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:**

Sistema di rilevamento incendi con Raspberry Pi e sensore di temperatura

---

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

---

3. **Descrizione del progetto:**

   Questo progetto mira a sviluppare un sistema di rilevamento incendi di base con allerta precoce utilizzando un Raspberry Pi e un sensore di temperatura. Il sistema monitora in tempo reale le variazioni di temperatura e attiva indicatori LED per indicare i livelli di rischio: verde (normale), giallo (attenzione) e rosso (critico). Questa soluzione economica e scalabile contribuisce alla protezione ambientale rilevando precocemente potenziali rischi di incendio, sia in ambienti forestali che in spazi interni.

---

4. **Parole chiave:**
   Rilevamento incendi, Raspberry Pi, IoT, Monitoraggio della temperatura, Protezione ambientale

---

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
   - Esperienza di base con PWM (Modulazione di larghezza di impulso) e servomotori

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 ai concetti base dell'IoT e all'integrazione dei sensori
   - Insegnare l'acquisizione e l'interpretazione dei dati in tempo reale
   - Sensibilizzare alla prevenzione degli incendi e alla sostenibilità ambientale
   - Sviluppare competenze pratiche in elettronica e programmazione Python
   - Incoraggiare la creazione di soluzioni a basso costo e scalabili per problemi reali

8. **Tempistica del progetto:**

9. **Risorse fornite:**
   - Raspberry Pi 5
   - Sensore di temperatura (ad es. DHT11 o DS18B20)
   - 3 LED (verde, giallo, rosso) e 3 resistori
   - Breadboard e cavi jumper
   - Modelli di codice Python
   - Guida didattica passo-passo

10. **Strumenti e tecnologie utilizzati:**

Co-funded by the European Union

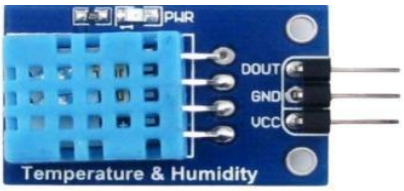
This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

10.1 Hardware:

MATERIALE	
Raspberry Pi 5	
27W USB Type-C Power Supply for Raspberry Pi 5	
Flat Jumper Wires F/F	
Breadboard	
Cavo singolo per breadboard	
LED diffuso da 5 mm rosso	
LED diffuso da 5 mm giallo	

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.

LED diffuso da 5 mm verde	
Resistor 330 Ohm 1/4 W	
Sensore di temperatura e umidità DHT11 (IM14)	

10.2 Software:

- Raspbian OS, Python 3
- Sistema operativo: Raspberry Pi OS
- Ambiente di programmazione: Geany o IDE Python
- Librerie Python: lgpio, time

Librerie: Adafruit_DHT, RPi.GPIO, board, time

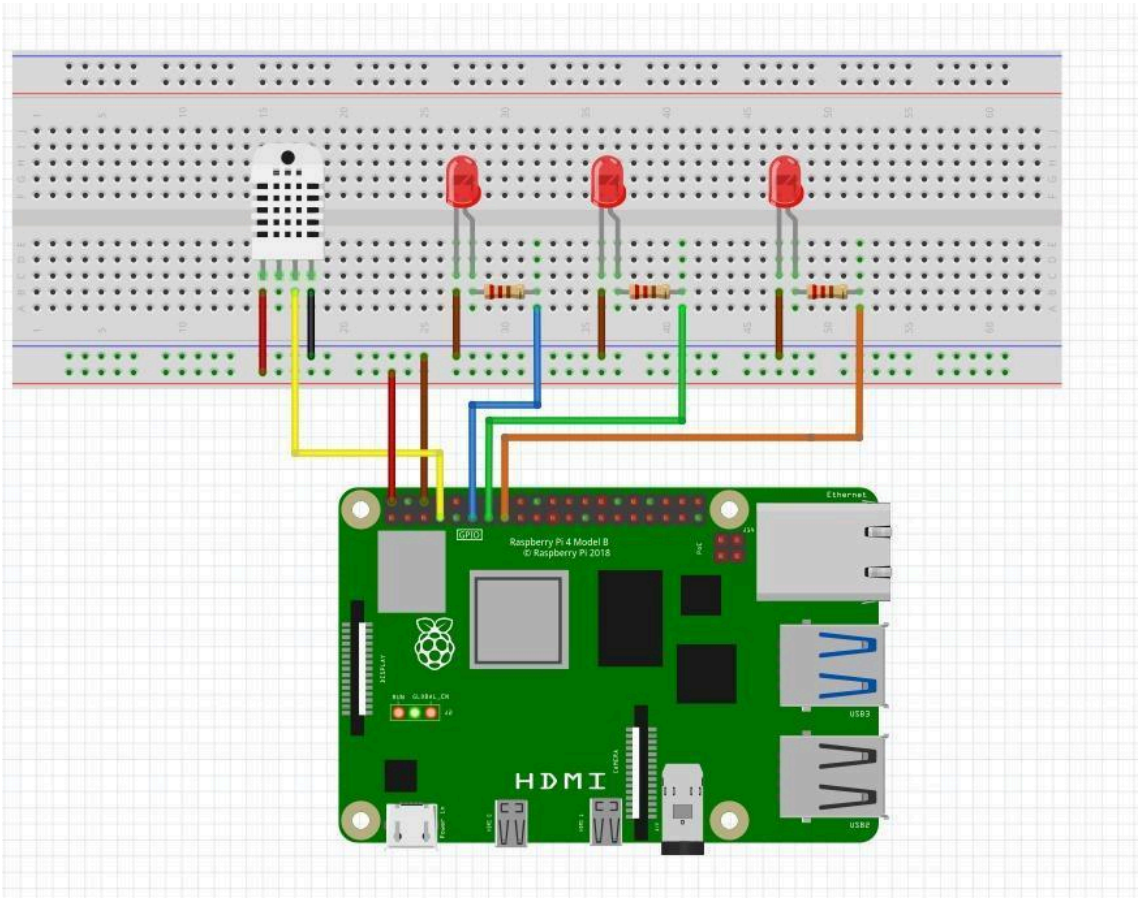
---

11. Piano di implementazione passo dopo passo:

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.

11.1 Configurazione hardware



Collegamenti del circuito

- DHT11 Sensore:
  - VCC ⟶ 5V (Pin 2)
  - GND ⟶ Massa (Pin 6)
  - DATA ⟶ GPIO 4 (Pin 7)

- LEDs (ciascuno con resistenza da 330Ω):
  - LED verde ⟶ GPIO 17 (Pin 11) ⟶ Resistenza 330Ω ⟶ Massa
  - LED giallo ⟶ GPIO 27 (Pin 13) ⟶ Resistenza 330Ω ⟶ Massa
  - LED rosso ⟶ GPIO 22 (Pin 15) ⟶ Resistenza 330Ω ⟶ Massa

  - **Connessione di massa**: Utilizzare il Pin 6 (GND) come massa comune per tutti i LED e il sensore DHT11

Passaggi di cablaggio:

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. Collegare il sensore DHT11: 5V, GND e GPIO 4
2. Posizionare 3 LED sulla breadboard con resistenze da 330Ω
3. Collegare i terminali positivi dei LED ai pin GPIO 17, 27, 22
4. Collegare tutti i terminali negativi dei LED (attraverso le resistenze da 330Ω) alla linea di massa comune
5. Collegare la linea di massa al Pin 6 (GND) del Raspberry Pi

⚠️**Importante:** una volta assemblato il circuito, controllare attentamente tutti i collegamenti prima di procedere. Assicurarsi che ogni cavo sia correttamente collegato al pin giusto: connessioni allentate o errate possono causare comportamenti imprevisti o danneggiare i componenti.

11.2 Configurazione del software

Ora che l'hardware è pronto, è il momento di configurare il codice che darà vita al tuo sistema di monitoraggio della temperatura.

1. Creare un ambiente virtuale

```
# Create project directory
mkdir temperature_monitor
cd temperature_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 per il monitoraggio della temperatura in `main.py`
- Assicurati che il file sia nella directory del progetto

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.

4. Avviare il sistema

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

# Run the monitoring system
python main.py
```

5. **Arrestare il sistema**
   - Premi **Ctrl+C** per fermare il programma in modo sicuro
   - Tutti i LED si spegneranno automaticamente

Comportamento del sistema

- **LED verde**: funzionamento normale
- **LED giallo**: temperatura in aumento (>1°C in 20 secondi)
- **LED rosso**: allarme critico (2 avvisi consecutivi) – il LED giallo si spegne
- **LED verde (lampeggiante)**: avvio del sistema

CODICE

```
import time
import board
import adafruit_dht
```

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 lgpio

# LED pin definitions
LED_NORMAL = 17      # Green LED - normal operation status
LED_ALERT = 27       # Yellow LED - temperature rise alert
LED_WARNING = 22     # Red LED - consecutive temperature rise warning

# DHT11 temperature and humidity sensor on GPIO 4 (board.D4)
dht = adafruit_dht.DHT11(board.D4)

# Initialize GPIO for LEDs
h = lgpio.gpiochip_open(0)  # Open GPIO chip 0
for pin in [LED_NORMAL, LED_ALERT, LED_WARNING]:
    lgpio.gpio_claim_output(h, pin)  # Set pins as outputs

def led_on(pin):
    """Turn on LED connected to specified pin"""
    lgpio.gpio_write(h, pin, 1)

def led_off(pin):
    """Turn off LED connected to specified pin"""
    lgpio.gpio_write(h, pin, 0)

try:
    # Startup sequence: Turn on NORMAL LED for 1s, then blink twice and stay on
    led_on(LED_NORMAL)
    time.sleep(1)

    # Blink sequence (2 times) to indicate system initialization
    for _ in range(2):
        led_off(LED_NORMAL)
        time.sleep(0.3)
        led_on(LED_NORMAL)
        time.sleep(0.3)

    # Initialize variables for temperature monitoring
    temps = []                      # List to store last 5 temperature readings
    alertas_consecutivos = 0        # Counter for consecutive temperature rise
alerts

    # Main monitoring loop
    while True:
        try:
            # Read temperature and humidity from DHT11 sensor
```

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.

```
            temperatura = dht.temperature
            humedad = dht.humidity
            print(f"Temperatura: {temperatura}°C")
            print(f"Humedad: {humedad}%")
        except RuntimeError as e:
            # Handle sensor reading errors (common with DHT sensors)
            print("Error leyendo del sensor:", e.args[0])
            temperatura = None

        if temperatura is not None:
            # Add current temperature to the list
            temps.append(temperatura)

               # Keep only the last 5 measurements (25 seconds of data at 5s
intervals)
            if len(temps) > 5:
                temps.pop(0)  # Remove oldest measurement

            # Check for temperature rise pattern (need 5 measurements)
            if len(temps) == 5:
                # Calculate temperature delta between newest and oldest reading
                  delta = temps[-1] - temps[0]  # Current temp - temp from 20s
ago

                if delta > 1:  # Temperature rose more than 1°C in 20 seconds
                    alertas_consecutivos += 1

                    # Turn on ALERT LED and turn off NORMAL LED
                    led_on(LED_ALERT)
                    led_off(LED_NORMAL)

                     # If 2 consecutive alerts, turn on WARNING LED and turn off
ALERT LED
                    if alertas_consecutivos >= 2:
                          led_on(LED_WARNING)    # Critical warning - consecutive
temperature rises
                         led_off(LED_ALERT)     # Turn off yellow LED when red is
active
                    else:
                        led_off(LED_WARNING)

                else:  # Temperature stable or decreasing
                    alertas_consecutivos = 0  # Reset consecutive alert counter
```

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.

```
                    # Normal operation: only NORMAL LED on
                    led_on(LED_NORMAL)
                    led_off(LED_ALERT)
                    led_off(LED_WARNING)

            else:
                # Not enough measurements yet - show normal status
                led_on(LED_NORMAL)
                led_off(LED_ALERT)
                led_off(LED_WARNING)

        else:
                # Sensor reading error - keep NORMAL LED on to indicate standby
mode
            led_on(LED_NORMAL)
            led_off(LED_ALERT)
            led_off(LED_WARNING)

        # Wait 5 seconds before next reading
        time.sleep(5)

except KeyboardInterrupt:
    # Handle Ctrl+C gracefully
    print("Saliendo y apagando LEDs...")

finally:
    # Cleanup: turn off all LEDs and close GPIO
    for pin in [LED_NORMAL, LED_ALERT, LED_WARNING]:
        led_off(pin)
    lgpio.gpiochip_close(h)  # Close GPIO chip connection
```

Ora puoi eseguire il programma e vedere il risultato:

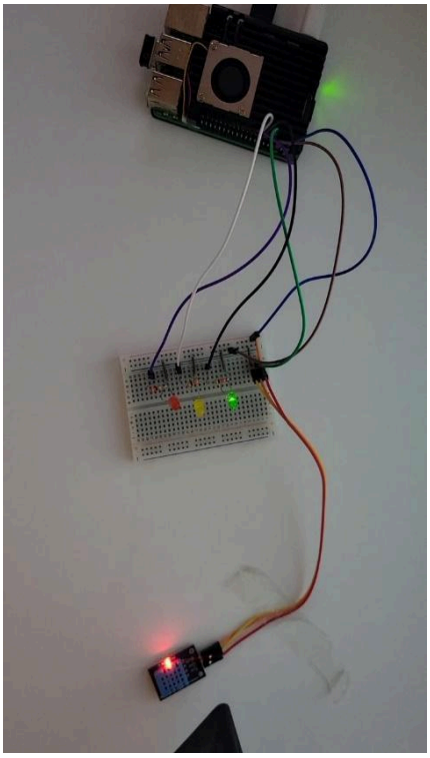
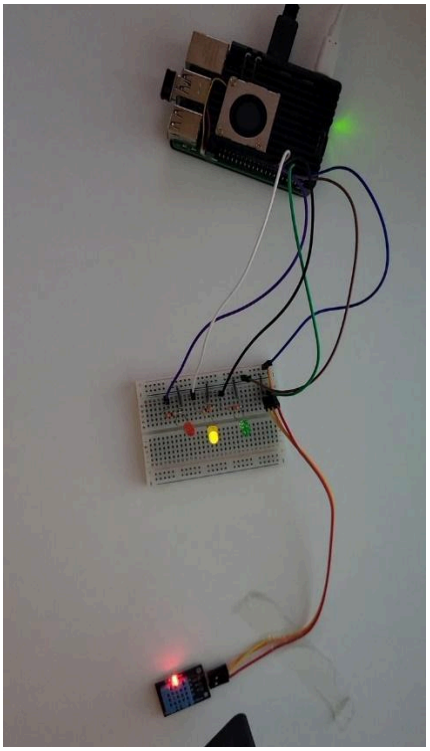
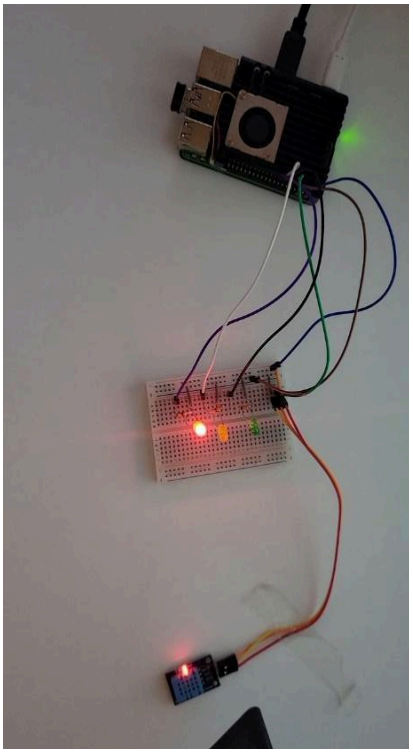
```
Temperature: 29.1°C
Humidity: 29%
Temperature: 29.2°C
Humidity: 30%
Temperature: 29.1°C
Humidity: 30%
Temperature: 29.2°C
Humidity: 30%
Temperature: 29.1°C
Humidity: 30%
Temperature: 29.2°C
Humidity: 30%
Temperature: 29.1°C
Humidity: 30%
Temperature: 29.2°C
Humidity: 30%
Temperature: 29.1°C
Humidity: 30%
Temperature: 29.1°C
Humidity: 30%
Temperature: 29.1°C
Humidity: 30%
Temperature: 29.1°C
Humidity: 30%
```

Co-funded by the European Union

This project has been
communication reflec
be held responsible fo
therein.

n. This
cannot
ntained

```
Temperature: 29.2°C
Humidity: 33%
Temperature: 29.2°C
Humidity: 33%
Temperature: 29.2°C
Humidity: 33%
Temperature: 36.6°C
Humidity: 30%
Temperature: 46.0°C
Humidity: 21%
Temperature: 47.0°C
Humidity: 16%
Temperature: 48.0°C
Humidity: 12%
Temperature: 48.0°C
Humidity: 10%
Temperature: 47.0°C
Humidity: 11%
Temperature: 46.0°C
Humidity: 11%
Temperature: 46.0°C
Humidity: 10%a
Temperature: 45.0°C
Humidity: 10%
```

12. Criteri di valutazione:

- **Configurazione hardware (35%)**: cablaggio corretto del sensore e dei LED; connessioni pulite e stabili; circuito funzionante.
- **Funzionalità del codice (30%)**: il codice viene eseguito senza errori; le variazioni di temperatura attivano correttamente i LED.

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.

- **Comprensione del sistema (20%)**: spiegazione chiara di come il sistema rileva il rischio di incendio e del contributo di ciascun componente.
- **Innovazione e miglioramenti (15%)**: proposte creative (es. allarme con buzzer, registrazione dei dati, integrazione cloud).

---

13. Lingue disponibili:

---

14. Contributori:

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.