INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

Rapport de BE C++

Implémentation d'une couveuse - ESP8266

Réalisé par
Chams Riman & Yiannis Manzo

Travail présenté à David Gauchard

Département de génie électronique et informatique - INSA Toulouse

19 Mai 2021

Table des matières

1 Introduction

Ce BE, dans la continuité des cours, TDs et TPs de C++, vise à nous pousser à réaliser un projet concret faisant intervenir les concepts phares de ce langage de programmation. Nous avons entrepris de créer une couveuse : ce système doit effectuer une régulation en température et humidité pour simuler correctement la couvaison d'une poule. En plus de la carte ESP8266 qui nous était fournie initialement, nous avons récupéré ou commandé :

— un capteur de température et d'humidité,

— une résistance chauffante,

— un brumisateur,

— un écran OLED.

Dans ce rapport, le fonctionnement général de notre système sera tout d'abord présenté par l'intermédiaire d'un diagramme de classe et de schémas explicatifs au niveau matériel et logiciel. Le code associé est disponible au lien suivant. Nous présenterons ensuite les problèmes auxquels nous nous sommes heurtés tout au long de ce projet. Nous conclurons enfin en proposant des idées d'amélioration de notre couveuse.

2 Diagramme de classe

La Figure 1 correspond au diagramme de classe de notre système. Les classes *Pin* et *U8X8* ne sont pas clairement explicitées puisque ce sont des librairies qui contiennent un grand nombre d'attributs et de méthodes.

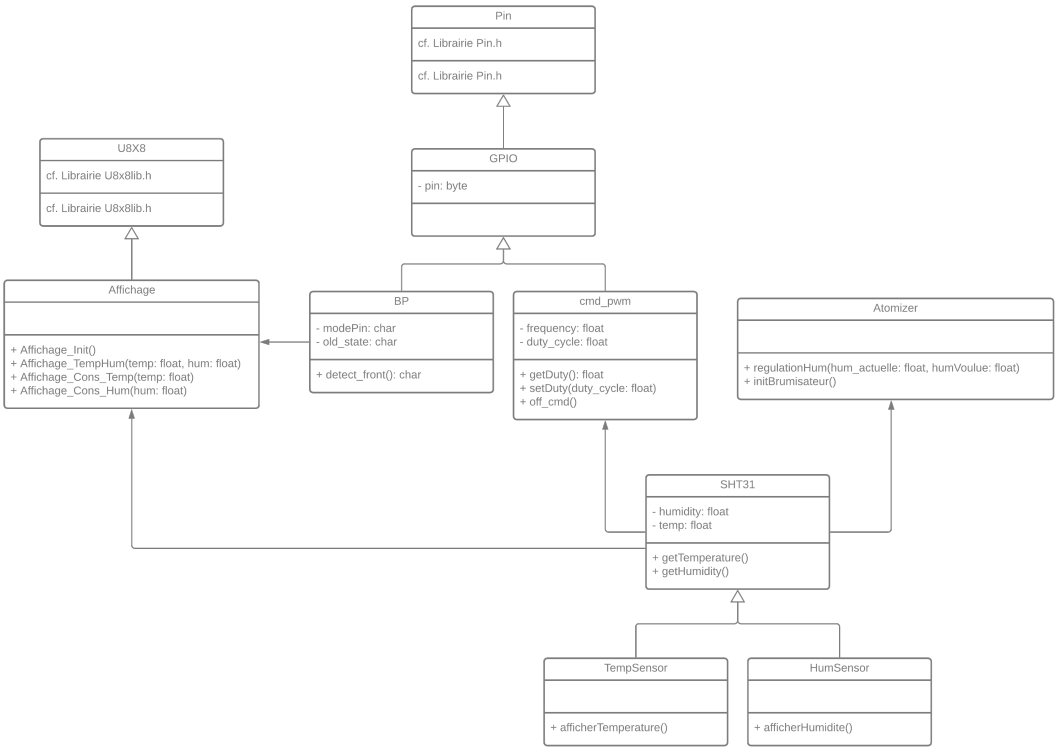

FIGURE 1 – Diagramme de classe du système

3 Schéma de fonctionnement

3.1 Matériel

La Figure 2 présente le schéma fonctionnel matériel de notre projet. Après téléversement par USB (qui l'alimente en courant), l'ESP8266 stocke et exécute le code. Elle intéragit alors avec le système. Grâce à une interruption, elle récupère les valeurs de température et d'humidité au sein de la couveuse par le capteur SHT31. Ces informations sont ensuite traitées par l'ESP8266 qui :

— envoie vers le circuit de gestion de la résistance chauffante une PWM dont le rapport cyclique varie selon l'erreur avec la température voulue ;

— active ou désactive (tout ou rien) le brumisateur selon la valeur de l'humidité ;

— actualise l'affichage de la température et de l'humidité sur l'écran OLED.

Notre système est conçu pour qu'il soit possible d'adapter les valeurs de température et d'humidité. Quatre boutons permettent ainsi de faire varier ces paramètres. La consigne est alors actualisée sur l'écran OLED.

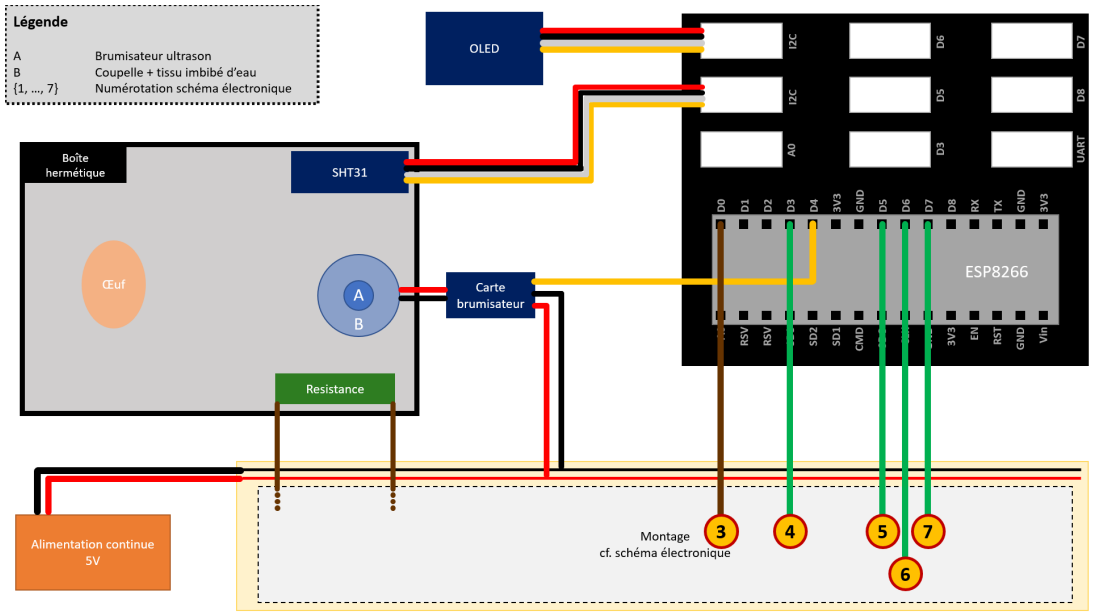

FIGURE 2 – Schéma fonctionnel matériel du système - Schéma électronique associé en annexe

3.2 Logiciel

La Figure 3 présente le schéma de fonctionnement logiciel. On y trouve d'abord le programme principal *main.ino* qui instancie, initialise et exécute la régulation en température et en humidité. Pour cela, il fait appel à plusieurs fichiers annexes :

— $Affichage.h$ qui logiquement gère l'affichage des consignes et des valeurs réelles sur l'écran OLED ;

— $Cmd\_R\_chauf.h$ qui définit la classe liée à la génération de PWM ainsi que celle liée à la gestion des boutons de modification de consignes (que nous aurions dû mettre dans un fichier à part mais que nous n'avons pas pu faire par manque de temps) ;

— *actuator.h* qui définit la classe liée au brumisateur (avec une méthode de régulation en humidité tout ou rien), faisant lui-même appel à :

— *sensor.h* qui contient les classes liées au capteur de température que nous faisons hériter directement de la classe $SHT31$.

La Figure 5 récapitule le fonctionnement du système.

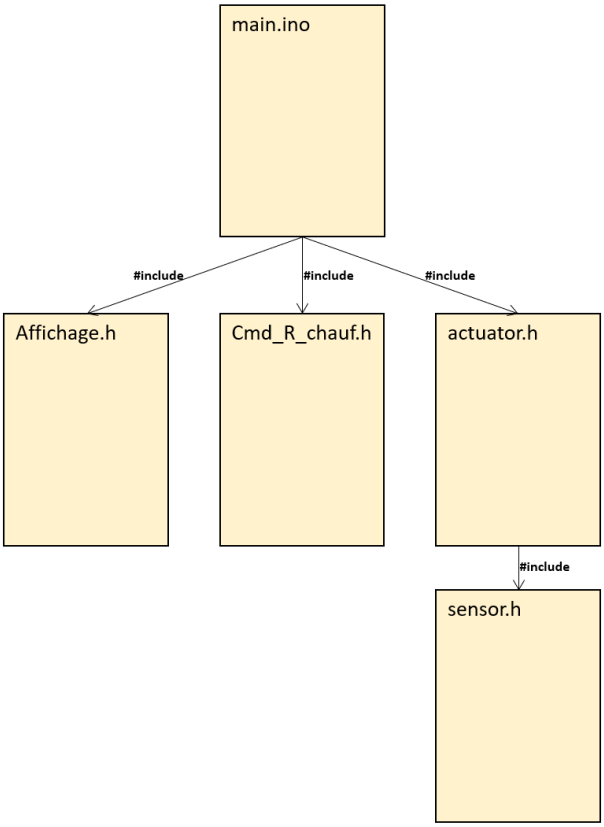

FIGURE 3 – Schéma de fonctionnement logiciel du système

4 Conclusion

Ravis d'avoir carte blanche pour le choix du projet, nous nous sommes beaucoup investis pour parvenir à un résultat concret au terme de ce BE C++. Bien qu'inutilisable en l'état actuel car ne chauffant pas suffisamment, notre couveuse a cependant un fonctionnement correct. En effet, le système est capable de générer les commandes permettant d'adapter la température et l'humidité par rapport à la consigne.

Ce BE nous a donné la possibilité de nous exercer au langage C++ en conditions réelles, tout en nous permettant de croiser cette matière avec de l'électronique et de l'automatique. Nous avons appris à gérer un projet de A à Z, de la conception à la réalisation en passant par le choix du matériel et la réalisation des schémas de fonctionnement.

Nous nous sommes heurtés à un grand nombre de problèmes dont certains non-résolus. Le plus ennuyeux est lié à la résistance chauffante : notre système a beau la rendre brûlante au toucher, elle ne rayonne presque pas et ne réchauffe donc pas du tout son environnement proche. En plus de cela, la boîte que nous utilisons est très peu hermétique. Nous avons également quelques problèmes liés à l'utilisation des boutons : l'un d'eux à tendance à faire geler l'affichage de l'écran OLED. Nous avons quelques pistes de solutions que nous n'avons pas eu le temps de creuser. Un dernier problème est plus scolaire : en effet, il ne nous a pas paru cohérent (et évident) de

redéfinir un opérateur ou d'utiliser la librairie STL dans notre cas. Par ailleurs, nous avons fait des héritages qui auraient pu être évités.
Nous pensons améliorer ce système de plusieurs façons :

- en utilisant une véritable résistance chauffante ;
- en nous munissant d'une boîte bien isolée ;
- en ajoutant une dimension automatique grâce à un PID sur la régulation de la température et de l'humidité ;
- en exploitant la connectivité WiFi de l'ESP8266 :
  - soit via une page HTML/CSS/Javascript (que nous avions commencé à faire),
  - soit via une application mobile.

A Annexes

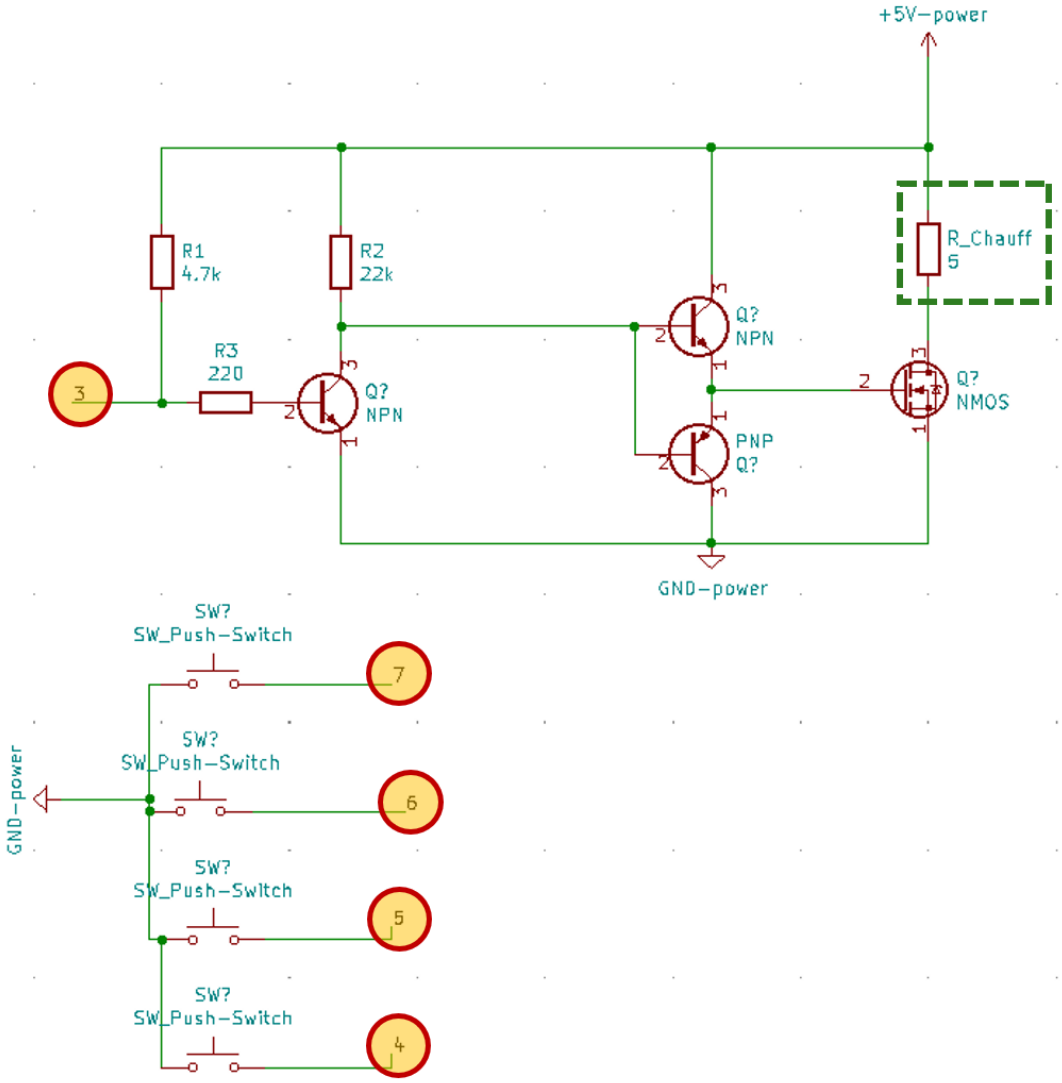

FIGURE 4 – Schéma électronique associé

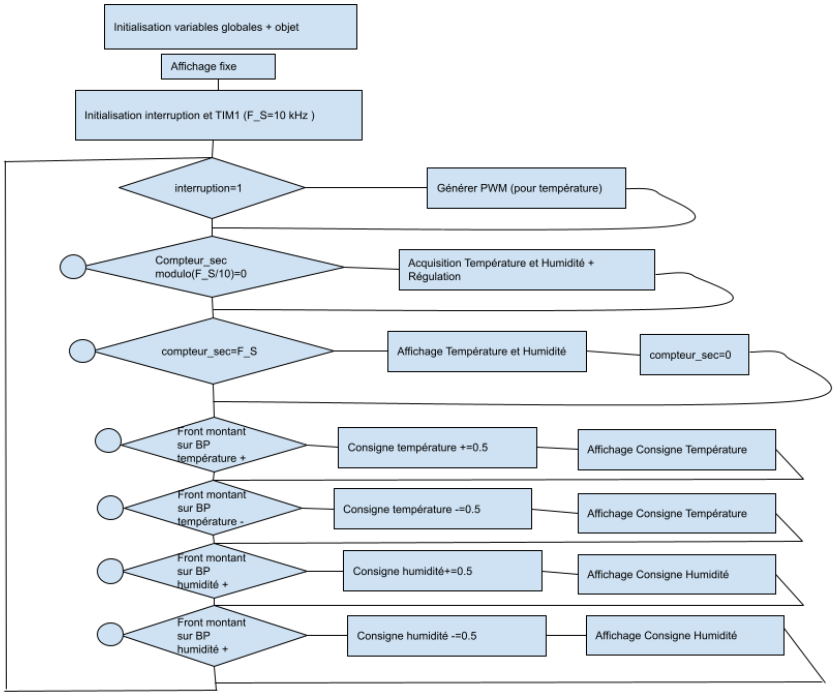

FIGURE 5 – Diagramme descriptif du système