access). L'architecture jouera en effet sur le degré de parallélisme, la gestion des ruptures de pipe-line, la facilité du multi-processing...

Les DSP sont des dispositifs très puissants pour le traitement de signal car ils peuvent traiter plusieurs instructions en parallèle suivant l'utilisation de plusieurs unités de calculs.

III.4 Microcontrôleurs

Les microcontrôleurs sont des circuits très utilisés pour réaliser le traitement des données des nez électroniques. Ces circuits incorporent des systèmes mémoires de type RAM et ROM, des convertisseurs (CAN, CNA), des interruptions, des timers [8]. Suivant ces caractéristiques, le microcontrôleur sera le circuit utilisé pour réaliser notre prototype de traitement des données.

Le *tableau 3.1* montre une comparaison entre les microcontrôleurs et les DSP.

Microcontrôleur	DSP
Pas d'application spécifique	Traitement numérique du signal
Mémoire à accès simple	Mémoire à accès multiple
Modes d'adressage généraux	Modes d'adressages particuliers
Mémoire externe très grande	Mémoire externe faible
Interfaces généralistes	Interfaces généralistes
Architecture Von Neumann (*figure 3.12*) ou Architecture Harvard Il faut faire attention pour ne pas écrire au dessus de la mémoire du programme	Architecture Harvard (typique)

Tableau 3.1 : Tableau comparatif entre les circuits DSP et microcontrôleurs.

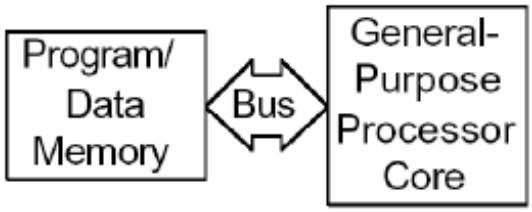


Figure 3.11 : Illustration architecture Von Neumann

Remarque : Les requêtes de contrôle en temps réel imposent à chaque fois des conditions plus exigeantes en ce qui concerne le calcul et l'acquisition des données. De ce fait, les fabricants de circuits apportent :
- Au microcontrôleur (microchip, ST, etc.), des caractéristiques de DSP (unités de calcul parallèles, pipeling, etc.)
- Au DSP (Texas, Motorola, Analog Device, etc.), des caractéristiques de microcontrôleur (A/D, ports digitales I/O, bloques PWM).

III.4.1 Structures des microcontrôleurs

Il existe trois principales architectures de microcontrôleurs, l'architecture CISC (traitement séquentiel), l'architecture RISC (traitement segmenté) et l'architecture SISC (traitement parallèle). Les deux premières architectures sont les plus utilisées, l'architecture SISC est la plus récente et utilise un répertoire d'instructions plus spécifique pour des applications précises et complexes.

Les caractéristiques de ces 3 architectures sont :

- Architecture CISC (Complex Instruction Set Computer) :
  Une instruction peut nécessiter plusieurs opérations et donc utiliser plusieurs cycles d'horloges pour son exécution.
  Plus de 80 instructions machine.
  Architecture Von-Newmann.

- Architecture RISC (Reduced Instruction Set Computer) :
  Le répertoire d'instructions est réduit et elles s'exécutent généralement en un cycle d'horloge.
  Exécution segmentée (Pipepiling) des instructions.
  Architecture Harvard.

- Architecture SISC (Specific Instruction Set Computer)
  Le répertoire d'instructions est réduit, spécifique et adaptable selon les applications.

III.4.2 Traitement des instructions

Comme pour l'architecture des microcontrôleurs, il existe plusieurs types de traitement des instructions :

- Le traitement séquentiel qui est le plus simple et consiste à exécuter les différentes instructions les unes après les autres.

- Le traitement segmenté (pipeline) réalise à la fois l'exécution d'une instruction et cherche l'instruction suivante. Les étapes successives du traitement segmenté sont :
  Fetch : Ce pas va chercher l'instruction dans la mémoire du registre d'instruction puis va chercher l'instruction suivante.

  Decode : Ce pas va décoder l'instruction, c'est-à-dire il calcule l'adresse des données A et B (si l'instruction en a besoin) et va extraite la valeur de celles-ci de la mémoire.

  Execute : Ce pas réalise l'instruction.

  WriteBack : Ce pas est réalisé pour écrire le résultat de l'instruction dans les registres si celui-ci provient de la mémoire ou de l'ALU.

- Le traitement parallélisme est assimilé à la multiplication de ressources, c'est à dire l'exécution en même temps de la même instruction mais sur plusieurs données différentes.

Pour mieux comprendre le principe de ces trois traitements d'instructions (séquentiel, pipeline et parallélisme), la *figure 3.12.b* illustre un exemple où deux instructions comprenant trois opérations sont exécutées (*figure 3.12.a*) [7].

Figure 3.12.a : Modélisation des 2 instructions identiques qui réalisent 3 opérations.

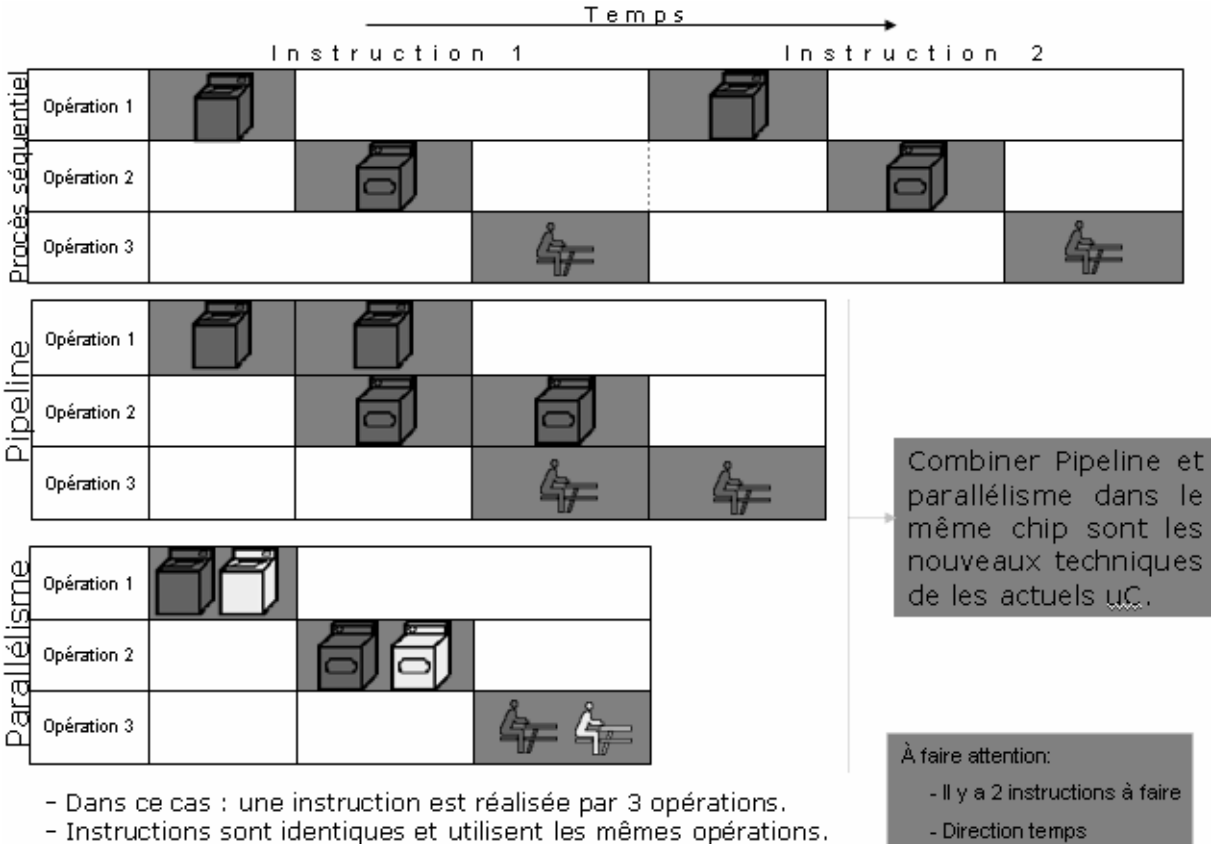


Figure 3.12.b : Les 3 types de traitements des données réalisant les 2 instructions.

Le traitement séquentiel réalise bien les 3 opérations de l'instruction les unes après les autres. Le traitement pipeline exécute bien 2 opérations différentes des 2 instructions. Pour le traitement parallélisme, celui-ci exécute bien les 2 opérations identiques des 2 instructions.
Ainsi dans le cas du traitement parallélisme, nous pouvons remarquer que celui-ci est 2 fois plus rapide que le traitement séquentiel.

Remarque : Un nouveau traitement des instructions combine les traitements pipeline et parallélisme améliorant la vitesse d'exécution des instructions. Par ce nouveau traitement, 4 instructions sont exécutées contre une pour le traitement séquentiel (*Figure 3.13*).

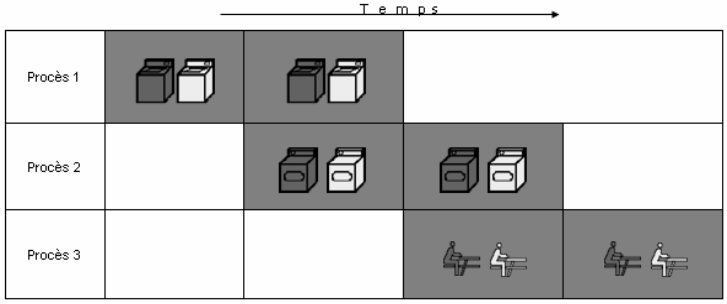


Figure 3.13 : Traitement des instructions combinant les traitements pipeline et parallélisme.

III.5 Conclusions

Suite à cette étude des différentes architectures (ASIC, FPGA, DSP, Microcontrôleur) et en ayant analyser les avantages et inconvénients de chacune d'elles, nous avons choisi d'utiliser comme architectures :
- Un microcontrôleur.
- Ou un microcontrôleur associé avec un circuit DSP.

Néanmoins, sachant qu'aucune méthode mathématique de discrimination et d'extraction de données sera implantée (seul les résultats de la méthode seront utilisés), nous utiliserons une architecture composée principalement d'un microcontrôleur.

Suivant les particularités des microcontrôleurs énumérées dans cette partie, nous avons choisi d'utiliser un microcontrôleur PIC, fabricant Microchip. Ce type de microcontrôleur se base sur une architecture RISC utilisant donc le traitement pipeline.

Ainsi, le schéma synoptique final est représenté *figure 3.14*.

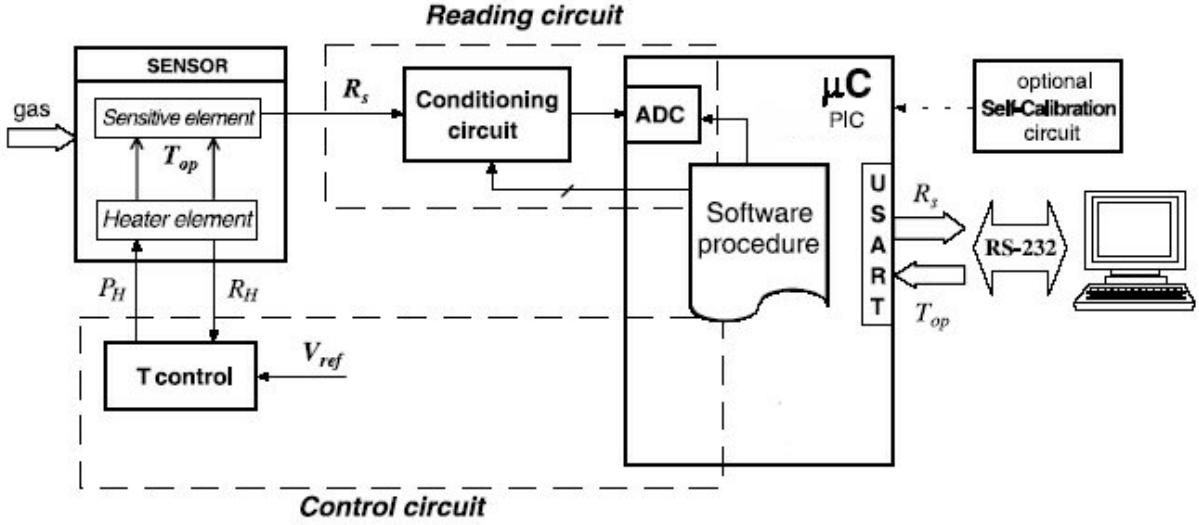


Figure 3.14 : Schéma synoptique de notre circuit portatif (Capteur SnO2; Contrôle de la température du heater ; Microcontrôleur PIC).

Le prochain chapitre concerne la programmation d'un arbre de décision permettant de discriminer l'air sec (AS) et 2 gaz, le monoxyde de carbone (CO) et l'éthylène (C_2H_4). À partir de ce programme, nous déterminerons le nombre d'instructions et de cycles d'horloges. De ces 2 caractéristiques, nous pourrons en déduire la taille de la zone mémoire ainsi que la fréquence du quartz associé au microcontrôleur.

IV Traitement des données dans l'air sec pour la discrimination de 2 gaz

Le programme que nous avons réalisé permet de discriminer l'air sec (AS) et 2 gaz, le monoxyde de carbone (CO) et l'éthylène (C_2H_4). Pour cela, nous utilisons 2 matrices d'entrées, l'une pour les données pré-traitées issues de deux capteurs (variables) et l'autre pour les vecteurs propres issus d'une analyse AFD.

IV.1 Résultats de la méthode AFD

Pour réaliser notre programme une méthode AFD est utilisée. À partir de variables, nous obtenons:

- La matrice des vecteurs propres.
- Les droites D1 et D2 pour notre arbre de décision.

IV.1.1 Variables

La *figure 4.1* représente le profil de la tension envoyé aux bornes du heater de 4 capteurs. Cette opération est réalisée 3 fois :

- Sous air sec.
- Sous CO (monoxyde de carbone).
- Sous C_2H_4 (éthylène).

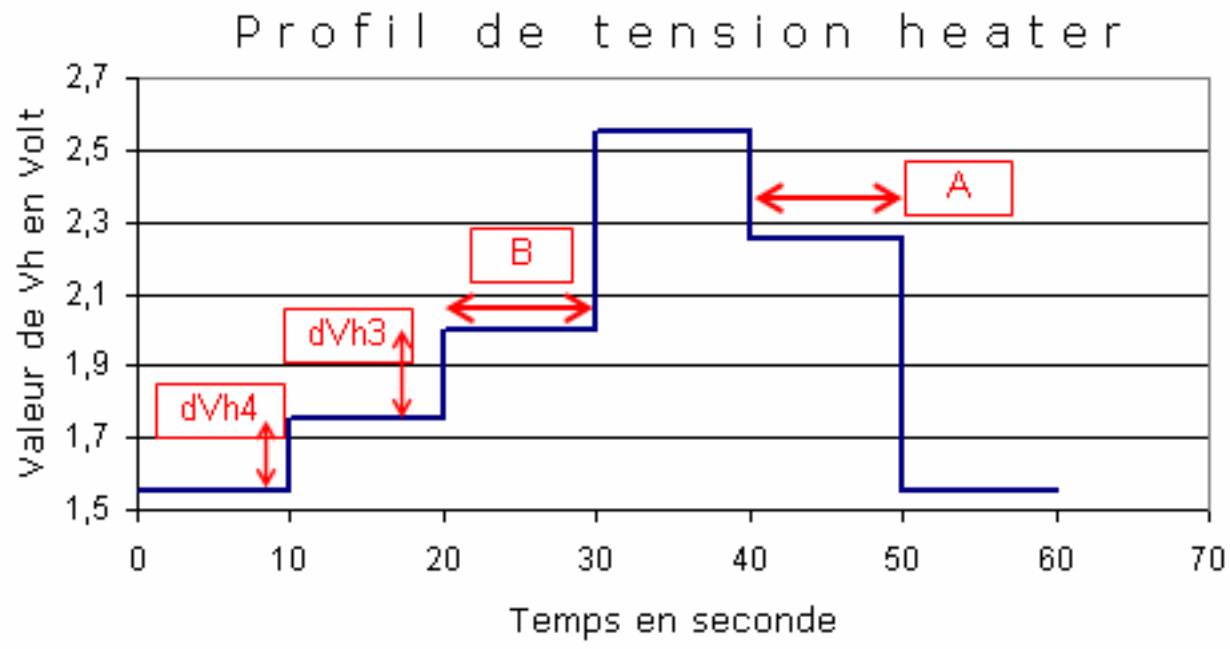


Figure 4.1 : Profil d'obtention des variables pour chaque capteur

À partir de cette application, nous avons choisi 4 variations de la résistance des couches sensibles que nous représentons sous forme de 4 variables (Ces variables sont illustrées par dVh4, dVh3, A, B). Ainsi la matrice des variables obtenue est représentée *figure 4.2*.

dVh3	dVh4	A	B		
0,1310	0,1416	0,4742	0,5912	capteur 1	sous AS
0,0899	0,1419	0,4739	0,5773	capteur 2	
0,0801	0,1512	0,5017	0,6117	capteur 3	
0,0548	0,1065	0,6075	0,7626	capteur 4	
0,2097	0,2517	0,7186	0,9332	capteur 1	sous C_2H_4
0,1462	0,2707	0,7863	1,0293	capteur 2	
0,1498	0,2635	0,8688	1,1332	capteur 3	
0,1270	0,1538	1,1731	1,4412	capteur 4	
0,3118	0,1392	0,5479	0,7056	capteur 1	sous CO
0,2294	0,1294	0,6092	0,7716	capteur 2	
0,2390	0,1178	0,6654	0,8438	capteur 3	
0,2060	0,0599	0,8220	1,0716	capteur 4	

Figure 4.2 : Variables dVh4, dVh3, A, B représentant la variation de la résistance des couches sensibles de 4 capteurs sous différents types de gaz (AS, CO, C_2H_4)

IV.1.2 Matrice des vecteurs propres

De la matrice des variables (*figure 4.2*), nous utilisons la méthode AFD pour obtenir la matrice des vecteurs propres (*figure 4.3*).

0,4041	0,5439	0,0781	-0,4349
0,8896	-0,5463	0,0437	-0,8754
0,1148	-0,5002	0,7721	0,1884
0,1792	0,3944	-0,6292	0,0947

Figure 4.3 : Matrice des vecteurs propres issues d'un analyse AFD sur la matrice de la *figure 4.2*

IV.1.3 Droites D1 et D2 pour notre arbre de décision

Nous réalisons la multiplication de la première matrice (*figure 4.2*) représentant les variables connues de la variation de la résistance des couches sensibles avec la matrice des vecteurs propres (*figure 4.3*). La matrice résultante est représentée *figure 4.4*.

X	Y			
0,3393	-0,0101	0,0106	-0,0356	sous AS
0,3204	-0,038	0,0158	-0,0193	
0,3341	-0,0487	0,0153	-0,0147	
0,3233	-0,0315	-0,0018	0,0697	
0,5584	-0,0149	-0,0049	-0,0878	sous C_2H_4
0,5747	-0,0557	-0,0172	-0,0549	
0,5978	-0,0502	-0,0190	-0,0248	
0,5811	-0,0334	0,0156	0,1676	
0,4392	0,0977	0,0095	-0,0874	sous CO
0,4160	0,0537	0,0085	-0,0252	
0,4290	0,0656	0,0067	-0,0018	
0,4229	0,0908	-0,0208	0,1144	

Figure 4.4 : Matrice résultante de la multiplication de la matrice des variables connues (*figure 4.2*) et de la matrice des vecteurs propres (*figure 4.3*)

Les 2 premières colonnes de la matrice résultante représentent respectivement les coordonnes X et Y des points représentés *figure 4.5*.

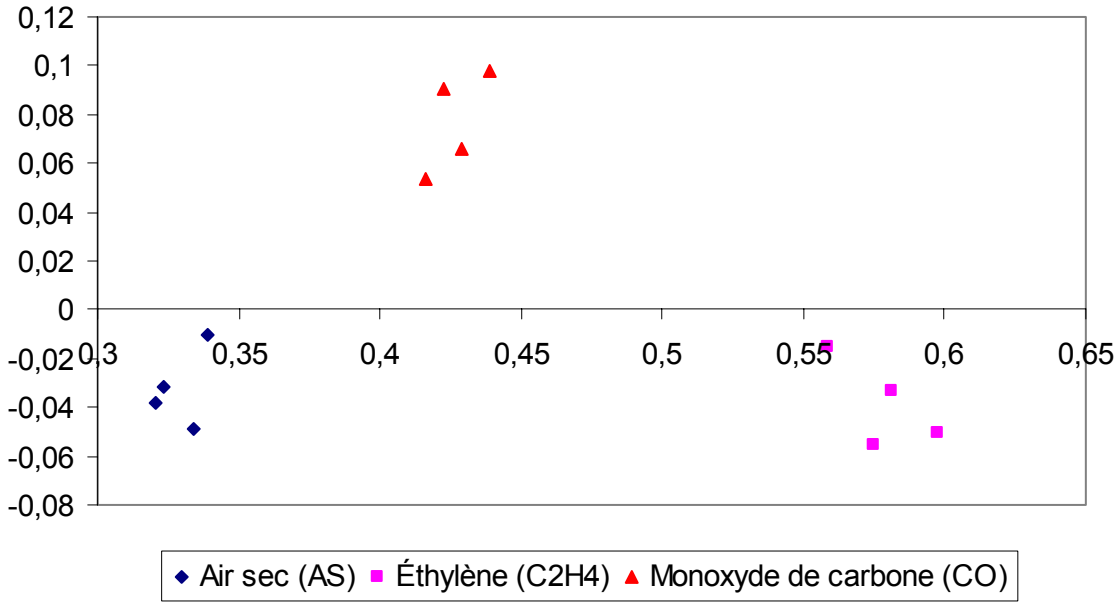


Figure 4.5 : Nuage de points de la matrice résultante (*figure 4.4*)

De la *figure 4.5* nous observons bien une séparation des points correspondant à la variation de la résistance des 4 couches sensibles sous l'air sec et les 2 gaz (◆AS; ▲CO; ■C_2H_4). De ces nuages de points, nous calculons leur centre d'inertie pour en déduire l'équation des droites D1 et D2 (*figure 4.6*). Ces 2 droites nous serviront pour discriminer les gaz à partir de notre arbre de décision.

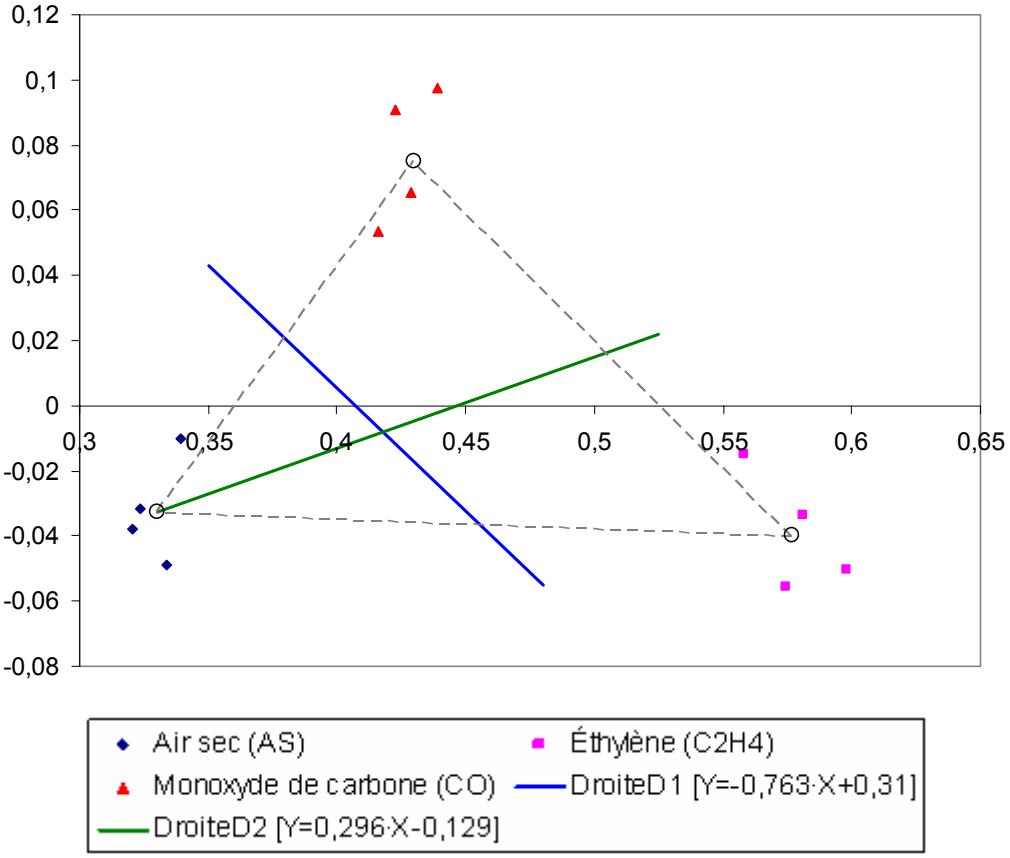


Figure 4.6 : Droites D1 et D2

IV.2 Algorithme du programme

IV.2.1 Données

A partir du même profil de tension envoyé aux bornes du heater de 2 autres capteurs de gaz SnO_2 (*figure 4.1*), les variables obtenues (variation de la résistance des 2 couches sensibles) sont représentées sous la forme d'une matrice appelée matrice d'entrée. Une autre matrice est utilisée pour représenter les vecteurs propres de la méthode AFD déterminés auparavant (IV.1.2).

La multiplication de ces deux matrices est réalisée et les résultats obtenus représentent donc les coordonnées en abscisses et en ordonnées des 2 points qui vont être introduits dans l'arbre de décision.

La figure 4.7 représente les 2 matrices, d'entrée et des vecteurs propres, et la matrice de résultat de la multiplication.

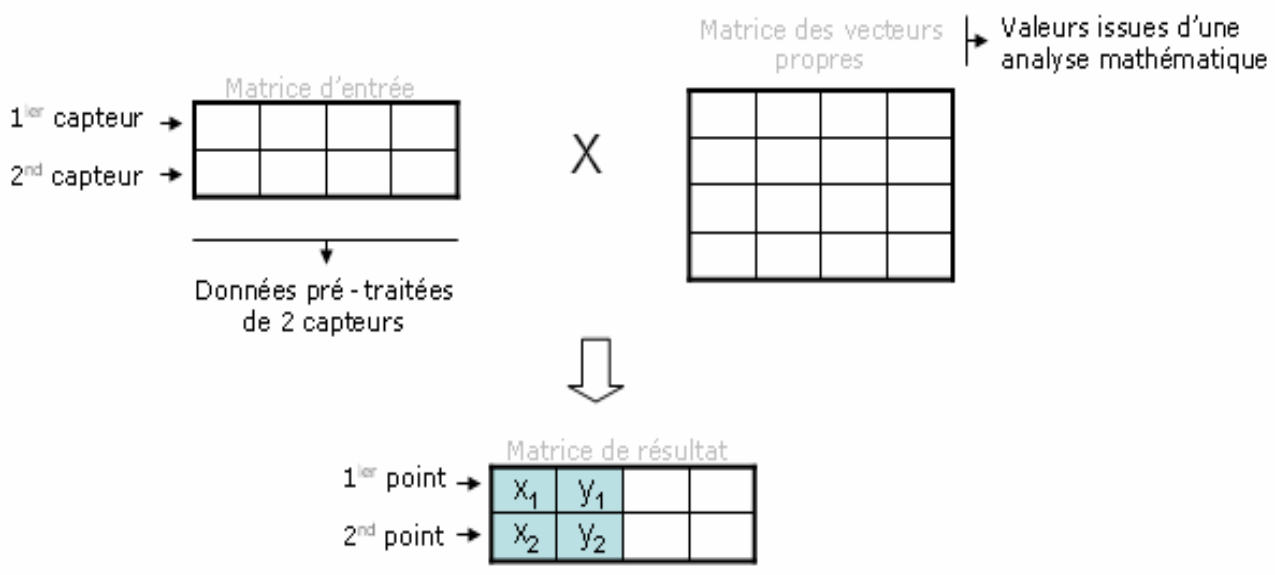


Figure 4.7 : Illustration des 3 matrices.

Remarque : Pour travailler avec ces données, nous ne pouvons pas déclarer de matrices multidimensionnelles dans le compilateur C *(section IV.3)*. Nous allons de ce fait utiliser des tableaux à une dimension (*figure 4.8*).

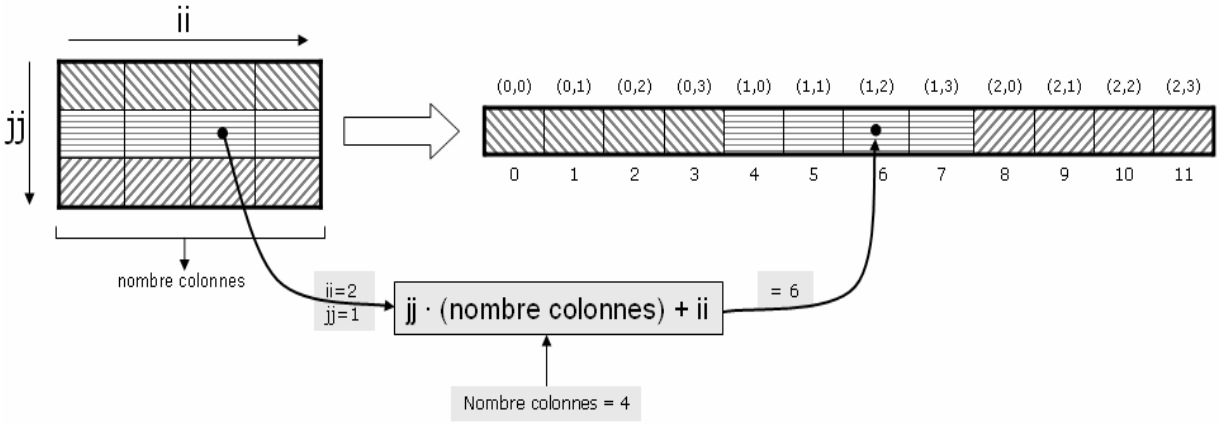


Figure 4.8 : Conversion d'une matrice à un vecteur

Ces vecteurs sont déclarés de type « float » car les données sont des décimaux à 4 chiffres après la virgule. Chaque élément (donnée) de ce vecteur nécessite 6 instructions en assembleur pour charger la valeur dans la mémoire. De plus la valeur de chaque donnée se compose de 3 octets, 3 positions de mémoires (*figure 4.9*)

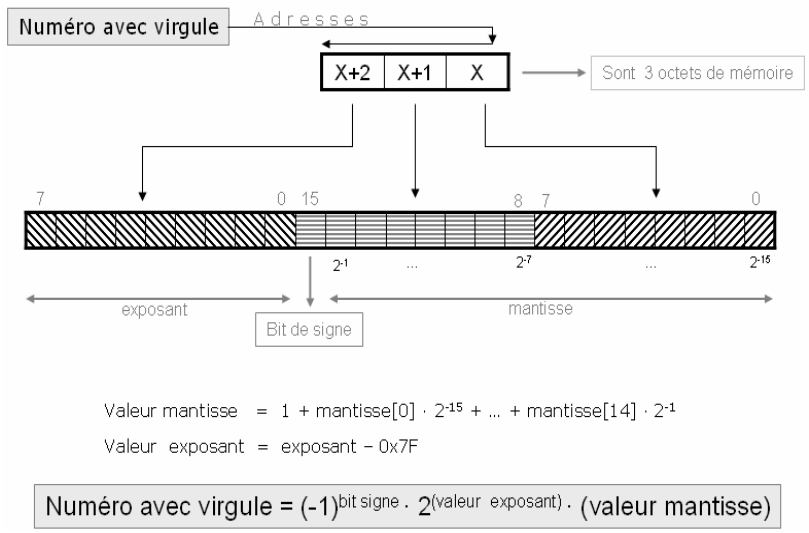


Figure 4.9 : Modélisation des décimaux à 4 chiffres après la virgule en langage assembleur.

IV.2.2 Arbre de décision

A partir du point donné (x, y) de la matrice de résultat, l'arbre de décision va nous servir à discriminer le gaz qui est en contact avec la couche sensible. Pour cela, nous nous servons des équations des 2 droites (DroiteD1 ; DroiteD2) obtenues dans le paragraphe IV.1.3 pour discriminer l'air sec AS et les 2 gaz, monoxyde de carbone CO et éthylène C_2H_4 (*figure 4.10*).

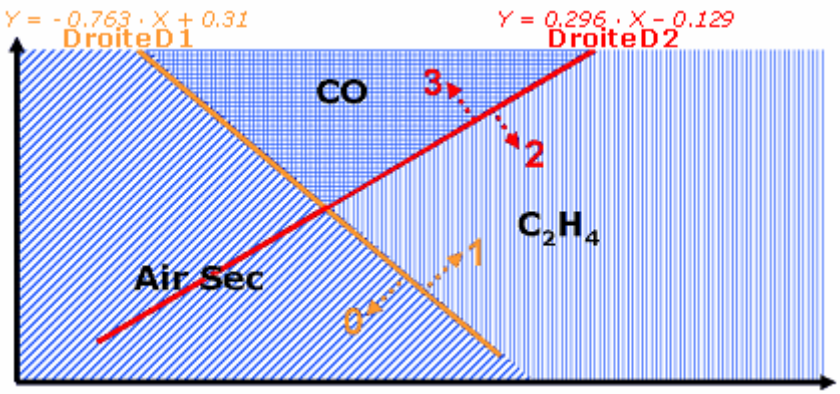


Figure 4.10 : Illustration de notre arbre de décision pour discriminer l'air sec (AS) et 2 gaz, le monoxyde de carbone (CO) et l'éthylène (C_2H_4).

Remarque : Dans notre programme en C, nous avons programmé une fonction qui s'appelle *arbredecision* (*ANNEXE A*). Cette fonction utilisant les coordonnées des points (matrice de résultat) nous permettra de discriminer le gaz.
Pour cette discrimination, cette fonction *arbredecision* fait appel à 2 fonctions, *DroiteD1* et *DroiteD2*. Comme le représente la *figure 4.10*, ces 2 fonctions (DroiteD1 ; DroiteD2) retournent chacune une valeur (0 – 1 pour DroiteD1; 2 – 3 pour DroiteD2) suivant que le point (matrice résultat) se trouve au dessus ou en dessous de la droite en question.

Ainsi, à partir des valeurs retournées par les fonctions DroiteD1 et DroiteD2, la fonction *arbredecison* va pouvoir discriminer le gaz :

droiteD1 = 0	→ Air Sec
droiteD1 = 1 et droiteD2 = 3	→ CO
droiteD1 = 1 et droiteD2 = 2	→ C_2H_4

IV.2.3 Algorithme du programme

L'algorithme du programme est représenté *figure 4.11*. Les différentes étapes sont :
- L'écriture des matrices d'entrée et des vecteurs propres.
- La multiplication des ces 2 matrices dont le résultat forme la matrice de résultat.
- L'utilisation de l'arbre de décision pour comparer les points avec les coordonnées des deux droites D1 et D2.

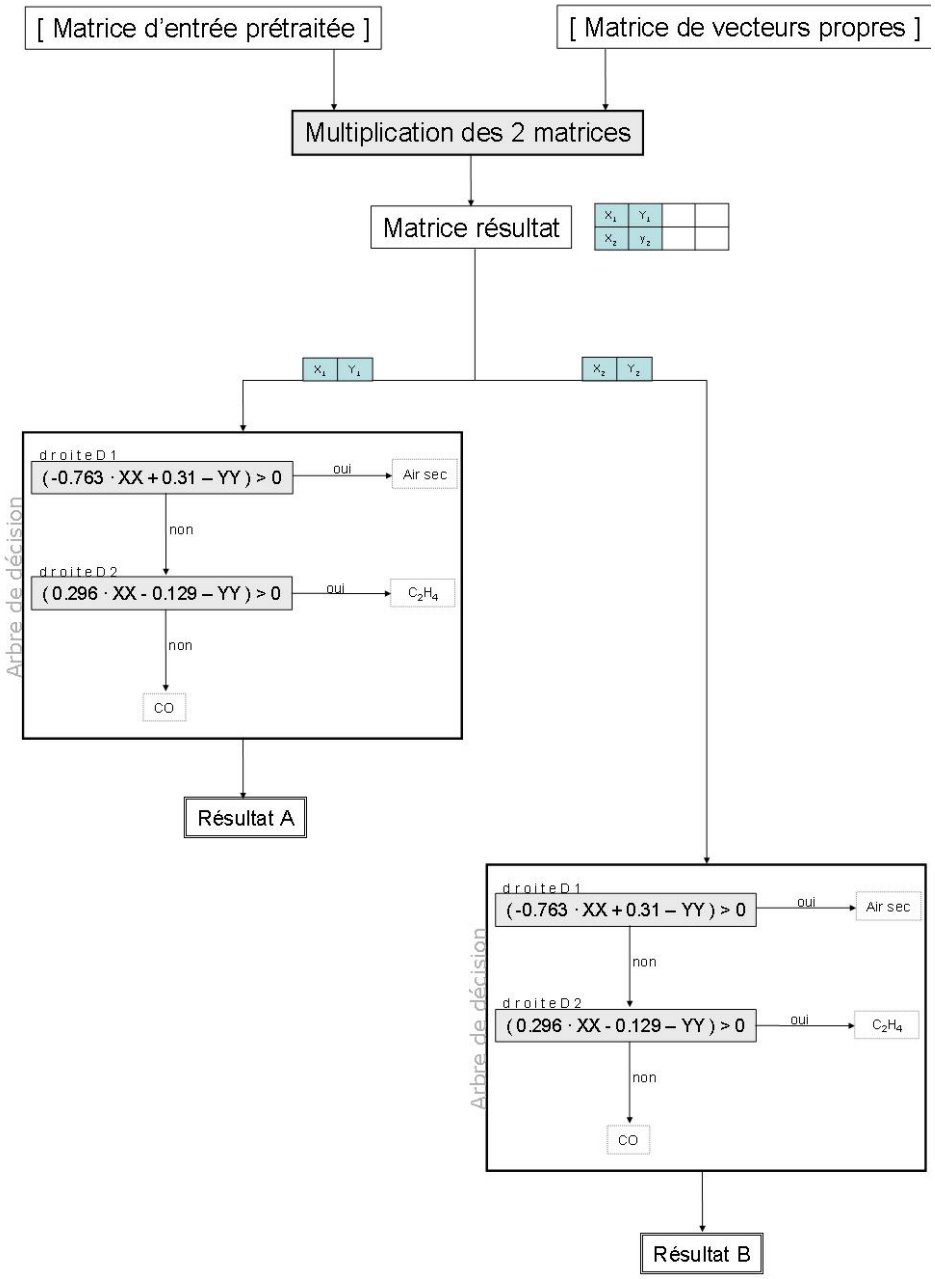


Figure 4.11 : Algorithme du programme.

IV.3 Extraction et analyse du programme en langage assembleur

IV.3.1 Logiciels

Pour programmer les microcontrôleurs PIC, le fabricant Microchip fournit un logiciel MPLAB qui est un éditeur, compilateur et simulateur en langage assembleur (*figure 4.12*). L'association avec le logiciel CC5x qui est compilateur C – assembleur nous a permis de programmer directement en langage C le programme de notre arbre de décision

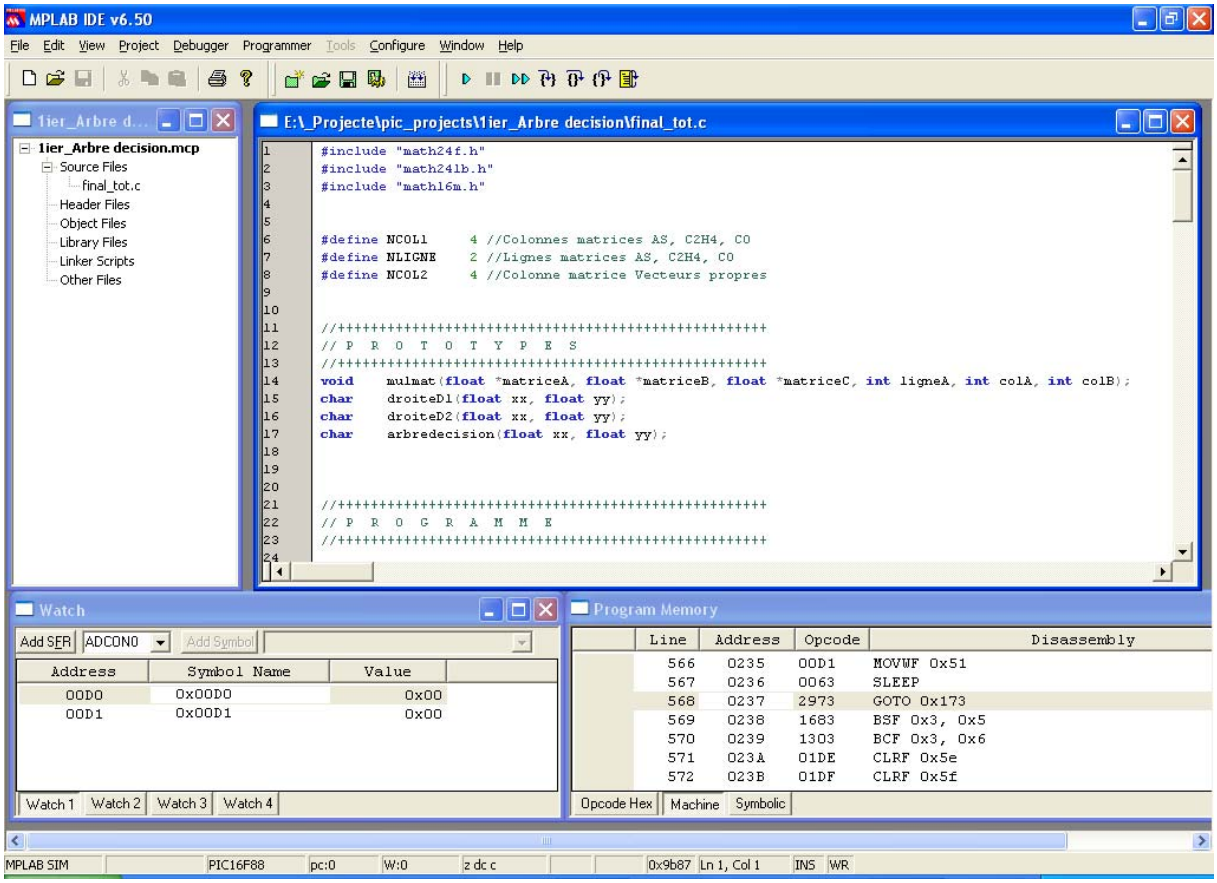


Figure 4.12 : Interface du logiciel MPLAB.

IV.3.2 Analyse du programme

Le programme en assembleur que nous avons obtenu, fait appel à d'autres fonctions prédéfinies, *fadd24*, *fmult24*, *multm8_8* et *fsub24*. Ces fonctions sont utilisées pour réaliser les opérations d'addition, de multiplication et de soustraction.

Comme nous l'avons expliqué dans le paragraphe VI.1.1, en analysant le programme en assembleur, chaque variable *float* est exécutée par 6 instructions et nécessite 3 octets de zone mémoire.

La mémoire RAM des microcontrôleurs PIC est divisée en plusieurs banques qui sont sélectionnées par les bits *RP0* et *RP1* du registre *STATUS*.

IV.3.3 Paramètres : nombre d'instructions et de cycles d'horloge

Pour choisir le type de microcontrôleur PIC, il nous faut prédire, voire connaître le nombre de cycles d'horloge et d'instructions nous permettant d'en déduire respectivement la fréquence du quartz associé au microcontrôleur et la dimension de la zone mémoire.

Avec 2 capteurs et 4 instructions, nous obtenons (tableau 4.1) :
- 981 instructions.
- environ 25.000 cycles d'horloges à ±5%.

Fonctions	Nombre d'instructions	Nombre de cycles d'horloge
MAIN	181	**24676**
MULTMAT	200	22346
ARBREDECISION	109	1066
DROIDED1	52	472
DROIDED2	52	472
_MULTM8_8	150	64
_MULT24	105	145
_FADD24	201	258
_FSUB24	9	11

Tableau 4.1 : Nombre d'instructions et de cycles d'horloge pour chaque fonction utilisée.

En se basant sur le tableau 4.1 et en analysant les instructions de chaque fonction, nous obtenons pour le cas le plus simple (c'est à dire 1 capteur avec 2 variables), les 2 caractéristiques (nombre d'instructions et de cycles d'horloge) :
- 857 instructions.
- 4000 cycles d'horloge à ±5%.

Ainsi de ces derniers résultats, nous en déduisons une relation qui va nous permettre d'établir le nombre de cycles d'horloges en fonction du nombre d'individus (capteurs) et du nombre de variables :

$$\begin{aligned} \text{Nbre cycles} &= \text{nbre individus} \cdot [\ 668 \cdot (\text{nbre variables})2 \ + \ \text{ARBREDECISION}\ ] \\ &= \mathbf{nombre\ individus \cdot\ [\ 668 \cdot\ (nombre\ variables)^2\ + 1066\ ]} \end{aligned} \quad (4.1)$$

IV.4 Conclusions

Le programme de notre arbre de décision nous a permis de discriminer l'air sec (AS) et les 2 gaz, monoxyde de carbone (CO), l'éthylène (C_2H_4) à partir de la variation de la résistance de la couche sensible des 2 capteurs SnO_2 à dopage différent.

De plus, nous avons pu en déduire une relation entre le nombre de cycles d'horloges de notre programme avec le nombre de capteur et de variables utilisés.

Ainsi connaissant, les nombres d'instructions et de cycles d'horloges de notre arbre de décision, nous pouvons en déduire un certain nombre de microcontrôleurs PIC pour implanter notre programme (*tableau 4.2*).

Microcontrôleur PIC	Mots (nombre d'instructions)	Nombre d'octets	Fréquence maximale du quartz
PIC16F688	4096	7168	20 MHz
PIC16F716	2048	3584	20 MHz
PIC16F870	2048	3584	20 MHz
PIC16F871	2048	3584	20 MHz
PIC16F872	2048	3584	20 MHz
PIC16F873	4096	7168	20 MHz
PIC16F876	8192	14336	20 MHz
PIC16F88	4096	7168	20 MHz

Tableau 4.2 : Premier choix des microcontrôleurs PIC

Remarque : Les microcontrôleurs PIC énumérés dans le *tableau 4.2* sont composés d'un convertisseur analogique numérique permettant l'acquisition de la valeur de la résistance (R_s) de la couche sensible du capteur (*figure 2.3*). De plus, ces microcontrôleurs PIC font partie de la famille des 16FXXX qui ont un jeu d'instructions de 14 bits. Ainsi en prenant l'exemple du PIC 16F870, la mémoire programme est de 28672 bits (2048x14) mis sous la forme de 3584 octets, de « cases » mémoires (28672/8).

Pour tous les PIC sélectionnés, la fréquence maximale du quartz est de 20MHz. Tenant compte une pré-division de 4, la fréquence associée au PIC sera de 5 MHz et donc, un cycle d'horloge sera réalisé en 0,2 μs. Suivant cette valeur, notre programme, formé de 981 instructions, sera réalisé en 5 ms (0,2 μs x 25000 cycles d'horloge).

Cependant, une partie du programme nous manque encore concernant l'acquisition et le prétraitement des données issues de la résistance du capteur. En effet, ces 2 opérations vont augmenter le nombre d'instructions, donc le nombre de cycles d'horloge. Par contre, il nous sera possible d'utiliser *l'équation 4.1* en tenant compte de ces 2 opérations pour connaître le nombre de cycles d'horloges de notre programme. Néanmoins, vu la taille de la mémoire programme des microcontrôleurs PIC sélectionnés (de 2048 à 8192 instructions) (tableau 4.2), nous pouvons supposer que le microcontrôleur qui sera utilisé fait partie de cette sélection.

V Conclusions

Le travail que j'ai réalisé pendant mon stage de 3 mois a porté sur l'étude du circuit électronique de traitement du signal de capteurs de gaz SnO_2 étudiés au LAAS.

De l'analyse des principales architectures dédiées aux traitements des données, nous avons choisi d'utiliser un microcontrôleur PIC pour réaliser notre prototype. Néanmoins, nous n'avons pas pu choisir un type de microcontrôleur car nous n'avions pas encore le nombre d'instructions et de cycles d'horloges totaux mais cette étude va permettre de faire un choix rapide de microcontrôleur dès que le système complet sera programmé.

Ce stage m'a permis d'approfondir mes connaissances dans le domaine de l'électronique avec l'analyse des circuits ASIC, FPGA, DSP et microcontrôleurs. De plus, j'ai pu à nouveau utiliser le logiciel MPLAB et ainsi accroître mes capacités dans la programmation de microcontrôleurs.

Ce stage m'a aussi montré les recherches réalisées dans le domaine des nez électroniques.

VI Annexes

VI.1 Annexe A : Programme en C

```
#include "math24f.h"
#include "math24lb.h"
#include "math16m.h"

#define NLIGNE    2 //Lignes   matrice d'entrée prétraité
#define NCOL1     4 //Colonnes matrice d'entrée prétraité
#define NCOL2     4 //Colonnes matrice Vecteurs propres


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++
// P  R  O  T  O  T  Y  P  E  S
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++
void   mulmat(float *matriceA, float *matriceB, float *matriceC, int ligneA, int colA, int colB);
char  droiteD1(float xx, float yy);
char  droiteD2(float xx, float yy);
char  arbredecision(float xx, float yy);


//+++++++++++++++++++++++++++++++++++++++++++++++++++++++
// P  R  O  G  R  A  M  M  E
//+++++++++++++++++++++++++++++++++++++++++++++++++++++++

void main()
{
      // matAS[NLIGNE*NCOL1]        : Matrice d'entrée prétraité
      // matfact[NCOL1*NCOL2]       : Matrice Vecteurs propres
      // matresult[NLIGNE*NCOL1]    : Matrice Résultat

      float matfact[NCOL1*NCOL2];
      #pragma rambank 1
      float matAS[NLIGNE*NCOL1];
      float matresult[NLIGNE*NCOL1];
      char resultatA,resultatB;

      /* Matrice d'entrée prétraité matAS */
      matAS[0]=0.1097;
      matAS[1]=0.1168;
      matAS[2]=0.4654;
      matAS[3]=0.5750;
      matAS[4]=0.0764;
      matAS[5]=0.0801;
      matAS[6]=0.5415;
      matAS[7]=0.6692;

      /* Matrice de vecteurs propres */
      matfact[0] = 0.4041;
      matfact[1] = 0.5439;
      matfact[2] = 0.0781;
      matfact[3] = -0.4349;
      matfact[4] = 0.8896;
      matfact[5] = -0.5463;
      matfact[6] = 0.0437;
      matfact[7] = -0.8754;
      matfact[8] = 0.1148;
      matfact[9] = -0.5002;
```

```
	matfact[10] = 0.7721;
	matfact[11] = 0.1884;
	matfact[12] = 0.1792;
	matfact[13] = 0.3944;
	matfact[14] = -0.6292;
	matfact[15] = 0.0947;

	/* Multiplication matresult = matAS * matfact */
	mulmat(&matAS[0],&matfact[0],&matresult[0],NLIGNE,NCOL1,NCOL2);

	resultatA=arbredecision(matresult[0*NCOL1+0],matresult[0*NCOL1+1]);
	resultatB=arbredecision(matresult[1*NCOL1+0],matresult[1*NCOL1+1]);
}

//+++++++++++++++++++++++++++++++++++++++++++++++++++
// F  O  N  C  T  I  O  N  S
//+++++++++++++++++++++++++++++++++++++++++++++++++++

//Cette fonction fait la multiplication de deux matrices C=A*B
void mulmat(float *matriceA, float *matriceB, float *matriceC, int ligneA, int colA, int colB);
{
	int ii,jj,gg;
	float mult,val = 0;
	int posA,posB;

	for(ii=0; ii<ligneA;ii++)
	{
		for(jj=0; jj<colB; jj++)
		{
			for(gg=0; gg<colA; gg++)
			{
				posA=ii*colA+gg;
				posB=gg*colB+jj;
				mult=matriceA[posA];
				mult=mult*matriceB[posB];
				val = val + mult;
			}
			posA=ii*colB+jj;
			matriceC[posA]=val;
			val=0;
		}
	}
}

//-----------------------------------
char droiteD1(float xx, float yy)
{
	if((-0.763*xx+0.31-yy)>00)
		return 0;
	else
		return 1;
}

//-----------------------------------
char droiteD2(float xx, float yy)
{
	if((0.296*xx-0.129-yy)>0)
		return 2;
	else
		return 3;
}
```

```
//-----------------------------------
char arbredecision(float xx,  float yy)
{
      if(droiteD1(xx,yy)==0)
            return(5);        //Air Sec
      if((droiteD1(xx,yy)==1) && (droiteD2(xx,yy)==3))
            return(6);        //CO
      else
            return(7);        //C2H4
}

//-----------------------------------
```

VII Bibliographie

[1] **S. ASTIE**, Intégration d'un capteur de gaz à oxyde semi-conducteur sur silicium. Thèse de doctorat (1998).

[2] **V. DELARME**, Thèse de doctorat nº934, Ecole polytechnique fédéral de Lausanne (1991).

[3] **Thierry VIARD**, Traitement de signal appliqué aux capteurs chimiques. Amélioration des performances : Précision, sélectivité, fiabilité. Thèse de doctorat (1999).

[4] **Arnaud TISSERAND**, Introduction aux circuits FPGA. Notes (Décembre 2003).

[5] **E. SISCARD, S. DELMAS-BENDHIA**, Une introduction aux DSP, Notes (2001).

[6] **Arnaud TISSERAND**, DSP : des processeurs dédiés au traitement numérique du signal. Notes (Mai 2003).

[7] **D. COHEN**, Paralelismo y segmentación. Notes (2003)

[8] SED basados en microcontroladores. Notes (2003).
http://www.fci.uach.cl/ceelectronica/gral/documentos/microcontroladores.pdf

Adresses d'internet

- MICROCHIP [http://www.microchip.com]
- CC5X [http://www.bknd.com/cc5x]