

BUREAU D'ÉTUDES : PROGRAMMATION ORIENTÉE OBJET

CONCEPTION ET RÉALISATION D'UN RÉVEIL NUMÉRIQUE

Compte-Rendu

Baptiste HERISSON

Lucas CHAMBAZ

Arthur NICOLA

4^{eme} Automatique et Électronique

Systèmes embarqués

mercredi 11 mai 2022

Table des matières

1	Introduction	2
2	Conception	2
2.1	Diagramme de cas d'utilisation	2
2.2	Diagramme de séquence	2
2.3	Diagramme de classe	3
3	Utilisation	3
4	Conclusion	4

1 Introduction

Lors de ce bureau d'étude nous avons choisi de nous lancer dans la conception d'un réveil numérique. En plus d'afficher l'heure, notre réveil affiche également la température ainsi que l'humidité ambiante. Ces données sont affichées sur un écran disposant d'un mode nuit et d'un mode jour en fonction de la luminosité de l'environnement.

Pour réaliser notre réveil nous allons utiliser le matériel suivant :

- Microcontrôleur (ESP8266 programmable en arduino)
- Capteur de luminosité
- Capteur de température et d'humidité (SHT11)
- Module RTC
- Buzzer actif arduino
- Ecran

Le code est disponible sur github https://github.com/P3PiT0/BE_C-. La version finale est dans "Projet/-Main".

2 Conception

2.1 Diagramme de cas d'utilisation

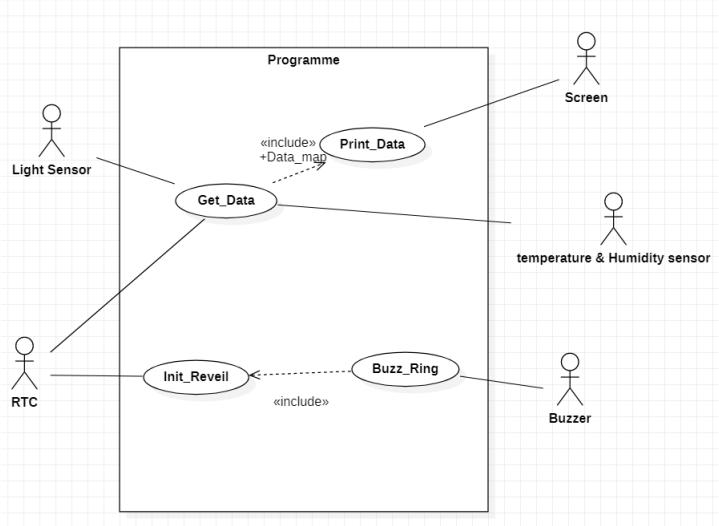


FIGURE 2.1 – Diagramme de cas d'utilisation

2.2 Diagramme de séquence

Le diagramme de séquence ci-dessous montre le fonctionnement global de notre réveil.

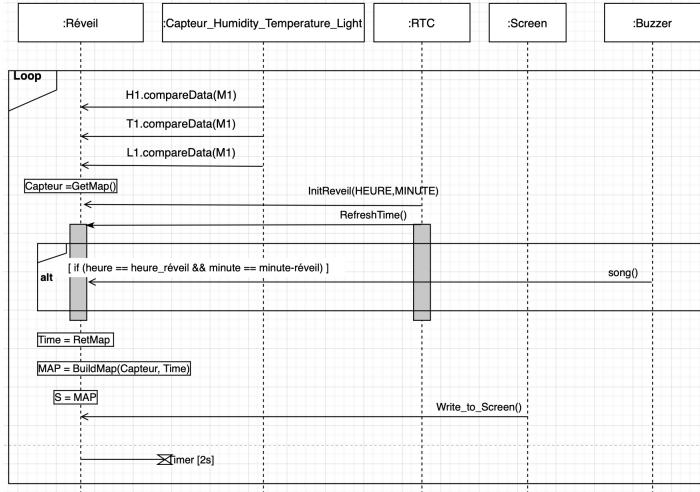


FIGURE 2.2 – Diagramme de séquence

2.3 Diagramme de classe

Ce diagramme de classe montre la répartition en classe de notre réveil ainsi que la construction du projet. Nous avons souhaité séparer les parties DATA, RTC et actionneurs (Buzzer et Ecran). Nous avons pu ainsi les traiter séparément (cf fichier.ino correspondant sur le github) et les assembler ensuite assez facilement au sein d'un même projet.

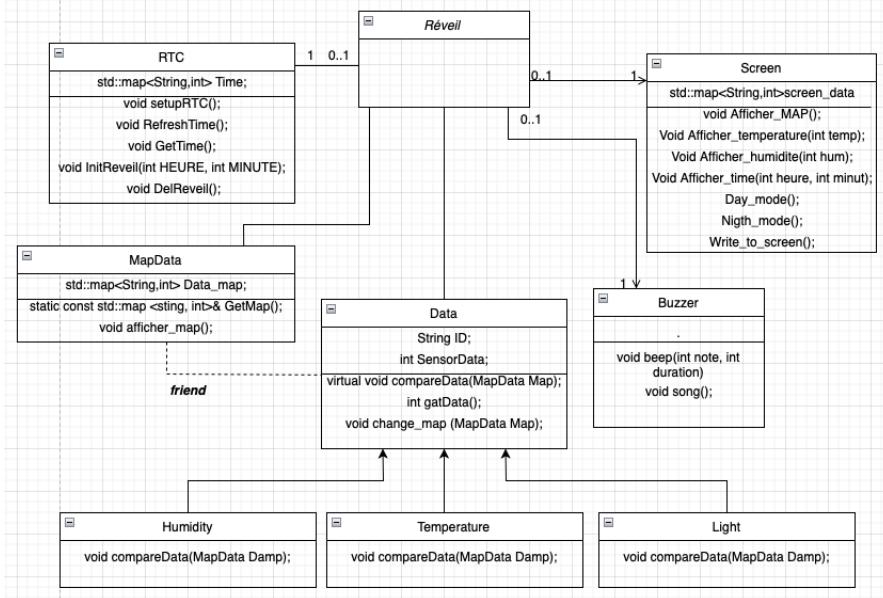


FIGURE 2.3 – Diagramme de classe

3 Utilisation

Pour utiliser notre réveil, il suffit de connecter les différents capteurs et actionneurs sur le GROVE BASE SHIELD de l'ESP8266 comme sur la figure 3.1. Le capteur de luminosité devra être sur le port analogique **A0**, le capteur de température et d'humidité devra être branché sur le port digital **D3**, l'écran et le module RTC sur les ports **I2C** et le buzzer sur le port digital **D7**.

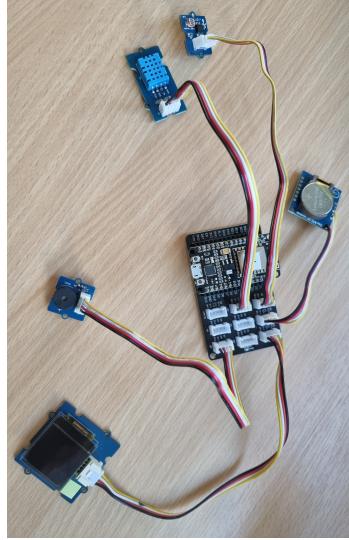


FIGURE 3.1 – Branchement des capteurs sur le GROVE BASE SHIELD de l'ESP8266

Une fois les branchements effectué il suffit de téléverser le code "**Projet**" du github sur la carte via le logiciel arduino. Projet contient un *Main.ino* ainsi que les différents headers et .cpp nécessaire au bon fonctionnement du réveil.

Si vous souhaitez modifier l'heure de réveil, vous devez la modifier directement dans le loop du main avec *T.InitReveil(19,8)*; (ici réveil à 19h08). Pour tester les différentes fonctionnalité, il suffit d'agir directement sur les capteurs (poser son doigt sur le capteur de température et d'humidité ou mettre le capteur de luminosité dans sa main à l'abris de la lumière).

Nous avons fait le choix de régler la période de notre loop à 2 secondes. Notre réveil n'affichant pas les secondes, cette période peut être augmentée mais la mise à jour de l'écran sera moins rapide. Vous pouvez changer ce laps de temps en modifiant dans Main la valeur dans *delay(valeur)* (1000 = 1 seconde) si vous souhaitez une actualisation plus ou moins régulière.

4 Conclusion

Nous avons pu lors de ce projet, utilisé la plupart des notions abordées lors des cours de C++ en créant plusieurs classes, en utilisant les mécanismes d'héritage (pour les capteurs), de surcharge d'opérateur (pour l'actualisation de la map de l'écran) et d'exception (pour l'activation du buzzer). Nous avons également utilisé la STL en se servant de maps pour transférer nos données entre les différentes classes.

Le projet était intéressant, autour d'un objet certes peu original mais très utile dans la vie de tout les jours et nous avons pu réaliser le projet que nous envisagions grâce à la diversité des capteurs et actionneurs disponibles. La charge de travail était assez conséquente pour le peu de créneaux mais nous avons pu le terminer en dehors des créneaux. Nous voulions ajouter un écran permettant à l'aide d'un bouton de régler l'heure du déclenchement du réveil et d'arrêt mais nous n'avons pas eu le temps car nous avons rencontrés quelques difficultés. Notamment pour la prise en main de l'IDE arduino avec la prise en charge des exceptions et des constructeurs (nous n'avons pas utilisé systématiquement de constructeurs mais des méthodes équivalentes pour palier à ce problème) mais également au niveau matériel avec notamment un manque de port I2C sur le SHIELD. Nous avons donc dû revoir notre conception en changeant de capteur de température et d'humidité.