**6. óra: Lázmérő**

1. 5 perces beszámoló tetszőleges tartalommal és üzenettel valamint tetszőlegesen választott retorikai eszközökkel. Nekünk kell eldönteni, hogy mi lehetett az üzenet. Figyelni kell a beszéd felépítésére (bevezető, fő rész és konklúzió) és arra, hogy a tartalom minden eleme támogassa az üzenetet, illetve, hogy használjunk retorikai eszközöket.
2. 3 perces véleményalkotás  
   - bevezető, fő rész, konklúzió  
   - 2 pozitív és egy konstruktív javaslat  
   - konklúzióban összefoglalni mi lehetett az üzenet, volt-e retorikai eszköz használva és adni még egy pozitív megjegyzést.
3. Analóg hőmérséklet mérése raspberryvel:
   1. *Bevezető*:  
      A COVID járvány időszakában nem nehéz amiatt aggódni, hogy vajon elkaptuk-e a vírust vagy sem. Egyik nap arra ébredtél, hogy furcsán érzed magad, mintha kicsit melegebb is lenne a homlokod, fáradtabb is vagy. Első lépésként gondoltad megméred a lázad, de nem találtál lázmérőt.

Mint egy ügyes kis tech alkotó, (maker) azonnal eszedbe jutott, hogy van neked thermisztorod, összedobhatsz gyorsan abból is egy lázmérőt. Mint az igazi lázmérőknél, egy gomb lenyomása után csak várunk, amíg a mérő kiírja a hőmérsékletünket.

* 1. *Előkészületek*:  
     Készítsünk ki próbapaneleket, jumper kábeleket, 10 kOhm ellenállásokat a thermisztornak, NTCC 10k thermisztorokat, nyomógombokat és az MCP3008 ADCt. Emellé szükség lesz mérőműszerre is.   
     - Rakjuk össze az áramkört. Teszteljük le műszerrel, hogy kb. 1.6 Vot mérünk-e a thermisztor lábain (feltéve, hogy szobahőmérséklet van, kb. 25 fok). Mindenesetre, amit mérünk a lábakon, azt kell majd az MCP3008al is mérnünk.   
     - Teszteljük le a műszerrel, hogy működnek-e illetve pythonból is, hogy jó e az áramkör.  
     - Teszteljük le az MCP3008at is pythonból, hogy azt olvas-e amit a műszer.   
       
     *Részletekre bontás:*Elkészíteni az áramkört.
     + Importálni az MCP3008, LED objektumokat, numpy csomagot, függvényt az interpolálásra és fájl beolvasó függvényt.
     + Inicializálni a gombot és az ADC-t.
     + Definiálni egy függvényt, ami egy megadott fájlból kinyeri a thermisztor hőmérséklet-ellenállás függését.
     + Definiálunk egy függvényt ami a mért feszültséget átalakítja ellenállássá a feszültségosztó elve alapján.
     + Definiálunk egy függvényt, hogy a hőmérséklet-ellenállás függés ismeretlen pontjait interpoláljuk.
     + Definiálunk egy összehasonlító függvényt ami egyenként összehasonlítja az utolsó három mérés értékét.
     + A nyomógombbal elindítani a mérést.
     + Egy végtelen while ciklusban mérjük a feszültséget, kiszámoljuk az ellenállás változást, abból az aktuális hőmérsékletet, és a hőmérsékleteket és megállapítjuk az összehasonlító függvénnyel, hogy mikor van vége a mérésnek.
  2. *Elektronika:*  
     - Készítsük el az áramkört. Érdemes kihasználni a közös földelést és tápot, hiszen sok dróttal dolgozunk majd.  
     - Vezessük le, hogyan számoljuk ki az ellenállást a mért feszültségből.
  3. *Kód:*  
     - Egeszítsük ki a lenti kódvázlatot.  
     - Vegyük át a string műveleteket különös tekintettel a *strip* és *split* metódusokra.  
     - Vegyük át a lista műveleteket megemlítve a *pop* és az append metódust is.   
     - Vegyük át az interpolálást és nézzünk rá példákat:  
       
     f = interp1d(x, y)

new\_x = np.array([-1.5, 3.5])

f(new\_x)  
  
- Nézzük át a műveleteket a *numpy array*el:  
  
a = np.array(a) # numpy.array letrehozasa

print(a)

print(type(a))

print(a\*2) # szamsor elemeinek duplazasa

print(a\*\*2) # szamsor elemeinek negyzetre emelese

print(np.sqrt(a)) # szamsor elemeinek negyzetgyoke  
  
- Vegyünk példát arra, hogyan lehet 2 tizedes pontossággal kinyomtatni értékeket.  
- Nézzük át, hogy kell egy szám abszolút értékét kiszámolni.  
- Nézzük meg, hogy összegezzük egy lista elemeinek értékét.

* 1. *Tesztelés:*  
     - teszteljük a thermisztorokat a műszerrel, illetve teszteljük a programot.
  2. *Fejlesztés:*  
     - Írjunk egy plusz feltételt a while ciklusba, hogy ha az nem áll meg 15 másodpercen belül, akkor az épp az aktuális utolsó 3 mérés átlaga legyen a hőmérséklet.

- A t\_body változót írjuk át numpy.array formátumúvá, és ennek megfelelően módosítsuk a kód többi részét, beleértve a compare függvényt is.

Kód minta:

# importald az MCP3008 es Button csomagokat a gpiozerobol

import numpy as np

# importald a sys es time csomagokat

from scipy.interpolate import interp1d

# a temperature\_functions modulbol importald a read\_temp\_raw fuggvenyt

mcp = # inicializald az ADC konvertert a megfelelo csatornaval

button = # inicializald a gombot

def read\_mapping(name):

    data = # olvasd be a name fajl tartalmat a read\_temp\_raw

    # a data listanak az elso eleme a fejlecet tartalmazza, igy ki kell venni a listabol a pop metodussal

    tc = # inicializalj egy ures listat a homersekleti ertekek reszere

    r = # inicializalj egy ures listat az ellenallasi ertekek reszere

    for line in data:

        clean\_data = # minden sor ket szamot tartalmaz ami vesszovel van elvalasztva. hasznald a strip metodust az ujsorok es az ures helyek eltuntetesere, majd a split metodust, hogy a stringet szetvagd a vesszok menten

        # EZT A SORT NE SZERKESZD: a clean\_data ket elemet fog tartalmazni, az elso a homerseklet, a masodik az ellenallas kOhmban kifejezve, string formatumban mind a ketto

        # bovitsd a tc listat a clean\_data elso elemevel (homerseklet), de vigyazz, hogy a float paranccsal at kell alakitani a stringet szamma

        # bovitsd az r listat a clean\_data masodik elemevel (ellenallas), de vigyazz, hogy a float paranccsal at kell alakitani a stringet szamma, valamint 1000el szorozni kell, hogy Ohmba kapjuk meg az erteket

    return np.array(tc), np.array(r)

def voltage2resistance(v\_out, v\_in=3.3, r=9.82e3):

    # a fuggveny adja vissza az adott ellenallast az alabbi keplet alapjan:

    # ellenallas \* mert feszultseg / (tapfeszultseg - mert feszultseg)

def interpolate\_temperature(resistance, tc, r):

    f = interp1d(r,tc)

    return f(resistance)

def compare(tc, eps=0.01):

    # NE SZERKESZD: tc egy lista ami 3 egymas utan mert homersekletet tartalmaz

    if ........: # ha a mert homersekletek paronkenti kulonbsegenek abszolut erteke kisebb mint eps, pl. abs(tc[0]-tc[1]) < eps

        t\_mean = # szamold ki a 3 homerseklet atlagat, hasznald a sum fuggvenyt, a lista elemeinek osszeadasara es a len fuggvenyt, hogy megkapd hany szamot atlagolsz

        # nyomtasd ki, hogy: A testhomersekleted t\_mean Celsius fok; t\_mean-t ket tizedes pontossaggal add meg

        sys.exit(0)

tc, resistance = # olvasd be a homerseklet ellenallas lekepzest a read\_mapping fuggvennyel. A fajl neve ntcc.csv

t\_body = []

print('Place your hand on the thermistor')

# varj a gomb lenyomasara. Ez az esemeny inditja a merest

while True:

    time.sleep(0.8)

    rt = voltage2resistance(......., v\_in=3.3, r=9.82e3)  # add meg a mert feszultseget bemeno parameternek

    t\_therm = # becsuld meg az aktualis ellenellasertekhez tartozo homersekletet az interpolate\_temperature fuggvennyel

    # bovitsd a t\_body listat a t\_therm homerseklettel

    if len(t\_body) < 3:

        continue

    else:

        # hivf meg compare fuggvenyt a mert homersekletekkel es homerseklet kulonbseg toleranciaval (eps)

        # vedd ki a t\_body listabol a legregebbi mert homersekletet