AI & SUSTAINABILITY IN VET EDUCATION

Name of the Author/s: Sucessos Criativos, Lda

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

1. Project Title: 4
2. Project Category: 4
3. Project Description: 4
4. Keywords: 4
5. Target Audience: 4
6. Project Objectives: 4
7. Project Timeline: 5
8. Resources Provided: 5
9. Tools and Technologies Used: 5
10. Step-by-Step Implementation Plan: 6
11. Evaluation Criteria: 14
12. Languages Available: 14
13. Contributors: 14
14. Feedback and Continuous Improvement: 15

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.

1. Project Title:

AI-Based Automatic Pet Door Using Vision and Distance Sensing

---

2. Project Category:

AI in Robotics

---

3. Project Description:

This project presents the development of an intelligent automatic door system designed specifically for pets, using computer vision and proximity sensing. The system continuously monitors the environment through a camera and applies an object detection model to identify the presence of pets, namely cats and dogs. To avoid accidental activation caused by distant detections or false positives, visual recognition alone is not sufficient to trigger the mechanism.

Once a pet is visually detected, an ultrasonic distance sensor is used to confirm that the animal is physically close to the door. Only when both conditions are met - correct pet detection and proximity within a defined threshold - the system activates a servo motor to open the door. The door remains open for a configurable amount of time and then automatically closes. This project demonstrates how AI perception and sensor-based validation can be combined to create reliable and safe automation systems for real-world domestic applications.

---

4. Keywords:

Raspberry Pi, Pet Automation, Smart Door, Computer Vision

---

5. Target Audience:

Both Educators and Students

---

6. Project Objectives:

- Detect pets (cats and dogs) in real time using a camera and an object detection model.

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.

- Combine computer vision with ultrasonic distance sensing to validate pet proximity.
- Control a servo motor to automatically open and close a door mechanism.
- Prevent unintended door activation by requiring multiple confirmation conditions.
- Understand how AI perception can be integrated with physical actuators.
- Explore real-world automation scenarios for smart home and pet care solutions.

---

7. Project Timeline:

With the provided resources, this project should take approximately 4-5 hours or less to be done calmly.

---

8. Resources Provided:

- Python codebase containing pet detection, distance measurement and servo control.
- Real-time visual feedback displaying detection status and door state
- Demo video of the result

---

9. Tools and Technologies Used:

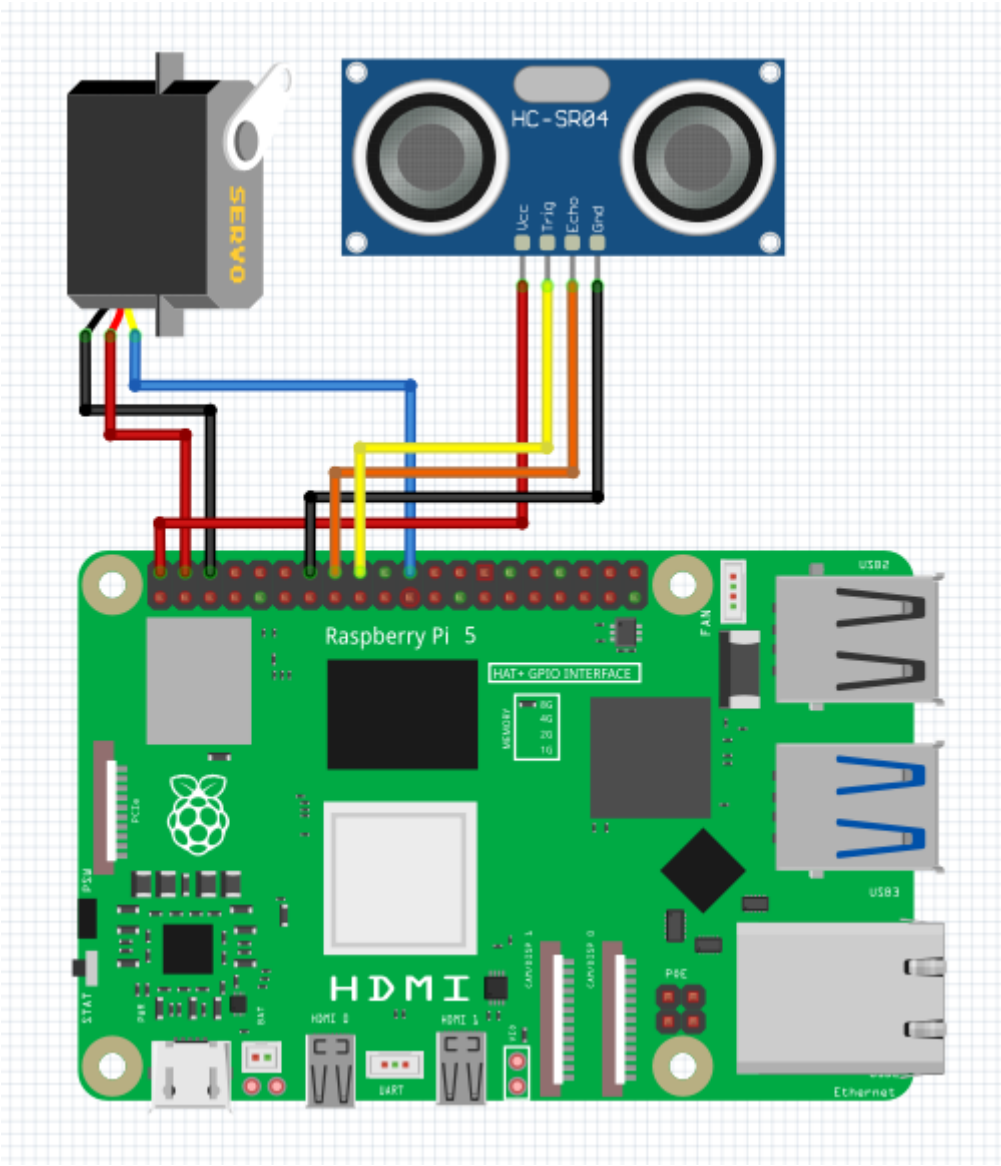
- Raspberry Pi 5
- Raspberry Pi Camera Module 4 v2
- Python programming language
- OpenCV for camera input and image processing
- Pre-trained Object Detection Model (YOLO11-Nano)
- Ultrasonic distance sensor
- Servo motor for door actuation
- GPIO and lgpio libraries for hardware control

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.

10. Step-by-Step Implementation Plan:

1. **Hardware Preparation:** Connect Raspberry Pi, the camera module, the servo motor and the ultrasonic sensor according to the wiring diagram.



2. **Structure Setup:** Create the structure for the result (the door and the wall to have ultrasonic sensor/camera fixed for test purposes).

3. **Software Setup:** Install the necessary software on Raspberry Pi, including Raspbian OS and Python libraries including OpenCV, Picamera2, Ultralytics YOLO, and GPIO libraries.

4. **Execute the Code:** Copy the provided code bellow and save it on your Raspberry Pi.

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.

```
import cv2

from picamera2 import Picamera2

from ultralytics import YOLO

import RPi.GPIO as GPIO

import lgpio

import time


# ===== PIN SETUP =====

# Ultrasonic sensor pins

TRIG = 24

ECHO = 23


# Servo pin

SERVO_PIN = 25


# ===== GPIO SETUP =====

# Setup for ultrasonic sensor (using RPi.GPIO)

GPIO.setmode(GPIO.BCM)

GPIO.setup(TRIG, GPIO.OUT)

GPIO.setup(ECHO, GPIO.IN)


# Setup for servo (using lgpio)

h = lgpio.gpiochip_open(0)

lgpio.gpio_claim_output(h, SERVO_PIN)


# ===== CAMERA SETUP =====

picam2 = Picamera2()
```

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.

```
picam2.preview_configuration.main.size = (640, 400)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()


# ===== YOLO MODEL =====
model = YOLO("yolo11n.pt")


# ===== CONFIGURATION =====
DISTANCE_THRESHOLD = 25  # cm
DOOR_OPEN_TIME = 15  # seconds
DOOR_CLOSED_ANGLE = 90
DOOR_OPEN_ANGLE = 0


door_is_open = False
door_open_timestamp = None



def measure_distance():
    """Measure distance using ultrasonic sensor"""
    try:
        # Ensure trigger is low
        GPIO.output(TRIG, False)
        time.sleep(0.01)


        # Send 10µs pulse to TRIG
        GPIO.output(TRIG, True)
```

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.

```
        time.sleep(0.00001)
        GPIO.output(TRIG, False)

        # Measure pulse duration on ECHO
        pulse_start = time.time()
        pulse_end = time.time()
        timeout = time.time() + 0.1  # 100ms timeout

        while GPIO.input(ECHO) == 0 and time.time() < timeout:
            pulse_start = time.time()

        while GPIO.input(ECHO) == 1 and time.time() < timeout:
            pulse_end = time.time()

        pulse_duration = pulse_end - pulse_start
        distance = pulse_duration * 34300 / 2  # Speed of sound 343 m/s

        return distance if distance < 400 else 400  # Max reasonable distance
    except:
        return 400  # Return max distance on error


def set_servo_angle(h, pin, angle):
    """Move servo to specified angle (0-180 degrees)"""
    pulse_width = 0.5 + (angle / 180.0) * 2.0  # milliseconds
    period = 20  # 20ms period = 50Hz

    # Send pulses for 0.5 seconds
```

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.

```
    end_time = time.time() + 0.5
    while time.time() < end_time:
        lgpio.gpio_write(h, pin, 1)
        time.sleep(pulse_width / 1000)
        lgpio.gpio_write(h, pin, 0)
        time.sleep((period - pulse_width) / 1000)



def open_door():
    """Open the door by rotating servo"""
    global door_is_open, door_open_timestamp
    print("> OPENING DOOR")
    set_servo_angle(h, SERVO_PIN, DOOR_OPEN_ANGLE)
    door_is_open = True
    door_open_timestamp = time.time()



def close_door():
    """Close the door by rotating servo"""
    global door_is_open, door_open_timestamp
    print("> CLOSING DOOR")
    set_servo_angle(h, SERVO_PIN, DOOR_CLOSED_ANGLE)
    door_is_open = False
    door_open_timestamp = None



print("> Pet Door System Started!")
print(f"Distance threshold: {DISTANCE_THRESHOLD}cm")
```

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.

```
print(f"Door open time: {DOOR_OPEN_TIME}s")

print("Press 'q' to quit\n")


# Ensure door starts closed

close_door()


try:

    while True:

        # Capture frame

        frame = picam2.capture_array()


        # Run YOLO inference

        results = model(frame)


        annotated_frame = frame.copy()

        pet_detected = False


        # Check for cats/dogs

        for result in results:

            for box in result.boxes:

                cls_id = int(box.cls[0])

                label = result.names[cls_id]


                if label.lower() == "cat" or label.lower() == "dog":

                    pet_detected = True

                    confidence = box.conf[0] * 100

                    x1, y1, x2, y2 = map(int, box.xyxy[0])
```

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.

```
                    # Draw bounding box
                    cv2.rectangle(annotated_frame, (x1, y1), (x2, y2), (0,
255, 0), 2)
                    label_text = f"{label} {confidence:.2f}%"
                    cv2.putText(
                        annotated_frame,
                        label_text,
                        (x1, y1 - 10),
                        cv2.FONT_HERSHEY_SIMPLEX,
                        0.5,
                        (0, 255, 0),
                        2
                    )

        # If pet detected, check distance
        if pet_detected and not door_is_open:
            distance = measure_distance()
            print(f"> Pet detected! Distance: {distance:.2f}cm")

            if distance <= DISTANCE_THRESHOLD:
                open_door()

        # Check if door should close
        if door_is_open and door_open_timestamp:
            elapsed = time.time() - door_open_timestamp
            if elapsed >= DOOR_OPEN_TIME:
                close_door()
            else:
```

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.

```
            # Show countdown on frame
            remaining = int(DOOR_OPEN_TIME - elapsed)
            cv2.putText(
                annotated_frame,
                f"Door closes in: {remaining}s",
                (10, 30),
                cv2.FONT_HERSHEY_SIMPLEX,
                0.7,
                (0, 0, 255),
                2
            )

    # Show status on frame
    status = "DOOR OPEN" if door_is_open else "DOOR CLOSED"
    color = (0, 255, 0) if door_is_open else (0, 0, 255)
    cv2.putText(
        annotated_frame,
        status,
        (10, 60),
        cv2.FONT_HERSHEY_SIMPLEX,
        0.7,
        color,
        2
    )

    # Display frame
    cv2.imshow("Pet Door System", annotated_frame)
```

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.

```
        # Quit with 'q'
        if cv2.waitKey(1) == ord('q'):
            break


except KeyboardInterrupt:
    print("\n> Interrupted by user")


finally:
    print("> Cleaning up...")
    close_door()
    cv2.destroyAllWindows()
    picam2.stop()
    GPIO.cleanup()
    lgpio.gpiochip_close(h)
```

---

11. Evaluation Criteria:

- **System Functionality:** Ability to replicate the project.
- **Creativity:** Suggestions for improvements (new features, etc.). If suggested and tried (with success or failure) should be more rewarded.
- **Optimizations:** Voluntary optimizations for the code with success or failure, especially on pet detection accuracy.

---

12. Languages Available:

English

---

13. Contributors:

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.

- **Fátima Pais, Luís Soares** – Project Designers and Facilitators

---

14. Feedback and Continuous Improvement:

Describe how feedback will be collected from participants and how this feedback will be used to improve future projects or iterations of this project.

- **Post-Implementation Survey:** After deploying the project, feedback will be collected through surveys targeting participants involved in the testing and observation process.
- **Team Reflection Sessions:** The project team will organize internal review sessions to discuss challenges faced, what worked well, and areas that need refinement.
- **Technical Evaluation:** Observations regarding detection accuracy, system stability, and hardware durability will be documented for improving future versions.
- **Iterative Updates:** Based on feedback and performance, updates to the software code and system design will be implemented to improve reliability and effectiveness.

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.