INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES TOULOUSE

4AE SE - Application du C++ au domaine des objets connectés

Bureau d'étude
Le poulailler connecté "Chicken ./run"

Auteurs:

Laurent Devez
Emilie Estival

May 28, 2020

1 Présentation du projet

Dans le cadre du bureau d'étude de langage C++, nous avons développé un projet de poulailler connecté. Celui-ci est monitoré à partir d'une carte Arduino, et comporte un ensemble de capteurs et d'actionneurs. Le but est d'automatiser la gestion d'un poulailler pour des particuliers, qui peut être parfois contraignante surtout lors de déplacements. Notre poulailler connecté "Chicken ./run" permet de réguler les niveaux d'eau et de grain, et gère l'ouverture et fermeture des portes en fonction de l'heure de la journée. Il est adaptatif, puisque l'utilisateur peut lui même rentrer le nombre de poules qu'il possède et peut personnaliser la taille de son poulailler. Plus de souci à se faire ! Tout est géré et les poules sont en sécurité.

2 Diagramme de classes

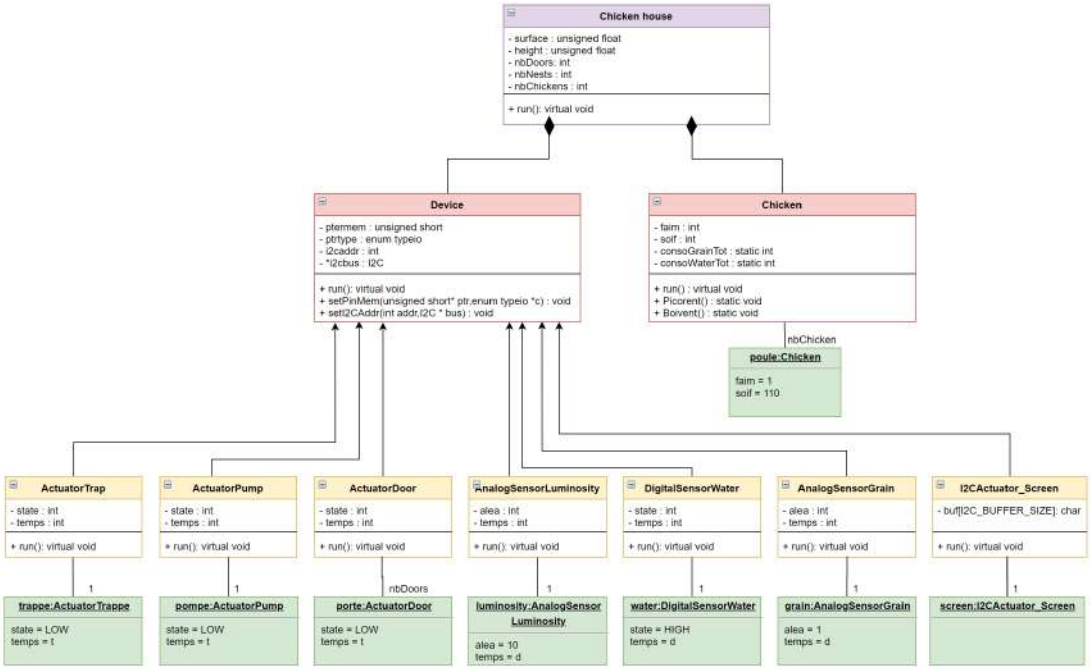

Figure 1: Diagramme de classes du système

Le diagramme de classes montre la structure interne du système *Chicken house*. Le poulailler nommé *Chicken house* sur la Figure 1 est composé de poules (*Chicken*) et d'un ensemble de capteurs et actionneurs (*Device*). Cependant, nous n'avons pas eu le temps d'implémenter cette classe. Tout ce qui suit est fonctionnel.

La classe *Chicken* reproduit le plus fidèlement possible le comportement d'une poule. Les attributs `faim` et `soif` permettent de personnaliser le comportement de chaque animal. Les attributs de classe `consoGrainTot` et `consoWaterTot` permettent de gérer de manière globale la consommation en eau et en grain. On peut ainsi savoir quand il est temps de remplir les mangeoires.

La classe *Device* est la classe mère de tous les capteurs et actionneurs. L'attribut `ptrmeme` fait le lien à l'emplacement mémoire des variables en lecture et en écriture.

Capteurs et actionneurs sont étroitement liés. A chaque capteur est associé un actionneur, comme suit. Pour plus de réalisme dans la simulation, nos capteurs analogiques présentent tous un aléa lors de la prise de mesure.

Le capteur de la classe *AnalogSensorGrain* permet d'accéder à la variable globale *grain_level*. Le capteur est un capteur à ultrasons, c'est un capteur analogique pour des questions de réalisation pratique. La *trappe* à grain s'ouvre (state HIGH) lorsque le niveau *grain_level* est inférieur à 10 pour remplir la mangeoire.

Le capteur de la classe *DigitalSensorWater* est similaire au capteur de grain sur le fonctionnement. C'est un capteur numérique, qui fonctionne en "tout ou rien", c'est à dire qu'un signal sera envoyé lorsque le niveau passe en dessous d'un certain seuil. Il permet d'accéder à la variable globale *water_level*. La *pompe* s'active si le niveau d'eau *water_level* est inférieur à 1000mL soit 1L (sachant que le contenant mesure 10L).

Le capteur de la classe *AnalogSensorLuminosity* permet d'accéder à la variable *luminosite_environnement*. C'est un capteur analogique, qui permet de relever la valeur exact de la luminosité grâce à une photo résistance à tout moment de la journée. Dans notre programme, la journée est simulée par un signal sinusoïdal qui fait varier la luminosité. La *porte* se ferme lorsque le capteur détecte la tombée du jour (luminosité $<$ 400LUX) et la *porte* s'ouvre au lever du jour (luminosité $>$ 450LUX).

Finalement, l'écran LCD *screen* de la classe $I2CActuator_Screen$ permet d'afficher périodiquement le niveau de grain et d'eau. Cela permet un monitoring extérieur simple et rapide.

3 Schéma de fonctionnement matériel et logiciel

Le schéma fonctionnel matériel présenté en Figure 2 permet d'identifier les différentes unités composant notre système. L'Arduino joue le rôle de l'unité de traitement, et interagit en I/O avec les capteurs et actionneurs. Ces derniers sont regroupés dans plusieurs FP en fonction de leurs caractéristiques.

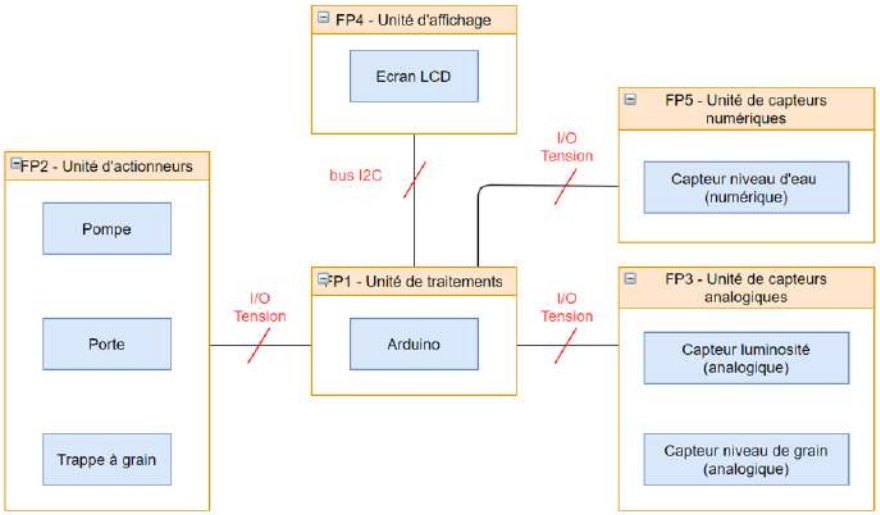

Figure 2: Schéma fonctionnel matériel

Par rapport au schéma fonctionnel matériel, le schéma fonctionnel logiciel présenté en Figure 3 montre en plus la partie simulateur environnement. L'environnement simulé permet de se rapprocher du fonctionnement du système en réel. On prend en considération les aléas pour les capteurs analogiques, et les attributs personnalisés pour la faim et la soif des poules. On gère aussi des variables globales d'environnement, pour monitorer le niveau total de grain et d'eau.

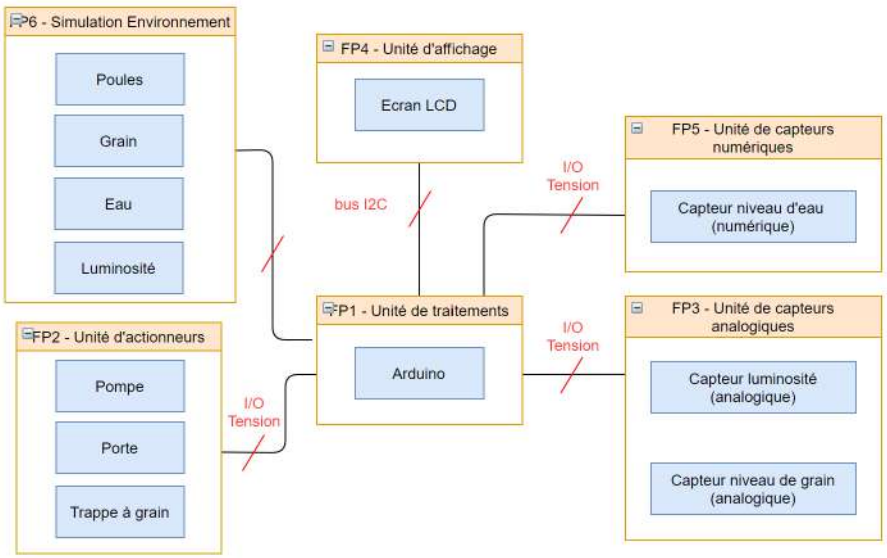

Figure 3: Schéma fonctionnel logiciel

4 Flowchart

La Figure 4 présente le schéma de fonctionnement du système dans son environnement. C'est plus ou moins une représentation graphique de notre "main" dans le code. Après la phase d'initialisation, on peut voir qu'on rentre dans une routine infinie. On récupère de manière périodique la valeur des variables globales de luminosité, de niveau de grain et de niveau d'eau. On réalisé ensuite un test sur chacune de ces variables, pour savoir comment agir sur l'environnement. Par exemple, si la luminosité est supérieure à 400LUX (coucher / lever su soleil), alors on va ouvrir la porte, si elle ne l'est pas déjà. Même fonctionnement pour la gestion du niveau de grain et d'eau.

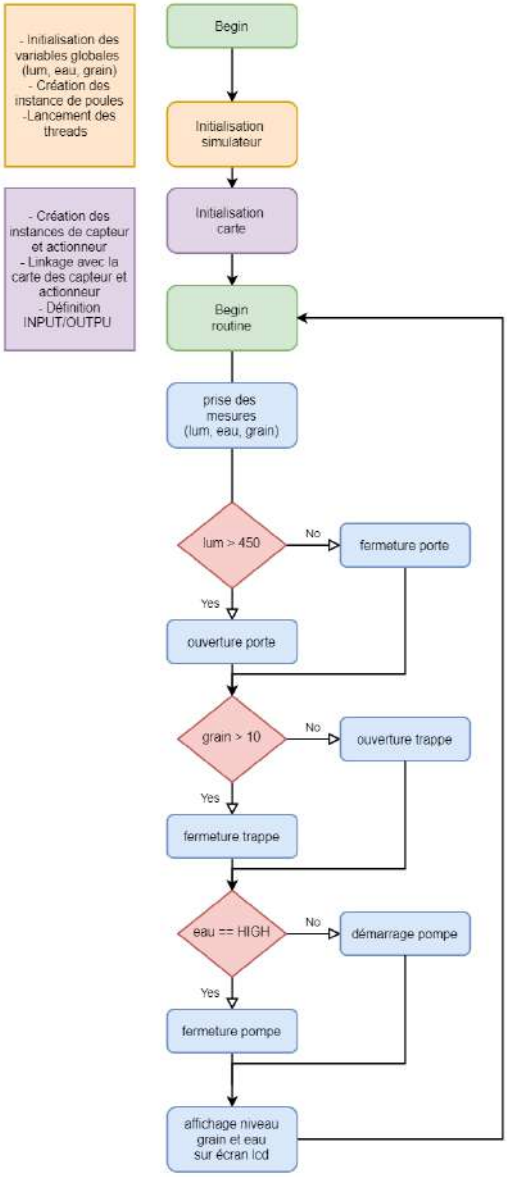

Figure 4: Schéma de fonctionnement du système

5 Architecture réelle envisagée

Nous allons détailler succinctement dans cette partie les capteurs que nous aurions utilisé si le projet avait pu être fait en réel.

5.1 Capteur de luminosité

Pour la mise en pratique du capteur de luminosité, nous aurions utilisé une simple photo-résistance, branchée sur une PIN INPUT de l'Arduino. On peut en trouver pour moins de 1€, et les performances fournies sont suffisantes. On ajoute une résistance de 10kOhm en série. On peut imaginer un montage très simple, comme présenté en Figure 5. Pour traduire la tension en intensité umineuse, on peut se servir du tableau en Figure 6.

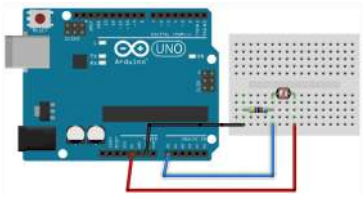

Figure 5: Exemple de montage avec photo-résistance pour mesure de luminosité

Éclairement lumineux	Exemple
120 000 lux	Soleil direct
20 000 lux	Ciel bleu à midi
10 000 - 25 000 lux	Ciel nuageux à midi
400 lux	Lever ou coucher du soleil, par temps clair
<200 lux	Ciel très nuageux
40 lux	Entièrement couvert
<1 lux	Nuage d'orage, cas exceptionnels

Figure 6: Tableau récapitulatif de la luminosité en fonction de la journée

5.2 Capteur de niveau d'eau

Pour contrôler le niveau d'eau, ce qui nous intéresse est un capteur type "tout ou rien". On ne s'intéresse pas particulièrement au niveau exact restant, on veut surtout déclencher l'allumage de la pompe pour remplissage de l'abreuvoir (Figure 7) quand le niveau devient bas. On peut par exemple utiliser un flotteur à contact magnétique, avec détection de niveau par flottaison, tel que présenté en Figure 8. On peut le trouver pour 5€ sur ce site.

Figure 7: Modèle d'abreuvoir, capacité 10L

Figure 8: Modèle de flotteur à contact magnétique

5.3 Capteur de niveau de grain

Pour contrôler le niveau de grain restant, on peut utiliser un capteur analogique. On peut très facilement mettre en place un capteur à ultrasons pour relever le niveau de grain, tel que présenté en Figure 9. Un tel capteur peut être trouvé pour moins de 3€ sur ce site par exemple. Pour ce qui est de la mise en place, le capteur peut être placé au dessus de la mangeoire comme présenté en Figure 10.

Figure 9: Modèle de mangeoire

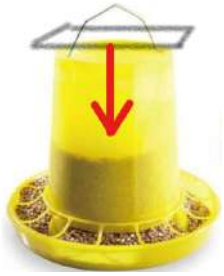

Figure 10: Exemple de montage sur mangeoire

6 Remarques et pistes d'amélioration

Nous aurions aimé aller encore plus loin dans le développement de notre poulailler connecté. Déjà, une mise en application en réel aurait été la bienvenue après cette phase de simulation fonctionnelle.

Ensuite, nous aurions aimé améliorer notre système en s'occupant également de la gestion de la ponte et des oeufs produits. Nous avions aussi pensé à mettre en place un système de régulation de la température. Un capteur d'humidité pour les réserves de grain aurait aussi été très pertinent à implémenter.

Pour ce qui est de l'alimentation en électricité, nous avions imaginé la mise en place de cellules photovoltaïques sur le toit du poulailler pour fournir en énergie la carte Arduino et les différents capteurs et actionneurs composant le système.

Enfin, une dernière piste d'amélioration aurait été de rendre notre projet plus "user-friendly" en créant par exemple une application téléphone de monitoring. On pourrait ainsi voir, depuis son smartphone, l'état de toutes les variables de niveau de grain et d'eau, ou encore la présence ou non d'oeufs dans le nid grâce à de la reconnaissance d'images. On peut imaginer placer sur les pattes des poules des bagues avec capteur RFID pour permettre une identification des individus dans le poulailler. Une analyse des données remontées sur l'application pourrait permettre des soins ciblés donnés aux poules: par exemple, se rendre compte quelle poule pond peu et pourquoi, pour remédier à ce problème (elle ne passe pas assez de temps dehors ? Son alimentation n'est pas suffisamment variée ?).

7 Conclusion

Finalement, nous avons réussi à développer notre projet de poulailler connecté ***Chicken ./run*** comme prévu. Nous avons réussi à rapidement prendre en main le simulateur Arduino.

Nous avons juste rencontré quelques difficultés pour mettre en place un environnement fonctionnel (gestion automatique de la luminosité en forme de sinusoïde pour simuler une journée par exemple). Finalement, cela a pu être très bien géré grâce à des threads qui tournent en parallèle de notre programme principal.

Un deuxième point de difficulté a été la mise en place d'une interaction avec cet environnement: par exemple, pour la gestion automatique de la consommation totale de grain et d'eau en fonction du nombre de poules, ayant chacune des attributs faim et soif différents. Au final, cette interaction extérieure a également pu être gérée grâce à des threads et des méthodes de classe appropriés.

Enfin, les pistes d'améliorations spécifiques à notre projet ont été largement discutées dans la Partie 5. Globalement, nous aurions aimé implémenter plus de capteurs pour améliorer notre produit final. Nous aurions aussi aimé se pencher sur la partie utilisateur, en allant par exemple jusqu'à la création d'une application smartphone pour faciliter le monitoring.