AI & SUSTAINABILITY IN VET EDUCATION

Name of the Author/s: Sucessos Criativos, Lda | P. Porto

Title of WP: WP3 – AI-SAVES@ProjectHub

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

European Vocational Training Association

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: 2

2. Project Category: 2

3. Project Description: 2

5. Target Audience: 2

6. Pre-Requisite knowledge: 2

7. Project Objectives: 3

8. Project Timeline: 3

9. Resources Provided: 3

10. Tools and Technologies Used: 4

11. Step-by-Step Implementation Plan: 6

12. Evaluation Criteria: 15

13. Languages Available: 15

14. Contributors: 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:

Apple Detection and Disease Classification System with Raspberry Pi and Computer Vision

---

2. Project Category:

- AI in IoT

---

3. Project Description:

This project develops an intelligent agricultural monitoring system using a Raspberry Pi and YOLOv11 neural network to detect apples and classify their health status in real-time. The system uses computer vision to identify four different apple conditions: Healthy, Blotch, Rot, and Scab diseases. Through live camera feed processing, the system provides immediate visual feedback with confidence scoring, making it invaluable for precision agriculture, quality control, and automated crop assessment.

---

4. Keywords:

Apple Detection, Disease Classification, YOLOv11, Computer Vision, Raspberry Pi, Agricultural AI, Real-time Processing, Crop Monitoring, Quality Control

---

5. Target Audience:

Both Educators and Students

---

6. Pre-Requisite knowledge:

- Basic understanding of Python programming (functions, loops, object-oriented programming)

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.

- Introduction to computer vision concepts and image processing
- Knowledge of Raspberry Pi setup and GPIO control
- Basic understanding of neural networks and machine learning concepts
- Familiarity with camera modules and video processing
- Understanding of agricultural plant diseases (optional but helpful)

---

7. Project Objectives:

- Introduce students to advanced AI applications in agriculture
- Teach real-time computer vision and object detection techniques
- Develop skills in neural network implementation and deployment
- Raise awareness about precision agriculture and sustainable farming
- Learn model training, validation, and deployment processes
- Create practical solutions for agricultural quality control and monitoring
- Understand the integration of AI with IoT devices for smart applications

---

8. Project Timeline:

---

9. Resources Provided:

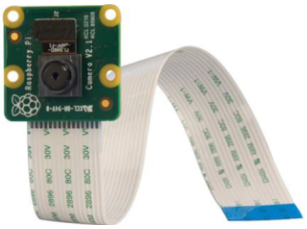
- Raspberry Pi 5
- Raspberry Pi Camera Module v2 or v3
- Keyboard for system control
- Python code templates and libraries
- YOLOv11 training Jupyter notebook
- Sample apple dataset for training (from Kaggle)
- Step-by-step implementation guide
- Troubleshooting documentation

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. Tools and Technologies Used:

10.1 Hardware:

MATERIAL	
Raspberry Pi 5	
27W USB Type-C Power Supply for Raspberry Pi 5	
Raspberry Pi Camera Board V2 8MP 1080p (IM8)	
Official FPC camera cable for Raspberry Pi 5 200mm (IM11)	

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.

10.2 Software:

- **Operating System:** Raspberry Pi OS
- **Programming Language:** Python 3.8+
- **Development Environment:** Thonny IDE or VS Code
- **Model Training Platform:** Google Colab with GPU acceleration
- **AI Framework:** YOLOv11 (You Only Look Once v11)
- **Model Format:** PyTorch (.pt)

Libraries: time, cv2, YOLO, PiCamera2, os, kagglehub, ultralytics.YOLO, shutil, yaml, pathlib.Path, cv2, numpy, sklearn, glob, matplotlib.pyplot, seaborn, PIL.Image, PIL.ImageDraw, PIL.ImageFont, pandas, json, datetime, torch, tqdm

---

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. Step-by-Step Implementation Plan:

11.1 Hardware Setup

Camera Module Installation

1. **Power down** the Raspberry Pi completely
2. **Connect the camera module** to the CSI port
3. **Secure the connection** and ensure proper seating
4. **Power up** and enable camera interface

Hardware Verification

```
# Test camera detection
vcgencmd get_camera
# Expected output: supported=1 detected=1

# Test camera functionality
libcamera-hello --timeout 5000
```

11.2 Model Training (Google Colab)

This section outlines the training process for a YOLOv11-based apple classification model using Google Colab with GPU acceleration. The goal is to develop a lightweight model capable of classifying apples into four distinct categories in real time on a Raspberry Pi.

Environment Setup

- Open Google Colab.
- Enable GPU:
  Runtime → Change runtime type → Hardware accelerator: GPU.
- Upload and open the notebook `train_model.ipynb`.
- Install required libraries and configure training parameters.

Dataset Configuration

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.

- Dataset consists of labeled images of apples in four categories:
  - **Healthy**
  - **Apple Rot**
  - **Scab**
  - **Blotch**
- Images are sourced from open datasets (e.g., Kaggle) and organized into class-specific folders.
- Converted to YOLOv11-compatible classification format with `.txt` label files per image.

Training Process

- **Preprocessing**: Image resizing, normalization, and optional augmentation (rotation, flipping, brightness).
- **Model**: YOLOv11n (classification variant optimized for edge deployment).
- **Training execution**
- **Evaluation**: Performed automatically at each epoch, tracking classification accuracy, loss, and confusion matrix.
- **Export**: Final model saved in `.pt` format for deployment on Raspberry Pi

Output

- The resulting `yolov11_apple.pt` model is optimized for inference on Raspberry Pi.
- Capable of classifying apple images into the four defined health categories in real time.

Output visualizations and metrics can be reviewed directly within Colab.

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.

11.3 Software Setup

Now that the hardware is ready, it's time to implement the light tracking control system.

1. Create Virtual Environment

```
# Create project directory
mkdir apple_detection_system
cd apple_detection_system

# Create virtual environment
python3 -m venv venv

# Activate virtual environment
source venv/bin/activate
```

2. Install Dependencies

```
# Install requirements
pip install -r requirements.txt
```

3. Add the Code

- Copy the apple classification system Python code to `main.py`
- Make sure the file is in your project directory

4. Add the model

- Copy the apple classification system model to `model/yolov11_apple.pt`
- Make sure the file is in your project directory

5. Run the System

```
# Make sure virtual environment is activated
source venv/bin/activate

# Run the light apple detection system
python main.py
```

6. Stop the System

- Press Ctrl+C to stop the program safely

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.

System Behavior:

The apple detection system operates with intelligent behavior patterns designed for reliable agricultural monitoring:

Detection Logic Flow

Initialization

- Condition: System startup
- Behavior: Camera preview activation
- Visual Feedback: Clean video feed

Model Loading

- Condition: AI model ready
- Behavior: YOLOv11 neural network active
- Visual Feedback: Status confirmation

Active Scanning

- Condition: Processing frames
- Behavior: Real-time inference running
- Visual Feedback: Live detection overlay

Apple Detected

- Condition: Confidence ≥ 70%
- Behavior: Best detection displayed
- Visual Feedback: Red text with class + confidence

Low Confidence

- Condition: Confidence < 70%
- Behavior: Detection filtered out
- Visual Feedback: No overlay shown

Multiple Apples

- Condition: Several detections
- Behavior: Highest confidence selected
- Visual Feedback: Single best result displayed

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.

No Detection

- Condition: No apples found
- Behavior: Clean processing
- Visual Feedback: Clear video feed

Detection States and Responses

🍎 Healthy Apple Detection

- Trigger: `APPLE_Healthy_Apple` class detected
- Display: Green-tinted text "APPLE_Healthy_Apple XX.XX%"
- Meaning: Normal, marketable fruit identified
- Agricultural Value: Quality control confirmation

🟤 Disease Detection States

Blotch Disease Detection

- Trigger: `APPLE_Blotch_Apple` class detected
- Meaning: Fungal infection detected
- Action Recommended: Isolation and treatment assessment

Rot Disease Detection

- Trigger: `APPLE_Rot_Apple` class detected
- Meaning: Advanced decay detected
- Action Recommended: Immediate removal from batch

Scab Disease Detection

- Trigger: `APPLE_Scab_Apple` class detected
- Meaning: Apple scab disease identified
- Action Recommended: Grade separation for processing

Confidence Threshold Behavior

90-100% Confidence Range

- System Response: High confidence detection
- Display Action: Bold display with strong color
- Reliability Level: **Excellent** - Definitive classification

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.

80-89% Confidence Range

- System Response: Good confidence detection
- Display Action: Standard display
- Reliability Level: **Good** - Reliable classification

70-79% Confidence Range

- System Response: Minimum acceptable detection
- Display Action: Lighter display
- Reliability Level: **Acceptable** - Monitor for consistency

Below 70% Confidence Range

- System Response: Detection filtered out
- Display Action: No display
- Reliability Level: **Unreliable** - Ignored to prevent false positives

Frame Processing Pipeline:

1. **Image Capture** (5-10 FPS)
2. **AI Inference** (~100-500ms per frame)
3. **Confidence Filtering** (70% threshold applied)
4. **Best Detection Selection** (highest confidence wins)
5. **Visual Overlay Rendering** (real-time text display)
6. **Display Update** (continuous video feed)

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.

CODE

```python
import cv2
from picamera2 import Picamera2
from ultralytics import YOLO
import time

# Initialize the camera
picam2 = Picamera2()
picam2.preview_configuration.main.size = (800, 600)
picam2.preview_configuration.main.format = "RGB888"
picam2.preview_configuration.align()
picam2.configure("preview")
picam2.start()

# Load the YOLO model

model = YOLO("model/yolov11_apple.pt")

# Detection loop
while True:
    # Capture frame
    frame = picam2.capture_array()

    # Run YOLO inference on the frame
    results = model(frame)

    # Create a copy of the frame for annotation
    annotated_frame = frame.copy()

    # Initialize variables to track the best detection
    best_label_text = ""

    # Process detection results
    for result in results:
        highest_conf = 0
        best_label = ""

        # Iterate through all detected boxes
        for box in result.boxes:
            # Get confidence score as percentage
            confidence = float(box.conf[0]) * 100
```

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.

```
            # Only consider detections with confidence > 70% and higher than
current best
            if confidence > 70 and confidence > highest_conf:
                cls_id = int(box.cls[0])  # Get class ID
                best_label = result.names[cls_id]  # Get class name
                highest_conf = confidence  # Update highest confidence

        # Format the best detection text if found
        if best_label:
            best_label_text = f"{best_label} {highest_conf:.2f}%"

    # Display the detection text on the frame
    if best_label_text:
        cv2.putText(
            annotated_frame,
            best_label_text,
            (30, 60),  # Position (x, y)
            cv2.FONT_HERSHEY_SIMPLEX,  # Font type
            1.5,  # Font scale
            (0, 0, 255),  # Color (BGR format - red)
            3,  # Thickness
            cv2.LINE_AA  # Anti-aliasing
        )

    # Display the frame in a window
    cv2.imshow("Camera", annotated_frame)

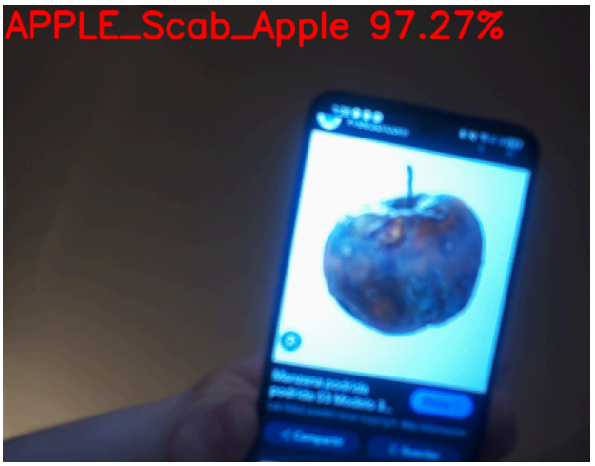
# Clean up resources
cv2.destroyAllWindows()
```

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.

Now, you can run the program and see the result:







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.

12. Evaluation Criteria:

- **Model Training and Performance (60%):** Correct execution of the provided Jupyter notebook for model training, proper configuration of training parameters, successful model convergence, evaluation of model accuracy and confidence thresholds, and ability to deploy and run the trained YOLOv11 model on the Raspberry Pi with real-time inference.
- **Hardware Setup (10%):** Proper installation and connection of the Raspberry Pi and camera module, ensuring stable and continuous video feed capture during system operation.
- **Software Functionality (20%):** Correct execution of detection code without crashes, real-time processing of video frames, accurate classification of apple conditions, filtering out low-confidence detections, and clear visual feedback.
- **Project Documentation and Understanding (10%):** Clear explanation of system architecture, training methodology, step-by-step instructions for setup and running the system, and interpretation of detection results.

---

13. Languages Available:

---

14. 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.