

Műszaki leírás

Hőmérséklet és páratartalom mérése és kijelzése diagrammal

Felhasznált hardver: - Raspberry Pi 4B Rev 1.5

- Bosch BME280 hőmérséklet-, páratartalom-, légnyomás-mérő szenzorral (I2C/SPI)
- monitor, egér, billentyűzet

Bosch BME280 szenzor jellemzői: A páratartalom mérése 0...100% $\pm 3\%$, a hőmérséklet -40...85 °C $\pm 1\%$, a légnyomás 300...1100 hPa ± 1 hPa pontosságú. Kompakt I2C eszköz, melyen mindhárom szenzor integrálva van. Csatlakoztatható 3 V vagy 5 V-os logikai mikrokontrollerekhez I2C vagy SPI adatátviteli móddal.

Technikai adatok

- Páratartalom mérési tartomány: 0..100%
- Hőmérséklet mérési tartomány: -40..85°C
- Légnyomás mérési tartomány: 300...1100hPa
- Tápfeszültség: 3-5V DC
- Csatlakozás módja: I2C/SPI
- I2C címe: 0x77, címütközés esetén beállíthatjuk a 0x76-os címet is az SDO láb GND vagy VCC-re kötésével

A Raspberry egy kosaras kártya méretű, egyetlen áramkörü lapra integrált egykártyás számítógép, amelyet az Egyesült Királyságban fejlesztettek oktatási célokra.

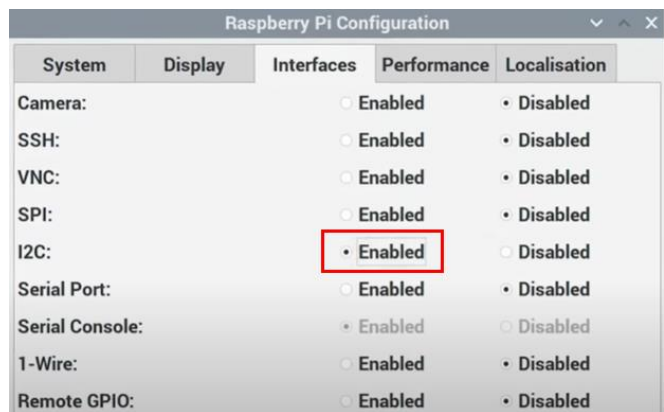
Raspberry Pi 4B technikai jellemzők

- Processzor: Broadcom BCM2835, 64-bit SoC @ 1.5GHz
- Memória: 4GB LPDDR4
- Kapcsolat: 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless LAN, Bluetooth 5.0, 2x USB 3.0 ports, 2x USB 2.0 ports, BLE Gigabit Ethernet
- GPIO (General Purpose Input/Output): 40 pin
- Videó és hang: 2xmicro HDMI ports,2-lane MIPI DSI display port,2-lane MIPI CSI camera port
- SD kártya támogatás: Micro SD az operációs rendszernek, illetve adattárolás céljából

Telepített operációs rendszer: Rasbian GNU/Linux Version:12

A megvalósított feladatot először a Micro SD kártyára letöltött Raspberry operációs rendszer telepítésével kezdtem. Ehhez a hivatalos rPi honlapról letöltöttem a Raspberry Pi Imager file-t Windowsra. Futtatás után kiválasztottam az ajánlott Raspberry Pi OS-t. Miután a telepítés befejeződött a Micro SD kártyára, behelyeztem a Raspberry-be és az operációs rendszer első betöltésénél elvégeztem a különböző beállításokat, mint időzóna, nyelv, jelszó választás.

Egy fontos beállítást is el kellett végezni annak érdekében, hogy a számítógép felismerje a Bosch BME280-as szenzort. Ez pedig a Raspberry konfiguráción belül az interfész menü alatt található I2C engedélyezést takarja.



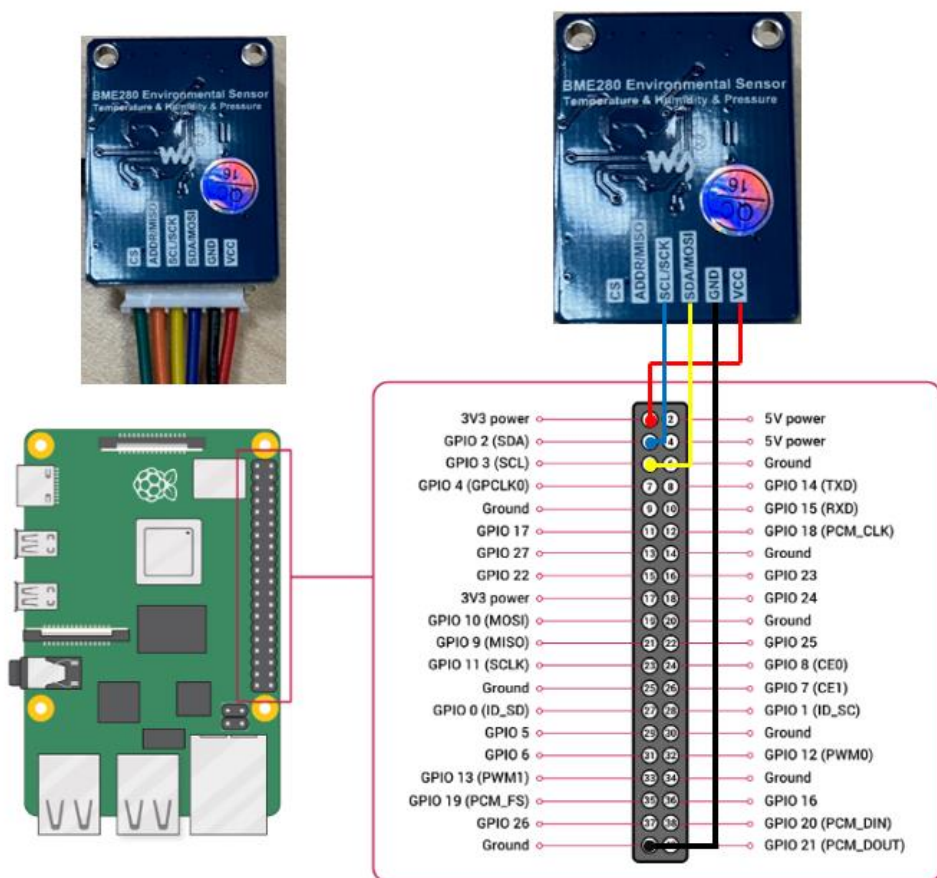
1. ábra I2C kommunikációs protokoll engedélyezése

Mindezek után kikapcsoltam a Raspberyyt és összeköttem a BME280-as szenzorral. Ehhez segítséget a honlapon olvastam, ahonnan a szenzort rendeltem. Itt volt egy hiba, illetve felcserélés a leírásban a PIN-ek esetében, ugyanis a PIN3-ra írta, hogy az a Pi SCL lába, míg a PIN5-re, hogy az a Pi SDA lába.

Raspberry Pi-vel a legegyszerűbben I2C eszközként csatlakoztattam. Egyszerűen kösd össze a Pi SCL lábát (PIN-3) a BME280 szenzor SCK lábával és a Pi SDA (PIN-5) lábát a BME280 szenzor SDI-vel. Már csak tápfeszültséget kell adni neki (3.3V lábat a VIN lábra) és természetesen a GND lábakat összekötni. Az áramkör összeállítását el is végeztük.

2. ábra Forrás: https://www.rpibolt.hu/termek/bosch_bme280_homerseket-paratartalom-legnyomasmero_szenzor_i2cspi.html

A bekötést sematikususan ábrázolva az alábbi módon valósítottam meg:



3. ábra Bosch BME280 szenzor csatlakoztatása

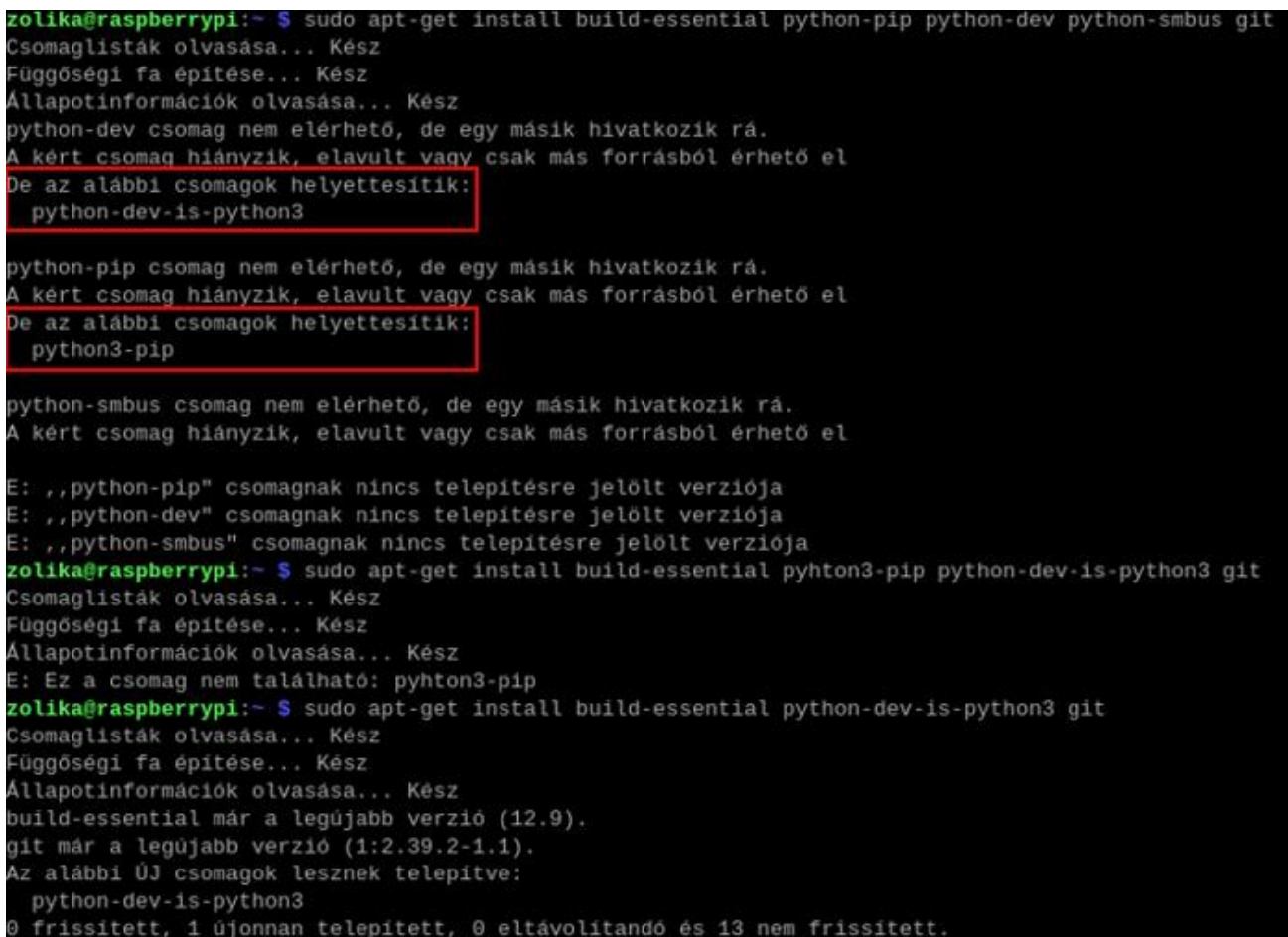
Miután a szenzort fizikálisan bekötöttem ismét bekapcsoltam a Pi-t és elkezdtem a szükséges driverek telepítését a terminálban beírható parancsok segítségével. Ehhez a honlap nyújtott segítséget, ahonnan a szenzort rendeltem. Első körben az Adafruit_Python_GPIO python drivert kell telepíteni, mert az I2C

```
sudo apt-get update
sudo apt-get install build-essential python-pip python-dev python-smbus git
git clone https://github.com/adafruit/Adafruit_Python_GPIO.git
cd Adafruit_Python_GPIO
sudo python setup.py install
```

4. ábra Adafruit driver telepítési útmutató

interfészt annak segítségével tudjuk használni, ezért a szenzor használata előtt azt mindenképp installálni szükséges. Másodjára le kell tölteni az Adafruit_Python_BME280 drivert. Ezt a következőképpen tehetjük meg a leírás szerint fentről lefelé haladva:

Itt a **sudo apt-get install build-essential python-pip python-dev python-smbus git** parancsot követően a következő problémák léptek fel:



```
zolika@raspberrypi:~$ sudo apt-get install build-essential python-pip python-dev python-smbus git
Csomaglisták olvasása... Kész
Függőségi fa építése... Kész
Állapotinformációk olvasása... Kész
python-dev csomag nem elérhető, de egy másik hivatkozik rá.
A kért csomag hiányzik, elavult vagy csak más forrásból érhető el
De az alábbi csomagok helyettesítik:
  python-dev-is-python3

python-pip csomag nem elérhető, de egy másik hivatkozik rá.
A kért csomag hiányzik, elavult vagy csak más forrásból érhető el
De az alábbi csomagok helyettesítik:
  python3-pip

python-smbus csomag nem elérhető, de egy másik hivatkozik rá.
A kért csomag hiányzik, elavult vagy csak más forrásból érhető el

E: „python-pip” csomagnak nincs telepítésre jelölt verziója
E: „python-dev” csomagnak nincs telepítésre jelölt verziója
E: „python-smbus” csomagnak nincs telepítésre jelölt verziója
zolika@raspberrypi:~$ sudo apt-get install python3-pip python-dev-is-python3 git
Csomaglisták olvasása... Kész
Függőségi fa építése... Kész
Állapotinformációk olvasása... Kész
E: Ez a csomag nem található: python3-pip
zolika@raspberrypi:~$ sudo apt-get install build-essential python-dev-is-python3 git
Csomaglisták olvasása... Kész
Függőségi fa építése... Kész
Állapotinformációk olvasása... Kész
build-essential már a legújabb verzió (12.9).
git már a legújabb verzió (1:2.39.2-1.1).
Az alábbi ÚJ csomagok lesznek telepítve:
  python-dev-is-python3
0 frissített, 1 újonnan telepített, 0 eltávolítandó és 13 nem frissített.
```

5. ábra Python-pip és python-dev telepítése

```
zolika@raspberrypi:~$ git clone https://github.com/adafruit/Adafruit_Python_GPIO.git
Cloning into 'Adafruit_Python_GPIO'...
remote: Enumerating objects: 515, done.
remote: Total 515 (delta 0), reused 0 (delta 0), pack-reused 515
Receiving objects: 100% (515/515), 208.91 KiB | 1.04 MiB/s, done.
Resolving deltas: 100% (319/319), done.
zolika@raspberrypi:~$ cd Adafruit_Python_GPIO
zolika@raspberrypi:~/Adafruit_Python_GPIO$ sudo python setup.py install
Adafruit GPIO Library
Works best with Python 2.7
THIS INSTALL SCRIPT MAY REQUIRE ROOT/ADMIN PERMISSIONS
Especially if you installed python for "all users" on Windows

try the following in your systems terminal if ensurepip is not sufficient:
$ python -m ensurepip --upgrade
$ python -m pip install --upgrade pip setuptools
running install
/usr/lib/python3/dist-packages/setuptools/command/install.py:34: SetuptoolsDeprecationWarning: setup.py install is deprecated. Use build and pip and other standa
  warnings.warn(
/usr/lib/python3/dist-packages/setuptools/command/easy_install.py:146: EasyInstallDeprecationWarning: easy_install command is deprecated. Use build and pip and o
  warnings.warn(
running bdist_egg
```

A szenzor bekötésének helyességét sem árt ellenőrizni ezen a ponton. Ezt az *i2cdetect -y 1* paranccsal tudjuk megtenni, hogy talál-e I2C eszközt. Amennyiben mindent jól csatlakoztattunk a megfelelő lábakhoz, a megjelenő táblázatban a 77-es címen látni fogjuk az eszközünket.

```
zolika@raspberrypi:~$ i2cdetect -y 1
    0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
```

Itt a projektfeladat megvalósításának következő fázisába lépünk, hiszen a méréseket el tudjuk kezdeni az előzőek után. Ehhez előbb be kell lépni a telepítési mappába és onnan elindítani a telepítés során felmásolt példa programot.

Ekkor újabb hiba lép fel:

```
zolika@raspberrypi:~/Adafruit_Python_BME280 $ python Adafruit_BME280_Example.py
File "/home/zolika/Adafruit_Python_BME280/Adafruit_BME280_Example.py", line 10
    print 'Temp      = {0:0.3f} deg C'.format(degrees)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
```

INTERNAL

Ezt egyszerűen lehet orvosolni a zárójelek kihelyezésével. Az operációs rendszer része a Geany fordító, amelyben megnyitva az Adafruit_BME280_Example.py fájlt, be tudjuk szerkeszteni a hiányzó zárójeleket.

```
Adafruit_BME280_Example.py x
1  from Adafruit_BME280 import *
2
3  sensor = BME280(t_mode=BME280_OSAMPLE_8, p_mode=BME280_OSAMPLE_8,
4
5  degrees = sensor.read_temperature()
6  pascals = sensor.read_pressure()
7  hectopascals = pascals / 100
8  humidity = sensor.read_humidity()
9
10 print ('Temp      = {0:0.3f} deg C'.format(degrees))
11 print ('Pressure  = {0:0.2f} hPa'.format(hectopascals))
12 print ('Humidity   = {0:0.2f} %'.format(humidity))
13
```

9. ábra Minta program javítása

A korrigálás után a terminálban újra futtatva a programot, ezúttal visszakapjuk a mért értékeket.

```
zolika@raspberrypi:~/Adafruit_Python_BME280 $ python Adafruit_BME280_Example.py
Temp      = 23.694 deg C
Pressure  = 1006.88 hPa
Humidity   = 31.13 %
```

10. ábra Szenzor által mért értékek

A minta programot tovább fejlesztve oly módon, hogy ne csak egy mérést hajtson végre, hanem valamilyen mérési sorozatot, illetve az így kapott értékeket egy fájlba mentse az alábbiak szerint valósítható meg.

Először egy LibreOffice alkalmazást telepítettem a következő paranccsal: **sudo apt install libreoffice**.

A következő lépésben pedig ezt a parancsot futtatjuk: **sudo apt install openpyxl**.

```
zolika@raspberrypi: ~
Fájl Szerkesztés Lapok Súgó
note: If you believe this is a mistake, please contact your Python installation or OS distribution provider. You can override this, at the risk of breaking your Python installation or OS, by passing --break-system-packages.
hint: See PEP 668 for the detailed specification.
zolika@raspberrypi:~$ sudo apt install python3-openpyxl
Csomaglisták olvasása... Kész
Függőségi fa építése... Kész
Állapotinformációk olvasása... Kész
A következő további csomagok lesznek telepítve:
python3-attr python3-et-xmlfile python3-iniconfig python3-jdcal
python3-packaging python3-pluggy python3-py python3-pytest
javasolt csomagok:
python-attr-doc subversion
Az alábbi új csomagok lesznek telepítve:
python3-attr python3-et-xmlfile python3-iniconfig python3-jdcal
python3-openpyxl python3-packaging python3-pluggy python3-py
python3-pytest
0 frissített, 9 újonnan telepített, 0 eltávolítandó és 0 nem frissített.
Letöltendő adatmennyiség: 627 kB.
A művelet után 3.151 kB lemezterület kerül felhasználásra.
Folytatni akarja? [Y/n]
```

11. ábra openpyxl telepítése

Az Openpyxl egy python könyvtár, amely lehetővé teszi adatok írását és olvasását Excel fájlokba/ból. Mindezt anélkül tudja megteremteni, hogy egy újabb alkalmazást ehhez telepítenünk kellene.

A telepítések után létrehoztam egy xlsx fájlt az Adafruit_Python_BME280 mappán belül és elneveztem 'weather.xlsx'-re. A munkafüzeten belül a munkalapot pedig átneveztem 'adatok'-ra.

A program szerkezete úgy fog kinézni, hogy a szükséges modulok importálása után az elején behívjuk az előzőleg létrehozott Excel fájlt, majd azon belül is definiáljuk, hogy melyik munkafüzetlapba dolgozzon (ha

csak egy van akkor nem szükséges deklarálni). Az utasítás ehhez:

wb= openpyxl.load_workbook('weather.xlsx') illetve **sheet = wb['adatok']**. Ezek után következhet a szenzor által mért adatok megjelenítése, célszerűen egy ciklusba ágyazva, hogy a mérések automatikusan megtörténjenek. A ciklusba időtagot is lehet tenni, hogy ezzel is a mérési frekvenciát szabályozni tudjuk. Az adatok Excel fájlba történő betöltését is formázhatjuk, illetve a végén a mentésről is gondoskodni kell. A mentéseket is még a cikluson belül kivitelezem, hogy minden körben frissüljenek az adatok.

Ciklusként a 'while True' szerkezetet alkalmazom, ezzel egy végtelenített körfolyamatot eredményezve. A 'while' önmagában nem teszi végtelenné a ciklust, hanem a 'True' szó az, amely ezt eredményezi, hiszen ez a logikai 'igen' mindig 'igen'-ként értékelődik ki.

A 'while True' ciklust a try...finally utasítás kombináción vagy blokkon belül használok, amely kivétel kezelést valósít meg. Tipikusan olyan feladatokhoz kézenfekvő használni ezt, amely során külső adatbázishoz csatlakozunk hálózaton keresztül vagy fájlal, GUI-val dolgozunk. GUI – Graphical User Interface. Ez a típusú megoldás gondoskodik arról, hogy a fájl bezáruljon még akkor is, ha a program végrehajtása során kivétel történik. A try...finally blokk finally része mindenképpen lefut.

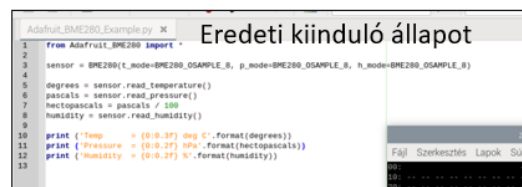
A 'while true' után helyezkedik el a program magja, ahol a mért adatok kijelzése, azok fájlba töltése és mentése van definiálva. Ezen felül egy időzítő elem is szerepel benne, ami lényegében meghatározott idő intervallumra szünetelteti a folyamatos ciklust. Ez a time.sleep() utasítás. A zárójeles részbe másodperc alapon értelmezi az értéket, amelyet megadunk, ezzel esetenben, ha például beleírjuk, hogy 60 akkor 60 másodpercig szünetel a program majd újból lefut a ciklus és ismételten megáll a futása 10 másodpercig. Így azt fogjuk látni a program elindítása után, hogy 60 másodpercenként kapunk egy újabb mérési eredményt. Ehhez a funkcióhoz az elején szükséges importálni a 'time' modult.

A komplett program:

```

Adafruit_BME280_Example_v4.py x live_graph.py x matplotlib_sample.py x graph.py x live_graph_v2.o
1 from Adafruit_BME280 import *
2 import datetime
3 import time
4 from datetime import date
5 import openpyxl
6 from openpyxl import load_workbook
7
8 sensor = BME280(t_mode=BME280_OSAMPLE_8, p_mode=BME280_OSAMPLE_8, h_mode=BME280_OSAMPLE_8)
9
10 # excel betöltése és a munkafüzet kiválasztása
11 wb = openpyxl.load_workbook('/home/zolika/Adafruit_Python_BME280/weather.xlsx')
12 sheet = wb['adatok']
13
14
15 try:
16     while True:
17
18         #adatok olvasása szenzorból és kijelzése, illetve az idő megjelenítése
19         degrees = round(sensor.read_temperature(),2)
20         pascals = round(sensor.read_pressure(),2)
21         dhectopascals = pascals / 100
22         humidity = round(sensor.read_humidity(),2)
23         today = date.today()
24         now = datetime.datetime.now().time()
25         formatted_time = now.strftime("%H:%M")
26
27         print (today)
28         print (formatted_time)
29         print ('Temp = {0:0.3f} deg C'.format(degrees))
30         print ('Pressure = {0:0.2f} hPa'.format(dhectopascals))
31         print ('Humidity = {0:0.2f} %'.format(humidity))
32
33         #adatok munkafüzetbe töltése
34         row = (formatted_time, degrees, humidity)
35         sheet.append(row)
36
37         #munkafüzet mentése
38         wb.save('/home/zolika/Adafruit_Python_BME280/weather.xlsx')
39
40         #időzítés
41         time.sleep(60)
42
43 finally:

```



12. ábra Komplett program ciklikusan mért adatok tárolásához

Az idő kijelzésénél további módosítást eszközöltem mert a másodperc kijelzést feleslegesnek tartottam, illetve rengeteg tizedesjegyet kiírt utána. Utasítás: ***formatted_time=now.strftime(„%H:%M”)***

```

zolika@raspberrypi:~/Adafruit_Python_BME280 $ python3
py
2023-10-21          eredeti állapot
10:43:38.962856
Temp      = 22.860 deg C
Pressure  = 975.84 hPa
Humidity   = 52.58 %

2023-10-21          módosított állapot
10:37
Temp      = 23.110 deg C
Pressure  = 975.88 hPa
Humidity   = 52.96 %
    
```

13. ábra Idő kijelzése

A mért értékeket is kerekítettem két tizedesjegyre **round()** függvénnyel, mert az alábbi módon kerültek mentésre a munkafüzetben:

Time	temperature	Pressure (hPa)	Humidity (%age)
11:35:22	23.63901743975	989.1324863345	41.607358135806
11:35:52	23.60110297704	989.1340742846	41.556145830401
11:36:22	23.57772239987	989.1018648595	41.295111245649
11:36:52	23.50694879879	989.10229183	41.266502785208
11:37:23	23.478512993	989.0937303342	41.515944248019
11:37:53	23.51579549581	989.0877791885	41.657882887264

```

degrees = round(sensor.read_temperature(),2)
pascals = round(sensor.read_pressure(),2)
dhectopascals = pascals / 100
humidity = round(sensor.read_humidity(),2)
    
```

14. ábra Értékek kerekítése két tizedesjegyre round() függvénnyel

Mindezek után a projektfeladat kidolgozásának utolsó fázisa következik, azaz a mért értékek valós időben történő megjelenítése grafikon segítségével. Raspberry használatával relatív egyszerű beszúrni egy html felületre egy matplotlib diagramot. A matplotlib egy, a Python programozási nyelvhez és annak numerikus matematikai kiterjesztéséhez, a NumPy-hoz írt ábrázoló könyvtár. A matplotlib segédprogramja kifejezetten a vizuális ábrázoló eszközökhöz kapcsolódik. Ezt a ***sudo apt-get install python3-matplotlib*** paranccsal tudjuk feltelepíteni. Az adatok kezeléséhez szükséges még az úgynevezett panda modult feltenni, amely egy nyílt forráskódú szoftverkönyvtár. Az adatkezelés, elemzés terén tud segítséget nyújtani. Jól használhatók különböző típusú adatokhoz, mint például táblázatokhoz, amelyek heterogénen tipizált oszlopokkal rendelkeznek vagy rendezett és rendezetlen idősoros adatokhoz is. Telepíteni a ***sudo apt-get install python3-pandas*** utasítással tudjuk.

```

Fájl Szerkesztés Lapok Súgó
zolika@raspberrypi:~ $ sudo apt-get install python3-pandas
Csomaglisták olvasása... Kész
Függőségi fa építése... Kész
Állapotinformációk olvasása... Kész
A következő további csomagok lesznek telepítve:
libaec0 libblosc1 libhdf5-103-1 libsz2 python-odf-doc python-odf-tools
python-tables-data python3-bottleneck python3-defusedxml python3-numexpr
python3-odf python3-pandas-lib python3-tables python3-tables-lib
Ajavaslott csomagok:
python-bottleneck-doc python-pandas-doc python3-statsmodels python3-netcdf4
python-tables-doc vitables
Az alábbi új csomagok lesznek telepítve:
libaec0 libblosc1 libhdf5-103-1 libsz2 python-odf-doc python-odf-tools
python-tables-data python3-bottleneck python3-defusedxml python3-numexpr
python3-odf python3-pandas python3-pandas-lib python3-tables
python3-tables-lib
0 frissített, 15 újonnan telepített, 0 eltávolítandó és 0 nem frissített.
Letöltendő adatmennyiség: 8.592 kB.
A művelet után 51,1 MB lemezterület kerül felhasználásra.
Folytatni akarja? [I/n] i
    
```

15. ábra Panda modul telepítése

Biztonsági okok miatt további modult is telepítettem, ez pedig az xlrd. Az xlrd segítségével adatokat tudunk kiolvasni Excel fájlokból, hozzáférésünk lesz a munkalapokhoz, sorokhoz, oszlopokhoz vagy cellákhoz. Mindezt anélkül tudja, hogy telepítenénk akár egy MS Officet. Az én esetemben ez nem áll fent, de installálom inkább. Parancs: ***sudo apt-get install python3-xlrd***

Majd a programozás része következett, amelyhez különböző internetes forrásokból igyekeztem hasznos információkat gyűjteni, illetve megjeleníteni a diagramot. Több változatot találtam és próbáltam ki majd készítettem egy egyszerű minta.xlsx fájlt ahol 3 oszlop szerepelt csupán pár adattal (1 X tengely és 2 Y) és azt kíséreltem megjeleníteni. Mindezt azért csináltam mert bár a diagram felület betöltődött, ám az üresen maradt minden esetben, mintha nem tudná értelmezni a kapott adatokat. Ez akkor nyert bizonyosságot amikor az alábbi programmal sikerült a minta.xlsx adatait megjelenítenem, viszont a weather.xlsx-el üresen maradt ismét a html felület. Ezen a ponton az adatok bevitelén oly módon módosítottam, hogy a korábbi 5 oszlopos adattárolás (date, time, temperature, pressure, humidity) helyett 3 oszlopos megoldást valósítottam meg.

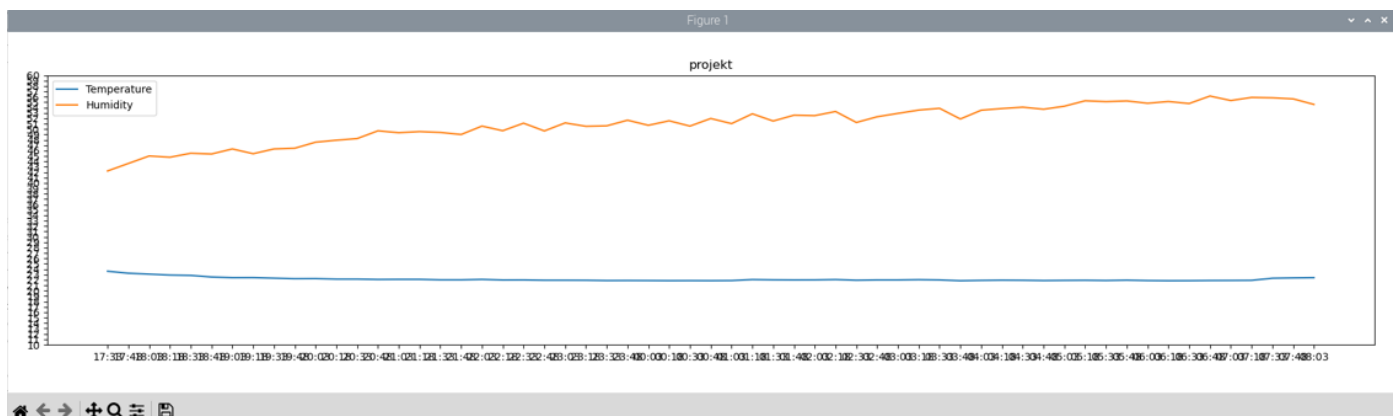
```
#adatok munkafuzetbe toltese
row = (formatted_time, degrees, humidity)
sheet.append(row)
```

← 3 adatoszlop

↑
adatok sorba töltése

16. ábra Adatok munkafüzetbe töltése (3 oszlopba)

Ezután már sikerült a diagramot tökéletesen megjelenítenem:



17. ábra Megjelenített diagram

Az 'X' tengely feliratait összefolynak kevés mintavételezés után, ezért azt elforgattam 90°-al. A bal felső sarokba került a jelmagyarázat, míg az 'Y' tengelyen től-ig határt állítottam be mert szükségtelen lett volna 0-tól 100-ig skálázni klimatizált irodai körülmények között.

Az egész program, amely a vizualizációt szolgálja:

```

Adafruit_BME280_Example_v4.py x live_graph.py x matplotlib_sample.py x graph.py x live_graph_v2_ok.py x
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import time
4
5
6 while True:
7     #kiolvasni az adatokat
8     updated_data = pd.read_excel('/home/zolika/Adafruit_Python_BME280/weather.xlsx')
9
10    #torolni az elozo diagramot (mindig uj felületre dolgozzon)
11    plt.clf()
12
13    #adatsorok elnevezese
14    x = updated_data['time']
15    y1 = updated_data['temperature']
16    y2 = updated_data['humidity']
17
18    #abrazolni az adatokat
19    plt.plot(x, y1, label = 'Temperature')
20    plt.plot(x, y2, label = 'Humidity')
21    plt.title('Projektfeladat')
22
23    # 'Y' tengely felosztas
24    plt.ylim(15,60)
25    plt.yticks(range(15, 60 +1, 1))
26
27    plt.legend(loc='upper left')
28    plt.xticks(rotation=90)
29
30    #adatok frissitese (x) masodpercenkent
31    plt.pause(10)
32
33
34

```

18. ábra Program az adatok diagramban történő megjelenítéséhez

Itt is a szükséges modulok meghívásával kezdődik a program. Jól látszik, hogy az xlrd végül kimaradt. A matplotlib és panda modulokon kívül a time-ot is importáltam, hogy hasonló módon a másik programhoz, ezen belül is ezzel adjak egy meghatározott másodperc alapú frekvenciát az adatok frissítéséhez.

'While true' ciklussal kezdődik a tényleges program, így ez is végtelenített azaz addig fut, amíg meg nem szakítjuk.

Az adatok kiolvasását panda 'dataframe'-be a **pd.read_excel()** függvénnyel valósul meg. Ehhez a zárójelek közé szükséges a pontos elérési utat megadni. Így meghívva a függvényt az összes adatot olvasni fogja, beleértve az oszlop elnevezéseket is. Updated_data elnevezést adtam neki.

Minden ciklusban törlésre kerül a grafikon (üres felületünk lesz), ezzel elkerülve, hogy több vonal egymással fedésbe kerüljön. Az utasítás, ami ezt végzi: **plt.clf()** - a matplotlib része.

A táblázatban lévő oszlopokat is elneveztem x,y1,y2 néven. Ehhez elegendő beírni pl.:

x=updated_data['time'] így a time oszlopban lévő adatok felelnek meg az x-nek a továbbiakban.

Az adatok ábrázolásához a **plt.plot()** függvényt kell alkalmazni. Zárójelbe az megjeleníteni kívánt adatsorok azonosítóit kell beírni. Tovább fűzve a parancsot, az y paramétereket elláttam címkékkel is, mint 'temperature' és 'humidity'. A **plt.title()** mondattal a diagramnak adunk címet.

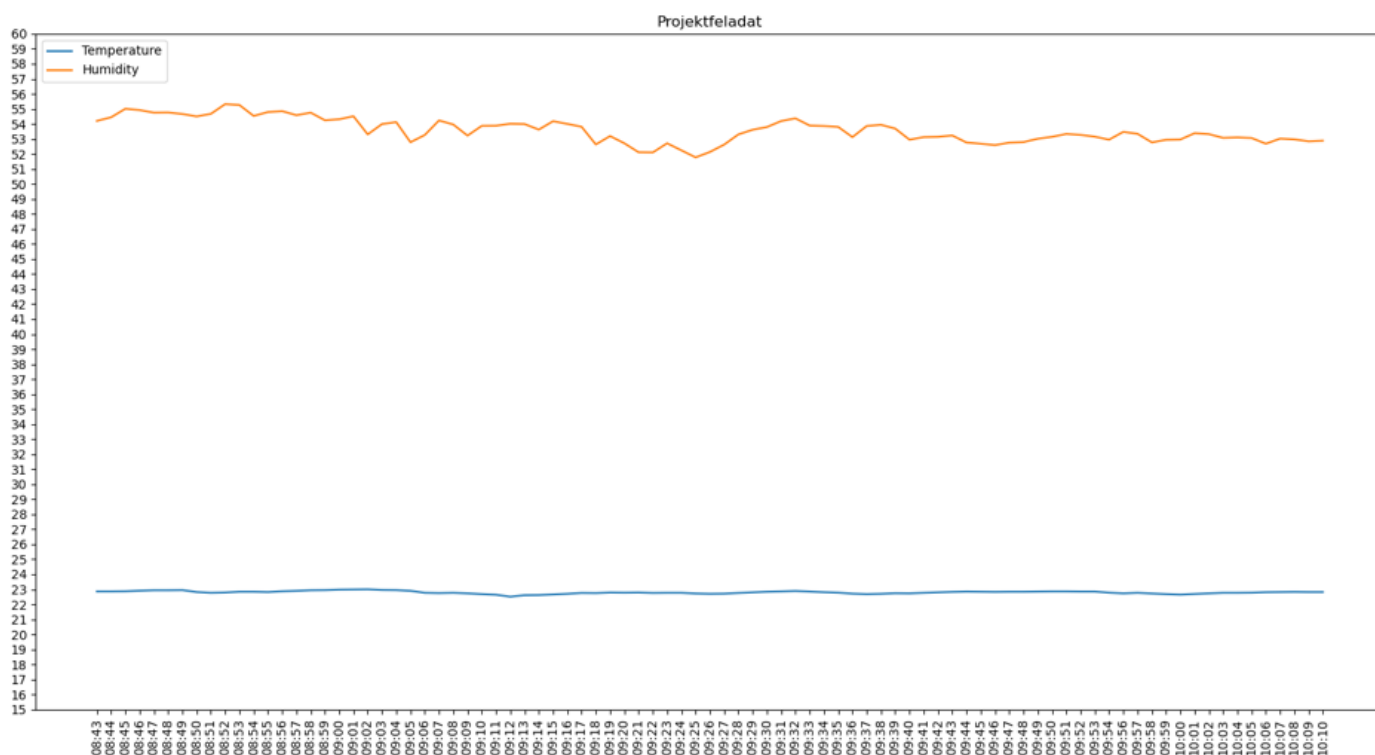
Az 'Y' tengelyt szükségesnek láttam formázni. Ez annyit takar, hogy tól-ig értéket adtam meg a skálázáshoz a **plt.ylim(min,max)** paranccsal, illetve a léptéket vagy felbontást beállítottam, hogy egyesével növekedjen. Ezt a **range** utáni zárójel párban definiálhatjuk értelemszerűen (min, max, lépték).

Nem utolsó sorban a jelmagyarázatot is elhelyezem a bal felső sarokba. Ehhez a **plt.legend()**-et kell beírni. Külön lehet még operálni a loc-al, mint location. Látható, hogy a loc='upper left' utasítással a diagram felületének bal felső részébe helyezi a téglalapot.

Az 'X' tengely feliratait elforgatjuk 90°-al az optimálisabb helykihasználás végett: **plt.xticks(rotation=90)**

Végezetül a frissítés gyakoriságát határozzuk meg úgy, hogy bizonyos hosszúságú szüneteket teszünk minden egyes ciklusba. A **plt.pause()** a matplotlib könyvtár pyplot modulja (ezt is importáltuk a program legelején), amellyel a beírt értéknek megfelelő ideig szünetelhetjük az ábrázolást.

A kész diagram a következőképp néz ki:



19. ábra Mért értékek valós idejű ábrázolása