

Mikroelektromechanikai rendszerek beadandó

**Rátkai Róbert
(YXLG5V)**

Mérnök Informatikus BSc – Levelező

2022/23/1





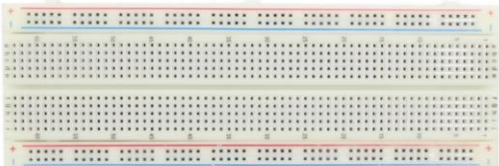
Tartalom

1. A feladat leírása	2
2. Használt eszközök	2
3. Felépítés	3
3.1. Kapcsolási rajz	3
3.2. Megvalósított megoldás	3
4. Konfiguráció	4
4.1. Interfészek bekapcsolása	4
4.2. W1-GPIO - One-Wire Interface beállítás.....	4
4.3. Redis adatbázis telepítése.....	4
5. Fejlesztői dokumentáció	5
5.1. Komponensek.....	5
5.1.1. Vezérlés (main.py).....	5
5.1.2. Webalkalmazás (api.py)	7
5.2. Weboldal	8
5.2.1. Weboldal (index.html).....	8
5.2.2. Diagram (Chart.js)	9
5.2.3. Példa diagram:.....	9
6. Hasznos linkek	9

1. A feladat leírása

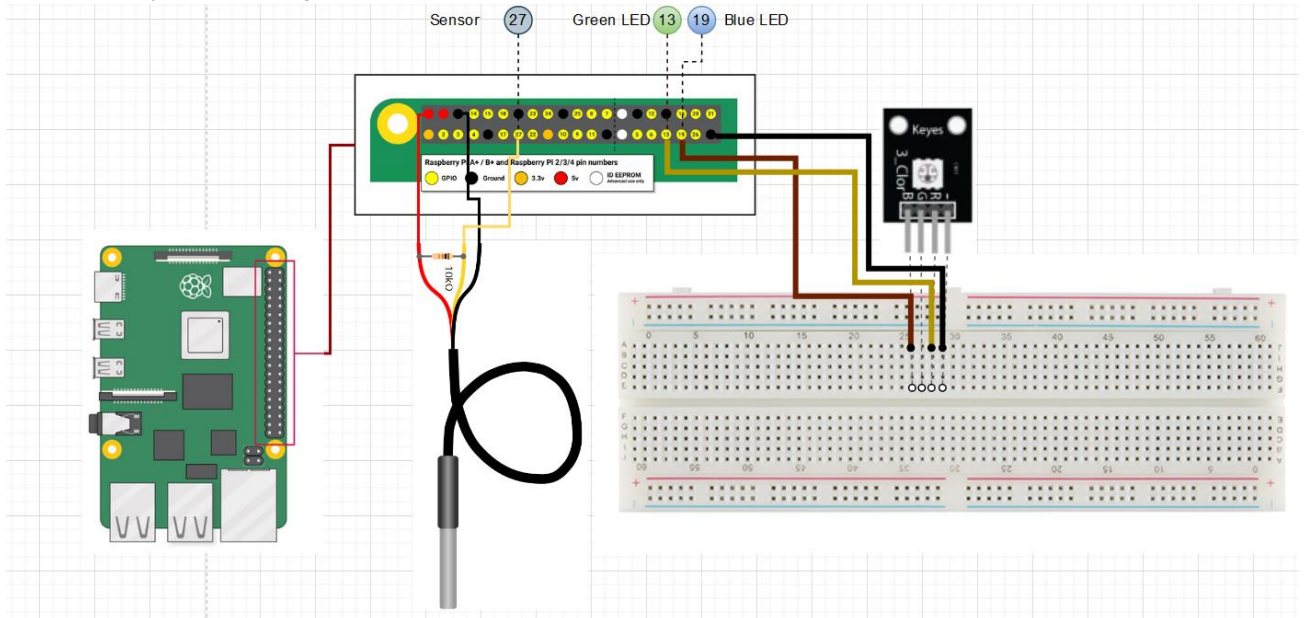
Egy Raspberry Pi-re kötünk egy szenzort – ami a hőmérsékletet méri- és egy 3 színű ledet (Piros, Zöld, Kék), ami a következők alapján ad visszajelzést. Ha a mért hőmérséklet nem éri el a 25 C°-ot kék led világít, ha meghaladja a mért hőmérséklet a 25 C°-ot a zöld led világít. A vezérlés a Raspberry Pi-n futtatott Python kód segítségével történik.

2. Használt eszközök

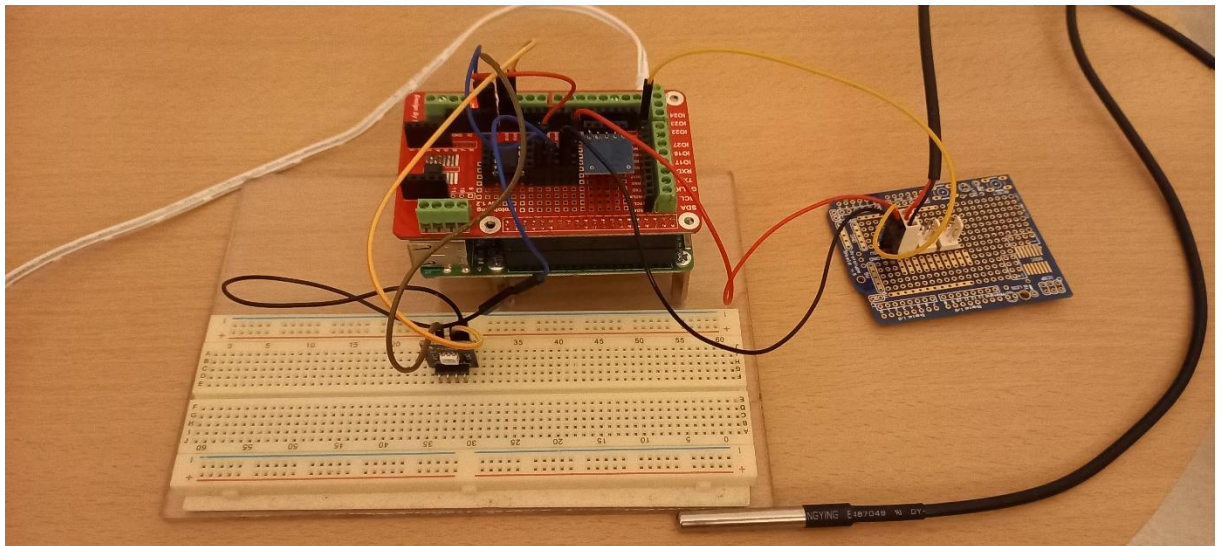
Megnevezés	Kép
<u>Raspberry pi: Raspberry Pi 3 Model B Rev 1.2</u>	
<u>RaspbPrototype Shield Expansion Board For Raspberry Pi B</u>	
<u>DS18B20 Temperature Sensor</u>	
<u>3 Color RGB SMD LED Module</u>	
Breadboard	

3. Felépítés

3.1. Kapcsolási rajz

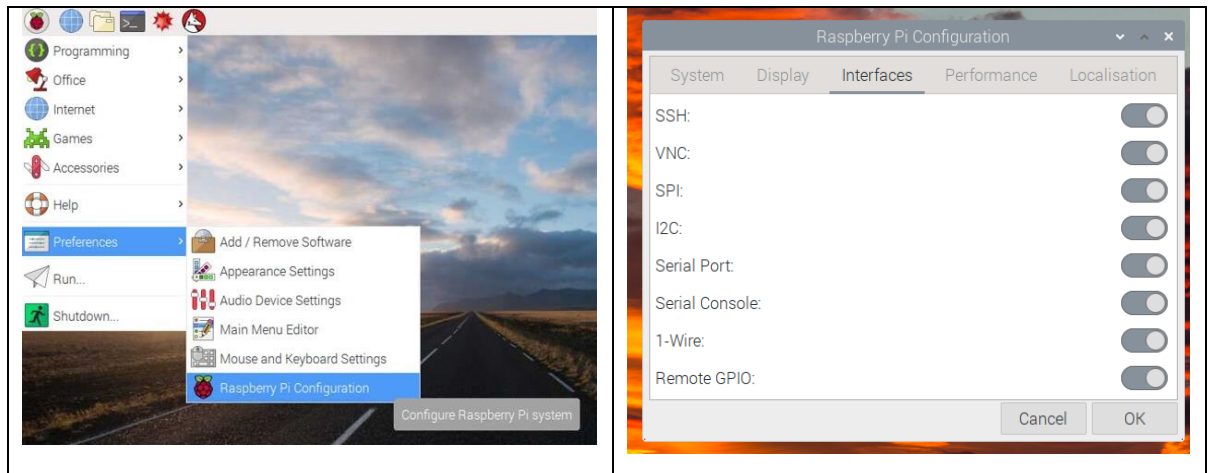


3.2. Megvalósított megoldás

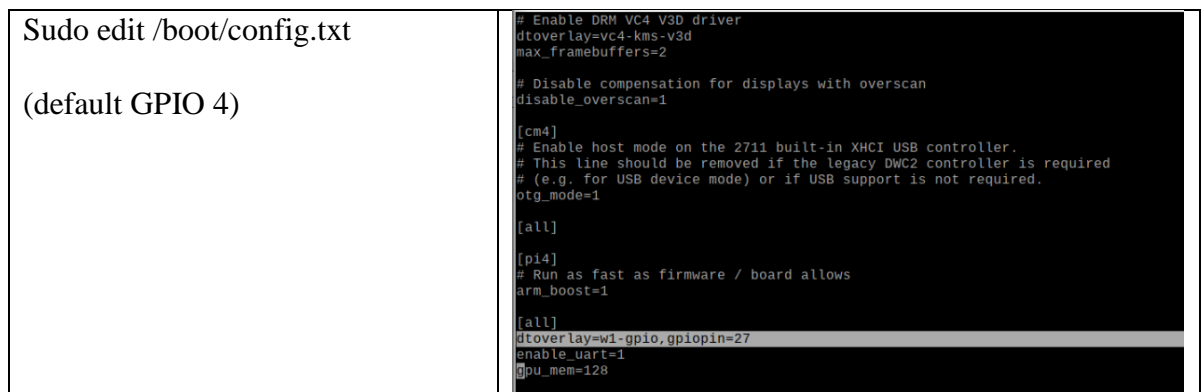


4. Konfiguráció

4.1. Interfészek bekapcsolása



4.2. W1-GPIO - One-Wire Interface beállítás

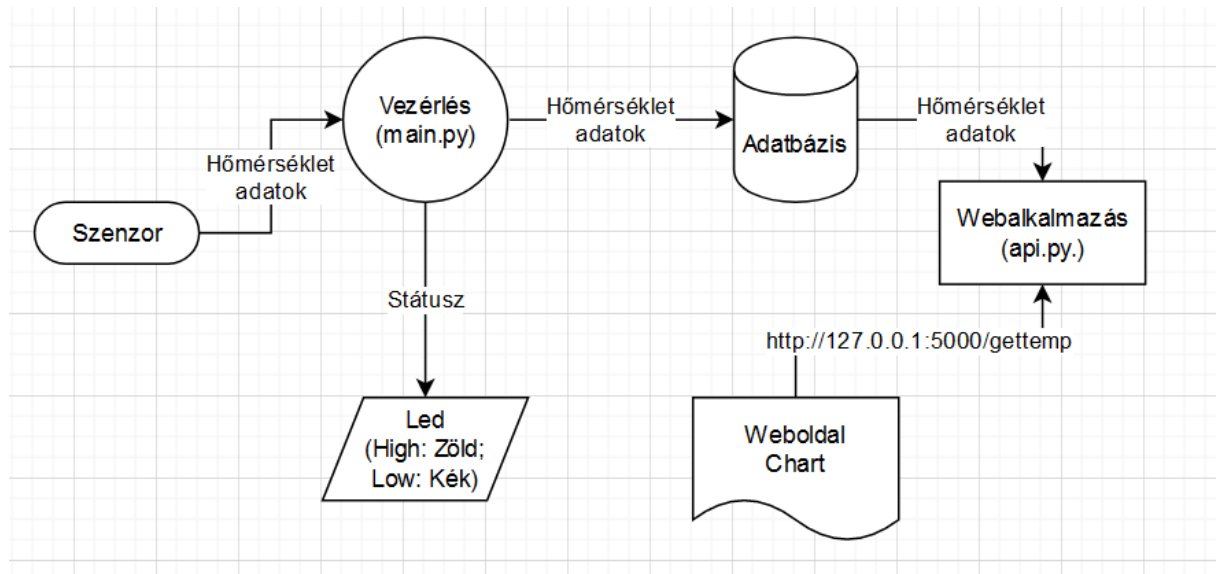


4.3. Redis adatbázis telepítése

sudo apt update – a csomaglista frissítése
sudo apt install redis-server – redis szerver telepítése
sudo systemctl enable redis – redis szolgáltatás bekapcsolása
sudo systemctl start redis – redis szolgáltatás elindítása

A konfigurációs állomány itt található: /etc/redis/redis.conf
Alapértelmezetten a localhoston fut a 6379-es porton. A használatához további konfigurációra nincs szükség, de a conf fájlban részletes útmutató található a finomhangoláshoz.

5. Fejlesztői dokumentáció



5.1. Komponensek

5.1.1. Vezérlés (main.py)

Feladata: Másodpercenként lekérdezi a szenzor által mért adatokat. Kiértékeli és az eredményt az adatbázisba menti. Amennyiben státusz változás történt (25 C° fölé vagy alá ment a hőmérséklet) a led színét is megváltoztatja. (Kék <25 C° > Zöld)

Modulok:

```
import os
import glob
import time
import RPi.GPIO as GPIO
import redis
```

Változók:

```
BLUE_LED = 19
GREEN_LED = 13
CHANGE = True
DATA = {"celsius": 0, "fahrenheit": 0, "status": "LOW/HIGH"}
stream_name = 'mystream'
base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'
db = redis.Redis("localhost")
```

Beállítások:

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(BLUE_LED,GPIO.OUT)
GPIO.setup(GREEN_LED,GPIO.OUT)
os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')
```

Függvények:

```
def read_temp_raw():
    f = open(device_file, 'r')
    lines = f.readlines()
    f.close()
    return lines

def read_temp():
    lines = read_temp_raw()
    while lines[0].strip()[-3:] != 'YES':
        time.sleep(0.2)
        lines = read_temp_raw()
    equals_pos = lines[1].find('t=')
    if equals_pos != -1:
        temp_string = lines[1][equals_pos+2:]
        temp_c = float(temp_string) / 1000.0
        temp_f = temp_c * 9.0 / 5.0 + 32.0
    return temp_c, temp_f
```

Vezérlés:

```
while True:
    time.sleep(1)
    DATA["celsius"],DATA["fahrenheit"] = read_temp()
    if DATA["celsius"] < 25:
        if DATA["status"] != "LOW":
            DATA["status"] = "LOW"
            CHANGE = True
    else:
        if DATA["status"] != "HIGH":
            DATA["status"] = "HIGH"
            CHANGE = True
    db.xadd(stream_name, DATA, id='*')
    print("Temperature: ",DATA["celsius"], "C° Status: ",DATA["status"])
    if CHANGE:
        print("STATUS has changed to ",DATA["status"],"!")
        CHANGE = False
        if DATA["status"] == "HIGH":
            GPIO.output(BLUE_LED, GPIO.LOW)
            GPIO.output(GREEN_LED, GPIO.HIGH)
        else:
            GPIO.output(BLUE_LED, GPIO.HIGH)
            GPIO.output(GREEN_LED, GPIO.LOW)
```

5.1.2. Webalkalmazás (api.py)

Feladata: Segítségével az utoljára mért 10 adatot le lehet kérdezni az adatbázisból.

Modulok

```
from flask import Flask, json
from flask_cors import CORS, cross_origin
import redis
```

Változók

```
db = redis.Redis("localhost")
DATA = [0, {"celsius": 0, "fahrenheit": 0, "status": "LOW/HIGH"}]
stream_name="mystream"

api = Flask(__name__)
cors = CORS(api)
api.config['CORS_HEADERS'] = 'Content-Type'
```

API definiálás és indítás

```
@api.route('/gettemp', methods=['GET'])
@cross_origin()
def get_temp():
    DATA = db.xrevrange(stream_name, "+", "-", count=10)
    senddata = []
    for x in reversed(DATA):
        x = json.dumps(x[1])
        x = json.loads(x)
        senddata.append(x["celsius"])
    print(json.dumps(senddata))
    return json.dumps(senddata)

if __name__ == '__main__':
    api.run()
```


5.2. Weboldal

5.2.1. Weboldal (index.html)

Feladata: a „GET” gombra kattintva lekérdezi az utolsó 10 adatot, megjeleníti egy diagramon és kiírja az értékeket.

```
<!DOCTYPE html>
<html>
<script src="Chart.js"></script>
<body>

    <canvas id="myChart" style="width:100%;max-width:600px"></canvas>

    <script>
        function dochart(xValues, yValues) {
            new Chart("myChart", {
                type: "line",
                data: {
                    labels: xValues,
                    datasets: [{
                        fill: false,
                        lineTension: 0,
                        backgroundColor: "rgba(0,0,255,1.0)",
                        borderColor: "rgba(0,0,255,0.1)",
                        data: yValues
                    }]
                },
                options: {
                    legend: {display: false},
                    scales: {
                        yAxes: [{ticks: {min: 20, max:30}}],
                    }
                }
            });
        }

        var x = [1,2,3,4,5,6,7,8,9,10];
        var y = [20,21,22,23,24,25,26,27,28,29,30];
        dochart(x, y);

    </script>

    <h1>Get Temperature</h1>

    <script type="text/javascript">
        function gettemp() {
            var apiUrl = 'http://127.0.0.1:5000/gettemp';
            fetch(apiUrl).then(response => {
                return response.json();
            }).then(tempdata => {
                document.getElementById('tempdata').innerHTML = JSON.stringify(tempdata);
                var x = [1,2,3,4,5,6,7,8,9,10];
                var y = tempdata;
                dochart(x, y);
            }).catch(err => {
                console.log("Something went wrong!");
            });
        }
    </script>

    <button onclick="gettemp()">GET</button>

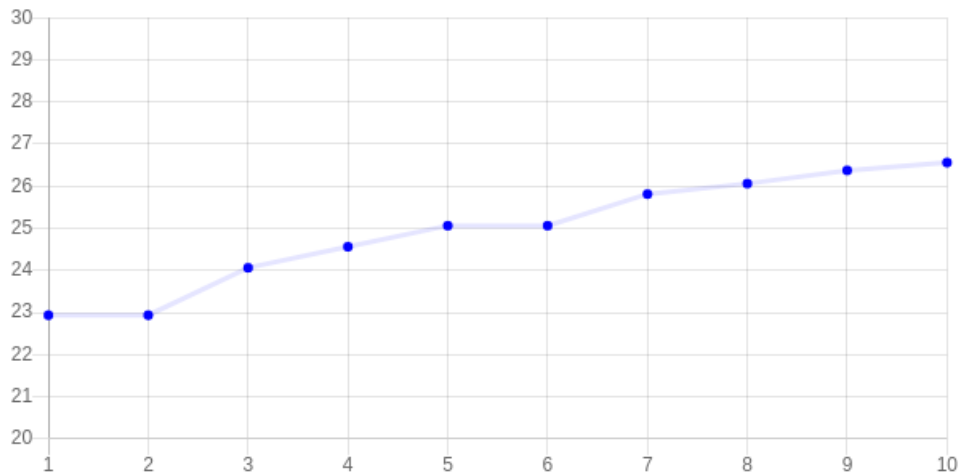
    <p>Results</p>
    <div id="tempdata"></div>

</body>
</html>
```

5.2.2. Diagram (Chart.js)

Feladata: A diagram megjelenítésért felelős. [Innen tölthető le.](#)

5.2.3. Példa diagram:



Get Temperature

GET

Results

["22.937","22.937","24.062","24.562","25.062","25.062","25.812","26.062","26.375","26.562"]

6. Hasznos linkek

[Raspberry Pi Tutorial Series: 1-Wire DS18B20 Sensor](#)

[Raspberry Pi pinout](#)

redis:

<https://pimylifeup.com/raspberry-pi-redis/>

<https://redis.io/docs/>

Led: <https://microdaz.com/ky-009-rgb-led-smd-module/>

Chart.js: <https://www.chartjs.org/>