MIKHIN Nikita
FRANCÉS Marcos

Bureau d'Etudes C++ : Objets Connectés

Problématique

Nous avons eu l'idée de créer une application qui permette de surveiller l'état d'une plante à la maison. L'objectif de notre projet est de relever les données d'humidité, de température, et de luminosité, afin de les stocker dans la carte et de les afficher sur un écran. De plus, lorsque ces valeurs ne rentrent pas dans un intervalle de confort pour la plante, un buzzer se déclenche pour prévenir l'utilisateur des mauvaises conditions atmosphériques pour la plante en question.

Pour accomplir cette mission, nous allons utiliser deux capteurs : un senseur de température et d'humidité (DHT22/AM2302) et un senseur de luminosité (Grove-GL5528). En ce qui concerne les actionneurs, il y en a deux : une alarme (Grove-Buzzer) et un écran de 1,12'' (Grove-OLED).

Bibliothèque en C++ pour la manipulation des composants

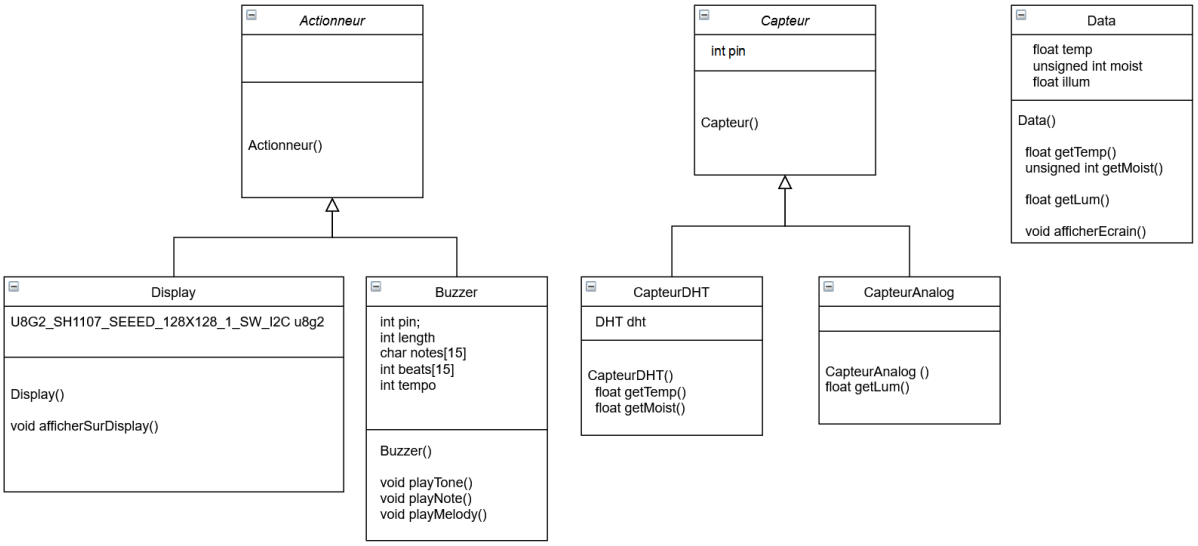
La première classe que nous avons créée est la classe *data*. Elle va contenir les trois informations relatives à un relevé de mesures à un instant t. Il y a temp pour la température, moist pour l'humidité et illum pour la luminosité. Ces trois attributs possèdent chacun leur méthode get. La classe contient aussi une méthode *CalculMoy*, qui va rendre la moyenne des trois valeurs pour une liste de points *data* donnée.

Ensuite, on a créé la classe capteur, qui sera divisée en deux sous-classes, *CapteurDHT* et *CapteurAnalog*. Elles vont être utilisées respectivement la première pour le capteur de température & humidité et la deuxième pour le capteur de luminosité. On a choisi cette structuration car elle va permettre au programme de maintenir un certain degré de généricité, dans le cas où il est utilisé dans le futur (passage des pins par arguments au constructeur par exemple). Pour les actionneurs, nous en avons deux types différents : le buzzer et l'écran LCD. Ces deux types vont être des classes qui héritent de la classe *Actionneur*.

Les librairies des capteurs et des actionneurs fournissent plein de fonctionnalités différentes, donc pour faciliter l'utilisation, on a décidé d'extraire les seules fonctions qui vont nous servir dans les classes correspondantes (par exemple, *getTemp()* de la classe CapteurDHT). Ainsi, nous avons ajouté un attribut qui est l'instance de la classe provenant des librairies (par exemple, l'attribut *dht* est une instance de la classe *DHT*, qui représente un capteur en particulier).

En ce qui concerne les bornes de qualité de la plante, elles ont des valeurs assez générales afin de faciliter les tests. Cependant, elles peuvent être éditées facilement, parce que ces six variables ont été déclarées en tant que constantes au début du programme principal.

Diagramme de classe



Utilisation de la STL

Dans notre projet, la STL est utilisée pour pouvoir stocker temporairement un tableau de données, provenant de la classe *data*. Pour cela, on a besoin d'inclure la classe *list* dans notre projet. Notre objectif étant de surveiller le bon état de l'environnement d'une plante, on a décidé de faire une liste de 10 échantillons, pris à chaque cinq secondes. Après cette étape, on réalise une moyenne sur cette liste pour vérifier que ces trois valeurs ne dépassent pas un certain seuil.

On a aussi pensé aux problèmes de stockage qui pourraient apparaître si cette liste continue à se remplir sans arrêt. Puisque le ESP8266 n'a pas une mémoire infinie, on a choisi d'effacer à chaque itération le contenu de la liste de données. Quand même ça aurait pu être intéressant de les stocker avec un attribut : *time*, qui se rajouterait dans la classe Data, pour avoir un suivi de l'évolution des conditions de la plante. Comme autre option, au lieu d'envoyer les données vers le moniteur série, on peut envoyer ces données à l'ordinateur via la connexion UART (*Serial.write()*) pour les récupérer sur l'ordinateur et les stocker.

Conclusion

Globalement, le programme répond aux exigences que l'on s'est imposé. Nous avons rencontré quelques problèmes concernant la librairie *DHT* (Seeedstudio) du senseur de température et d'humidité. On a dû prendre une librairie différente car celle-ci ne fonctionnait pas correctement. En ce qui concerne une possible amélioration, originellement la carte allait être connectée à un réseau Wi-Fi, afin de transmettre sans fil le flux de données de type *data*. Cependant, le temps limité consacré à ce BE ne nous a pas permis d'explorer suffisamment cette possibilité.