Proiect Sisteme cu Microprocesoare (SM)

1. Informații generale

§ Titlu proiect

Sistem embedded de monitorizare și alertare în caz de cutremure și incendii

§ Nume student

Parasca Raul, grupa 1310B

§ Adresa de email

raul.parasca@student.tuiasi.ro

2. Motivația pentru alegerea temei

Motivația alegerii acestei teme constă în dorința de a implementa un sistem embedded de monitorizare a mediului înconjurător, utilizând o placă **Raspberry Pi Pico W** cu conectivitate Wi-Fi și senzori de temperatură, umiditate și vibrații. Scopul principal este dezvoltarea unei soluții simple, dar eficiente, care să poată detecta **condiții critice de mediu** (precum temperaturi ridicate, umiditate excesivă sau mișcări seismice) și să alerteze utilizatorul printr-o **interfață web în timp real**.

Tema aduce cu sine **relevanță practică**, prin aplicabilitatea sa în **sisteme de siguranță casnică**, **monitorizare industrială** sau **dispozitive inteligente de detecție**. Utilizarea unei plăci de dezvoltare de tip low-power și low-cost permite înțelegerea implementării sistemelor distribuite, conectate și eficiente energetic.

Proiectul oferă, de asemenea, oportunitatea de a lucra cu protocoale de rețea, servere HTTP, interfețe web și senzori fizici, fiind o platformă ideală pentru **învățarea integrată** a conceptelor fundamentale din domeniul sistemelor înglobate și IoT.

3. Rezumat

Proiectul urmărește construirea unui **sistem embedded de monitorizare a temperaturii, umidității și vibrațiilor**, utilizând placa **Raspberry Pi Pico W**. Sistemul este echipat cu:

- un senzor **DHT11** pentru măsurarea temperaturii și umidității,
- un **senzor de vibrații SW-420**,
- un **led RGB** pentru avertizare vizuală,
- un **buzzer activ** pentru avertizare sonoră,
- și un **potențiometru 10K** pentru reglarea sunetului.

Datele colectate sunt procesate local, iar starea sistemului este expusă printr-o **interfață web simplă**, generată automat și actualizată la fiecare 3 secunde. Atunci când temperatura depășește 40°C, umiditatea 80% sau se detectează o vibrație, sistemul semnalează o alertă atât vizual (prin LED), cât și sonor (buzzer).

Proiectul combină elemente de hardware (senzori, LED, buzzer) cu componente software (server HTTP, generare dinamică HTML) și rețelistică, fiind o aplicație completă în domeniul IoT.

4. Importanța / utilitatea în domeniul Embedded Systems

Acest proiect este un exemplu clar al modului în care tehnologiile embedded pot fi utilizate pentru dezvoltarea de **sisteme inteligente de detecție și notificare** în timp real, cu aplicații în:

- **smart home** (siguranță și monitorizare ambientală),
- **monitorizare industrială** (temperatură/umiditate în medii critice),
- **detecție seismică** sau vibrațională în medii sensibile.

Prin integrarea rețelei Wi-Fi și serverului web în cadrul unui microcontroler cu resurse limitate (Pico W), proiectul demonstrează viabilitatea construirii de soluții complete și autonome, fără dependență de platforme externe sau cloud.

5. Analiza – Design – Implementare

§ Analiza

Scopul acestui proiect este dezvoltarea unui sistem înglobat capabil să monitorizeze **temperatura, umiditatea și vibrațiile** din mediul înconjurător și să reacționeze prin intermediul unor semnale vizuale și acustice. Proiectul se bazează pe utilizarea plăcii **Raspberry Pi Pico W**, senzorului **DHT11** pentru temperatură și umiditate, și a unui **senzor de vibrații digital**. În plus, sunt utilizate un **LED RGB** pentru semnalizare vizuală și un **buzzer activ** pentru alertă sonoră. Interacțiunea și vizualizarea datelor se realizează printr-o **interfață web locală**, accesibilă de pe orice browser.

§ Design

Arhitectura proiectului combină mai multe componente hardware simple, interconectate cu Raspberry Pi Pico W:

- **Senzorul DHT11** trimite date despre temperatura și umiditatea ambientală către un pin digital al plăcii.
- **Senzorul de vibrații** detectează șocuri sau mișcări și trimite un semnal digital către microcontroller.
- **Buzzerul activ** și **LED-ul RGB** sunt folosite pentru semnalizare auditivă și vizuală, în funcție de valorile citite.
- Pico W găzduiește o **pagină HTML statică** ce permite afișarea în timp real a parametrilor monitorizați.

Designul software implică dezvoltarea unui cod în **MicroPython**, utilizând **Thonny IDE** pentru programare și depanare, fără a fi necesar un IDE complex. Comunicarea cu utilizatorul final este facilitată de o interfață web simplă, generată local de serverul HTTP integrat în MicroPython.

§ Implementare

Implementarea a constat în următoarele etape:

1. **Configurarea circuitului hardware**:
    - Conectarea senzorului DHT11 la un pin digital al plăcii (ex: GPIO 12).
    - Conectarea senzorului de vibrații la GPIO 11.

- LED-ul RGB este conectat la trei GPIO-uri (16 - roșu, 17 - verde, 18 - albastru).
- Buzzerul este conectat la un potentiometru 10K si el primeste putere de la GPIO 10.
- Alimentarea tuturor componentelor prin 3.3V și GND de pe Pico.

2. **Scrierea codului în MicroPython**:
   - Citirea valorilor de temperatură și umiditate de la DHT11.
   - Monitorizarea vibrațiilor.
   - Controlul LED-ului RGB și buzzerului pe baza valorilor citite.
   - Generarea unei interfețe HTML simple pentru afișarea valorilor în browser.
3. **Testare și optimizare**:
   - S-au testat praguri pentru temperatură și umiditate.
   - S-a verificat comportamentul LED-ului și al buzzerului la detecția de vibrații.
   - S-a optimizat timpul de răspuns și s-a asigurat stabilitatea interfeței web locale.

6. Detalii secvență demonstrativă

Atât codul sursă cât și secvența demonstrativă sunt disponibile în următoarele link-uri:

Codul: https://github.com/raulp-04/Raspberry_Pi_Pico_W-Fire-Earthquacke_sensing_station

Video marketing: https://www.youtube.com/watch?v=QlnsyX_MSLE

7. Ce se învață dacă se replică proiectul

§ Înțelegerea sistemelor înglobate:

Proiectul oferă o bază excelentă pentru înțelegerea arhitecturii sistemelor înglobate, a modului de integrare a mai multor senzori cu un microcontroller și a funcționării acestora într-un sistem real-time.

§ Lucrul cu senzori și semnalizare:

Se învață cum să citești și să interpretezi semnale de la senzori de mediu (DHT11) și mișcare (KY-002), dar și cum să folosești buzzerul și LED-urile RGB pentru feedback vizual și auditiv.

§ Dezvoltare software în MicroPython:

Replicarea proiectului ajută la familiarizarea cu MicroPython – un limbaj simplu și intuitiv, potrivit pentru proiecte embedded. Se capătă experiență în lucrul cu timere, GPIO-uri și librării pentru senzori.

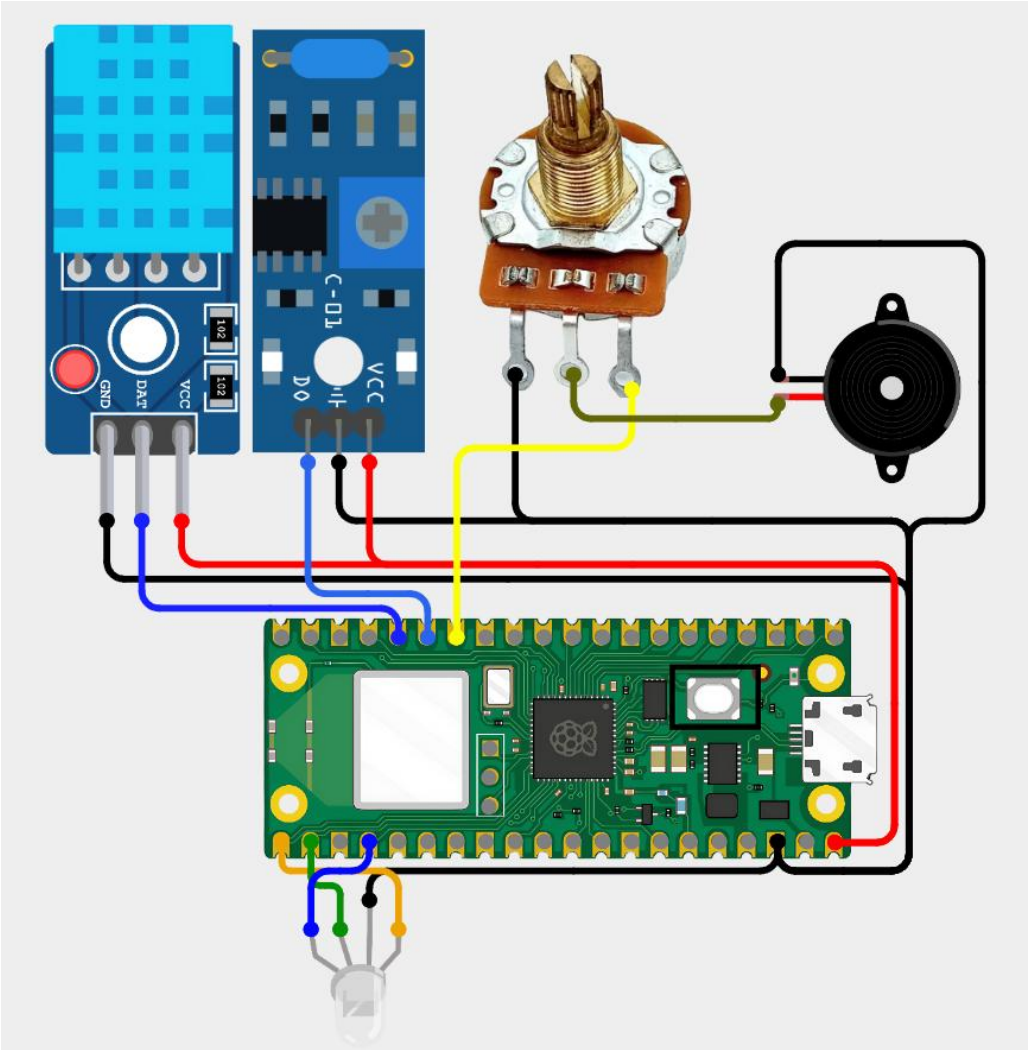
§ Interfață web pe dispozitive embedded:

Prin crearea unei pagini HTML locale, se învață cum să construiești interfețe web de bază care afișează în timp real date de la senzori, folosind un server HTTP minimalist încorporat în Raspberry Pi Pico W.

§ Aplicații practice:

Acest proiect are potențial aplicativ în domenii precum monitorizarea mediului, siguranță, case inteligente sau dispozitive de asistență. Este ușor de extins cu senzori suplimentari sau integrare cu rețele IoT.

8. Prezentare



9. Cod

§ main.py:

```
import socket
import network
from machine import Pin
from time import sleep
from webpage import generate_html
import dht

active_buzzer = Pin(10, Pin.OUT)
vibration_sensor = Pin(11, Pin.IN)
atmospheric_sensor = dht.DHT11(Pin(12))
red_led = Pin(16, Pin.OUT)
green_led = Pin(17, Pin.OUT)
blue_led = Pin(18, Pin.OUT)

# Connect to Wi-Fi
wlan = network.WLAN(network.STA_IF)
wlan.active(True)
wlan.connect("Tenda_84B8E8", "Alex2024")

while not wlan.isconnected():
    sleep(1)

print("Connected, IP address:", wlan.ifconfig()[0])

# Setup server
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
```

```
s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
s.bind(addr)
s.listen(1)

red_led.value(0)
green_led.value(0)
blue_led.value(0)
while True:
    green_led.value(1)
    atmospheric_sensor.measure()
    temperature = atmospheric_sensor.temperature()
    humidity = atmospheric_sensor.humidity()
    response = generate_html(temperature, humidity,
vibration_sensor.value())

    print("Temperature : {}C".format(temperature))
    print("Humidity : {}%".format(humidity))

    if (temperature > 40 or humidity > 80) or
vibration_sensor.value():
        active_buzzer.value(1)
        green_led.value(0)
        if temperature > 40 or humidity > 80:
            red_led.value(1)
        if vibration_sensor.value():
            blue_led.value(1)
        sleep(1)
        active_buzzer.value(0)
        red_led.value(0)
        blue_led.value(0)
        green_led.value(1)

    cl, addr = s.accept()
```

```
    print('Client connected from', addr)
    cl_file = cl.makefile('rwb', 0)

    request = cl_file.readline().decode('utf-8')
    if "buzzer=on" in request:
        active_buzzer.value(1)
        sleep(1)
        active_buzzer.value(0)
        cl.send(b"HTTP/1.1 302 Found\r\n")
        cl.send(b"Location: /\r\n")
        cl.send(b"Connection: close\r\n")
        cl.send(b"\r\n")
        cl.close()
        continue

    while True:
        line = cl_file.readline()
        if not line or line == b'\r\n':
            break

    cl.send('HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n')
    cl.send(response)
    cl.close()

    sleep(1)
```

§ webpage.py:

```
def html_too_hot():
    return """<!DOCTYPE html>
<html>
<head>
    <title>Monitorizare RPi Pico W</title>
```

```
    <meta http-equiv="refresh" content="3">
    <style>
        body {
            background-color: red;
            color: white;
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 40px;
        }
        h1 {
            font-size: 2em;
        }
    </style>
</head>
<body>
    <h1>TEMPERATURE/HUMIDITY IS VERY HIGH</h1>
</body>
</html>"""

def html_earthquake():
    return """<!DOCTYPE html>
<html>
<head>
    <title>Monitorizare RPi Pico W</title>
    <meta http-equiv="refresh" content="3">
    <style>
        body {
            background-color: orange;
            color: black;
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 40px;
        }
```

```
        h1 {
            font-size: 2em;
        }
    </style>
</head>
<body>
    <h1>EARTHQUAKE DETECTED</h1>
</body>
</html>"""

def generate_html(temp, hum, is_vibration):
    if temp > 40 or hum > 80:
        return html_too_hot()
    elif is_vibration:
        return html_earthquake()
    else:
        return f"""<!DOCTYPE html>
<html>
<head>
    <title>Monitorizare RPi Pico W</title>
    <meta http-equiv="refresh" content="3">
    <style>
        body {{
            font-family: Arial, sans-serif;
            background-color: #f0f0f0;
            color: #333;
            text-align: center;
            padding: 40px;
        }}
        h1 {{
            color: #0066cc;
            margin-bottom: 30px;
        }}
```

```
        p {{
            font-size: 1.5em;
            margin: 20px 0;
            background-color: #ffffff;
            padding: 15px;
            border-radius: 8px;
            box-shadow: 2px 2px 6px rgba(0,0,0,0.1);
            display: inline-block;
        }}
    </style>
</head>
<body>
    <h1>Date de la senzori</h1>
    <p>Temperatura: {temp}C</p>
    <p>Umiditate: {hum}%</p><br>
    <form action="/" method="get">
        <button type="submit" name="buzzer" value="on">Testeaza
Buzzer-ul</button>
    </form>
</body>
</html>"""
```

10. Bibliografie

- **MicroPython Documentation.** (n.d.). *Official MicroPython Documentation.* Retrieved from https://docs.micropython.org

- **Raspberry Pi Foundation.** (n.d.). *Getting Started with Raspberry Pi Pico W.* Retrieved from https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html

- **DHT11 Temperature and Humidity Sensor.** (n.d.). *Datasheet and Usage Guide.* Retrieved from https://components101.com/sensors/dht11-temperature-sensor

- **Socket Programming in MicroPython.** (n.d.). *Micropython Web Server with Socket Library.* Retrieved from https://techtutorialsx.com/2020/07/27/micropython-esp32-socket-server/

- **HTML & CSS Styling.** (n.d.). *MDN Web Docs - Mozilla.* Retrieved from https://developer.mozilla.org/en-US/docs/Web/HTML

- **DHT Sensor Library for MicroPython.** (n.d.). *Adafruit DHT Library adapted for MicroPython.* Retrieved from https://github.com/adafruit/Adafruit_Python_DHT

- **ChatGPT (OpenAI).** (2025). *Asistență la dezvoltarea aplicației cu Raspberry Pi Pico W, MicroPython și senzori.* Consultat la https://chat.openai.com