

Smart Car : Détection de Somnolence et d'Alcool

Projet Fin de Semestre – CAO / PCB

Aya Yahyaoui

Yosr Derbali

Encadrante : Hajer Gassouma

Université de Monastir

Institut Supérieur d'Informatique et de Mathématiques de Monastir (ISIMM)

Département Technologie

Année Universitaire : 2025–2026

Table des matières

Introduction.....	1
1 Matériel et Outils Utilisés.....	2
1.1 Composants matériels.....	2
1.2 Logiciels et outils	2
1.3 Place pour l'image du matériel	3
2 Conception du Système	3
2.1 Architecture générale du projet.....	3
2.2 Schéma fonctionnel	4
3 Détection de la Somnolence	4
3.1 Principe.....	4
3.2 Place pour captures Python.....	5
3.3 Code Python	5
4 Détection d'Alcool avec le Capteur MQ3	10
4.1 Principe.....	10
4.2 Tableau des mesures	10
4.3 Place pour image MQ3.....	10
4.4 Code Arduino	11

5 Simulation sur Proteus	13
5.1 Description.....	13
5.2 Place pour la simulation.....	13
6 Réalisation CAO–PCB	14
6.1 Place pour le schéma PCB.....	14
Conclusion	14
A Annexes	20
A.1 Place pour vidéos.....	20

Introduction

La sécurité routière est aujourd’hui au cœur des priorités technologiques. Ce projet intitulé Smart Car : Détection de Somnolence et d’Alcool vise à concevoir un système embarqué capable d’alerter le conducteur en cas de somnolence ou de présence d’alcool.

Ce travail combine :

- l’utilisation d’un Arduino Uno pour la détection d’alcool via le capteur MQ3,
- la vision par ordinateur avec une webcam PC pour détecter la fermeture prolongée des yeux,
- la communication Python–Arduino via pyserial,
- une simulation électronique sur Proteus 8,
- la réalisation d’un schéma CAO–PCB du circuit.

Les deux parties du projet fonctionnent simultanément : si les yeux du conducteur restent fermés plus de trois secondes, ou si un seuil d’alcool est détecté, une alarme sonore et visuelle est déclenchée.

Partie 1:

Matériel et Outils Utilisés

1.1 Composants matériels

- Arduino UNO
- Capteur de gaz MQ3 (détection d'alcool)
- Webcam PC (détection de somnolence)
- LED externe + résistance
- Buzzer
- Breadboard et câbles Dupont

1.2 Logiciels et outils

- Proteus 8 (simulation électronique)
- VS Code (Python + OpenCV + pyserial)
- Arduino IDE
- Outils CAO-PCB

1.3 Place pour l'image du matériel

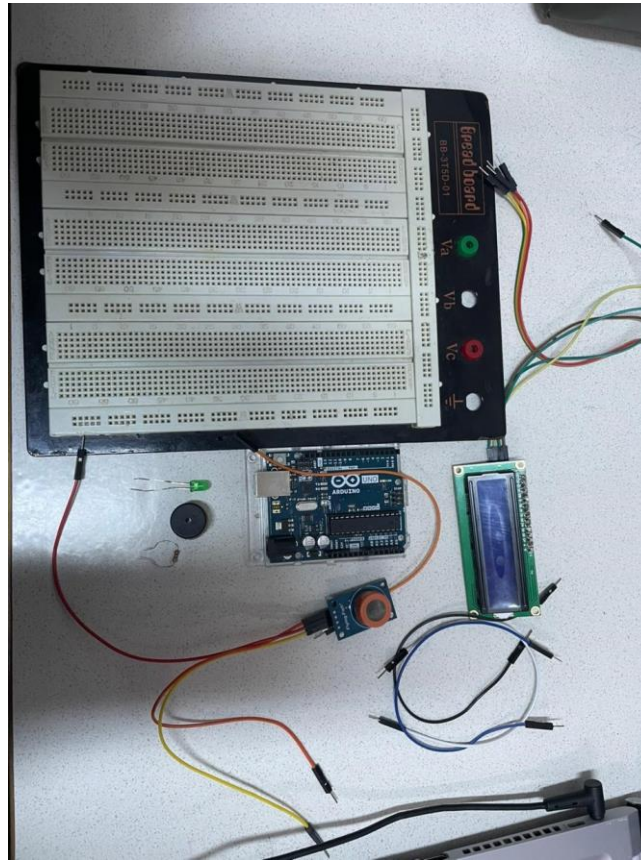


Figure 1.1 – matériels utilisés

partie 2:

Conception du Système

2.1 Architecture générale du projet

Le système est composé de deux sous-parties :

- Détection de somnolence : réalisée en Python grâce à une webcam et OpenCV. Si les yeux restent fermés plus que 3 secondes, une alerte est envoyée à l'Arduino via le port COM3.
- Détection d'alcool : effectuée par le capteur MQ3. Un seuil de 20 a été fixé, car le parfum utilisé ne contient qu'un faible pourcentage d'alcool.

2.2 Schéma fonctionnel

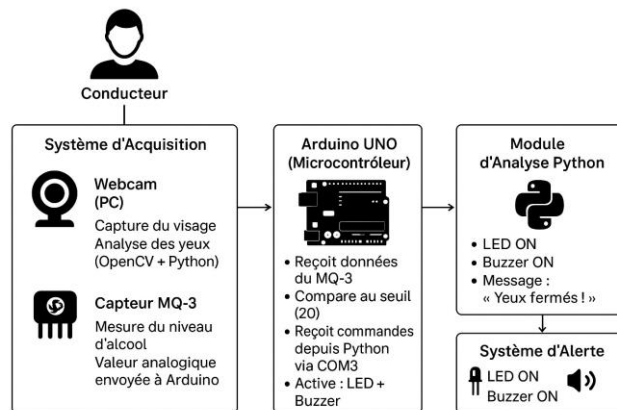


Figure 2.2 – Schéma fonctionnel du projet –

Partie 3:

Détection de la Somnolence

3.1 Principe

La détection se base sur :

- reconnaissance du visage,
- détection des yeux,
- mesure du temps de fermeture.

3.2 captures Python:

```
import cv2
import time
import serial
import numpy as np

class SimpleEyeDetector:
    def __init__(self, com_port='COM3', baud_rate=9600):
        # Load face and eye cascade classifiers
        self.face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
        self.eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

        # Timing variables
        self.eyes_closed_start_time = None
        self.ALERT_THRESHOLD = 1.0 # 1 seconds
        self.alert_triggered = False

        # Serial connection to Arduino
        try:
            self.arduino = serial.Serial(com_port, baud_rate, timeout=1)
            time.sleep(2)
            print(f"Connected to Arduino on {com_port}")
            self.send_to_arduino('0') # Initialize OFF
        except:
            print("Running in simulation mode (no Arduino)")
            self.arduino = None

    def send_to_arduino(self, state):
        """Send state to Arduino"""
        if self.arduino and self.arduino.is_open:
            try:
                self.arduino.write(state.encode())
                print(f"Arduino: {'ALERT ON' if state == '1' else 'normal OFF'}")
            except Exception as e:
                print(f"Serial error: {e}")

    def detect_eyes_closed(self, frame, eyes):
        """Detect if eyes are closed based on pixel intensity"""
```

```
while True:
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame")
        break

    # Flip for mirror effect
    frame = cv2.flip(frame, 1)

    # Convert to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces
    faces = self.face_cascade.detectMultiScale(gray, 1.3, 5)

    status_text = "No face"
    status_color = (0, 0, 255)
    eyes_detected = 0

    for (x, y, w, h) in faces:
        # Draw face rectangle
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

        # Region of Interest for eyes (upper half of face)
        roi_gray = gray[y:y+h//2, x:x+w]
        roi_color = frame[y:y+h//2, x:x+w]

        # Detect eyes within face region
        eyes = self.eye_cascade.detectMultiScale(roi_gray, 1.3, 5)
        eyes_detected = len(eyes)

        # Draw eye rectangles
        for (ex, ey, ew, eh) in eyes:
            cv2.rectangle(roi_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)
```

3.3 Code Python

(import cv2 import
time import serial
import numpy as np

```
class SimpleEyeDetector:
    def __init__(self, com_port='COM3', baud_rate=9600):
        # Load face and eye cascade classifiers
```

```
while True:
    ret, frame = cap.read()
    if not ret:
        print("Can't receive frame")
        break

    # Flip for mirror effect
    frame = cv2.flip(frame, 1)

    # Convert to grayscale for face detection
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detect faces
    faces = self.face_cascade.detectMultiScale(gray, 1.3, 5)

    status_text = "No face"
    status_color = (0, 0, 255)
    eyes_detected = 0

    for (x, y, w, h) in faces:
        # Draw face rectangle
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

        # Region of Interest for eyes (upper half of face)
        roi_gray = gray[y:y+h//2, x:x+w]
        roi_color = frame[y:y+h//2, x:x+w]

        # Detect eyes within face region
        eyes = self.eye_cascade.detectMultiScale(roi_gray, 1.3, 5)
        eyes_detected = len(eyes)
```

```

self.face_cascade = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_f
self.eye_cascade =
cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_ey

# Timing variables
self.eyes_closed_start_time = None self.ALERT_THRESHOLD = 3.0 #
3 seconds self.alert_triggered = False

# Serial connection to Arduino try:
    self.arduino = serial.Serial(com_port, baud_rate, timeout=1) time.sleep(2)
    print(f"Connected to Arduino on {com_port}") self.send_to_arduino('0') # Initialize OFF
except:
    print("Running in simulation mode (no Arduino)") self.arduino = None

def send_to_arduino(self, state): """Send state to
Arduino""" if self.arduino and self.arduino.is_open:
    try:
        self.arduino.write(state.encode())
        print(f"Arduino: {'ALERT ON' if state == '1' else 'Normal OFF'}")
    except Exception as e:
        print(f"Serial error: {e}")

def detect_eyes_closed(self, frame, eyes):
    """Detect if eyes are closed based on pixel intensity""" if len(eyes) == 0:
        return True # No eyes detected = considered closed

    for (ex, ey, ew, eh) in eyes: # Extract
        eye region
        eye_region = frame[ey:ey+eh, ex:ex+ew]

        # Convert to grayscale
        gray_eye = cv2.cvtColor(eye_region, cv2.COLOR_BGR2GRAY)

        # Apply threshold to detect dark pixels (closed eyes)
        _, threshold = cv2.threshold(gray_eye, 30, 255, cv2.THRESH_BINARY_INV)

        # Calculate percentage of black pixels black_pixels =
        np.sum(threshold == 255) total_pixels = ew * eh
        black_percentage = black_pixels / total_pixels

```

```

        # If eye region is mostly dark (black pixels > 70%), eye is closed if black_percentage
        < 0.7: # 70% threshold return False

    return True

def run(self):
    cap = cv2.VideoCapture(0)

    if not cap.isOpened():
        print("Cannot open camera") return

    print("Starting simple eye detection...") print("Press 'q' to quit")

    while True: ret, frame = cap.read() if not ret:
        print("Can't receive frame") break

        # Flip for mirror effect
        frame = cv2.flip(frame, 1)

        # Convert to grayscale for face detection gray =
        cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Detect faces
        faces = self.face_cascade.detectMultiScale(gray, 1.3, 5)

        status_text = "No face" status_color = (0, 0,
        255) eyes_detected = 0

        for (x, y, w, h) in faces: # Draw
            face rectangle
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

            # Region of Interest for eyes (upper half of face) roi_gray =
            gray[y:y+h//2, x:x+w] roi_color = frame[y:y+h//2, x:x+w]

            # Detect eyes within face region
            eyes = self.eye_cascade.detectMultiScale(roig_gray, 1.3, 5) eyes_detected = len(eyes)

            # Draw eye rectangles for (ex, ey,
            ew, eh) in eyes:
                cv2.rectangle(roig_color, (ex, ey), (ex+ew, ey+eh), (0, 255, 0), 2)

```

```

# Check if eyes are closed
eyes_closed = self.detect_eyes_closed(roi_color, eyes)

if eyes_closed:
    status_text = "Eyes CLOSED" status_color = (0, 0,
    255)

    # Start timer if not already started if
    self.eyes_closed_start_time is None:
        self.eyes_closed_start_time = time.time()

        closed_duration = time.time() - self.eyes_closed_start_time

    # Check if closed for more than 3 seconds
    if closed_duration > self.ALERT_THRESHOLD and not self.alert_trigger
        self.alert_triggered = True
        print(f"ALERT! Eyes closed for {closed_duration:.1f}s")
        self.send_to_arduino('1')

    # Display timer
    cv2.putText(frame, f"Closed: {closed_duration:.1f}s", (x, y-40),
        cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
else:
    status_text = "Eyes OPEN" status_color = (0,
    255, 0)

    # Reset timer when eyes open self.eyes_closed_start_time = None

    if self.alert_triggered:
        self.alert_triggered = False print("Eyes opened -
        Alert cleared") self.send_to_arduino('0')

break # Only process first face

# Display status on frame
cv2.putText(frame, f"Status: {status_text}", (10, 30),
    cv2.FONT_HERSHEY_SIMPLEX, 0.8, status_color, 2)
cv2.putText(frame, f"Eyes detected: {eyes_detected}", (10, 60),
    cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 2)

if self.alert_triggered:

```

```

        cv2.putText(frame, "ALERT! LED & BUZZER ON", (10, 90),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

    # Show frame
    cv2.imshow('Simple Eye Detection', frame)

    # Exit on 'q' key if cv2.waitKey(1) & 0xFF ==
    ord('q'):
        break
    # Cleanup cap.release()
    cv2.destroyAllWindows() if
    self.arduino:
        self.arduino.close()
    print("Program ended")

if __name__ == "__main__": #
    Run the detector
    detector = SimpleEyeDetector(com_port='COM3') detector.run()
    )

```

Partie 4:

Détection d'Alcool avec le Capteur MQ3

4.1 Principe

Le capteur MQ3 mesure la concentration de vapeur d'alcool ambiante et renvoie une valeur analogique. Un seuil de 20 a été choisi.

4.2 Tableau des mesures

Valeur MQ3	Interprétation
0–20	Aucun alcool détecté
20+	Alerte : alcool détecté

Table 4.1 – Seuils du capteur MQ3

4.3 Place pour image MQ3

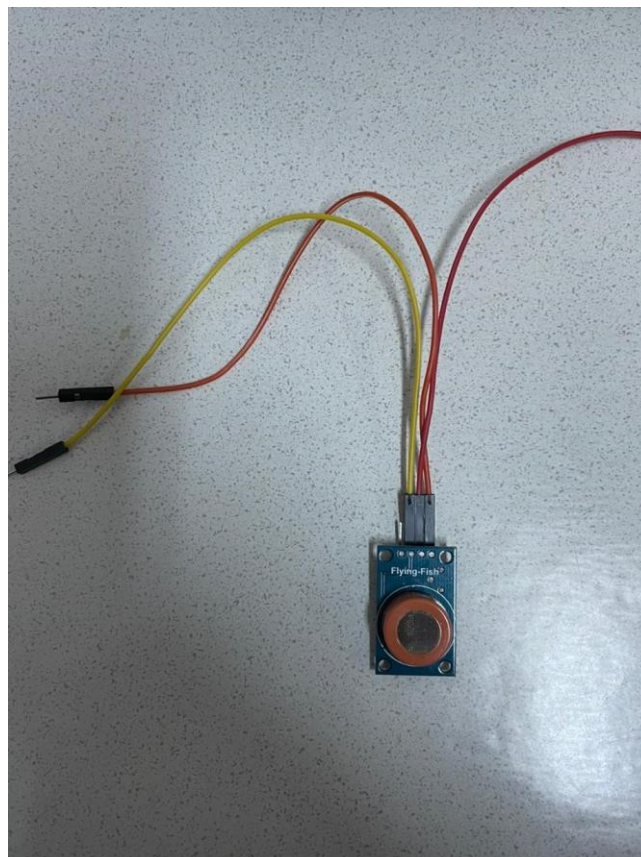


Figure 4.2 – Capteur MQ3 –

4.4 Code Arduino

```
(#include <LiquidCrystal.h> // Add this line at the top

// Pin definitions - ONLY ONCE const int MQ3_PIN = A0;    //
MQ3 sensor on A0 const int BUZZER_PIN = 8;              // Buzzer
on Pin 8 const int LED_PIN = 13;                        // LED on Pin 13

// Initialize LCD with correct pin connections
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

// Threshold for alcohol detection (adjust based on calibration) const int
ALCOHOL_THRESHOLD = 20;
void setup() {
    // Initialize serial communication
    Serial.begin(9600);

    // Initialize LCD lcd.begin(16, 2);
    lcd.print("Breathalyzer");
    lcd.setCursor(0, 1);
    lcd.print("Initializing...");

    // Set pin modes pinMode(BUZZER_PIN,
    OUTPUT); pinMode(LED_PIN, OUTPUT);

    // Wait for sensor warm-up
    delay(2000); lcd.clear();
}

void loop() { // Read MQ3 sensor value int
    sensorValue = analogRead(MQ3_PIN);

    // Display sensor value on LCD
    lcd.clear(); lcd.print("Value: ");
    lcd.print(sensorValue);

    // Display status on second line lcd.setCursor(0,
    1); if (sensorValue > ALCOHOL_THRESHOLD) {
        lcd.print("ALCOHOL DETECTED!");
    }
```

```

        // Activate alarm digitalWrite(LED_PIN, HIGH);
        tone(BUZZER_PIN, 1000, 500); // 1000Hz tone for 500ms
    } else {
        lcd.print("No Alcohol");
        // Turn off alarm digitalWrite(LED_PIN, LOW);
        noTone(BUZZER_PIN);
    }

    // Send data to Serial Monitor
    Serial.print("Sensor Value: ");
    Serial.print(sensorValue);
    Serial.print(" - Status: ");
    Serial.println(sensorValue > ALCOHOL_THRESHOLD ? "ALCOHOL DETECTED" : "CLEAR");

    // Wait before next reading delay(1000);
})

```

Partie 5:

Simulation sur Proteus

5.1 Description

La simulation a été conçue et réalisée par Aya Yahyaoui. Elle comprend l'Arduino, le capteur MQ3, une LED, un buzzer et les connexions nécessaires.

5.2 Place pour la simulation

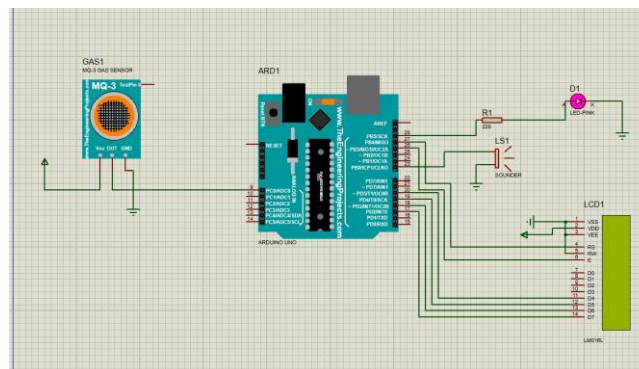


Figure 5.4 – Simulation Proteus

partie 6:

Réalisation CAO–PCB

6.1 le schéma PCB

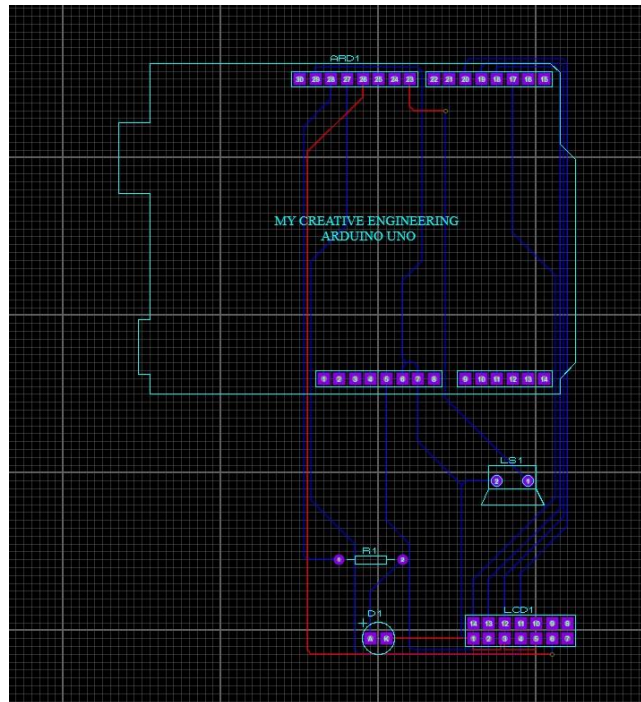


Figure 6.3 – Schéma PCB

Conclusion:

Ce projet nous a permis de combiner électronique, programmation embarquée et vision artificielle dans un système innovant de sécurité routière. La réalisation de la Smart Car nous a permis de :

- comprendre l'utilisation avancée du capteur MQ3,
- maîtriser la communication série Python–Arduino,
- utiliser OpenCV pour analyser les mouvements des yeux,
- concevoir un circuit complet avec simulation et PCB,
- travailler en équipe de manière organisée : Aya pour la simulation et Yosr pour le code.

Le résultat final démontre qu'il est possible de créer un système d'alerte fiable, simple et peu coûteux visant à réduire les risques d'accident liés à la fatigue et à l'alcool.

