



Systeme d'Optimisation des Menus des Sportifs

ANNEXES



Nom et Prénom

N° SCEI

Session

**SOUADI
Nouamane**

18516

2024

Liste des figures

page4

1. Performances des sportifs dans les académies du sport
2. Variété des besoins selon le sexe, l'âge...

page5

3. Menus non personnalisés causent aussi un gaspillage alimentaire
4. Food Waste Index report, objectif: réduire le gaspillage alimentaire

page7

5. Parties du système

page8

6. Organigramme du principe de fonctionnement

page10

7. Diagramme de cas d'utilisation

page11

8. Diagramme des blocs internes

page12

9. Diagramme des exigences

page15

10. Emplacement du Raspberry pi
11. Étapes de réalisation du traitement d'image

page16

12. Principe de détection YOLO
13. Architecture interne de YOLO

page18

14. Ensemble des images (base des données)
15. Comparaison des versions de YOLO
16. Faire apprendre le modèle

page19

17. Code d'apprentissage python
18. Contenu du fichier data.yaml
19. Courbes significatives après entraînement

page20

20. Validité du modèle

page21

21. Code de détection en temps réel
22. Test sur des nouvelles entrées

page22

23. Résultats de détection sur Raspberry pi

page24

24. Importance de l'asservissement
25. Asservissement de la position angulaire du support

page25

26. Cas défavorable du chargement

page26

27. Modèle équivalent du moteur
28. Identification du moteur [Code A1](#)

page27

29. Acquisitions pour plusieurs tensions d'échelon [Code A2](#)
30. Utilisation du capteur rotatif KY-040

31. Détermination du rapport de réduction [page28](#)

32. Régression pour l'inductance [Code A3](#)

33. Régression de plusieurs échelons [page29](#)

34. Validation du modèle avec Scilab [page30](#)

35. Critères à vérifier pour l'asservissement

36. Bloc d'asservissement global [page31](#)

37. Bloc d'asservissement sans perturbation

38. Réponse sans correction [page32](#)

39. Structure interne du PI

40. Organigramme de détermination de Kp et Ki [page33](#)

41. Bode sans correction

42. Bode après application de Kp [page34](#)

43. Bode après correction

44. Réponse du modèle avec perturbation

45. Réponse du modèle sans perturbation [page35](#)

46. Asservissement de position angulaire réellement

47. Organigramme simplifié d'asservissement de position angulaire [Code A4](#)

page36

48. Réponse du système sans perturbation [page37](#)

49. Réponse du système avec perturbation [page39](#)

50. RFID-RC522 et son tag [page40](#)

51. Constituants de RFID-RC522

52. Fréquences d'utilisation des RFID [page41](#)

53. Montage expérience arduino

54. Résultats [Code A5](#)

55. Principe de communication entre arduino et python [page43](#)

56. Principe global de fonctionnement GUI [page44](#)

57. Structure de la base des données des adhérents et leurs plats [page45](#)

58. Parties du GUI [page46](#)

59. Résultats personnalisés pour chaque adhérent [Code A6](#)

page48

60. Réalisation du prototype

A1 : code d'identification

```
// Déclaration des broches du codeur incrémental
const int pinA = 2; // Broche A du codeur connectée à la broche 2 d'Arduino (interrupt 0)
const int pinB = 3; // Broche B du codeur connectée à la broche 3 d'Arduino
const int IN2 = 10;
const int IN1 = 11;

// Variables pour le comptage des impulsions du codeur
int pulseCount = 0; // Compteur d'impulsions
int previousPulseCount = 0; // Nombre d'impulsions précédent pour le calcul de la vitesse
unsigned long previousMillis = 0; // Temps précédent pour le calcul du délai
unsigned long intervalle = 50000; // Intervalle de temps pour le calcul de la vitesse (en Microsecondes)
int pulseParRevolution = 48;

void setup() {
    // Initialisation des broches du codeur
    pinMode(pinA, INPUT);
    pinMode(pinB, INPUT);
    pinMode(IN1, OUTPUT);
    pinMode(IN2, OUTPUT);

    // Activation de l'interruption pour la broche A (interrupt 0)
    attachInterrupt(digitalPinToInterrupt(pinA), countPulse, RISING);
    attachInterrupt(digitalPinToInterrupt(pinB), countPulse, RISING);
    attachInterrupt(digitalPinToInterrupt(pinA), countPulse, FALLING);
    attachInterrupt(digitalPinToInterrupt(pinB), countPulse, FALLING);

    // Démarrage de la communication série
    Serial.begin(9600);
    digitalWrite(IN1, 1);
    digitalWrite(IN2, 0);
    Serial.println("t:w(t)");
    Serial.println("0;0");
}

void loop() {
    unsigned long currentMillis = micros();

    // Calcul du temps écoulé depuis le dernier calcul de vitesse
    if (currentMillis - previousMillis >= intervalle) {
        // Enregistrement du nombre d'impulsions pendant l'intervalle
        float A = pulseCount - previousPulseCount;
        // Calcul de la vitesse en impulsions par seconde
        float speed = 2*PI*A / intervalle / pulseParRevolution * 1000000.0; // rad/s

        // Affichage de la vitesse
        Serial.print(currentMillis);
        Serial.print(";");
        Serial.println(speed);

        // Mise à jour des variables pour le prochain calcul de vitesse
        previousMillis = currentMillis;
        previousPulseCount = pulseCount;
    }
}

void countPulse() {
    // Augmentation du compteur d'impulsions à chaque interruption
    pulseCount++;
}
```

A2 : régression linéaire de K et C_m

regression U(w).py > ...

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 data = pd.read_csv("ressources/liste U(w).txt", delimiter=";")
6 x, y = np.array(data['w']), np.array(data["U"])
7 [a, b] = np.polyfit(x, y, 1)
8 def fct(x):
9     return a*x + b
10 R = 9.3 # Par multimètre
11 Cm = b*a/R
12 print(f"a = K = {a} V.s/rad, Cm = b.K/R = {Cm} N.m")
13
14 x2 = np.linspace(min(x), max(x), 2)
15
16 plt.plot(x, y, 'o', label="acquisition U(w)")
17 plt.plot(x2, fct(x2), label="polyfit")
18 plt.title(f"$K=a={round(a, 2)}$ V.s/rad, $C_m=b.K/R={round(Cm, 3)}$ N.m")
19 plt.legend()
20 plt.grid()
21 plt.show()
```

A3 : code de régression de L

regression L.py > ...

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from scipy.optimize import curve_fit
5
6 """
7 Forme  $i(t) = [1 - \exp(-t/\tau)] I_{per}$  avec  $I_{per} = U/Re$ 
8  $Re = R_m + 220$ 
9 """
10 Re = 9.3 + 220
11 data = pd.read_csv("ressources/acquisitionL.txt", delimiter=",")
12 t, u = np.array(data["t"]), np.array(data["Ur(t)"])
13 i = u/Re
14
15 indice0 = 0
16 while t[indice0] < 0:
17     indice0 += 1
18 t = t[indice0:]
19 i = i[indice0:]
20
21 def f(t, Iper, tau):
22     return Iper*(1-np.exp(-t/tau)) # tau = L/Re
23 [Iper, tau], val = curve_fit(f, t, i)
24 plt.plot(t*1e3, i*1e3, label="acquisition")
25 plt.plot(t*1e3, f(t, Iper, tau)*1e3, label="curve_fit")
26 plt.xlabel("t (ms)")
27 plt.ylabel("i(t) (mA)")
28 L = Re*tau
29 plt.title(f"L = Re/tau = {round(L, 3)}H")
30 print(f"L= {L} H")
31 plt.show()
```

**élimination de la partie
négative du temps**

A4 : code d'asservissement (arduino)

```
// Déclaration des broches du codeur incrémental
const int pinA = 2; // Broche A du codeur connectée à la broche 2 d'Arduino (interrupt 0)
const int pinB = 3; // Broche B du codeur connectée à la broche 3 d'Arduino
const int IN2 = 10;
const int IN1 = 11;
const int ENA = 9;

// Variables pour le comptage des impulsions du codeur
volatile int pulseCount = 0; // Compteur d'impulsions
int previousPulseCount = 0; // Nombre d'impulsions précédent pour le calcul de la vitesse
unsigned long previousMillis = 0; // Temps précédent pour le calcul du délai
unsigned long interval = 50000; // Intervalle de temps pour le calcul de la vitesse (en Microsecondes)
int pulseParRevolution = 48;

// Variable asservissement
const float kp = 4310;
const float ki = 4310/1.32;
const int max_pwm = 255;
const int teta_c = 180;
float teta = 0;
float erreur = 0;
float somme_erreur = 0;
float speed = 0;
float var = 0;
const float r = 0.057438;

void setup() {
  // Initialisation des broches du codeur
  pinMode(pinA, INPUT);
  pinMode(pinB, INPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(ENA, OUTPUT);

  // Activation de l'interruption pour la broche A (interrupt 0)
  attachInterrupt(digitalPinToInterrupt(pinA), countPulse, RISING);
  attachInterrupt(digitalPinToInterrupt(pinB), countPulse, RISING);
  attachInterrupt(digitalPinToInterrupt(pinA), countPulse, FALLING);
  attachInterrupt(digitalPinToInterrupt(pinB), countPulse, FALLING);

  // Démarrage de la communication série
  Serial.begin(9600);
  Serial.println("t;teta(t);wr(t);erreur;pwm");
  Serial.print("0;0.00;0.00;");
  Serial.print(teta_c-teta);
  Serial.println(";0.00");
}

void loop() {
  unsigned long currentMillis = micros();

  // Calcul du temps écoulé depuis le dernier calcul de vitesse
  if (currentMillis - previousMillis >= interval) {
    // bloc d'asservissement
    teta += speed * interval / 1000000.00;
    erreur = teta_c - teta;
    somme_erreur += erreur * interval / 1000000.00;
    var = kp * erreur + ki * somme_erreur;

    // Enregistrement du nombre d'impulsions pendant l'intervalle
    float A = pulseCount - previousPulseCount;
    // Calcul de la vitesse en impulsions par seconde
    speed = 360*(float)A / (float)interval / pulseParRevolution * 1000000.0; // °/s
  }
}
```

```
speed *= r; // sortie reducteur
```

```
if (var > 0) {
    digitalWrite(IN1, 1);
    digitalWrite(IN2, 0);
} else {
    digitalWrite(IN1, 0);
    digitalWrite(IN2, 1);
    speed *= -1;
    var *= -1;
}

if (var > max_pwm){
    var = max_pwm;
}

analogWrite(ENA, var);

// Affichage
Serial.print(currentMillis);
Serial.print(";");
Serial.print(teta);
Serial.print(";");
Serial.print(speed);
Serial.print(";");
Serial.print(teta_c-teta);
Serial.print(";");
Serial.println(var);

// Mise à jour des variables pour le prochain calcul de vitesse
previousMillis = currentMillis;
previousPulseCount = pulseCount;
}
}
```

```
void countPulse() {
    // Augmentation du compteur d'impulsions à chaque interruption
    pulseCount++;
}
}
```

A5: code RFID (arduino)

```
#include <SPI.h>
#include <MFRC522.h>
#define SS_PIN 10
#define RST_PIN 9

MFRC522 rfid(SS_PIN, RST_PIN); // Instance of the class
MFRC522::MIFARE_Key key;
byte nuidPICC[4];
void setup() {
    Serial.begin(9600);
    SPI.begin(); // Init SPI bus
    rfid.PCD_Init(); // Init MFRC522
}

void loop() {

    // Reset the loop if no new card present on the sensor/reader. This saves the entire process when idle.
    if ( ! rfid.PICC_IsNewCardPresent())
        return;

    // Verify if the NUID has been readed
    if ( ! rfid.PICC_ReadCardSerial())
        return;
    printHex(rfid.uid.uidByte, rfid.uid.size);
    Serial.println();
    // Halt PICC
    rfid.PICC_HaltA();
    // Stop encryption on PCD
    rfid.PCD_StopCryptol();
```

```

}
void printHex(byte *buffer, byte bufferSize) {
  for (byte i = 0; i < bufferSize; i++) {
    Serial.print(buffer[i] < 0x10 ? " 0" : " ");
    Serial.print(buffer[i], HEX);
  }
}
}

```

A6: code RFID (python)

GUI TIPE.py > afficher_adherent_et_repas

```

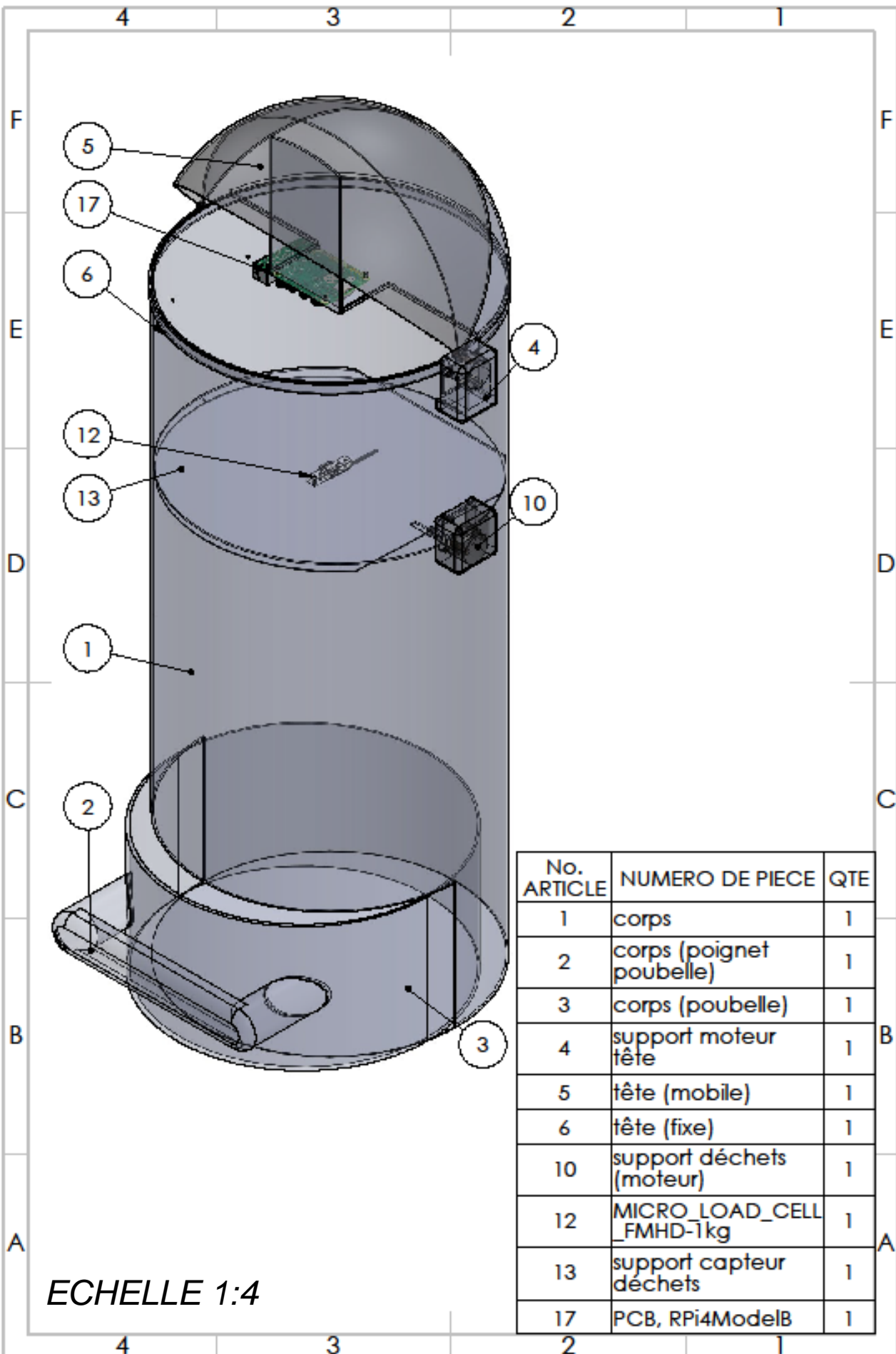
1  import tkinter as tk
2  from tkinter import ttk
3  from PIL import Image, ImageTk
4  import sqlite3
5  import serial
6
7  # Create the main window
8  fenetre = tk.Tk()
9
10 bg = "#99D9EA"
11 fenetre.configure(bg=bg)
12
13 # Calculate the coordinates to center the window on the screen
14 largeur_ecran = fenetre.winfo_screenwidth()
15 hauteur_ecran = fenetre.winfo_screenheight()
16 x = (largeur_ecran - 700) // 2
17 y = (hauteur_ecran - 700) // 2
18
19 # Position the window at the center of the screen
20 fenetre.geometry(f"700x700+{x}+{y}")
21
22 def afficher_adherent_et_repas(id, fenetre):
23     try:
24         # Connect to the database
25         conn = sqlite3.connect('sqlite_TIPE.db')
26         conn.row_factory = sqlite3.Row
27         cur = conn.cursor()
28     except sqlite3.Error as e:
29         print(f"Erreur de connexion à la base de données: {e}")
30         return
31
32     cur.execute("SELECT * FROM adherent WHERE id=?", (id,))
33     adherent = cur.fetchone()
34
35     if adherent is None:
36         print(f"Aucun adhérent trouvé avec cet ID: '{id}'")
37         conn.close()
38         return
39
40     print(f"ID détecté: '{id}'")
41
42     # Clear the previous contents of the window
43     for widget in fenetre.winfo_children():
44         widget.destroy()
45
46     # Create a frame to group the adherent and repas information
47     frame_principale = tk.Frame(fenetre, bg=bg)
48     frame_principale.pack(expand=True, fill="both")
49
50     # Add a title label
51     label_titre = tk.Label(frame_principale, text="Informations de l'adhérent et du repas", font=("Helvetica", 16),
52                             bg=bg)
53     label_titre.pack(pady=10)
54
55     # Add the adherent information
56     # Image de profil en haut à droite
57     img_profil = Image.open(adherent['img_profil'])

```

```

58 img_profil = img_profil.resize((100, 100))
59 img_profil_tk = ImageTk.PhotoImage(img_profil)
60 label_img_profil = tk.Label(frame_principale, image=img_profil_tk, bg=bg)
61 label_img_profil.pack(side="right", anchor="n", padx=10, pady=10)
62
63 # Nom et prénom à gauche de l'image de profil
64 label_nom_prenom = tk.Label(frame_principale, text=f"Nom: {adherent['nom']}\nPrénom: {adherent['prenom']}",
65                             font=("Helvetica", 14), bg=bg)
66 label_nom_prenom.pack(side="left", anchor="n", padx=10, pady=10)
67
68 # Add the repas information
69 # Table des items avec leurs quantités (centrée)
70 cur.execute("SELECT item, quantite FROM plat WHERE plat_id=?", (id,))
71 items_quantites = cur.fetchall()
72
73 # Créer une Treeview (table) avec des colonnes "Élément" et "Quantité"
74 table_items = ttk.Treeview(frame_principale, columns=("Element", "Quantite"), show="headings")
75 table_items.heading("Element", text="Élément")
76 table_items.heading("Quantite", text="Quantité")
77
78 for item_quantite in items_quantites:
79     table_items.insert("", "end", values=(item_quantite['item'], item_quantite['quantite']))
80
81 table_items.pack(side="bottom", anchor="center", pady=20)
82
83 # Image du plat centrée
84 cur.execute("SELECT plat_id, plat_img FROM plat WHERE plat_id=?", (id,))
85 plat = cur.fetchone()
86
87 if plat:
88     img_plat = Image.open(plat['plat_img'])
89     img_plat = img_plat.resize((200, 200))
90     img_plat_tk = ImageTk.PhotoImage(img_plat)
91     label_img_plat = tk.Label(frame_principale, image=img_plat_tk, bg=bg)
92     label_img_plat.pack(side="bottom", pady=20)
93
94 conn.close()
95 fenetre.update()
96
97 def read_rfid_data():
98     while True:
99         try:
100             rfid_data = arduino_serial_port.readline().decode('utf-8').strip()
101             afficher_adherent_et_repas(rfid_data, fenetre)
102         except serial.SerialException as e:
103             print(f"Erreur lecture du port série: {e}")
104
105
106 # Open the serial port
107 arduino_serial_port = serial.Serial('COM3')
108
109 # Start reading RFID data
110 read_rfid_data()
111
112 # Run the main loop
113 fenetre.mainloop()

```

No. ARTICLE	NUMERO DE PIECE	QTE
1	corps	1
2	corps (poignet poubelle)	1
3	corps (poubelle)	1
4	support moteur tête	1
5	tête (mobile)	1
6	tête (fixe)	1
10	support déchets (moteur)	1
12	MICRO_LOAD_CELL_FMHD-1kg	1
13	support capteur déchets	1
17	PCB, RPi4ModelB	1

ECHELLE 1:4