Rapport de projet
SmartBin

Table des matières

Présentation et motivations

Dans le cadre de la formation de prépa intégrée de Polytech'Nice Sophia, nous avons dû réaliser un projet à l'aide des connaissances acquises durant le premier semestre, notamment en électronique. Il a pour but de mettre à l'épreuve nos capacités d'organisation et de travail en équipe, et d'appliquer les cours suivis durant le premier semestre (électronique, mais aussi informatique, communication…).

Nous devions trouver un projet s'inscrivant dans les exigences des professeurs mais aussi dans nos exigences personnelles. Des discussions préliminaires à la date officielle de début du projet nous ont menées vers l'écologie et le développement durable (et dans la limite de nos capacités). Dans cette optique, nous avons décidé de réaliser une poubelle intelligente, qui aurait différentes fonctionnalités pour rendre plus pratique la vie des agents d'entretien.

La poubelle intelligente *SmartBin* comporte trois fonctionnalités principales :

- Mesure du poids de la poubelle en temps réel
- Mesure du taux de remplissage en temps réel
- Visualisation de ces caractéristiques sur un écran d'ordinateur et localisation GPS

Et deux fonctionnalités secondaires :

- Verrouillage de la poubelle lorsque le poids critique est atteint
- Déverrouillage manuel de la poubelle pour la vider

Dans un premier temps, nous présenterons chaque fonctionnalité séparément, d'un point de vue théorique et ensuite pratique, puis nous évoquerons la mise en commun matérielle du projet. Dans un second temps, nous ferons une rétrospective sur notre organisation ainsi que sur notre planning.

Fonctionnalités principales

I – Mesurer la masse du sac de déchets

Après quelques recherches sur le matériel utilisable avec Arduino, nous avons choisi d'utiliser la cellule de pesage circulaire TAS606 (cf fig. 1.a et 1.b), car elle semblait la plus adaptée à la forme d'une poubelle de forme cylindrique.

Figure 1.a : cellule de pesage circulaire

Figure 1.b : cellule de pesage (plan rapproché)

Il avait donc été prévu de coller cette cellule de pesage au fond de la poubelle, sur laquelle nous aurions placé un disque du même diamètre que la poubelle et de placer le sac sur ledit disque. Finalement, il nous a été mis à disposition un capteur de poids plus adapté (cf fig 2), avec un poids limite de 3 kg contre 50kg pour la cellule circulaire.

Ce dernier est plus pratique compte tenu des dimensions de la poubelle, et que pour un prototype simple nous n'aurons pas à mesurer des masses au-delà de 3 kg. Comme convenu, la cellule de pesage serait placée sous le sac, à l'intérieur de la poubelle, comme sur la figure 3.

Pour récupérer le poids du sac, la cellule de pesage agit comme une résistance, dans laquelle on fait passer un courant. Avec la déformation de la cellule sous le poids du sac, la valeur de la résistance est modifiée, changeant le courant et la tension. Pour récupérer les données (la variation de courant/tension étant infime), on utilise l'amplificateur de signal HX711 (cf fig 4). Enfin, pour lire les données sur Arduino, nous avons utilisé une librairie trouvée sur circuits4you.com, qui permet aussi d'effectuer le « tare » pour calibrer le poids du sac vide.

```
poids = scale.get_units();
scale.tare();  //Reset the scale to zero
```

Les deux lignes de code qui font marcher la majeure partie du système de pesage

Dès la réception du matériel, nous avons pu faire marcher l'ensemble amplificateur + cellule, et rapidement effectuer des tests. Cette partie n'a posé aucun problème majeur pouvant influencer notre projet.

Figure 2 : Cellule de pesage

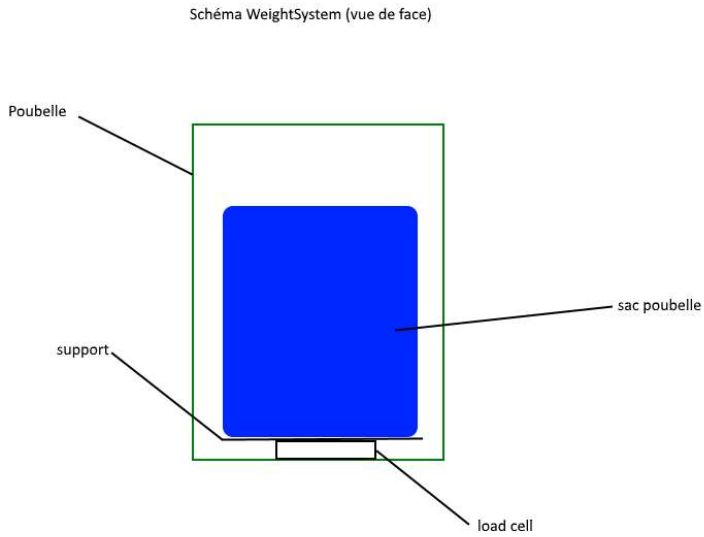


Figure 3 : Schéma du WeightSystem (vue de face)

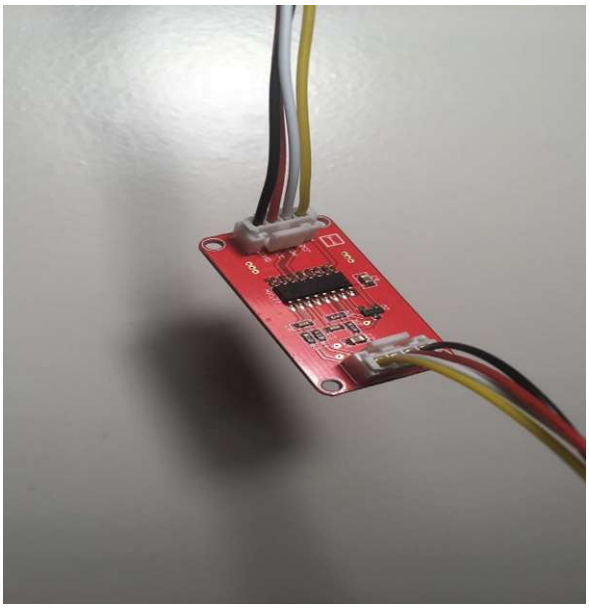


Figure 4 : amplificateur de signal HX711

II – Mesure du taux de remplissage

Pour mesurer le taux de remplissage, nous avons choisi d'utiliser un capteur de distance HC-SR04 (voir fig 1) fourni en cours, au premier semestre.

Figure 1 : capteur de distance

Ce dernier fonctionne par envoi et réception de micro-ondes, sur 5 volts (important pour comprendre les problèmes rencontrés plus tard). Un des deux haut-parleurs envoie une impulsion sonore, qui va « rebondir » au fond de la poubelle ou sur les déchets, et le deuxième haut-parleur réceptionne cette onde. A partir du temps de parcours on peut retrouver la distance parcourue par l'onde, et donc la distance entre le capteur et l'objet en question.

Le capteur est fixé sur la partie inférieure du couvercle, de façon à ce que quand la poubelle est fermée, le capteur soit face au fond de la poubelle (cf fig. 2).

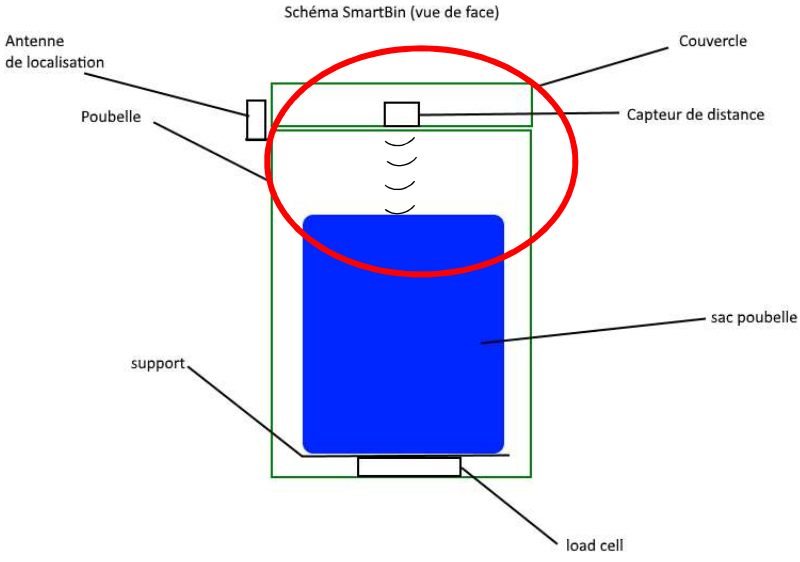


Figure 2 : Système de mesure du taux de remplissage (entouré en rouge)

On peut donc mesurer, en temps réel, la distance entre les déchets et le couvercle. Avec la hauteur totale de la poubelle, on peut calculer un pourcentage de remplissage, et ainsi définir un pourcentage maximal à atteindre avant d'envoyer une notification pour vider la poubelle.

Nous avons réutilisé la librairie NewPing et le code étudié en début d'année (rappel : tous les codes sont visualisables sur notre GitHub, onglet « doc/code ») pour récupérer la distance.

III – Envoi et visualisation des données

1) Présentation de l'UCA_Board

Une des caractéristiques majeures du projet est de pouvoir contrôler et vérifier les valeurs de poids et de remplissage. Si une de ces valeurs atteint un seuil défini, une alerte est envoyée à l'utilisateur et ce dernier reçoit les coordonnées GPS de la poubelle. Pour ce faire, nous avions d'abord pensé à utiliser un module GPS (cf fig 1), mais étant donné que la poubelle est censée être portable et autonome, il fallait un élément qui consommait moins de courant et qui puisse tenir au moins 2 jours avant que l'on doive recharger la batterie qui alimente le tout, ou changer les piles.

Avec l'aide d'un de nos encadrants (et professeur), nous avons eu accès à une UCA_Board, sur laquelle nous avons soudé une carte Arduino Mini Pro (cf fig 2). L'UCA_Board est une plaque, munie d'une antenne, qui permet de recevoir/envoyer des données. L'avantage par rapport au module GPS est sa consommation de courant, qui est très faible, ce qui rend son utilisation pertinente compte tenu des exigences du projet.

Figure 1 : module GPS

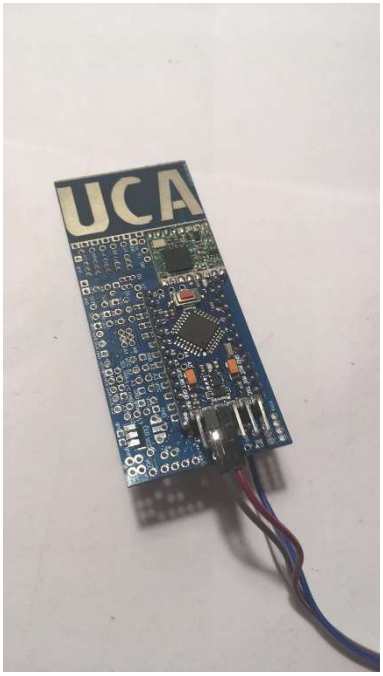


Figure 2 : Arduino Mini Pro soudée à l'UCA_Board

2) Envoi des données : principe

L'envoi des données se fait par un programme qui nous a été fourni, et qui par défaut envoie la valeur de la tension de la batterie de l'antenne par où l'UCA_Board envoie et reçoit les données. Le code est assez long est parfois un peu au-delà de nos compétences en informatique, mais nous avons réussi à trouver le procédé pour envoyer des valeurs. Les données sont récupérées sur The Things Network, et il est même possible de visualiser la localisation de la carte sur un plan, à partir des coordonnées envoyées par la carte.

3) Envoi des données : WeightSystem

Tout l'envoi se fait dans le code fourni avec l'UCA_Board, en envoyant 4 octets correspondant à des informations différentes que la console sur Internet Of Things va interpréter. Dans l'ordre sur la figure 3 : numéro assigné à l'affichage de la valeur (channel value), type de valeur (par exemple analogique), et la valeur elle-même.

```
unsigned char mydata[4];
mydata[0] = 0x2;  // 2nd Channel
mydata[1] = 0x2;  // Analog Value
mydata[2] = bat >> 8;
mydata[3] = bat & 0xFF;
```

Figure 3 : paquet de données à envoyer à la console IOT

Fonctionnalités secondaires

I – Verrouillage de la poubelle

1) Pourquoi ?

Il arrive souvent que des poubelles cassent, ou débordent, mais qu'elles continuent d'être remplies jusqu'à ce qu'elles ne puissent plus être utilisées. Pour pallier ce problème, nous avons décidé de mettre en place un système (assez naïf pour ce prototype) de verrouillage, pour empêcher quiconque de casser ou trop remplir la poubelle.

2) Comment ?

Pour commander le verrouillage, nous nous sommes procurés un servomoteur à rotation continue (cf fig 1), commandé simplement par notre carte Arduino. Pour fermer la poubelle, nous devions transformer le mouvement de rotation du servomoteur en un mouvement de translation, et avons donc décidé d'utiliser un système proche de la bielle-manivelle, en utilisant des pièces de Lego Technic (cf fig 2a, 2b). Malheureusement nous n'avons pas pu nous procurer ces pièces, et il va donc falloir s'en tenir à de la théorie.

Tout le système de verrouillage repose sur cette petite partie de code :

```
if ((poids >= poids_critique || distance<distance_critique) && ouvert){
  ouvert = false;
  analogWrite(servo,100);
  delay(500);
  analogWrite(servo,0);
}
```

Si le poids mesuré est supérieur au poids critique défini, alors on fait tourner le servomoteur pendant 500 millisecondes, assez pour faire translater la tige de façon à verrouiller. De façon analogue, si la distance avec les déchets est inférieure à une distance critique définie, alors on fait tourner le servo moteur. Nous avons aussi introduit une variable booléenne « ouvert », qui permet de ne pas faire tourner le moteur en continu. Si « ouvert » est vrai, cela signifie que la poubelle est virtuellement « ouverte ». Dans ce cas, on autorise la fermeture.

II – Déverrouillage de la poubelle

1) Pourquoi ?

Une fois que la poubelle est verrouillée, un agent d'entretien est notifié et est envoyé pour vider la poubelle. Ce dernier doit donc être la seule personne capable d'ouvrir la poubelle après verrouillage, il lui faut donc un « pass », ou quelque chose qui lui permette d'actionner l'ouverture de la poubelle.

2) Comment ?

Pour réaliser cette fonction, nous avons choisi d'utiliser un capteur à effet Hall (cf fig 1.a), qui fonctionne de pair avec un aimant (cf fig 1.b). On lit d'abord la tension sur la pin « OUT ». Au

repos, cette tension est d'environ 5V. Lorsqu'on passe le pôle Sud de l'aimant devant le capteur, la tension baisse jusqu'à environ 0V. Il suffit d'interpréter ce changement de tension pour faire tourner le moteur :

```
if (tension_hall<=tension_critique && ouvert==false) {
  ouvert = true;
  analogWrite(servo,100);
  delay(500);
  analogWrite(servo,0);
}
```

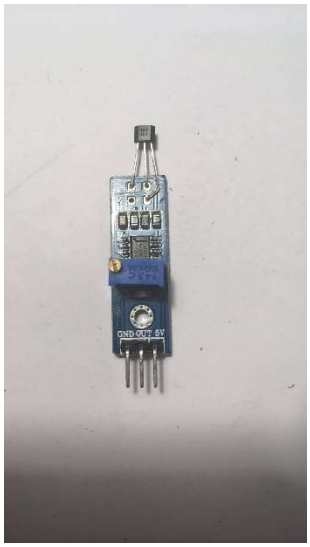
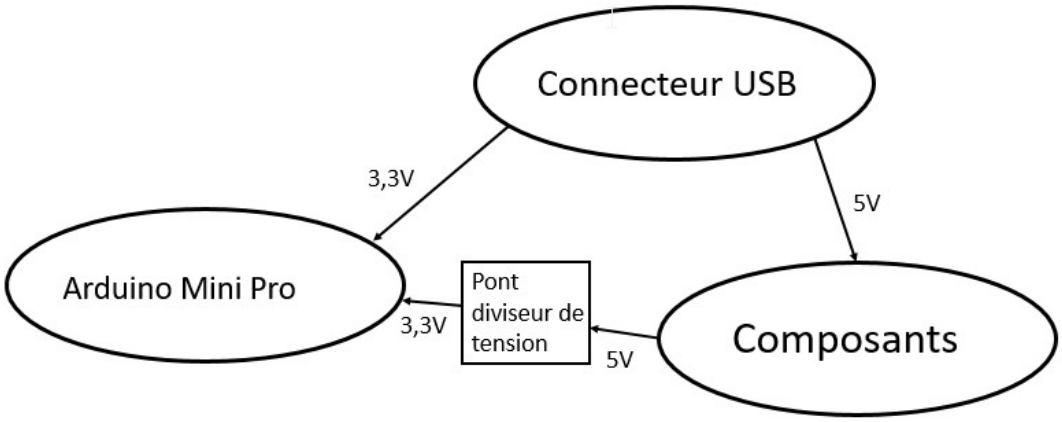


Figure 1.a : capteur à effet Hall

Figure 1.b : aimant

Mise en commun matérielle

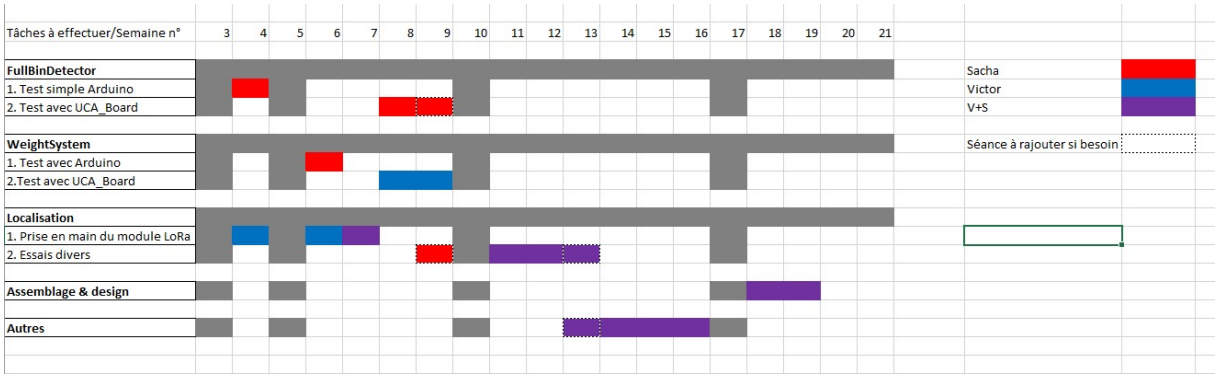
Pour finaliser le projet, nous avions dans l'optique de relier tous les capteurs à l'Arduino Mini Pro, récupérer les données de chaque composant et les envoyer sur la console d'IOT (Internet Of Things). Malheureusement, les capteurs fonctionnant sur 5V, ces derniers envoie leurs données sur 5V. Or, l'Arduino Mini Pro ne peut interpréter que des signaux en 3,3V. Il faut donc d'abord faire passer les données dans un pont diviseur de tension, comme expliqué sur le schéma suivant :



Après de nombreuses tentatives, il était difficile d'envoyer les deux valeurs de distance et de poids en même temps, sachant qu'il y a beaucoup de faux contacts en branchant sur la board, et qu'il suffit de bouger très peu un fil pour que tout plante. Dans l'ensemble, nous avons réussi à établir une communication entre IOT et l'Arduino, mais nous n'avons pas pu établir un flux continu d'échange de données à intervalles réguliers. Une solution à ce problème serait de souder les fils, mais étant donné que nous devons remettre le matériel à la fin, nous devons nous contenter de cette méthode.

Rétrospective

Au commencement du projet, nous avons dû établir un planning, pour gérer notre temps et nous organiser convenablement. A ce moment-là, nous n'avions évidemment aucune idée de comment fonctionnait le module LoRa, et par conséquent nous avons choisi de prendre une marge de plusieurs séances pour comprendre son fonctionnement. Nous nous sommes divisé le travail sur les deux premières fonctions principales, WeightSystem et FullBinDetector :



Comme nous avions pris une bonne marge pour chaque tâche, nous n'avons pas eu trop de contretemps. Le seul petit souci venait du fait que nous avions mal soudé la carte Arduino à la board, et nous avons mis une petite semaine pour comprendre l'origine du problème, et tout ressouder correctement.

Après les présentations de mi-parcours, nous avons actualisé notre planning, de la façon suivante :

A cette période, nous avions donc bien terminé la fonction de pesage, mais pas encore compris comment envoyer les données à la console IOT. De plus, le problème de tensions entre l'Arduino Mini et les capteurs (5V contre 3,3V) a complètement déréglé notre planning, car nous avons mis du temps à comprendre tous les petits problèmes qui empêchaient une bonne lecture des données (dans un premier temps, sur le moniteur série, puis sur la console IOT). Comme évoqué précédemment, nous avons dû faire face à quelques soucis de faux contacts, mais aussi à des problèmes liés à l'informatique, notamment une mauvaise communication, ou un mauvais affichage de données, entre autres. Nous avons

donc tenté de résoudre ces problèmes pendant plusieurs semaines, jusqu'à avoir pris en compte toutes les sources éventuelles d'erreurs.

Il a fallu d'abord être capable d'afficher les données envoyées par les trois capteurs (tension, poids, distance), ce que nous avons réussi à faire. Ensuite, il fallait réussir à lire les données sur la console IOT, toutes les 180 secondes (délai prédéfini). Par manque de temps, nous n'avons pas réussi à effectuer cette tâche. Aussi, toute la partie « Assemblage et design » n'a pas pu être exploitée pleinement, à cause des problèmes évoqués précédemment, notamment le fait que nous devions rendre une partie du matériel.

Conclusion

Pour élaborer ce projet, nous avons dû mélanger nos propres exigences avec celles de nos professeurs. Nous avons construit une démarche pour trouver des solutions à nos problèmes et pouvoir réaliser la majeure partie de nos idées. Le prototype que nous avons réalisé est loin d'être totalement terminé, pour des raisons déjà évoquées. Quelques améliorations possibles seraient par exemple de mettre le capteur de distance avec le pont de résistances dans une petite boîte de polystyrène, pour plus d'ergonomie et pour ne pas abîmer les composants. Il aurait aussi fallu bien fixer le servomoteur avec le système de verrouillage, et pourquoi pas ajouter des accessoires purement esthétiques, comme des LED qui s'allument pour indiquer le taux de remplissage.

Ce projet avait pour but de mettre à l'épreuve nos capacités d'organisation, de travail en équipe, mais aussi de restitution de connaissances acquises tout au long du premier semestre. Il a fallu faire preuve d'autonomie, car nous devions apprendre nous-même le fonctionnement de nouveaux composants, de nouvelles librairies pour coder. Il a servi d'avant-goût pour des projets plus importants que nous devrons réaliser plus tard, dans notre vie professionnelle comme dans notre vie étudiante.

Remerciements

Nous tenons à remercier tout particulièrement nos professeurs et encadrants de projet (l'ordre des remerciements n'a aucune pertinence ici). D'abord, Mr Fabien FERRERO, pour son aide, ses conseils, et évidemment pour l'UCA_Board qu'il a lui-même réalisée. Ensuite, Mr Pascal MASSON, pour avoir aidé à financer tous les projets, grâce à qui nous n'avons pas eu à sortir un seul centime de notre poche. Merci à eux pour avoir supporté les présentations de tous les projets.

Merci aussi aux autres étudiants qui nous ont aidés et qui se sont retenus de rire de notre projet de « Poubelle intelligente ».

Sources

Librairies :

- NewPing (playground.arduino.cc)
- HX711 (circuits4you.com)
- LowPower / arduin-lmic-custom (https://github.com/FabienFerrero)

Materiel:

- Ebay.com
- Banggood.com

Et de nombreuses autres sources qui nous ont permi de trouver nos idées et de comprendre le fonctionnement de composants.

Matériel utilisé

Arduino Mini Pro

UCA_Board

Module LoRa

Capteur de distance HC-SR04

Cellule de pesage RB-Phi-119

Amplificateur de signal HX711

Capteur à effet Hall

Aimant