SOMMAIRE

Introduction

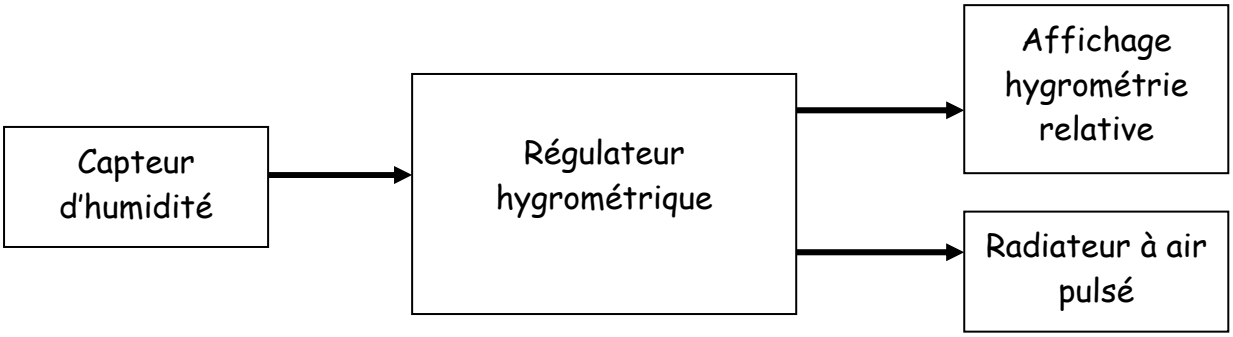
Dans le cadre du bureau d'étude de deuxième de l'IUP AISEM, nous avons été amenés à réaliser un projet. Celui-ci a pour but de mettre en pratique nos connaissances acquises en cours et de travailler en équipe.

Notre travail sera orienté sur la réalisation d'un régulateur hygrométrique à microcontrôleur.

II Présentation de l'objet technique

II-1 Cahier des charges

Réalisation d'un régulateur hygrométrique ayant pour but d'afficher et de maintenir le taux d'hygrométrie d'un local inférieur à 50% en brassant et chauffant l'air de celui-ci. Il est composé d'un capteur d'hygrométrie de type humidistance (Sensirion ref :SHT1x), d'une carte électronique de contrôle et d'un radiateur de 500W.



II-2 Document de spécification

Le système à réaliser est un régulateur hygrométrique contrôlé par microcontrôleur ATMEL.

Il donnera le taux d'hygrométrie d'une pièce grâce à un capteur ayant une plage de mesure allant de 0% à 100% avec une résolution de 0.03%. Ce taux sera ensuite affiché sur un afficheur LCD avec une précision de 0.1%.

Si le taux d'hygrométrie est inférieur à 50%, un radiateur à air pulsé sera automatiquement mis en route.

Fonctionnel :

La maquette réalisée doit permettre de :

- Piloter un capteur d'hygrométrie du type SHT1X
- Traiter les données fournies par ce capteur
- Commander un radiateur à air pulsé
- Afficher les informations sur l'hygrométrie

Matériel :

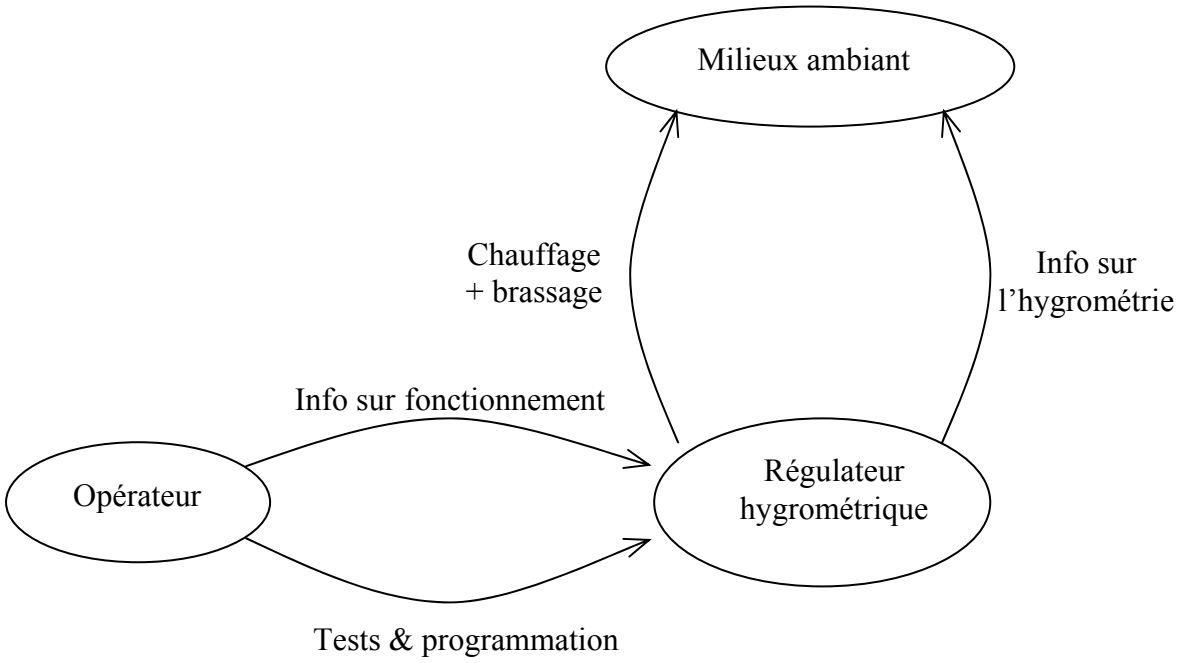
- Carte ATMEL ICE200
- Alimentation de laboratoire
- capteur SHT11

Logiciel :

Les logiciels à réaliser sont :

- la commande du capteur d'hygrométrie SHT1X
- la commande de l'afficheur LCD CMC216-04
- la commande du radiateur à air pulsé
- le calcul de l'humidité relative

II-3 Diagramme sagittal

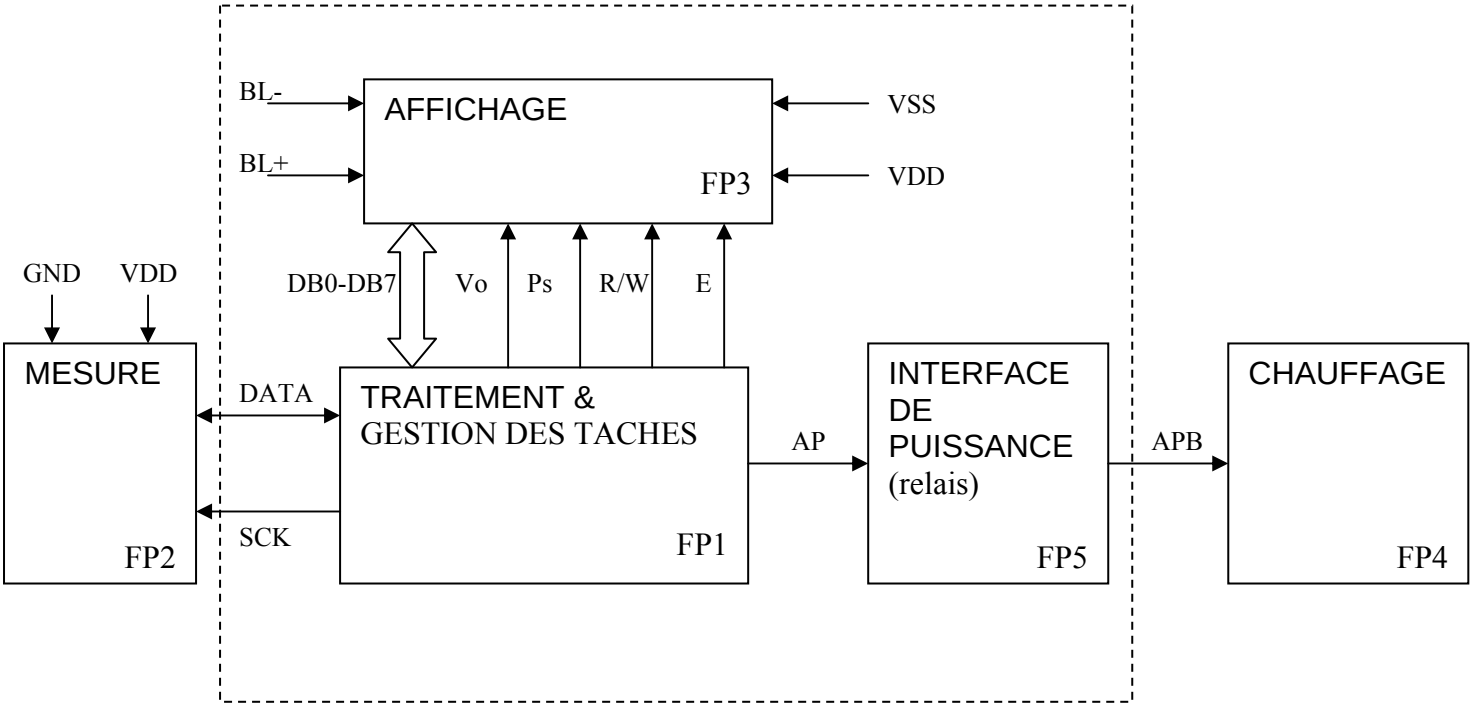


III Développement du projet

III-1 Schéma fonctionnel

Après avoir étudier précisément chaque partie de l'objet technique, nous avons cherché les différentes options qui seraient susceptibles de fonctionner avec nos exigences pour respecter le cahier des charges.

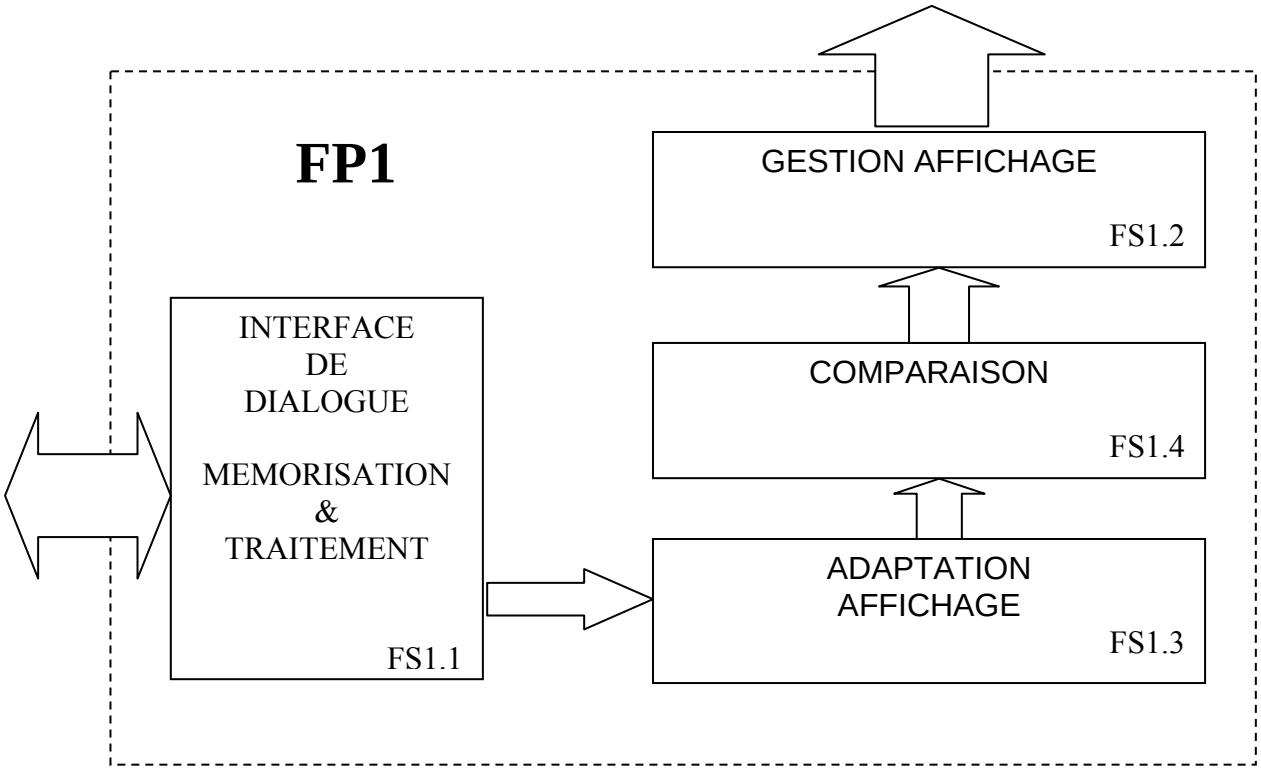
Pas conséquent, nous avons défini 5 fonctions matérielles principales :



III-2 Description et analyse des fonctions principales

III-2-1 Traitement et gestion des tâches

Cette fonction est réalisée grâce à un microcontrôleur de type ATMEL ATS90S8535, composant qui répond exactement à nos exigences en rapidité et en compatibilité (taille mémoire), et son environnement associé permettant de gérer l'ensemble du système. C'est le cœur du système. Il stock en mémoire tous les programmes permettant d'effectuer les différentes tâches, à savoir la commande du capteur de mesure d'hygrométrie, le calcul de l'humidité relative, la commande de l'afficheur LCD et celle du radiateur à air pulsé.



- **FS1.1 :** Interface de dialogue, mémorisation & traitement :

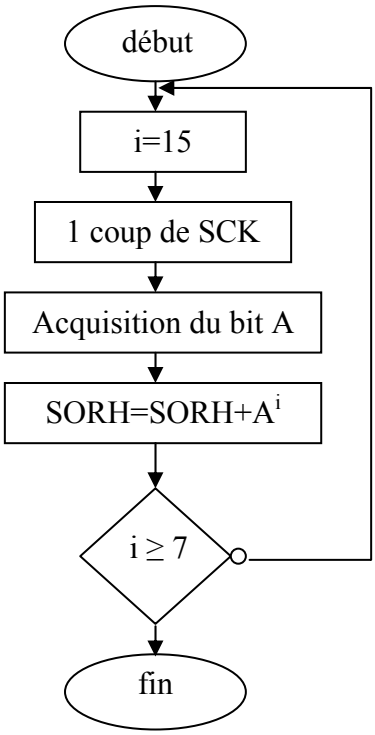
Cette fonction secondaire est une fonction très complète qui permet de créer un mot de 16 bits appelé SORH, qui est une image binaire de l'hygrométrie.
Elle se décompose en trois parties qui sont :

- Sous programme d'acquisition de la valeur haute (8 premiers bits). Ce programme va communiquer avec le capteur bit après bit, chaque bit est enregistrer et pondérer par son rang, d'où l'utilisation d'une somme itérative.
- Sous programme d'acquisition de la valeur basse (7 derniers bits). Ce programme prend lui aussi en charge la communication avec le capteur, il utilise aussi la même méthode de conversion binaire → décimale.
- Sous programme « 8ème ». Permet d'additionner le dernier bit à la somme SORH.

Remarque : La méthode de conversion binaire → décimale utilisée nécessite la création d'une fonction « puissance » qui élève un nombre x à la puissance y. (fonction « pow »).

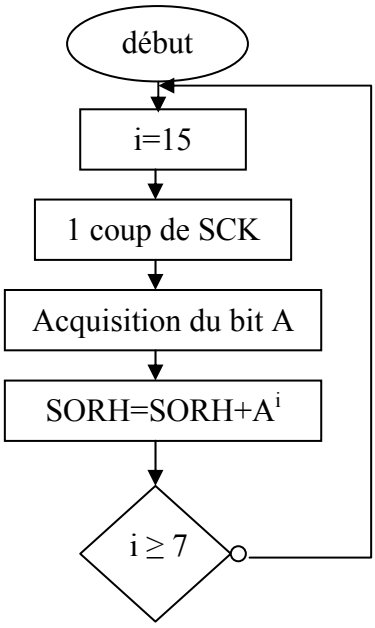
Algorigrammes :

SCAN_L

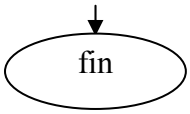



```
void scan_l()
 {
 int y,byt;
 /*commande du capteur & conversion binaire->décimal de la VALEUR BASSE
  remarque :juste les 7 derniers bits le 8eme est ajouter plus tard*/
          for(y=7;y>=1;y--)
          {
          SCK=0;
          delay_ms(5);
          SCK=1;
          delay_ms(10);
          byt=PIND.4;
        SORH = SORH + (byt*pow(2,y));
        SCK=0;
          delay_ms(5);
        }
 }
```

SCAN_H



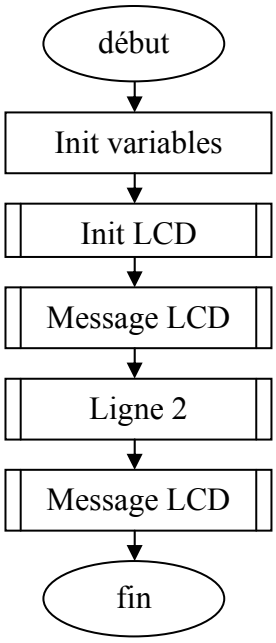

```
void scan_h()
{
int y,byt;
/*commande du capteur & conversion binaire->décimal de la VALEUR HAUTE*/
          SORH=0; /*mise a 0 de SORH(Valeur décimale)*/
          for(y=15;y>7;y--)
          {
          SCK=0;
          delay_ms(5);
          SCK=1;
          delay_ms(10);
          byt=PIND.4;
        SORH = SORH + (byt*pow(2,y));
        SCK=0;
          delay_ms(5);
        }
 }
```



- **FS1.2** : Gestion affichage :

Le cahier des charges impose l'utilisation d'un afficheur LCD de type 2x16 caractères, dont le fonctionnement est décrit dans le chapitre traitant la fonction principale 3 (FP3). Nous n'allons donc pas redéfinir le fonctionnement de l'afficheur. Il est cependant important de connaître l'ordre des différentes instructions.
Remarque : Pour une simplification du programme, nous n'avons utilisé qu'un seule fonction d'écriture dans le LCD, ce qui nous impose un temps de latence non négligeable.

Algorigramme :



Programme en annexe 1

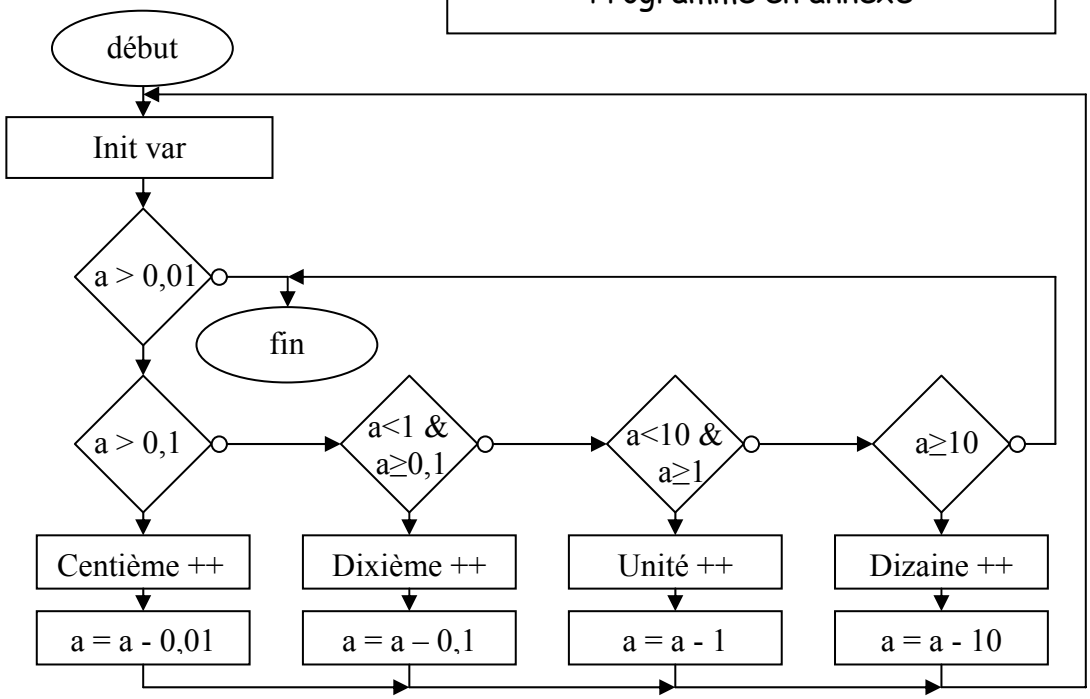
- **FS1.3** : Adaptation d'affichage

Le sous programme de la fonction FS1.1 transmet un mot (décimal) de 16 bits appelé RH_{lin} qui est donc la valeur brut de l'image de l'hygrométrie. Nous avons besoin d'afficher cette valeur, mais le programme d'affichage tel qu'il est ne le permet pas, car il ne peut prendre en paramètres que des valeurs de type « char ». D'où l'utilité du sous programme « udcd » (Unité Dizaine Centième Dixième), nous pouvons lui passer en paramètre la valeur de type « réel », a

décomposer et grâce à 4 variables globales de type « char » que l'on va incrémenter.

Algorigramme :

Programme en annexe



- **FS1.4** : Fonction comparateur

Cette fonction est très simple car selon le cahier des charges, lorsque l'hygrométrie est supérieure à 50%, on allume le chauffage. Il nous suffit donc de comparer le chiffre des dizaines « d » à 5 pour mettre en route le radiateur.

III-2-2 Mesure

Le SHT11 de Sensirion est un capteur d'humidité et de température, c'est celui-ci que nous allons utiliser. Il ne nécessite pas de calibrage, et est complètement numérique. Il est de plus, d'un point de vue structurel, facile à mettre en œuvre.

Extrait documentation constructeur :

2 Interface Specifications

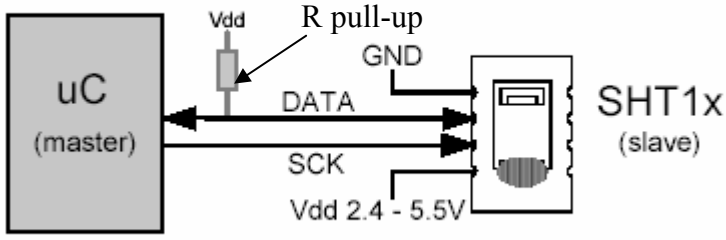


Figure 2 Typical application circuit

Remarque : la résistance de pull-up est intégrée dans le µcontrôleur.

Le Sensirion SHT11 est constitué de 2 capteurs (température et humidité), d'un convertisseur Analogique/Numérique 14 bits, d'une mémoire servant aux paramètres de calibrage et d'une interface série I_2C. Le calibrage s'effectue en usine. Il peut être alimenté en 3V ou 5V, nous prendrons cette dernière qui est déjà présente sur la carte pour alimenter le µcontrôleur et l'afficheur LCD.

Pour obtenir une mesure, il suffit d'envoyer une commande précédée de l'adresse correspondante au capteur. Les signaux que renvoient le capteur sont codés sur 12 ou 14 bits pour la température et 8 ou 12 bits pour l'hygrométrie. Mais ces signaux ne sont pas exploitables directement, ils nécessitent un traitement numérique plus ou moins complexe pour les convertir en valeur physique.

L'hygrométrie : Le capteur possède deux types de mesures du taux d'hygrométrie, une mesure dite « linéaire » appelée RH_{lin} et une autre dite « vraie » appelée RH_{VRAI} . Nous nous sommes concentrés sur la mesure de RH_{lin}, car du fait que la mesure de RH_{VRAI} ait besoin d'une compensation en température, l'acquisition devient beaucoup plus complexe.

Nous utilisons le capteur en mode 12 bits pour une plus grande précision. Le port série bidirectionnel de données (DATA) et l'horloge (SCK) sont connectés au port D.

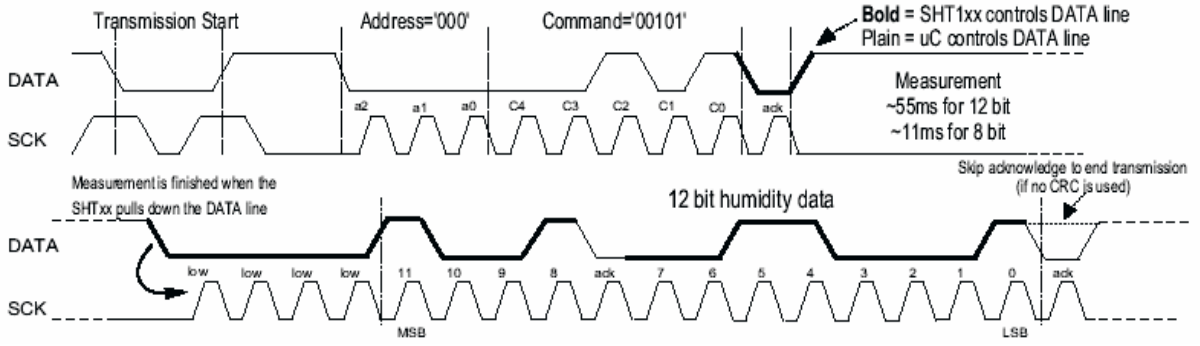
La transmission s'effectue en plusieurs temps :

- envoie d'une séquence de début de transmission
- envoie de l'adresse du capteur suivit de la commande (température ou humidité)
- envoie d'un accusé de réception par le capteur (mise à 0 de la ligne DATA)
- attente pendant que le capteur effectue la mesure
- réception de 8 bits de données (poids fort)
- envoie d'un accusé de réception par l'ATMEL
- réception de 8 bits de données (poids faible)

L'adresse du capteur est « 000 » car ici il n'y en a qu'un seul, mais il serait possible de mettre plusieurs capteurs sur la même liaison série, et de les interroger individuellement.

Exemple de transmission hygrométrie sur 12 bits :

Documentation constructeur :



Remarque : Le programme relatif à cette fonction se situe page 12

Après avoir transmis la commande, le capteur envoie un accusé de réception, il faut donc passer le port D en entrée pour détecter le passage de la ligne de DATA à 0. Ensuite on reçoit les 8 bits de poids fort de la donnée, mais les quatre premiers ne servent pas car la transmission se fait sur 12 bits. Le port D repasse alors en sortie pour passer la ligne DATA à 0 et ainsi indiquer au capteur que les données ont bien été reçues. Après avoir repasser le port D en entrée, l'ATMEL reçoit les 8 bits de poids faibles.

III-2-3 Affichage

La fonction affichage est principalement composée d'un afficheur LCD 2x16 caractères. Sa mise en œuvre est relativement simple car la communication s'effectue unidirectionnellement, de l'ATMEL vers le module LCD. Le boîtier se compose de 16 broches dont 5 sont déjà câblées sur la platine (alimentation), reste un bus de données, une entrée E de validation, une entrée RS de sélection du registre et une entrée R/W de commande de lecture/écriture. Cette dernière sera fixée à un niveau logique bas car nous n'utilisons l'afficheur qu'en mode écriture.

La mise en œuvre nécessite une initialisation qui consiste à envoyer une série de codes sur la bus DATA. Ces différents codes requièrent un temps d'exécution maximum de 1,53 ms, nous l'avons fixé à 2 ms en pratique afin d'être

sûr de la bonne prise en compte de l'instruction. Les différents codes et leurs temps relatifs sont disponibles en annexe 2.

III-2-4 Chauffage

&

III-2-5 Interface de puissance

Le chauffage qui est selon le cahier des charges, un radiateur à air pulsé, sera modélisé sur la plaquette par l'allumage d'une LED, l'affichage d'un message à l'écran et un signal sonore.

La fonction « interface de puissance » ne sera pas réalisée en pratique, cependant elle aurait permit la commande de tension élevée qui serait nécessaire au fonctionnement d'un radiateur de 500W. Il est important d'isoler galvaniquement la partie puissance de la partie commande.

Photo représentant l'afficheur lors de la mise an marche du chauffage avec la LED « LED0 » allumée.

IV Programme principal

Nous avons utiliser la compilation séparé pour le programme principal, ce qui allège son écriture et sa compréhension. L'include « hygro.h » et le fichier contenant le code des fonctions (hygro.c) sont disponible en annexe 3.

Programme :

```
#include <90s8535.h>
#include <delay.h>
#include <stdio.h>
#include "hygro.h"

#define DATA PORTD.4
#define SCK PORTD.2
#define VCC PORTD.0
#define GND PORTD.6
#define DATA_LCD PORTC
#define RS PORTA.7
#define E PORTA.6

int centieme=0;
int dixieme=0;
int unite=0;
int dizaine=0;
int SORH=0;


void main (void)
{
float RHlin;
DDRA=0xff;
DDRB=0xff;
DDRC=0xff;
DDRD=0x55;
VCC=1;
GND=0;

while(1)
{
centieme=0;
dixieme=0;
unite=0;
dizaine=0;


        PORTB.4=1;
        transtart();
        address();
        command1();
```

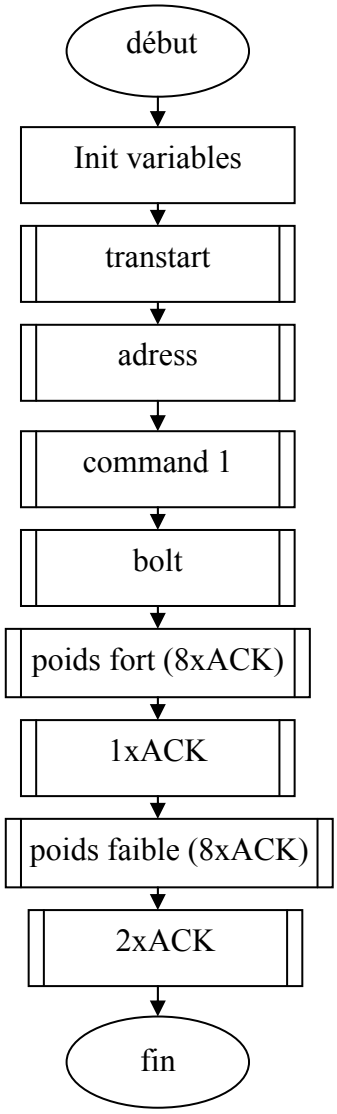
```
        bold();
        scan_h();

        DDRD=0x55; /*DATA en sortie*/
        DATA=0;
        sck(1); /*ack*/
        DATA=1;

        DDRD=0x45; /*DATA en entree*/
        PIND.4=1;

        scan_l();
      huitieme();
        DDRD=0x55; /*DATA en sortie*/
        DATA=1; /*forcage de de DATA a 1, car pas de gestion des erreurs(CRC)*/
        sck(1); /*ack*/

      RHlin=(0.0405*SORH)+(-(2.8e-6)*SORH*SORH)-4;

      udcd(RHlin);
      if(dizaine>=5)
      {
      Efface_LCD();
        Init_Lcd();
      Message_LCD("taux > 50%");
      Ligne_LCD(2);
      Message_LCD("chauffage ON");
      PORTB.6=0;                    /*led allumée*/
      PORTB.4=0;                    /*buser=0*/
      delay_ms(100);
      PORTB.4=1;                    /*buser=1*/
      }
      else
      {
      PORTB.6=1;
      Efface_LCD();
        Init_Lcd();
        Message_LCD("hygrometrie :");
        Ligne_LCD(2);
        Wr_LCD(dizaine+48,1);
        Wr_LCD(unite+48,1);
        Message_LCD(",");
        Wr_LCD(dixieme+48,1);
        Wr_LCD(centieme+48,1);
        Message_LCD("%");
        }
        }
}
```

V Test et validation sur plaquette de développement

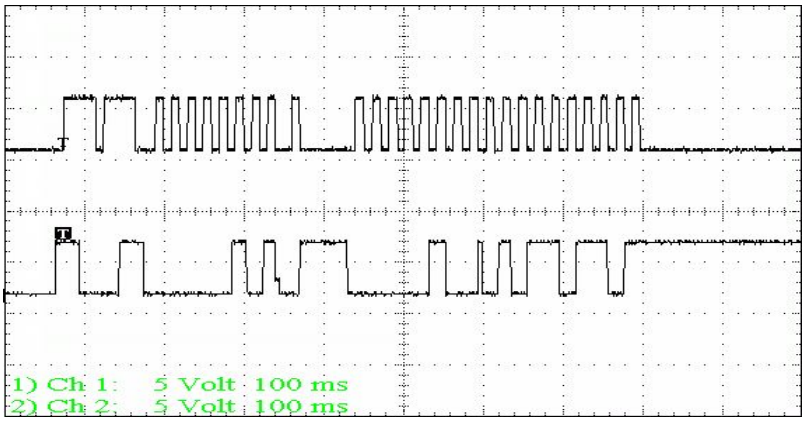
Chaque fonction doit pouvoir être indépendamment validée, nous avons séparé ces tests en deux parties : test de la fonction mesure et test de la fonction affichage. Pour cela nous avons développé deux programmes indépendant.

Test capteur :

Ce test consiste à envoyer la série d'instruction demandée par le constructeur afin de visualiser à l'oscilloscope les trames permettant le calcul de l'hygrométrie.



Programme en annexe 4



Chronogramme visible sur PORTD.4 et PORTD.2

Test afficheur :

Test permettant de tester la totalité des caractères de l'afficheur. On peut voir « testafficheurLCD » sur les deux lignes. L'algorigramme est visible page 8.

```
#include <90s8535.h>
#include <delay.h>

#define DATA_LCD PORTC
#define RS PORTA.7
#define E PORTA.6

void Message_LCD(const char* texte);
void Wr_LCD(unsigned char cVar,int LCD_RS);
void Efface_LCD(void);
void Init_Lcd(void);
void Ligne_LCD(char cLigne) ;

void main (void)
{
DDRA=0b11111111;
DDRC=0b11111111;

Efface_LCD();
Init_Lcd();
Message_LCD("testafficheurLCD");
Ligne_LCD(2);
Message_LCD("testafficheurLCD");
}


void Init_Lcd(void)
```

```
{
Wr_LCD(0b00111000,0);
Wr_LCD(0b00001110,0);
Wr_LCD(0b00000110,0);
Wr_LCD(0b00000010,0);
}

void Message_LCD(const char* texte)
{
int i;
i=0;
while(texte[i]!='\0')
Wr_LCD(texte[i++],1);
}

void Wr_LCD(unsigned char cVar,int LCD_RS)
{
RS=LCD_RS;
DATA_LCD=cVar;
E=1;
delay_ms(2);
E=0;
}

void Ligne_LCD(char cLigne)
{
if(cLigne==1) Wr_LCD(0x80,0); // afficheur ligne 1
else if (cLigne==2) Wr_LCD(0xC0,0); // afficheur ligne 2
}

void Efface_LCD(void)
{
Wr_LCD(1,0);
}
```

VI Optimisation du projet

Dans le but d'améliorer la précision de la mesure de l'hygrométrie, il est utile de mettre en œuvre le capteur de température. Celui-ci se commande de la même manière que le capteur d'hygrométrie avec comme différence l'acquisition qui s'effectue sur 14 bits au lieu de 12 bits. L'acquisition de la température est un paramètre à considérer dans le calcul de l'hygrométrie dite réelle.

Extrait documentation constructeur :

3.1.1 Compensation of RH/Temperature dependency

For temperatures significantly different from 25 °C (~77 °F) the temperature coefficient of the RH sensor should be considered:

$$RH_{true} = (T_{°C} - 25) \bullet (t_1 + t_2 \bullet SO_{RH}) + RH_{linear}$$

SO_{RH}	t_1	t_2
12 bit	0.01	0.00008
8 bit	0.01	0.00128

Table 7 Temperature compensation coefficients

Le programme effectue alors deux mesures à la suite, une d'hygrométrie et une de température. Ces deux valeurs sont utilisées pour le calcul numérique de l'hygrométrie réelle. Programme principal en annexe 5.

Conclusion

Ce projet de bureau d'étude nous a permis d'accroître nos connaissances dans le domaine de la programmation de microcontrôleur.

La maquette fonctionne selon les attentes du cahier des charges, de plus, la modularité et la souplesse de la programmation en C nous a permis d'apporter des modifications rapidement.

Nous remercions Mr J.L. BOIZARD de nous avoir permis de développer le projet en langage C.

ANNEXES