Puscalau Robert - 1309B robert-george.puscalau@student.tuiasi.ro
(poza).

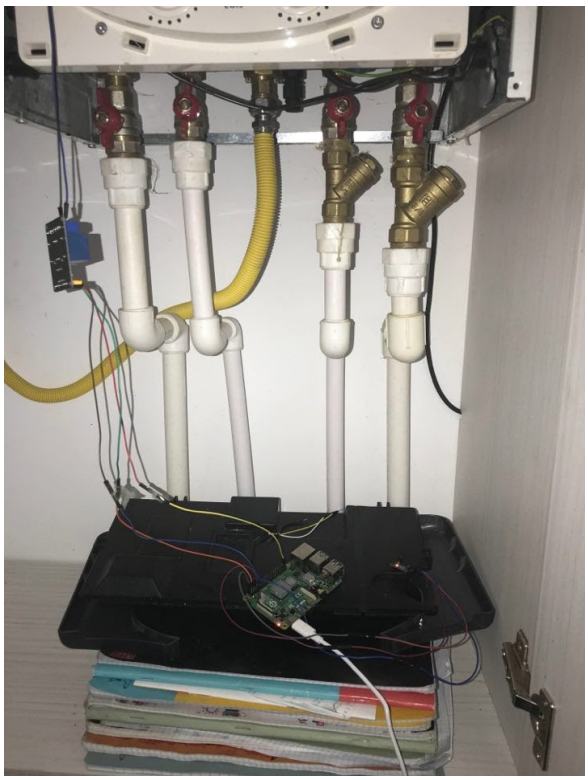
Mihaluca Sergiu Adrian - 1305A mihaluca.sergiu@yahoo.com

Ciobanasu Gabriel Viorel - 1305A ciobanasugabriel@yahoo.com

Name: Control Centrală termică de la distanță

Link proiect hackster.io https://www.hackster.io/347580/home-made-thermostat-for-thermal-plant-7a96e3

Elevator Pitch: Acest proiect își propune dezvoltarea unui sistem care permite controlul temperaturii unui imobil de la distanță, printr-o interfață web.
Cover Image:



Story:

Din lipsa unui termostat in apartament, am decis să dezvoltam singuri sistemul acesta deoarece am considerat ca acest proiect este accesibil pe bază de Raspberry Pi si poate fi extins foarte mult prin software.

Acest proiect este destinat persoanelor care doresc să își automatizeze locuința folosind propriile cunoștințe după bunul plac.

Dispozitivul își propune să implementeze o interfață web prin care locuitorii unui imobil pot controla temperatura locuinței pe baza unor senzori de temperatură plasați în încăperi și raportul energiei termice generate de centrala termică și sistemul de încălzire, afișând informații relevante în timp real.

Components and apps:

-> Rpi4 Model B

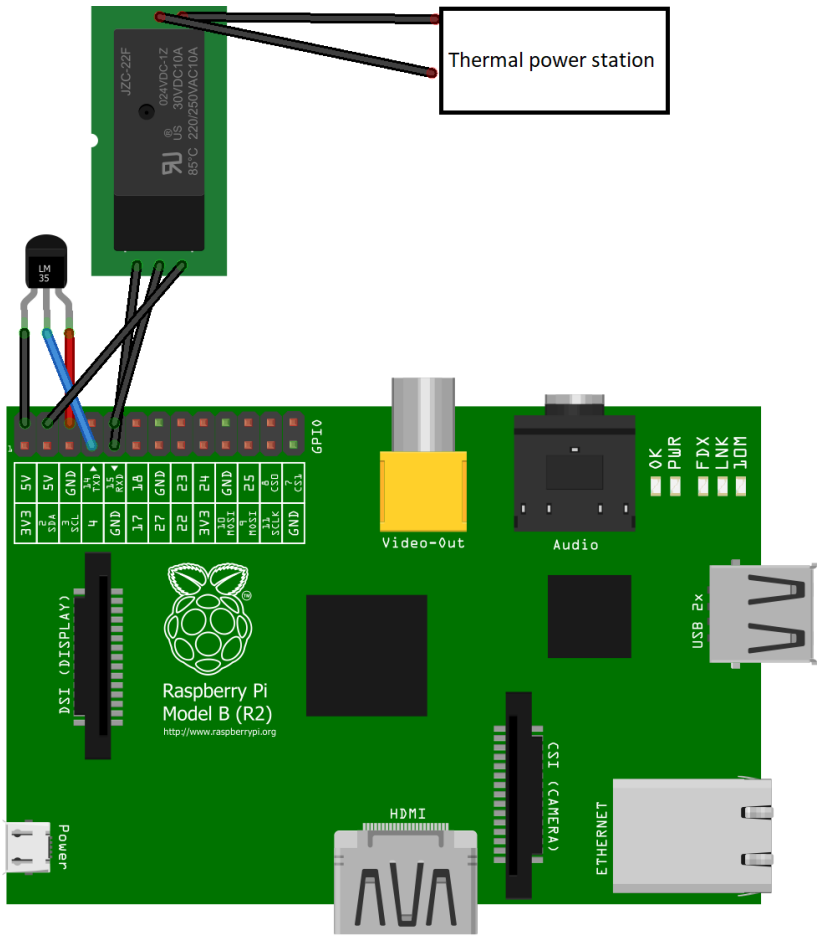
-> Senzor de temperatura ds18b20

-> Modul releu SRD 5v ce actioneaza ca switch pentru centrala

-> Server Python(logica programului)

-> Server web(NodeJS)

Schematics:



(Din pacate, layout-ul pinilor din imaginea de mai sus nu sunt cei de la rpi4,imaginea avand doar scop orientativ, explicatiile exacte urmand mai jos.)

Senzorul de temperatura ds18b20 l-am conectat la Raspberry Pi cu cei trei pini ai sai:

Ground - Pin06

VCC - Pin02

Data - Pin07 (GPIO14)

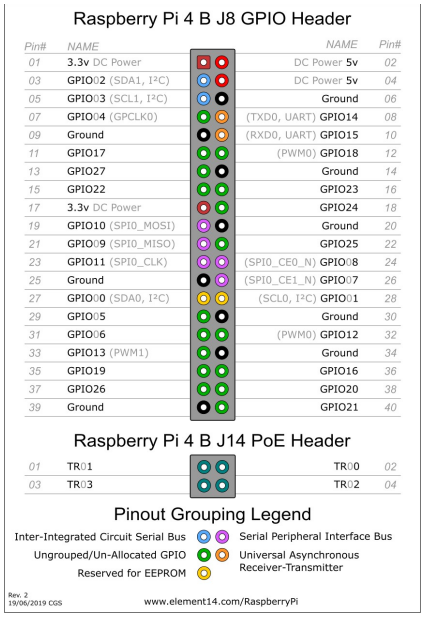
Modulul releu l-am conectat la Raspberry Pi cu cei trei pini ai sai:

Ground - Pin09

VCC - Pin04

Data - Pin40 (GPIO21)

Centrala a fost legata la modulul releu pe modul normal deschis, releul functionand ca si switch.



Sunt doua aplicatii:
Serverul Python primeste comenzi dintr-o baza de date, prelucreaza ultima comanda si in functie de comanda primita(on-desired temperature / off) si temperatura din camera(achizitionata de senzorul de temperatura), acesta ia deciziile necesare pentru centrala si trimite statusul actual server-ului web(nodejs) printr-un fisier json.
Serverul web are rol ca sistemul sa fie disponibil de la distanta pentru locuitorii apartamentului, asigurand cu usurinta acces la acesta si posibilitatea de a trimite comenzi si a verifica statusul actual al centralei.

Cod aferent serverului Nodejs(din cauza dimensiunii mari l-am urcat pe github):
https://github.com/gabrielviorelciobanasu/proiect-sm
Cod aferent serverului Python:

main.py

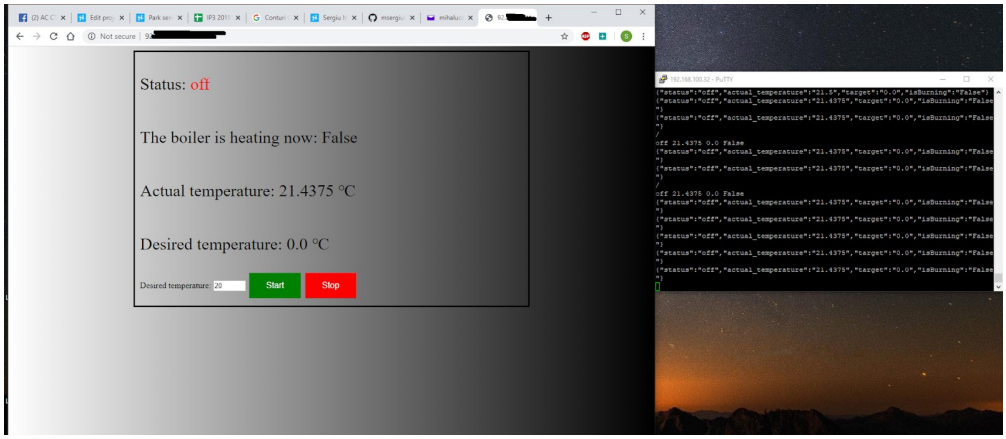
```
#import RPi.GPIO as GPIO
#from w1thermsensor import W1ThermSensor
import ...



# --------------- INIT ---------------
channel=21
#GPIO.setmode(GPIO.BCM)
#GPIO.setup(channel,GPIO.OUT)
$#ensor=W1ThermSensor()
isBurning = False
pin = 21

con=MySQLdb.connect(user='auto',passwd='',host='localhost',db='sm')
c=con.cursor()




# ---------------- CLASSES ------------
class Command:
    def __init__(self, id, data, tip, target):
        self.id = id
        self.data = data
        self.tip = tip
        self.target = target



# -------------- FUNCTIONS ------------
def getCommand():
     con.commit()
     query=("SELECT id, data, tip_comanda, target FROM commands where (select max(id) from commands) = id")
     c.execute(query)
     for(id,data,tip_comanda,target) in c:
         command=Command(id,data,tip_comanda,target)

     return command
def readTemp():
    #return sensor.get_temperature()-5
    return 27
```

main.py

```
    #return sensor.get_temperature()-5
    return 27

def maintainCurrentTemp():
    global pin
    global currentCommand
    global isBurning

    if(currentCommand.tip == "on"):
        if(readTemp() > currentCommand.target+1):
            #GPIO.output(pin, GPIO.LOW)
            isBurning = False
        if(readTemp() <= currentCommand.target-1):
            #GPIO.output(pin, GPIO.HIGH)
            isBurning = True
    if(currentCommand.tip == "off"):
        #GPIO.output(pin, GPIO.LOW)
        isBurning = False

# ---------------- MAIN ---------------
def main():
    global currentCommand
    currentCommand = getCommand()
    while(True):
    # ----- MAIN LOOP -----
        # Rutina daca avem comanda noua
        command = getCommand()
        if(command.id > currentCommand.id):
            currentCommand = command

        # Rutina de pastrat temperatura

        maintainCurrentTemp()
        aux = readTemp()
        data = '{"status":"'+currentCommand.tip+'","actual_temperature":"'+str(aux)+'"' \
               ,"target":"'+str(currentCommand.target)+'","isBurning":"'+str(isBurning)+'"}'
        with open('/home/pi/projects/sm_nodejs/project-sm/information.json','w') as outJson:
            outJson.write(data)
        print(data)



if __name__ == "__main__":
    main()


con.close()
```

Demo Functionalitate:

Intru de unde doresc la adresa ip specifica si pe portul asociat aplicatii. In acest caz centrala nu incalzeste deoarece(este vara :).



Deoarece imi este frig, trimit comanda de incalzire.(Se observa cum serverul Python primeste comanda, centrala porneste - -isBurning -- iar interfata web este actualizata). Micile fluctuatii de temperatura sunt din cauza senzorului ieftin(achizitionat acum din motive de test).