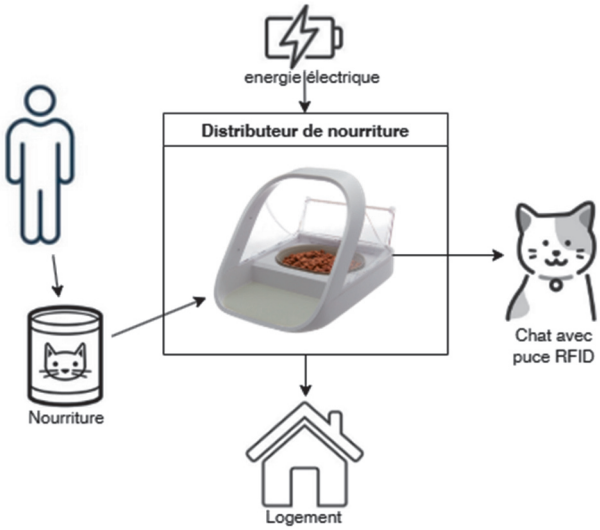
Dossier technique : distributeur de nourriture pour chat.

1. Découverte du produit et de la problématique technique

Découverte du produit :

La société SurePetcare commercialise le distributeur automatique de nourriture pour chat SureFeed. La nourriture est disposée dans un bac protégé par une trappe motorisée. Un détecteur de puce électronique (tag RFID), fixé au collier du chat, lui permet d'accéder à la nourriture.



Si un animal non enregistré passe à côté du distributeur, le couvercle ne s'ouvre pas. Le distributeur SureFeed garantit ainsi que la nourriture est consommée par le bon animal domestique. Une fois que l'animal a mangé et s'est éloigné du distributeur, le tag RFID n'est plus détecté et le couvercle se referme automatiquement.

La vidéo « Presentation_distributeur_de_nourriture.mp4 » illustre le fonctionnement du distributeur.

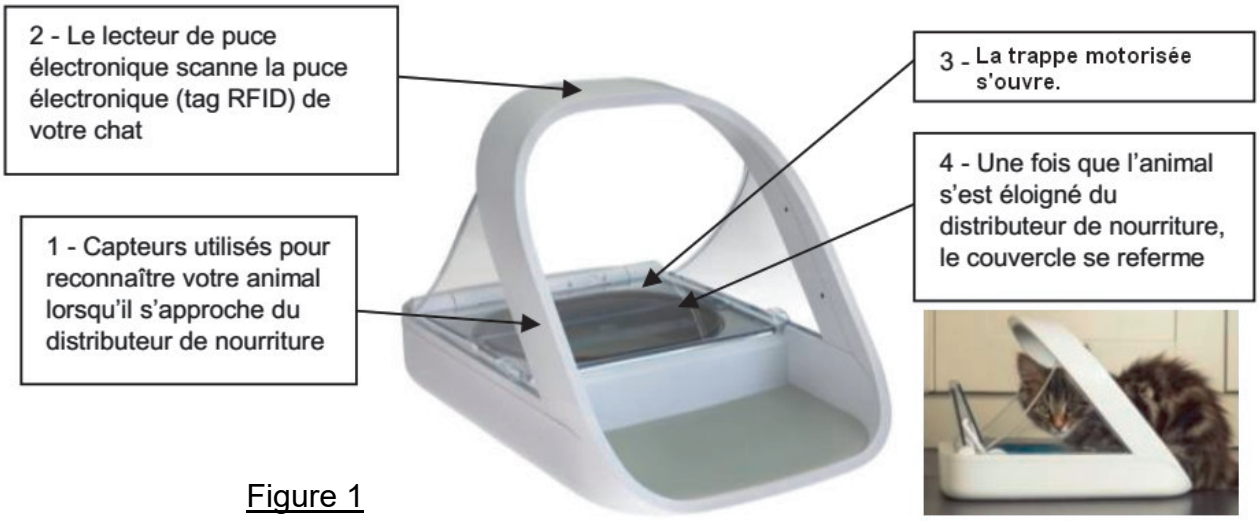


Figure 1

Caractéristiques du distributeur :

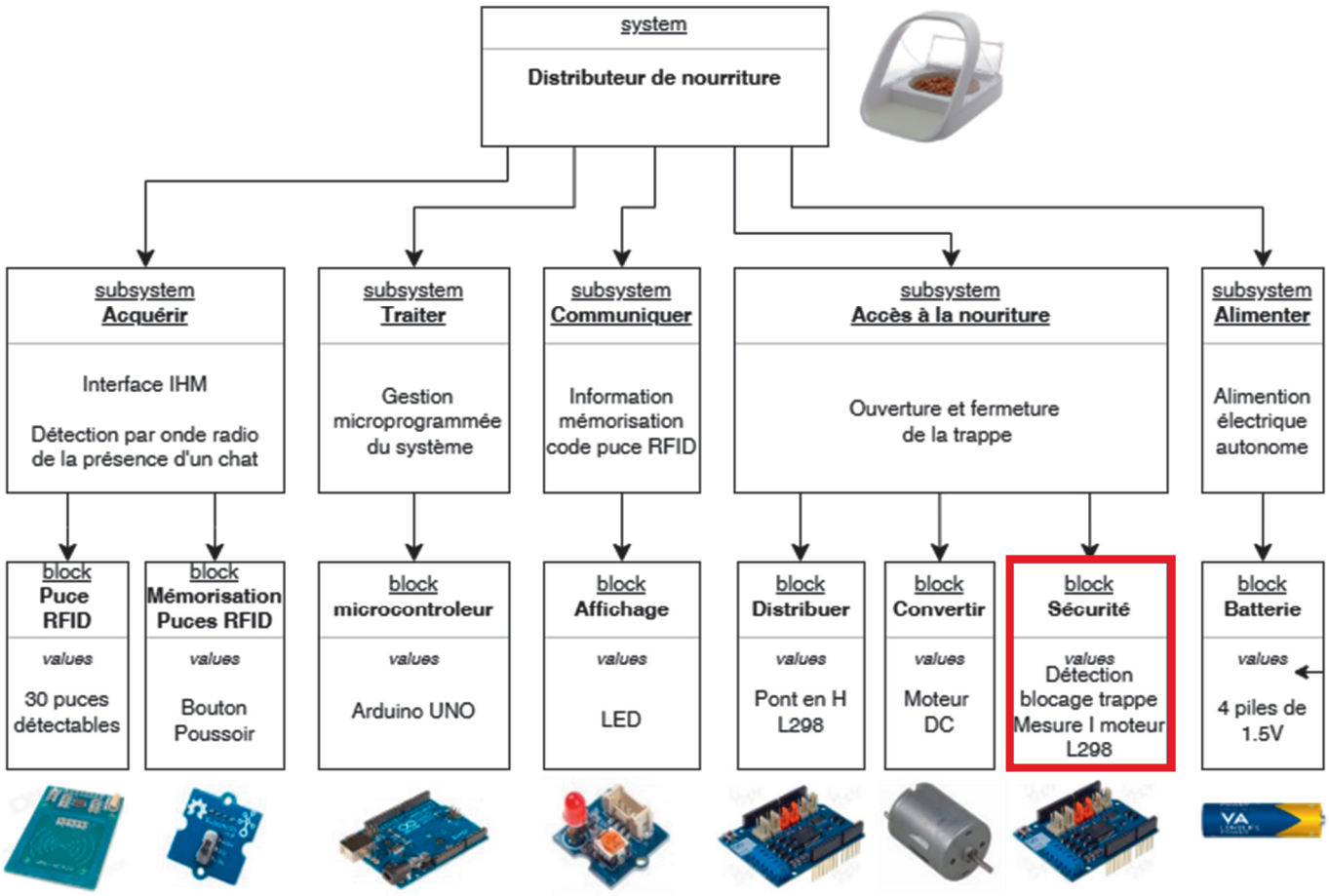
- Dimensions du distributeur : 31 cm × 23 cm × 21 cm.
- Alimentation du produit : 4 piles de 1,5 V

Problématique :

Dans un souci économique et de fiabilité, la société désire remplacer les capteurs de fins de courses mécaniques de la trappe par une solution sans contact.

Le choix s'est porté sur une détection du blocage du moteur par une mesure du courant qui le parcours.

Diagramme de blocs



Principe de la détection de fin de course par mesure du courant

Dans le système de distribution de nourriture, lorsque la trappe arrive en butée (ouverte ou fermée), le moteur se bloque provoquant une augmentation importante du courant absorbé. En détectant cette surintensité, on déterminera les positions maximales de la trappe, sans utiliser de capteurs de fin course mécaniques.

Relevé de l'intensité du courant moteur

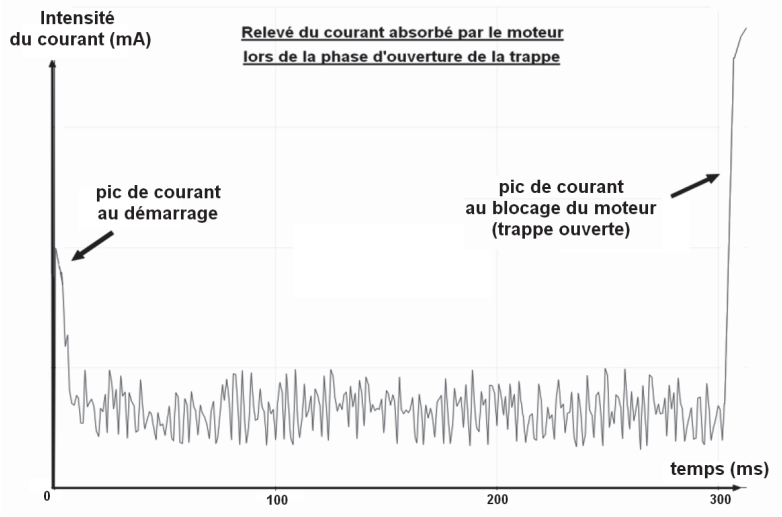


Figure 2

2. Simulation

a. Schéma structurel existant

Sur le modèle fourni figure 3 :

- Le circuit L298 (« pont en H ») commande le moteur dans 2 situations différentes :
  - Moteur en phase d'ouverture de la trappe
  - Moteur bloqué (trappe ouverte ou fermée)

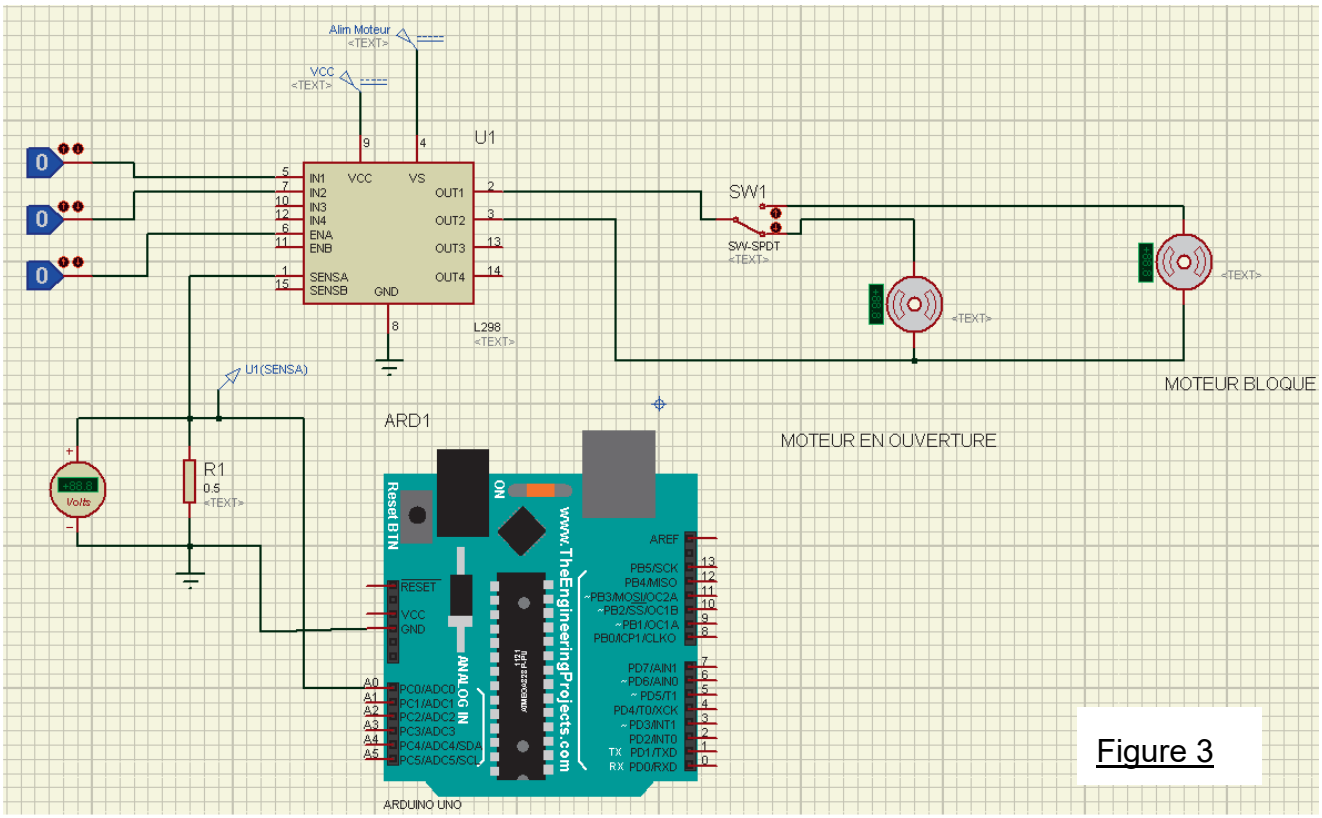


Figure 3

Tableau de fonctionnement du circuit de pilotage L298

Entrées de sélection

ENA	IN1	IN2	Résultat, en sortie
L	X	X	Moteur en roue libre (à l'arrêt, sans frein)
H	L	L	Blocage du moteur (arrêt rapide, freinage fort)
	L	H	Marche arrière
	H	L	Marche avant
	H	H	Blocage du moteur (arrêt rapide, freinage fort)

X = peu importe
L = état bas (0V, par exemple)
H = état haut (+5V, par exemple)

Figure 4

b. Tutoriel PROTEUS

Ouvrer le fichier « Distributeur_pour_chat_Sim_moteur »

Démarrer la simulation  (icone en bas à gauche de la feuille de travail)

Ajuster les valeurs des entrées IN1, IN2 et ENA pour faire tourner le moteur dans le mode désiré.

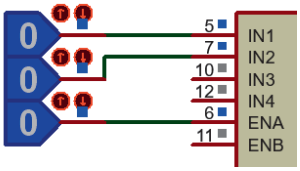


Figure 5

Sélectionner les situations à l'aide de SW1 : « moteur en ouverture » ou « bloqué » et relever les valeurs de la tension U-SHUNT. Compléter le tableau ci-dessous.

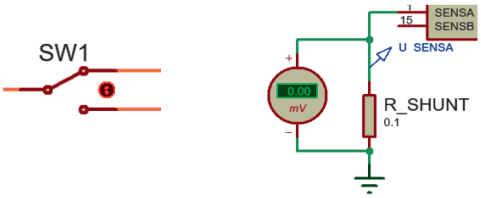
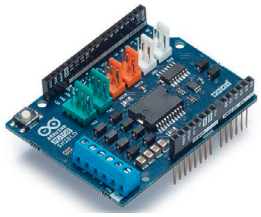


Figure 6

	Moteur en ouverture	Moteur bloqué
Tension R_ SHUNT	U_{OT} =	U_{AM} =
Courant R_ SHUNT = Courant moteur	I_{OT} =	I_{AM} =

3. Conception

a. Extrait de la documentation technique du shield de commande moteur Rev3



Le module Arduino Motor Shield Rev 3 permet de contrôler la vitesse et le sens de rotation de deux moteurs CC sur les deux canaux indépendamment. Une broche permet de connaître la consommation de chaque moteur. Le module est basé sur un L298.

Broches de commande du moteur A :

- D12 – Direction du moteur
- D3 - Vitesse du moteur (commande MLI par rapport cyclique variable)
- A0 – mesure de courant.

Câblage :

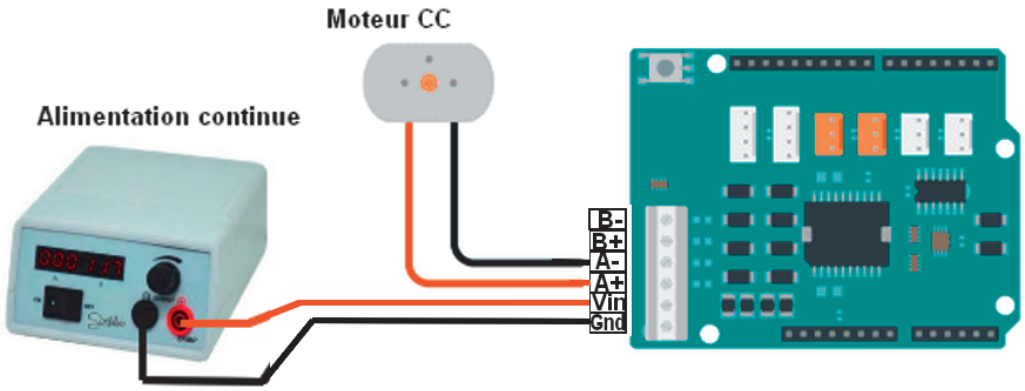


Figure 7

Programmation de la carte :

- **int** *broche_direction* **= 12;** // définit la broche de commande de direction du moteur
- **int** *broche_vitesse* **= 3;** // définit la broche de commande de vitesse du moteur
- **int** *broche_mesurecourant* **= A0 ;** // définit la broche de mesure du courant moteur
- **digitalWrite(***broche_direction***,** *état***) ;** // fixe la direction de rotation du moteur par un niveau HIGH (haut)ou LOW (bas).
- **analogWrite(***broche_vitesse***,** *valeur***) ;** // affecte une valeur de vitesse entre 0 et 100 %
- *courant* **= analogRead(***broche_mesurecourant***) ;** // lit la valeur du courant consommé par le moteur

Exemple :

b. Algorigramme du programme de commande de la trappe

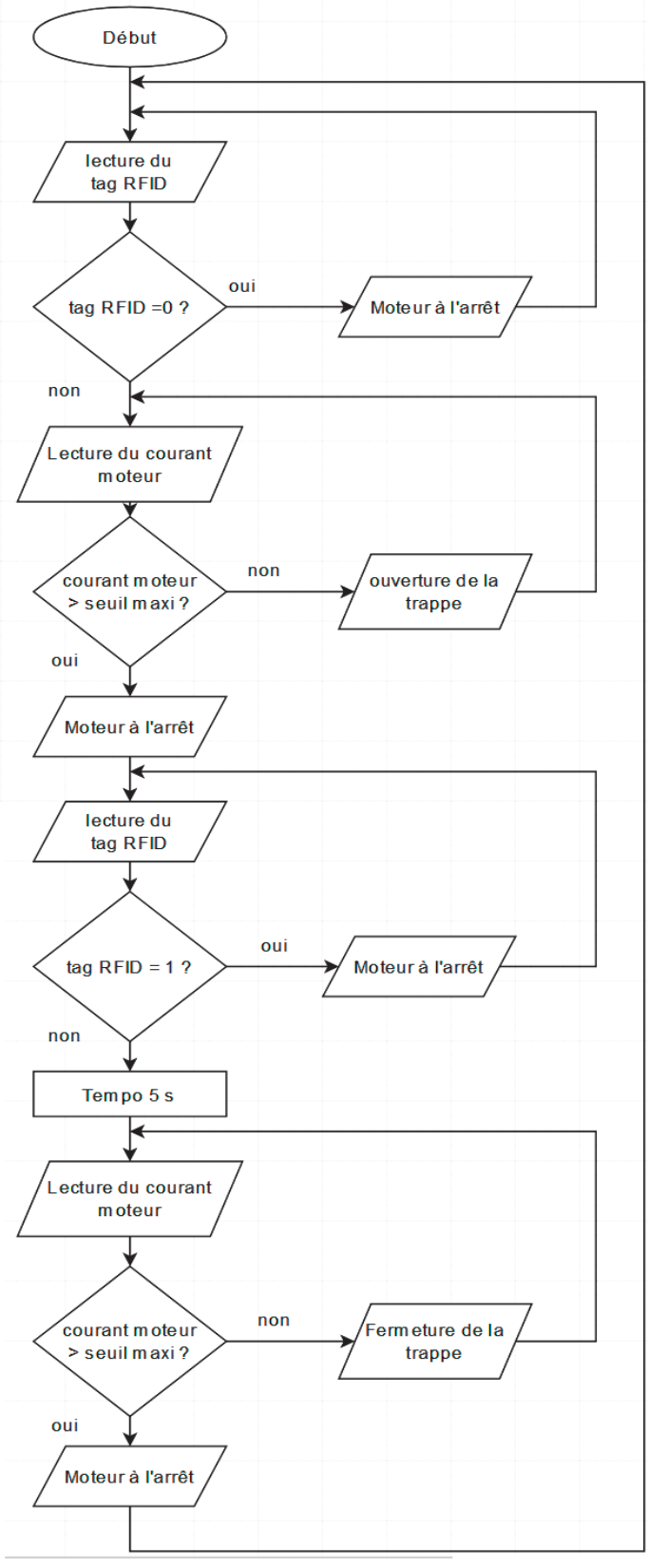


Figure 9

c. Programme partiel de commande de la trappe

Le CAN possède une résolution de 10 bits.

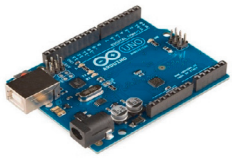
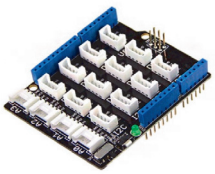
```
const int INTER = 2; // Broche 2 : interrupteur simulant la détection du tag RFID
const int LED = 4;   // Broche 4: visualisation de la détection RFID
const int DIR = 12;  // Broche 12: commande de direction du moteur M1
const int VIT = 3;   // Broche 3: commande de vitesse du moteur M1
boolean RFID;        // Variable où sera stockée l'info RFID
int IMAGE_COURANT = 0;   // Variable où sera stockée la valeur du courant moteur

void setup() {

pinMode (INTER, INPUT);
pinMode (LED, OUTPUT);
pinMode(DIR, OUTPUT);
pinMode(VIT, OUTPUT);

}

void loop() {

  //***** Tant qu'un tag RFID n'est pas détecté, le moteur M1 est à l'arrêt

  RFID = digitalRead(INTER);
while (RFID == 0)

  {

    digitalWrite(LED,.....); // Led de visualisation de détection RFID éteinte

    digitalWrite(DIR, LOW);

    analogWrite(VIT, 0); // Moteur à l'arrêt

    RFID = digitalRead(INTER); // Lecture  de l'état de la détection RFID

  }

  //***** Détection d'un tag RFID. Tant que le courant moteur < limite, on ouvre la trappe

  IMAGE_COURANT = analogRead(A0);
while (IMAGE_COURANT < .....)

  {

    digitalWrite(LED, .....); // Led de visualisation de détection tag RFID allumée

    digitalWrite(DIR, HIGH); // Ouverture de la trappe

    analogWrite(VIT, 100);

    delay(1000);

    IMAGE_COURANT = analogRead(A0); //Lecture de l'image du courant moteur M1

  }

  //***** Trappe ouverte : on arrête le moteur M1

  analogWrite(VIT, 0);// Moteur à l'arrêt

  //***** Tant que détection RFID activée, la trappe reste ouverte (moteur à l'arrêt)

  RFID = digitalRead(INTER);

  while (RFID == 1) // Tant que la broche RFID est  activée

  {

    analogWrite(VIT, 0);// Moteur M1 à l'arrêt
```

```
    RFID = digitalRead(INTER); //Lecture  de la détection RFID
  }
  //***** Détection RFID désactivée : Tempo 5s puis fermeture de la trappe après butée
  delay(5000);
  ……
…. .
….
…
  }
  //***** Arrêt du moteur M1
  analogWrite(VIT, 0); // Moteur à l'arrêt
}
```

4. Expérimentation

Pour effectuer l'expérimentation, le matériel suivant est à disposition :

- Une carte de développement Arduino.
- Une alimentation stabilisée.
- Un moteur à courant continu.
- Un shield de commande moteur et de mesure du courant moteur.
- Un shield de connexion "Base", sur lequel on pourra relier :
  - Un module interrupteur grove (simulation de la détection RFID).
  - Un module led grove (visualisation de la détection RFID).

 Carte Arduino Uno R3	 Shield moteur	 Shield grove
 Module led grove	 Module interrupteur grove	 Alimentation stabilisée
 Moteur à courant continu		