BUREAU D'ÉTUDES DE C++ :
HappySeed, un projet de monitoring de plante d’intérieur

Enseignant : Raphaël Deau
Xiaohu Zhang - Alice Forsans - 4AE-SE2
31 mai 2020

1. Présentation du projet :

A) Principe général :

Nous souhaitons optimiser l'environnement d'une plante d'intérieur en jouant sur différents paramètres pour qu'elle puisse s'épanouir au mieux. Nous supposons qu'elle se situe dans une pièce où se trouve une lampe, un système d'arrosage, un chauffage, un ventilateur et une fenêtre commandables.

B) Matériel utilisé :

Notre système est donc constitué de 4 capteurs :

- un détecteur d'humidité (placé dans la terre du pot)
- un détecteur de température
- un détecteur de luminosité
- un détecteur du taux de CO2

Il comporte également 6 actionneurs :

- un servo-moteur permettant d'actionner l'ouverture / la fermeture de la fenêtre
- un système de chauffage, modélisé par un servo-moteur également
- un ventilateur, modélisé par un servo-moteur
- une lampe possédant un état haut et un état bas (allumée / éteinte)
- un système d'arrosage à deux états (robinet ouvert / fermé)
- un écran relié par I2C à la carte, qui permet à l'utilisateur d'accéder aux mesures des capteurs.

Nous voulons donc contrôler le taux d'humidité de la terre de la plante : si elle n'a pas assez d'eau, l'arrosage se met en route ; et si elle en a trop, la fenêtre s'ouvre pour accélérer l'évaporation de l'eau. De même, si la température est trop faible, un chauffage se met en route ; si elle est trop élevée, le ventilateur démarre. Si la plante est plongée dans l'obscurité, une lampe s'allume pour lui permettre de bénéficier d'un éclairage optimal.

La récupération des valeurs des capteurs se fait par les fonctions AnalogRead et DigitalRead de la Board.
Nous supposons que les mesures sont prises à une fréquence très faible : une mesure toutes les 30 min/1h par ex. En effet, la plante n'a pas besoin de conditions optimales en permanence, elle possède une certaine souplesse.

La fonction HappySeed :: main() permet de faire le lien entre les informations reçues des capteurs, l'état de la plante, et les consignes à transmettre aux actionneurs.

Les consignes sont ensuite transmises aux actionneurs de la Board, par analogWrite et digitalWrite. En dehors de la lampe et du système d'arrosage, qui fonctionnent de manière "binaire" (allumés/éteints), les autres actionneurs sont analogiques, et possèdent plusieurs niveaux de

consigne (par exemple, pour le ventilateur, la vitesse peut aller de 0 quand il est éteint, à 10). De plus, certains actionneurs modélisés par des moteurs sont caractérisés par leur vitesse (le chauffage, le ventilateur) tandis que d'autres sont davantage caractérisés par une position à atteindre (l'angle de la fenêtre). Nous utilisons donc des paramètres différents en fonction des cas.

Les fonctions run() des différents actionneurs permettent de modifier les paramètres d'environnement impactés. L'influence des actionneurs sur l'environnement est donc proportionnelle à la consigne : la température augmentera plus vite si le chauffage est à puissance maximale.

Etant donné la faible fréquence des mesures prises, les délais de mise en marche / arrêt des actionneurs (le temps d'ouverture ou de fermeture de la fenêtre, la mise en marche du système d'arrosage, du ventilateur ou du chauffage) sont négligeables. De même, on suppose que l'arrosage a une durée limitée dans le temps (5 minutes par exemple) au bout duquel il s'éteint automatiquement, jusqu'à la prochaine mesure du taux d'humidité, qui enclenchera (ou non) une deuxième "routine".

Pour illustrer la "résilience" de la plante, nous lui avons assigné une quantité d'énergie, qui augmente si elle est "à son aise" et qui se détériore si les conditions sont mauvaises trop longtemps. La plante meurt si son énergie est totalement épuisée. Les informations sur l'état de la plante sont stockées dans la classe Plante.

Quant aux paramètres d'environnement (luminosité etc) ils sont stockés dans la classe Environnement, en tant que variables static, pour un accès plus simple.

C) Lancement de la simulation

Pour lancer le programme, il faut d'abord créer l'exécutable :

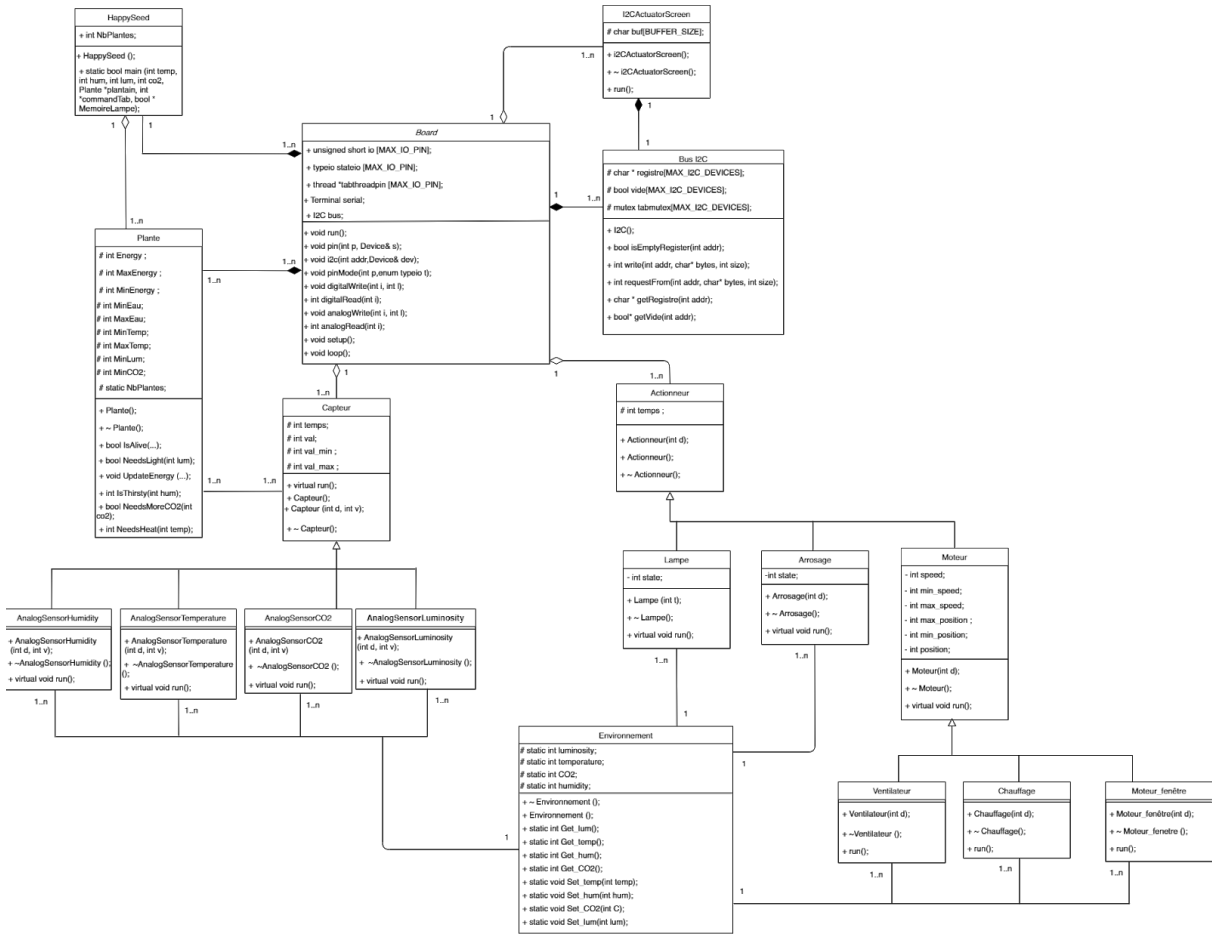
- modifier les valeurs d'environnement "originelles" en fonction du paramètre que l'on veut tester. Pour cela, ouvrir environnement.cpp et modifier les valeurs des paramètres, aux lignes 11 à 14. Enregistrer le fichier.
- ouvrir le shell, et se placer dans le sous-dossier src
- taper dans le shell **../compile_linux.sh**

Une fois la compilation faite, taper **./arduino**

2. Notre diagramme de classe :

Vous trouverez ci-dessous notre diagramme de classe. Pour plus de lisibilité, vous pouvez également l'ouvrir en cliquant sur le lien suivant :

https://drive.google.com/file/d/1wESSjNVtp-w1KXKPs0_AFWIpGysOtGhZ/view?usp=sharing

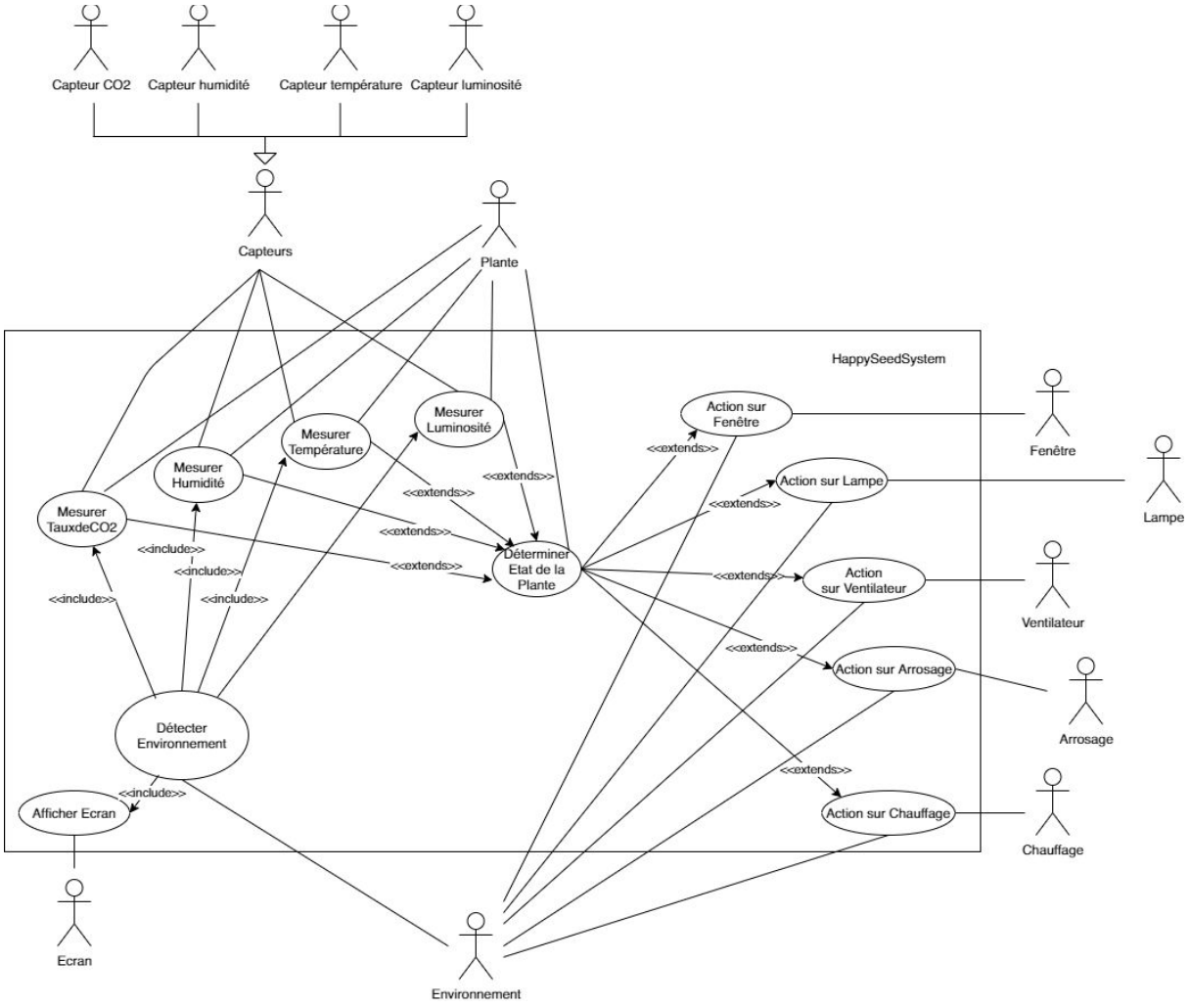


Nous constatons que la classe Environnement est avant tout liée aux capteurs et aux actionneurs, avec lesquels elle échange des informations. Capteurs et actionneurs sont ensuite reliés à notre carte Arduino, qui fait également l'interface avec l'écran I2C, et les parties logicielles (classe Plante et HappySeed). Nous avons supposé que ces parties logicielles, une fois chargées sur la carte, étaient "fortement liées" à cette carte : si elle est détruite, le code présent dessus l'est aussi.

3. Schéma de fonctionnement matériel et logiciel :

Afin de clarifier le fonctionnement de notre projet, nous avons mis au point un diagramme "use case", que vous pourrez trouver ci-dessous, et au lien suivant :

https://drive.google.com/file/d/1EuZmXo_xfpDlSN8w3E3q1eYOQOAOyron/view?usp=sharing



4. Retours sur le projet :

A) Objectifs atteints

Nous avons réussi à simuler notre système tel que nous le souhaitions. Les informations d'humidité, de luminosité, de taux de CO2 et de température sont bien affichées sur l'écran. De plus, on peut bien voir comment les actionneurs améliorent l'environnement si la situation n'est pas optimale pour la plante. On peut aussi observer le processus de perte et de récupération d'énergie des plantes.

B) Problèmes rencontrés

Les principaux problèmes rencontrés ont été les suivants :
Du point de vue de la compréhension, nous avions au départ défini beaucoup de fonctions qui ne nous ont pas été utiles, parce qu'elles étaient déjà implémentées d'une autre manière (par ex, une fonction Get_Val() dans la classe Capteur était en fait redondante avec analogRead() de la classe Board). Il nous a fallu un peu de temps pour vraiment comprendre le fonctionnement de l'architecture implémentée. De même, il a fallu réfléchir à la manière de transmettre les

informations d'une classe / d'une méthode à l'autre, sans avoir à instancier des objets. Pour cela, nous avons pu utiliser des pointeurs, des variables de classe statiques ou encore des define.
Nous avions également défini un paramètre "taux de CO2" mais nous ne savions pas comment le mesurer, quel "appareil" permettrait de modifier ce taux dans la "vraie vie". Il rentrait néanmoins en compte dans le calcul de l'énergie de la plante.

Enfin, certaines difficultés étaient liées au contexte du projet "à distance" : la salle de TP virtuelle ne marchait pas très bien sur l'ordinateur de Xiaohu, qui n'a pas pu non plus installer de machine virtuelle. Cela limitait notre capacité de travail. Le simulateur présentait également quelques défauts, en particulier au niveau de certaines pins qui ne pouvaient être utilisées qu'en input ou qu'en output, sans qu'on comprenne vraiment pourquoi. Mais nous comprenons tout à fait que le simulateur a été développé dans l'urgence, il est donc normal qu'il possède encore quelques imperfections !

C) <u>Améliorations possibles du projet :</u>

La plupart des évolutions auxquelles nous pensons concernent le fait de rendre "l'environnement" simulé plus réaliste. Une piste par exemple serait de recréer des périodes de jour et de nuit, à l'aide d'une variable correspondant au temps qui s'écoule (qui permettrait d'afficher l'heure). Les périodes de jour et de nuit auraient alors des paramètres très différents de luminosité, température, humidité... De plus, comme la plante n'a pas besoin de lumière en permanence, il serait plus réaliste de modifier ses besoins en lumière en fonction de l'heure. De même, l'eau présente dans le pot est bue par la plante, donc sa quantité devrait diminuer naturellement.

Dans le même esprit, les différents actionneurs que nous avons mis au point devraient normalement influer sur plusieurs facteurs à la fois. Par exemple, la fenêtre ouverte a une influence sur l'humidité, mais aussi la luminosité et le taux de CO2. Une piste pourrait être de modifier les paramètres impactés par les actionneurs. Cependant, cela impliquerait de mettre au point des lois de contrôle plus poussées, pour éviter que le réglage d'un paramètre n'en dérègle un autre. Cela nous semblait ambitieux de mettre cela en place en si peu de temps.

Le principe de "l'énergie" accumulée par la plante n'est pas tout à fait réaliste non plus : dans notre simulation, l'impact sur la plante est le même quel que soit le problème (mauvaise température, sécheresse...) et quel que soit le "degré d'inconfort" ressenti par la plante. Pour une simulation réaliste, il faudrait que, au delà d'un certain écart à la zone de confort, la plante perde son énergie plus rapidement (situation dangereuse pour elle) ou meure immédiatement (elle brûle, gèle, meurt de soif...)

De plus, en fonction des espèces de plantes et de leur stade de développement, les conditions optimales (la meilleure humidité, température, luminosité) sont différentes. Créer une classe pour identifier les espèces végétales et leurs stades de croissance permettrait de rendre cette programmation plus générique.

Il serait également intéressant de permettre à une seule carte Arduino de surveiller plusieurs plantes en parallèle, chacune ayant ses paramètres. Cela impliquerait certainement de modifier certains

détails de l'architecture du projet, voire de créer des listes de plantes que l'on parcourait pour les gérer les unes après les autres

Enfin, une dernière amélioration possible serait de faire davantage interagir l'utilisateur. Pour le moment, celui-ci choisit les conditions initiales (en modifiant le code, il serait plus intéressant qu'il écrive les paramètres voulus comme arguments de ./arduino, ou dans un fichier texte), et voit la plante s'y adapter. Il pourrait être intéressant de lui permettre de modifier ces paramètres en cours de route, ce qui permettrait de simuler des événements météorologiques "imprévus", comme un orage, une sécheresse...

Il reste donc beaucoup de possibilités d'amélioration de notre projet, qu'il aurait été intéressant d'étoffer avec un peu plus de temps !