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: Connect the Servo Motor 7
Step 4: Run the Program 10
11. Evaluation Criteria: 11
12. Languages Available: 12
13. Contributors: 12
17. Feedback and Continuous Improvement: 12

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: Controlling a Servo Motor with Raspberry Pi 5

---

2. Project Category:

Microelectronics Fundamentals

---

3. Project Description:

This project introduces participants to the control of motion through servo motors using the Raspberry Pi 5. Building on the previous setup and LED blink project, this activity demonstrates how to use the GPIO system to send PWM (Pulse Width Modulation) signals to rotate a servo to specific angles. It teaches participants how code can control movement and forms the basis for future automation or robotic arm projects.

---

4. Keywords:

Raspberry Pi, Python, Virtual Environment, GPIO, Servo Motor, PWM, Geany

---

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.

- Review and confirm the Raspberry Pi environment and GPIO libraries are ready.
- Build a simple circuit to connect a servo motor.
- Write and run a Python script to rotate the servo to specific angles.
- Understand how PWM signals control the servo's movement.

---

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, Micro Servo)

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	
Microservo SG90 1.6Kg 180º (IM4)	

---

9. Tools and Technologies Used:

- Raspberry Pi OS
- Python 3 (in virtual environment)
- `lgpio` 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 first go through the essential setup steps to prepare your Raspberry Pi to control physical components like a servo motor.

In this step, you will:

- Install Raspberry Pi OS using the Raspberry Pi Imager tool.
- Configure basic system settings such as Wi-Fi, keyboard layout, and time zone.
- Enable key interfaces including SPI, I2C, 1-Wire, and VNC using the Raspberry Pi Configuration menu.
- Identify the Raspberry Pi’s IP address using ifconfig and connect remotely via TigerVNC Viewer.
- Perform a full system update
- Install the lgpio library

Step 2: Connect the Servo Motor

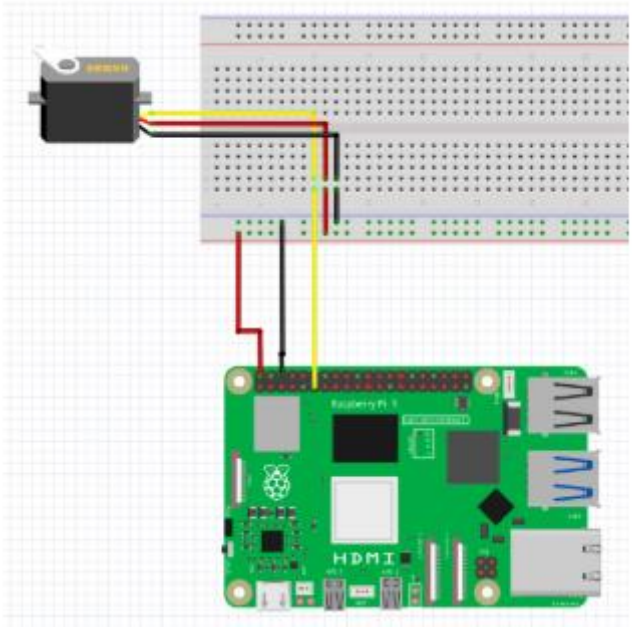
In this step, you will build a simple circuit to control a servo motor using the Raspberry Pi 5. The servo has three wires: power, ground, and signal.

To connect the servo:

- Connect the brown or black wire (GND) to physical pin 6 on the Raspberry Pi.
- Connect the red wire (5V) to physical pin 2 or pin 4.
- Connect the orange or yellow wire (signal) to GPIO 18 (physical pin 12).

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 Servo

Now that the servo is connected, you'll write a Python script to rotate it to multiple positions using the lgpio library. This library allows precise PWM (Pulse Width Modulation) control, which is required to operate servo motors.

Open Geany and enter the following code:

```
import lgpio
import time
# GPIO pin for the servo (e.g., GPIO 17)
SERVO_PIN = 17
# Initialize GPIO chip (the default GPIO chip is usually "/dev/gpiochip0")
h = lgpio.gpiochip_open(0)
# Claim the GPIO pin for PWM (Servo control)
lgpio.gpio_claim_output(h, SERVO_PIN)
# Function to generate PWM for servo motor
def set_servo_pulsewidth(h, pin, pulse_width):
    """Sends a high signal to the servo pin for the pulse width (in
milliseconds)."""
    lgpio.gpio_write(h, pin, 1)            # Set pin HIGH (start of pulse)
    time.sleep(pulse_width / 1000)         # Wait for pulse width duration
(pulse width is in milliseconds)
```

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.

```
    lgpio.gpio_write(h, pin, 0)             # Set pin LOW (end of pulse)
# Function to move the servo to a specified angle
def move_servo(angle):
    """
    Convert the angle (between 0 and 180 degrees) to the corresponding pulse
width
    (between 1ms and 2ms) and move the servo to that angle.
    """
    pulse_width = 1 + (angle / 180.0)       # Map angle to pulse width in
milliseconds (1ms to 2ms)
    set_servo_pulsewidth(h, SERVO_PIN, pulse_width)  # Generate PWM signal for
the calculated pulse width
# Main loop to control the servo
try:
    while True:
        # Move servo to the middle position (90 degrees)
        print("Moving to middle position")
        move_servo(90)                      # Move to 90 degrees
        time.sleep(0.5)                     # Short delay
        # Move servo to minimum position (0 degrees)
        print("Moving to minimum position")
        move_servo(0)                       # Move to 0 degrees
        time.sleep(1)                       # Longer delay
        # Move servo to middle position again (90 degrees)
        print("Moving to middle position")
        move_servo(90)                      # Move to 90 degrees again
        time.sleep(0.5)                     # Short delay
        # Move servo to maximum position (180 degrees)
        print("Moving to maximum position")
        move_servo(180)                     # Move to 180 degrees
        time.sleep(1)                       # Longer delay
except KeyboardInterrupt:
    # Handle the case when the user interrupts the program (Ctrl + C)
    print("Stopping servo control")
    time.sleep(1)
    lgpio.gpio_write(h, SERVO_PIN, 0)       # Turn off the SERVO_PIN before
exit
    lgpio.gpiochip_close(h)
```

Code Breakdown:

Setup and Imports:
The script starts by importing the necessary libraries: `lgpio` to control the GPIO, and `time` to manage delays.

Pin Definition and Initialization:
It sets up **GPIO 17** as the output pin for the servo signal. The GPIO chip is opened, and

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.

the pin is claimed for output so we can send precise on/off signals.

PWM Function:
A function called `set_servo_pulsewidth` manually sends a high signal (pulse) to the pin for a short period — the length of this pulse tells the servo what angle to rotate to. For servos, this usually ranges from **1ms (0°)** to **2ms (180°)**.

Angle Mapping:
Another function, `move_servo`, converts angles like 0°, 90°, and 180° into the correct pulse width. It uses a simple formula to calculate the duration based on the angle, then calls the PWM function.

Main Loop:
Inside the main loop, the servo moves between different angles (90 → 0 → 90 → 180) with short pauses in between. You can clearly see the movement with each step.

Interrupt Handling:
If you press **Ctrl + C** to stop the program, it safely shuts down the GPIO connection and turns the servo off, preventing jitter or sudden movement.

Step 4: Run the Program

Once your code is written and saved (e.g., servo_manual_pwm.py), you can run it using Geany or the terminal. The servo will begin cycling between the defined positions.

To run the program:

- Click the Run button in Geany (or press F5).
- Alternatively, in the terminal:

  python servo_manual_pwm.py

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.

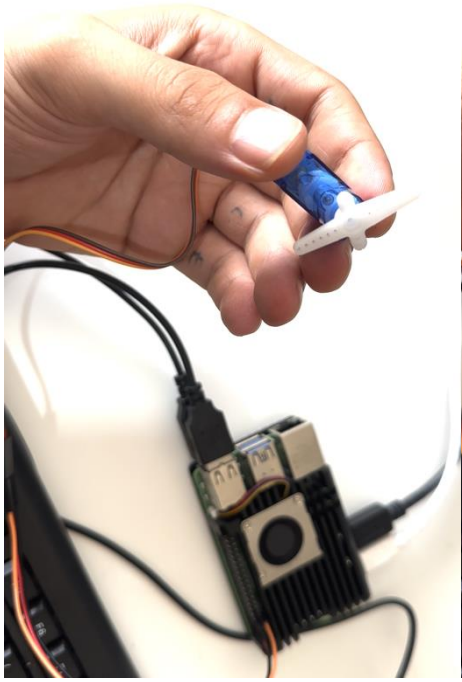


Figure 1- Servo on Middle Position.

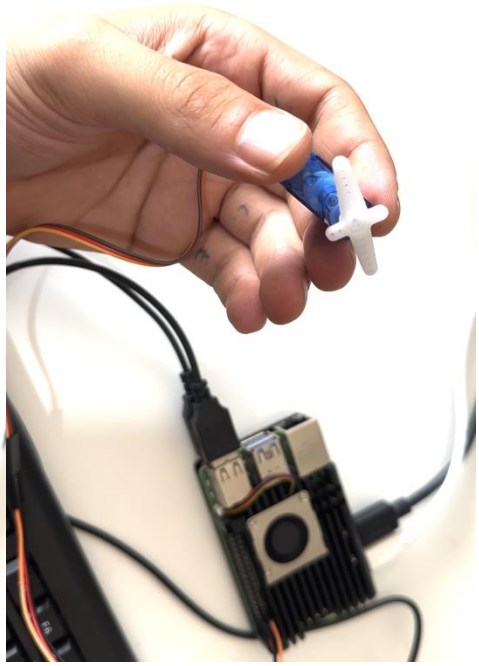


Figure 2- Servo on Maximum Position.

11. Evaluation Criteria:

- **Project Completion:** Servo moves between defined angles as expected.
- **Circuit Accuracy:** Wiring correctly supplies power and signal.
- **Code Execution:** Script runs without error using `lgpio`.
- **Understanding PWM:** Participant grasps the relationship between pulse width and motion.
- **Participant Feedback:** Reflection on how clearly movement control was achieved.

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. Languages Available:

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

---

13. Contributors:

Fátima Pais, Luís Soares – Project Designers and Facilitators

---

17. Feedback and Continuous Improvement:

- **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.