AI & SUSTAINABILITY IN VET EDUCATION

Nome dell'autore/i: Sucessos Criativos, Lda | P. Porto

Titolo del WP: WP3 – AI-SAVES@ProjectHub

Riferimento dell'attività: 3.2 – Modello di progetto per 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.

Indice dei contenuti

1. Titolo del progetto: 2

2. Categoria del progetto: 2

3. Descrizione del progetto: 2

5. Pubblico target: 2

6. Conoscenze preliminari: 2

7. Obiettivi del progetto: 3

8. Tempistica del progetto: 3

9. Risorse fornite: 3

10. Strumenti e tecnologie utilizzati: 4

11. Piano di implementazione passo dopo passo: 6

12. Criteri di valutazione: 13

13. Lingue disponibili: 13

14. Collaboratori: 13

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. **Titolo del progetto:**

Stazione di monitoraggio ambientale con Raspberry Pi e integrazione multi-sensore

---

2. **Categoria del progetto:**
   - IA nell'IoT

---

3. **Descrizione del progetto:**

Questo progetto mira a sviluppare un sistema completo di monitoraggio ambientale utilizzando un Raspberry Pi con più sensori e un display OLED interattivo. Il sistema monitora continuamente temperatura, umidità e livelli di luce ambientale, fornendo consapevolezza ambientale in tempo reale tramite un'interfaccia intuitiva controllata da pulsanti. Gli studenti impareranno a integrare più sensori, creare interfacce utente e sviluppare sistemi che promuovono l'efficienza energetica e la consapevolezza ambientale. Il sistema include valutazioni automatiche dei livelli di comfort, raccomandazioni per il risparmio energetico e gestione robusta degli errori per un funzionamento affidabile.

---

4. **Parole chiave:**
   Monitoraggio ambientale, Raspberry Pi, IoT, Integrazione multi-sensore, Display OLED, Monitoraggio temperatura, Controllo umidità, Rilevamento luce, Efficienza energetica, Sostenibilità

---

5. **Pubblico target:**
   Sia docenti che studenti

---

6. **Conoscenze preliminari:**
   - Comprensione di base della legge di Ohm e dei circuiti elettrici
   - Introduzione alla programmazione Python (variabili, cicli, funzioni)
   - Conoscenza del controllo GPIO del Raspberry Pi e comunicazione I2C
   - Esperienza di base con l'integrazione dei sensori e segnali digitali/analogici
   - Comprensione delle resistenze pull-up e del debouncing dei pulsanti

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.

7. **Obiettivi del progetto:**
   - Introdurre gli studenti all'integrazione di sistemi IoT multi-sensore
   - Insegnare l'acquisizione e l'interpretazione dei dati ambientali in tempo reale
   - Sviluppare competenze nella progettazione di interfacce utente con display OLED
   - Sensibilizzare sulla qualità dell'aria interna e sull'efficienza energetica
   - Fornire esperienza pratica con protocolli di comunicazione I2C
   - Incoraggiare la creazione di soluzioni di monitoraggio sostenibili
   - Promuovere la comprensione dell'interazione uomo-computer nei sistemi embedded
   - Favorire la consapevolezza ambientale tramite la tecnologia

8. **Tempistica del progetto:**
   Tempistica flessibile in base al calendario della classe o del workshop

9. **Risorse fornite:**
   - Raspberry Pi 5
   - Sensore di temperatura e umidità DHT11
   - Sensore LDR (Light Dependent Resistor)
   - Display OLED SSD1306 (128x64, I2C)
   - 1x pulsante (interruttore momentaneo)
   - 1x resistenza da 330Ω (per pull-up del pulsante)
   - Breadboard e cavi jumper
   - Modelli di codice Python
   - Guida didattica completa passo-passo
   - Script per la configurazione dell'ambiente Python preconfigurato
   - Codice sorgente completo con commenti dettagliati
   - File dei requisiti per l'installazione
   - Guida alla risoluzione dei problemi
   - Esempi di configurazione

10. **Strumenti e tecnologie utilizzati:**

Co-funded by the European Union

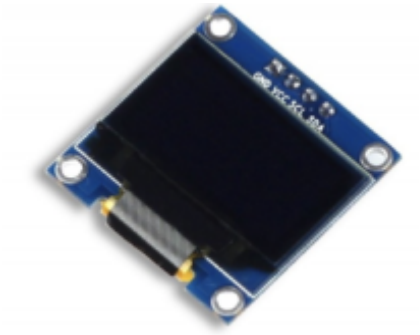
This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

10.1 Hardware:

MATERIALE	
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	

10.2 Software:

- **Sistema operativo**: Raspberry Pi OS
- **Linguaggio di programmazione**: Python 3.7+
- **Ambiente di programmazione**: Geany, Thonny o IDE Python

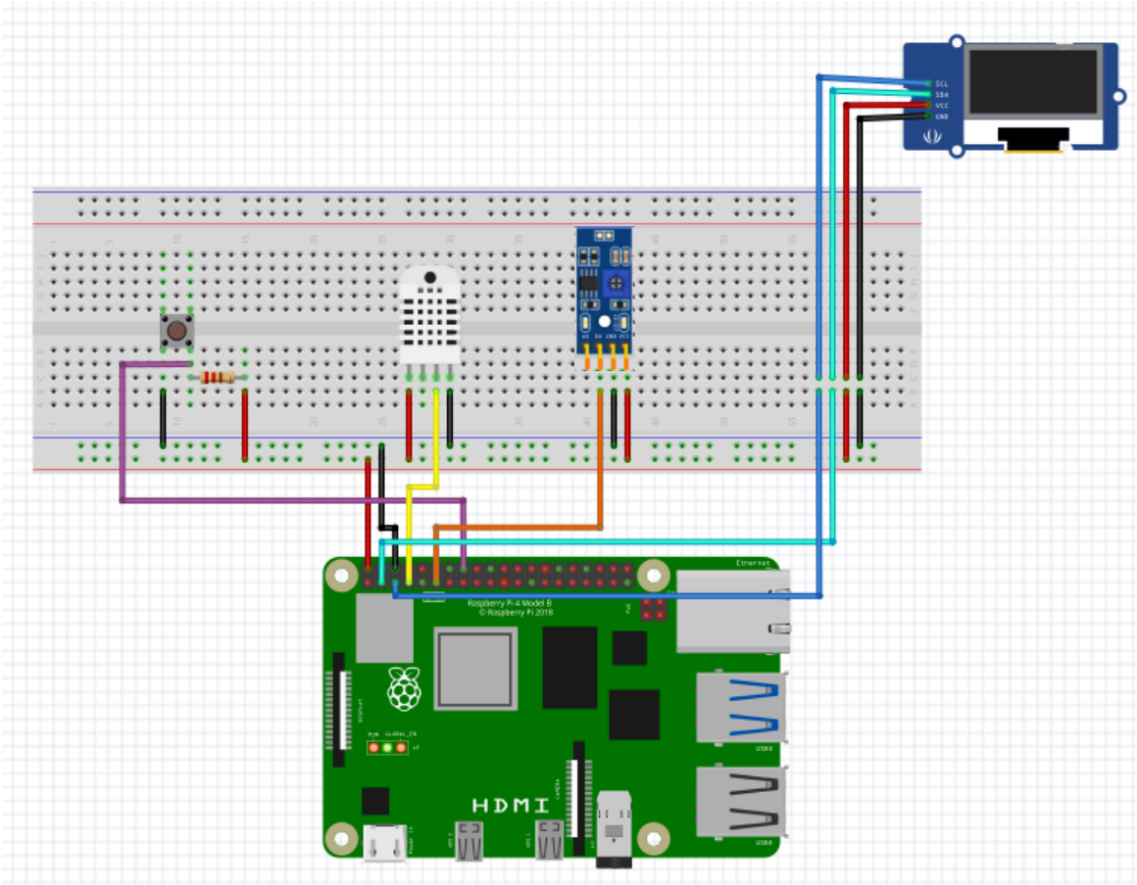
Librerie: Adafruit_DHT, RPi.GPIO

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. Piano di implementazione passo dopo passo:

11.1 Configurazione hardware



Collegamenti del circuito

Sensore di temperatura e umidità DHT11:

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

SSD1306 OLED Display (I2C):

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

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.

- SCL → GPIO 3/SCL (Pin 5)

Sensore di Luce LDR:

- Un terminale → GPIO 17 (Pin 11)
- Altro terminale → Massa tramite configurazione **pull-down**

Pulsante:

- Un terminale → GPIO 23 (Pin 16)
- Altro terminale → Massa (Pin 6)
- Resistenza **pull-up interna** utilizzata via software

Connessioni comuni di massa

- Utilizzare il **Pin 6 (GND)** come massa comune per tutti i componenti
- Assicurarsi di avere una distribuzione stabile dell'alimentazione

Passaggi di cablaggio:

1. Abilitare l'interfaccia **I2C**
2. Collegare il sensore DHT11: alimentazione, massa e pin dati su **GPIO 4**
3. Collegare il display OLED: connessioni I2C (**SDA, SCL**) e alimentazione
4. Installare il sensore LDR: configurazione ingresso digitale su **GPIO 17**
5. Aggiungere il pulsante: **GPIO 23** con resistenza pull-up interna
6. Verificare tutti i collegamenti: controllare attentamente il cablaggio prima dell'accensione

⚠ **Importante:** Prima di alimentare il sistema, verifica attentamente tutti i collegamenti. Un cablaggio errato può danneggiare i componenti o causare instabilità del sistema..

11.2 Configurazione del software

Ora che l'hardware è pronto, è il momento di implementare il sistema di controllo per il tracciamento della luce.

1. Creare un ambiente virtuale

```
# Create project directory
mkdir environmental_monitor
```

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.

```
cd environmental_monitor

# Create virtual environment
python3 -m venv venv

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

2. Installare le dipendenze

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

3. Aggiungere il codice

- Copia il codice Python del monitor ambientale in `main.py`
- Assicurati che il file si trovi nella directory del tuo progetto

4. Avviare il sistema

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

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

5. **Arrestare il sistema**
   - Premi **Ctrl+C** per fermare il programma in modo sicuro
   - Tutti i componenti verranno ripristinati automaticamente in modo corretto.

Comportamento del sistema

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.

Sequenza di avvio

1. **Inizializzazione display:** Il display OLED mostra il messaggio di avvio "ENV STATION"
2. **Calibrazione sensori:** Il sistema esegue le letture iniziali dei sensori
3. **Selezione modalità:** Il sistema parte di default in modalità "Tutti i dati"
4. **Indicazione stato:** Il messaggio di avvio viene mostrato per 2 secondi prima del funzionamento normale

Stati operativi

Funzionamento normale:

- **Monitoraggio continuo:** I sensori vengono letti ogni 0,5 secondi
- **Aggiornamenti in tempo reale del display:** Lo schermo OLED si aggiorna con i dati correnti
- **Reattività del pulsante:** Possibilità di cambiare modalità in qualsiasi momento
- **Recupero errori:** Rilevamento automatico degli errori dei sensori e ripristino

Indicatori di stato dei sensori:

- **Temperatura/Umidità disponibile:** Mostra i valori effettivi con valutazioni del comfort
- **Stato di errore del sensore:** Mostra il messaggio "Sensor Error" quando il DHT11 non riesce a leggere
- **Rilevamento luce attivo:** Monitoraggio in tempo reale dei livelli di luce ambientale
- **Stabilità del sistema:** Continua a funzionare anche in caso di guasti di singoli sensori

Funzionalità interattive

Operazioni del pulsante:

- **Pressione singola:** Cicla attraverso le modalità di visualizzazione in sequenza
- **Protezione da rimbalzo (debounce):** Ritardo di 300 ms per evitare pressioni multiple accidentali
- **Feedback visivo:** Conferma immediata del cambio di modalità sul display
- **Persistenza della modalità:** La modalità corrente rimane attiva fino alla successiva pressione

Ciclo delle modalità di visualizzazione:

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. **Modalità Tutti i dati:** Schermata panoramica con valori T, H, L
2. **Modalità Temperatura:** Visualizzazione della temperatura in grande con livello di comfort
3. **Modalità Umidità:** Percentuale di umidità con indicatore del range ottimale
4. **Modalità Luce:** Livello di luce con consigli per il risparmio energetico
5. **Ritorno alla Modalità Tutti i dati:** Completa il ciclo

Valutazioni ambientali

Livelli di comfort della temperatura:

- **Freddo:** < 18°C – Valutare il riscaldamento
- **Confortevole:** 18°C – 26°C – Intervallo ottimale
- **Caldo:** > 26°C – Valutare il raffreddamento

Livelli di umidità:

- **Secco:** < 30% – Potrebbe essere necessario un umidificatore
- **Ottimale:** 30% – 70% – Intervallo ideale
- **Umido:** > 70% – Valutare la deumidificazione

Rilevamento della luce:

- **Luminoso:** Luce naturale adeguata rilevata
- **Scuro:** Luce insufficiente, si consiglia illuminazione artificiale

CODICE

```
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)
```

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.

```
# 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

        # 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"""
```

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.

```
        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

    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"
```

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.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)

        # 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()
```

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.

```
        # 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)

        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}"
```

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.

```
        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:
            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()
```

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.

```
                # 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)

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

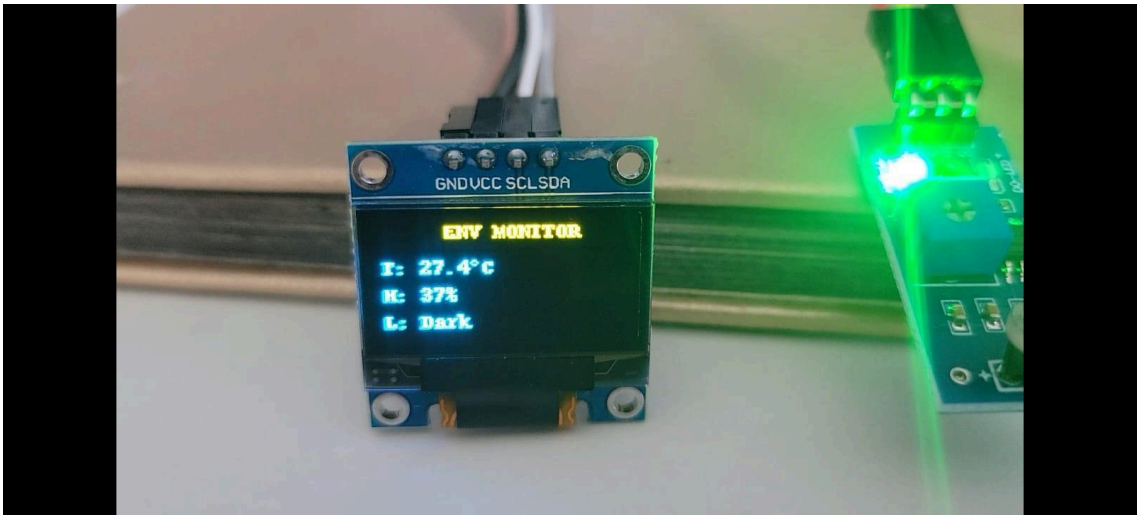
    try:
        station.run()
    except KeyboardInterrupt:
        print("\nProgram interrupted by user")
    finally:
        # Cleanup
        print("Cleaning up GPIO...")
        oled.fill(0)  # Clear display
        oled.show()
        lgpio.gpiochip_close(h)  # Close GPIO
        print("Environmental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

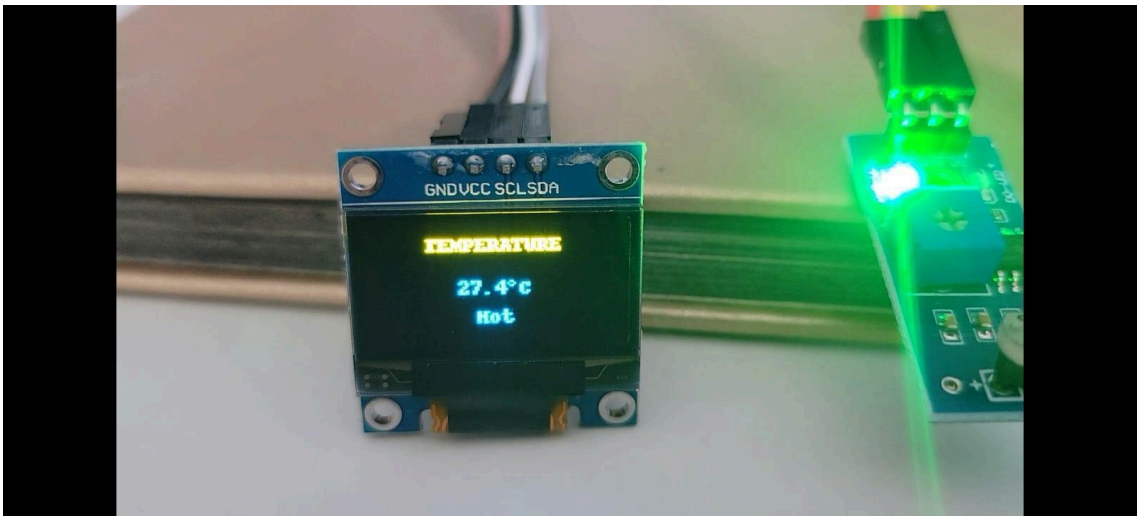
Ora puoi eseguire il programma e vedere il risultato:

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.

```
(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
```

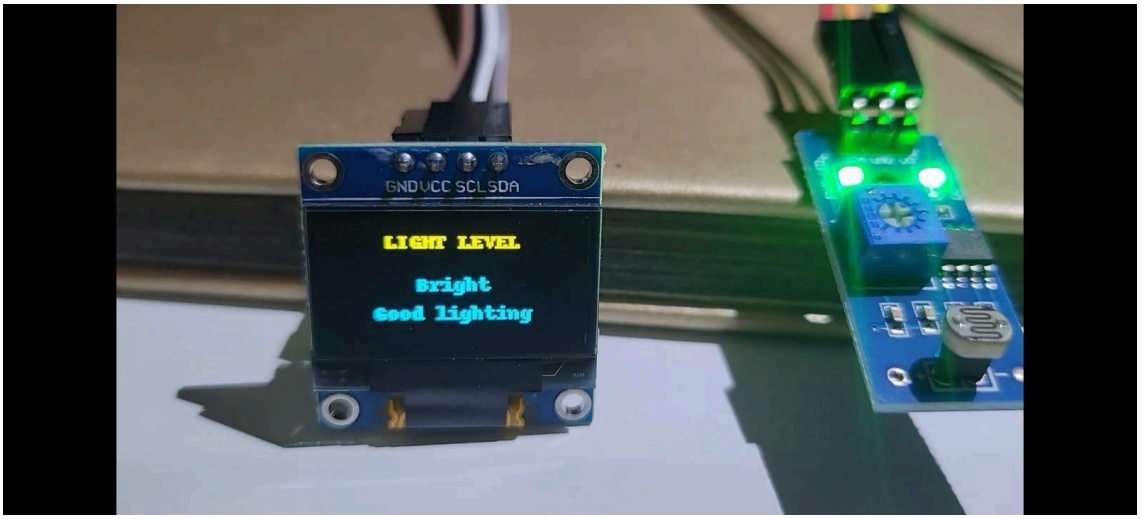




Co-funded by the European Union

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.





12. Criteri di valutazione:

- **Configurazione hardware (30%)**: Collegamento corretto di Raspberry Pi, DHT11, display OLED, sensore LDR e pulsante; cablaggio stabile senza connessioni allentate; corretta configurazione I2C e distribuzione dell'alimentazione.
- **Funzionalità del codice (25%)**: Il codice viene eseguito senza errori; i sensori vengono letti in tempo reale; l'OLED si aggiorna correttamente; il pulsante cicla tra le modalità di visualizzazione come previsto; il sistema gestisce eventuali guasti dei sensori in modo appropriato.
- **Comprensione del sistema (20%)**: Spiegazione chiara del funzionamento di ciascun sensore, di come i dati vengono raccolti e interpretati, e di come il pulsante e l'interfaccia OLED operano all'interno del sistema di monitoraggio.

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.

- **Consapevolezza ambientale e miglioramenti (15%)**: Sensibilità all'efficienza energetica e al monitoraggio ambientale; suggerimenti per miglioramenti, ad esempio registrazione remota dei dati, avvisi basati su IA o integrazione con domotica.
- **Interfaccia utente e interazione (10%)**: Il display OLED è informativo e facile da comprendere; le modalità di visualizzazione sono distinte e utili; l'interazione tramite pulsante è intuitiva e con corretta gestione del debounce.

---

13. Lingue disponibili:

---

14. Collaboratori:

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.