Université Euro Méditerranéenne Fès
Euro Mediterranean University of Fez

Ecole d'Ingénierie Digitale et d'Intelligence Artificielle
(EIDIA)

Projet de Fin de module

Filière : 2ème année classes préparatoires Integrées

Semestre : 4

Module : Electronique embarquée

Thème :

Machine de remplissage sécurisée

Soutenu le 22 /05/24,

Encadré par :

Pr. A. SLIMANI

Préparé par les étudiants:

- NAAOUACHE Salma
- TOUZANI Rayhane

Année Universitaire : 2023-2024

Objectif du projet :

Concevoir une machine de remplissage automatique avec Arduino offre une occasion fascinante d'explorer les capacités de ce microcontrôleur incroyablement adaptable. Ce projet nécessite à la fois une expertise dans la programmation des fonctionnalités clés de la machine et une connaissance du câblage pour intégrer les divers composants le plus efficacement possible. Avec la cinétique du remplissage qui est automatisée, il vise à garantir un dosage précis et une distribution efficace du café. En outre, le projet donne une occasion d'apprendre divers concepts d'ingénierie, y compris la façon dont les capteurs et les actionneurs peuvent être conçus, comment gérer l'énergie, et comment la mécanique fonctionne.

Introduction :

La machine de remplissage fonctionne de manière interactive et sécurisée grâce à l'utilisation d'une carte RFID. Lorsqu'un utilisateur souhaite utiliser la machine, il doit scanner sa carte RFID pour accéder au système. La machine vérifie alors la validité de la carte avant de passer à l'étape suivante. Une fois que la carte est reconnue comme valide, l'écran LCD de la machine affiche un message invitant l'utilisateur à faire son choix parmi une sélection de cinq boissons différentes. À l'aide du clavier numérique, l'utilisateur sélectionne le numéro correspondant à sa boisson préférée parmi les options proposées, comme le café pur, diverses proportions de café et de lait, ou simplement du lait. Une fois le choix effectué, la machine démarre le processus de remplissage. Les pompes intégrées dosent soigneusement les ingrédients pour préparer la boisson choisie, remplissant la tasse de manière précise. Une fois le remplissage terminé, une LED s'allume pour signaler à l'utilisateur que sa boisson est prête. De plus, la machine offre une touche finale interactive : un capteur ultrasonique détecte la présence de la main de l'utilisateur sous une LED spéciale. Lorsque la main est détectée, un moteur pas à pas est activé pour libérer un sachet de sucre, ajoutant ainsi une touche personnalisée à la boisson. Ce processus ingénieux combine à la fois la précision du dosage, l'interaction utilisateur et une petite touche ludique pour offrir une expérience de café personnalisée et agréable.

Année Universitaire : 2023-2024

Phase de réalisation :

➢ Choix des composants et idée du projet :

On a eu l'idée de réalisé une machine de remplissage, et voilà notre premier sketch, qui a été modifié divers fois.

Les composants électroniques dont on a pensé en premier, étaient :

- **Carte Arduino Mega :** pour brancher les composants.
- **Breadboard**.
- **Files** (m-m, m-f et f-f).
- **Keypad 4x3 :** pour entrer le choix de la boisson voulue.
- **Ecran LCD (module I2C) :** pour afficher si l'accès est autorisé, et afficher le choix que l'utilisateur a fait.
- **Mini Pompe DC 5V (x2) :** pour pomper le café et le lait.
- **Moteur Pas à Pas :** pour faire tourner le fil en acier et faire tomber le sucre

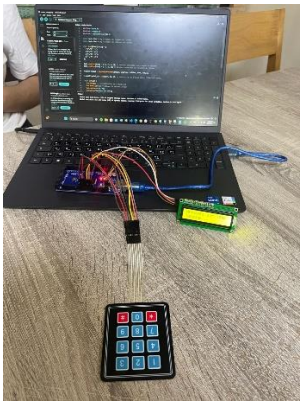
On a aussi acheté du *carton* pour la conception de la machine, des *tuyeaux* pour faire passer le liquide, et du *fil en acier* pour faire tomber le sucre.

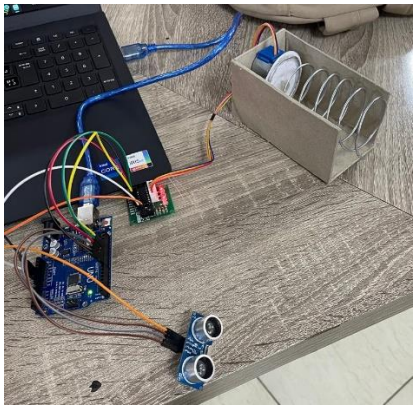
Puis on a eu l'idée d'ajouter une **carte RFID** pour assurer la sécurité de la machine. Et pour le sucre on a ajouté l'option d'obtenir un sachet de sucre à l'aide d'un **capteur ultrason (HC-SR04)** qui détecte la main et donne un signal au moteur pour tourner et fait tomber le sachet.

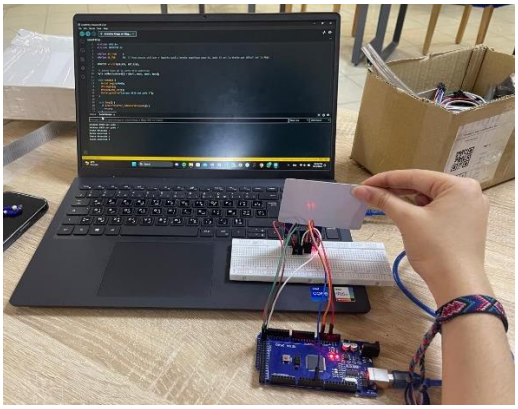
➢ Assemblage et Conception :

Après avoir ramasser les composants nécessaires pour notre projet, on a commencé la conception de la machine en premier.

Puis on a fait des testes sur les différents composants individuellement avec de simples montages et des codes basics de test, pour s'assurer que les composants marchent correctement.

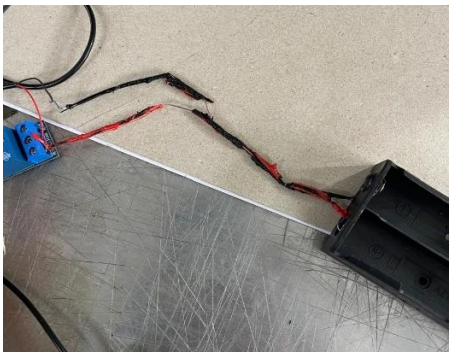


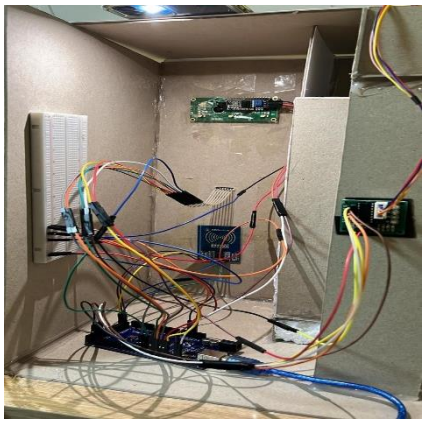




On a eu des problèmes avec quelques composants qui ne marchaient pas, et d'autres qui ont arrêté de marcher du jour au lendemain. On a eu affaire a des courts circuits et des problèmes d'alimentation qui nous on appris plein de leçons à appliquer même dans notre vie quotidienne.

Après plusieurs tests, et divers points de vue, on a commencé peu à peu à monter notre projet en commençant du support en carton, puis le câblage et ensuite les tests finaux qui se sont avéré un peu plus complexe que ce croyait, car on avait choisi le carton comme matière pour fabriquer notre machine et la moindre erreur de pompage aurait entrainer de gros problème, mais à la fin on est arrivé à bien métriser notre projet et à s'assurer que tout marchait à la perfection.





Le coût total du projet est à peu près 1500 dh. Comme c'est un projet qu'on a fait nous même de A à Z, le coût a été élevé, d'autant plus que quelques composants ne marchaient pas donc on a dû refaire plusieurs commandes, et à chaque fois on se rendait compte qu'un composant nous manquait. Ce projet nous a appris à être créatifs et innovants, on a appris gérer notre temps et travailler efficacement. En outre, on a appris les principes et les bases de l'électronique.

- Code et fonctionnalités :

Code_Complet.ino

```ino
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <Keypad.h>
#include <NewPing.h>
#include <Stepper.h>
#include <SPI.h>
#include <MFRC522.h>

#define TRIGGER_PIN 12
#define ECHO_PIN 11
#define distance 15
#define RELAY_1_PIN 2 // Broche de commande du relai 1
#define RELAY_2_PIN 3 // Broche de commande du relai 2

LiquidCrystal_I2C lcd(0x27, 16, 2); // LCD
NewPing sonar(TRIGGER_PIN, ECHO_PIN, distance); //Ultra-son

// KEY-PAD
const byte ROWS = 4;
const byte COLS = 3;
char keys[ROWS][COLS] = {
  {'1', '2', '3'},
  {'4', '5', '6'},
  {'7', '8', '9'},
  {'*', '0', '#'}
};
byte rowPins[ROWS] = {7, 8, 9, 10};
byte colPins[COLS] = {11, 12, 13};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

const int trigPin = 6;  // Broche de déclenchement
const int echoPin = 4; // Broche de retour
```

- Wire.h : Bibliothèque pour la communication I2C.

- LiquidCrystal_I2C.h : Bibliothèque pour les écrans LCD utilisant l'interface I2C.

- Keypad.h : Bibliothèque pour la gestion des claviers matriciels.

- NewPing.h : Bibliothèque pour les capteurs à ultrasons.

- Stepper.h : Bibliothèque pour le contrôle des moteurs pas à pas.

- SPI.h : Bibliothèque pour la communication SPI.

- MFRC522.h : Bibliothèque pour le contrôle des lecteurs RFID

- TRIGGER_PIN et ECHO_PIN : Broches utilisées par le capteur à ultrasons.

- distance : Distance maximale mesurée par le capteur à ultrasons.

- RELAY_1_PIN et RELAY_2_PIN : Broches pour commander les relais.

- lcd : Objet pour contrôler l'écran LCD.

- sonar : Objet pour contrôler le capteur à ultrasons.

- ROWS et COLS : Nombre de lignes et de colonnes du clavier.

- keys : Matrice représentant les touches du clavier.

- rowPins et colPins : Broches Arduino connectées aux lignes et colonnes du clavier.

- keypad : Objet pour contrôler le clavier matriciel.

```
33  const int stepsPerRevolution = 2048; // Pour le moteur pas à pas 28BYJ-48
34  Stepper myStepper(stepsPerRevolution, 22, 24, 26, 28); // (pas, pin1, pin2, pin3, pin4)
35
36  long duration;
37  int Distance;
38
39  // RFID
40  #define SS_PIN 53 // SDA pin of RFID
41  #define RST_PIN 5 // RST pin of RFID
42  MFRC522 rfid(SS_PIN, RST_PIN);
43
44  // List of authorized UIDs
45  byte authorizedUIDs[][4] = {
46    {0xF3, 0xFA, 0x20, 0xB7}, // Replace with actual UIDs
47    {0xF3, 0xDD, 0x0F, 0xAA}
48  };
49
```

- stepsPerRevolution : Nombre de pas nécessaires pour une révolution complète du moteur.

- myStepper : Objet pour contrôler le moteur pas à pas avec les broches spécifiées.

- duration : Durée de l'impulsion ultrasonique.

- Distance : Distance mesurée par le capteur ultrasonique.

- SS_PIN et RST_PIN : Broches pour le lecteur RFID.

- rfid : Objet pour contrôler le lecteur RFID.

- authorizedUIDs : Tableau contenant les UIDs autorisés pour les cartes RFID.

```
50  void setup() {
51    Serial.begin(9600);
52    lcd.init();
53    lcd.backlight();
54    pinMode(RELAY_1_PIN, OUTPUT); // Initialisation du relai 1
55    pinMode(RELAY_2_PIN, OUTPUT);
56    digitalWrite(RELAY_1_PIN, HIGH); // Initialisation du relai 2
57    digitalWrite(RELAY_2_PIN, HIGH); // Initialisation du relai 2
58
59    lcd.setCursor(0, 0);
60    lcd.print("Scan RFID card");
61
62    myStepper.setSpeed(5); // Réglage de la vitesse en tr/min, à ajuster selon les besoins
63    pinMode(trigPin, OUTPUT);
64    pinMode(echoPin, INPUT);
65
66    // Initialize SPI bus and MFRC522
67    SPI.begin();
68    rfid.PCD_Init();
69  }
70
```

- Serial.begin(9600) : Initialisation de la communication série.

- lcd.init() et lcd.backlight() : Initialisation de l'écran LCD et allumage du rétroéclairage.

- pinMode et digitalWrite : Configuration des broches des relais en sortie et désactivation initiale.

- lcd.setCursor et lcd.print : Affichage d'un message sur l'écran LCD.

- myStepper.setSpeed : Réglage de la vitesse du moteur pas à pas.

- pinMode pour trigPin et echoPin : Configuration des broches du capteur à ultrasons.

- SPI.begin et rfid.PCD_Init : Initialisation de la communication SPI et du lecteur RFID.

```
71  bool isAuthorizedUID(byte *uid) {
72    for (int i = 0; i < sizeof(authorizedUIDs) / sizeof(authorizedUIDs[0]); i++) {
73      bool match = true;
74      for (int j = 0; j < 4; j++) {
75        if (authorizedUIDs[i][j] != uid[j]) {
76          match = false;
77          break;
78        }
79      }
80      if (match) {
81        return true;
82      }
83    }
84    return false;
85  }
86
```

- La fonction isAuthorizedUID vérifie si l'UID (identifiant utilisateur) fourni appartient à une liste prédéfinie d'UID autorisés.

- Elle commence par initialiser une variable booléenne match à false.

- Ensuite, elle parcourt deux boucles imbriquées. La boucle extérieure itère sur chaque élément du tableau authorizedUIDs, tandis que la boucle intérieure compare chaque octet de l'UID actuel avec l'octet correspondant de l'UID d'entrée (uid).

- Si tous les octets correspondent pour un UID dans la liste, match est défini à true et les boucles sont interrompues.

- Après avoir quitté les boucles, si match reste true, cela signifie qu'une correspondance complète a été trouvée, et la fonction renvoie true. Sinon, elle renvoie false.

```arduino
void loop() {
  if (!rfid.PICC_IsNewCardPresent() || !rfid.PICC_ReadCardSerial()) {
    return;
  }

  // Check if the card is authorized
  if (!isAuthorizedUID(rfid.uid.uidByte)) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Unauthorized");
    lcd.setCursor(0, 1);
    lcd.print("Scan again");
    delay(2000);
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Scan RFID card");
    rfid.PICC_HaltA();
    rfid.PCD_StopCrypto1();
    return;
  }

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Selectionnez");
  lcd.setCursor(0, 1);
  lcd.print("votre boisson:");
  rfid.PICC_HaltA();
  rfid.PCD_StopCrypto1();

  char key = 0;
  while (!key) {
    key = keypad.getKey();
```

```arduino
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Boisson ");
    lcd.print(key);

    switch (key) {
      case '1':
        controlPumps(100, 0);
        break;
      case '2':
        controlPumps(50, 50);
        break;
      case '3':
        controlPumps(25, 75);
        break;
      case '4':
        controlPumps(25, 75);
        break;
      case '5':
        controlPumps(0, 100);
        break;
      default:
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Selection err");
        delay(2000);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("Scan RFID card");
        return;
    }
```

1. Boucle principale (void loop())

- Vérifie si une nouvelle carte RFID est présente et si elle peut être lue. Si aucune carte n'est détectée, la boucle se termine et recommence.

- Si une carte est détectée, le code vérifie si elle est autorisée à l'aide de la fonction isAuthorizedUID.

2. Vérification de la carte autorisée

- Si la carte n'est pas autorisée, affiche "Unauthorized" sur l'écran LCD et demande à l'utilisateur de scanner à nouveau après un délai de 2 secondes.

- Si la carte est autorisée, affiche "Selectionnez votre boisson:" sur l'écran LCD et arrête la communication avec la carte RFID.

3. Sélection de la boisson

- Attends jusqu'à ce qu'une touche soit pressée sur le clavier.

- Affiche la touche pressée sur l'écran LCD.

- Utilise une structure switch pour contrôler les pompes en fonction de la touche pressée.

4. Action des pompes

- Chaque case du switch appelle la fonction controlPumps avec des paramètres spécifiques pour contrôler les pompes.

5. Réinitialisation de l'affichage

- Réinitialise l'affichage LCD et demande à l'utilisateur de scanner à nouveau la carte RFID.

```
  delay(3000);

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Scan RFID card");

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);

  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  Distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.print(Distance);
  Serial.println(" cm");

  if (Distance < distance) {
    myStepper.step(stepsPerRevolution);
    delay(1000);
  }

  delay(6000);
}
```

6. Mesure de la distance avec un capteur à ultrasons

- Envoie une impulsion et mesure le temps que prend l'écho à revenir pour calculer la distance.

- Si la distance est inférieure à une valeur définie (distance), le moteur pas-à-pas (mystepper) effectue un pas.

7. Délai final

- Attends 6 secondes avant de redémarrer la boucle.

```
void controlPumps(int pump1, int pump2) {
  // Active les relais en fonction de la configuration des pompes
  if (pump1 > 0) {
    digitalWrite(RELAY_1_PIN, LOW);
    delay(pump1 * 50); // Convertit le temps en secondes en millisecondes
    digitalWrite(RELAY_1_PIN, HIGH); // Arrête la pompe 1
  }

  if (pump2 > 0) {
    digitalWrite(RELAY_2_PIN, LOW);
    delay(pump2 * 50); // Convertit le temps en secondes en millisecondes
    digitalWrite(RELAY_2_PIN, HIGH); // Arrête la pompe 2
  }

  // Re-display the initial message after pump operation
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Scan RFID card");
}
```

Contrôle des pompes

- Active les relais correspondant aux pompes 1 et 2 en fonction des paramètres pump1 et pump2.

- Utilise des délais pour contrôler la durée de fonctionnement des pompes.

- Réinitialise l'affichage LCD après l'opération des pompes.

Conclusion :

Ce projet porte sur la conception et la réalisation d'une machine de remplissage automatisée et sécurisée, basée sur la technologie Arduino. L'objectif principal est de développer une solution automatisée pour le remplissage de boissons, intégrant des fonctionnalités de sécurité et de sélection personnalisée.

Le système utilise un lecteur RFID pour sécuriser l'accès, un écran LCD pour l'interface utilisateur, un clavier pour la sélection des boissons, des pompes pour distribuer les liquides, et un capteur de distance pour assurer l'interaction de l'utilisateur avec la machine. Le code Arduino gère l'ensemble des opérations, depuis la vérification de l'authenticité des utilisateurs jusqu'au contrôle précis des pompes.

Le rapport détaille les composants matériels utilisés, incluant leurs spécifications techniques, ainsi que les schémas de câblage nécessaires pour assembler le système. La partie logicielle est également décrite en profondeur, avec des fragments de code et des explications pour chaque fonctionnalité implémentée.

Des tests rigoureux ont été effectués pour valider chaque aspect de la machine, depuis la reconnaissance RFID jusqu'au contrôle des pompes. Les résultats des tests montrent que le système fonctionne de manière fiable et efficace, bien que certaines améliorations puissent encore être apportées pour optimiser les performances.

En conclusion, ce projet démontre la faisabilité de créer une machine de remplissage automatisée sécurisée avec des composants abordables et une programmation adéquate. Les perspectives futures incluent l'amélioration de l'interface utilisateur, l'augmentation de la capacité de stockage de boissons, et l'intégration de nouvelles fonctionnalités pour accroître l'automatisation et la sécurité du système.