Proiect Sisteme Incorporate
Munteanu Ioan-Dorian
1406B

1.Stabilire domeniu / tema

Tema: Sistem termostat cu aplicatie IBM IoT .

2.Rezumat , resurse

Rezumat:Raspberry Pi primeste informatii prin 1-Wire . Aceasta afișeaza datele în aplicatie și asteapta ca utilizatorul sa trimita comenzi.

Resurse:

- Raspberry PI Zero W
- Senzor Temperatura DS18B20
- Rezistor 4.7 KΩ
- Servomotor

3.Explorare documentara / alternative solutie

Am ales Raspberry Pi Zero deoarece corespunde cerințelor dar în locul acesteia se putea folosi și BananaPi sau Intel Galileo.

https://www.raspberrypi.org/learning/hardware-guide/
http://www.banana-pi.org/
https://www.arduino.cc/en/ArduinoCertified/IntelGalileo

Aplicatia ar putea fi rulata si printr-un server HTTP Apache , dar am ales IBM IoT din dorinta de a explora aceasta tehnologie si din cauza tool-ului Node-RED ce ofera dezvoltare flow-based pentru dispositive hardware conectate la platforma.

Link-uri catre alte proiecte asemănătoare:

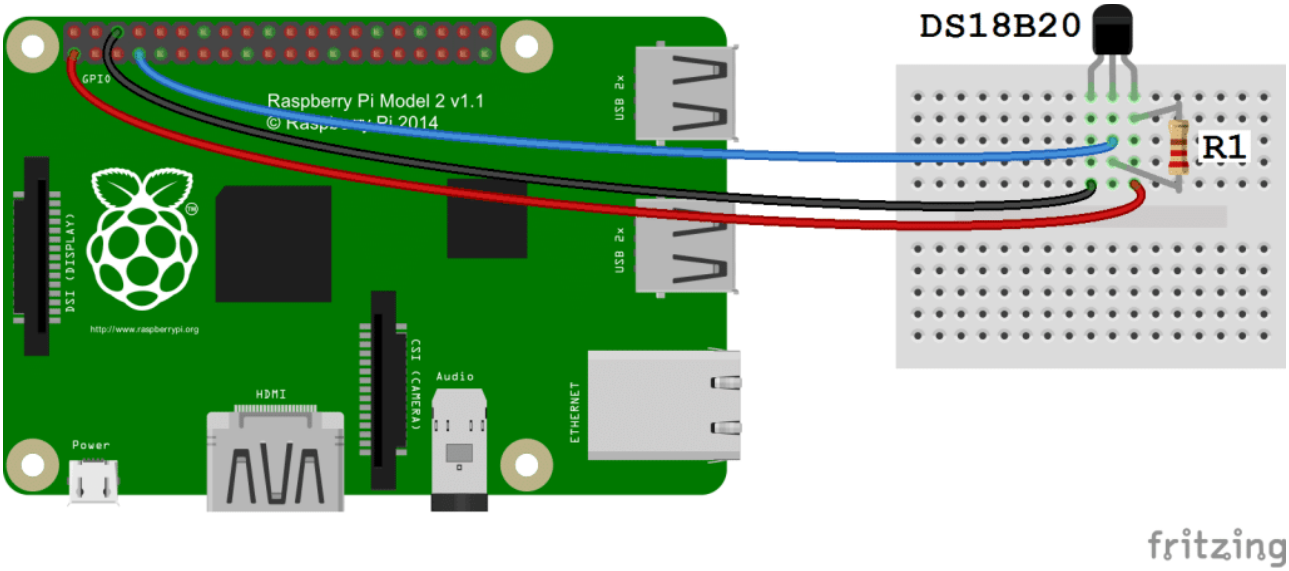
Watering System : https://www.youtube.com/watch?v=nUHizmtyt74

IBM Watson IoT to Rpi Connection https://www.youtube.com/watch?v=0rPA1w6M2Q0&t=1s

Node-RED Home Automation https://www.youtube.com/watch?v=GeN7g4bdHiM&t=380s

Ciorna schema :

Wiring pentru senzorul digital de temperatura



Script Python pentru scheduling in Cron :

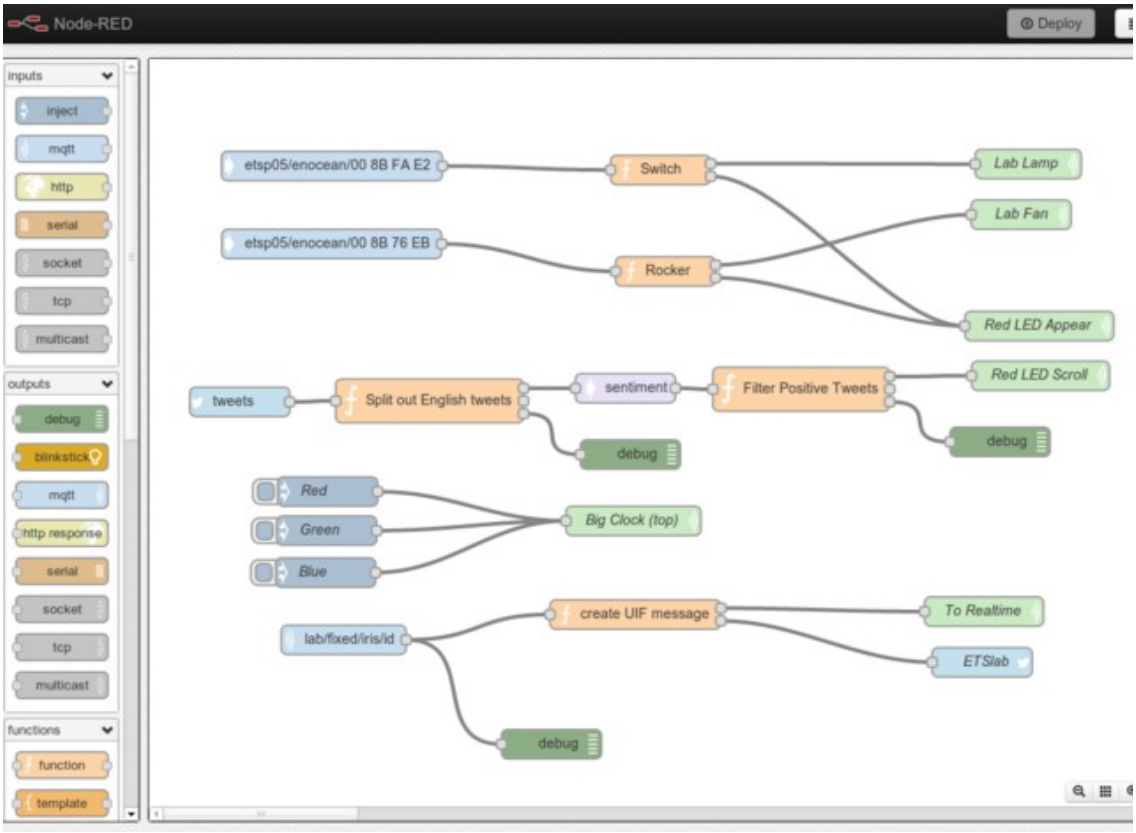
```
import os
import glob
import time

os.system('modprobe w1-gpio')
os.system('modprobe w1-therm')

base_dir = '/sys/bus/w1/devices/'
device_folder = glob.glob(base_dir + '28*')[0]
device_file = device_folder + '/w1_slave'

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
```

```
while True:
        print(read_temp())
        time.sleep(1)
```

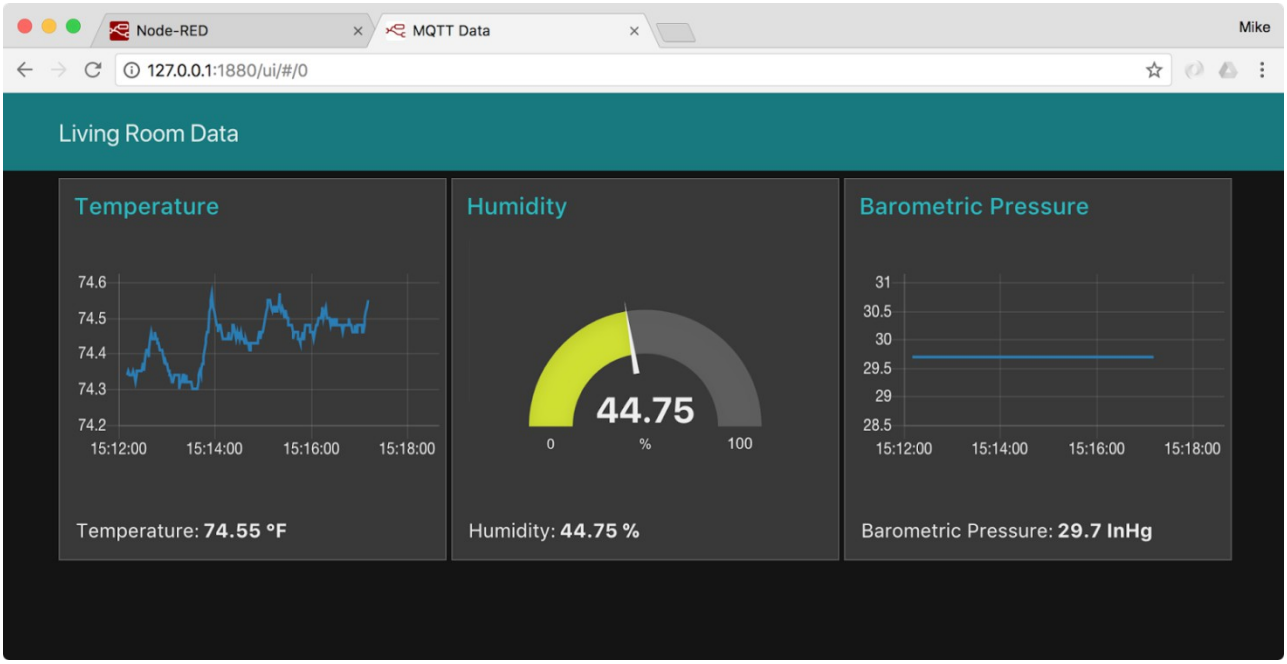
Pentru interfata cu utilizatorul am folosit NODE-Red , un pachet pe baza de Node.js , dezvoltat pentru IBM IoT ce faciliteaza folosirea Raspberry Pi pentru aplicatii web , metoda principala dezvoltarii unui astfel de program fiind flow-based development.

In urma instalarii pachetului , se scrie in terminal node-red , urmand apoi sa se intre de pe un browser la adresa placutei cu extensia “:1880”.

Dashboard-ul prin care se dezvolta aplicatia.



UI pentru urmarire a datelor dupa ce se ruleaza in dashboard flow-ul creeat



In urma adaugarii butoanelor de start si stop am adaugat si un feature de telefonie mobila.

Folosind serviciul Twilio , la apasarea butonului Start , se primeste un mesaj pe telefon de forma “Twilio trial account- Servomotorul a fost activat!”.