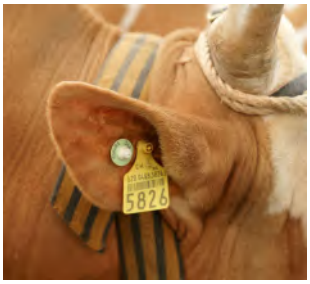
Faire de la physique avec ARDUINO – 10

Nicolas HERVE, Laurent FAURE, Gilles ESPINASSE (ENSFEA)

Dans ce dixième article nous allons créer un projet Arduino qui a pour but de simuler un DAC (Distributeur Automatique de Concentrés) dans un élevage de vaches. Comme dans le bulletin précédent, nous allons avancer pas à pas et complexifier le programme petit à petit, en y ajoutant à chaque étape les besoins que nous avons.



Oreille d'une vache équipée d'une étiquette d'identification et d'une puce RFID (Wikipedia)

Distributeur Automatique de Concentrés (http://www.orvalex.com/web/distributeur-automatique-concentre-alimentation.html)

Objets et notions abordés :

- Badges et lecteur RFID
- Servomoteur (déjà utilisé dans le bulletin n°182, faire de la physique avec Arduino n°4)
- La fonction millis() d'Arduino et la comparaison de variables pour le temps
- La gestion des données sous forme de matrice avec la classe « structure »

1- Présentation :

Le montage ci-après est un montage de prototypage à but éducatif. Il ne pourrait pas être retranscrit pour une application réelle avec les capteurs présentés.

Dans notre cas les vaches seront simulées par une carte ou jetons RFID sur lequel sera simplement indiqué le numéro de la carte. Vous pouvez, pour améliorer la maquette, découper ou fabriquer une vache fictive sur laquelle vous fixerez le badge RFID.

Nous utiliserons ensuite un lecteur RFID pour lire ces puces électroniques ou tag (puce électronique avec antenne dans son contenant) afin d'actionner le distributeur de rationnement en fonction de la vache identifiée.

Pour distribuer la nourriture, plusieurs solutions sont possibles bien sûr, un moteur pas à pas qui fait tourner une vis sans fin d'un certain nombre de tours, ou bien un servo moteur qui ouvre une trappe durant un temps fixé afin de laisser écouler la ration par gravité, etc.

Nous allons voir ici cette dernière solution en utilisant un servo moteur. Nous n'allons pas fabriquer de maquette, mais vous pouvez, si vous le souhaitez, fabriquer un contenant avec une trappe basculante que le servomoteur ouvrira ou fermera. Attention en fonction de la taille de la trappe et du poids de faire en sorte que le servomoteur ait le couple suffisant (vérifier les caractéristiques du servomoteur).

2- Lecteur RFID et Tag :

Le modèle de lecteur RFID que nous allons utiliser avec ces cartes et/ou jetons est un lecteur livré dans un kit de marque Arduino : RFID-RC522. Ce sont des lecteurs RFID basés sur le circuit MFRC-522 utilisés pour lire et écrire sur des tags de protocole Mifare de fréquence 13,56MHz. La distance optimale de lecture étant de 10mm, ce n'est pas ce type de lecteur et badge qui conviendrait donc dans la réalité.

Tableau de câblage du lecteur (https://arduino-france.site/rfid-rc522/) :

MFRC522	Arduino
GND	GND
VCC	3,3V
MOSI	11
MISO	12
SCK	13
SDA	10
RST	9

Attention, sur certaines cartes d'autres fabricants, les connectiques peuvent être différentes. Bien regarder la documentation technique.

Dans notre cas il n'y a pas beaucoup de connexions, nous pouvons donc nous affranchir de la platine d'essai (« breadboard »), en utilisant des fils male/femelle directement sur la carte Arduino.

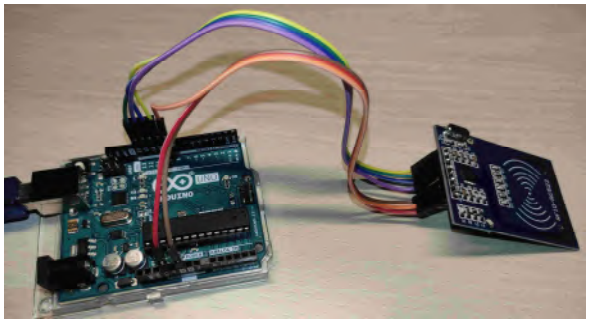
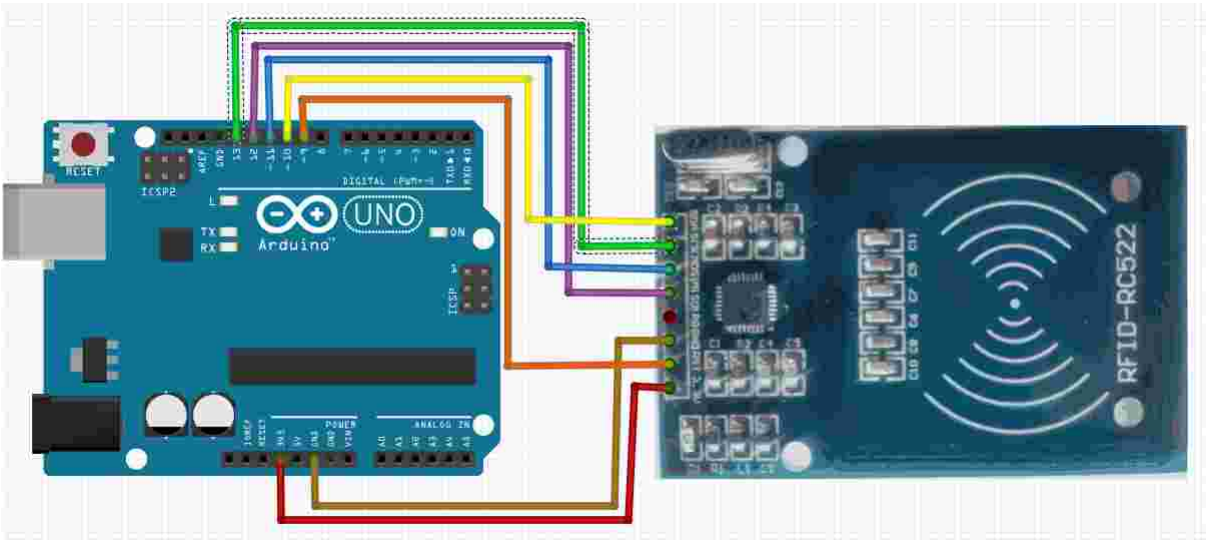


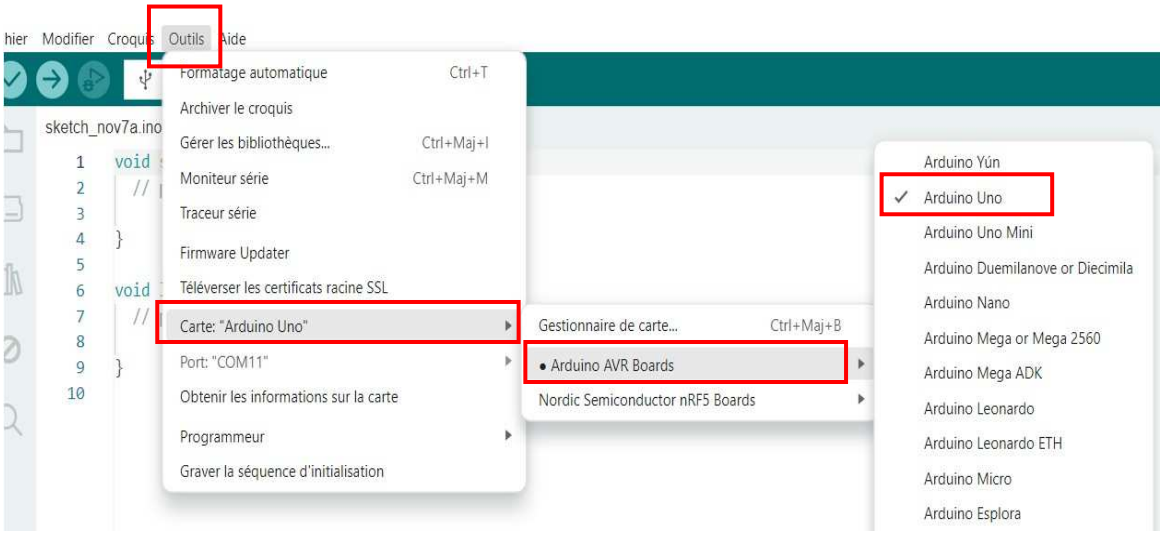
Schéma du montage :



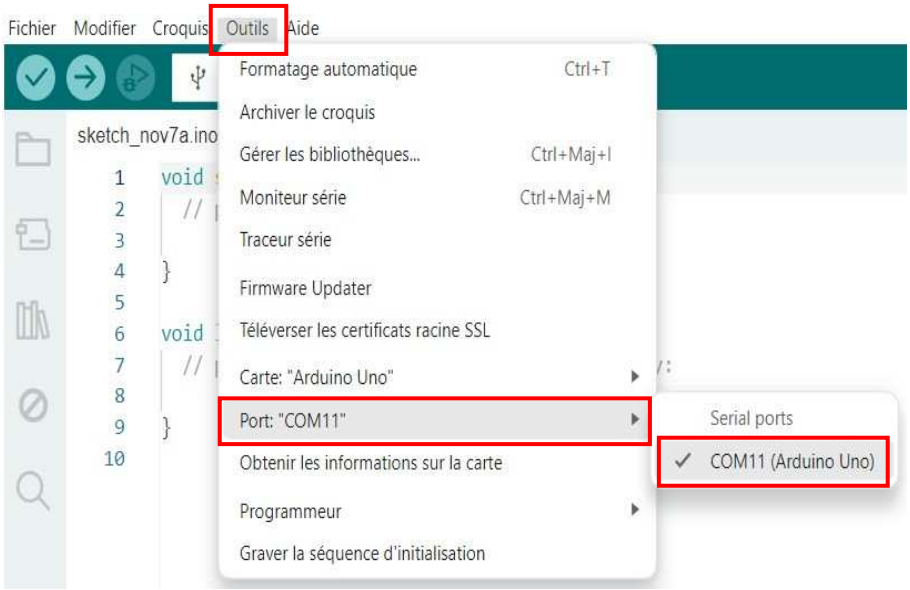
3- Coté IDE Arduino :

Nous allons maintenant ouvrir l'IDE Arduino pour téléverser sur la carte Arduino le programme qui va permettre de lire les tags RFID afin de les identifier (si vous souhaitez changer la langue de l'IDE, cliquez sur « *file* » puis « *préférences* » et sélectionnez la langue choisie, elle sera appliquée au démarrage suivant).

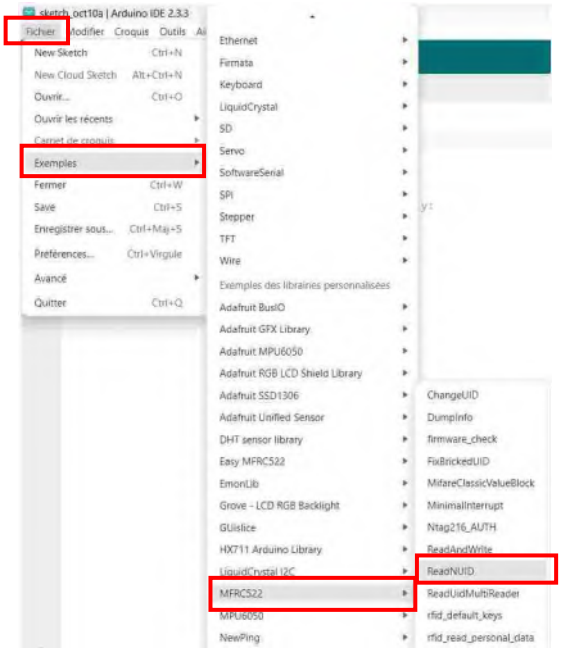
Pour cela connectez la carte Arduino à l'ordinateur avec le câble USB. Vérifiez le modèle de la carte Arduino dans l'onglet « *outils* », puis « *carte :Arduino Uno* » (dans notre cas), sinon cherchez votre carte en cliquant sur « *Arduino AVR Board* ».



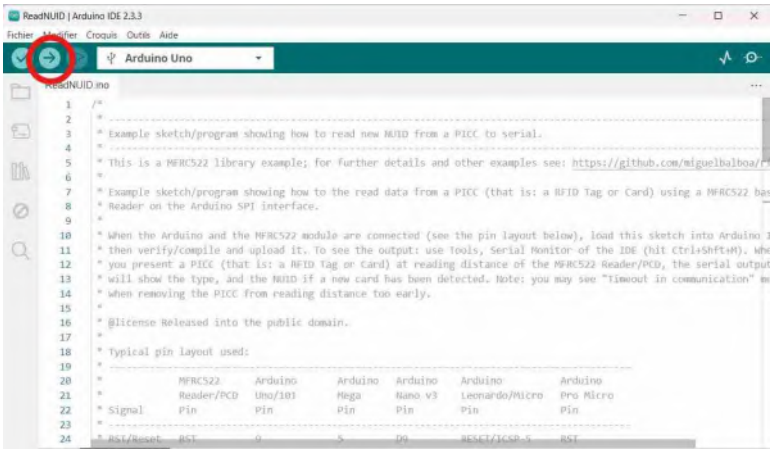
Vérifiez aussi le port « com », toujours dans l'onglet « *outils* », puis « *Port :Com* », il faut qu'il soit coché.



Ouvrez maintenant l'onglet « *Fichier* », puis « *Exemples* » et descendez pour trouver dans bibliothèque MFRC522 et l'exemple « *ReadNUID* ». Une nouvelle fenêtre de IDE s'ouvre avec le programme.

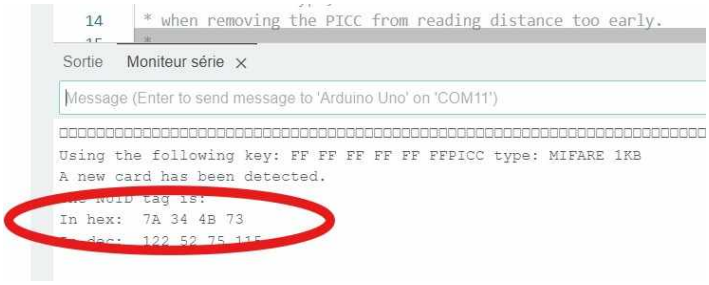


Si la bibliothèque n'est pas installée : petit rappel pour installer une bibliothèque. Cliquez sur l'onglet « *outils* » puis « *gérer les bibliothèques* ». Dans la barre de recherche tapez « *MFRC522* », en dessous apparaissent plusieurs bibliothèques, choisissez celle qui s'appelle vraiment MFRC522 et cliquez sur « *INSTALLER* ». Si une bibliothèque est déjà installée et que vous souhaitez la désinstaller, recherchez-la de la même façon et cliquez sur « supprimer », cela peut être le cas lorsqu'il y a plusieurs bibliothèques du même genre et qu'elles sont en conflit.



Téléversez ce programme à l'aide du bouton téléverser en haut à gauche

Une fois le programme téléversé, ouvrez le port série de l'Arduino en cliquant sur le bouton  en haut à droite de l'IDE. Présentez ensuite une carte ou badge devant le lecteur, son numéro en hexadécimal est alors indiqué dans la fenêtre du port série en bas de l'IDE. Associez bien ce numéro à la carte et/ou badge (en collant une étiquette par exemple) afin de correctement l'identifier par la suite.



Exécutez cette opération pour toutes les cartes et/ou badges que vous utiliserez.

Voici un programme simplifié (que nous allons utiliser par la suite) qui fait la lecture de la carte et/ou badge (TAG), vous pouvez utiliser celui-ci plutôt que celui de l'exemple dans la bibliothèque :

```
#include <SPI.h> //bibliothèque pour la communication série
#include <MFRC522.h> // bibliothèque pour le détecteur RFID

#define SS_PIN 10 // Pin pour le SS du module RC522
#define RST_PIN 9 // Pin pour le RST du module RC522

MFRC522 Lecteur(SS_PIN, RST_PIN); //Déclare un objet de type MFRC522 et de nom
"Lecteur"

void setup() {
  Serial.begin(9600); // Initialisation de la communication série
  SPI.begin();        // Initialisation de la communication SPI
  Lecteur.PCD_Init(); // Initialisation du module RFID
  Serial.println("Prêt pour la lecture"); //indique que le setup a été exécuté si
cette phrase s'inscrit
}

void loop() {
  // Vérifie si une carte est présente
  if (!Lecteur.PICC_IsNewCardPresent() || !Lecteur.PICC_ReadCardSerial()) {
    delay(50);
    return;
  }

  // Obtient l'IDentifiant de la carte en hexadécimal
  String numeroLu = ""; //définie une variable vide de chaine de caractère nommée
"TagID"
  for (byte i = 0; i < Lecteur.uid.size; i++) {
   numeroLu += String(Lecteur.uid.uidByte[i], HEX);
  }

  // Affiche le tagID lu
  Serial.print("numero Lu: ");
  Serial.println(numeroLu);
  // Arrête la lecture de la carte
  Lecteur.PICC_HaltA();
}
```

4- Le servomoteur :

Pour simuler la distribution de la ration alimentaire, nous allons utiliser un servomoteur, un angle de 0° simulera une trappe fermée et un angle de 180° celui de la trappe ouverte (à adapter si vous réalisez une vraie maquette).

Celui que nous allons utiliser ici est un servomoteur souvent fourni dans les kits, le DF9GMS :

Branchement du servomoteur :

Le fil marron au GND (masse) de l'Arduino, le fil rouge au +5V de l'Arduino et le fil orange à une sortie numérique repérée avec un tilde, par exemple la 6 de l'Arduino.

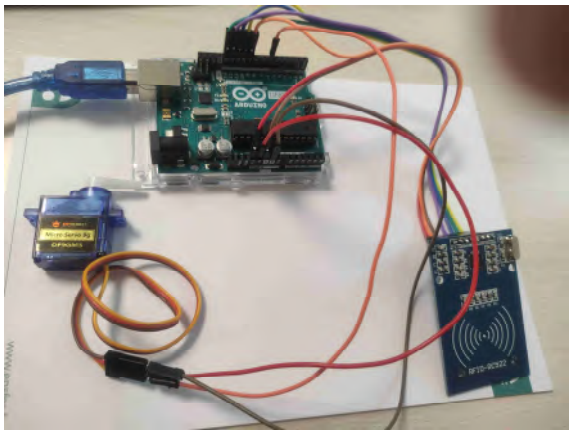
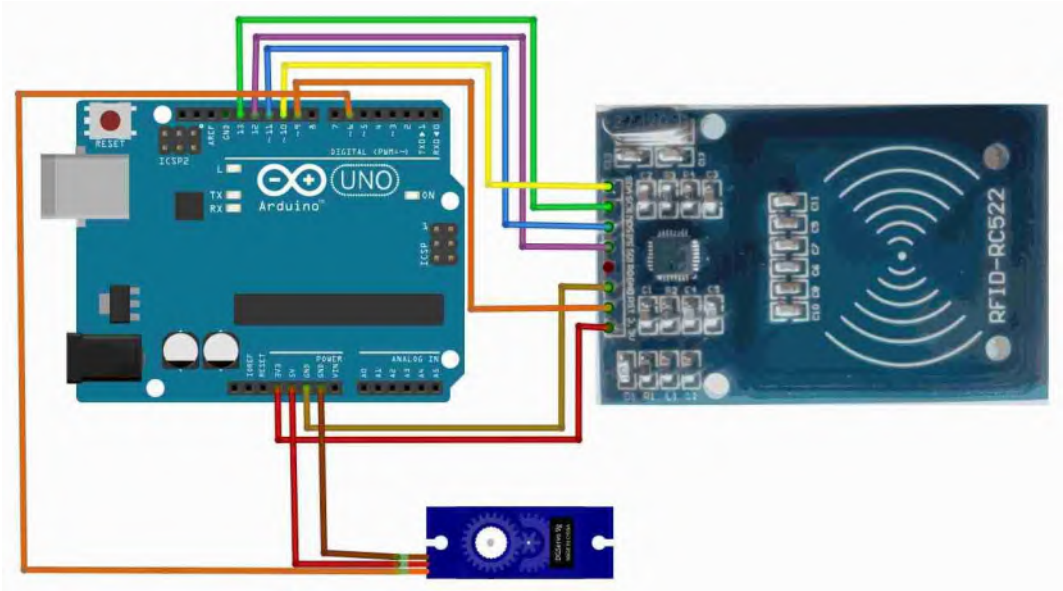


Schéma du montage :



Nous allons donc écrire un programme Arduino qui va, en fonction de la puce RFID détectée, tourner le servomoteur de 180° pendant trois laps de temps différents pour distribuer trois rations différentes. Ici, 1seconde pour les vaches 1 et 2, 3 secondes pour les vaches 3 et 4, et 6 secondes pour la vache 5.

Vous pourrez par la suite modifier, à votre convenance, le nombre de puces RFID, ainsi que le nombre de rations différentes.

Programme :

Vous trouverez, surlignées en jaune les parties rajoutées concernant le servo moteur, et en vert les parties pour différencier les vaches (attention à la casse des identifiants des vaches et ne mettez pas d'espace)

```
#include <SPI.h> //bibliothèque pour la communication série
#include <MFRC522.h> // bibliothèque pour le détecteur RFID
#include <Servo.h> // bibliothèque pour le servomoteur

#define SS_PIN 10 // Pin pour le SS du module RC522
#define RST_PIN 9 // Pin pour le RST du module RC522

MFRC522 Lecteur(SS_PIN, RST_PIN); //Déclare un objet de type MFRC522 et de nom
"Lecteur"
Servo distributeur; // création de l'objet servomoteur qui ce nomme "distributeur"

const int ration1=1000; // la variable ration1 reçoit le nombre entier 1000
milliseconde comme délais d'ouverture du distributeur
const int ration2=3000; // la variable ration2 reçoit le nombre entier 3000
milliseconde comme délais d'ouverture du distributeur
const int ration3=6000; // la variable ration3 reçoit le nombre entier 6000
milliseconde comme délais d'ouverture du distributeur
const String Vache1="6367d493"; // code hexadécimal de la puce de la vache1
const String Vache2="1376e293"; // code hexadécimal de la puce de la vache2
const String Vache3="e37bff93"; // code hexadécimal de la puce de la vache3
const String Vache4="b68d7724"; // code hexadécimal de la puce de la vache4
const String Vache5="7a344b73"; // code hexadécimal de la puce de la vache5

void setup() {
  Serial.begin(9600); // Initialisation de la communication série
  SPI.begin();        // Initialisation de la communication SPI
  Lecteur.PCD_Init(); // Initialisation du module RFID
  Serial.println("Prêt pour la lecture"); //indique que le setup a été exécuté si
cette phrase s'inscrit
  distributeur.attach(6); //indique que le servomoteur est branché sur la sortie 6
  distributeur.write(0); //on s'assure que le servomoteur est bien initialement à 0°
}

void loop() {
  // Vérifie si une carte est présente
  if (!Lecteur.PICC_IsNewCardPresent() || !Lecteur.PICC_ReadCardSerial()) {
```

```
    delay(50);
    return;
  }

  // Obtient l'IDentifiant de la carte en hexadécimal
  String numeroLu = ""; //définie une variable vide de chaine de caractère nommée
    "numeroLu"
  for (byte i = 0; i < Lecteur.uid.size; i++) {
    numeroLu += String(Lecteur.uid.uidByte[i], HEX);
  }
  // Affiche le numéro lu
  Serial.print("numero lu: ");
  Serial.println(numeroLu);
   // Arrête la lecture de la carte
  Lecteur.PICC_HaltA();

  if (numeroLu == Vache1 || numeroLu == Vache2){
    distributeur.write(180);
    delay(ration1);
    distributeur.write(0);
  }

  if (numeroLu == Vache3 || numeroLu == Vache4){
    distributeur.write(180);
    delay(ration2);
    distributeur.write(0);
  }

  if (numeroLu == Vache5){
    distributeur.write(180);
    delay(ration3);
    distributeur.write(0);
  }
}
```

Dans ce programme, les vaches peuvent repasser devant le détecteur RFID et bénéficier à nouveau d'une nouvelle ration. Nous allons donc le modifier pour que le distributeur ne fonctionne qu'une seule fois par jour pour chacune des vaches.

Pour cela nous allons définir des états pour chaque vache, état « 0 » elles n'ont pas eu de ration, état « 1 » elles ont déjà eu leur ration. Et nous allons utiliser la fonction millis() qui est une fonction interne à l'Arduino, c'est un compteur de millisecondes qui démarre à zéro lors du démarrage de l'Arduino (mise sous tension ou reset).

Attention la variable millis() prend des valeurs de 32 bits soit de 0 à 4 294 967 295 millisecondes, c'est à dire qu'après 49 jours 17h 2 min 47,295 s le compteur repart à 0. Donc pour que le programme fonctionne indéfiniment, il va donc falloir vérifier quand ce compteur passe à zéro et rajouter ce chiffre à notre variable de comparaison.

Voici un exemple de programme (les nouveaux éléments sont surlignés en jaune) :

```
#include <SPI.h> //bibliothèque pour la communication série
#include <MFRC522.h> // bibliothèque pour le détecteur RFID
#include <Servo.h> // bibliothèque pour le servomoteur

#define SS_PIN 10 // Pin pour le SS du module RC522
#define RST_PIN 9 // Pin pour le RST du module RC522

MFRC522 Lecteur(SS_PIN, RST_PIN); //Déclare un objet de type MFRC522 et de nom
"Lecteur"
Servo distributeur; // création de l'objet servomoteur qui ce nomme "distributeur"

const int ration1=1000; // la variable ration1 reçoit le nombre entier 1000
milliseconde comme délais d'ouverture du distributeur
const int ration2=3000; // la variable ration2 reçoit le nombre entier 3000
milliseconde comme délais d'ouverture du distributeur
const int ration3=6000; // la variable ration3 reçoit le nombre entier 6000
milliseconde comme délais d'ouverture du distributeur
const String Vache1="6367d493"; // code hexadécimal de la puce de la vache1
const String Vache2="1376e293"; // code hexadécimal de la puce de la vache2
const String Vache3="e37bff93"; // code hexadécimal de la puce de la vache3
const String Vache4="b68d7724"; // code hexadécimal de la puce de la vache4
const String Vache5="7a344b73"; // code hexadécimal de la puce de la vache5
String numeroLu ="";  //définie une variable vide de chaine de caractère nommée
"numeroLu"
bool etatVache1=0, etatVache2=0, etatVache3=0, etatVache4=0, etatVache5=0;
unsigned long tempsinitial;

void setup() {
  Serial.begin(9600); // Initialisation de la communication série
  SPI.begin();        // Initialisation de la communication SPI
  Lecteur.PCD_Init(); // Initialisation du module RFID
  Serial.println("Prêt pour la lecture"); //indique que le setup a été exécuté si
cette phrase s'inscrit
  distributeur.attach(6); //indique que le servomoteur est branché sur la sortie 6
  distributeur.write(0); //on s'assure que le servomoteur est bien initialement à 0°
  tempsinitial=millis(); //on met dans la variable "tempsinitial" le temps en
millseconde de la carte Arduino
}

void loop() {
 unsigned long tempsActuel = millis();

  if (tempsActuel<tempsinitial){  // cela signifie que la variable millis() est
revenue à zéro
    tempsActuel += 4294967295; //on ajoute la valeur max de millis() au
tempsActuel
  }
```

```
  if (tempsActuel -tempsinitial>86400000) { //si le temps écoulé entre le
temspinitial et maintenant est supérieur à 1 jour (86400 000 milliseconde). Ce temps
est à modifier pour voir réellement le fonctionnement (2 ou 3 minutes)
   tempsinitial=millis();  // on reprend cet instant comme temps initial
    etatVache1=0, etatVache2=0, etatVache3=0, etatVache4=0, etatVache5=0; // on met
les variables état des rations des vaches à 0
  }
  // Vérifie si une carte est présente
  if (!Lecteur.PICC_IsNewCardPresent() || !Lecteur.PICC_ReadCardSerial()) {
    delay(50);
    return;
  }

  // Obtient l'IDentifiant de la carte en hexadécimal
  String numeroLu = ""; //définie une variable vide de chaine de caractère nommée
"numeroLu"
  for (byte i = 0; i < Lecteur.uid.size; i++) {
    numeroLu += String(Lecteur.uid.uidByte[i], HEX);
  }
  // Affiche le numéro lu
  Serial.print("numero lu: ");
  Serial.println(numeroLu);
   // Arrête la lecture de la carte
  Lecteur.PICC_HaltA();

  if (numeroLu == Vache1 || numeroLu == Vache2){
    if (numeroLu == Vache1 && etatVache1==0){
      distributeur.write(180);
      delay(ration1);
      distributeur.write(0);
      etatVache1=1;
    }
    if (numeroLu == Vache2 && etatVache2==0){
      distributeur.write(180);
      delay(ration1);
      distributeur.write(0);
      etatVache2=1;
    }
  }

  if (numeroLu == Vache3 || numeroLu == Vache4){
    if (numeroLu == Vache3 && etatVache3==0){
      distributeur.write(180);
      delay(ration2);
      distributeur.write(0);
      etatVache3=1;
    }
    if (numeroLu == Vache4 && etatVache4==0){
      distributeur.write(180);
      delay(ration2);
```

```
      distributeur.write(0);
      etatVache4=1;
    }
  }

  if (numeroLu == Vache5 && etatVache5==0){
    distributeur.write(180);
    delay(ration3);
    distributeur.write(0);
    etatVache5=1;
  }
}
```

5- Amélioration du programme en utilisant « struct » :

Ce programme fonctionne mais commence à être bien trop long pour sa lisibilité, la déclaration des vaches et des temps de rations, puis les comparaisons dans la boucle « loop » tiennent trop de place, surtout si vous souhaitez ajouter des vaches ou des rations.

Nous allons donc expliquer comment regrouper des données en « paquets », afin de pouvoir, d'une part les manipuler plus facilement, et d'autre part, améliorer la lisibilité du programme.

Pour cela il existe l'outil « structure » qui peut contenir des types différents (int, bool, string…) sous forme d'une matrice.

Définissons dans ce qui suit la structure « vache » qui prend dans la première colonne le numéro de la carte, dans la deuxième le numéro de la vache correspondante, dans la troisième la durée de la ration et dans le quatrième un booléen pour savoir si elle a déjà mangé ou non (on pourrait par la suite en ajouter d'autre).

Attention aux points-virgules dans la structure et après les accolades et aux virgules dans la déclaration :

```
// Organisation de la structure pour chaque vache
struct Vache {
  String numeroLu;         // Code RFID en hexadécimal
  int numeroVache;           // Numéro de la vache
  int dureeAlimentation; // Durée d'alimentation en secondes
  bool aMangee;         // Variable pour savoir si la vache a mangé aujourd'hui
} ;


// Déclaration de la table des vaches (numéro du Tag, numéro de la vache, temps en seconde d'ouverture du distributeur, ration donnée dans la journée vrai ou faux)
Vache vaches[] = {
  {"6367d493", 1, 1, false}, //tag 6367d493, vache 1, ration 1 seconde, la vache a mangé : faux
  {"1376e293", 2, 1, false},
  {"e37bff93", 3, 3, false},
  {"b68d7724", 4, 3, false},
  {"7a344b73", 5, 6, false},
};
```

Voici le programme avec la structure décrite ci-dessus :

```
/* ---------------------------------------------------------------------------------
 * Programme permettant de lire le numéro d'une carte/badge RFID et actionne un
servomoteur pendant un durée différentes en fonction de la carte/badge
 * ---------------------------------------------------------------------------------
 * Ce programme utilise la bibliothèque MFRC522 pour un lecteur RFID basé sur ce
MFRC522
 *
 * Pour connecter le lecteur à l'Arduino, voici le tableau de connections des broches :
 * ---------------------------------------------------------------------------------
 *             MFRC522      Arduino       Arduino   Arduino    Arduino          Arduino
 *             Reader/PCD   Uno/101       Mega      Nano v3    Leonardo/Micro   Pro Micro
 * Signal      Pin          Pin           Pin       Pin        Pin              Pin
 * ---------------------------------------------------------------------------------
-----
 * RST/Reset   RST          9             5         D9         RESET/ICSP-5     RST
 * SPI SS      SDA(SS)      10            53        D10        10               10
 * SPI MOSI    MOSI         11 / ICSP-4   51        D11        ICSP-4           16
 * SPI MISO    MISO         12 / ICSP-1   50        D12        ICSP-1           14
 * SPI SCK     SCK          13 / ICSP-3   52        D13        ICSP-3           15
 *
 * ainsi que les bibliothèques SPI pour la communication série, MFRC522 pour le lecteur
RFID et Servo pour le servomoteur
 */


#include <SPI.h> //bibliothèque pour la communication série
#include <MFRC522.h> // bibliothèque pour le détecteur RFID
#include <Servo.h> // bibliothèque pour le servomoteur

#define SS_PIN 10 // Pin pour le SS du module RC522
#define RST_PIN 9 // Pin pour le RST du module RC522

MFRC522 Lecteur(SS_PIN, RST_PIN); //Déclare un objet de type MFRC522 et de nom
"Lecteur"
Servo distributeur; // création de l'objet servomoteur qui se nomme "distributeur"

// Organisation de la structure pour chaque vache
struct Vache {
  String numeroLu;          // Code RFID en hexadécimal
  int numeroVache;            // Numéro de la vache
  int dureeAlimentation; // Durée d'alimentation en secondes
  bool aMangee;          // Variable pour savoir si la vache a mangé aujourd'hui
};

// Déclaration de la table des vaches (numéro du Tag, numéro de la vache, temps en
seconde d'ouverture du distributeur, ration donnée dans la journée vrai ou faux)
Vache vaches[] = {
  {"6367d493", 1, 1, false}, //tag 6367d493, vache 1, ration 1 seconde, la vache a
mangé: faux
```

```
  {"1376e293", 2, 1, false},
  {"e37bff93", 3, 3, false},
  {"b68d7724", 4, 3, false},
  {"7a344b73", 5, 6, false},
};
int nombreVaches = sizeof(vaches) / sizeof(vaches[0]); //calcule le nombre de vaches quelque soit le nombre de ligne rentrée dans la matrice

unsigned long tempsinitial; //variable de 32bits


void setup() {
  Serial.begin(9600); // Initialisation de la communication série
  SPI.begin();        // Initialisation de la communication SPI
  Lecteur.PCD_Init(); // Initialisation du module RFID
  Serial.println("Prêt pour la lecture"); //indique que le setup a été exécuté si cette phrase s'inscrit
  distributeur.attach(6); //indique que le servomoteur est branché sur la sortie 6
  distributeur.write(0); //on s'assure que le servomoteur est bien initialement à 0°
  tempsinitial=millis(); //on met dans la variable "tempsinitial" le temps en millseconde de la carte Arduino
}

void loop() {
 unsigned long tempsActuel = millis();

  if (tempsActuel<tempsinitial){  // cela signifie que la variable millis() est revenue à zéro
    tempsActuel += 4294967295; //on ajoute la valeur max de millis() au tempsActuel
  }

  if (tempsActuel -tempsinitial>86400000) { //si le temps écoulé entre le tempsinitial et maintenant est supérieur à 1 jour (86400 000 milliseconde) ce temps est à modifier pour voir réellement le fonctionnement (2 ou 3 minutes)
    tempsinitial=millis();  // on reprend cet instant comme temps initial
    for (int i = 0; i < nombreVaches; i++) { // on met les variables état des rations des vaches à 0 (fausse)
      vaches[i].aMangee=false;
    }
  }
  // Vérifie si une carte est présente
  if (!Lecteur.PICC_IsNewCardPresent() || !Lecteur.PICC_ReadCardSerial()) {
    delay(50);
    return;
  }

  // Obtient l'IDentifiant de la carte en hexadécimal
  String numeroLu = ""; //définie une variable vide de chaine de caractère nommée "numeroLu"
  for (byte i = 0; i < Lecteur.uid.size; i++) {
```

```
    numeroLu += String(Lecteur.uid.uidByte[i], HEX);
  }
  // Affiche le numéro lu
  Serial.print("numero lu: ");
  Serial.println(numeroLu);
   // Arrête la lecture de la carte
  Lecteur.PICC_HaltA();

  // Recherche de la vache associée au tag
  for (int i = 0; i < nombreVaches; i++) {
    if (vaches[i].numeroLu == numeroLu) {
      if (!vaches[i].aMangee) { //si la vache n'a pas mangée
        //écrit sur le port série les informations relative à la vache
        Serial.print("Vache ");
        Serial.print(vaches[i].numeroVache);
        Serial.print(" mange pour ");
        Serial.print(vaches[i].dureeAlimentation);
        Serial.println(" secondes.");
        //distribution de la ration
        distributeur.write(180);
        delay(vaches[i].dureeAlimentation * 1000); // Durée d'alimentation en secondes
x1000 pour l'avoir en milliseconde
        distributeur.write(0);

        // Marquer la vache comme ayant mangé
        vaches[i].aMangee = true;
        Serial.print("Vache ");
        Serial.print(vaches[i].numeroVache);
        Serial.println(" a terminé de manger.");

      } else {
        //sinon on indique qu'elle a déjà mangé
        Serial.print("Vache ");
        Serial.print(vaches[i].numeroVache);
        Serial.println(" a déjà mangé aujourd'hui.");
      }
    }
  }
}
```

Comme toujours les programmes Arduino peuvent être améliorés à votre guise en fonction de vos besoins. Vous pouvez par exemple ajouter une horloge RTC pour ne pas utiliser la fonction millis().

Application pédagogique du projet

Ce projet numérique peut être mené par exemple dans la pluri « Capteurs connectés au service des transitions », en lien avec l'enseignement de PA et/ou de TIM. L'enseignement de Physique-Chimie peut permettre aux élèves de comprendre le fonctionnement d'une technologie existante et répandue, dont ils peuvent visiter une installation réelle dans l'enseignement de PA. La construction d'une maquette ou d'un prototype permet ainsi de rentrer dans le fonctionnement matériel et logiciel de cette technologie. L'enseignement TIM peut permettre aux élèves d'aller plus loin dans les éléments de programmation, voire de prolonger ce projet en mettant en place un suivi sur smartphone de l'alimentation automatique des vaches.

Prix du projet

Liste des composants électroniques et tarifs TTC approximatif :

- Arduino Uno3 (23,90€) ou carte compatible Gotronic (14,90€)
- Lecteur RFID avec une carte et un badge (9,90€)
- Lot de 5 badges Mifare-one 13,56MHz (3,10€)
- Lot de 5 cartes Mifare-one 13,56MHz (3,90€)
- Ecran LCD I2C facultatif (8,80€)

Total : entre 31,80€ et 49,60€ TTC