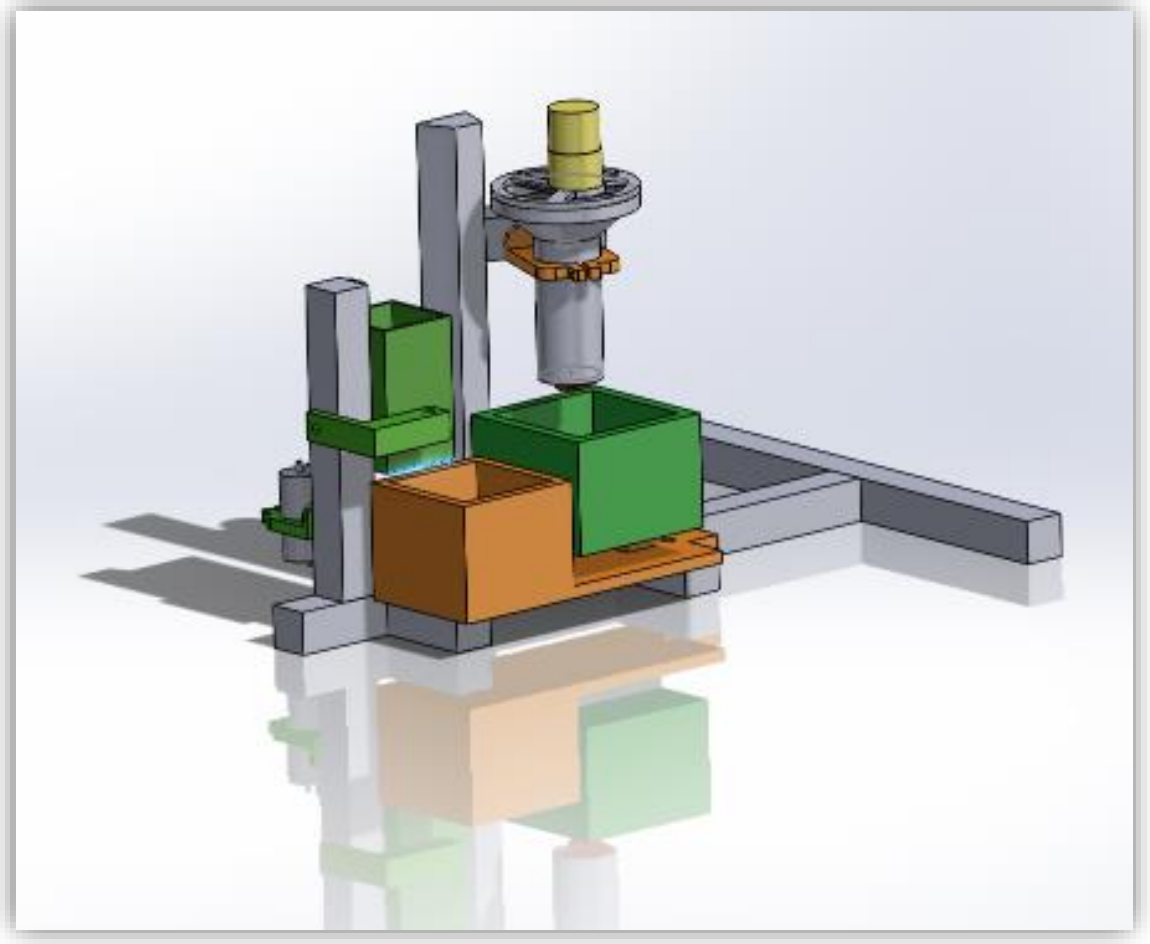
LYCEE DURZY

Sciences de l'ingénieur

Projet Interdisciplinaire

Distributeur de nourriture autonome pour animaux domestiques



Cecilia Koumba Koumba
Enzo Correddu
Anthony Cœur d'Acier
Anthony Duport

2017-2018

Sommaire

I. Partie commune

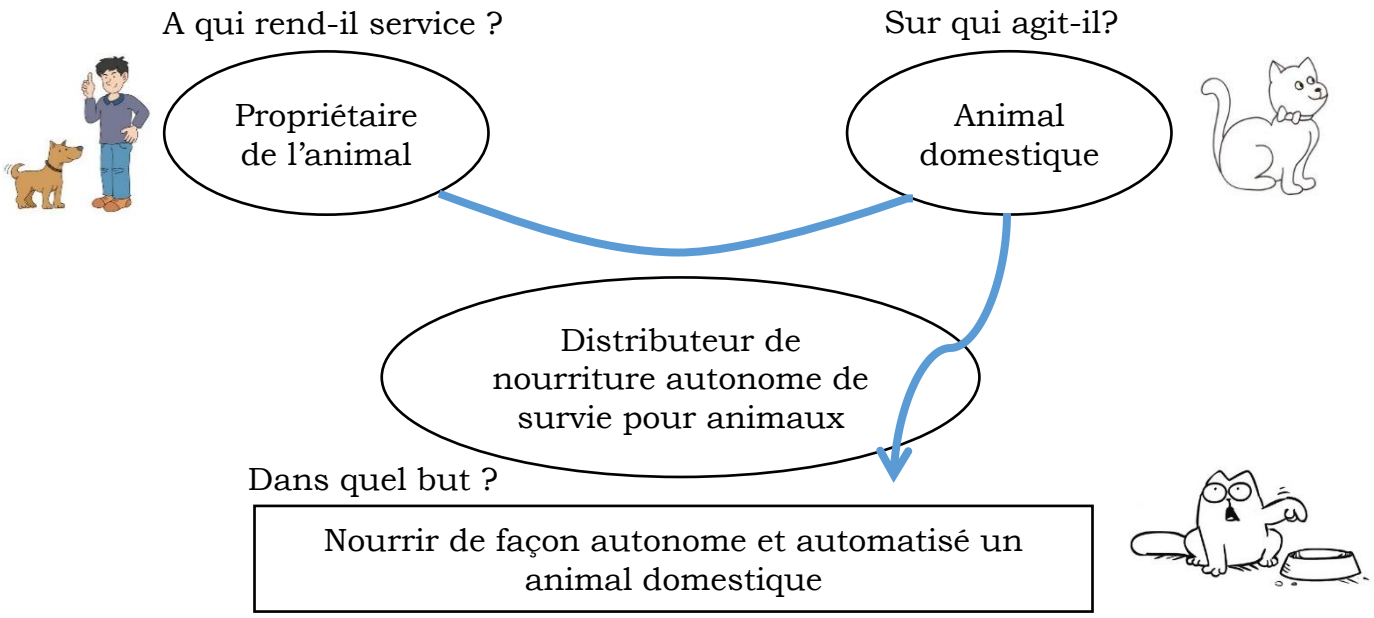
A. Description du projet

Le contexte a été défini tel que lorsque le propriétaire de l'animal est absent, il faut nourrir l'animal dans n'importe quel endroit. De plus, il faut le nourrir à des heures définies avec des quantités de nourriture, solide et liquide, adaptées à l'animal et pouvoir communiquer avec l'appareil à distance.

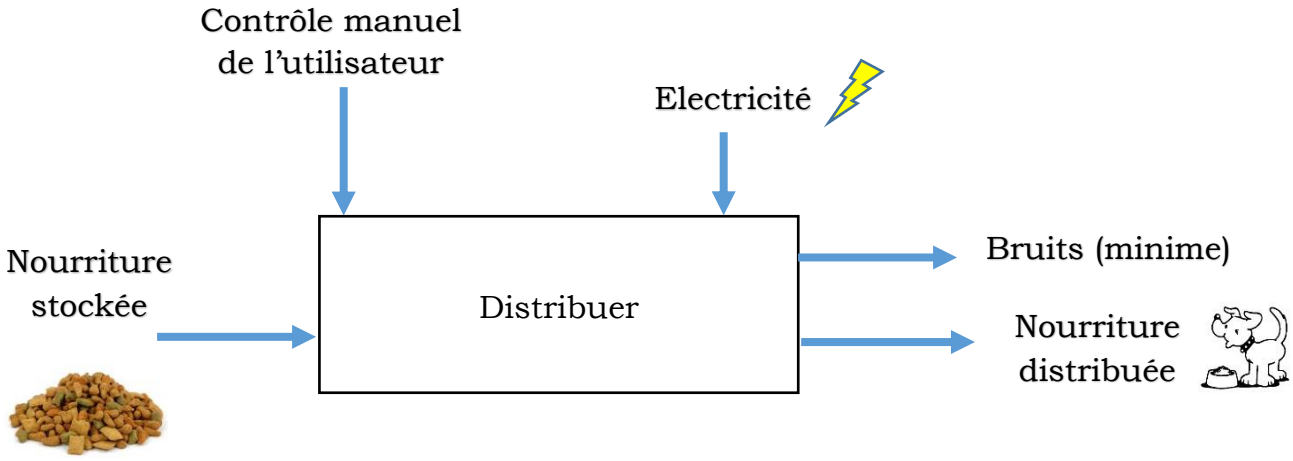
Ainsi, il faudra une nourriture convenable pour un animal domestique, le chat par exemple, quand il est sans son maître durant la journée.

B. Analyse fonctionnelle

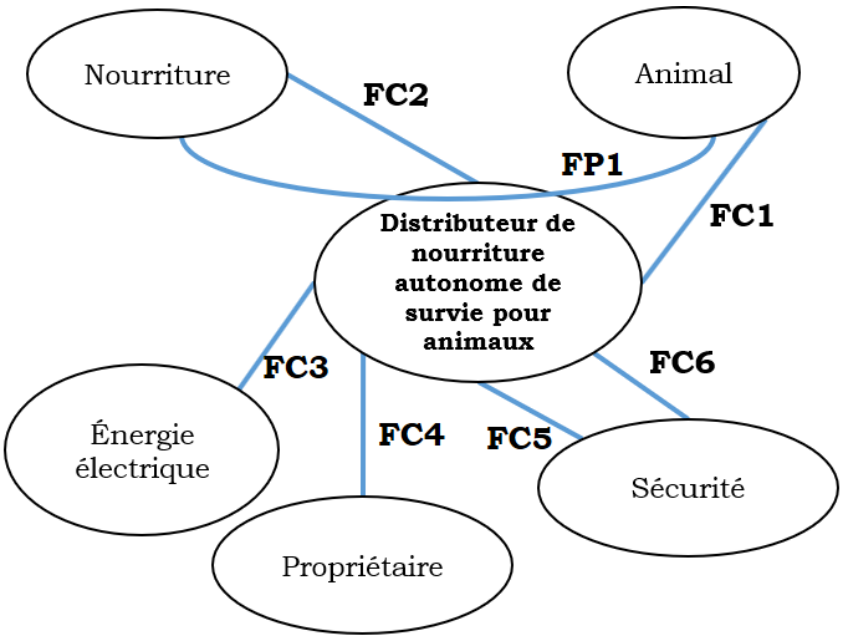
1. Expression du besoin



2. Actigramme de niveau A-0



3. Diagramme des intéracteurs



FP1 : Distribuer l'eau et distribuer les croquettes à intervalles réguliers.

FC1 : S'adapter à tous types d'animaux.

FC2: Permettre un stockage suffisant de nourriture.

FC3: Avoir une autonomie suffisante.

FC4: Etre facilement transportable.

FC5: Permettre la sécurité de l'animal.

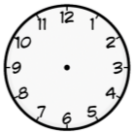
FC6: Etre au norme vis-à-vis de l'hygiène (croquettes).

4. Cahier des charges

- Pouvoir accepter **différents types de croquettes** pour animaux.
- Utiliser un récipient pour la nourriture **liquide et solide** et les stocker (Solide 2kg : Liquide 2L).
- Maintenir dans un réceptacle un **niveau d'eau constant** de 30 cl ± 3cl.

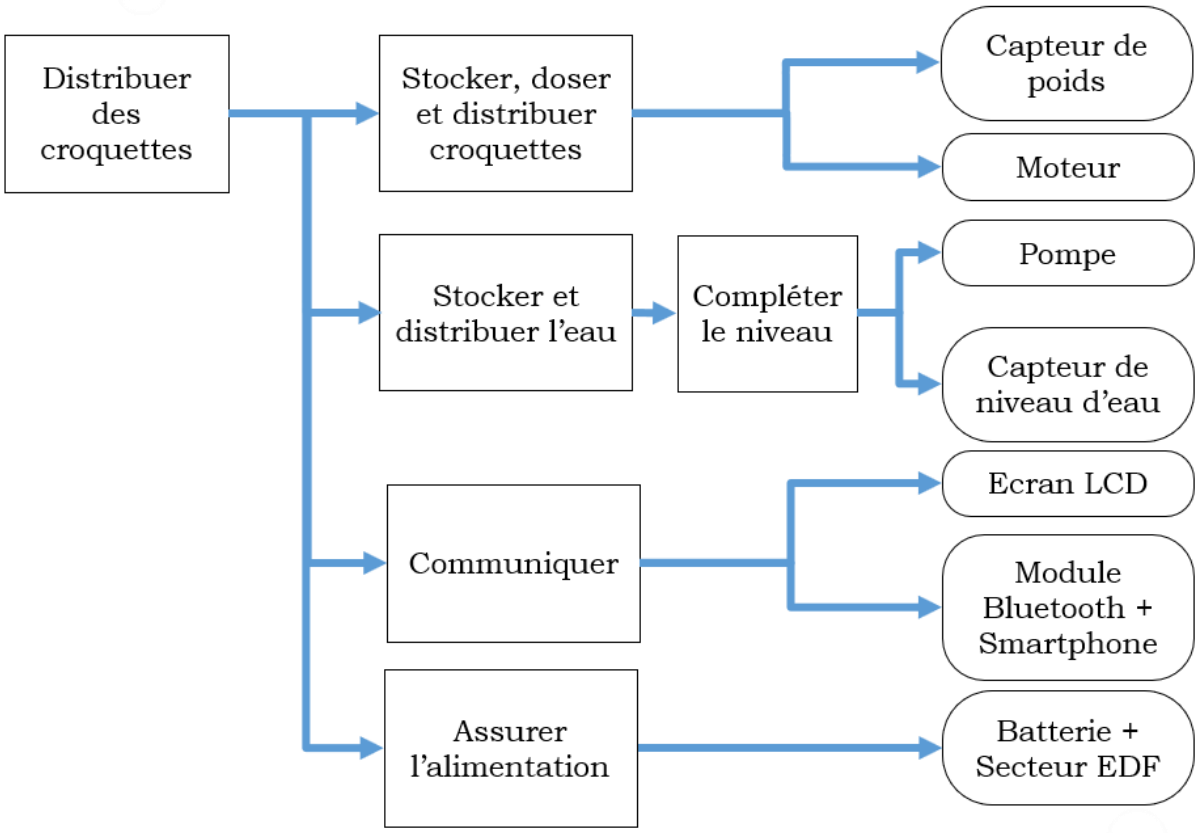
- Dose de croquette de **50 grammes par jour en trois versements** ±2g par versement.

- Être **compatible avec le secteur** comme source d'alimentation.
- Posséder **plusieurs choix prédéfinis de réglage** pour les intervalles de nourriture.

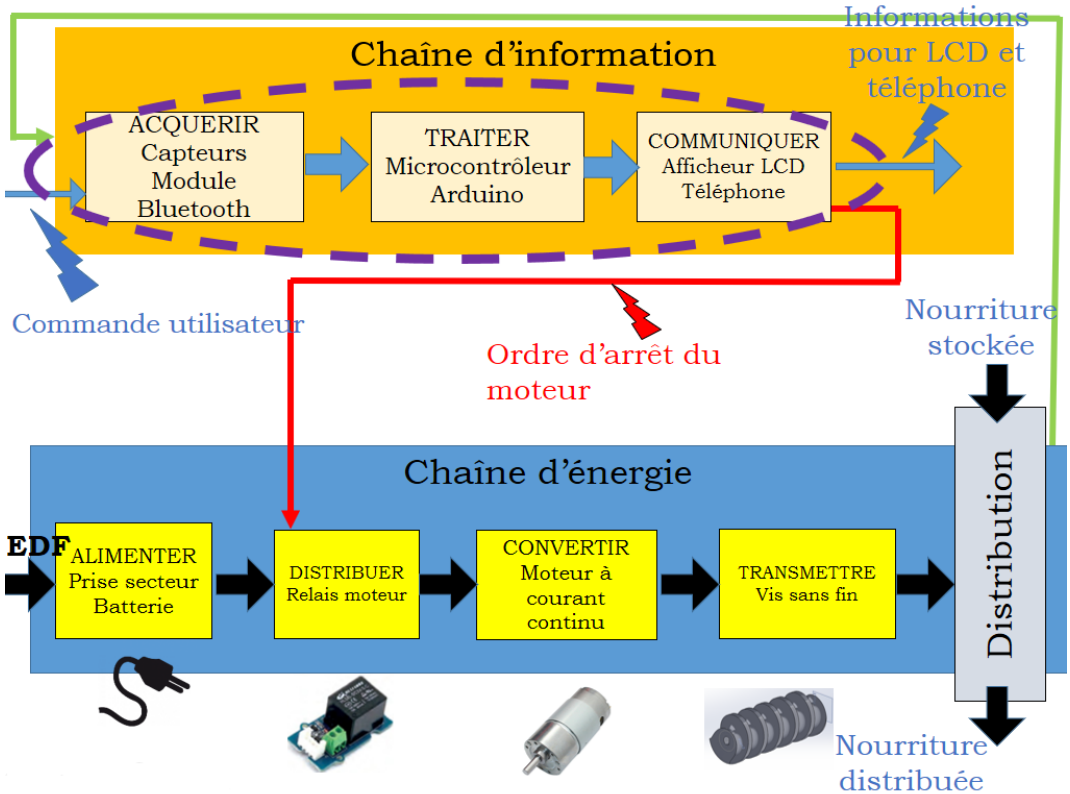


- Être **piloté par une carte électronique** programmable.
- Un **cycle de distribution** devra durer **maximum une minute**.
- Durée de **fonctionnement sans secteur** : maximum **4 heures**.
- L'ensemble devra **peser maximum 10kg.**
- Être **facilement transportable**.

5. Diagramme FAST



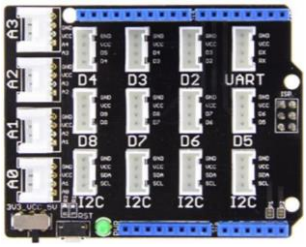
C. Chaîne d'information et chaîne d'énergie



D. Répartition des tâches

- Cécilia Kumba Kumba s'occupe de l'interface homme-machine via l'application sur smartphone (Android).
- Enzo Correddu s'occupe de la partie batterie avec notamment le dimensionnement de celle-ci.
- Anthony Cœur d'Acier s'occupe de la partie mécanique et de la maquette virtuelle.
- Anthony Duport, moi-même, m'occupe de la partie programmation.

II. Partie Individuel : Gérer et contrôler la distribution de nourriture



Ma partie concerne les fonctions ACQUERIR, TRAITER ET COMMUNIQUER de la chaîne d'information. Pour cela, j'ai choisi de programmer en ARDUINO car c'est un langage assez facile et pas compliqué à apprendre. J'ai donc dû apprendre à programmer en C donc en ARDUINO car je ne connaissais rien du tout. Pour cela, je suis allé me documenter sur internet. De plus, j'ai choisis de programmer avec des modules GROVE car c'est plus simple à raccorder à la ARDUINO et cela évite les interférences entre fils.

A. Raccordement au module GROVE

Pour satisfaire toutes les fonctions de notre projet, j'ai donc choisi ces modules :

Deux modules relais pour pouvoir piloter le moteur et la pompe. (1 « pompe » et 1 « moteur »)

Un capteur de niveau d'eau pour pouvoir récupérer le niveau du réceptacle d'eau.

Un capteur de poids pour pouvoir récupérer le poids de la gamelle de croquettes.

Un module Bluetooth pour pouvoir communiquer avec le smartphone.

Une horloge en temps réel (RTC) pour savoir l'heure et pouvoir savoir quand le moteur doit se déclencher.

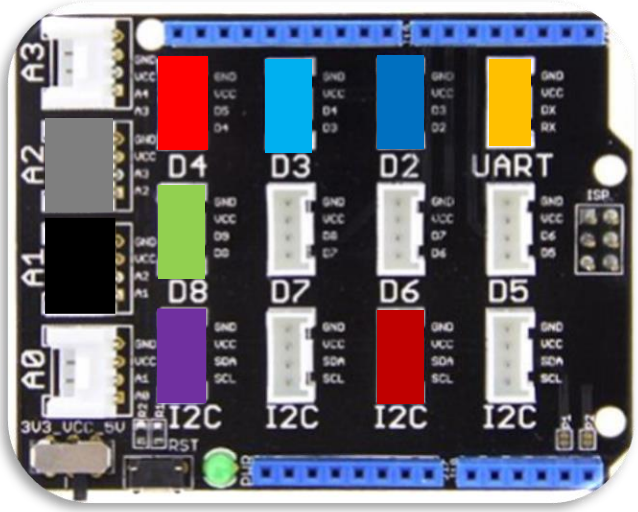
Un écran LCD pour pouvoir communiquer avec l'utilisateur

Deux boutons poussoirs pour pouvoir naviguer dans le menu du LCD. (1 « haut » et 1 « bas »)

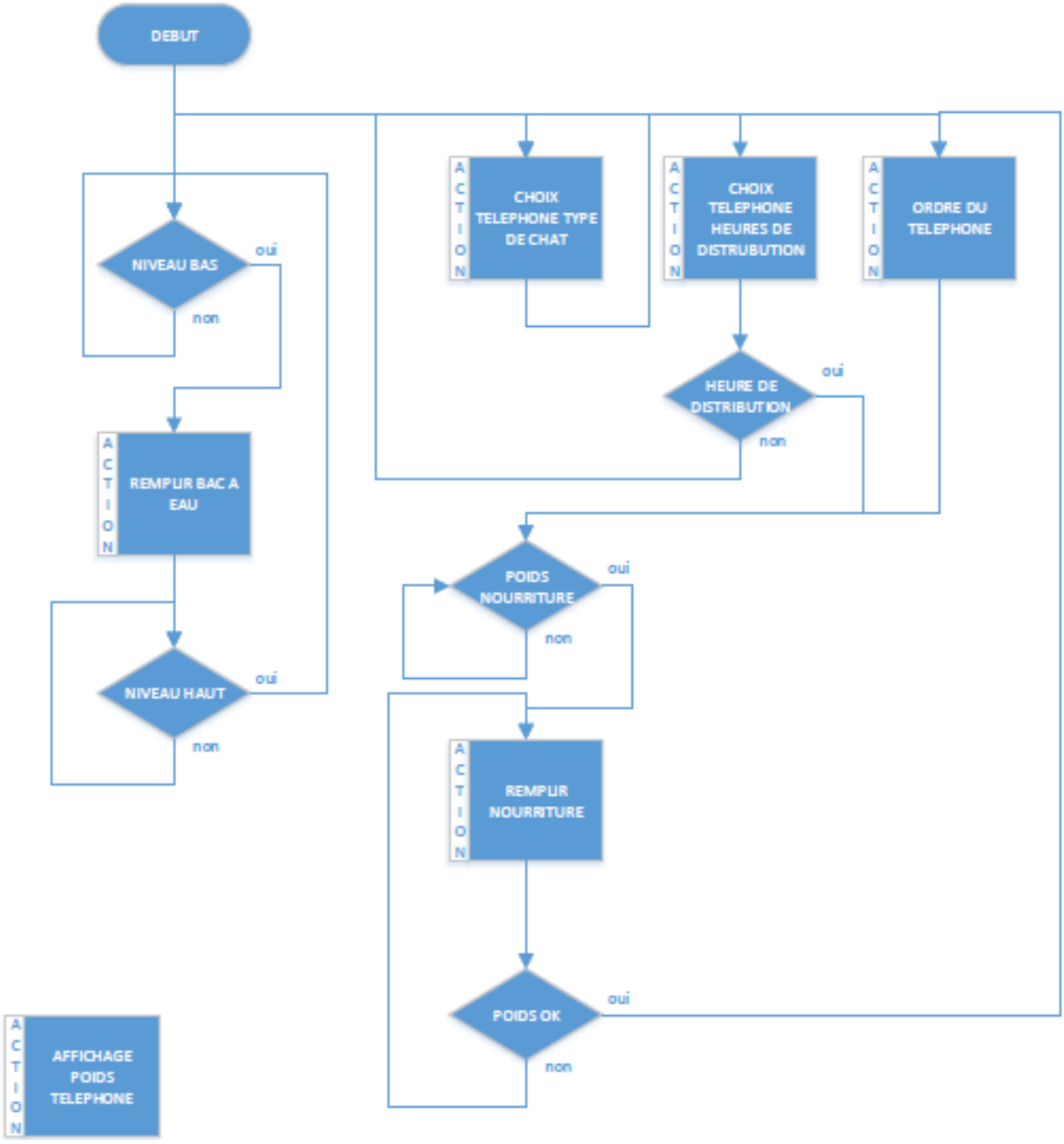
Un potentiomètre pour simuler la valeur de la batterie.

Ensuite, il faut relier les composantes au module GROVE. Pour cela, on va les relier comme ci-dessous :

- Le bouton poussoir « haut » sera relié sur le port **D4**.
- Le bouton poussoir « bas » sera relié sur le port **D8**.
- Le relais « moteur » sur le port **D2**.
- Le relais « pompe » sur le port **D3**.
- Le « RTC » sur un port **I2C**.
- L'écran LCD sur un port **I2C**.
- Le potentiomètre sur le port A0.
- Le capteur de niveau d'eau sur le port **A1**.
- Le capteur de poids sur le port **A2**.
- Le module Bluetooth sur le port **UART.**

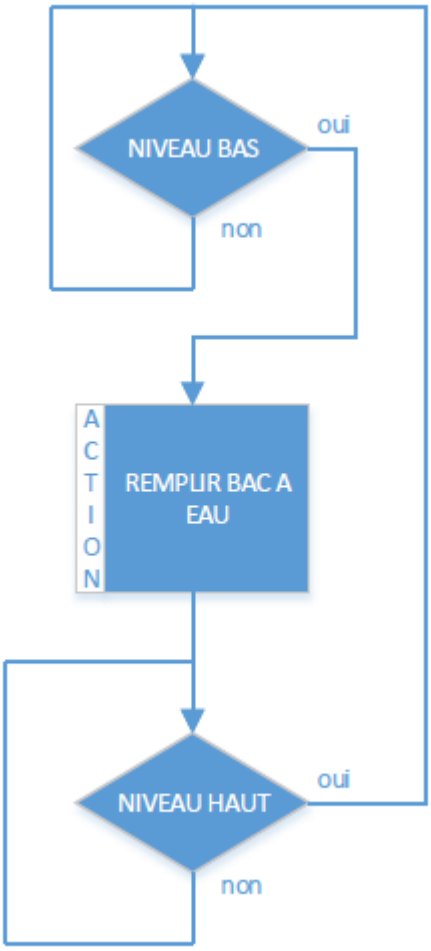


B. Logigramme de fonctionnement

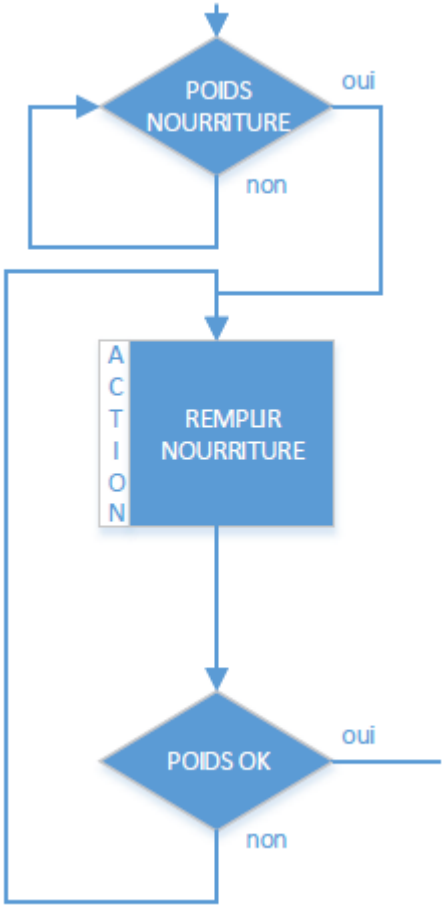


C. Problématique technique : Comment gérer avec précision les doses de croquettes et le volume d'eau?

Dans cette partie, nous allons nous intéresser à comment gérer avec précision les doses de croquettes et le volume d'eau? Cela se traduit par les fonctions suivantes du logigramme :



Remplissage de la gamelle à eau



Remplissage de la gamelle à croquettes

D. Programme et principe de fonctionnement

1. Pour le remplissage de l'eau :

```
int pompe = 3,//pompe sur le port D3
    niveauo = 1,// capteur de niveau sur le port A1
    valeurniv = 0, // valeur du capteur de niveau eau
    niveauEau = 200; //niveau d'eau limite

void setup() {
  pinMode (pompe, OUTPUT);
  //Définition du port arduino comme sortie
  digitalWrite(pompe, LOW);//la pompe est désactivée
}

void loop() {
  valeurniv = analogRead(niveauo);
  //lecture de la valeur du capteur de niveau
  if (valeurniv > niveauEau)  {
    Gamelle();
  }
}

void Gamelle() {
  valeurniv = analogRead(niveauo);
  //lecture de la valeur du capteur de niveau
  while (valeurniv > niveauEau) {
    //tant que la valeur souhaitée n'est pas atteinte
    valeurniv = analogRead(niveauo);
    //lecture de la valeur du capteur de niveau
    digitalWrite(pompe, HIGH);//on actionne la pompe
  }
  digitalWrite(pompe, LOW);//on éteint la pompe
  if (niveauEau == 500) {
    niveauEau = 150;
  }
  else {
    niveauEau += 10;
  }
}
```

2. Pour la distribution des portions

```
#include "HX711.h"
//librairie du capteur de poids
HX711 scale;//capteur de poids
float valCaptGamelle, valDistribBT;
//définition nombre flottant valeur du poids gamelle et
//valeur définit par l'utilisateur
int mot = 2; // moteur est sur la pin D2
int heures, minutes;
//définition heures et secondes du RTC comme entier
int HeurePredefini1, HeurePredefini2,
    HeurePredefini3;
//définition comme entier des variables heures pour les intervalles
boolean motOn = false, portion = false;
//défintion variables activation des heures comme booléen

void setup() {
  pinMode(mot, OUTPUT);
  //Défintion du port arduino comme sortie
  digitalWrite(mot, LOW);//moteur désactivé
  scale.begin(A2, A3);//Démarrage du capteur poids
  scale.set_scale(2280.f);
  //définition coeficient pour capteur du poids
  scale.tare();
  //remise à 0 valeur du capteur poids
}

void loop() {
  distrib();
  //on lance le sous-prog "distrib"
}

void distrib () {
  if (heures == HeurePredefini1 && minutes == 0
      || heures == HeurePredefini2 && minutes == 0
      || heures == HeurePredefini3 && minutes == 0
      || motOn == true) {
    //si les heures définies sont égales au temps présent ou
    //si l'utilisateur à choisis de faire une distribution
    valCaptGamelle = scale.get_units(10), 1;
    //lecture de la valeur du capteur de poids
    if (portion == true || motOn == true) {
      //si la portion est définie ou l'utilisateur décide
      //de distribuer une portion
      while (valCaptGamelle < valDistribBT) {
        //tant que la valeur souhaitée n'est pas atteinte
        valCaptGamelle = scale.get_units(10), 1;
        //lecture de la valeur du capteur de poids
        digitalWrite(mot, HIGH);
        //on actionne le moteur
      }
      digitalWrite(mot, LOW);//on éteint le moteur
    }
  }
  motOn = false;
}
```

3. Programme général

Pour des raisons de place et de longueur de mon programme, je ne l'ai pas mis dans mon dossier. Vous retrouvez le lien dans la dernière partie de mon dossier.

E. Définitions des heures et de la portion de croquettes à distribuer suivant le poids et le type de chats

1. Définitions des heures

Comme indiqué dans le cahier des charges, nous devons définir des heures de distribution pour les portions de croquettes.

Liste numéro	Distribution n°1	Distribution n°2	Distribution n°3
Liste n°1	6h00	13h00	18h00
Liste n°2	8h00	15h00	20h00
Liste n°3	10h00	17h00	22h00
Liste n°4	12h00	19h00	00h00

L'utilisateur, via l'application du smartphone, devra choisir une liste pour définir les intervalles des distributions.

2. Définition de la portion de croquettes suivant le poids et le type de chats

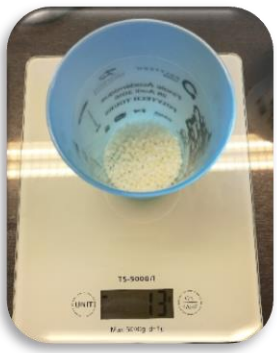
Poids du chat	2kg	3kg	4kg
Chat d'intérieur	**100g**	150g	200g
Chat actif	110g	165g	220g
Chat stérilisé	80g	120g	160g

Ensuite, il devra choisir les portions à donner par jour à son chat suivant son poids et son type. Pour l'exemple du chat d'intérieur de 2kg, il recevra 100g de croquettes par jour soit environ 33g par versement.

F. Etude des écarts

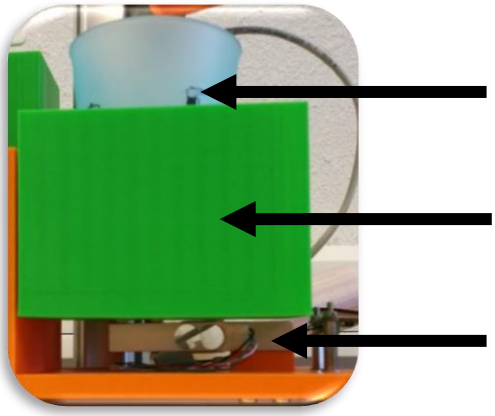
1. Entre une balance et le capteur de poids

a) Expérience



Pour savoir si le capteur de poids de notre maquette donne une vraie valeur du poids, j'ai pesé une quantité de « croquettes » sur une balance, et elle affiche 13g (voir image ci-contre). On part du principe que la balance affiche un poids précis.

Ensuite, j'ai versé le verre avec les « croquettes » dans un autre situé sur le capteur de poids. De plus, j'ai écrit un programme pour que la valeur du capteur de poids soit inscrite sur l'écran LCD.



On peut voir que la valeur qui s'affiche sur l'écran est 12g.

b) Conclusion

La valeur affichée sur la balance est 13g et celle sur le capteur de poids est 12g.

L'écart relatif est donc : $\frac{13-12}{13} = 0.08\ soit\ 8\%$

La précision du capteur est donc de 92%.

2. Le capteur de niveau d'eau

a) *Expérience n°1*

Le capteur de niveau d'eau fonctionne avec une résistance de 1MΩ. En fonction d'où le niveau d'eau est sur les traces du capteur, il va renvoyer une valeur entre 0 et 1024.

J'ai donc fait l'expérience de fixer mon capteur sur la paroi de la gamelle et de remplir petit à petit l'eau. Lorsque je n'ajoute plus d'eau, la valeur continue à varier alors que le niveau est stable.

b) *Conclusion*

On peut donc en conclure que le capteur de niveau d'eau n'est pas précis donc on ne pourra pas valider le cahier des charges qui disait qu'il fallait 15 ± 3 cl constamment dans le niveau de la gamme d'eau. Il faudrait plutôt mettre un capteur digital « tout ou rien », ce que fait aussi ce module, mais il faudra le brancher sur un port digital.

c) *Expérience n°2*

```
int pompe = 3,//pompe sur le port D3
    niveauo = 6;// capteur de niveau sur le port D6
boolean etatNivo; // valeur du capteur de niveau eau
                  // soit 0 soit 1

void setup() {
  pinMode(niveauo, INPUT), (pompe, OUTPUT);
  //Définition des ports Arduino
  digitalWrite(pompe, LOW);//la pompe est désactivée
}

void loop() {
  etatNivo = digitalRead(niveauo);
  //lecture de la valeur du capteur de niveau
  if (etatNivo == HIGH)  {
    Gamelle();
  }
}

void Gamelle() {
  etatNivo = digitalRead(niveauo);
  //lecture de la valeur du capteur de niveau
  while (etatNivo == HIGH) {
    //tant que la valeur n'est pas égale à 0
    etatNivo = digitalRead(niveauo);
    //lecture de la valeur du capteur de niveau
    digitalWrite(pompe, HIGH);//on actionne la pompe
  }
  digitalWrite(pompe, LOW);//on éteint la pompe
}
```

On branche donc notre capteur sur le port D6. Lorsque celui-ci sera en contact avec l'eau, il rend une valeur « LOW » et quand il ne se pas en contact avec l'eau, il rend une valeur « HIGH ». J'ai donc modifié mon programme pour changer la précédente fonction, ce qui donne :

Le capteur est donc plus précis.

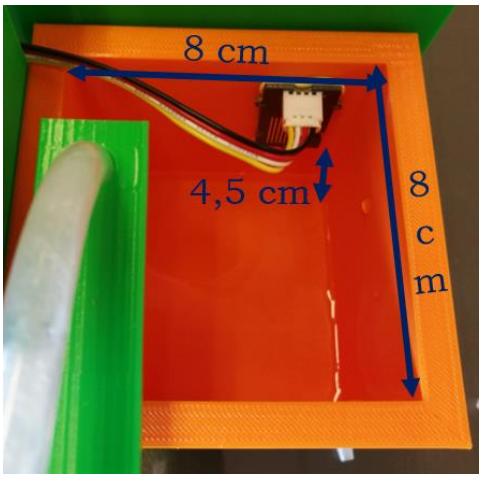
3. Mesure sur la maquette et constations des écarts

a) Mesure

Dans le cahier des charges, il est indiqué :

« Maintenir dans un réceptacle un niveau d'eau constant de 30 cl ± 3cl. »

Pour cela, j'ai mesuré les différentes valeurs de la gamelle (image ci-contre).



Ensuite, on va donc calculer le volume théorique d'eau **V_t**.

$V_t = 8 \times 8 \times 4{,}5 = 288\ cm^3$ Soit environ **30cl.**

Ensuite, à l'aide d'une seringue, j'ai pompé l'eau contenue dans la gamelle :

- 12 Seringues de 24 ml
- 22ml
- 11ml
- 5 ml
- 8 ml

En additionnant ces valeurs, on obtient le volume mesuré **V_m**.

$V_m = 12 \times 24 + 22 + 11 + 5 + 8 = 334\ ml$ Soit 33cl.

b) Constations des écarts

$$Ecart\ relatif : \frac{Vm-Vt}{Vt} \times 100 = \frac{33-30}{30} \times 100 = 10\%$$

L'écart relatif est donc de 10% et le cahier des charges est validé car la valeur mesuré est 33 cl et il autorise entre 27 et 33 cl.

III. Conclusion du projet

Pour conclure, ce projet m'a permis de découvrir l'environnement Arduino avec ses différents modules ainsi que le langage de programmation qui parfois peut être complexe. De plus, j'ai développé ma curiosité, et j'ai été confronté à des problèmes comme par exemple lorsque j'écrivais un code qui me semblait bon et que le résultat n'était pas le même que celui que j'attendais.

IV. Diffusion du projet

Retrouvez le programme en entier ainsi que l'application et les différents documents sur ce lien :

- https://github.com/antho4572/projet_SI