Système d'Optimisation des Menus des Sportifs

Nom et Prénom	N° SCEI	Session
SOUADI Nouamane	18516	2024

PLAN DE PRESENTATION

INTRODUCTION

CAHIER DES CHARGES

OBJECTIFS

PROTOTYPE

CONCLUSION

INTRODUCTION

-Contextualisation

-Problématique

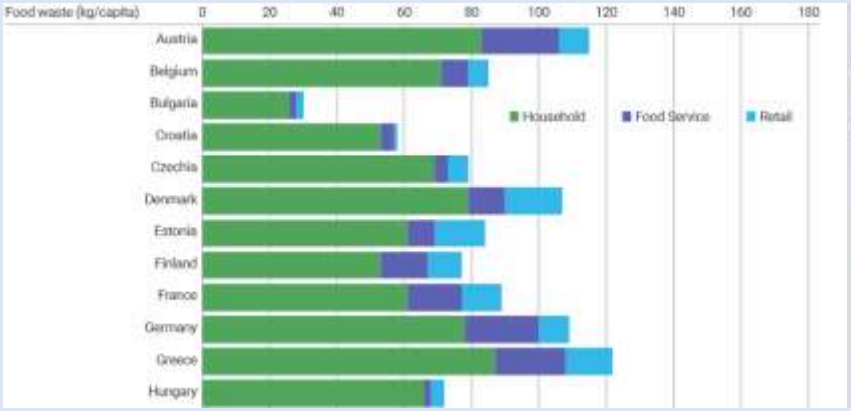
Figure 1: performances des sportifs dans les académies du sport

Âge (garçons)	Non actifs	Parfois actifs	Actifs
4 – 8 ans	1200–1400 calories	1400–1600 calories	1600–2000 calories
9 – 13 ans	1600–2000 calories	1800–2200 calories	2000–2600 calories
14 – 18 ans	2000–2400 calories	2400–2800 calories	2800–3200 calories
Âge (filles)	**Non actives**	**Parfois Actives**	**Actives**
4 – 8 ans	1200–1400 calories	1400–1600 calories	1400–1800 calories
9 – 13 ans	1600–2000 calories	1600–2000 calories	1800–2200 calories
14 – 18 ans	1800 calories	2000 calories	2400 calories

Figure 2: variété des besoins selon le sexe, l'âge…

INTRODUCTION

Figure 3: menus non personnalisés causent aussi un gaspillage alimentaire



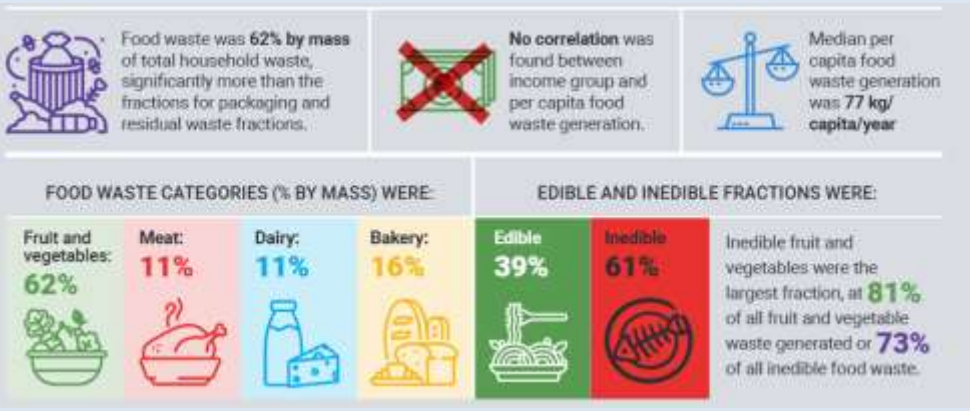


Figure 4: ***Food Waste Index report 2024****, objectif: réduire le gaspillage alimentaire*

PROBLEMATIQUE

- Comment détecter les déchets ?
- Comment identifier les adhérents ?
- Comment afficher les menus personnalisés ?

PRINCIPE DE FONCTIONNEMENT

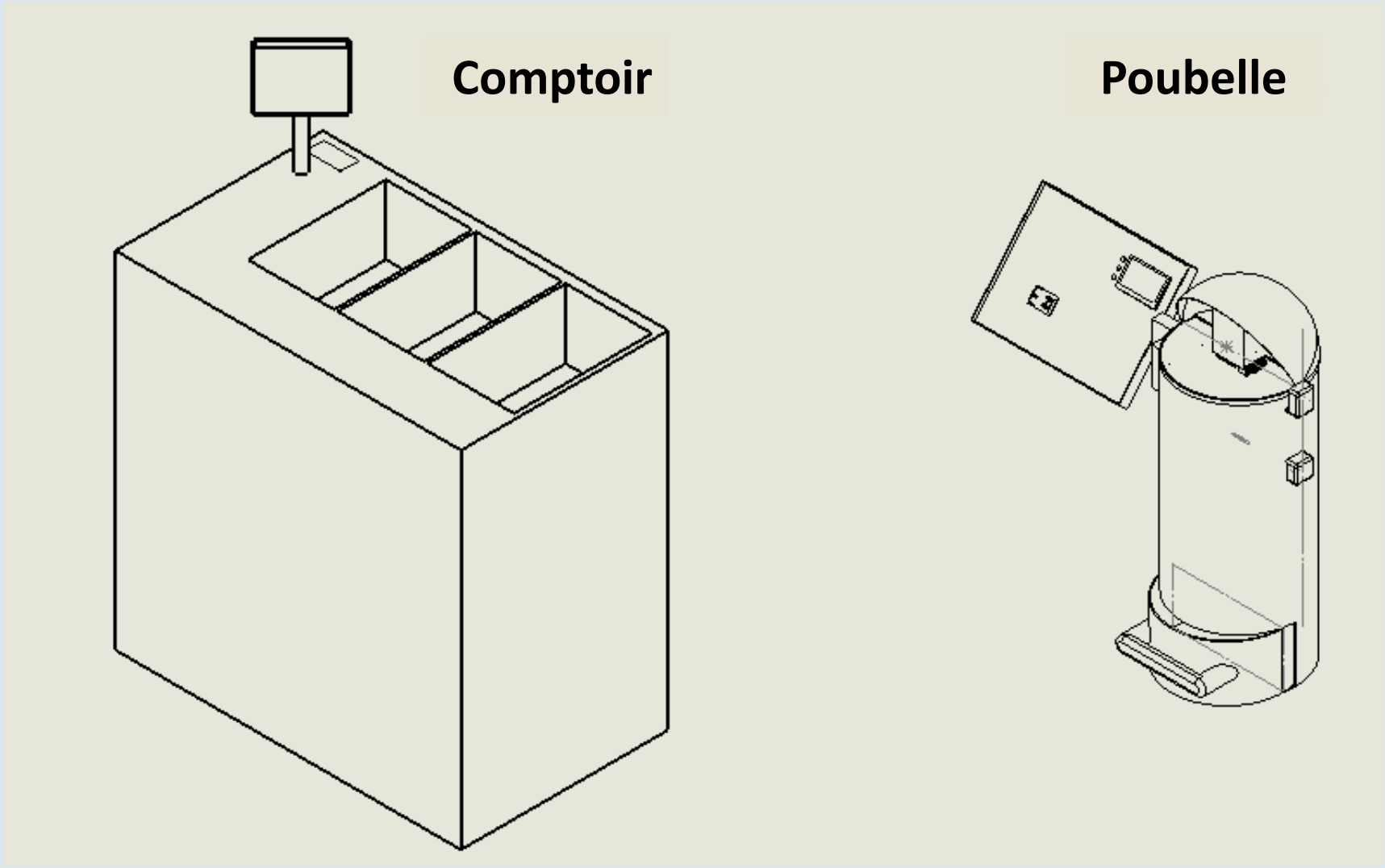


Figure 5: parties du système

PRINCIPE DE FONCTIONNEMENT

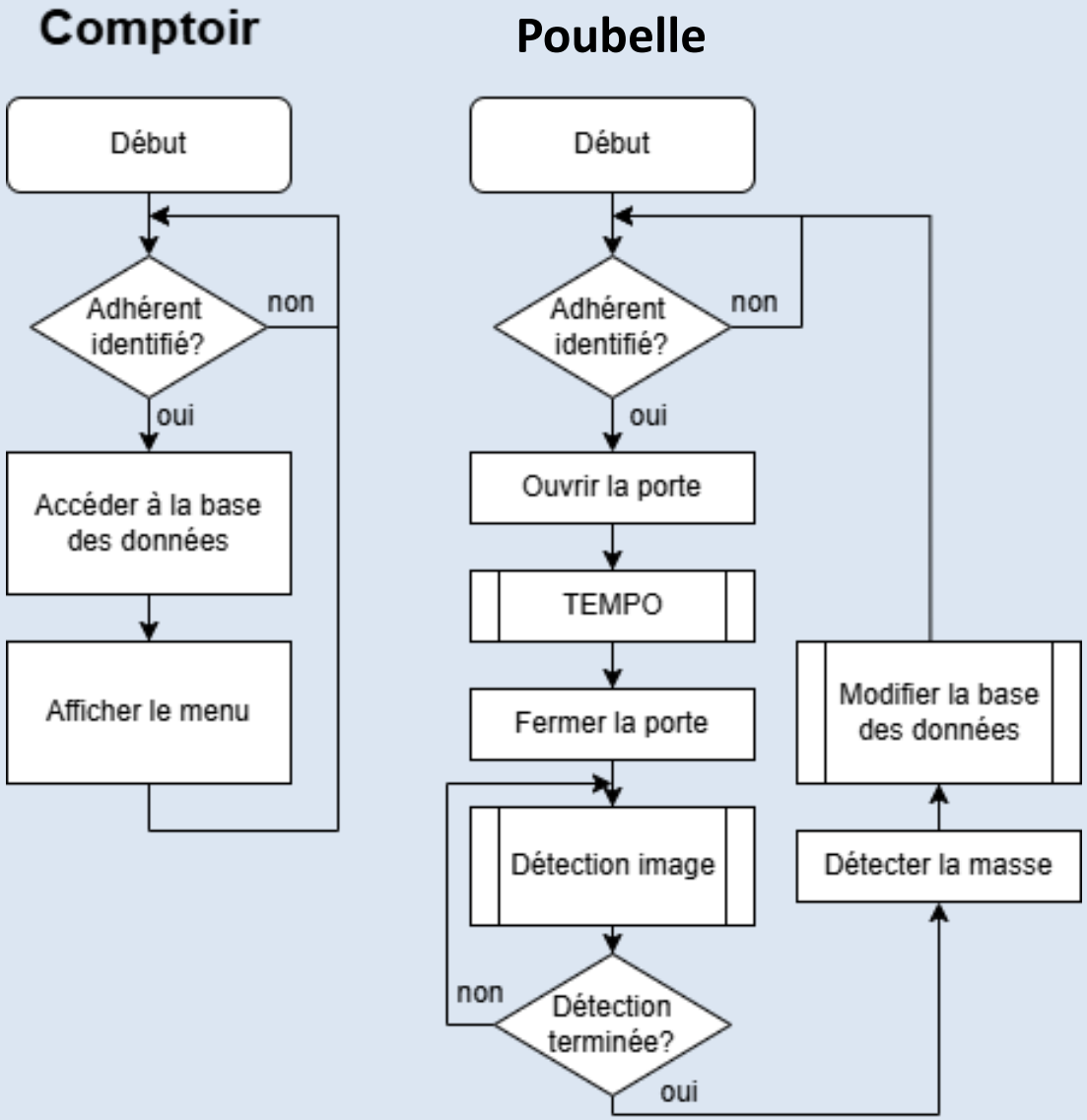


Figure 6: organigramme du principe de fonctionnement

CAHIER DES CHARGES FONCTIONNEL

-Diagramme de cas d'utilisation (**UC**)

-Diagramme des exigences (**REQ**)

-Diagramme des blocs internes (**IBD**)

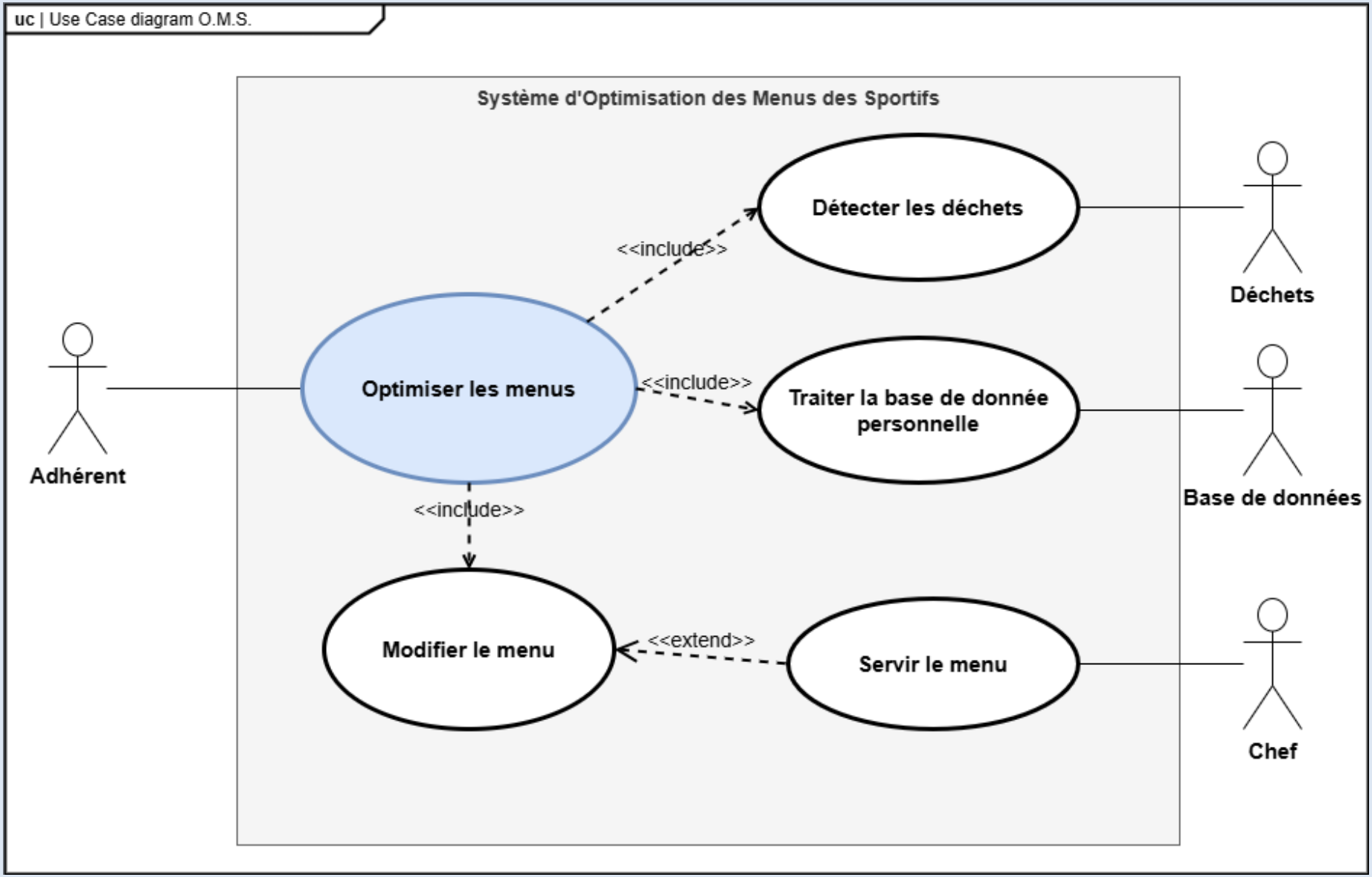


Figure 7: diagramme de cas d'utilisation

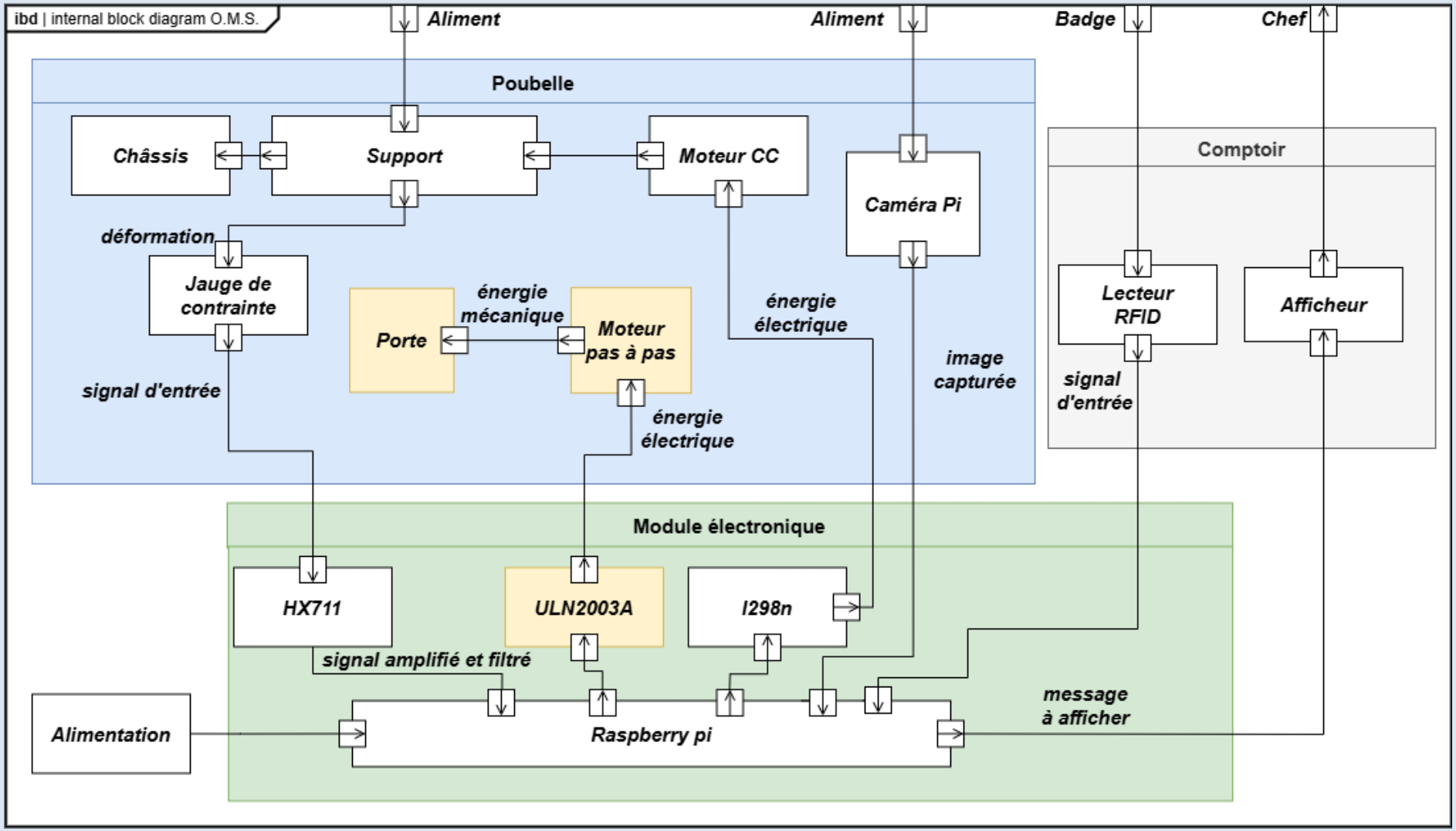


Figure 8: diagramme des blocs internes

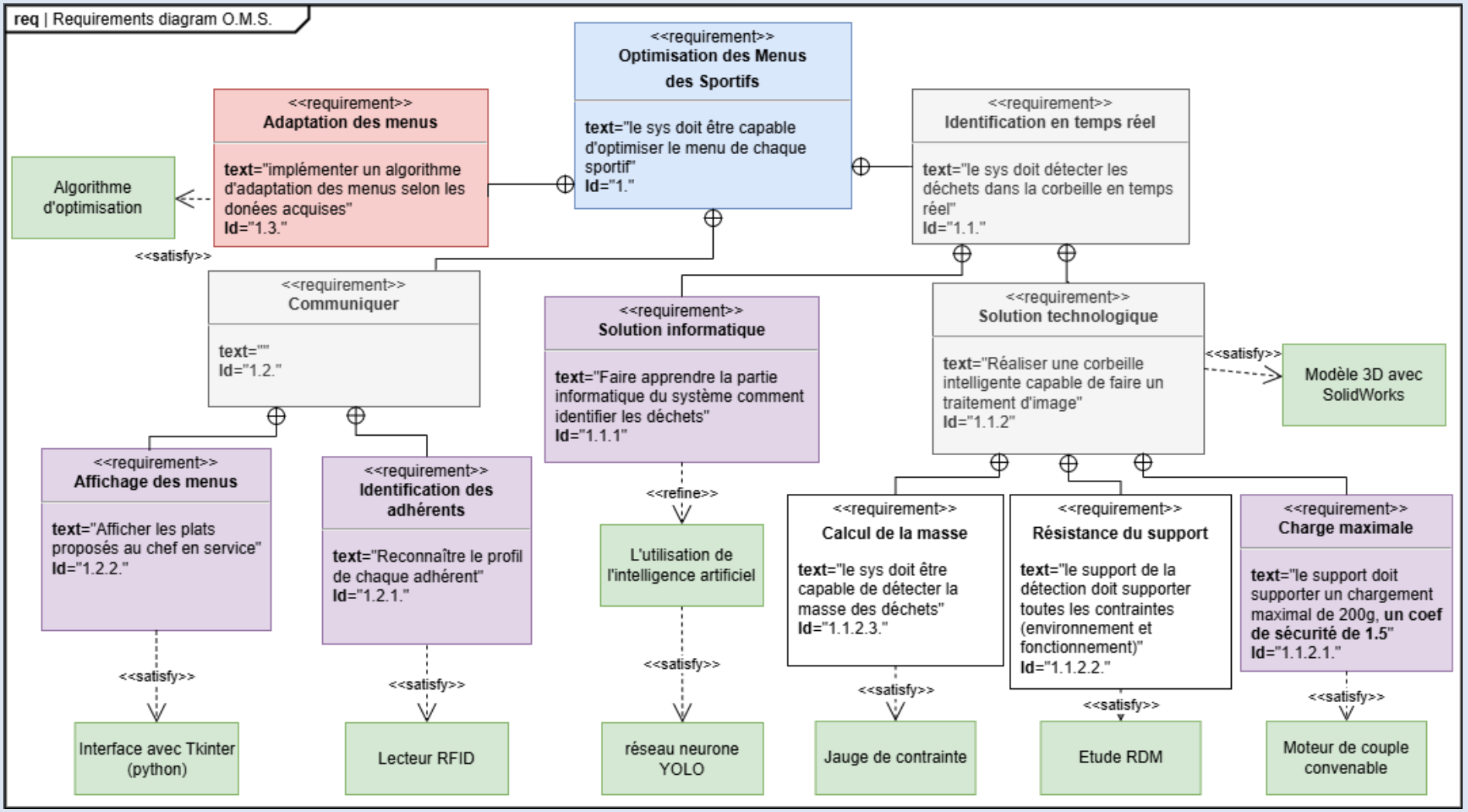


Figure 9: diagramme des exigences

OBJECTIFS

-1er Objectif:
Identification des déchets

-2ème Objectif:
Asservissement en position

-3ème Objectif:
Identification des adhérents

-4ème Objectif:
GUI avec Tkinter

1

Identification des déchets

Identifier les déchets versés par l’adhérent dans la poubelle

Principe

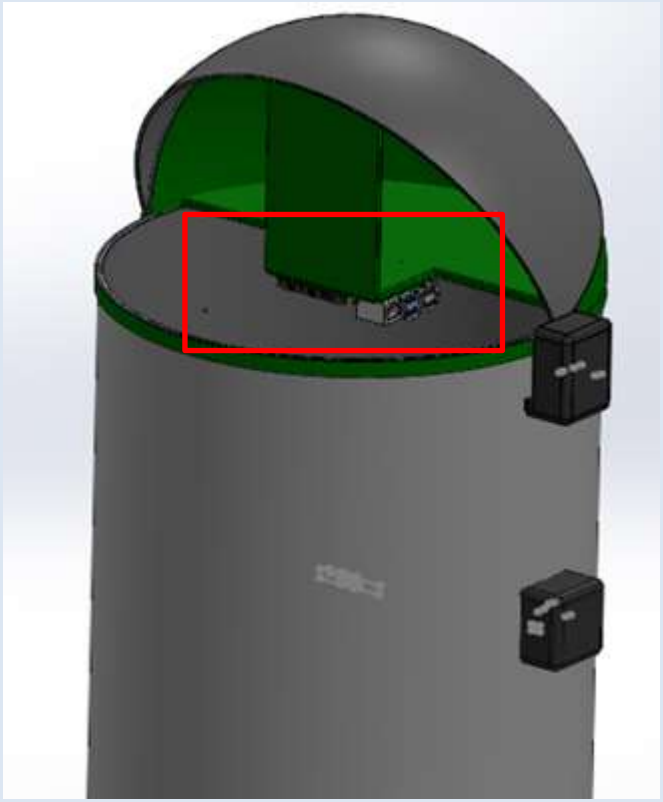


Figure 10: Emplacement du Raspberry pi

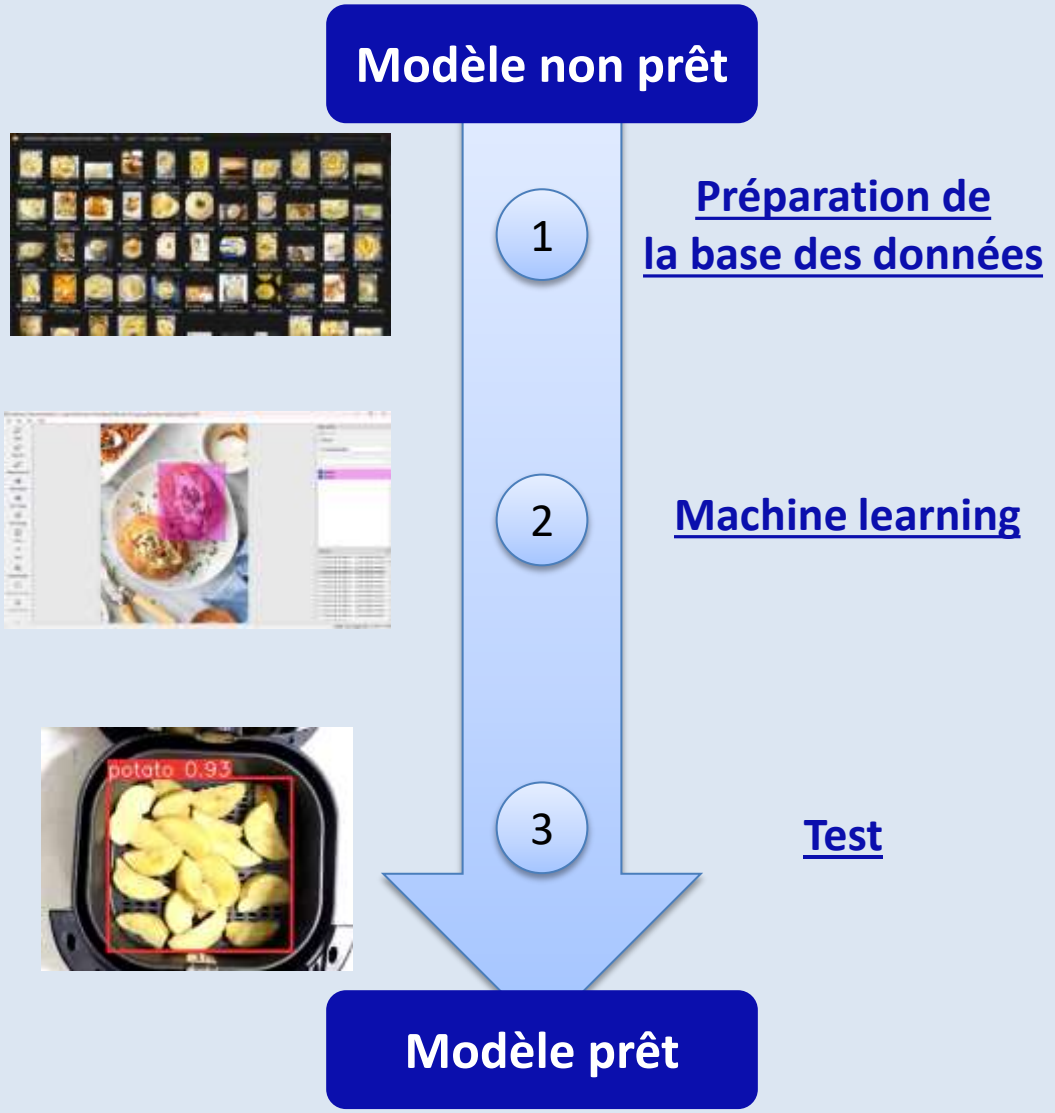
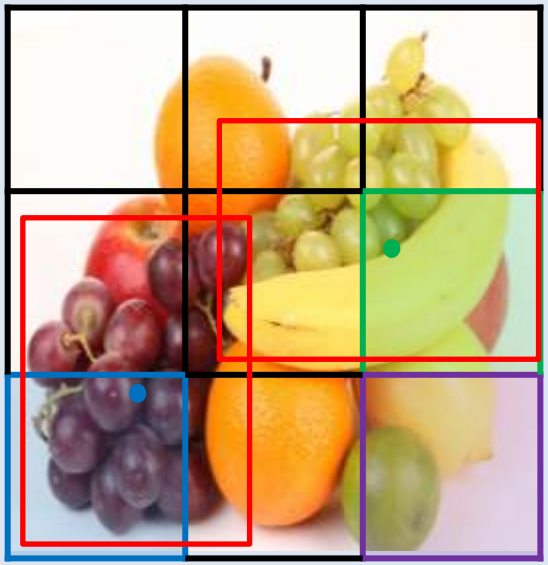


Figure 11: étapes de réalisation du traitement d'image

Principe de « You Only Look Once (YOLO) »



$$y = \begin{pmatrix} p_c \\ b_x \\ b_y \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ \vdots \end{pmatrix} \quad \begin{pmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ \vdots \end{pmatrix} \quad \begin{pmatrix} 1 \\ b_x \\ b_y \\ b_w \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix} \quad \begin{pmatrix} 1 \\ b_x \\ b_y \\ b_w \\ 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

$\boldsymbol{p_c}$ = booléen qui indique si la cellule possède un objet
$\boldsymbol{(b_x, b_y)}$ = les coordonnées du centre de l'objet
$\boldsymbol{(b_h, b_w)}$ = longueur et largeur du *bounding* box (rouge)
$\boldsymbol{c_i}$ = confidence du type d'objet de classe **i**

Figure 12: principe de détection YOLO

D'après: YOLO Object Detection Explained: A Beginner's Guide | DataCamp

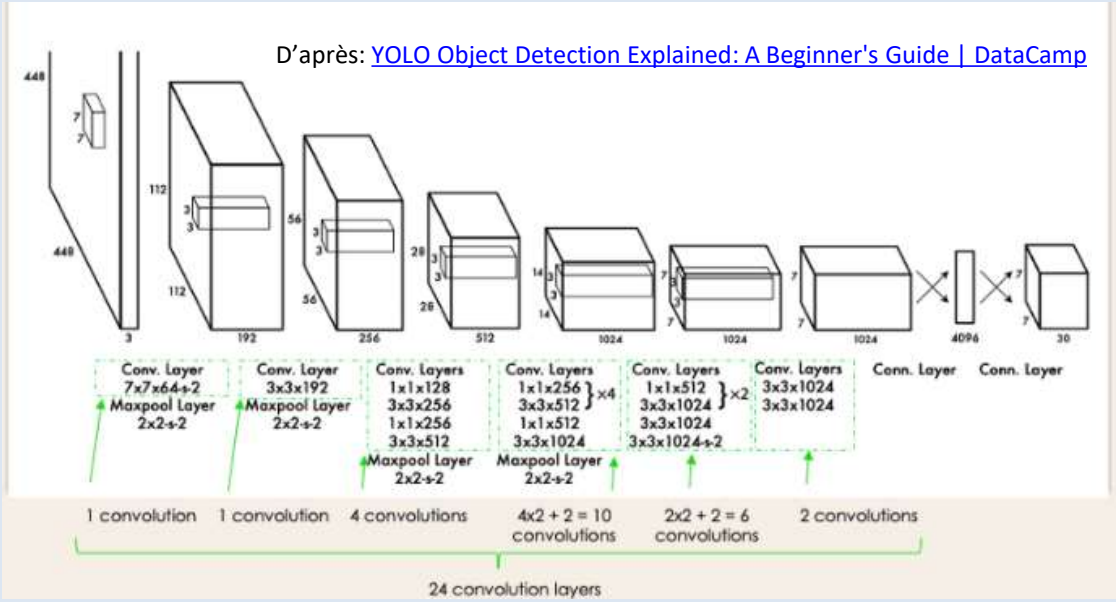


Figure 13: architecture interne de YOLO

Modèle utilisé

Préparation et machine learning

C'est **Facile** et **rapide** !!

Figure 14: ensemble des images (base des données)

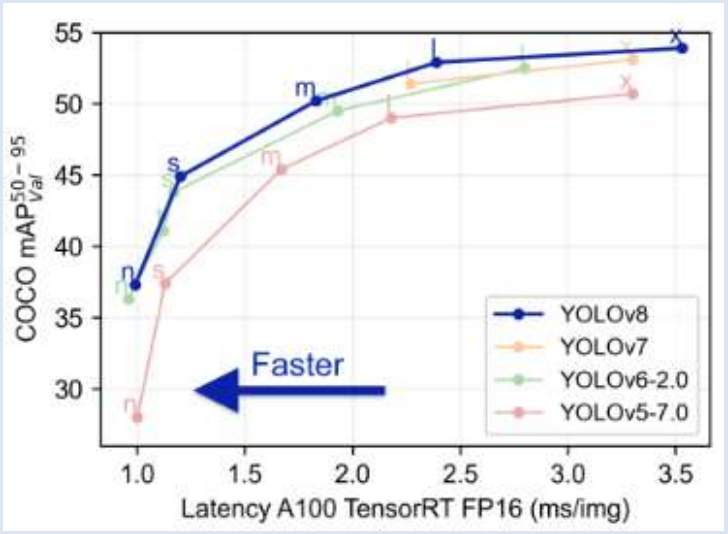


Figure 15: comparaison des versions de YOLO

Figure 16: faire apprendre le modèle

Résultats d'entrainement du modèle

C: > Users > hp > Documents > YOLOV8 > train.py > ...

```python
from ultralytics import YOLO
model = YOLO()
model.train(data="data.yaml", epochs=50)
```

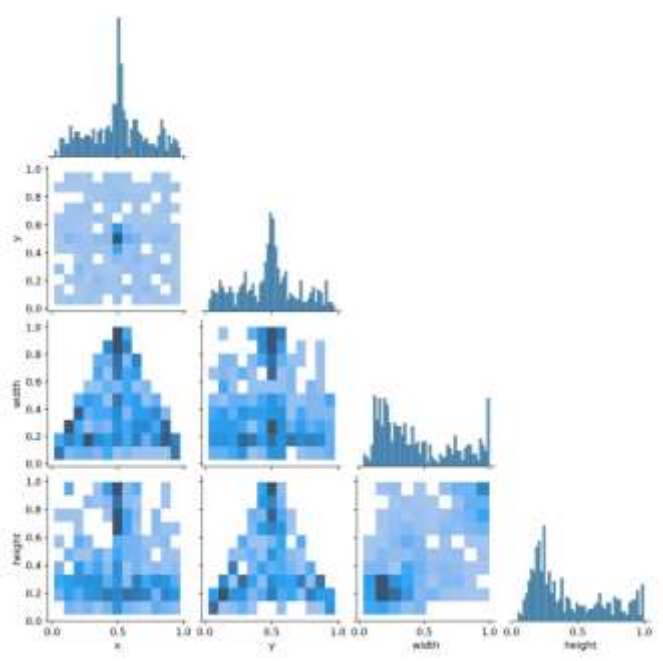
Figure 17: code d'apprentissage python

! data.yaml

```yaml
train: C:\Users\hp\Desktop\YOLOV8/train/images
val: C:\Users\hp\Desktop\YOLOV8/val/images

nc: 1
names: ['potato']
```

Figure 18: contenu du fichier ***data.yaml***



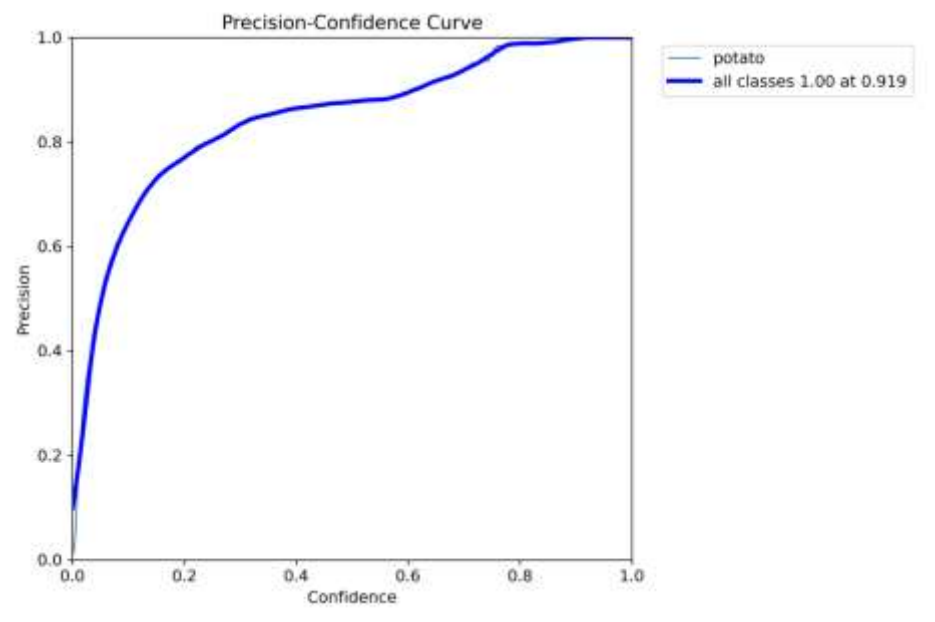


Figure 19: courbes significatives après entrainement

Comparaison entre réel et prédis après entrainement

Entrée (db)

Prédit

Figure 20: validité du modèle

Test sur des nouvelles entrées

```python
detect_vid_webcam.py > ...
1   from ultralytics import YOLO
2
3   model = YOLO("runs/detect/train3/weights/best.pt")
4   model(source=0, conf=0.5, show=True, save=True)
```

Figure 21: code de détection en temps réel

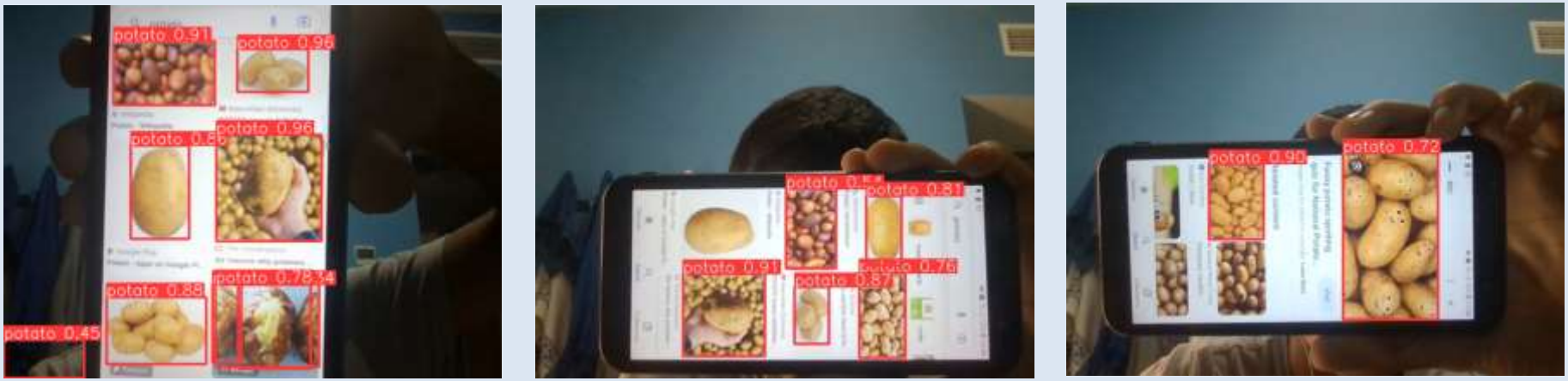


Figure 22: test sur des nouvelles entrées

Implémentation sur Raspberry pi

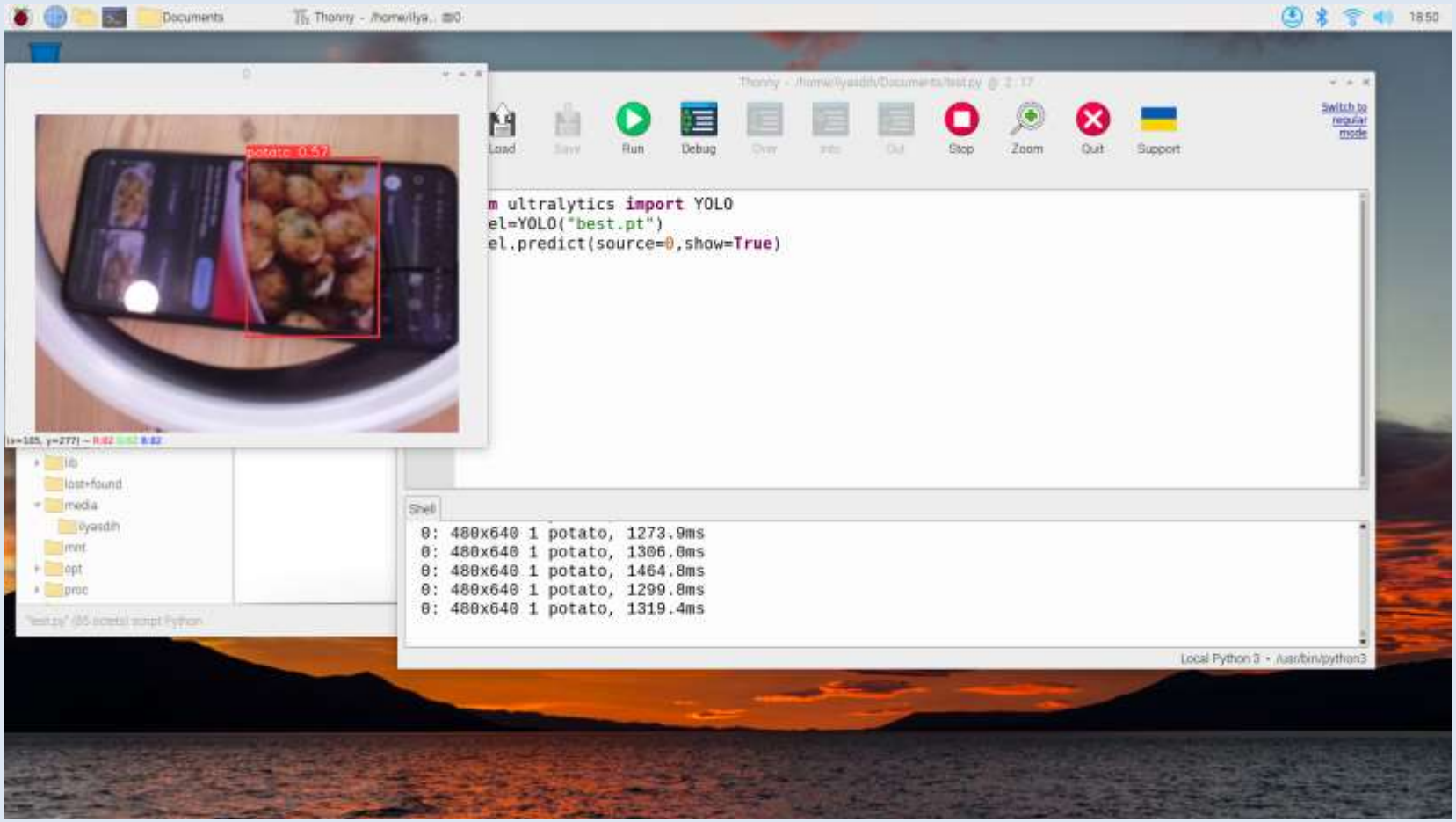
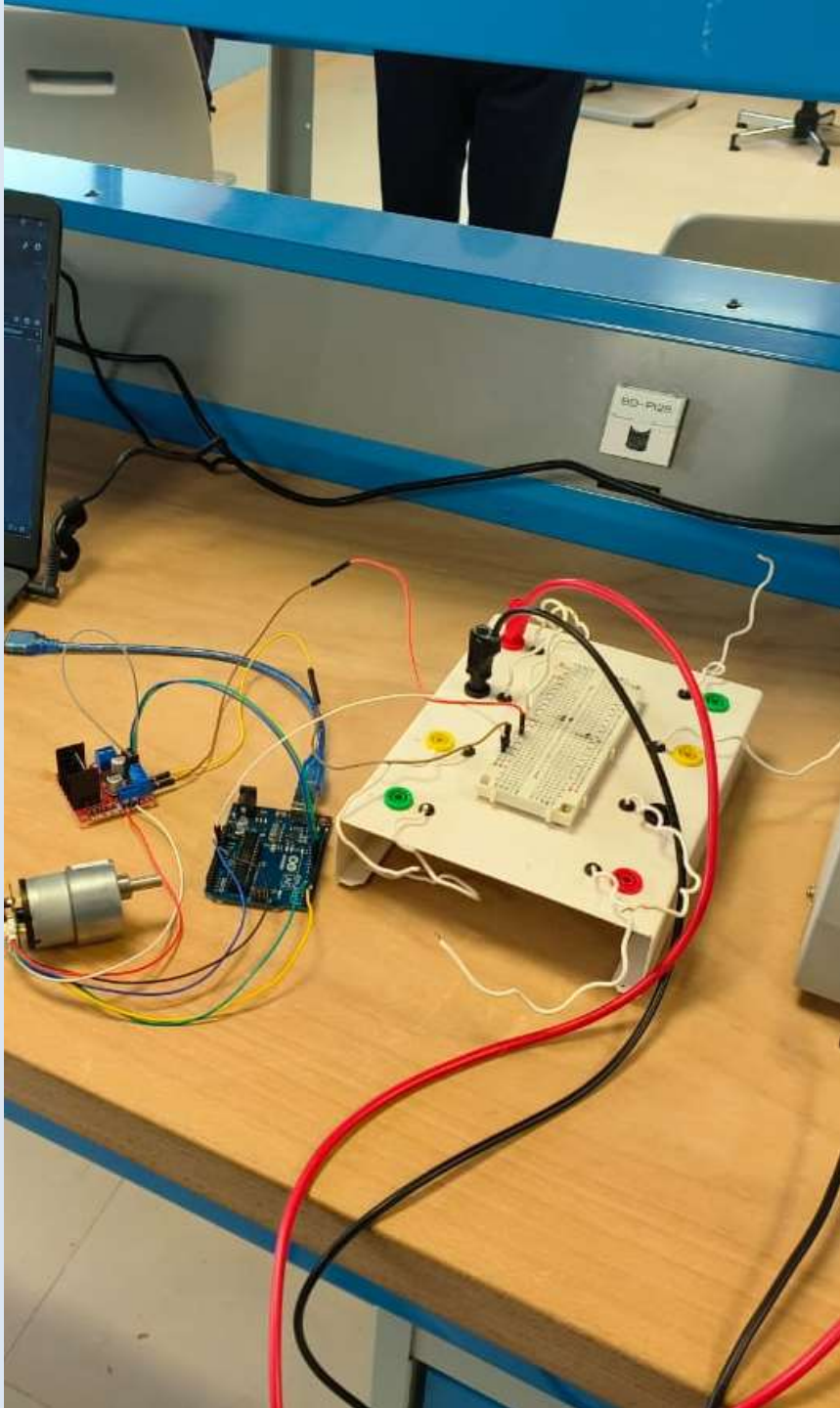


Figure 23: résultats de détection sur Raspberry pi

2

Asservissement en position

- Identifier le moteur CC
- Asservir la position angulaire du support des déchets



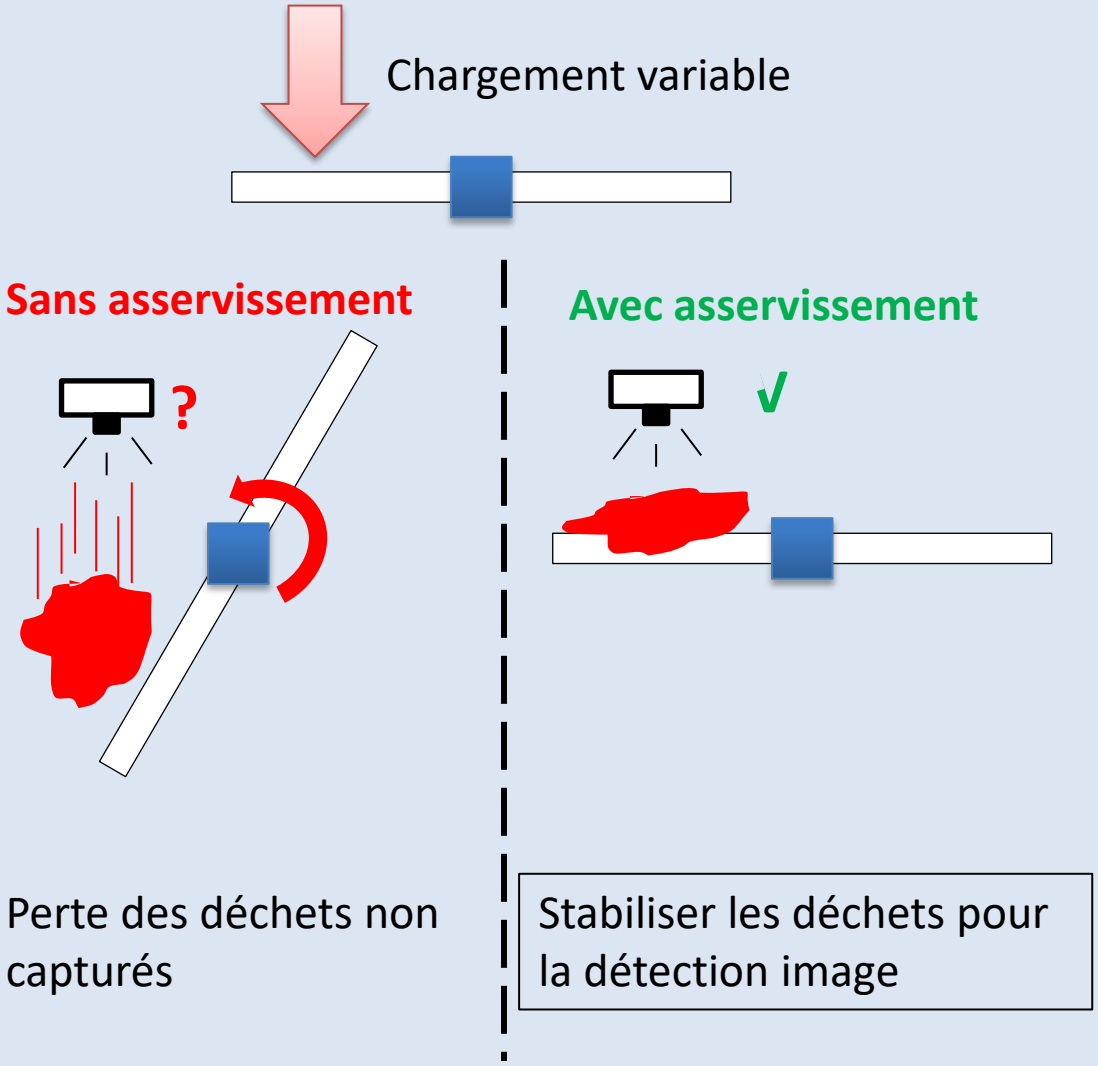


Figure 24: Importance de l'asservissement

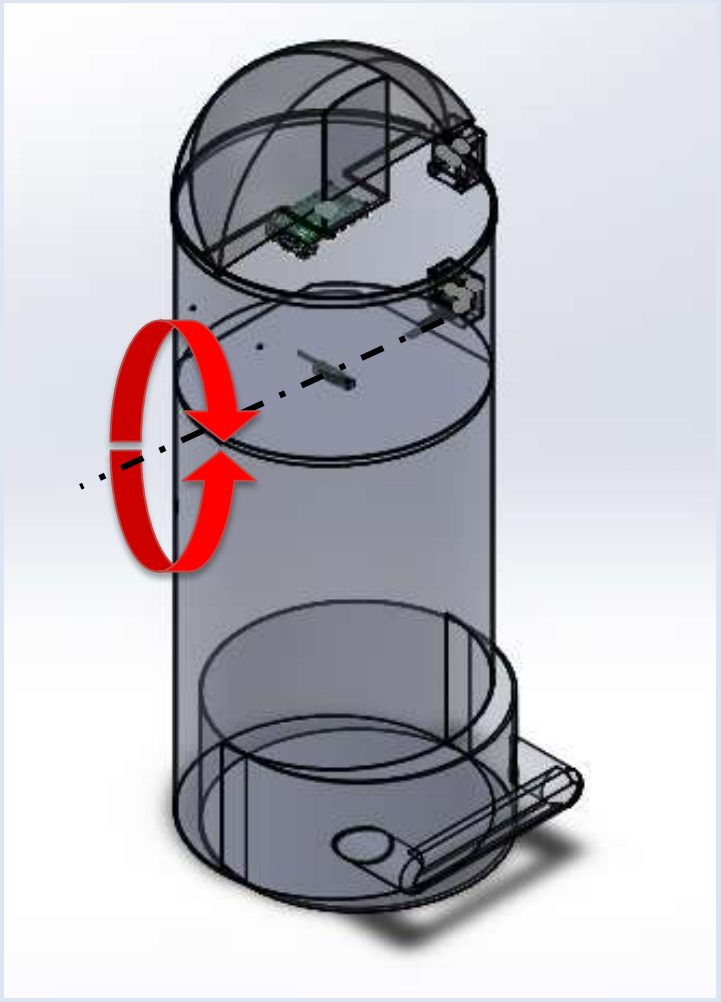


Figure 25: Asservissement de la position angulaire du support

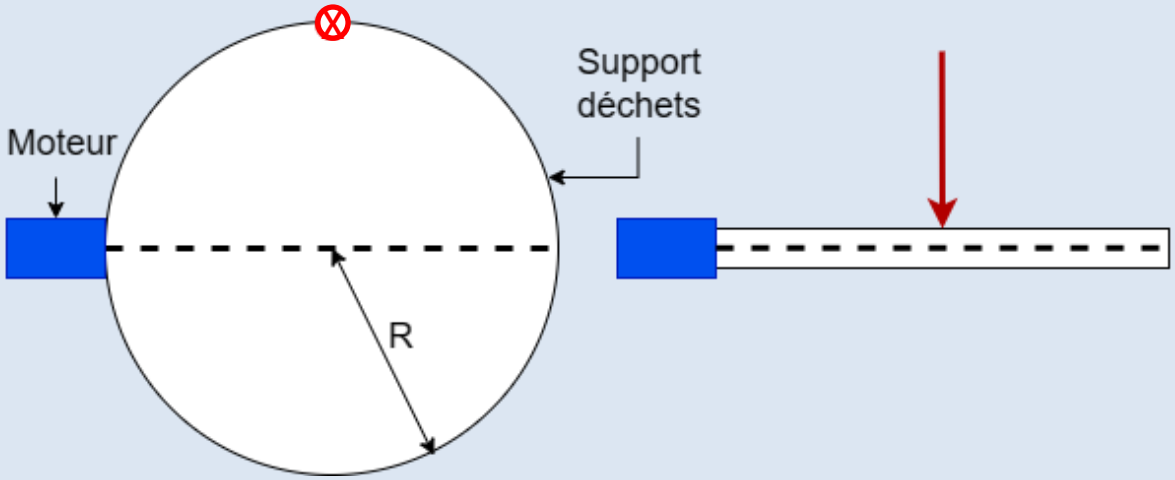


Figure 26: cas défavorable du chargement

Etude statique:

A l'équilibre:

$$C_0 = P * R = m * g * R = 0.3\ N.m \ \ (\mathrm{m} = 200\mathrm{g} \text{ et } \mathrm{R}{=}0{,}15\mathrm{m})$$

Pour vérifier le cahier des charges (référence 1.1.2.1.):

$$\boxed{C_0 \geq 0.45\ N.m}$$

Vérifier le choix du moteur

$$U = E + R.i + L\frac{di}{dt} \quad (1)$$

$$C_m = K.i \quad (2)$$

$$E = K.\omega_m \quad (3)$$

$$J_{eq}\frac{d\omega}{dt} = C_m - C_r \quad (4)$$

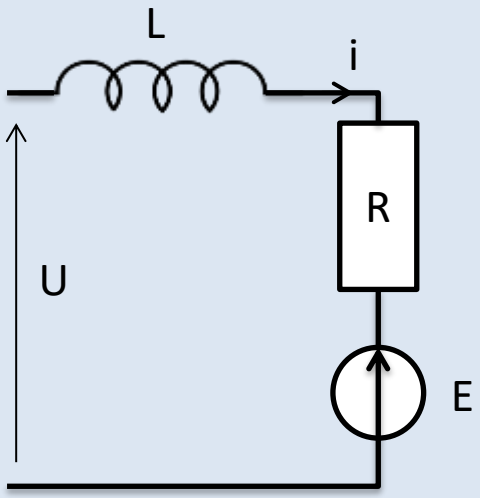


Figure 27: modèle équivalent du moteur JGB37-520

*Figure 28: Identification du moteur **Code A1***

Vérifier le choix du moteur

En régime permanent:

$$U = E + R.i = K.\Omega_m + R.\frac{C_m}{K}$$

$$\Rightarrow U(\Omega_m) = a.\Omega_m + b \quad \text{avec: } a = K, b = R.\frac{C_m}{K}$$

$$K = 0.18\ \text{V.s/rad} \ \text{ et } C_m = 0.034\ \text{N.m}$$

$$C_r = 0.59\ \text{N.m}$$

Moteur est convenable

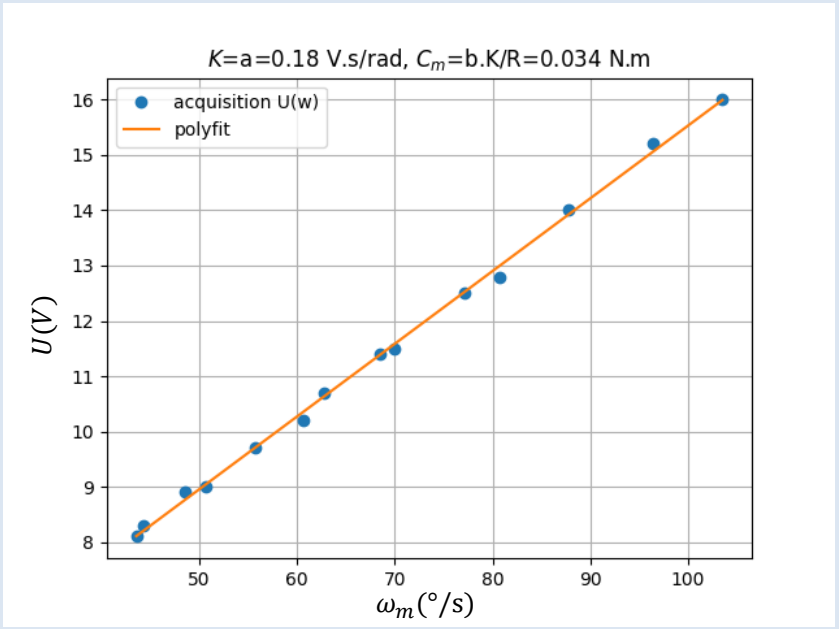


Figure 29: acquisitions pour plusieurs tensions d'échelon **Code A2**

Figure 30: utilisation du capteur rotatif KY-040

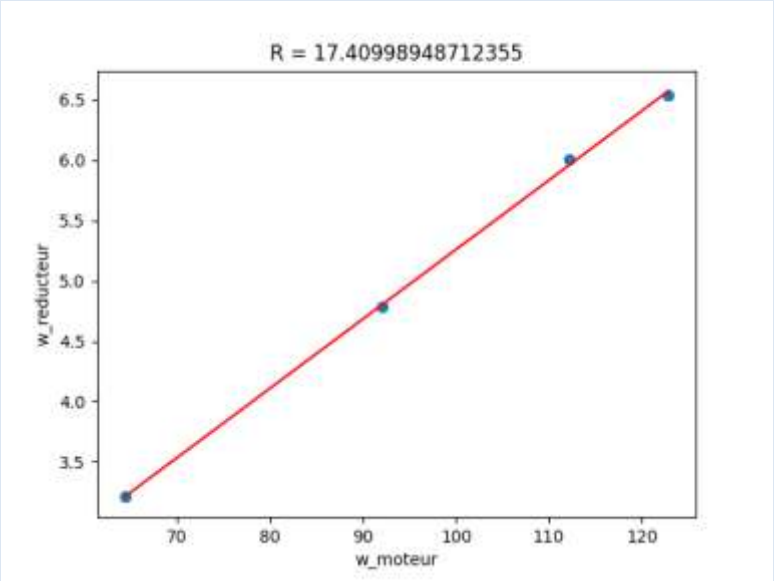
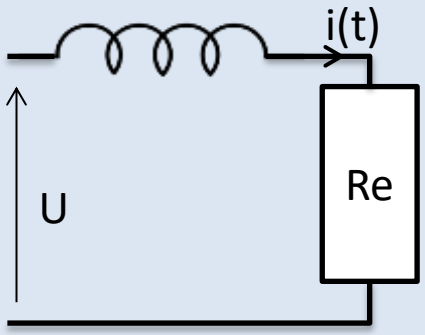


Figure 31: détermination du rapport de réduction

Identification des autres paramètres

En bloquant le moteur:

$$E = K.\omega_m = 0$$



$$\Rightarrow \quad U = R.i(t) + L.\frac{di}{dt}$$

$$\Rightarrow \quad \frac{di}{dt} + \frac{R}{L}.\frac{di}{dt} = \frac{U}{L}$$

$$\Rightarrow i(t) = I_{per}(1 - e^{-\frac{t}{\tau}})$$

avec: $I_{per} = \frac{U}{R}$ et $\tau = \frac{Re}{L}$ $\Rightarrow$ $L \approx 16\,mH$

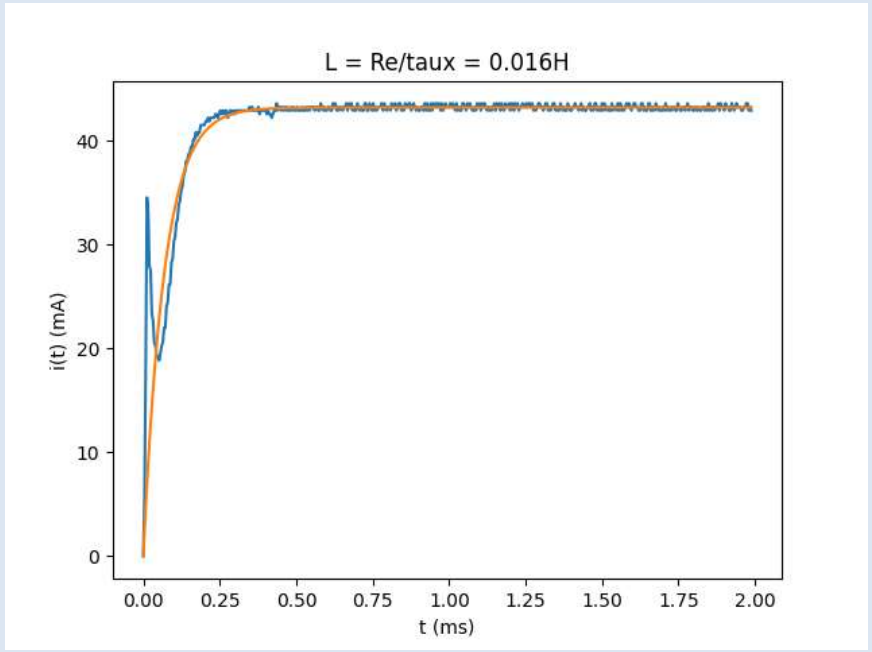


Figure 32: régression pour l'inductance ***Code A3***

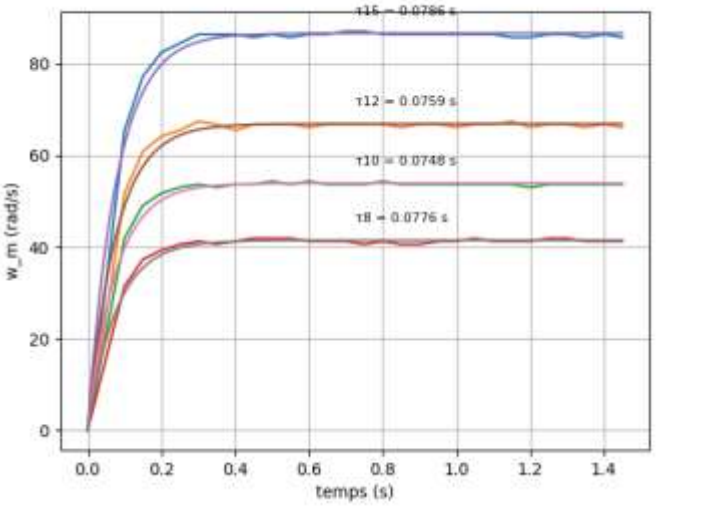


Figure 33: régression de plusieurs échelons

Et une constante de temps moyenne:

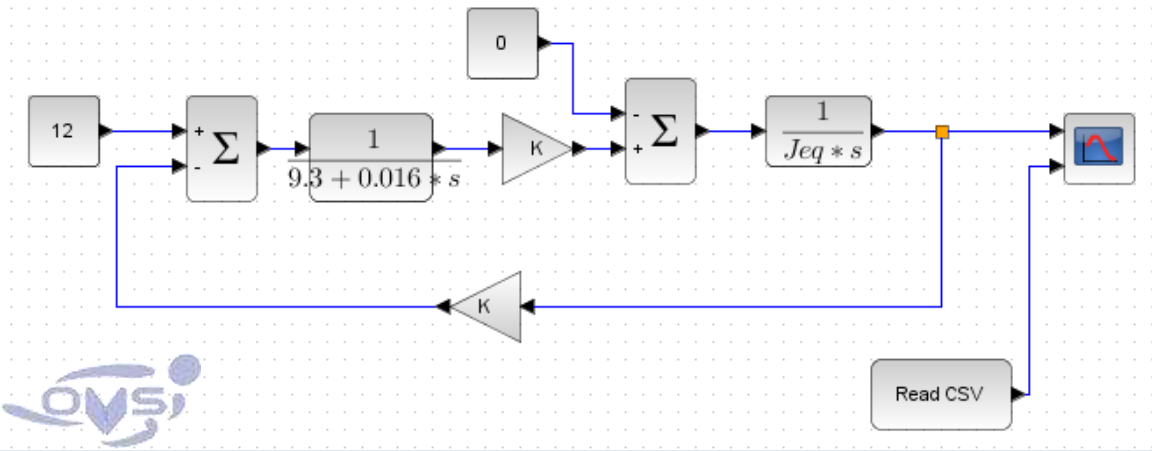
$$\tau_{moy} \approx 0.077s$$

Fonction de transfert d'un moteur CC:

$$M(p) = \frac{1/K}{1 + \frac{RJ_{eq}}{K^2}.p + \frac{LJ_{eq}}{K^2}.p^2} \overset{1^{er}\ ordre}{\approx} \frac{1/K}{1 + \tau.p}$$

d'où

$$J_{eq} = \frac{K^2\tau}{R} = 2.682\ \ 10^{-4}\ \ \mathrm{kg.m^2}$$



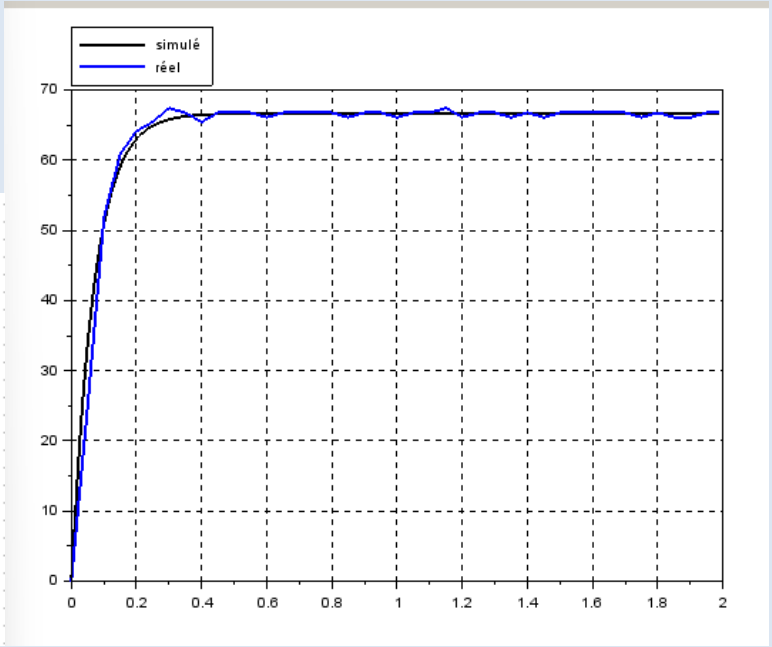


Figure 34: validation du modèle avec **Scilab**

Modèle validé

Critère	valeur
ε *statique*	NULLE
ε *de régulation*	NULLE
stabilité	$M\phi > 50°$
rapidité	$t_{5\%} < 5s$

Figure 35: critères à vérifier pour l'asservissement

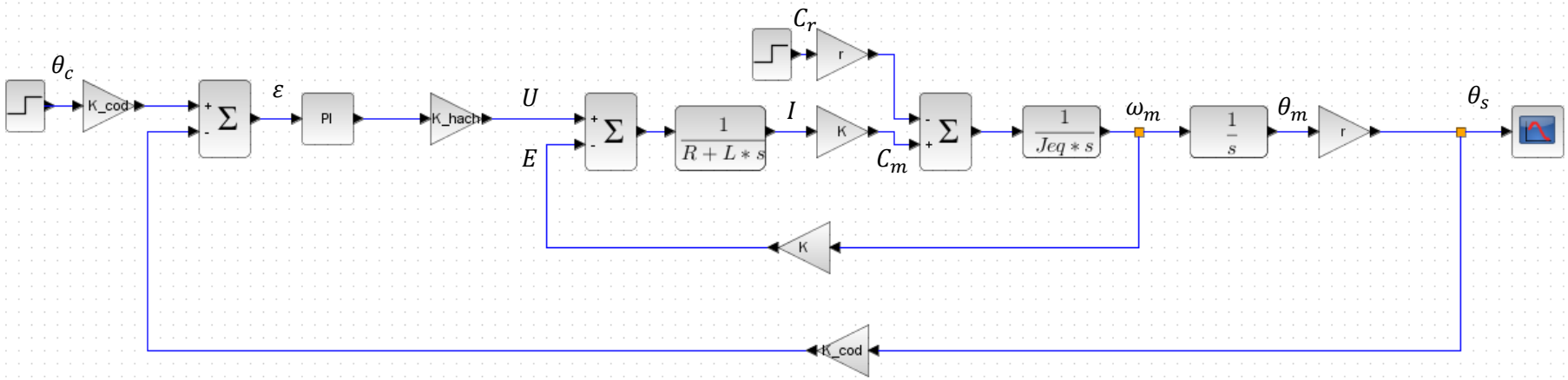


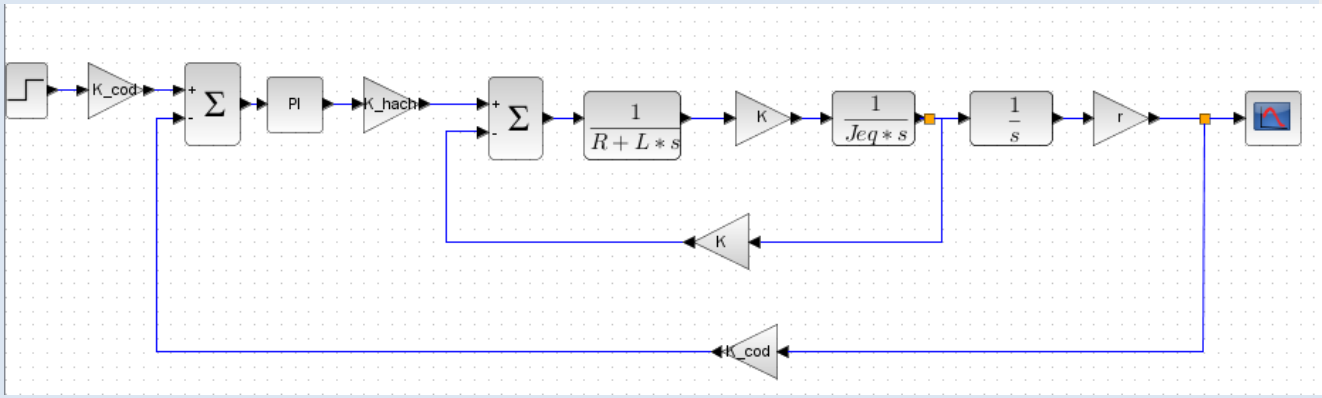
Figure 36: bloc d'asservissement global

Asservir l'angle du support

Validation de ε *statique*:

Le bloc d'asservissement contient une intégration, d'où :

$$\varepsilon \text{ statique} = 0$$



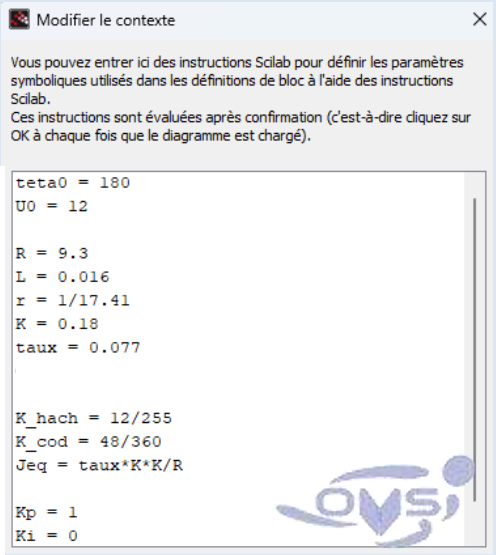


Figure 37: bloc d'asservissement sans perturbation

Sans correction:

Le modèle est précis mais **très long**!!!

$$\varepsilon \text{ de régulation} \neq 0$$

D'où la nécessité d'un correcteur **PI**

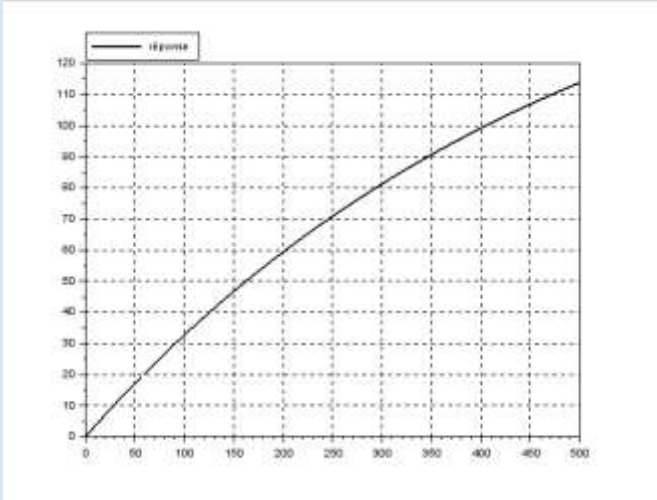


Figure 38: réponse sans correction

Caractéristiques du correcteur PI

Effet *proportionnel* corrige la **rapidité**

Effet *intégral* corrige la **régulation**

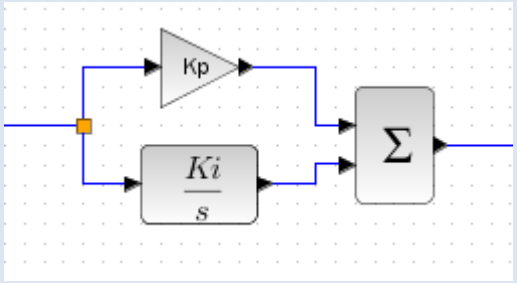


Figure 39: structure interne du PI

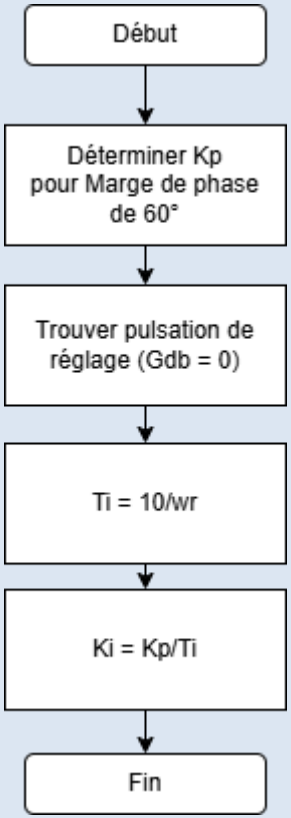


Figure 40: organigramme de détermination de Kp et Ki

Fonction de transfert du PI:

$$PI(p) = K_p + \frac{K_i}{p} = K_p\left(1 + \frac{K_i}{K_p.p}\right) = K_p\left(1 + \frac{1}{T_i.p}\right)$$

$$PI(p) = K_p\left(\frac{1 + T_i.p}{T_i.p}\right)$$

Détermination du PI

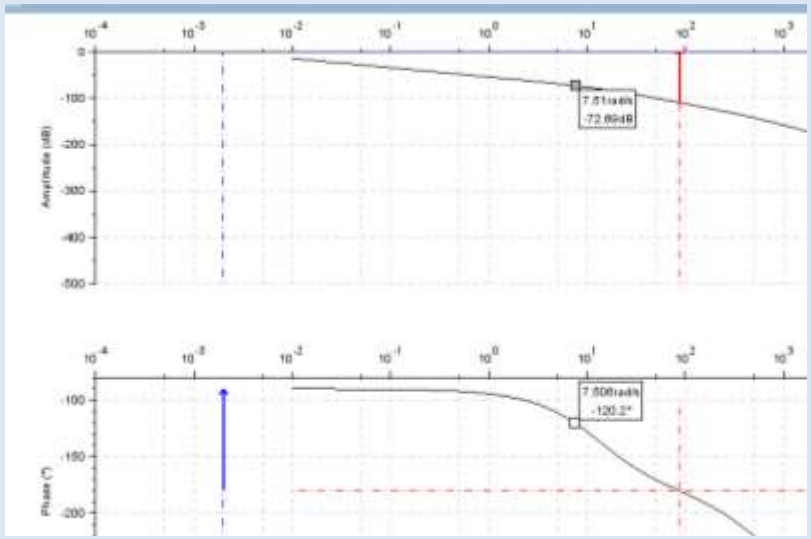


Figure 41: bode sans correction

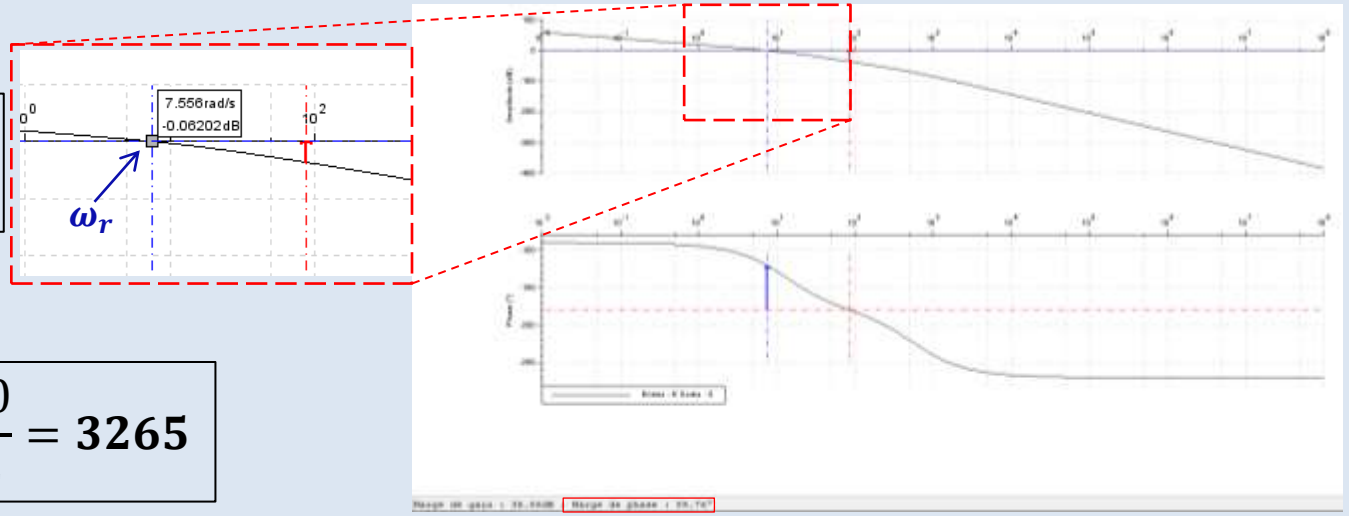
Pour $M\phi = 60° \ (\varphi = -120°)$:

$G_{dB} = -72.69\ dB$ donc:

$$K_{p_{max}} = 10^{\frac{72,69}{20}} = \mathbf{4310}$$

Application de Kp

$$T_i = \frac{10}{\omega_r} = \frac{10}{7.55} = 1.32s$$



$$K_i = \frac{K_p}{T_i} = \frac{4310}{1.32} = \mathbf{3265}$$

Figure 42: bode après application de Kp

Validation de l'asservissement

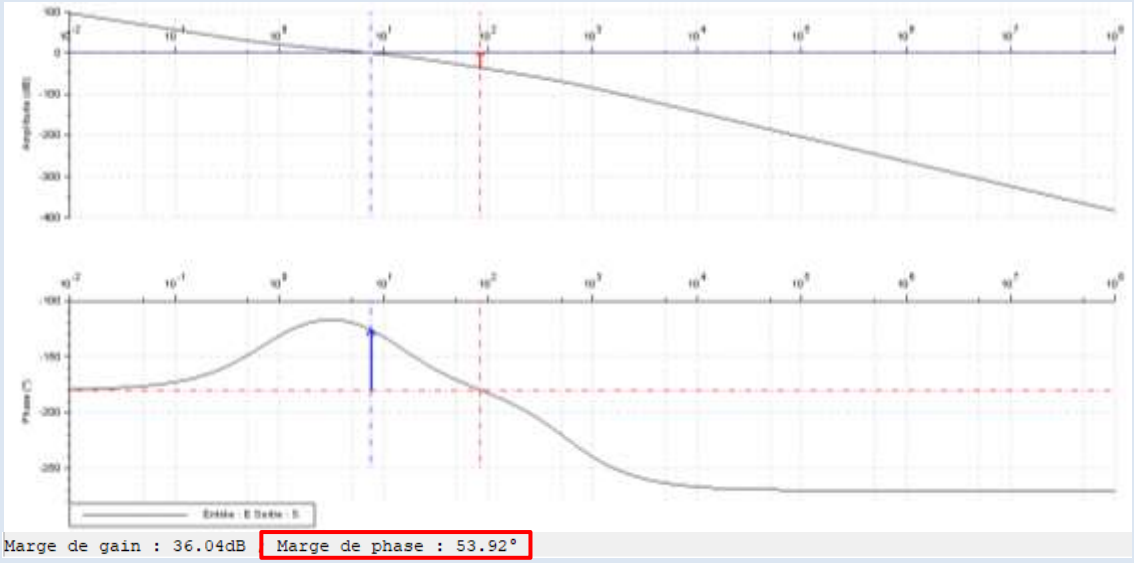


Figure 43: bode après correction

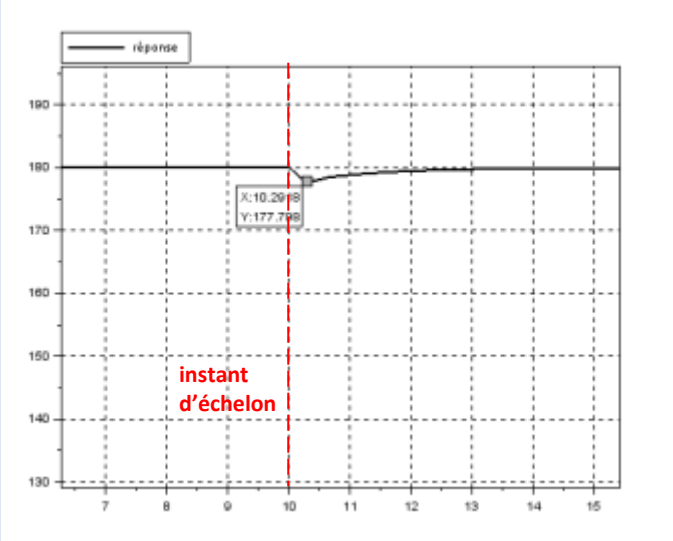


Figure 44: réponse du modèle ***avec*** perturbation

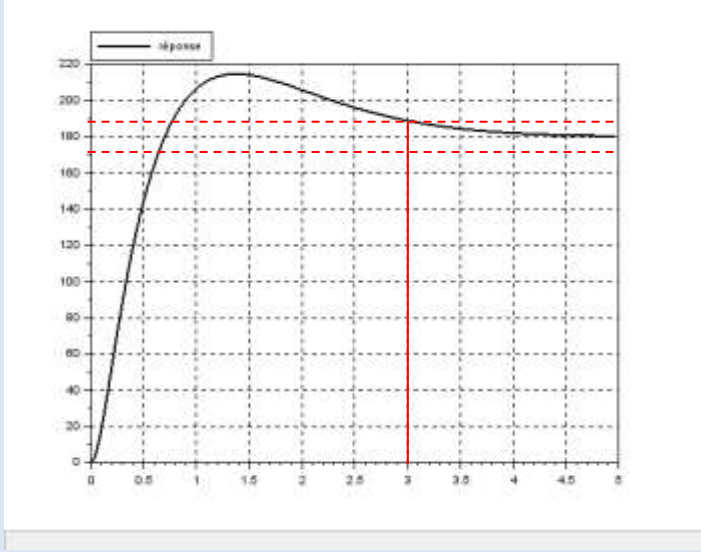


Figure 45: réponse du modèle ***sans*** perturbation

Expérience réelle d'asservissement

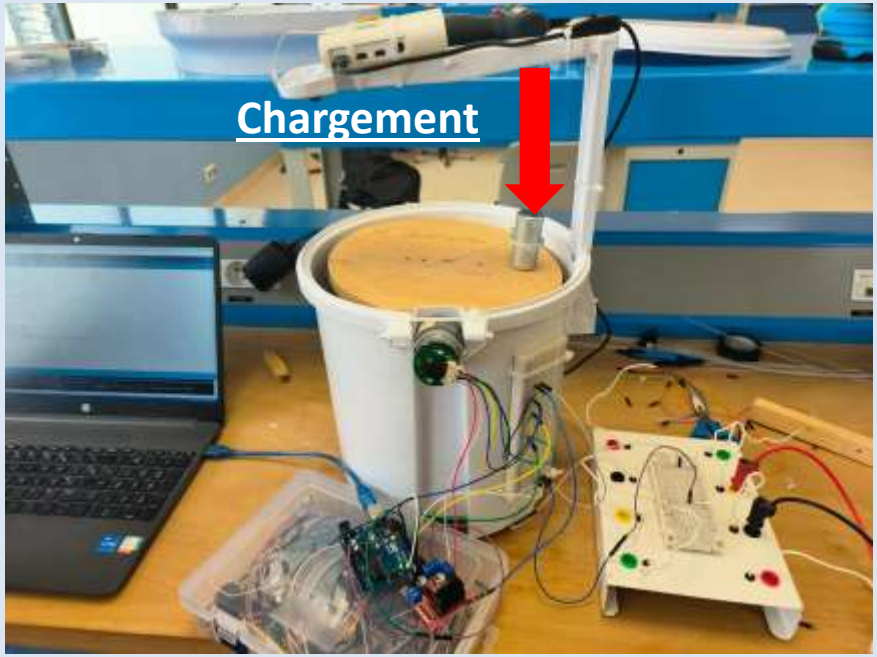


Figure 46: asservissement de position angulaire réellement

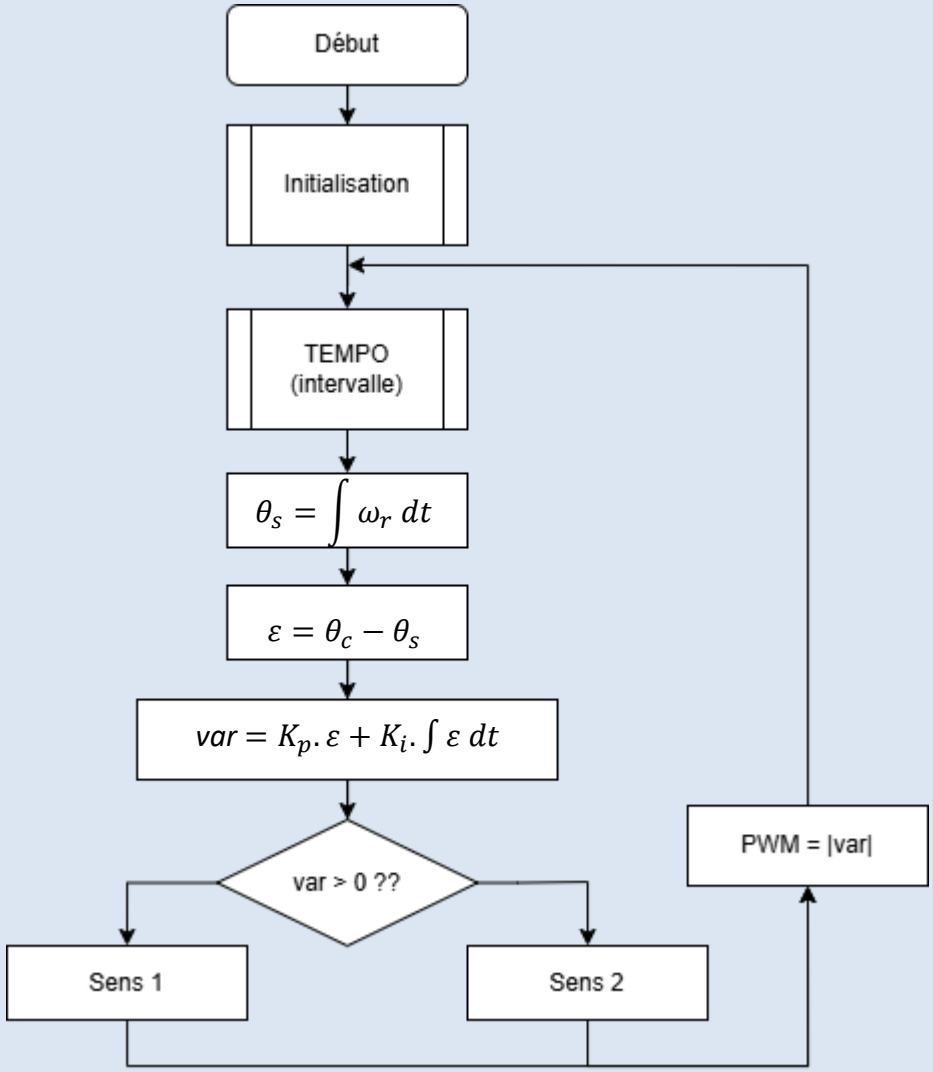


Figure 47: organigramme simplifié d'asservissement de position angulaire **Code A4**

Courbes des acquisitions

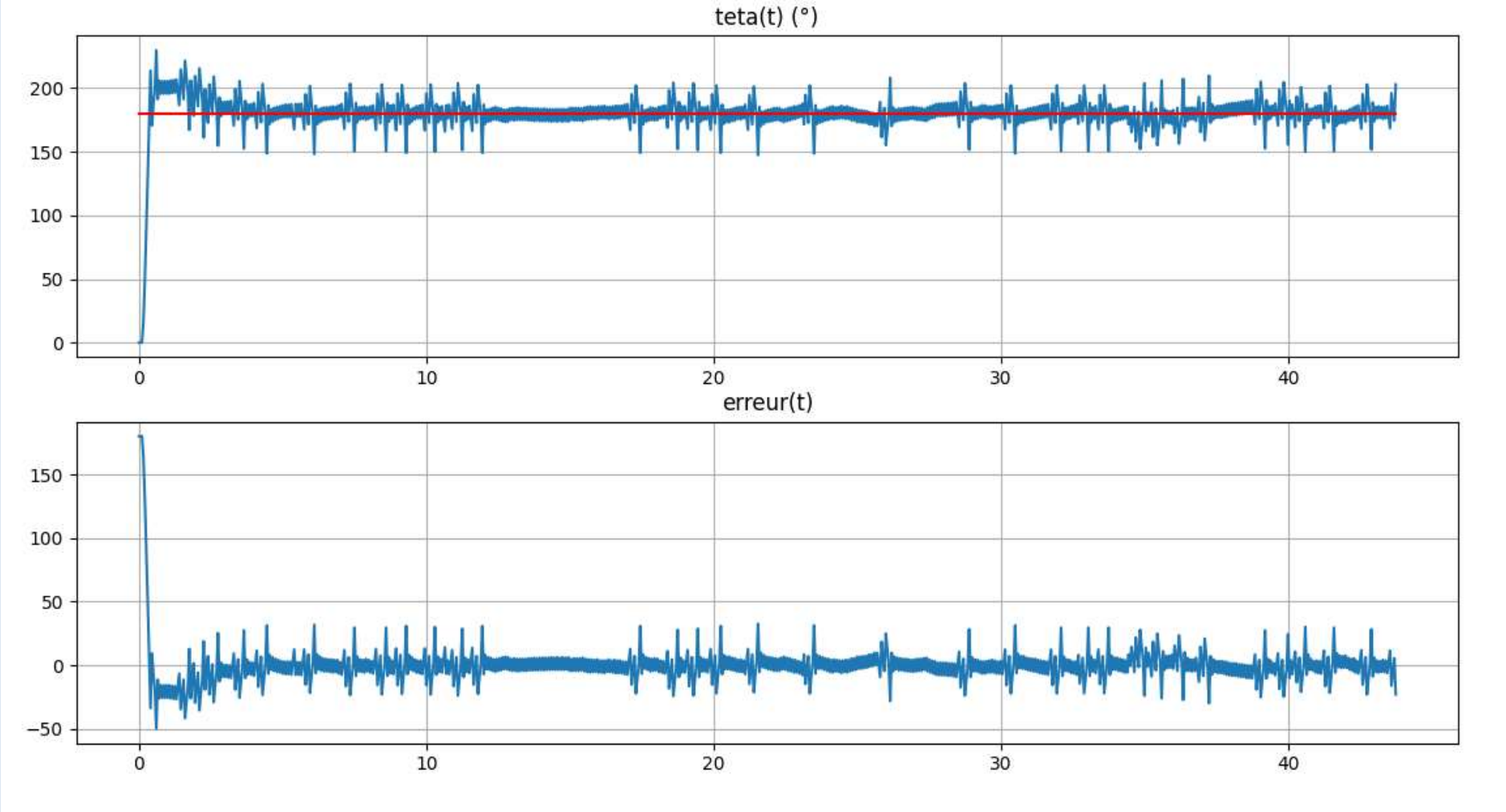


*Figure 48: réponse du système **sans** perturbation*

Courbes des acquisitions

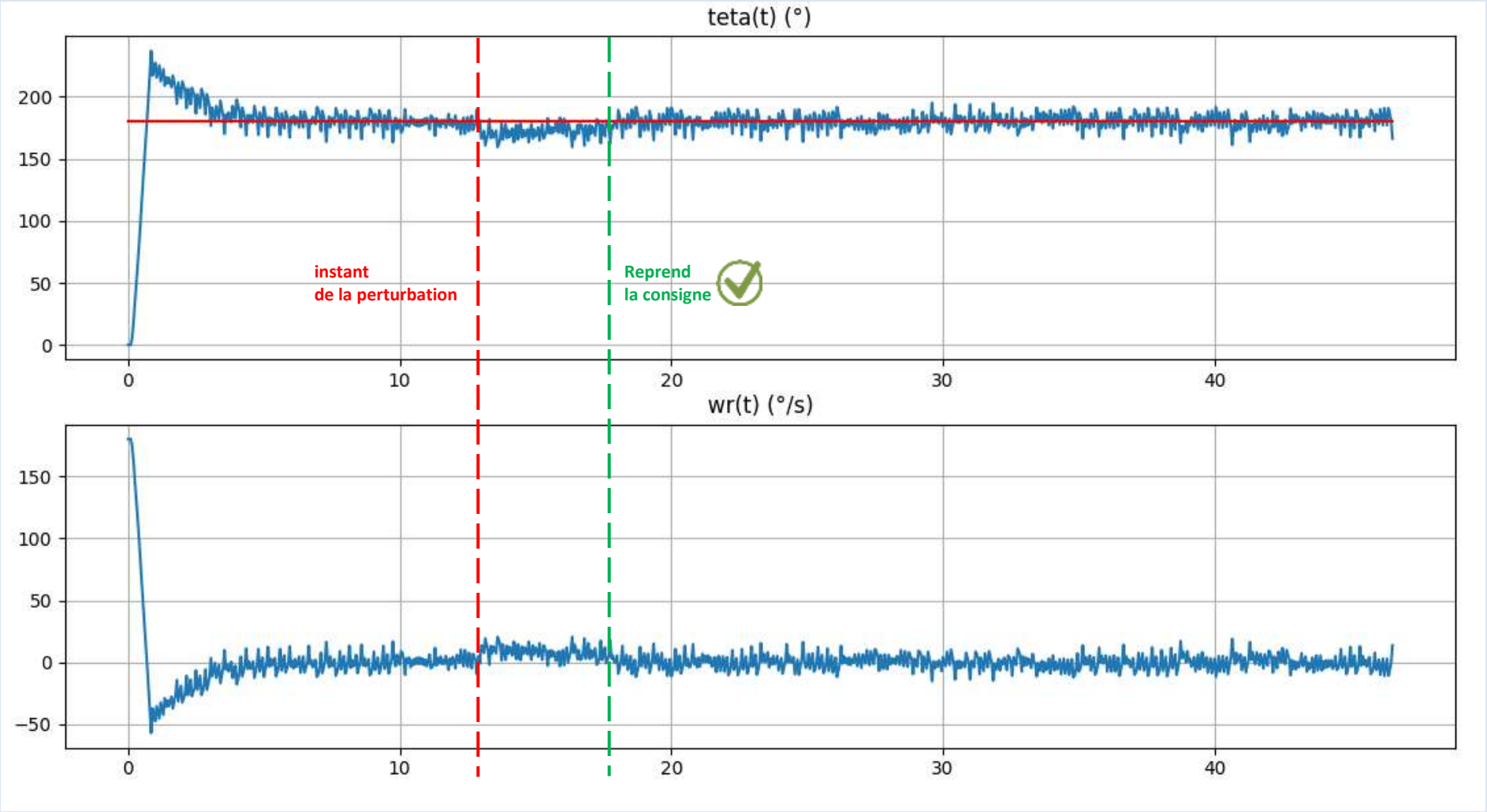
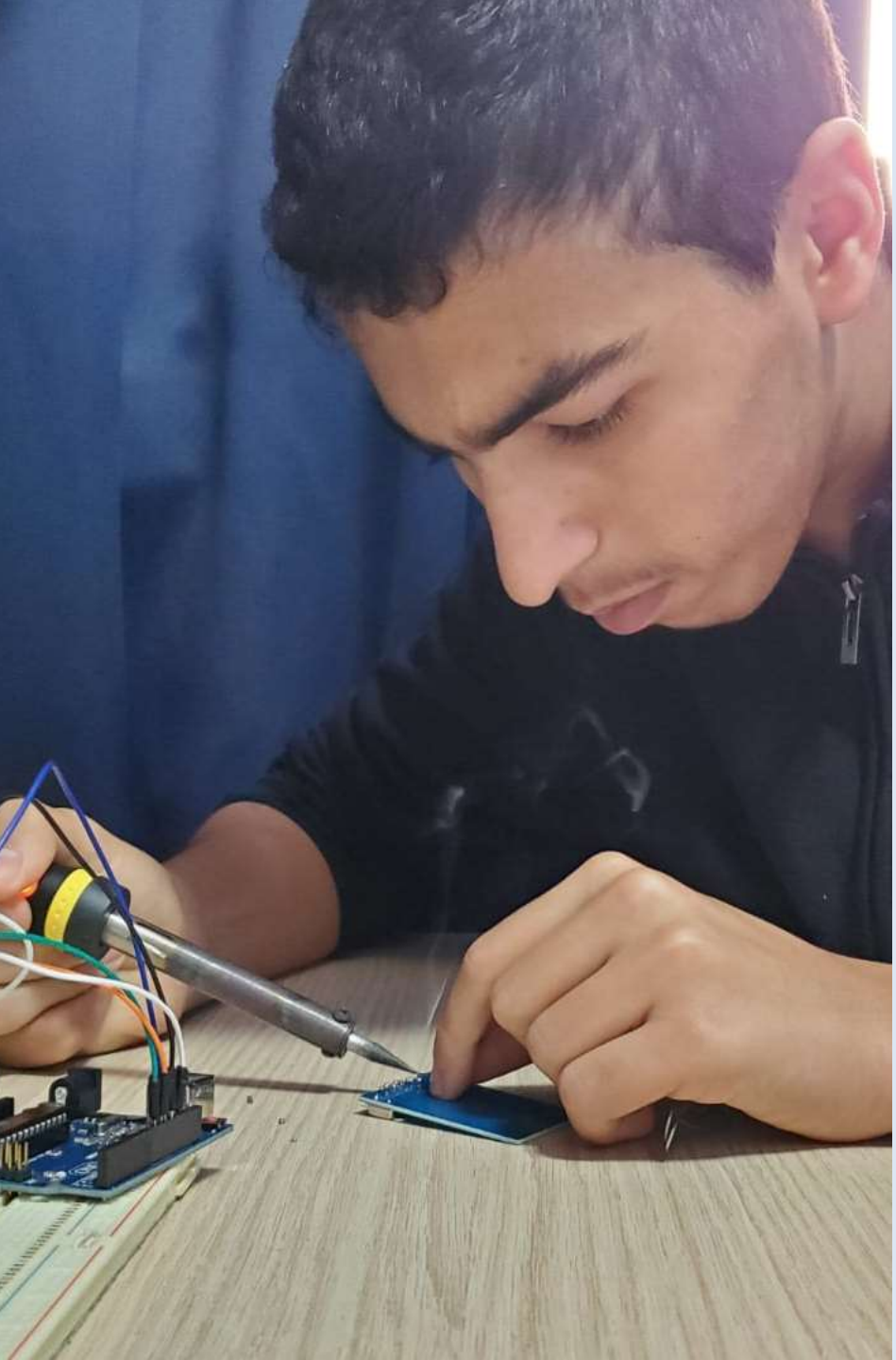


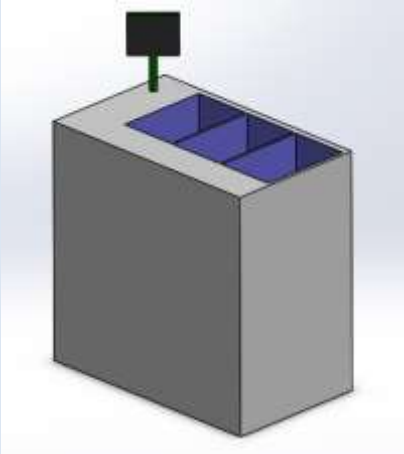
*Figure 49: réponse du système **avec** perturbation*



3

Identification Des adhérents

Identifier chaque adhérent

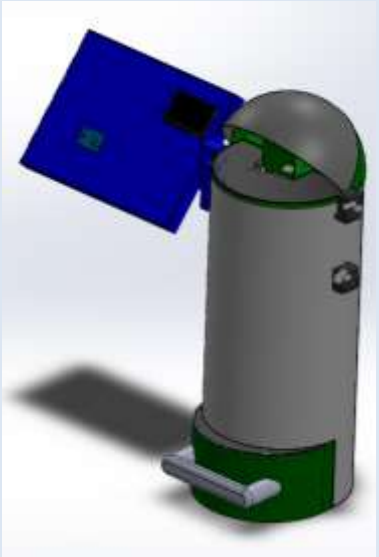


Dans le comptoir

Identification des adhérents…

- Afin d'afficher les menus personnalisés de chaque sportif (**objectif 4**)

- Pour modifier les menus après détection des déchets (**partie d'adaptation non traitée**)



Dans la poubelle

On utilise pour cet objectif:

Un lecteur **RFID**
(**R**adio **F**requency **I**dentification **D**evice)

Figure 50: RFID-RC522 et son tag

Principe de fonctionnement

Tag/badge..:

ou **Mifare** est une technologie de **carte à puce sans contact** la plus répandue dans le monde fabriquée surtout par la société NXP.

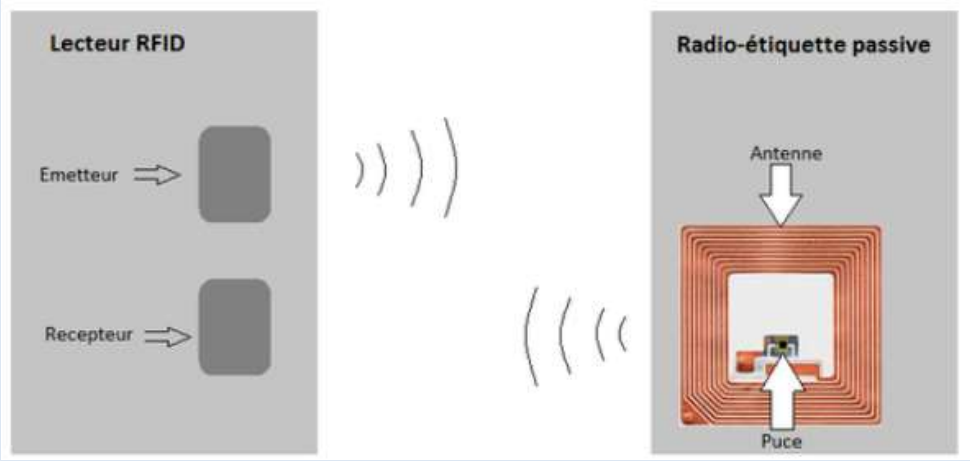


Figure 51: constituants de RFID-RC522

Types de fréquence	Fréquence de fonctionnement
Basse fréquence	< 135 kHz
Haute fréquence	**13.56 Mhz**
Très haute fréquence	863 ~ 915 Mhz

Figure 52: fréquences d'utilisation des RFID

Lecteur RFID:

L'émetteur-récepteur envoie un **champ magnétique** de faible portée.

Expérience

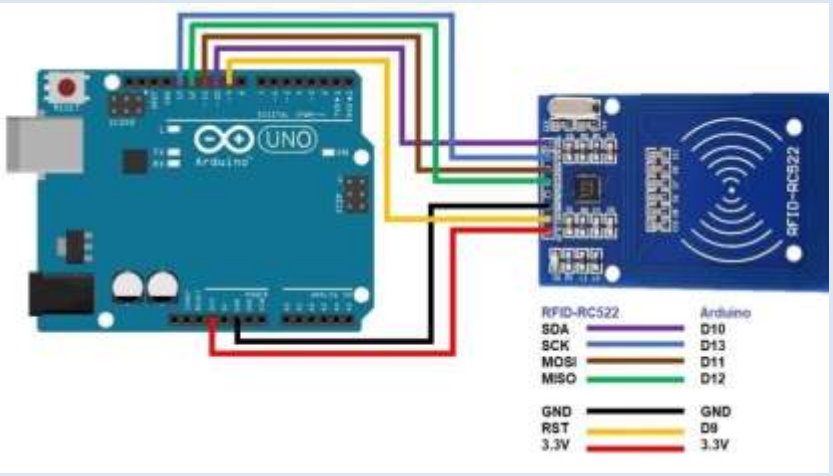


Figure 53: montage expérience arduino

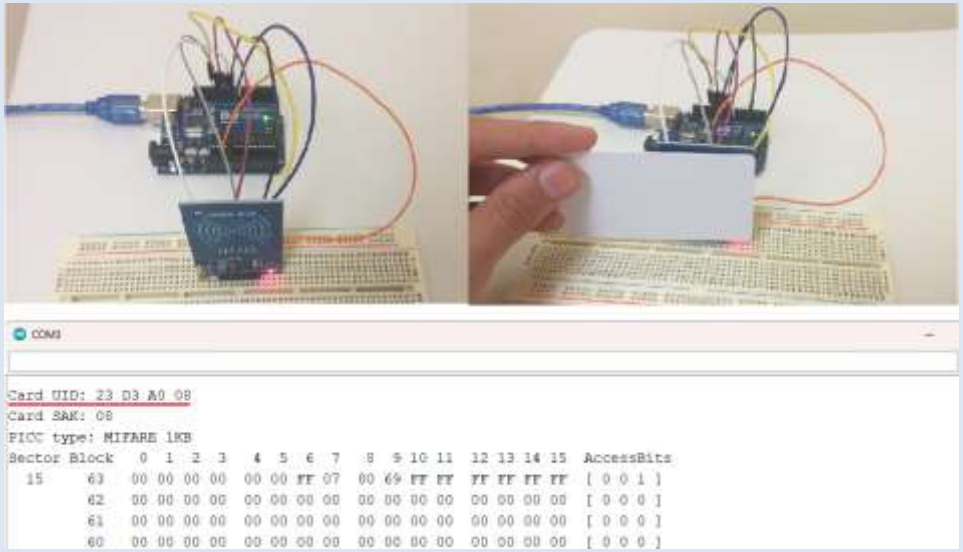


Figure 54: résultats **Code A5**

Après on ne retient que l'**UID**

Envoie l'**UID**

Lecture du port série (**COMx**)

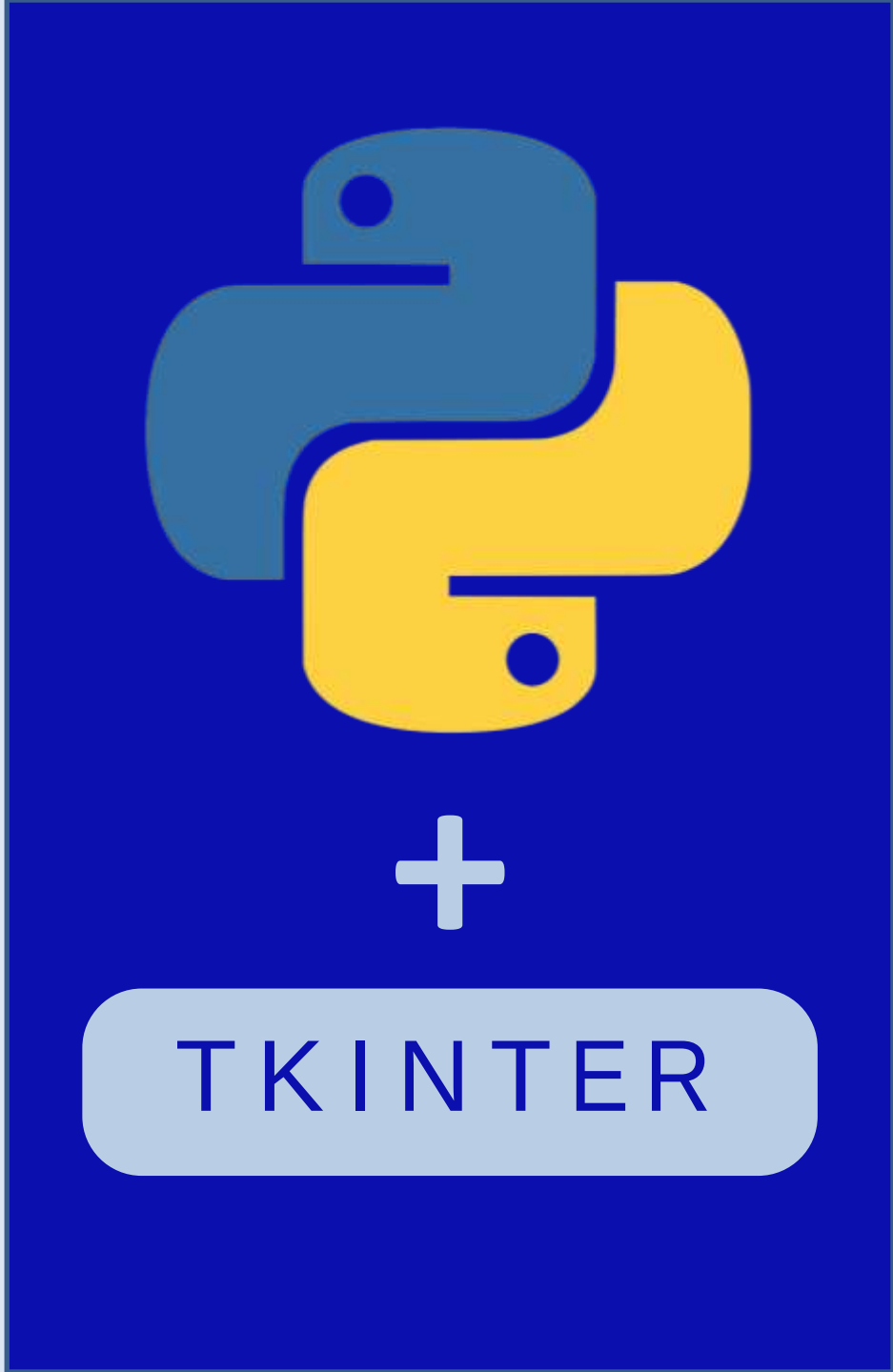
Affichage des menus

Figure 55: principe de communication entre arduino et python

4

GUI avec Tkinter

Réaliser une interface d’affichage des portions



Principe

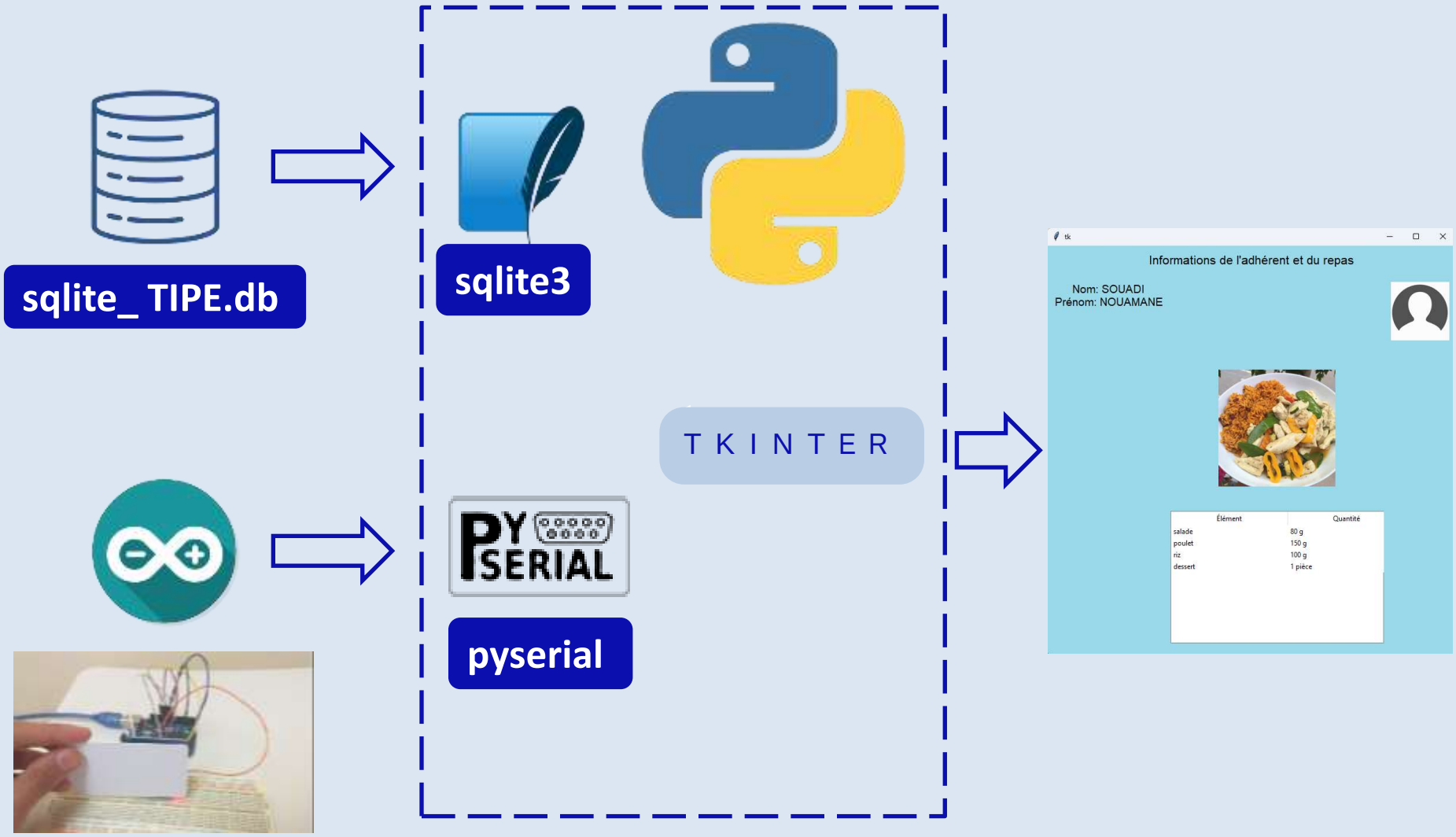


Figure 56: principe global de fonctionnement GUI

Base des données

adherent

id	(entier)
nom	(texte)
prenom	(texte)
img_profil	(texte)

plat

id	(entier)
plat_id	(entier)
plat_img	(texte)
item	(texte)
quantite	(texte)

sqlite_TIPE.db

```
SELECT * FROM adherent
```

id	nom	prenom	img_profil
3A 64 A8 1A	SOUADI	NOUAMANE	ressources/profile.png
23 D3 A0 08	DIHMANI	ILYAS	ressources/profile.png

sqlite_TIPE.db

```
SELECT * FROM plat
```

id	plat_id	plat_img	item	quantite
1	3A 64 A8 1A	ressources/plat...	salade	80 g
2	3A 64 A8 1A	ressources/plat...	poulet	150 g
3	3A 64 A8 1A	ressources/plat...	riz	100 g
4	3A 64 A8 1A	ressources/plat...	dessert	1 pièce
5	23 D3 A0 08	ressources/plat...	salade	100 g
6	23 D3 A0 08	ressources/plat...	poulet	200 g
7	23 D3 A0 08	ressources/plat...	riz	80 g
8	23 D3 A0 08	ressources/plat...	dessert	2 pièces
9	23 D3 A0 08	ressources/plat...	banane	1 pièce

Figure 57: structure de la base des données des adhérents et leurs plats

Composition d'interface

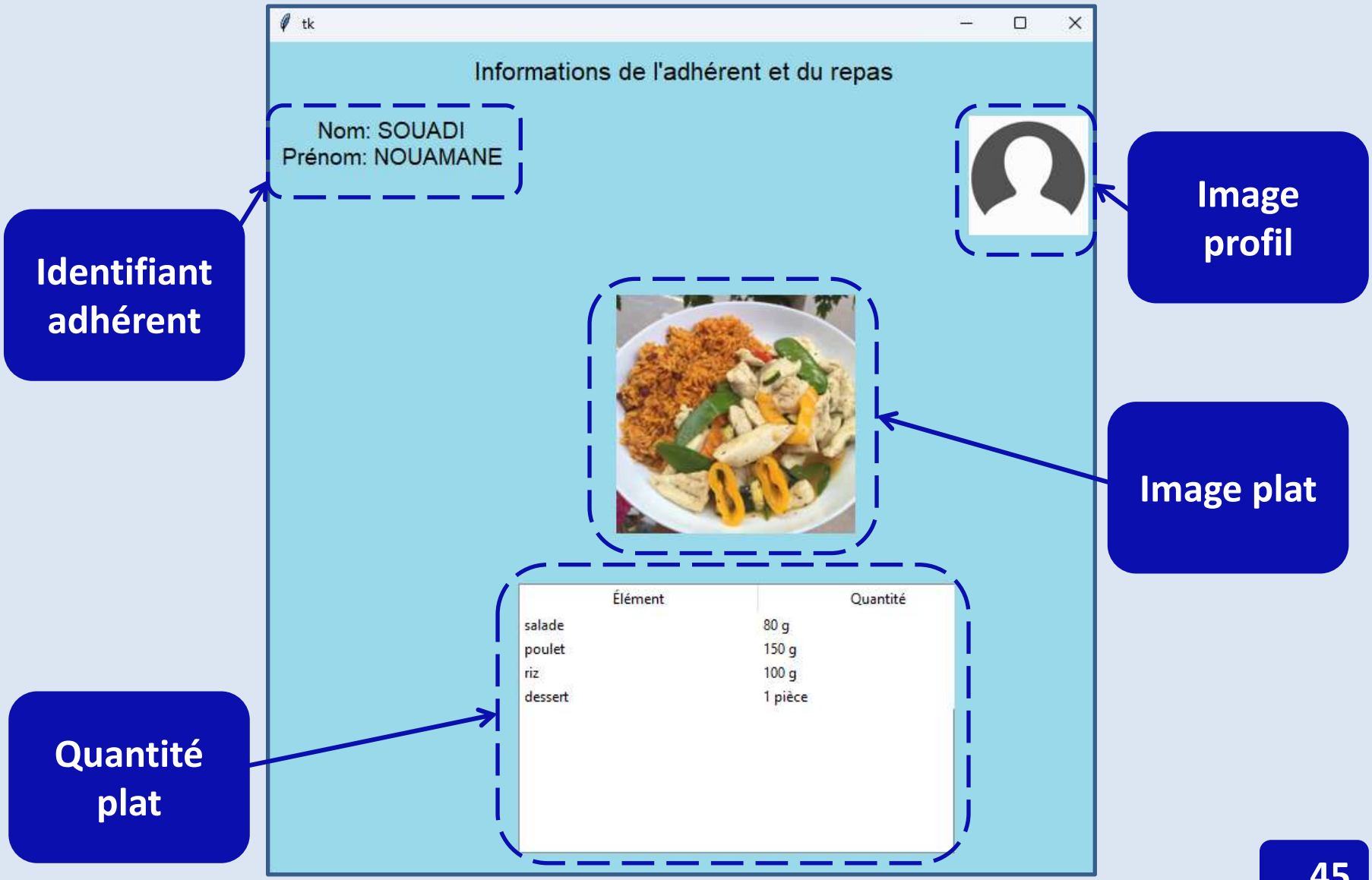
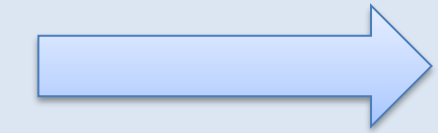


Figure 58: parties du GUI

Résultats

Badges personnels



Menus personnels

Figure 59: résultats personnalisés pour chaque adhérent ***Code A6***

PROTOTYPE

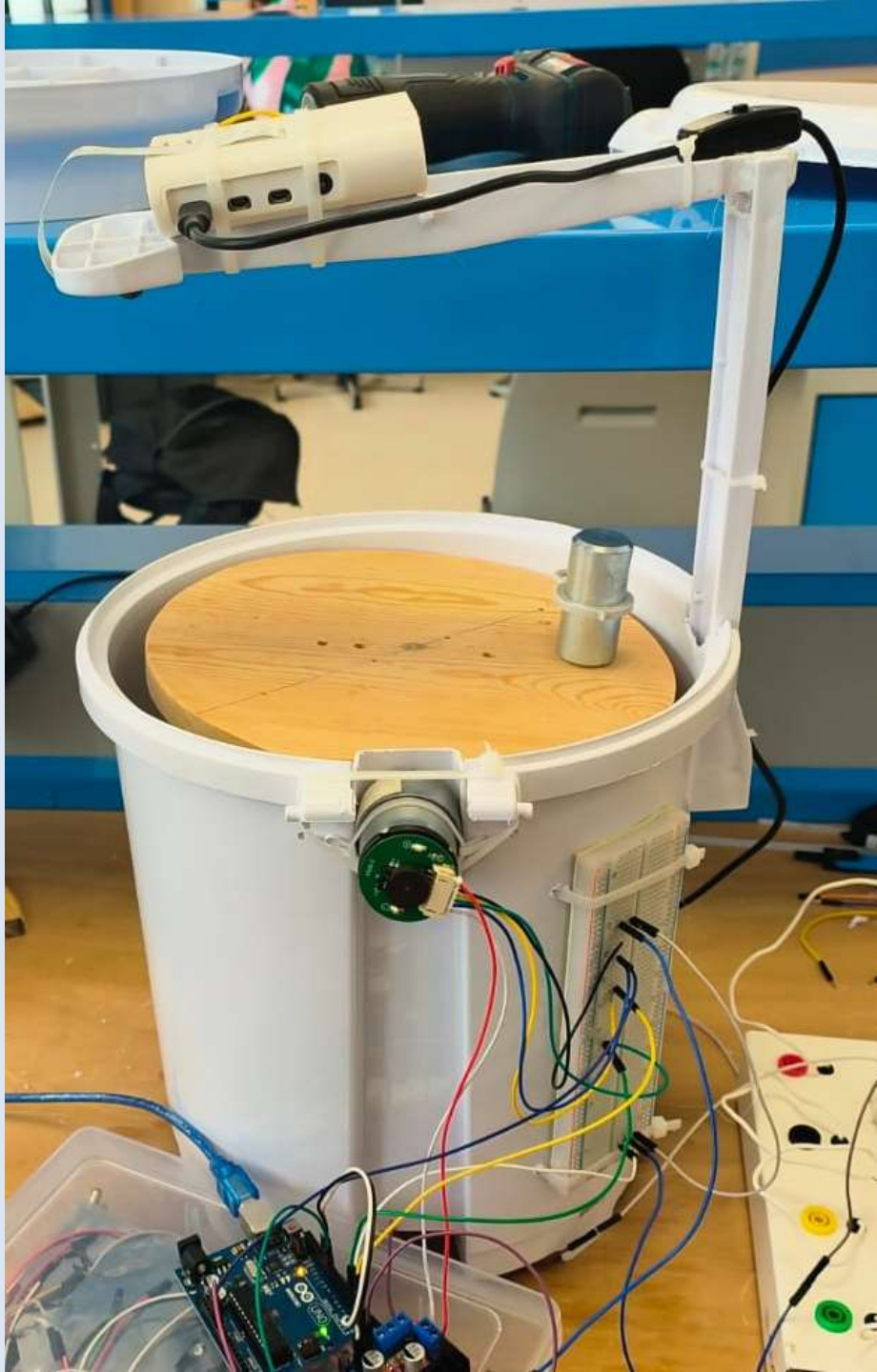
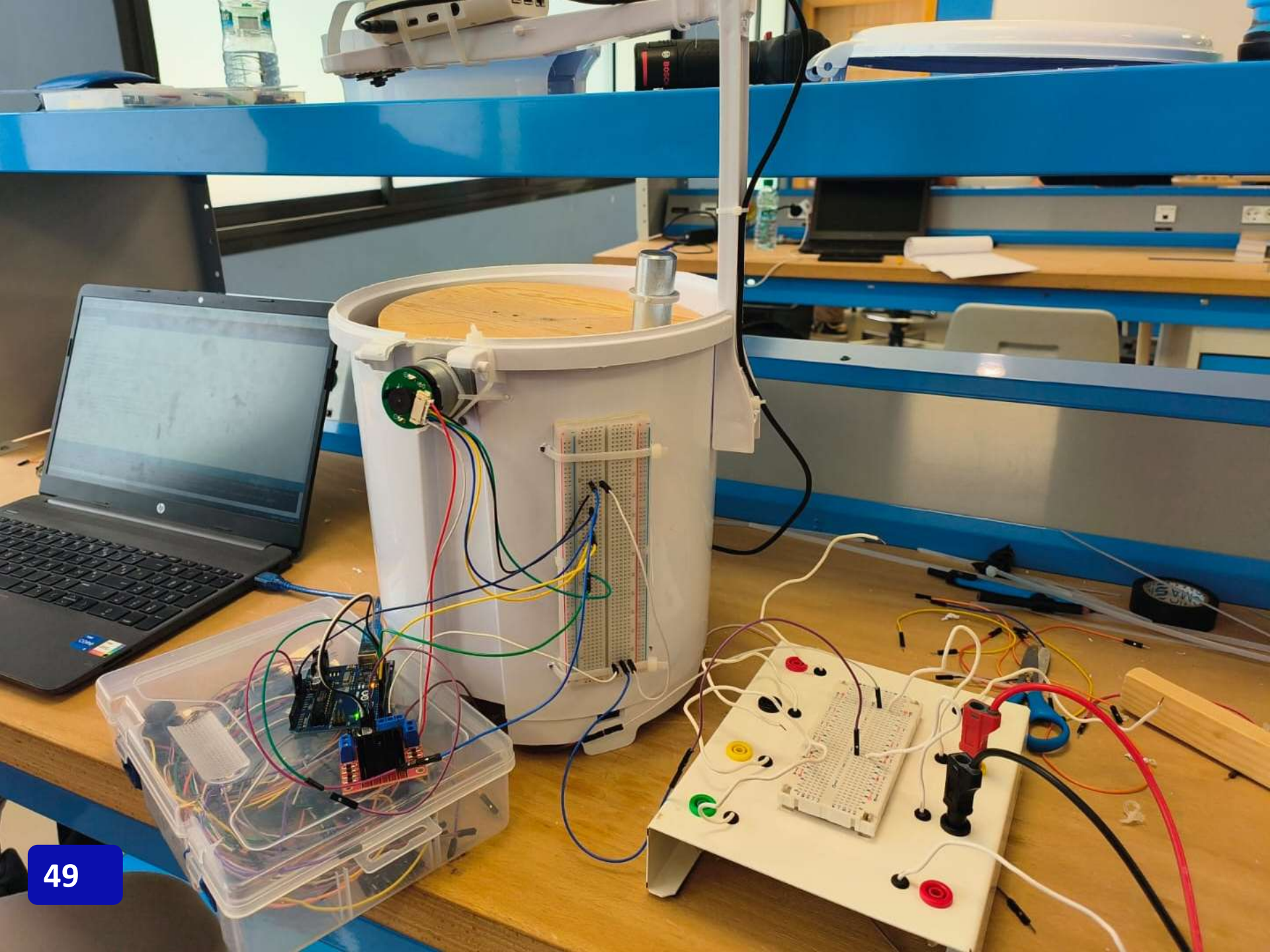


Figure 60: réalisation du prototype



CONCLUSION


Merci pour votre
attention

ANNEXES

A1 : code d'identification (1/2)

```
// 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");
}
```

A1 : code d'identification (2/2)

```
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 > ...

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

data = pd.read_csv("ressources/liste U(w).txt", delimiter=";")
x, y = np.array(data['w']), np.array(data["U"])
[a, b] = np.polyfit(x, y, 1)
def fct(x):
    return a*x + b
R = 9.3 # Par multimètre
Cm = b*a/R
print(f"a = K = {a} V.s/rad, Cm = b.K/R = {Cm} N.m")

x2 = np.linspace(min(x), max(x), 2)

plt.plot(x, y, 'o', label="acquisition U(w)")
plt.plot(x2, fct(x2), label="polyfit")
plt.title(f"$K$=a={round(a, 2)} V.s/rad, $C_m$=b.K/R={round(Cm, 3)} N.m")
plt.legend()
plt.grid()
plt.show()
```

A3 : code de régression de *L*

regression L.py > ...

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

"""
Forme i(t) = [1-exp(-t/taux)]Iper avec Iper = U/Re
Re = Rm + 220
"""
Re = 9.3 + 220
data = pd.read_csv("ressources/acquisitionL.txt", delimiter=",")
t, u = np.array(data["t"]), np.array(data["Ur(t)"])
i = u/Re

indice0 = 0
while t[indice0] < 0:
    indice0 += 1
t = t[indice0:]
i = i[indice0:]

def f(t, Iper, taux):
    return Iper*(1-np.exp(-t/taux)) # taux = L/Re
[Iper, taux], val = curve_fit(f, t, i)
plt.plot(t*1e3, i*1e3, label="acquisition")
plt.plot(t*1e3, f(t, Iper, taux)*1e3, label="curve_fit")
plt.xlabel("t (ms)")
plt.ylabel("i(t) (mA)")
L = Re*taux
plt.title(f"L = Re/taux = {round(L, 3)}H")
print(f"L= {L} H")
plt.show()
```

élimination de la partie négative du temps

A4 : code d'asservissement (arduino) (1/3)

```
// 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);
```

A4 : code d'asservissement (arduino) (2/3)

```
  // 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);
```

A4 : code d'asservissement (arduino) (3/3)

```
      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) (1/4)

GUI TIPE.py > afficher_adherent_et_repas

```python
import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
import sqlite3
import serial

# Create the main window
fenetre = tk.Tk()

bg = "#99D9EA"
fenetre.configure(bg=bg)

# Calculate the coordinates to center the window on the screen
largeur_ecran = fenetre.winfo_screenwidth()
hauteur_ecran = fenetre.winfo_screenheight()
x = (largeur_ecran - 700) // 2
y = (hauteur_ecran - 700) // 2

# Position the window at the center of the screen
fenetre.geometry(f"700x700+{x}+{y}")

def afficher_adherent_et_repas(id, fenetre):
    try:
        # Connect to the database
        conn = sqlite3.connect('sqlite_TIPE.db')
        conn.row_factory = sqlite3.Row
        cur = conn.cursor()
    except sqlite3.Error as e:
        print(f"Erreur de connexion à la base de données: {e}")
        return

    cur.execute("SELECT * FROM adherent WHERE id=?", (id,))
    adherent = cur.fetchone()
```

A6 : code RFID (python) (2/4)

```
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_principal = tk.Frame(fenetre, bg=bg)
48      frame_principal.pack(expand=True, fill="both")
49
50      # Add a title label
51      label_titre = tk.Label(frame_principal, 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_principal, 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_principal, 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
```

A6 : code RFID (python) (3/4)

```python
    # Add the repas information
    # Table des items avec leurs quantités (centrée)
    cur.execute("SELECT item, quantite FROM plat WHERE plat_id=?", (id,))
    items_quantites = cur.fetchall()

    # Créer une Treeview (table) avec des colonnes "Élément" et "Quantité"
    table_items = ttk.Treeview(frame_principal, columns=("Element", "Quantite"), show="headings")
    table_items.heading("Element", text="Élément")
    table_items.heading("Quantite", text="Quantité")

    for item_quantite in items_quantites:
        table_items.insert("", "end", values=(item_quantite['item'], item_quantite['quantite']))

    table_items.pack(side="bottom", anchor="center", pady=20)

    # Image du plat centrée
    cur.execute("SELECT plat_id, plat_img FROM plat WHERE plat_id=?", (id,))
    plat = cur.fetchone()

    if plat:
        img_plat = Image.open(plat['plat_img'])
        img_plat = img_plat.resize((200, 200))
        img_plat_tk = ImageTk.PhotoImage(img_plat)
        label_img_plat = tk.Label(frame_principal, image=img_plat_tk, bg=bg)
        label_img_plat.pack(side="bottom", pady=20)

    conn.close()
    fenetre.update()

def read_rfid_data():
    while True:
        try:
            rfid_data = arduino_serial_port.readline().decode('utf-8').strip()
            afficher_adherent_et_repas(rfid_data, fenetre)
```

A6 : code RFID (python) (4/4)

```
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()
```