AI & SUSTAINABILITY IN VET EDUCATION

Name of the Author/s: TR Team 1

Title of WP: WP3 – AI-SAVES@ProjectHub

Activity Reference: 3.2 – Project template for AI-SAVES@ProjectHub

Logo:



Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Table of Contents

Table of Contents **2**

1. Project Title: 3
2. Project Category: 3
3. Project Description: 3
4. Keywords: 3
5. Target Audience: 3
6. Project Objectives: 3
7. Project Timeline: 3
8. Resources Provided: 4
9. Tools and Technologies Used: 4
10. Step-by-Step Implementation Plan: 4
11. Evaluation Criteria: 8
12. Languages Available: 8
13. Contributors: 8
14. Feedback and Continuous Improvement: 9

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

AI-SAVES@PROJECTHUB

1. **Project Title:**

   Traffic Light Recognition and Controlled Vehicle

2. **Project Category:**

   AI in Robotics

3. **Project Description:**

   This project aims to develop an autonomous vehicle capable of recognizing traffic lights using a Raspberry Pi 5 and Raspberry Camera v2. The project will introduce practical concepts of artificial intelligence and robotics by teaching the vehicle to stop at red lights and proceed at green lights.

4. **Keywords:**

   AI, Robotics, Raspberry Pi, Traffic Light Recognition, Autonomous Vehicle, Python

5. **Target Audience:**

   Both Educators and Students

6. **Project Objectives:**

   - Develop practical knowledge in AI and robotics.
   - Understand applications of image processing and object recognition (traffic lights).
   - Foster hands-on skills with Raspberry Pi and motor control.
   - Improve vocational education skills in integrating AI into teaching practices.

7. **Project Timeline:**
   - Start Date: 01/03/2025
   - End Date: 30/06/2025
   - Key Milestones:
     - Raspberry Pi Setup and Camera Integration: 15/03/2025
     - Motor and Driver Integration: 30/03/2025

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

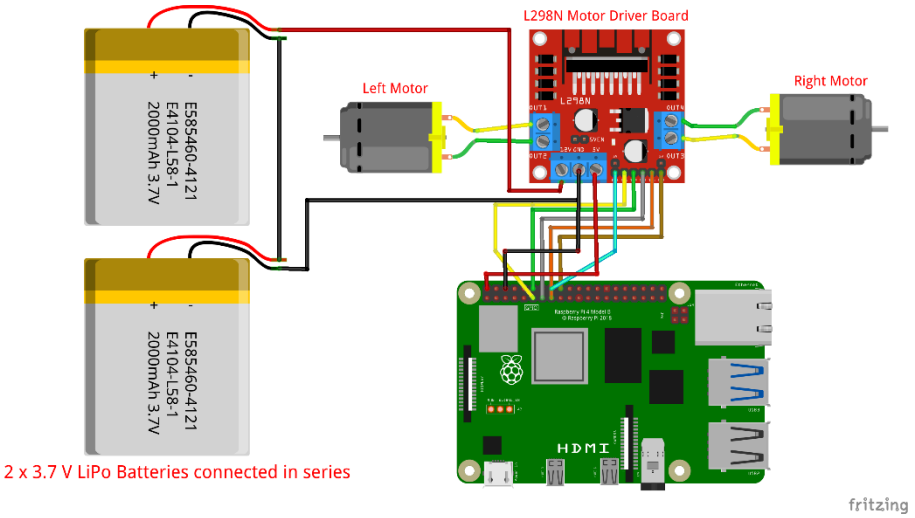
- Image Processing and Traffic Light Recognition Model Training: 15/05/2025
- Vehicle Control Algorithm Development: 15/06/2025
- Project Testing and Optimization: 30/06/2025

8. Resources Provided:

- Raspberry Pi 5 8GB RAM (Hardware)
- Raspberry Camera v2 (Hardware)
- Two yellow motors, L298N motor driver, LIPO 3S 1000mAh, car body, castor wheel materials (Hardware)
- Python programming guides for AI implementation.
- Lesson plan: "Introduction to AI and Computer Vision" (PDF).
- Project report template (DOCX).

9. Tools and Technologies Used:

- Raspberry Pi 5 8GB RAM
- Raspberry Camera v2
- Python programming language
- OpenCV library (for image processing)
- TensorFlow/Keras (for machine learning model)
- Thonny IDE (for Python development)
- GitHub for version control and project collaboration



10. Step-by-Step Implementation Plan:

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- **Step 1: Hardware Setup and Connections**
  - Set up the Raspberry Pi 5 with the necessary operating system (Raspberry Pi OS).
  - Connect the Raspberry Camera v2 to the Raspberry Pi.
  - Connect the motors and L298N motor driver to the Raspberry Pi's GPIO pins.
  - Connect the LIPO 3S 1000mAh battery to the motor driver.
  - Assemble the car body and mount the motors, wheels, Raspberry Pi, and other components.

- **Step 2: Development Environment Setup**
  - Install Python and necessary libraries (OpenCV, TensorFlow/Keras) on the Raspberry Pi.
  - Set up a development environment like Thonny IDE.

- **Step 3: Training the Traffic Light Recognition Model**
  - Collect or use an existing dataset of traffic light images.
  - Train an image classification model (e.g., CNN-based) using the collected data. The model should be able to recognize red, green, and other colors/states.
  - Convert the model to a format suitable for integration with Raspberry Pi (e.g., TensorFlow Lite).

- **Step 4: Developing the Vehicle Control Algorithm**
  - Obtain a live video stream from the Raspberry Camera.
  - Use the trained model to recognize traffic lights in each frame of the video stream.
  - Write Python code to control the vehicle's movement based on the recognized traffic light color:
    - Stop the motors when a red light is detected.
    - Move the motors forward when a green light is detected.
    - Define a default behavior when no light is detected or in ambiguous situations.

- **Step 5: Integration and Testing**
  - Integrate all hardware and software components.
  - Test if the vehicle responds correctly to traffic lights.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

- Conduct tests under different lighting conditions and angles to optimize the performance of the model and control algorithm.

- **Step 6: Monitoring Project Progress and Evaluation**
  - Monitor project progress through weekly reviews.
  - Evaluate student understanding through project demonstrations.

Python Code Example (Basic Structure):

```
import cv2
import numpy as np
import RPi.GPIO as GPIO
import time
from tensorflow.keras.models import load_model # If using
Keras/TensorFlow model

# GPIO pin definitions (for motor driver L298N)
# These pin numbers should be adjusted according to your Raspberry
Pi's GPIO scheme
IN1 = 17
IN2 = 18
IN3 = 27
IN4 = 22
ENA = 23 # Enable A for Motor 1
ENB = 24 # Enable B for Motor 2

GPIO.setmode(GPIO.BCM)
GPIO.setup(IN1, GPIO.OUT)
GPIO.setup(IN2, GPIO.OUT)
GPIO.setup(IN3, GPIO.OUT)
GPIO.setup(IN4, GPIO.OUT)
GPIO.setup(ENA, GPIO.OUT)
GPIO.setup(ENB, GPIO.OUT)

# PWM settings (for speed control)
pwmA = GPIO.PWM(ENA, 100) # 100 Hz frequency
pwmB = GPIO.PWM(ENB, 100)
pwmA.start(0) # Motors stopped initially
pwmB.start(0)

# Motor control functions
def move_forward():
    GPIO.output(IN1, GPIO.HIGH)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.HIGH)
    GPIO.output(IN4, GPIO.LOW)
    pwmA.ChangeDutyCycle(60) # 60% speed
    pwmB.ChangeDutyCycle(60)
    print("Moving forward")
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
def stop_motors():
    GPIO.output(IN1, GPIO.LOW)
    GPIO.output(IN2, GPIO.LOW)
    GPIO.output(IN3, GPIO.LOW)
    GPIO.output(IN4, GPIO.LOW)
    pwmA.ChangeDutyCycle(0)
    pwmB.ChangeDutyCycle(0)
    print("Motors stopped")

# Load traffic light recognition model
# This part will vary depending on the file path and structure of your
trained model
# model = load_model('path/to/your/traffic_light_model.h5')
# LABELS = ['red', 'green', 'yellow', 'none'] # Output labels of your
model

def preprocess_frame(frame):
    # Preprocessing steps suitable for your model
    # Resizing, normalization, etc.
    resized_frame = cv2.resize(frame, (224, 224)) # Example size
    normalized_frame = resized_frame / 255.0
    return np.expand_dims(normalized_frame, axis=0)

def predict_traffic_light(frame):
    # Process the frame and get prediction from the model
    # processed_frame = preprocess_frame(frame)
    # predictions = model.predict(processed_frame)
    # predicted_class_index = np.argmax(predictions[0])
    # return LABELS[predicted_class_index]

    # Simple color detection (temporary placeholder for the model)
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # HSV color ranges for red light
    lower_red1 = np.array([0, 100, 100])
    upper_red1 = np.array([10, 255, 255])
    lower_red2 = np.array([160, 100, 100])
    upper_red2 = np.array([180, 255, 255])

    # HSV color range for green light
    lower_green = np.array([40, 40, 40])
    upper_green = np.array([80, 255, 255])

    mask_red1 = cv2.inRange(hsv, lower_red1, upper_red1)
    mask_red2 = cv2.inRange(hsv, lower_red2, upper_red2)
    mask_red = cv2.bitwise_or(mask_red1, mask_red2)

    mask_green = cv2.inRange(hsv, lower_green, upper_green)

    red_pixels = cv2.countNonZero(mask_red)
    green_pixels = cv2.countNonZero(mask_green)
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
    if red_pixels > 1000: # Adjust threshold value through
experimentation
        return "red"
    elif green_pixels > 1000: # Adjust threshold value through
experimentation
        return "green"
    else:
        return "none"


# Initialize camera
cap = cv2.VideoCapture(0) # 0 for default camera

if not cap.isOpened():
    print("Camera could not be opened!")
    exit()

try:
    while True:
        ret, frame = cap.read()
        if not ret:
            print("Frame could not be retrieved!")
            break

        # Resize the image (for processing speed)
        frame = cv2.resize(frame, (320, 240))

        # Traffic light detection
        traffic_light_status = predict_traffic_light(frame)

        if traffic_light_status == "red":
            stop_motors()
        elif traffic_light_status == "green":
            move_forward()
        else:
            # Stop if no light or ambiguous
            stop_motors()

        # Display the image on screen (optional, not needed if no GUI
on Raspberry Pi)
        cv2.imshow('Frame', frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

        time.sleep(0.1) # Short delay to reduce processor load

except KeyboardInterrupt:
    print("Program terminated.")
finally:
    cap.release()
    cv2.destroyAllWindows()
    GPIO.cleanup()
```

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

11. Evaluation Criteria:

- **Student Feedback:** Gather feedback on the learning process through surveys.
- **Project Success Metrics:** Successful completion of robotic tasks using AI applications.
- **Skills Assessment:** Measure students' proficiency in AI programming and robotics assembly.

12. Languages Available:

English.

13. Contributors:

Türkiye–Project Team 1

- **Mehmet Emin Uğur** (Project Lead) – Lead educator responsible for project development and coordination.
- **Harun Yenişan** (Technical Expert) – Provided expertise in AI programming and robotics.
- **Ali İmran Şahin** (Curriculum Designer) – Developed lesson plans for AI-based handwriting recognition.
- **Ahmet Karagöz** (Instructional Material Developer) – Prepared supporting documents and coding materials used in the project.

14. Feedback and Continuous Improvement:

- **Surveys:** At the end of each milestone, students and educators will complete feedback surveys to assess the effectiveness of the project.
- **Project Reviews:** After project completion, a final review meeting will be held to discuss potential improvements for future AI-based projects.
- **Data Analysis:** Feedback data will be analyzed to identify strengths and areas for improvement, informing future projects.

Co-funded by the European Union

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.