



**SZÉCHENYI  
EGYETEM**  
UNIVERSITY OF GYŐR  
GÉPÉSZMÉRNÖKI, INFORMATIKAI  
ÉS VILLAMOSMÉRNÖKI KAR

# **Mikroelektromechanikai rendszerek beadandó**

## **Raspberry Pi hőmérséklet és páramérő állomás DHT22 szonda, fejlesztői dokumentáció**

**Bacsa Balázs**

**IOFNFT**

**[2023]**

## 1. Tervezési fázis

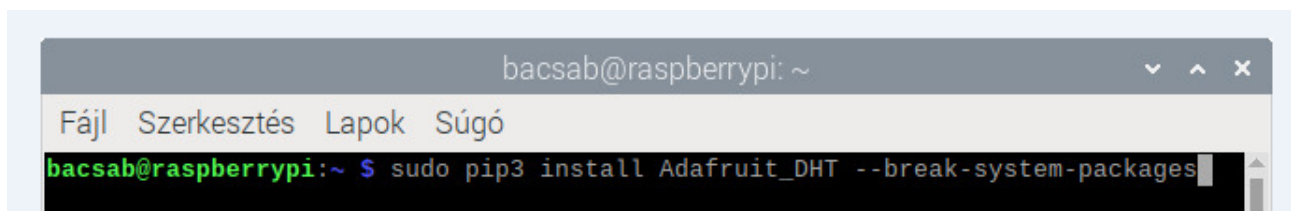
A projekt kivitelezésére egy Raspberry Pi 3B típusú mikroszámítógépet választottam, amihez a webáruházakban kerestem egy DHT22 típusú hőmérséklet- és páramérő szondát. Az elgondolásom alapján mikro SD kártyára került telepítésre a Raspbian operációs rendszer, majd Python nyelven történik a szoftver megírása, amely kommunikál a szondával és az adatok eltárolja adatbázisban, illetve azokat grafikonon is meg fogja tudni jeleníteni. A tervezési fázis során próbáltam információkat nyerni különböző weboldalakból a projekt elkészítéséhez, mivel még hasonló feladatot nem végeztem.

## 2. Felmerülő problémák a kivitelezés során

Az első nehézség, amivel szembe kellett néznem, hogy mikroszámítógépre telepítésre kerüljön az operációs rendszer. Ehhez szükségem volt egy mikro SD kártyára és a le kellett töltenem a Raspbian operációs rendszert. Maga a telepítés egyszerűbb volt, mint gondoltam, könnyedén elvégezhető volt.

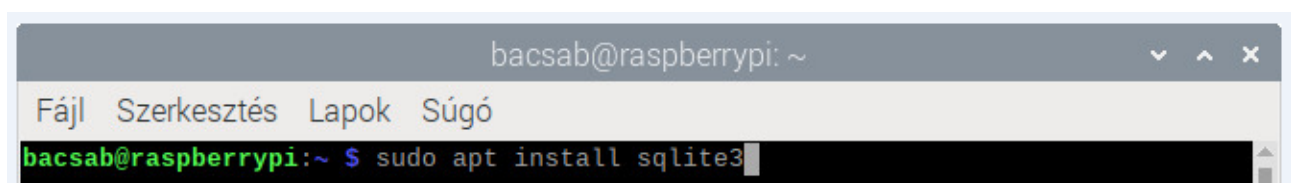
A következő lépés a szonda csatlakoztatása volt, amit a 3. pontban található dokumentációk alapján végeztem el.

A csatlakoztatás követően megpróbáltam adatokat kinyerni a szenzorból, hogy lássam a működését, illetve, hogy milyen formátumban tudom megjeleníttetni. Természetesen ehhez telepíteni kellett az Adafruit\_DHT csomagot az eszközre. A csomag telepítése problémába ütközött, így a hibaüzenet alapján kikerestem a megfelelő parancsot.



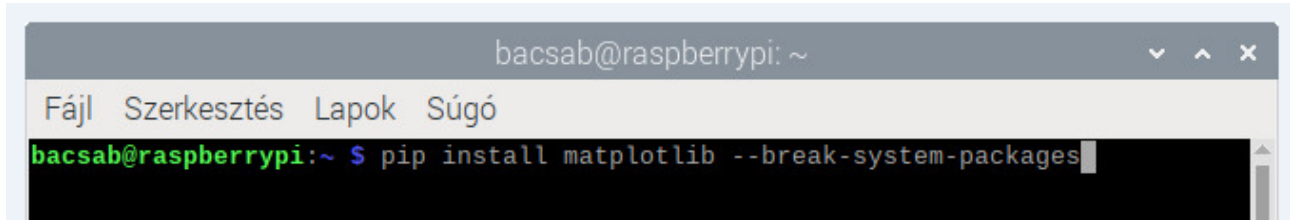
```
bacsab@raspberrypi: ~  
Fájl Szerkesztés Lapok Súgó  
bacsab@raspberrypi:~ $ sudo pip3 install Adafruit_DHT --break-system-packages
```

Ezt követően az adatbázisban eltárolást szerettem volna megvalósítani, amihez az sqlite3 csomag telepítésére volt szükség, a telepítés során itt is hibába futottam, amit a szonda telepítése során eszközölt módon orvosoltam.



```
bacsab@raspberrypi: ~  
Fájl Szerkesztés Lapok Súgó  
bacsab@raspberrypi:~ $ sudo apt install sqlite3
```

A mért adatokat szerettem volna grafikonon megjeleníteni, amihez a matplotlib.pyplot csomag telepítésére volt szükség. Ezzel grafikusán megjelenítettem az adatokat, azonban az X tengelyt el kellett forgatni 45 fokkal, hogy esztétikusabb élményt nyújtson.



```
bacsab@raspberrypi: ~  
Fájl Szerkesztés Lapok Súgó  
bacsab@raspberrypi:~ $ pip install matplotlib --break-system-packages
```

Ekkor találkoztam azzal a nehézséggel, hogy az adatbázisból kinyerni próbált adatokat nem tudtam megjeleníteni a grafikonon, hiába olvastam utána fórumokon. Ezt követően az az ötletem támadt, hogy az adatbázis tartalmát exportálom egy CSV fájlba, amit már könnyedén meg is tudtam jeleníteni grafikusán.

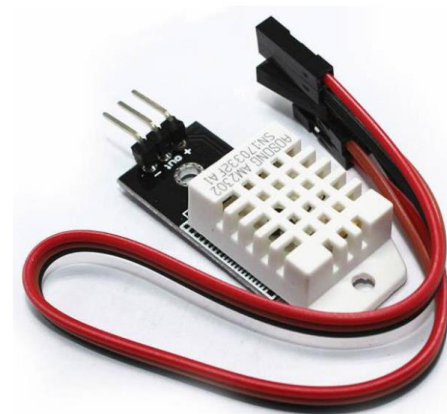
A méréseknél valamiért nem frissültek megfelelően a grafikon értékei és nem végezte el 30 másodpercenként az új adatok beolvasását, így másik módon igyekeztem megjeleníteni azokat. Egy megadott idő után bezárom a grafikont és beolvas egy új adatot, amivel frissíti grafikont is. Sajnos tudom, hogy nem a legjobb megoldás, de ezt a hibát nem sikerült orvosolnom.

### 3. Áramköri rajzok, felhasznált elemek

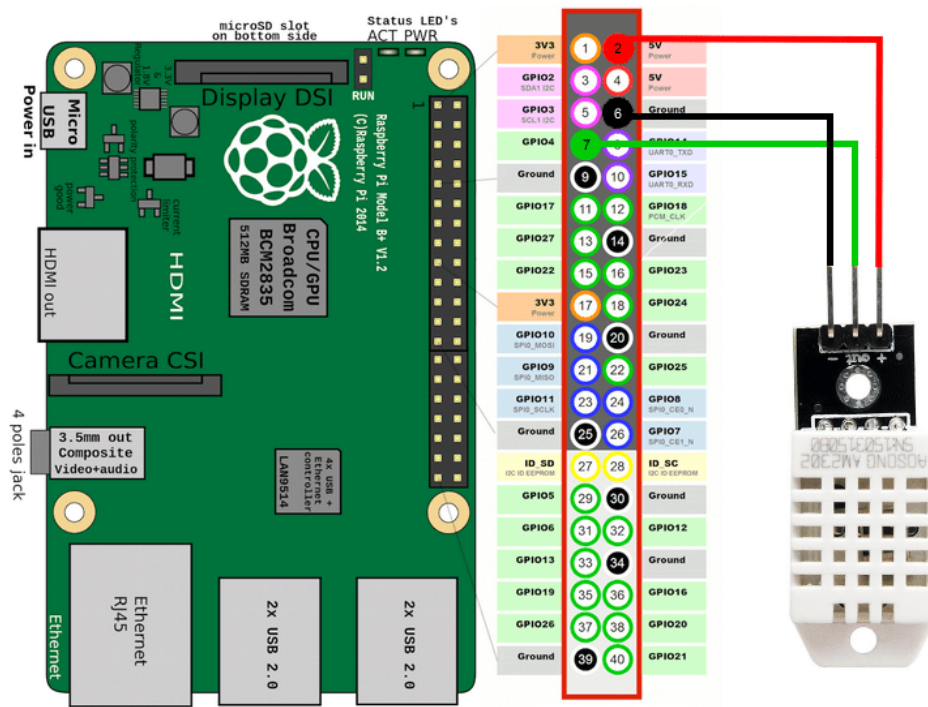
Raspberry Pi 3B modell



DTH22 szonda



DHT22 szonda bekötéséhez használt útmutató



Az eszköz összeállított állapotban



## 4. Hardver specifikációk

Raspberry Pi 3B modell specifikációja:

- Processzor: Broadcom BCM2837 chipset. 64 bit 1.2GHz Quad-Core ARM Cortex-A53
- Wireless: BCM43143 802.11 b/g/n Wireless LAN and Bluetooth 4.1 (Bluetooth Classic and LE)
- GPU: Dual Core VideoCore IV® Multimedia Co-Processor. Open GL ES 2.0, hardware-accelerated OpenVG, and 1080p30 H.264 high-profile decode. Capable of 1Gpixel/s, 1.5Gtexel/s or 24GFLOPs with texture filtering and DMA infrastructure
- Memória: 1GB LPDDR2
- Operációs Rendszer: Micro SD-ről futtatható Linux OS vagy Windows 10 IoT
- Méret: 85 x 56 x 17mm
- Tápigény: Micro USB 5.1V, 2.5A

DTH22 szonda specifikációja:

- Méretek 45 x 15 x 10 mm
- Szállítási érzékelő, áthidaló kábel
- Bemeneti feszültség 3,3 - 6 V
- Csatlakozások 3 érintkezők: 1x VCC, 1x digitális, 1x GND
- Felbontási pontosság 0,1
- Hőmérséklet-mérési tartomány -40 - 80 ° C
- Hőmérséklet mérési pontosság  $\pm 0,5$  ° C
- Páratartalom 0 - 100%
- Luftf. -Mérési pontosság  $\pm 2\%$  relatív páratartalom
- Energiafogyasztás 1 - 1,5mA
- Készenléti energiafogyasztás 40-50 $\mu$ A
- Mintavételezés: 2 másodperc

## 5. Szoftver specifikáció

A program célja, egy adott helyiség hőmérséklet- és páratartalmának monitorozása megadott időközönként. A mért adatokat eltároljuk adatbázisban, illetve grafikonon is megjelenítjük a felhasználók számára.

Az elkészített szoftver indítást követően importálja működéshez szükséges funkciókat (szonda kezelése, sql és csv kezelés, továbbá a dátum és grafikonok kezelése).

Megadjuk a szenzor típusát és a GPIO tűskét, amire csatlakoztatva van. Ellenőrizzük, hogy az adatbázis mér létezik-e amibe dolgozni szeretnénk. Amennyiben már megtalálható tovább lépünk, ha pedig nem létezik, akkor elkészítjük.

Kiolvassuk a mért adatokat a szenzorból, és formázzuk a dátumot, majd áttöltjük adatbázisba. Esetleg, ha nem működne a szenzor jelezzük a felhasználó felé egy "Szenzorolvasási hiba" üzenettel. A mért értéket megjelenítem a konzolon is, majd bezárom a kapcsolatot az adatbázissal és exportálom a tartalmát egy CSV fájlba. Az így kapott értékeket pedig beolvasom és grafikonon megjelenítem. A program automatikusan 30 másodperc után bezárja a grafikont és egy újabb mérést végez.

Ezt addig folytatja, ameddig meg nem szakítjuk manuálisan a program futását.

## 6. Rendszerkövetelmények

Rendszerkövetelmény tekintetében a minimum ajánlott konfiguráció az alábbiak szerint alakul:

- Raspberry Pi 3B modell. Régebbi modelleken lassú futást és működést tapasztalható.
- Raspberry Pi OS az alábbiak szerint:
  - Kiadás dátuma: 2023.10.10
  - Rendszer: 64-bit
  - Kernel verzió: 6.1
  - Debian verzió: 12 (bookworm)
- Telepítendő csomagok:
  - Adafruit\_DHT
  - sqlite3
  - csv
  - matplotlib.pyplot

## 7. Fejlesztett kód részletezése

```
import Adafruit_DHT
import sqlite3
import csv
import matplotlib.pyplot as plt
from datetime import datetime

import time
```

Importálásra kerültek a programban használni kívánt funkciók:

- Szenzor felismeréséhez szükséges
- Adatbázis létrehozásához szükséges
- Adatbázis CSV-be exportálásához szükséges
- Grafikon megjelenítéséhez szükséges
- Dátum és idő kezeléséhez szükséges
- Időzítés használatához szükséges

```
sensor = Adafruit_DHT.DHT22
pin = 4
```

Itt megadtam, a szenzor pontos típusát és hogy melyik GPIO tűskén történik a kommunikáció.

```
while True:
    # Adatbázis csatlakozás
    conn = sqlite3.connect('szenzor_adatok.db')
    cursor = conn.cursor()
```

Ciklusba szerveztem az adatbázishoz történő csatlakozást és majd későbbiekben látható, hogy a mérések is ebbe a részbe lesznek elvégezve.

```
cursor.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='meresek'")
if cursor.fetchone() is None:
    cursor.execute('''
        CREATE TABLE meresek (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            datum TIMESTAMP,
            homerseklet REAL,
            paratartalom REAL
        )
    ''')
```

Először ellenőrizzük, hogy létezik-e az adatbázisunk, majd ha nem találja a rendszer, akkor létrehoz egy „meresek” nemű táblát „id”, „datum”, „homerseklet” és „paratartalom” struktúra alapján.



```
humidity, temperature = Adafruit DHT.read_retry(sensor, pin)
now = datetime.now()
formatted_date= now.strftime("%Y-%m-%d %H:%M:%S")
now=formatted_date
```

A mérési adatokat kiolvassuk a szenzorból, a standard dátumformátummal, majd formázzuk használathoz célszerű módon megfelelően.

```
if humidity is not None and temperature is not None:
    cursor.execute("INSERT INTO meresek (datum, homerseklet, paratartalom) VALUES (?, ?, ?)", (now, temperature, humidity))
    conn.commit()
else:
    print('Szenzorolvasási hiba')
```

Amennyiben a páratartalom és a hőmérséklet mért értéke nem „üres” azaz a szonda működik, eltárolja az adatbázis a pontos időpontot, a hőmérsékletet és a páratartalmat is az adatbázisban. Azonban, ha nem képes mérést végezni, visszajelez a felhasználónak, hogy szenzorolvasási hiba történt.

```
cursor.execute("SELECT * FROM meresek")
records = cursor.fetchall()
for record in records:
    id, datum, homerseklet, paratartalom = record
    print(f"ID: {id}, Dátum: {datum}, Hőmérséklet: {homerseklet:.1f} °C, Páratartalom: {paratartalom:.1f} %")
```

Lekérdezzük az adatbázis tartalmát és kiírjuk konzolra az összes benne található értékeket az alábbiaknak megfelelő módon. Minden lefutás esetén kiírja az adatbázis teljes tartalmát.

```
ID: 1, Dátum: 2023-12-02 08:41:38, Hőmérséklet: 24.9 °C, Páratartalom: 37.5 %
ID: 2, Dátum: 2023-12-02 08:42:12, Hőmérséklet: 25.0 °C, Páratartalom: 37.1 %
ID: 3, Dátum: 2023-12-02 08:42:42, Hőmérséklet: 25.0 °C, Páratartalom: 37.1 %
```

```
conn.close()
```

Bezárjuk az összes kapcsolatot az adatbázissal, mert nyitott kapcsolat esetén nem lehet elvégezni az adatbázis áttöltését CSV formátumba.

```
with open('exported_data.csv', 'w', newline='') as csv_file:
    csv_writer = csv.writer(csv_file)

    csv_writer.writerow(["ID", "Dátum", "Hőmérséklet (°C)", "Páratartalom (%)"])

    csv_writer.writerows(records)
```

Megnyitunk egy CSV fájlt írásra csv\_file változóként, aminek a neve exported\_data.csv és a beállítjuk a sorvégi karaktert. A fejlécbe létrehozuk az oszlopok nevét, majd beleírjuk a rekordokat.

```
csv_file.close()
```

Ezzel a paranccsal bezárjuk a CSV fájlt, mivel a következő ciklus futásáig nem lesz rá szükségünk és szeretnénk kirajzolni a benne található értékeket.



```

with open('exported_data.csv', 'r') as csv_file:
    csv_reader = csv.reader(csv_file)
    next(csv_reader)
    data = list(csv_reader)

    dates = [row[1] for row in data]
    temperatures = [float(row[2]) for row in data]
    humidities = [float(row[3]) for row in data]

```

Megnyitjuk olvasásra a CSV fájlunkat, átugorjuk a fejléct, majd beolvassuk az értékeket. A hőmérséklet és a páratartalom értékeket lebegőpontos típusban jelenítjük majd meg.

```

plt.figure(figsize=(12, 6))
plt.plot(dates, temperatures, label='Hőmérséklet (°C)', marker='o', linestyle='-', color='b')
plt.plot(dates, humidities, label='Páratartalom (%)', marker='o', linestyle='-', color='g')

```

Készítünk egy vonaldiagrammot a matplotlib könyvtár segítségével, amely 12 hüvelyk széles és 6 hüvelykes magassággal jelenik meg a kijelzőn. Ezen a hőmérséklet kék vonallal kerül ábrázolásra, a mért értékeket pedig pedig „O”-val jelöli majd. A páratartalom pedig hasonló módon jelenik meg, mint a hőmérséklet, egyedül a diagramm színe változik zöldre.

```

plt.title('Hőmérséklet és Páratartalom')
plt.xlabel('Dátum')
plt.xticks(rotation=45)
plt.ylabel('Érték')
plt.legend()

```

A diagramm címének beállításra kerül a „Hőmérséklet és Páratartalom” szöveg. X tengelyt elnevezzük „Dátum”-nak, majd elforgatjuk a feliratait 45 fokkal a könnyebb olvashatóság érdekében. Y tengelyt elnevezzük „Érték”-nek, mivel itt jelenítjük meg a fokot és a páratartalmat is. Végül hozzáadjuk a jelmagyarázatot a diagrammhoz.

```

plt.tight_layout()
plt.show(block=False)
plt.pause(30)

plt.close()

```

Automatikusan beállítjuk az elrendezését a diagrammnak, hogy azok a lehető legoptimálisabb módon elférjenek a megjelenítés során. A program további futtatását biztosítjuk, hogy megállna az ábra megjelenítése miatt, tehát így a mérések folytatódnak. 30 másodpercet várunk, mielőtt bezárjuk az ábrát, majd folytatjuk tovább a ciklust az elejéről a „while” résztől.

A program a végtelenségig fut, sajnos bezárására csak manuális megszakítással van lehetőség. Túlzottan nagy adatmennyiség esetén a diagramm túlszűfolttá válik, így célszerű az adatbázist kiüríteni.

## 8. Pillanatkép a program futásáról

