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

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: 6

10. Step-by-Step Implementation Plan: 7

Step 1: Prepare the Raspberry Pi for Hardware Interaction 7

Step 2: Build the Circuit 8

Step 3: Write the Python Program to Control the Window 9

Step 4: Run the Program and Observe The Output 12

11. Evaluation Criteria: 16

12. Languages Available: 16

13. Contributors: 16

14. Feedback and Continuous Improvement: 16

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:

First Steps with AI-VET Kit: Working With The OLED Display

---

2. Project Category:

Microelectronics Fundamentals

---

3. Project Description:

In this project, learners will explore how to display information using a 0.95-inch OLED display connected to a Raspberry Pi 5. Using I^2C communication, they will control the screen via Python and visualize text or images, such as sensor readings or logos. This project develops skills in data visualization, screen initialization, and hardware communication protocols—vital aspects of physical computing in smart devices.

---

4. Keywords:

OLED, SSD1306, Raspberry Pi, Python, I2C, Display, Image Rendering

---

5. Target Audience:

Educators, learners, and makers with or without prior experience who are beginning their journey into microelectronics and physical computing.

---

6. Project Objectives:

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.

- Connect and configure an I²C OLED display with a Raspberry Pi.
- Display and update text in real time using Python.
- Load and render images using PIL on SSD1306 OLED.
- Understand I²C protocol usage in sensor/display communication.

---

7. Project Timeline:

The estimated duration is approximately **2.5 to 3 hours**, including environment setup and physical assembly.

---

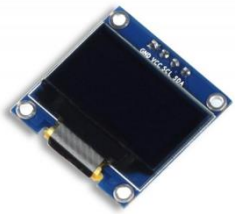
8. Resources Provided:

- AI-VET Kit (Raspberry Pi 5, LED, Breadboard, Jumpers, OLED Display)

MATERIAL	
Raspberry Pi 5	
27W USB Type-C Power Supply for Raspberry Pi 5	

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.

Flat Jumper Wires F/F	
Breadboard	
SINGLE-FILE WIRE FOR BREADBOARDS	
0.95inch OLED i2c Display Blue/Yellow SSD1306	

---

9. Tools and Technologies Used:

- Raspberry Pi OS
- Python 3 (virtual environment)
- `adafruit-circuitpython-ssd1306` library
- PIL (Python Imaging Library)
- Geany IDE
- Terminal (bash)

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:

Step 1: Prepare the Raspberry Pi for Hardware Interaction

This project builds directly on the tasks completed in **"Getting Started with Raspberry Pi 5: Setup, Configuration, and Hello World in Python."** If you're starting here, or need a refresher, you'll begin by setting up your Raspberry Pi for I²C communication and Python development.

In this step, you will:

- Install Raspberry Pi OS using the **Raspberry Pi Imager** tool.
- Configure system settings such as **Wi-Fi**, **keyboard layout**, and **time zone**.
- Enable key interfaces — especially **I²C**, but also **SPI**, **1-Wire**, and **VNC** — via the **Raspberry Pi Configuration** menu.
- Use the terminal command `ifconfig` to identify the Raspberry Pi's IP address and connect remotely via **TigerVNC Viewer**.
- Install required Python packages, especially the Adafruit SSD1306 driver, using.This library specifically provides functions and tools to interact with SSD1306 OLED displays via I2C or SPI communication. To install run the following command::

```
pip3 install adafruit-circuitpython-ssd1306
```

While this project doesn't require low-level GPIO control, it introduces you to **I²C communication**, display rendering, and using external libraries in Python for embedded interfaces.

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 2: Build the Circuit

In this step, you'll wire the **OLED display module** to the Raspberry Pi using the I^2C communication protocol. I^2C (Inter-Integrated Circuit) is a common protocol used to connect sensors and displays with just two data lines.

OLED Display (Visual Output):

The OLED screen receives text or images from the Raspberry Pi and visually displays them. It uses the **I^2C interface**, which simplifies wiring by sharing two data lines among multiple devices.

Connections:

- **VCC → Raspberry Pi 5V (pin 2)**
  Supplies power to the OLED display.

- **GND → Raspberry Pi GND (pin 6)**
  Connects the display's ground to the Raspberry Pi's ground.

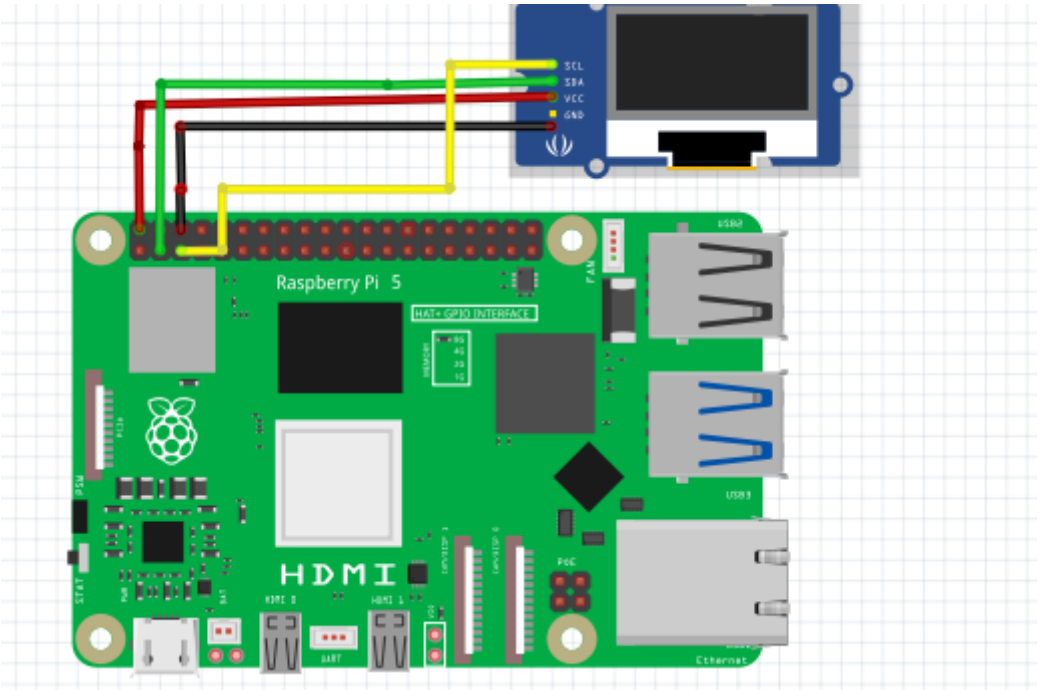
- **SDA (Serial Data Line) → GPIO 2 (pin 3)**
  Transmits data from the Raspberry Pi to the OLED display.

- **SCL (Serial Clock Line) → GPIO 3 (pin 5)**
  Sends timing signals used to synchronize communication.

These four connections are all you need to communicate with the OLED screen. Once wired, the Raspberry Pi will send pixel data through the I^2C bus to control what's 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.



Step 3: Write the Python Program to Control the Window

With the OLED display connected, you'll now write a Python program that initializes the screen and shows centered text. This program helps you understand how to send visual output through I^2C using a simple graphics library.

Open Geany and enter the following code:

```
import board
import digitalio
from PIL import Image, ImageDraw, ImageFont
```

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 adafruit_ssd1306
# Initialize OLED display dimensions (128x64 pixels)
WIDTH = 128
HEIGHT = 64
# Set up I2C communication with the OLED display
# Uses the default I2C pins on the board (SCL and SDA)
i2c = board.I2C()  # Initializes I2C communication
# Create an SSD1306 OLED display instance with the specified width, height,
and I2C interface
oled = adafruit_ssd1306.SSD1306_I2C(WIDTH, HEIGHT, i2c, addr=0x3C)
# Clear the OLED display (fills the screen with black)
oled.fill(0)
oled.show()
# Create a new image object with "1-bit" color (black and white) for drawing
on the display
image = Image.new("1", (oled.width, oled.height))
# Obtain a drawing object from the image to draw shapes and text
draw = ImageDraw.Draw(image)
# Load the default font to use for displaying text on the OLED
font = ImageFont.load_default()
# Define a helper function to calculate the pixel size of the text (width and
height)
def getfontsize(font, text):
    # This calculates the bounding box of the text, returning its width and
height
    left, top, right, bottom = font.getbbox(text)
    return right - left, bottom - top
# Define the text to display on the OLED screen
text = "AI-VET"
# Get the width and height of the text in pixels to help with centering it on
the display
(font_width, font_height) = getfontsize(font, text)
# Draw only the text on the image, centered horizontally and vertically on the
OLED display
draw.text(
    (oled.width // 2 - font_width // 2, oled.height // 2 - font_height // 2),
# Center the text
    text,  # Text to display
    font=font,  # Use the default font
    fill=255    # White color for the text
)
# Send the image (which now contains only the text) to the OLED display
oled.image(image)
# Display the image on the OLED screen
oled.show()
```

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.

Save the file as **`OLED.py`**

Code Breakdown:

Setup and Imports:

The script starts by importing:

- `board` and `digitalio` to handle hardware pin mapping.
- `PIL` (Python Imaging Library) modules for image and text rendering.
- `adafruit_ssd1306` to interface with the OLED display.

Display Initialization:

- The display width and height are defined as 128x64.
- The Raspberry Pi's default I^2C pins are used to set up communication.
- An OLED object is created, cleared, and made ready for drawing.

Drawing the Text:

- A 1-bit image (black and white) is created in memory.
- The string `"AI-VET"` is prepared for display using the default font.
- A helper function calculates the size of the text so it can be centered.
- The text is drawn directly to the image canvas.

Rendering the Output:

- The image is transferred to the OLED screen.
- The `oled.show()` function makes the content visible on the display.

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 4: Run the Program and Observe The Output

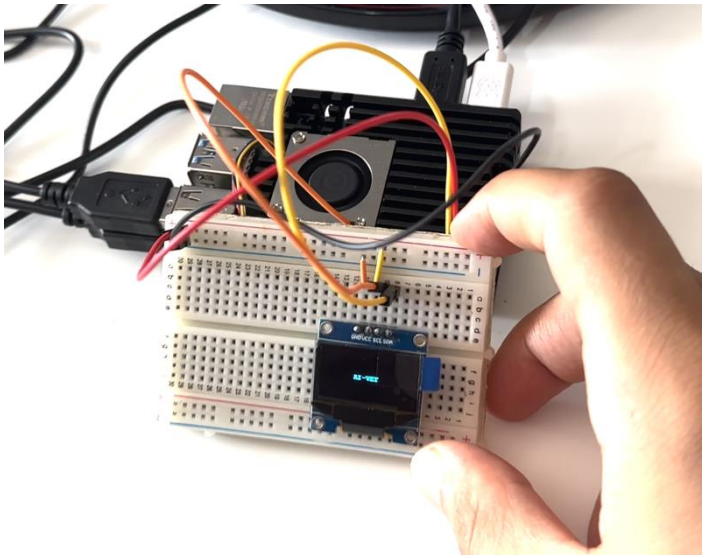
With the code written and saved, it's time to test the setup and observe how it responds to nearby objects.

To run the script:

Save the script as `OLED.py`

Press F5 in **Geany** or **open** the **terminal** and **run**:

python3 oled.py



Step 5: Extend the Program to Alternate Between Text and Image

Now that you've displayed static text, let's take it further. In this step, you'll create a dynamic program that alternates between two screens on the OLED: one displaying the text `"AI-VET"` and the other showing the AI-VET logo.

This exercise introduces:

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.

- Reusable functions for OLED output.
- Image handling with PIL.
- Looping behavior for dynamic displays.

Before You Begin:

- **Download** the AI-VET logo from ai-vet.eu
- **Right-click** the logo and choose “Save Image As”
- **Save** it as `logo.png` in the same folder as your Python script

Open Geany and enter the following code:

```
import time
import board
import digitalio
from PIL import Image, ImageDraw, ImageFont
import adafruit_ssd1306
# Initialize OLED display dimensions (128x64 pixels)
WIDTH = 128
HEIGHT = 64
# Set up I2C communication with the OLED display
i2c = board.I2C()  # Initializes I2C communication
# Create an SSD1306 OLED display instance with the specified width, height,
and I2C interface
oled = adafruit_ssd1306.SSD1306_I2C(WIDTH, HEIGHT, i2c, addr=0x3C)
# Helper function to display text
def display_text(text):
    # Create a new image object with "1-bit" color (black and white) for
drawing on the display
    image = Image.new("1", (oled.width, oled.height))
    draw = ImageDraw.Draw(image)
    # Load the default font to use for displaying text on the OLED
    font = ImageFont.load_default()
    # Calculate the pixel size of the text (width and height)
    font_width, font_height = font.getsize(text)
    # Draw the text on the image, centered horizontally and vertically on the
OLED display
    draw.text(
        (oled.width // 2 - font_width // 2, oled.height // 2 - font_height //
2),  # Center the text
        text,  # Text to display
```

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.

```
        font=font,  # Use the default font
        fill=255  # White color for the text
    )
    # Send the image (which now contains only the text) to the OLED display
    oled.image(image)
    oled.show()
# Helper function to display an image
def display_image(image_path):
    # Load an image from file (replace with the correct path to your image)
    image = Image.open(image_path).convert("1")  # Convert to 1-bit color
    # Get the original image size
    original_width, original_height = image.size
    # Calculate the scaling factor to fit the image within the display
(128x64) without distortion
    scaling_factor = min(WIDTH / original_width, HEIGHT / original_height)
    # Calculate the new size, maintaining aspect ratio
    new_width = int(original_width * scaling_factor)
    new_height = int(original_height * scaling_factor)
    # Resize the image with the new width and height
    image = image.resize((new_width, new_height))
    # Create a blank image the size of the OLED display (128x64 pixels)
    final_image = Image.new("1", (oled.width, oled.height))
    # Calculate position to center the image on the display
    x_position = (oled.width - new_width) // 2
    y_position = (oled.height - new_height) // 2
    # Paste the resized image onto the blank canvas, centered
    final_image.paste(image, (x_position, y_position))
    # Send the final image to the OLED display
    oled.image(final_image)
    oled.show()
# Main loop to alternate between text and image
try:
    while True:
        # Display the text for 2 seconds
        display_text("AI-VET")
        time.sleep(2)
        # Display the image for 2 seconds
        display_image("logo.png")  # Replace with the correct image file path
        time.sleep(2)
except KeyboardInterrupt:
    # Clear the display before exiting
    oled.fill(0)
    oled.show()
    print("Program exited.")
```

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 6: Run the Program and Observe the Alternating Display

With everything in place, it's time to test your upgraded script.

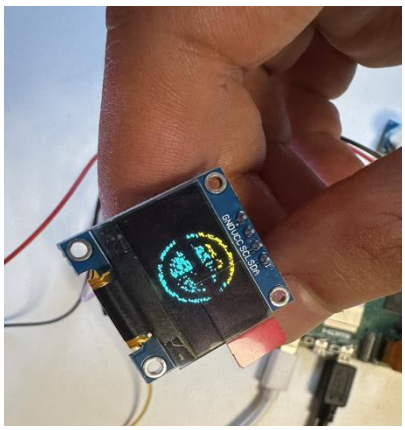
To run the program:

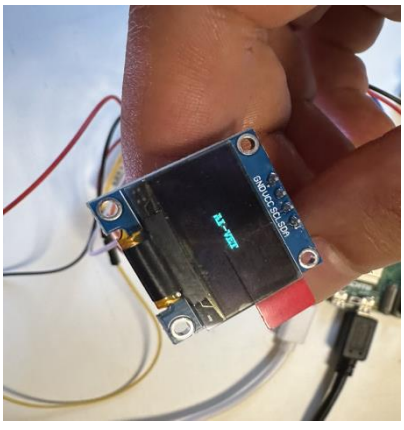
1. **Save** your file as `oled_alternate.py`
2. Make sure the **logo.png i**s in the same folder.
3. **Press F5** in **Geany** or run the script via terminal:

```
python3 oled_alternate.py
```

You should see the OLED:

- First show the text "AI-VET" centered on the screen
- Then switch to the AI-VET logo image
- Continue alternating every 2 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.

11. Evaluation Criteria:

- **Circuit Accuracy:** OLED is correctly connected to I^2C pins (SDA, SCL).
- **Display Initialization:** The screen is successfully detected and cleared by the script.
- **Text Rendering:** Text is properly centered and readable on the screen.
- **Image Handling:** The program successfully loads and resizes images without distortion.
- **Script Stability:** No syntax or runtime errors; script exits cleanly on Ctrl + C.

---

12. Languages Available:

English, **Portuguese** (translations in progress)

---

13. Contributors:

Fátima Pais, Luís Soares – Project Designers and Facilitators

---

14. Feedback and Continuous Improvement:

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.

- **Surveys**: At the end of this 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.