25/05/2025

Veronique Cloutier & Félix-Alexandre Morneau-Carrier

Cégep Limoilou

Projet – partie 2

Distributeur automatique de nourriture avec reconnaissance

Table des matières

[Introduction 1](#_Toc198564641)

[Photos du montage 2](#_Toc198564642)

[Liste des composants 5](#_Toc198564643)

[Code 6](#_Toc198564644)

[Problèmes rencontrés 12](#_Toc198564645)

[Annexe 1 : Tableau des liens entre les composants 13](#_Toc198564646)

# Introduction

Ce projet vise à concevoir un distributeur de nourriture automatique et intelligent pour chats, basé sur les technologies IoT. L’objectif est de contrôler l’accès à la nourriture à l’aide de la reconnaissance RFID, afin que seuls les chats autorisés puissent manger. Le système détecte la présence de l’animal, vérifie son identité via une puce RFID, et ouvre automatiquement le compartiment à l’aide d’un servomoteur.

Le prototype repose sur un microcontrôleur Arduino Uno, qui gère plusieurs composants : un lecteur RFID (RC522), un capteur de présence (PIR), un moteur pas-à-pas pour distribuer la nourriture, et une cellule de charge avec module HX711 pour mesurer la quantité de nourriture. Ce rapport présente le montage électronique, le code utilisé, et les étapes de développement, incluant les difficultés rencontrées et les solutions apportées.

# Photos du montage

Une image contenant fils électriques, câble, Ingénierie électronique, Appareils électroniques

Le contenu généré par l’IA peut être incorrect.

Une image contenant Appareils électroniques, fils électriques, câble, Ingénierie électronique

Le contenu généré par l’IA peut être incorrect.Une image contenant table, conteneur, fournitures de bureau, carton

Le contenu généré par l’IA peut être incorrect.Une image contenant intérieur, boîte, Emballages, Empaquetage et étiquetage

Le contenu généré par l’IA peut être incorrect.

# Liste des composants

|  |  |
| --- | --- |
| **Quantité** | **Composant** |
| **1** | Lecteur RFID + puce/tag RFID  (RC522) |
| **1** | HC-SR04 (capteur de distance à ultrasons) |
| **1** | Servomoteur FS90 |
| **1** | Step-Motor 28BYJ-48 + ULN2003 |
| **1** | Microcontrôleur Uno R3 |
| **1** | ENC28J60 (pour internet) |
| **1** | BreadBoard - Half Size |
| **1** | Mini BreadBoard |
| **2** | Boutons |
| **1** | RTC DS1307 (real time clock) |

Voir en **annexe 1** la liste des liens entre les composants

# Code

#include <SPI.h>                    // Bibliothèque pour la communication SPI (utilisée par le module RFID)

// #include <UIPEthernet.h>         // Commenté car on n'utilise pas encore internet

#include <MFRC522.h>                // Bibliothèque pour le lecteur RFID RC522

#include <Servo.h>                  // Bibliothèque pour contrôler un servo moteur

#include <Wire.h>                   // Bibliothèque pour la communication I2C (utilisée par le module RTC)

#include <RTClib.h>                 // Bibliothèque pour utiliser un module d'horloge temps réel (DS1307)

// Définition des broches utilisées pour les différents composants

#define MOTOR\_STEP\_PIN   3          // IN1 : Broche de contrôle pour le moteur pas à pas

#define MOTOR\_DIR\_PIN    4          // IN2 : Broche de contrôle pour le moteur pas à pas

#define MOTOR\_IN3\_PIN    5          // IN3 : Broche de contrôle pour le moteur pas à pas

#define MOTOR\_IN4\_PIN    6          // IN4 : Broche de contrôle pour le moteur pas à pas (partagée avec le servo !)

#define SERVO\_PIN        7          // Broche de signal pour contrôler le servo moteur

#define RFID\_RST\_PIN     9          // Broche RST (reset) du module RFID

#define RFID\_SDA\_PIN     10         // Broche SS (Slave Select / Chip Select) du module RFID

#define BTN\_INC\_PIN      A0         // Broche pour le bouton d’augmentation d’intervalle

#define BTN\_DEC\_PIN      A1         // Broche pour le bouton de diminution d’intervalle

#define TRIG\_PIN         A2         // Broche TRIG du capteur à ultrasons HC-SR04

#define ECHO\_PIN         A3         // Broche ECHO du capteur à ultrasons HC-SR04

// Initialisation des objets

MFRC522 rfid(RFID\_SDA\_PIN, RFID\_RST\_PIN);  // Création de l'objet RFID

Servo servo;                               // Création de l'objet Servo

RTC\_DS1307 rtc;                            // Création de l'objet horloge RTC

// EthernetServer server(80);              // Commenté pour désactiver Ethernet pour l'instant

// Constantes et variables de configuration

const int stepsPerRevolution = 2048;       // Nombre de pas pour une révolution complète du moteur

int stepperSpeed = 30;                     // Vitesse du moteur en tours par minute (non utilisée ici)

const int stepDelay = 5;                   // Délai entre chaque pas du moteur (en ms)

int angleOuvert = 90;                      // Angle pour ouvrir le couvercle avec le servo

int angleFerme = 0;                        // Angle pour fermer le couvercle

bool estOuvert = false;                    // État actuel du couvercle (ouvert ou fermé)

unsigned long lastMotorTime = 0;           // Dernier moment où le moteur a été activé

unsigned long motorInterval = 8UL \* 60 \* 60 \* 1000;  // Intervalle entre activations auto (8h en ms)

unsigned long motorDuration = 10000;       // Durée d’activation du moteur (non utilisée ici)

unsigned long lastDebounceTimeInc = 0;     // Dernier moment d'appui sur le bouton +

unsigned long lastDebounceTimeDec = 0;     // Dernier moment d'appui sur le bouton -

unsigned long debounceDelay = 50;          // Anti-rebond pour les boutons (en ms)

String expectedUID = "ac1ff21";            // UID autorisé pour ouvrir le couvercle via RFID

void setup() {

  Serial.begin(9600);           // Initialisation du port série

  SPI.begin();                  // Initialisation de la communication SPI

  rfid.PCD\_Init();              // Initialisation du lecteur RFID

  servo.attach(SERVO\_PIN);      // Attache du servo à la broche définie

  servo.write(angleFerme);      // Position initiale du servo : fermé

  // Configuration des boutons en entrée avec résistance pull-up (+ et -)

  pinMode(BTN\_INC\_PIN, INPUT\_PULLUP);

  pinMode(BTN\_DEC\_PIN, INPUT\_PULLUP);

  // Configuration des broches de contrôle du moteur pas à pas

  pinMode(MOTOR\_STEP\_PIN, OUTPUT);

  pinMode(MOTOR\_DIR\_PIN, OUTPUT);

  pinMode(MOTOR\_IN3\_PIN, OUTPUT);

  pinMode(MOTOR\_IN4\_PIN, OUTPUT);

  // Configuration du capteur à ultrasons

  pinMode(TRIG\_PIN, OUTPUT);

  pinMode(ECHO\_PIN, INPUT);

  // Initialisation du serveur Ethernet (commenté)

  // uint8\_t mac[6] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};

  // Ethernet.begin(mac);

  // server.begin();

  // Serial.print("Serveur web démarré à l'adresse IP : ");

  // Serial.println(Ethernet.localIP());

  // Initialisation du module RTC

  Wire.begin();

  if (!rtc.begin()) {

    Serial.println("Impossible de trouver un RTC DS1307");

    while (1); // Stoppe le programme si RTC non trouvé

  }

  // Vérifier si le module RTC fonctionne

  if (!rtc.isrunning()) {

    Serial.println("RTC n'est pas en marche, mise à jour de l'heure");

    // Si le RTC n'est pas en marche, on initialise avec la date/heure actuelle

    rtc.adjust(DateTime(F(\_\_DATE\_\_), F(\_\_TIME\_\_)));

  }

}

void loop() {

  DateTime now = rtc.now();  // Récupération de l'heure actuelle

  // Serveur web (commenté)

  // EthernetClient client = server.available();

  // if (client) {

  //     // Code du serveur web ici

  // }

  // Activation automatique du moteur après une période définie

  if ((millis() - lastMotorTime >= motorInterval) && !estOuvert) {

    activerMoteur();

  }

  // Lecture d'une carte RFID

  if (rfid.PICC\_IsNewCardPresent() && rfid.PICC\_ReadCardSerial()) {

    Serial.println("Puce RFID détectée !");

    if (!estOuvert) {

      // Construction de l'UID sous forme de chaîne hexadécimale

      String uid = "";

      for (byte i = 0; i < rfid.uid.size; i++) {

        uid += String(rfid.uid.uidByte[i], HEX);

      }

      // Comparaison avec l'UID autorisé

      if (uid == expectedUID) {

        Serial.println("Puce autorisée détectée !");

        servo.write(angleOuvert);  // Ouvre le couvercle

        estOuvert = true;

        Serial.println("Couvercle ouvert.");

      } else {

        Serial.println("Puce non autorisée.");

      }

    }

    delay(1000); // Attente pour éviter plusieurs lectures

  }

  // Si le couvercle est ouvert, vérifie la distance pour savoir si refermer

  long distance = mesurerDistance();

  if (estOuvert && distance > 15) { // Si rien devant (distance > 15 cm), refermer

    servo.write(angleFerme);

    estOuvert = false;

    Serial.println("Couvercle fermé.");

    delay(1000); // Pause pour laisser le temps de fermer

  }

  // Gestion du bouton d’augmentation de l’intervalle

  if (digitalRead(BTN\_INC\_PIN) == LOW && (millis() - lastDebounceTimeInc) > debounceDelay) {

    lastDebounceTimeInc = millis();

    if(motorInterval < 24.0 \* 60 \* 60 \* 1000) {  // Limite supérieure : 24h

      motorInterval += 30.0 \* 60 \* 1000;  // +30 minutes

      Serial.print("Intervalle augmenté à : ");

      Serial.print(motorInterval / 1000 / 60 / 60); // Affiche en heures

      Serial.print("h");

      unsigned long minutes = motorInterval / 1000 / 60 % 60;

      if(minutes < 10) Serial.print("0");

      Serial.println(minutes);

      delay(1000); // Pause pour éviter double appui

    }

  }

  // Gestion du bouton de diminution de l’intervalle

  if (digitalRead(BTN\_DEC\_PIN) == LOW && (millis() - lastDebounceTimeDec) > debounceDelay) {

    lastDebounceTimeDec = millis();

    if (motorInterval > 30.0 \* 60 \* 1000) { // Limite inférieure : 30 minutes

      motorInterval -= 30.0 \* 60 \* 1000; // -30 minutes

      Serial.print("Intervalle réduit à : ");

      Serial.print(motorInterval / 1000 / 60 / 60); // Affiche en heures

      Serial.print("h");

      unsigned long minutes = motorInterval / 1000 / 60 % 60;

      if(minutes < 10) Serial.print("0");

      Serial.println(minutes);

      delay(1000); // Pause pour éviter double appui

    }

  }

}

// Fonction pour activer le moteur pas à pas pendant une révolution complète

void activerMoteur() {

  Serial.println("Activation du moteur pas à pas...");

  lastMotorTime = millis();  // Mise à jour du moment de dernière activation

  for (int i = 0; i < stepsPerRevolution; i++) {

    tournerUnPas();          // Effectue un pas

    delay(stepDelay);        // Petite pause entre chaque pas

  }

  Serial.println("Moteur désactivé.");

}

// Fonction pour exécuter une séquence de pas du moteur

void tournerUnPas() {

  static int stepIndex = 0;

  // Séquence de 8 demi-pas pour un moteur 28BYJ-48

  const int sequence[8][4] = {

    {1, 0, 0, 0},

    {1, 1, 0, 0},

    {0, 1, 0, 0},

    {0, 1, 1, 0},

    {0, 0, 1, 0},

    {0, 0, 1, 1},

    {0, 0, 0, 1},

    {1, 0, 0, 1}

  };

  // Application de la séquence sur les broches du moteur

  digitalWrite(MOTOR\_STEP\_PIN, sequence[stepIndex][0]);

  digitalWrite(MOTOR\_DIR\_PIN, sequence[stepIndex][1]);

  digitalWrite(MOTOR\_IN3\_PIN, sequence[stepIndex][2]);

  digitalWrite(MOTOR\_IN4\_PIN, sequence[stepIndex][3]);

  stepIndex = (stepIndex + 1) % 8; // Passage au pas suivant

}

// Fonction pour mesurer la distance à l'aide du capteur à ultrasons HC-SR04

long mesurerDistance() {

  digitalWrite(TRIG\_PIN, LOW);

  delayMicroseconds(2);                     // Attente brève

  digitalWrite(TRIG\_PIN, HIGH);

  delayMicroseconds(10);                    // Envoie un pulse de 10µs

  digitalWrite(TRIG\_PIN, LOW);

  long duration = pulseIn(ECHO\_PIN, HIGH);  // Mesure du temps de retour de l'écho

  long distance = duration \* 0.034 / 2;     // Conversion en cm (vitesse du son : 0.034 cm/µs)

  return distance;

}

# Problèmes rencontrés

1. Problème : Avant on avait un HC-SR501 pour détecter si le chat a quitté, mais cela était peu précis et il y avait un bon délai avant qu’on ait l’info que le chat a quitté.

Solution : On a décidé de remplacer par un capteur de présence à ultrasons, qui permet un retour rapide.

1. Problème : On voulait afficher dans le serial monitor la fréquence de repas du chat suite à l’incrémentation/décrémentation via le bouton, en format (HH:MM), mais en tentant de formater pour L’affichage on obtenait toujours 00 :00!

Solution : Ce sont les parenthèses autour du calcul des heures et minutes qui causaient problème. On les a donc enlevées.

1. Problème : On ne savait pas comment faire le mécanisme de distribution de nourriture, pour que cela fonctionne physiquement parlant.

Solution : Nous avons fait un mécanisme ressemblant à une hélice qui tourne via le stepper motor pour faire écouler la nourriture dans le bol. Nous avons fait divers tests pour arriver à une solution assez fonctionnelle et qui soit solide, ce qui était un défi vu que nous avons surtout utilisé du carton.

# Annexe 1 : Tableau des liens entre les composants

|  |  |  |  |
| --- | --- | --- | --- |
| **Module / Composant** | **Broche du module** | **Arduino Uno R3 Pin** | **Remarque** |
| **RFID RC522** | SDA (SS) | D10 | CS (SS) réservé au RFID |
|  | SCK | D13 | SPI |
|  | MOSI | D11 | SPI |
|  | MISO | D12 | SPI |
|  | RST | D9 |  |
|  | VCC | **3.3V** | ⚠ Jamais 5V |
|  | GND | GND |  |
| **Ethernet ENC28J60** | CS (SS) | **D8** | ✅ Pas en conflit |
|  | SCK, MISO, MOSI | D13, D12, D11 | SPI partagé |
|  | VCC | 3.3V **ou** 5V | Selon le régulateur du module |
|  | GND | GND |  |
| **RTC DS1307** | SDA | A4 | I2C |
|  | SCL | A5 | I2C |
|  | VCC | 5V | OK |
|  | GND | GND |  |
| **Servo moteur** | Signal | D7 | PWM |
|  | VCC / GND | 5V / GND |  |
| **Capteur PIR** | OUT | D2 | Digital input |
| **Bouton +** |  | **A0 (D14)** | INPUT\_PULLUP |
| **Bouton -** |  | **A1 (D15)** | INPUT\_PULLUP |
| **Moteur pas à pas** | IN1 | D3 | ULN2003 - Séquence 1 |
|  | IN2 | D4 | ULN2003 - Séquence 2 |
|  | IN3 | D5 | ULN2003 - Séquence 3 |
|  | IN4 | D6 | ULN2003 - Séquence 4 |
|  | VCC / GND | 5V / GND | Alimente ULN2003 |

                 +----------------------+

                 |     Arduino UNO R3   |

                 |                      |

                 | [A0] <--- Bouton + --+ BTN\_INC\_PIN ✅

                 | [A1] <--- Bouton - --+ BTN\_DEC\_PIN ✅

                 | [A4] <--- SDA <------ RTC DS3231

                 | [A5] <--- SCL <------ RTC DS3231

                 |                      |

                 | [D2] <--- PIR -------+ OUT (signal)

                 | [D3] ---> ULN2003 ---+ IN1 (28BYJ-48)

                 | [D4] ---> ULN2003 ---+ IN2

                 | [D5] ---> ULN2003 ---+ IN3

                 | [D6] ---> ULN2003 ---+ IN4

                 | [D7] <-- Servo---------Signal

                 | [D8] <--- Ethernet --+ CS ENC28J60

                 | [D9] <--- RFID ------+ RST

                 | [D10]<-- RFID -------+ CS (SS)

                 | [D11]<-- MOSI -------+ SPI partagé

                 | [D12]<-- MISO -------+ SPI partagé

                 | [D13]<-- SCK  -------+ SPI partagé

                 +----------------------+

                        POWER & GND

                        ------------

                 [3.3V] ---> RFID RC522, ENC28J60

                 [5V]   ---> RTC DS1307, PIR, Servo, ULN2003