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: 20

13. Languages Available: 20

14. Contributors: 20

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:

Environmental Monitoring Station with Raspberry Pi and Multi-Sensor Integration

---

2. Project Category:

- AI in IoT

---

3. Project Description:

This project aims to develop a comprehensive environmental monitoring system using a Raspberry Pi with multiple sensors and an interactive OLED display. The system continuously monitors temperature, humidity, and ambient light levels, providing real-time environmental awareness through an intuitive button-controlled interface. Students will learn to integrate multiple sensors, create user interfaces, and develop systems that promote energy efficiency and environmental consciousness. The system features automatic comfort level assessments, energy-saving recommendations, and robust error handling for reliable operation.

---

4. Keywords:

Environmental Monitoring, Raspberry Pi, IoT, Multi-Sensor Integration, OLED Display, Temperature Monitoring, Humidity Control, Light Detection, Energy Efficiency, Sustainability

---

5. Target Audience:

Both Educators and Students

---

6. Pre-Requisite knowledge:

- Basic understanding of Ohm's Law and electrical circuits

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 Python programming (variables, loops, functions, classes)
- Knowledge of Raspberry Pi GPIO control and I2C communication
- Basic experience with sensor integration and digital/analog signals
- Understanding of pull-up resistors and button debouncing

---

7. Project Objectives:

- Introduce students to multi-sensor IoT system integration
- Teach real-time environmental data acquisition and interpretation
- Develop skills in user interface design with OLED displays
- Raise awareness about indoor air quality and energy efficiency
- Build hands-on experience with I2C communication protocols
- Encourage creation of sustainable monitoring solutions
- Foster understanding of human-computer interaction in embedded systems
- Promote environmental consciousness through technology

---

8. Project Timeline:

---

9. Resources Provided:

- Raspberry Pi 5
- DHT11 Temperature and Humidity Sensor
- LDR (Light Dependent Resistor)
- SSD1306 OLED Display (128x64, I2C)
- 1x Push button (momentary switch)
- 1x 330Ω resistor (for button pull-up)
- Breadboard and jumper wires
- Python code templates
- Comprehensive step-by-step instructional guide
- Pre-configured Python environment setup scripts
- Complete source code with detailed comments
- Installation requirements file
- Troubleshooting guide
- Configuration examples

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.

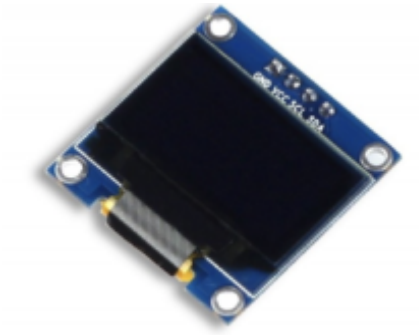
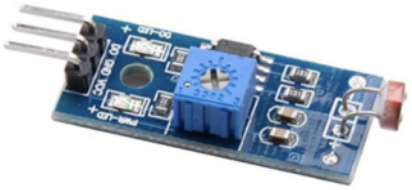
10. Tools and Technologies Used:

10.1 Hardware:

MATERIAL	
Raspberry Pi 5	
27W USB Type-C Power Supply for Raspberry Pi 5	
Flat Jumper Wires F/F	
Breadboard	
SINGLE-FILE WIRE FOR BREADBOARDS	
Resistor 330 Ohm 1/4 W	
DHT11 Temperature-Humidity Sensor (IM14)	

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.

0.95inch OLED i2c Display Blue/Yellow SSD1306	
Tactile Button Switch Kit	
LDR Sensor Module with Potentiometer	

10.2 Software:

- **Operating System:** Raspberry Pi OS
- **Programming Language:** Python 3.7+
- **Programming Environment:** Geany, Thonny, or Python IDE

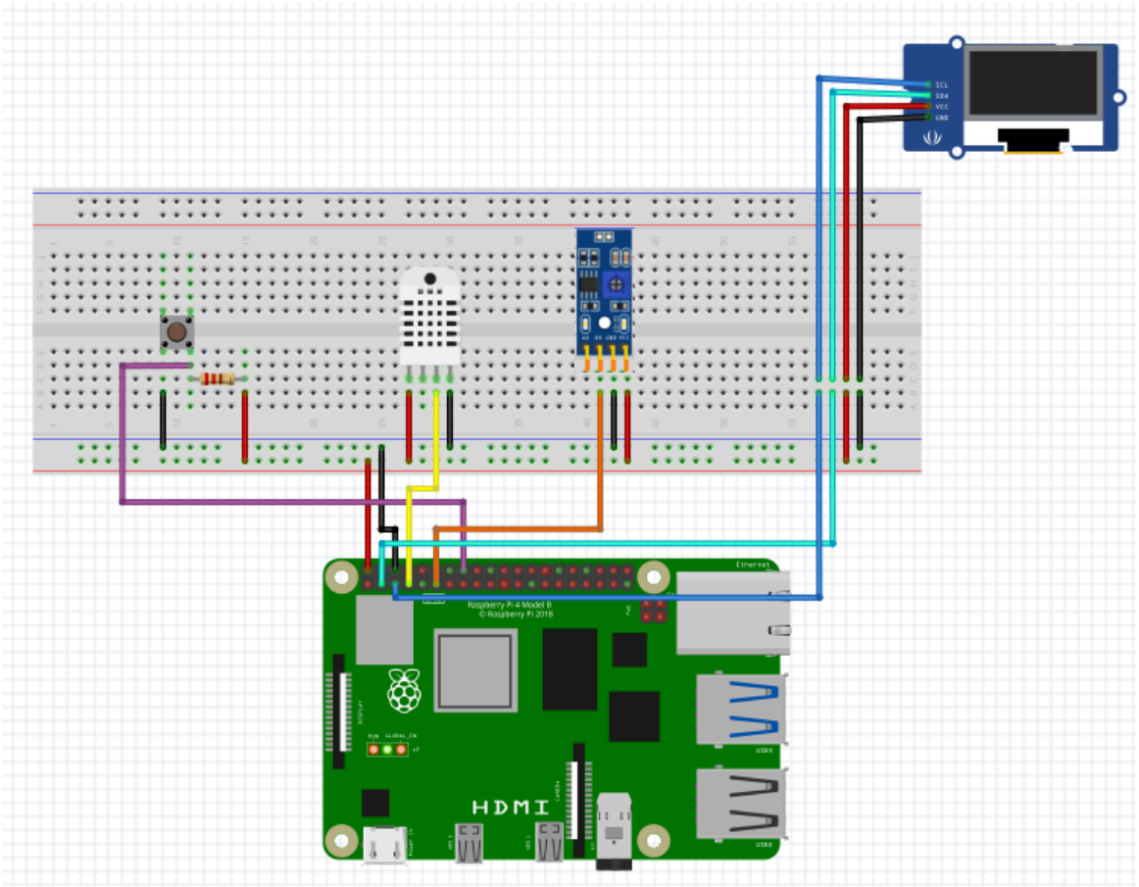
Libraries: RPi.GPIO, time, PIL, Adafruit_DHT, Adafruit_ssd1306

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



Circuit Connections

DHT11 Temperature-Humidity Sensor:

- VCC ⟶ 3.3V (Pin 1) or 5V (Pin 2)
- GND ⟶ Ground (Pin 6)
- DATA ⟶ GPIO 4 (Pin 7)

SSD1306 OLED Display (I2C):

- VCC ⟶ 3.3V (Pin 1)
- GND ⟶ Ground (Pin 6)
- SDA ⟶ GPIO 2/SDA (Pin 3)
- SCL ⟶ GPIO 3/SCL (Pin 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.

LDR Light Sensor:

- One leg ⟶ GPIO 17 (Pin 11)
- Other leg ⟶ Ground through pull-down configuration

Push Button:

- One terminal ⟶ GPIO 23 (Pin 16)
- Other terminal ⟶ Ground (Pin 6)
- Built-in pull-up resistor used in software

Common Ground Connections:

- Use Pin 6 (Ground) as common ground for all components
- Ensure stable power distribution

Wiring Steps

1. **Enable I2C Interface**
2. **Connect DHT11 sensor:** Power, ground, and data to GPIO 4
3. **Connect OLED display:** I2C connections (SDA, SCL) and power
4. **Install LDR sensor:** Digital input configuration on GPIO 17
5. **Add push button:** GPIO 23 with internal pull-up resistor
6. **Verify all connections:** Double-check wiring before power-on

⚠️ Important: Before powering on the system, carefully verify all connections. Incorrect wiring can damage components or cause system instability.

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.

11.2 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 environmental_monitor
cd environmental_monitor

# 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 environmental monitor Python code to `main.py`
- Make sure the file is in your project directory

4. Run the System

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

# Run the light environmental monitor
python main.py
```

5. Stop the System

- Press Ctrl+C to stop the program safely
- All components will be properly cleaned up automatically

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:

Startup Sequence

1. **Display Initialization:** OLED screen displays "ENV STATION" startup message
2. **Sensor Calibration:** System performs initial sensor readings
3. **Mode Selection:** System starts in "All Data" mode by default
4. **Status Indication:** Startup message shown for 2 seconds before normal operation

Operational States

Normal Operation:

- **Continuous Monitoring:** Sensors read every 0.5 seconds
- **Real-time Display Updates:** OLED screen refreshes with current data
- **Button Responsiveness:** Mode switching available at any time
- **Error Recovery:** Automatic sensor error detection and recovery

Sensor Status Indicators:

- **Temperature/Humidity Available:** Displays actual values with comfort assessments
- **Sensor Error State:** Shows "Sensor Error" message when DHT11 fails to read
- **Light Detection Active:** Real-time ambient light level monitoring
- **System Stability:** Continues operation even with individual sensor failures

Interactive Features

Button Operation:

- **Single Press:** Cycles through display modes in sequence
- **Debounce Protection:** 300ms delay prevents accidental multiple presses
- **Visual Feedback:** Immediate display mode change confirmation
- **Mode Persistence:** Current mode maintained until button pressed

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.

Display Mode Cycling:

1. **All Data Mode:** Overview screen showing T, H, L values
2. **Temperature Mode:** Large temperature display with comfort level
3. **Humidity Mode:** Humidity percentage with optimal range indicator
4. **Light Mode:** Light level with energy-saving recommendations
5. **Returns to All Data Mode:** Completes the cycle

Environmental Assessments

Temperature Comfort Levels:

- **Cold:** < 18°C - Consider heating
- **Comfortable:** 18°C - 26°C - Optimal range
- **Hot:** > 26°C - Consider cooling

Humidity Levels:

- **Dry:** < 30% - May need humidifier
- **Optimal:** 30% - 70% - Ideal range
- **Humid:** > 70% - Consider dehumidification

Light Detection:

- **Bright:** Adequate natural light detected
- **Dark:** Insufficient light, artificial lighting recommended

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

```
import time
import board
import adafruit_dht
import adafruit_ssd1306
import lgpio
from PIL import Image, ImageDraw, ImageFont

# Pin definitions
DHT_PIN = board.D4          # DHT11 temperature and humidity sensor
LDR_PIN = 17                # Light sensor
BUTTON_MODE = 23            # Single button to cycle display modes

# OLED display dimensions
WIDTH = 128
HEIGHT = 64

# Initialize sensors and display
dht = adafruit_dht.DHT11(DHT_PIN)
i2c = board.I2C()
oled = adafruit_ssd1306.SSD1306_I2C(WIDTH, HEIGHT, i2c, addr=0x3C)

# Initialize GPIO
h = lgpio.gpiochip_open(0)

# Set up GPIO pins
lgpio.gpio_claim_input(h, LDR_PIN)
lgpio.gpio_claim_input(h, BUTTON_MODE, lgpio.SET_PULL_UP)

# Display modes
MODE_TEMP = 0
MODE_HUMIDITY = 1
MODE_LIGHT = 2
MODE_ALL = 3

class EnvironmentalStation:
    def __init__(self):
        self.current_mode = MODE_ALL
        self.temperature = None
        self.humidity = None
        self.light_level = "Unknown"
        self.last_button_time = 0
        self.button_debounce = 0.3  # 300ms debounce
```

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.

```
        # Clear display
        oled.fill(0)
        oled.show()

    def read_sensors(self):
        """Read all sensor values"""
        # Read temperature and humidity
        try:
            self.temperature = dht.temperature
            self.humidity = dht.humidity
        except RuntimeError as e:
            print(f"DHT sensor error: {e.args[0]}")

        # Read light sensor
        light_reading = lgpio.gpio_read(h, LDR_PIN)

        # Determine light level based on sensor reading
        if light_reading == 0:
            self.light_level = "Bright"
        else:
            self.light_level = "Dark"

    def check_buttons(self):
        """Check button press with debouncing"""
        current_time = time.time()

        if current_time - self.last_button_time < self.button_debounce:
            return

        if lgpio.gpio_read(h, BUTTON_MODE) == 0:
            # Cycle through modes: ALL -> TEMP -> HUMIDITY -> LIGHT -> ALL
            self.current_mode = (self.current_mode + 1) % 4
            self.last_button_time = current_time

            mode_names = ["All Data", "Temperature", "Humidity", "Light"]
            print(f"{mode_names[self.current_mode]} mode selected")

    def get_font_size(self, font, text):
        """Calculate text dimensions"""
        left, top, right, bottom = font.getbbox(text)
        return right - left, bottom - top
```

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 draw_centered_text(self, draw, text, font, y_offset=0):
        """Draw text centered horizontally"""
        font_width, font_height = self.get_font_size(font, text)
        x = (WIDTH - font_width) // 2
        y = (HEIGHT - font_height) // 2 + y_offset
        draw.text((x, y), text, font=font, fill=255)

    def display_temperature(self):
        """Display temperature screen"""
        image = Image.new("1", (WIDTH, HEIGHT))
        draw = ImageDraw.Draw(image)
        font = ImageFont.load_default()

        # Title
        self.draw_centered_text(draw, "TEMPERATURE", font, -20)

        # Temperature value
        if self.temperature is not None:
            temp_text = f"{self.temperature}°C"
            # Check comfort level
            if self.temperature < 18:
                comfort = "Cold"
            elif self.temperature > 26:
                comfort = "Hot"
            else:
                comfort = "Comfortable"

            self.draw_centered_text(draw, temp_text, font, 0)
            self.draw_centered_text(draw, comfort, font, 15)
        else:
            self.draw_centered_text(draw, "Sensor Error", font, 0)

        oled.image(image)
        oled.show()

    def display_humidity(self):
        """Display humidity screen"""
        image = Image.new("1", (WIDTH, HEIGHT))
        draw = ImageDraw.Draw(image)
        font = ImageFont.load_default()

        # Title
        self.draw_centered_text(draw, "HUMIDITY", font, -20)
```

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.

```
        # Humidity value
        if self.humidity is not None:
            humidity_text = f"{self.humidity}%"

            # Check humidity level
            if self.humidity < 30:
                level = "Dry"
            elif self.humidity > 70:
                level = "Humid"
            else:
                level = "Optimal"

            self.draw_centered_text(draw, humidity_text, font, 0)
            self.draw_centered_text(draw, level, font, 15)
        else:
            self.draw_centered_text(draw, "Sensor Error", font, 0)

        oled.image(image)
        oled.show()

    def display_light(self):
        """Display light information screen"""
        image = Image.new("1", (WIDTH, HEIGHT))
        draw = ImageDraw.Draw(image)
        font = ImageFont.load_default()

        # Title
        self.draw_centered_text(draw, "LIGHT LEVEL", font, -20)

        # Light level
        self.draw_centered_text(draw, self.light_level, font, 0)

        # Energy saving tip
        if "Dark" in self.light_level:
            tip = "Turn on lights"
        elif "Bright" in self.light_level:
            tip = "Good lighting"
        else:
            tip = "Adequate lighting"

        self.draw_centered_text(draw, tip, font, 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.

```
        oled.image(image)
        oled.show()

    def display_all_data(self):
        """Display all sensor data on one screen"""
        image = Image.new("1", (WIDTH, HEIGHT))
        draw = ImageDraw.Draw(image)
        font = ImageFont.load_default()

        # Title
        draw.text((35, 2), "ENV MONITOR", font=font, fill=255)

        # Temperature
        if self.temperature is not None:
            temp_text = f"T: {self.temperature}°C"
        else:
            temp_text = "T: Error"
        draw.text((5, 18), temp_text, font=font, fill=255)

        # Humidity
        if self.humidity is not None:
            hum_text = f"H: {self.humidity}%"
        else:
            hum_text = "H: Error"
        draw.text((5, 32), hum_text, font=font, fill=255)

        # Light
        light_text = f"L: {self.light_level}"
        if len(light_text) > 16:  # Truncate if too long
            light_text = light_text[:13] + "..."
        draw.text((5, 46), light_text, font=font, fill=255)

        # Instructions
        # draw.text((5, 58), "Press button to cycle", font=font, fill=255)

        oled.image(image)
        oled.show()

    def update_display(self):
        """Update display based on current mode"""
        if self.current_mode == MODE_TEMP:
            self.display_temperature()
        elif self.current_mode == MODE_HUMIDITY:
```

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.

```
            self.display_humidity()
        elif self.current_mode == MODE_LIGHT:
            self.display_light()
        else:  # MODE_ALL
            self.display_all_data()

    def run(self):
        """Main monitoring loop"""
        print("Environmental Monitoring Station Started")
        print("Press button to cycle through display modes:")
        print("- All Data -> Temperature -> Humidity -> Light -> All Data")

        # Show startup message
        image = Image.new("1", (WIDTH, HEIGHT))
        draw = ImageDraw.Draw(image)
        font = ImageFont.load_default()
        self.draw_centered_text(draw, "ENV STATION", font, -10)
        self.draw_centered_text(draw, "Starting...", font, 10)
        oled.image(image)
        oled.show()
        time.sleep(2)

        while True:
            try:
                # Read all sensors
                self.read_sensors()

                # Check for button presses
                self.check_buttons()

                # Update display
                self.update_display()

                # Brief delay before next update
                time.sleep(0.5)

            except KeyboardInterrupt:
                print("\nShutting down Environmental Station...")
                break
            except Exception as e:
                print(f"Error: {e}")
                time.sleep(1)
```

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.

```
# Create and run the environmental station
if __name__ == "__main__":
    station = EnvironmentalStation()

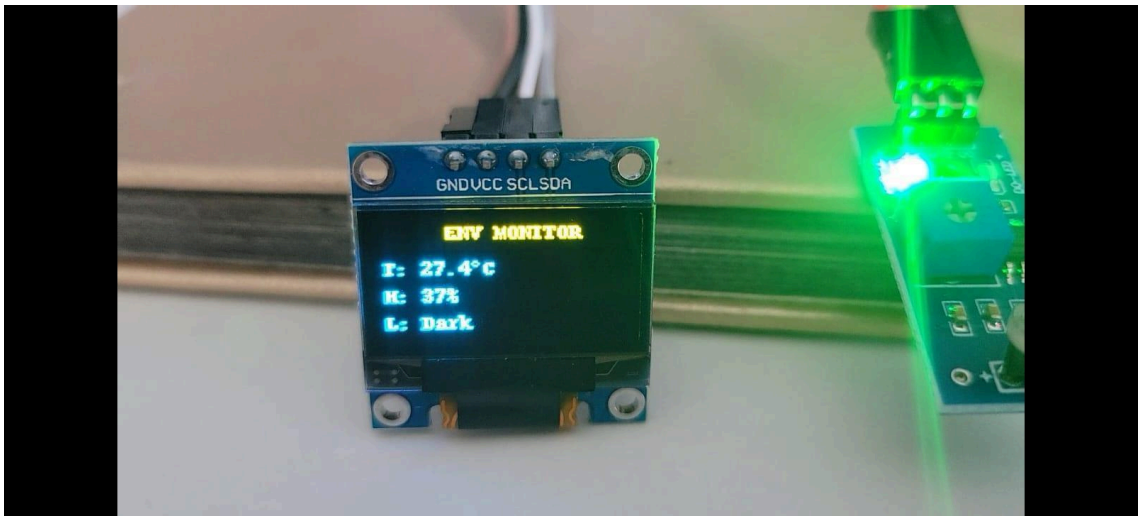
    try:
        station.run()
    except KeyboardInterrupt:
        print("\nProgram interrupted by user")
    finally:
        # Cleanup
        print("Cleaning up GPIO...")
        oled.fill(0)  # Clear display
        oled.show()
        lgpio.gpiochip_close(h)  # Close GPIO
        print("Environmental Station shutdown complete.")
```

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.

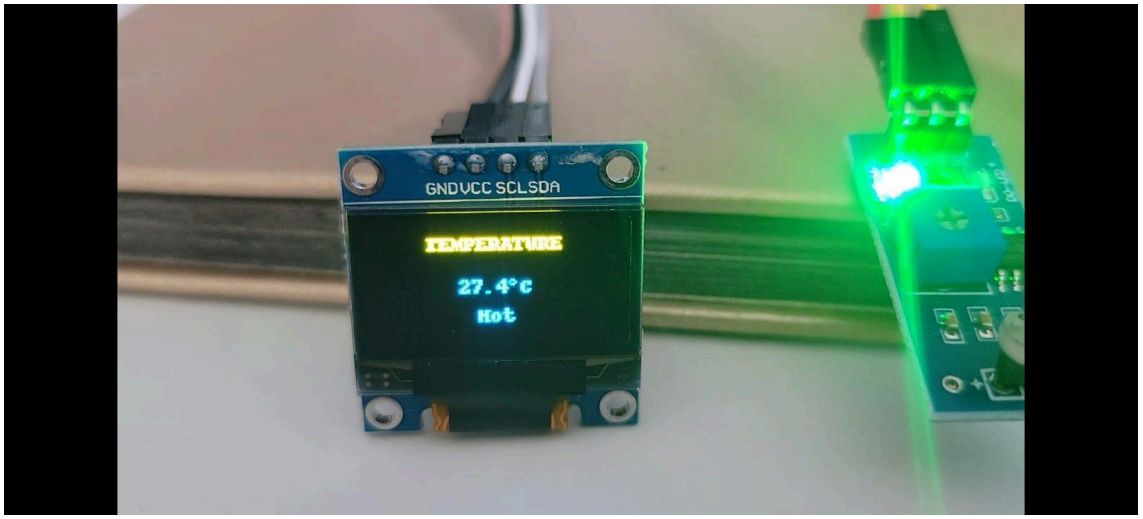
Now, you can run the program and see the result:

```
(env) pi@raspberrypi-2:~ $ python enviroment_station.py
Environmental Monitoring Station Started
Press button to cycle through display modes:
- All Data -> Temperature -> Humidity -> Light -> All Data
All Data mode selected
Temperature mode selected
Humidity mode selected
Light mode selected
All Data mode selected
```

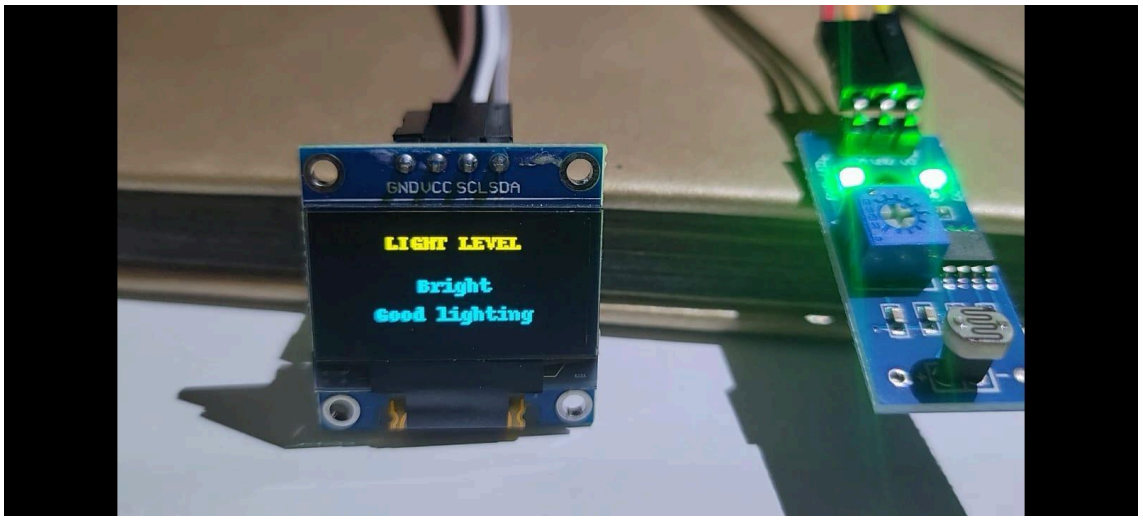


Co-funded by the European Union

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.







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:

- **Hardware Setup (30%)**: Correct connection of Raspberry Pi, DHT11, OLED display, LDR sensor, and button; stable wiring with no loose connections; proper I2C configuration and power distribution.
- **Code Functionality (25%)**: Code runs without errors; sensors are read in real time; OLED updates correctly; button cycles display modes as expected; system handles sensor failures gracefully.
- **System Understanding (20%)**: Clear explanation of how each sensor works, how data is collected and interpreted, and how the button and OLED interface function within the monitoring system.
- **Environmental Insight & Enhancement (15%)**: Awareness of energy efficiency and environmental monitoring; suggestions for improvement, e.g., remote data logging, AI-based alerts, or integration with home automation.
- **User Interface & Interaction (10%)**: OLED display is informative and easy to understand; display modes are distinct and useful; button interaction is intuitive and debounced correctly.

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.