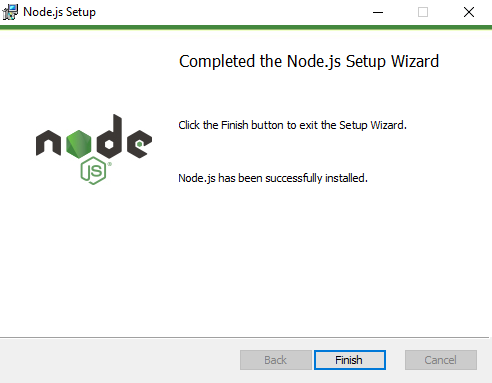
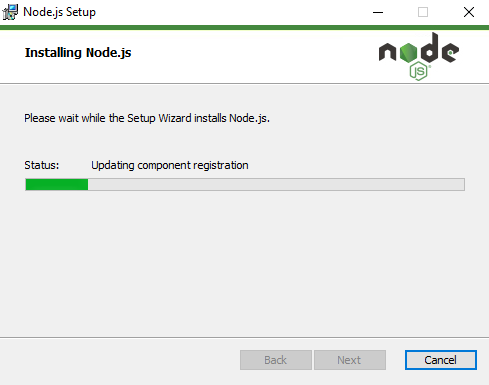
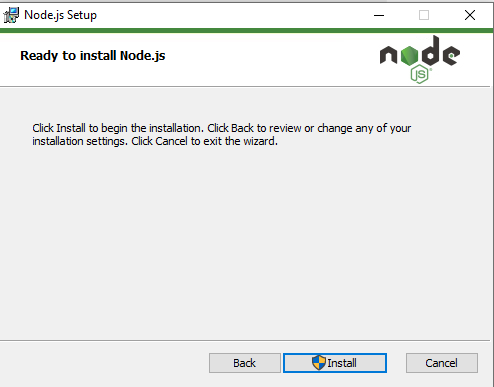
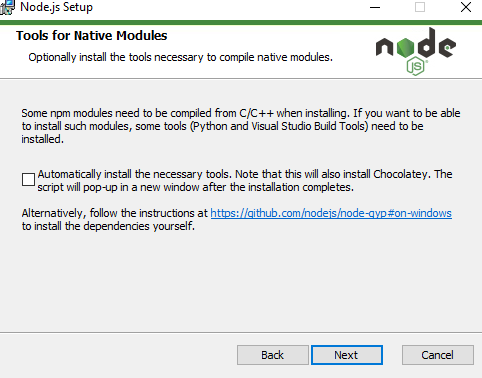
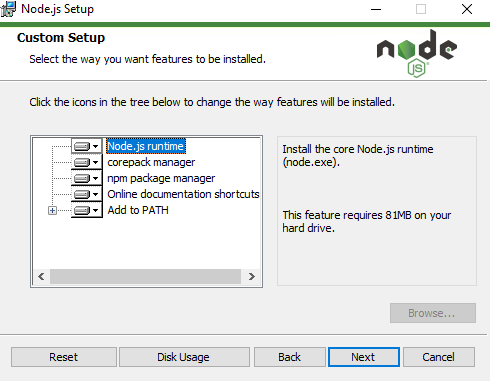
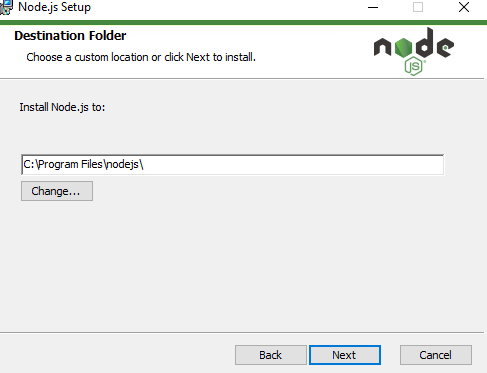
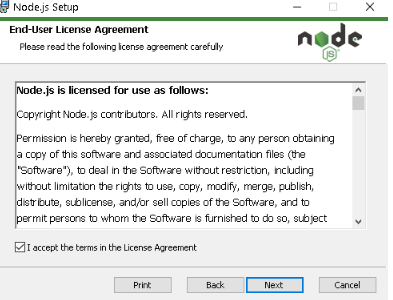
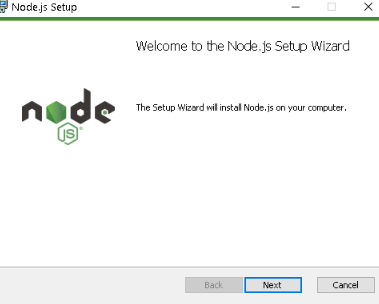
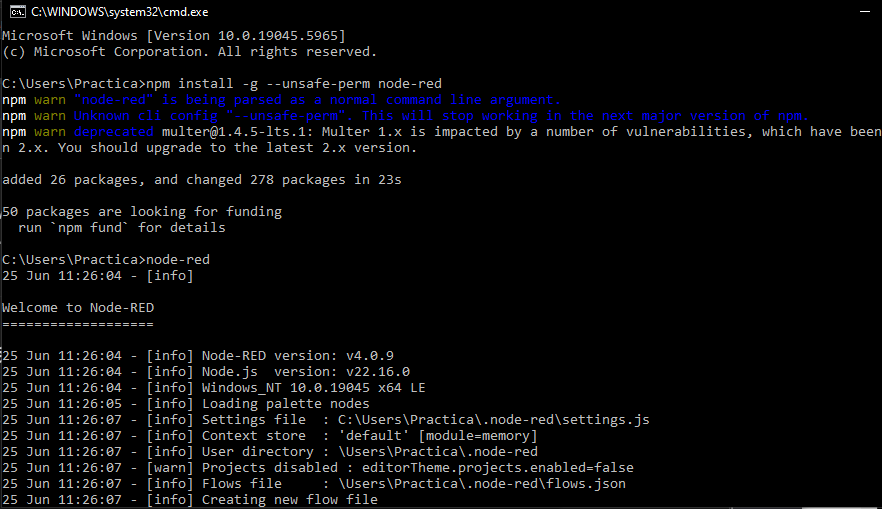
Pre-Practica Project Documentation

Node-RED

To access Node-RED, I had to install Node.js manually from<https://nodejs.org/en/download> and chose to install it in the **C:** drive.

During the custom setup steps, I made sure to check the **"Node.js runtime"** option, and I did **not** select the **Chocolatey** option.





### PHYTON

I developed a Python script that simulates sensor data by generating random humidity and temperature values.

import paho.mqtt.client as mqtt

import time

import random

import json

import os

broker = "mqtt.beia-telemetrie.ro"

port = 1883

topic = "training/device/ionut-barbalan"

client\_id = "simulator-ionut-barbalan"

backup\_file = "backup\_ionut.json"

main\_log\_file = "ionut-barbalan.txt"

client = mqtt.Client(client\_id=client\_id, protocol=mqtt.MQTTv311, transport="tcp")

client\_connected = False

def on\_connect(client, userdata, flags, rc):

global client\_connected

if rc == 0:

print("Conectat la broker MQTT")

client\_connected = True

else:

print("Conectare eșuată. Cod:", rc)

client.on\_connect = on\_connect

client.connect(broker, port)

client.loop\_start()

def save\_to\_backup(payload):

with open(backup\_file, "a") as f:

f.write(payload + "\n")

def resend\_backup\_messages():

if not os.path.exists(backup\_file):

return

print("🔁 Resincronizare cu brokerul...")

with open(backup\_file, "r") as f:

lines = f.readlines()

success\_lines = []

for line in lines:

line = line.strip()

if line:

result = client.publish(topic, line)

if result.rc == 0:

print(f"Resincronizat: {line}")

success\_lines.append(line)

with open(main\_log\_file, "a") as log:

log.write(line + "\n")

with open(backup\_file, "w") as f:

for line in lines:

if line.strip() not in success\_lines:

f.write(line)

while not client\_connected:

print("Aștept conectarea...")

time.sleep(2)

resend\_backup\_messages()

while True:

data = {

"temperature": round(random.uniform(20.0, 30.0), 2),

"humidity": round(random.uniform(40.0, 60.0), 2)

}

payload = json.dumps(data)

result = client.publish(topic, payload)

if result.rc == 0:

print(f"Trimis: {payload}")

with open(main\_log\_file, "a") as f:

f.write(payload + "\n")

else:

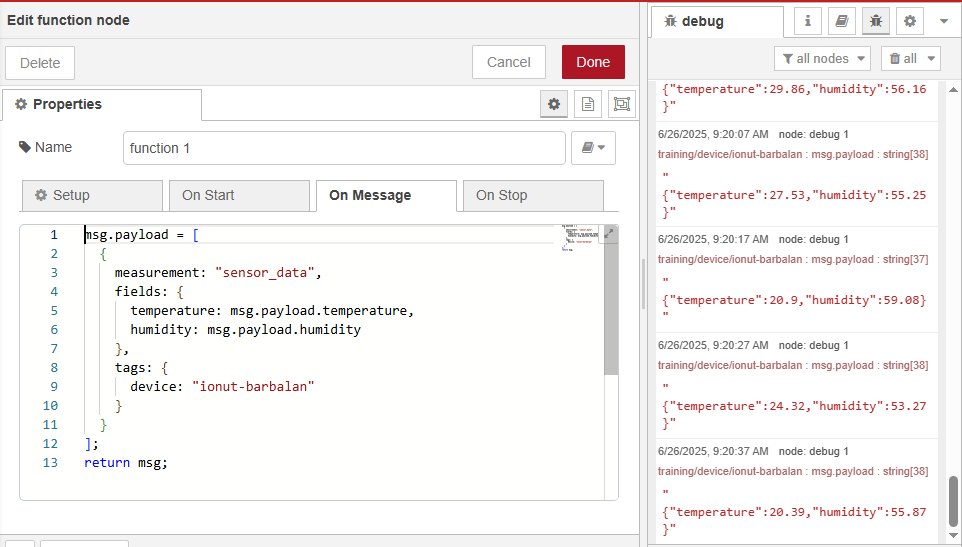
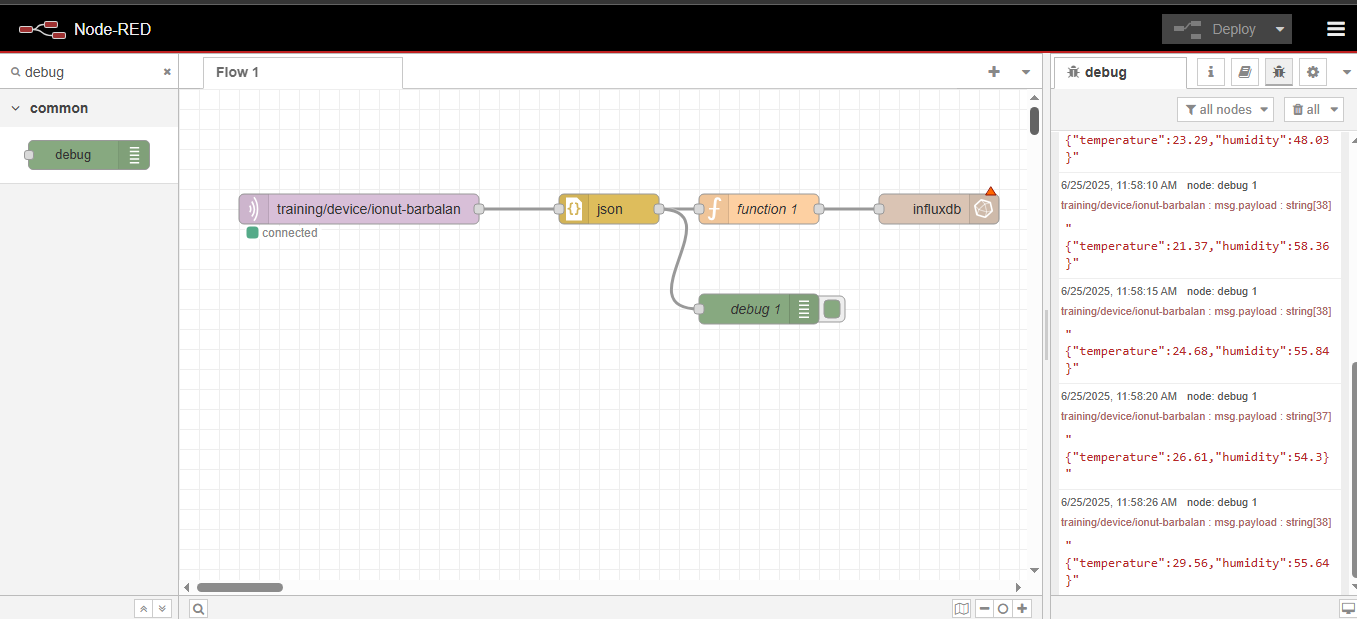
print("Eroare la trimitere. Salvez local...")

save\_to\_backup(payload)

time.sleep(10)

### NODE-RED

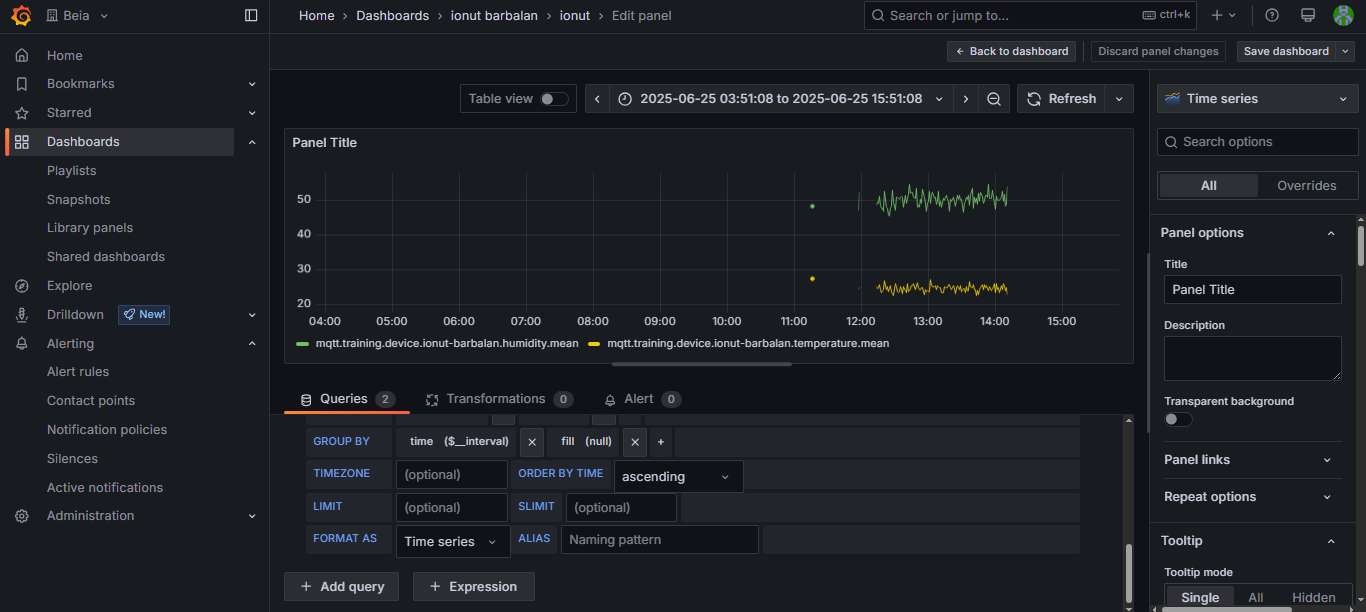
Node-RED is used to retrieve data from the MQTT broker, process it, save it locally, and send it to an InfluxDB database



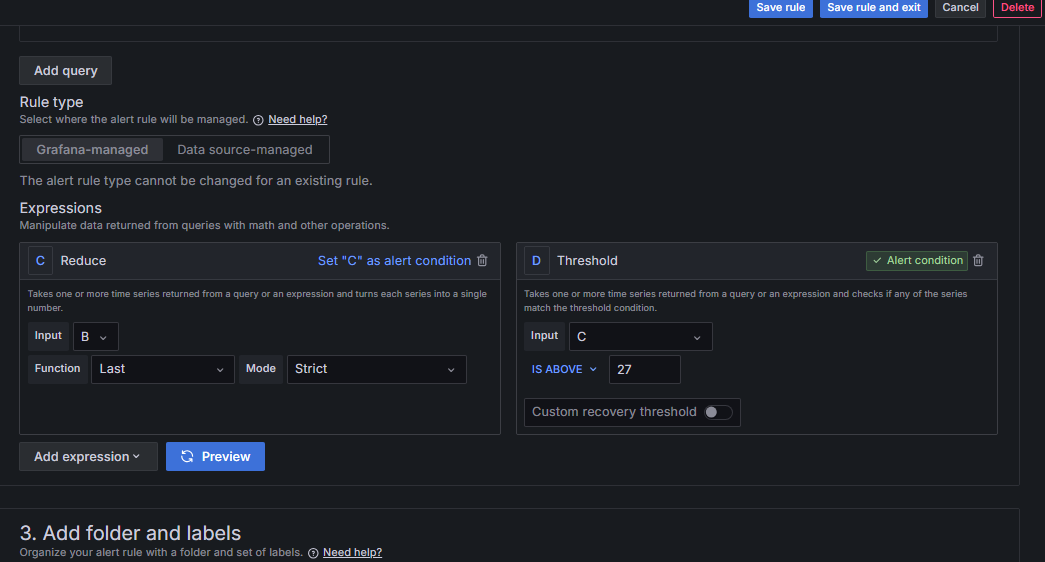
### GRAFANA

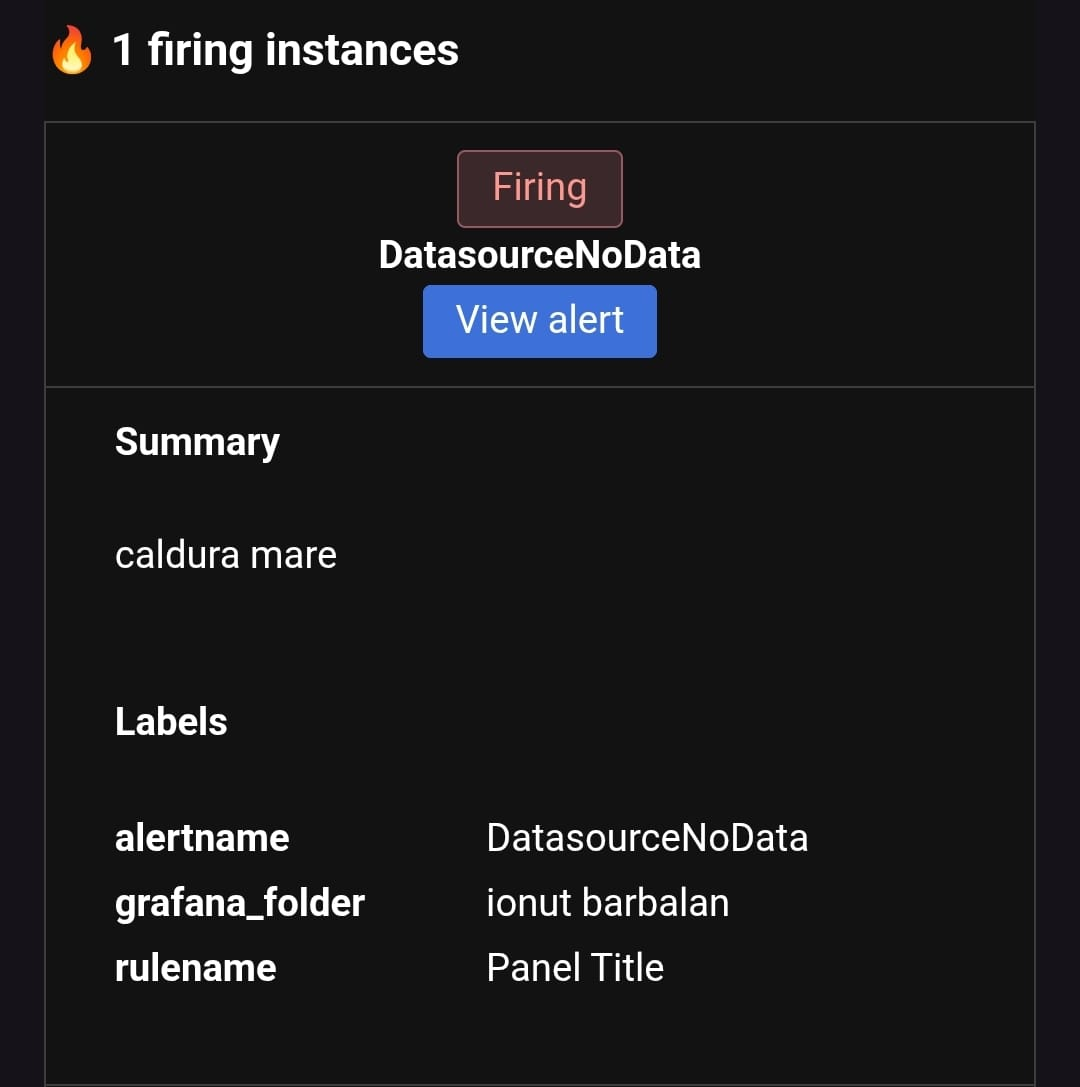
I set up Grafana to visualize the sensor data stored in InfluxDB. I configured dashboards that display live and historical temperature and humidity readings.

The graph for temperature and humidity in Grafana



I set up an email alert in Grafana that is triggered when the temperature rises above 27°C.

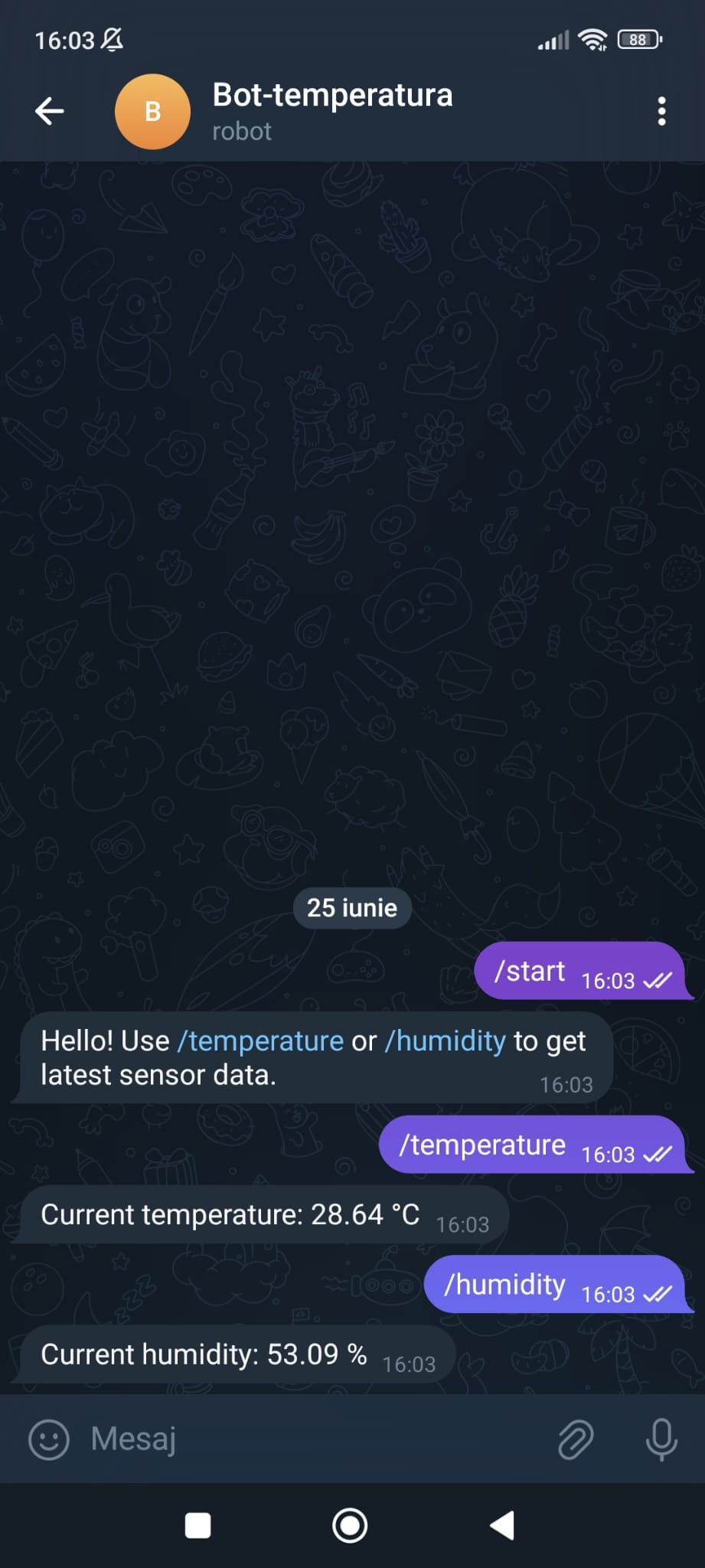




### CHATBOT

I created a Telegram chatbot that shows the current temperature and humidity.The bot displays the most recent temperature and humidity value recorded.

Telegram bot:<https://telegram.me/BotFather>

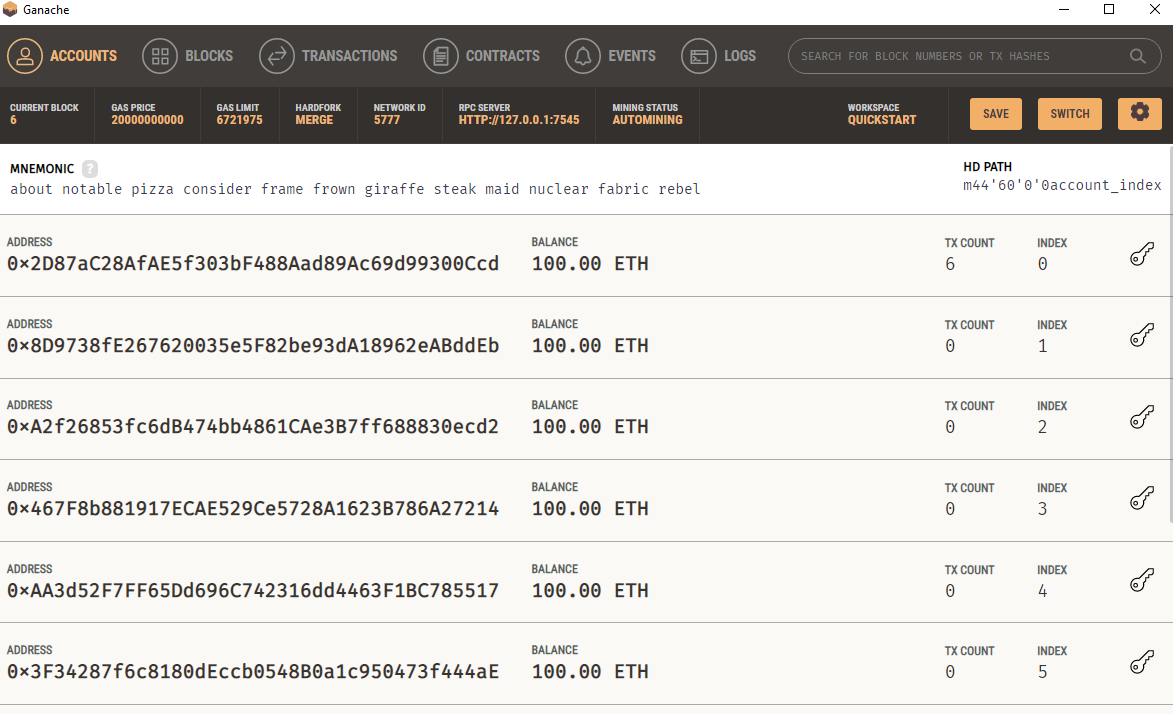


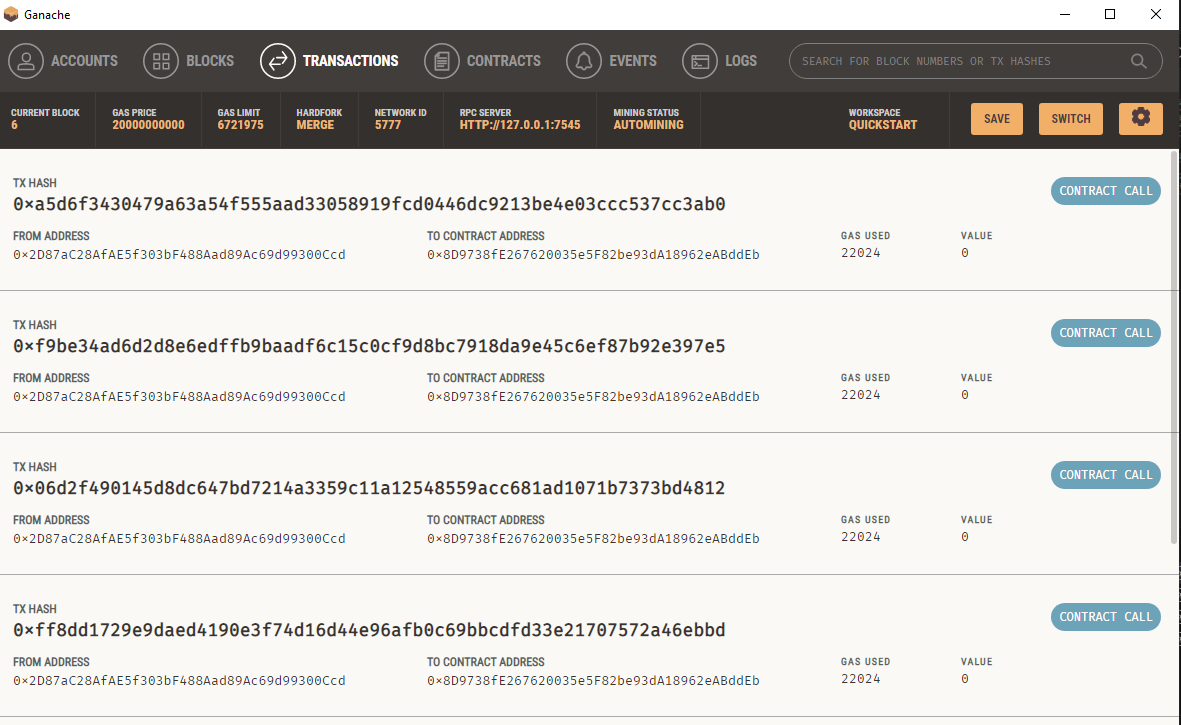
### BLOCKCHAIN

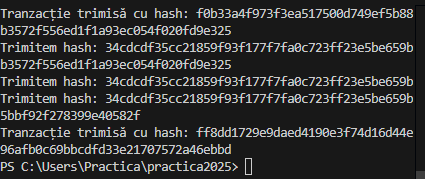
Blockchain is used in this project to securely store and verify IoT data, such as temperature and humidity readings. By recording a hash of each data entry on the blockchain, we ensure the integrity and immutability of the information, making it tamper-proof and reliable for later verification.

For the blockchain was implemented using Ganache, a local Ethereum development blockchain.

Ganache:<https://archive.trufflesuite.com/ganache/>







the code

from web3 import Web3

import hashlib

import time

ganache\_url = "http://127.0.0.1:7545"

web3 = Web3(Web3.HTTPProvider(ganache\_url))

if not web3.is\_connected():

print("Eroare: Nu mă pot conecta la Ganache.")

exit()

sender = "0x2D87aC28AfAE5f303bF488Aad89Ac69d99300Ccd"

recipient = "0x8D9738fE267620035e5F82be93dA18962eABddEb"

nonce = web3.eth.get\_transaction\_count(sender)

file\_path = "ionut-barbalan.txt"

with open(file\_path, "r") as f:

lines = f.readlines()

for line in lines:

line = line.strip()

if not line:

continue

data\_hash = hashlib.sha256(line.encode()).hexdigest()

print(f"Trimitem hash: {data\_hash}")

txn = {

'from': sender,

'to': recipient,

'value': 0,

'gas': 200000,

'gasPrice': web3.to\_wei('20', 'gwei'),

'nonce': nonce,

'data': web3.to\_hex(text=data\_hash)

}

txn\_hash = web3.eth.send\_transaction(txn)

print(f"Tranzacție trimisă cu hash: {txn\_hash.hex()}")

nonce += 1

time.sleep(1)