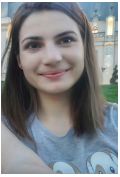


## Radu Cosmina-Mihaela



Titlu: Smart Home

This project implements a system for an intelligent home. It consists of an alarm which can be turned off or on through a web page. It has been used PIR sensor. When it detects some movements, it will send an email which must be set from the interface.

It also consists of a temperature detection sensor which tells us the temperature value, the humidity and it also helps you to turn the light on. This all could be possible due to the usage of a temperature and humidity sensor. Once the actual temperature is wanted, the user must just press a button. Regarding the light bulb controlling, a relay has been used as well as a tension source which can be turned on/off. All these can be controlled from the interface.

We used Flask framework and html, which can help the user to easily interact with the components.

Name: Smart Home

Team members:

Fechet Ionela Paula- 1309B

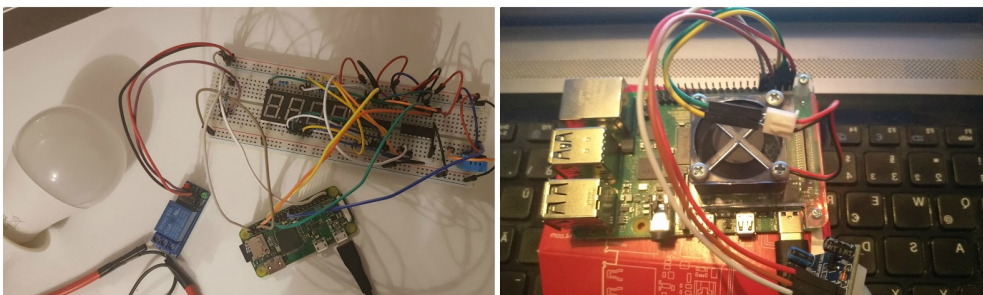
Heghea Mihail Cristian- 1309A

Maftai Elena Claudia- 1310A

Radu Cosmina Mihaela- 1309B

Elevator pitch: Get comfortable in your own home and this smart application will help you live better than before.

Cover image:



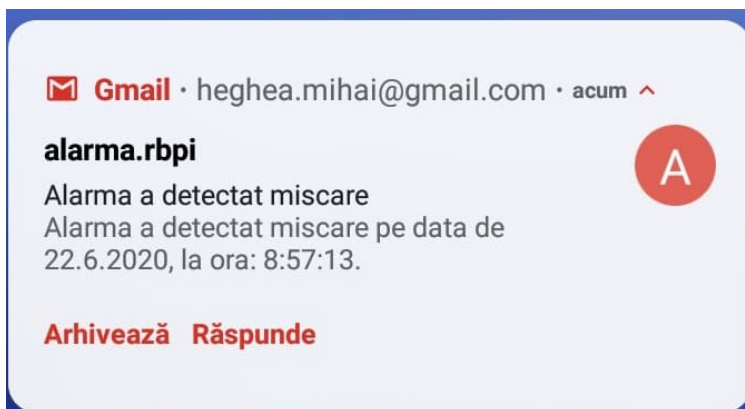
Story:

In order to have a better secured home and live a in good conditions in your own house, this application is the best for you:

-it awares you of the temperature from your room

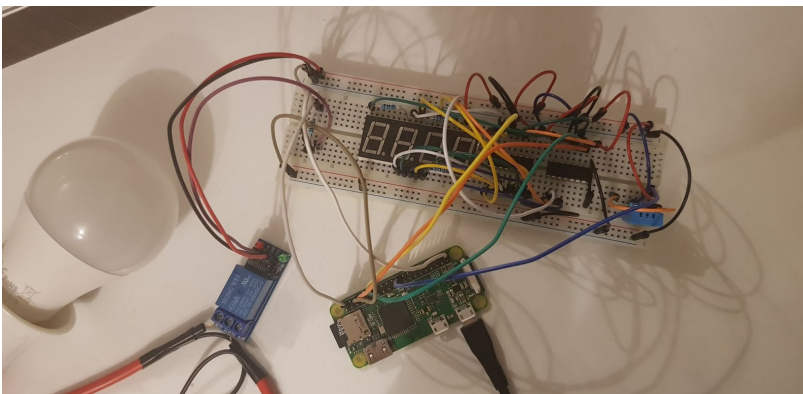
- it helps you turn the light on/off just by pressing a button
- it awares you of the humidity from your room, as humidity can affect human health because it affects our thermal comfort - in other words, whether we feel too hot or too cold.
- you have the option to enable an alarm which can make you aware of the presence of an unknown person in your own house. The message is sent through an email which is set through a web interface. The message tells you that it has detected some movements in your house as well as the moment of time when it happened.

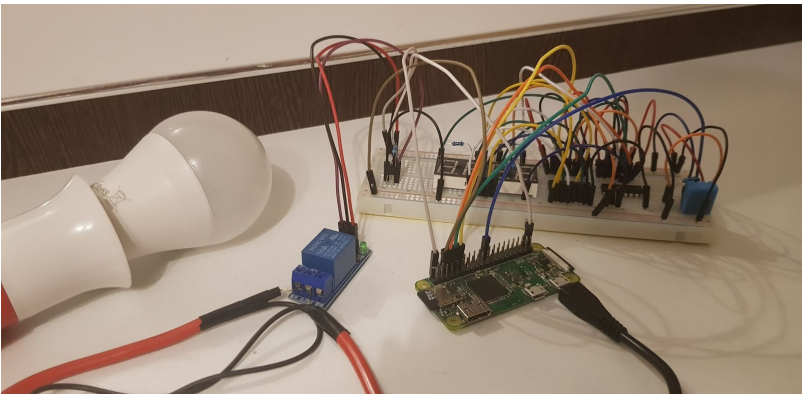
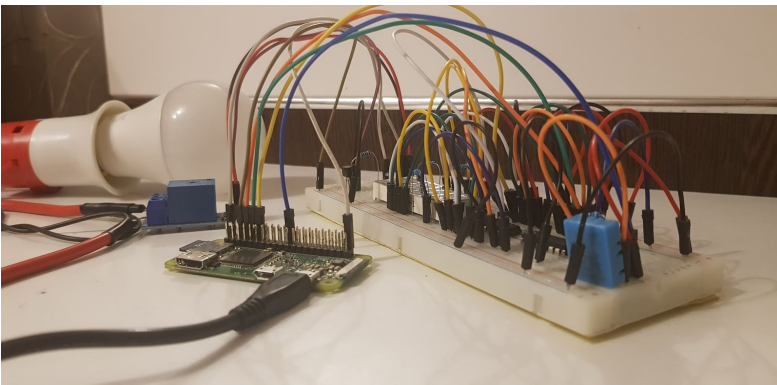
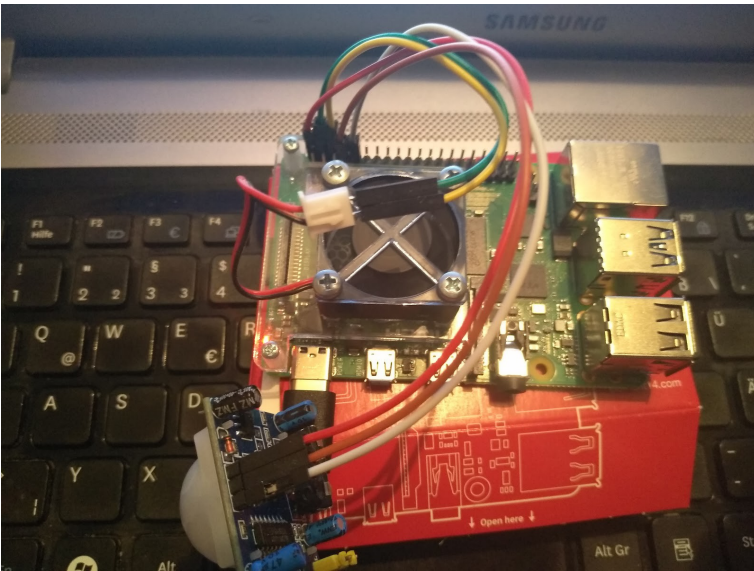
To enable all these functionalities, there are some buttons and links on the web interface. The user can anytime find out what the temperature is by pressing the "Temperature" link, as well as the "Humidity" when he wants to know about humidity. When it comes to setting the alarm, there are 2 buttons : on and off. "On" button enables the alarm and once it observes some movements in your house, you will be notified by receiving an email. An example of email:



Video: <https://www.youtube.com/watch?v=Vh1hknU9L3M&feature=youtu.be>

Photos:





Components and apps:

Hardware Components:

- Raspberry Pi Zero W
- DHT11 Temperature & Humidity Sensor (4 pins)
- PIR sensor for movement detection
- Raspberry Pi 4
- BreadBoard

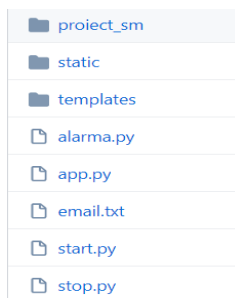
- Light bulb
- Relay
- Resistor 5100Ohm
- 2 shifting registers for displaying (74HC595)
- 4 resistors 150Ohm, one digit (4 Digit 7 Segment).

Software Components:

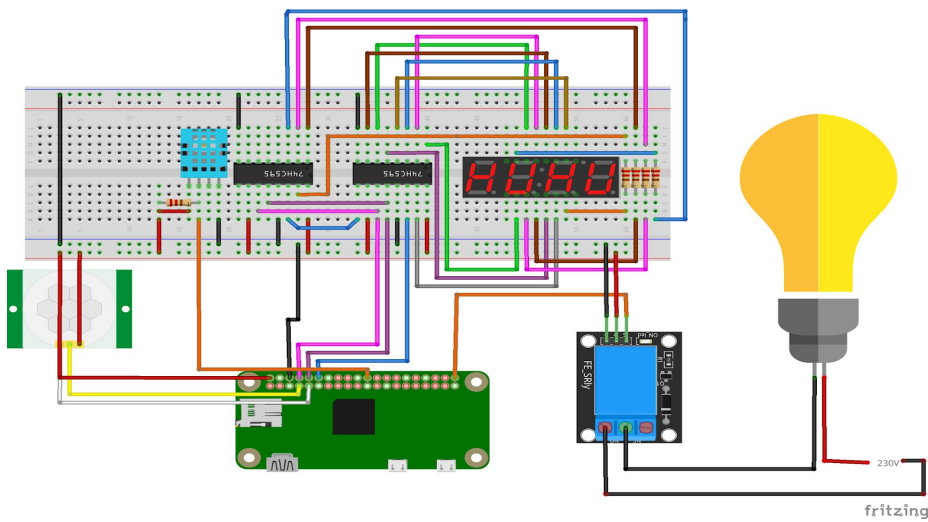
- Raspbian
- Python
- Flask

### **Project Structure**

The project folders must be as in the picture below in order to make the application work properly. In static files, we used resources such as images and css. In “templates” folder we included everything which is related to html. Everything that ends in .py is a python script which allows us to execute commands (e.g: turn on the alarm, turn on the light, get the temperature/humidity).



**Schematics:**



Code:

The code can also be found here: <https://github.com/PaulaFechet/SM>

### **Alarma.py**

It will detect some movements and then send an email to the owner and he will be notified about this

```
import RPi.GPIO as GPIO
import time
import smtplib

def readEmail():
    f = open("email.txt", "r") # se deschide fişierul cu adresa de mail
    return f.read() # se întoarce un şir de caractere care contine adresa de
mail

server = smtplib.SMTP('smtp.gmail.com', 587) #instantiere server smtp
server.starttls() #porinre server smtp
server.login("alarma.rbpi@gmail.com","Alarma123") # conectare cu adresa de
mail care o sa trimita mail utilizatorului
GPIO.setmode(GPIO.BOARD) # alegerea modului de utilizare al pinilor
move=7 # pinul pe care se primeşte semnalul atunci cand senzorul detecteaza
```

```

miscare
GPIO.setup(move, GPIO.IN) # pinul 7 trebuie sa fie de tip input
time.sleep(0.1)
movesDetected = 0 # inițial nu s-a detectat nici o mișcare

prevDetection = time.localtime() # momentul detectării mișcării anterioare
atunci cand porneste aplicația este momentul pornirii acesteia
try:
    while 1:
        if GPIO.input(move)==1: # atunci cand detecteaza miscare
            movesDetected+=1 # numărul de mișcări detectate de la ultima avertizare
este incrementat
            timeDetection = time.localtime() # se preia momentul detectiei miscarii
            minDetection = timeDetection.tm_min
            print("Detectie miscare")

            if prevDetection.tm_min > timeDetection.tm_min: # dacă minutul
detectiei este din ora următoare sau mai târziu
                minDetection = 60+timeDetection.tm_min # se aduna cu 60 (numărul de
minute al unei ore) pentru a se putea face diferența mai jos
                if prevDetection.tm_mday != timeDetection.tm_mday or
prevDetection.tm_hour != timeDetection.tm_hour or (minDetection-
prevDetection.tm_min > 2 and movesDetected >=2) or movesDetected>4:
                    #dacă se detectează mișcare în alta zi sau alt ora sau în interval de
2 minute se înregistrează cel puțin 2 mișcări sau se
                    # înregistrează mai mult de 4 mișcări în mai puțin de 2 minute, este
trimis mail către utilizator cu o avertizare
                    movesDetected=0 # se reseteaza numărul de mișcări detectate
                    headers = ["From: alarma.rbpi@gmail.com", "Subject: Alarma a detectat
miscare", "To: heghea.mihai@gmail.com", "MIME-Version: 1.0", "Content-Type:
text/html"]
                    headers = "\r\n".join(headers) # se construiește header-ul mail-
ului
                    msg = "Alarma a detectat miscare pe data de "
                    msg+= str(timeDetection.tm_mday)+". "

```

```

msg+= str(timeDetection.tm_mon)+"."
msg+= str(timeDetection.tm_year)+", la ora: "
msg+=str(timeDetection.tm_hour)+":"
msg+=str(timeDetection.tm_min)+":"
msg+=str(timeDetection.tm_sec)+"." # se construiește mesajul trimis
prin e-mail, acesta conține data, ora, minutul și secunda la care s-a
raportat acea avertizare

print(msg)

email = readEmail() # se preia adresa de mail către care sa se
trimită acel email, adresa se va adăuga din interfața web

print("Mesaj trimis catre: "+email)

server.sendmail("alarma.rbpi@gmail.com",email ,headers+"\r\n\r\n"+
msg) # trimiterea efectiva a mail-ului

prevDetection = timeDetection # momentul detecției anterioare devine
momentul detecției curente

if minDetection-prevDetection.tm_min>2 and movesDetected<2:

    movesDetected=0 # în cazul în care în 2 minute nu se detectează cel
puțin 2 mișcări, se reseteaza numărul de mișcări și implicit și intervalul

    time.sleep(4)
except Exception as excep:

    print(excep)
finally:

    GPIO.cleanup()

```

### Digit\_temp.py for digitally displaying the temperature

```

import RPi.GPIO as GPIO

import Adafruit_DHT #import librarie pentru a calcula umiditatea si
temperatura

import time

import math as math

GPIO.setmode(GPIO.BCM) #alegere pini

```

```
dataPin = 18
latchPin = 15
clockPin = 14

GPIO.setup(dataPin, GPIO.OUT) #setare pini
GPIO.setup(latchPin, GPIO.OUT)
GPIO.setup(clockPin, GPIO.OUT)

GPIO.output(dataPin, GPIO.LOW) #configurare pini
GPIO.output(latchPin, GPIO.LOW)
GPIO.output(clockPin, GPIO.LOW)

#reinitializare variabile pentru afișarea pe digit
g = 0b01000000
dot = 0b10000000
zero = 191 #0b10111111
zero_no_dot = 63 #0b00111111
one = 134 #0b10000110
one_no_dot = 6 #0b00000110
two = 219 #0b11011011
two_no_dot = 91 #0b01011011
three = 207 #0b11001111
three_no_dot = 79 #0b01001111
four = 230 #0b11100110
four_no_dot = 102 #0b01100110
five = 237 #0b11101101
five_no_dot = 109 #0b01101101
six = 253 #0b11111101
six_no_dot = 125 #0b01111101
seven = 135 #0b10000111
seven_no_dot = 7 #0b00000111
eight = 255 #0b11111111
eight_no_dot = 127 #0b01111111
```



```

nine = 239 #0b11101111

nine_no_dot = 111 #0b01101111

digit = 0

def Digit(x):
    global digit
    if x == 1:
        digit = 14 #0b00001110 activează primul digit punand pe 0 catodul
        corespunzător acestuia
    elif x == 2:
        digit = 13 #0b00001101 activează al doilea digit punand pe 0 catodul
        corespunzător
    elif x == 3:
        digit = 11 #0b00001011 activeaza al treilea digit
    elif x == 4:
        digit = 7 #0b00000111 activeaza al patrulea digit
    elif x == 5:
        digit = 0 #0b00000000 activează punctul

def shift(buffer): #functie de shiftare ce face posibilă afișarea unor valori
diferite pe fiecare digit

    global digit

    for i in range(0,8):
        GPIO.output(dataPin, (128 & (digit << i)))
        GPIO.output(clockPin, GPIO.HIGH)
        time.sleep(0.001)
        GPIO.output(clockPin, GPIO.LOW)

```

```
for i in range(0,8):

    GPIO.output(dataPin, (128 & (buffer << i)))

    GPIO.output(clockPin, GPIO.HIGH)

    time.sleep(0.001)

    GPIO.output(clockPin, GPIO.LOW)

GPIO.output(latchPin, GPIO.HIGH)

time.sleep(0.001)

GPIO.output(latchPin, GPIO.LOW)

def afla_nr(x): #functie ce determina ce leduri se vor aprinde în functie de
numărul dat ca parametru

    nr = 0

    global zero_no_dot
    global one_no_dot
    global two_no_dot
    global three_no_dot
    global four_no_dot
    global five_no_dot
    global six_no_dot
    global seven_no_dot
    global eight_no_dot
    global nine_no_dot

    if x == 0:

        nr = zero_no_dot
    elif x == 1:

        nr = one_no_dot
    elif x == 2:

        nr = two_no_dot
    elif x == 3:

        nr = three_no_dot
    elif x == 4:
```

```

    nr = four_no_dot
elif x == 5:
    nr = five_no_dot
elif x == 6:
    nr = six_no_dot
elif x == 7:
    nr = seven_no_dot
elif x == 8:
    nr = eight_no_dot
elif x == 9:
    nr = nine_no_dot
return nr

# functie ce afișează temperatura pe digit
# formatul este xy.z
#primul digit ramane nefolosit
def display(temperature):
    x = math.floor((temperature *10) /100)
    a = afla_nr(x)
    y = ((temperature * 10) / 10) % 10
    b = afla_nr(y)
    z = (temperature * 10) % 10
    c = afla_nr(z)
    i = 1000

    i = 50
    while i>0:
        i = i-1

        if temperature < 0:
            Digit(1) #dacă temperatura e negativă, se activeaza ledul din mijloc
            pentru a arata semnul minus
            shift(g)

```

```

    time.sleep(0.0000001)

    Digit(2) #se afișează cifra zecilor

    shift(a)

    time.sleep(0.0000001)

    Digit(3) #se afișează cifra unităților

    shift(b)

    time.sleep(0.0000001)

    shift(dot) #se afiseaza punct

    time.sleep(0.0000001)

    Digit(4) #se afișează zecimea

    shift(c)

    time.sleep(0.0000001)

def printfile (str): # functie afișare fișier temperatura +umiditate

    file = open("file.txt","w")

    file.write(str)

    file.close()

def readfile(): #functie citire fisier

    file =open("file.txt","r")

    str = ''

    str += file.read()

    file.close()

    return str

def temp_print(temp): #funcție scriere în fișier separat temperatura, pentru
a outea fi citita si afisata pe digit

    file = open("temp.txt", "w")

    file.write(temp)

    file.close()

```

```

try:
    while True:
        file = open("temp.txt", "r") #citire temperatura din fisier
        temp = ''
        temp +=file.read()
        temperature = float(temp)
        display(temperature) #apel functie afisare temperatura pe display
        file.close()
except KeyboardInterrupt:
    pass

GPIO.cleanup()

```

### Start.py

```

import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)

port = 21 #alege pin 21 ca pin de iesire

GPIO.setwarnings(False)

GPIO.setup(port, GPIO.OUT) #seteaza pinul 21 ca pin de ieşire

GPIO.output(port, GPIO.HIGH) #pune pinul pe HIGH pentru a aprinde becul

```

### Stop.py

```

import RPi.GPIO as GPIO

```

```
GPIO.setmode(GPIO.BCM)

port = 21 #alege pinul 21 ca pin de ieşire

GPIO.setwarnings(False)

GPIO.setup(port, GPIO.OUT) #seteaza pinul 21 ca pin de ieşire

GPIO.output(port, GPIO.LOW) #pune pinul pe LOW pentru a stinge becul
```

### Senzor\_temp.py

```
import RPi.GPIO as GPIO

import Adafruit_DHT #import librărie pentru luarea de la senzor a
temperaturii și a umiditatii

import time

import math as math

import threading

GPIO.setmode(GPIO.BCM)

type = Adafruit_DHT.DHT11

dht11 = 25 #conectare senor la raspberry pi, pinul 25
GPIO.setup(dht11, GPIO.IN)

def printfile (str): #funcție scriere în fișier temperatura și umiditate

    file = open("file.txt","w")

    file.write(str)

    file.close()
```

```

def readfile(): #funcție citire din fișier
    file = open("file.txt", "r")
    str = ''
    str += file.read()
    file.close()
    return str

def temp_print(temp): #funcție scriere în fișier temperatura
    file = open("temp.txt", "w")
    file.write(temp)
    file.close()

try:

    while True:
        humidity, temperature = Adafruit_DHT.read_retry(type, dht11) #luare de la
senzor a temperaturii si umiditatii
        if humidity is not None and temperature is not None:
            temp_print(str(temperature)) #scriere temperatura în fișier, de unde va
fi citită pentru senzor
            buffer = ' Temperature = ' + str(temperature) + '\t' + 'Umiditate = ' +
str(humidity)
            printfile(buffer) #scriere temperatura și umiditate în fișier

except KeyboardInterrupt:
    pass

GPIO.cleanup()

```

## App.py

```
from flask import Flask,
render_template, flash, url_for, session, request, logging, redirect
from http import cookies

import time

import os

from wtforms import Form, StringField, TextAreaField, validators

import RPi.GPIO as GPIO

import Adafruit_DHT

import time

import math as math

import threading

GPIO.setmode(GPIO.BCM)

type = Adafruit_DHT.DHT11

dht11 = 25

GPIO.setup(dht11, GPIO.IN)

app = Flask(__name__)

def read_temp():
    # citeste temperatura de la senzor
    humidity, temperature = Adafruit_DHT.read_retry(type, dht11)
```



```
temp = str(temperature)

return temp

def read_umid():
    # citește umiditatea de la senzor
    humidity, temperature = Adafruit_DHT.read_retry(type, dht11)
    umidit = str(humidity)
    return umidit

class CheForm(Form):
    email = StringField('Email', [validators.Length(min=1, max=50)])

# se primește adresa de email de pe interfața apoi se scrie în fișier
@app.route("/", methods = ['GET', 'POST'])
def home_1():
    form = CheForm(request.form)
    if request.method == 'POST' and form.validate():
        mesaj = form.email.data
        printfile(mesaj)
```

```
    return redirect(url_for('home'))
    return render_template('register.html', form=form)

@app.route("/home")
def home():
    return render_template('index.html')

@app.route("/temp", methods = ['GET', 'POST'])
def temp():
    temperatura = read_temp()
    if request.method == 'POST':
        # pornire script de scriere pe display
```

```
cmd = "python3 digit_temp.py"
```

```
os.system(cmd)
```

```
    return redirect(url_for('temp'))
    return render_template('temperatura.html',temperatura=temperatura)

@app.route("/umiditate")
def umiditate():
    # se apelează funcția care returnează umiditatea de la senzor apoi se
    trimite pe interfața pentru afișare
    umiditate = read_umid()
    return render_template('umiditate.html',umiditate=umiditate)

def printfile (mesaj):
    file = open("email.txt","w")
    file.write(mesaj)
    file.close()

@app.route("/alarma")
def alarma():
    return render_template('alarma.html')

# se primește adresa de email de pe interfața apoi se scrie în fișier
@app.route("/register",methods = ['GET','POST'])
def register():
    form = CheForm(request.form)
    if request.method == 'POST' and form.validate():
        mesaj = form.email.data
        printfile(mesaj)
```

```
return redirect(url_for('home'))  
return render_template('register.html', form=form)
```

```
@app.route("/alarma-on")
def alarma_on():
    #pentru pornit alarma
    cmd = "sudo python alarma.py"
    os.system(cmd)
    msg = "Alarma a fost pornita!"
    # se trimite mesajul de pornire alarma pe interfata
    return render_template('alarma.html',msg=msg)

@app.route("/alarma-off")
def alarma_off():
    # se opreste alarma
    msg = "Alarma a fost oprita!"
    cmd=" ps aux | grep -ie 'python alarma.py' | awk '{print $2}' | xargs sudo
kill -9"
    os.system(cmd)
    # se trimite mesajul de oprire alarma pe interfata
    return render_template('alarma.html',msg=msg)

@app.route("/led")
def led():
    return render_template('led.html')

@app.route("/led-on")
def led_on():
    # Ruleaza scriptul de start led
    cmd = 'python3 start.py'
    os.system(cmd)
    msg = "Becul a fost pornit!"
    return render_template('led.html',msg=msg)
```

```
@app.route("/led-off")
def led_off():
    # Ruleaza scriptul de stop led
    cmd = 'python3 stop.py'
    os.system(cmd)
    msg = "Becul a fost oprit!"
    return render_template('led.html',msg=msg)

@app.route("/digit_temp")
def temp_digit():
    #Ruleaza scriptul de afisare temperatura
    cmd = "sudo python senzor_temp.py"
    t = threading.Thread(target=os.system, args=(cmd,))
    t.start()
    cmd = "sudo python digit_temp.py"
    t2 = threading.Thread(target=os.system, args=(cmd,))
    t2.start()
    return render_template('temperatura.html')

@app.route("/stop_display")
def stop_display():
    cmd = " ps aux | grep -ie 'python senzor_temp.py' | awk '{print $2}' | xargs sudo kill -9"
    os.system(cmd)
    cmd = " ps aux | grep -ie 'python digit_temp.py' | awk '{print $2}' | xargs sudo kill -9"
    os.system(cmd)
    return render_template("temperatura.html")

if __name__ == '__main__':
    app.run(host="0.0.0.0", debug=True)
```

## Alarma.html

```
{% extends 'layout.html' %}

{% block body %}

<center>

<div class="background">

  <center> <h1> <p face="impact" color='navy'>
    Smart HOME </p></h1>

  <p color="white" face="arial"size="3">
    Learn to control the world from anywhere</p>

  </p>

  <br>

<center>

  <h2>Alarma</h2>

  <p >

  <a class="butoane" href='alarma-on'>ON</a>

  <a class="butoane" href='alarma-off'>OFF</a>

  </p>

  <h3> {{ msg }}</h3>

  <br>

</div>

{% endblock %}
```

## \_navbar.html

```
<nav class="navbar navbar-expand-md navbar-dark bg-dark ">

  <div class="container">

    <div class="navbar-header">

      <a class="navbar-brand" href="/home">Smart Home</a>

      <button class="navbar-toggler" type="button" data-toggle="collapse" data-
```



```

target="#navbarsExampleDefault" aria-controls="navbarsExampleDefault" aria-
expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
</button>
</div>
<div class="collapse navbar-collapse" id="navbar">
<ul class="navbar-nav mr-auto">
    <li><a class="nav-link" href="/temp">Temperatura</a></li>
    <li><a class="nav-link" href="/umiditate">Umiditate</a></li>
    <li><a class="nav-link" href="/alarma">Alarma</a></li>
    <li><a class="nav-link" href="/led">Led</a></li>

</ul>

<form class="form-inline my-2 my-lg-0">
    <ul class="navbar-nav mr-auto">
        <li><a class="nav-link" href="/register">Register</a></li>
    </ul>

</form>
</div>
</div>
</nav>

```

## Index.html

```

{% extends 'layout.html' %}

{% block body %}
    <center>
    <div class="background">
        <center> <h1> <p face="impact" color='navy'>

```

```

Smart HOME </p></h1>

<p color="white" face="arial"size="3">

Learn to control the world from anywhere</p>

</p>

<br>

<p>

    Smart home technology, also often referred to as home automation or
domotics

    (from the Latin "domus" meaning home), provides homeowners security,
comfort,

    convenience and energy efficiency by allowing them to control smart
devices,

    often by a smart home app on their smartphone or other networked
device.

    A part of the internet of things (IoT), smart home systems and
devices often operate together,

    sharing

    consumer usage data among themselves and automating actions based on
the homeowners' preferences.

</p>

<br>



<br>

<br><br>

<center>

</div>

{% endblock %}

```

*Layout.html*

```
<!DOCTYPE html>
```

```

<html>
  <head bgcolor="silver" >
    <meta charset="utf-8">
    <title>FlaskApp</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">
    <link rel="stylesheet" href =
"https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">
    <style>

  </style>
</head>
<body >
  {% include 'includes/_navbar.html' %}
  <div class = "container">
    {% block body %}{% endblock %}
  </div>
  <script src =
"https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></s
cript>
</body>
</html>

```

#### led.html

```

{% extends 'layout.html' %}

{% block body %}
  <center>
  <div class="background">
    <center> <h1> <p face="impact" color='navy'>
      Smart HOME  </p></h1>
    <p color="white" face ="arial"size="3">
      Learn to control the world from anywhere</p>
  </div>
  </center>

```

```

    </p>

    <br>

    <center>

    <h2>Led</h2>

    <p>

        <a class="butoane" href='led-on'>ON</a>

        <a class="butoane" href='led-off'>OFF</a>

    </p>

    <h3> {{ msg }}</h3>

    <br>

</div>

{% endblock %}

```

#### Register.html

```

{% extends 'layout.html' %}

{% block body %}

    <center>

    <div class="background">

        <center> <h1> <p face="impact" color='navy'>

            Smart HOME </p></h1>

            <p color="white" face="arial"size="3">

                Learn to control the world from anywhere</p>

            </p>

            <br>

        <center>

            <h2>Register</h2>

            <p>

                {% from "includes/_formhelpers.html" import render_field %}

                <form method="POST" action="">

                    <div id="input" class="form-group">

```

```

        {{ render_field(form.email, class="form-control") }}

    </div>

    <p><input class="btn btn-primary" type="submit" value="Register"></p>

</form>

</p>

    <br>

</div>

{% endblock %}

```

### Temperatura.html

```

{% extends 'layout.html' %}

{% block body %}

    <center>

    <div class="background">

        <center> <h1> <p face="impact" color='navy'>

            Smart HOME </p></h1>

            <p color="white" face="arial" size="3">

                Learn to control the world from anywhere</p>

            </p>

            <br>

        <center>

            <h2>Temperatura</h2>

            <a href="digit_temp">Display</a>

            <a href="stop_display">Stop_display</a>

            <p>{{ temperatura }} °C</p>

            <br>

        </div>

    </center>

{% endblock %}

```

## Umiditate.html

```
{% extends 'layout.html' %}

{% block body %}

<center>

<div class="background">

  <center> <h1> <p face="impact" color='navy'>

    Smart HOME  </p></h1>

  <p color="white" face="arial"size="3">

    Learn to control the world from anywhere</p>

  </p>

  <br>

<center>

  <h2>Umiditate</h2>

  <p>{{ umiditate }}</p>

  <br>

</div>

{% endblock %}
```

Sisteme cu microprocesoare      Examen 24 iunie

Nume student: Radu Cosmina-Mihaela

Grupa: 1309B

Raspundeti personalizat la urmatoarele intrebari:

1. Ce componente hardware-software ati utilizat pentru implementarea proiectului la SM cu titlul Smart home?

Componentele hardware pe care le-am folosit sunt: Raspberry i zero, senzor de temperatura, senzor PIR pentru detectia miscarii, Raspberry Pi4, BreadBoard, Light bulb, rezistor 5100 Ohm. Componentele software pe care le-am folosit sunt: Raspbian, Python, Flask.

2. In ce consta cea mai importanta realizare din proiect?

Întrucât fiecare membru din echipa a avut o contribuție proprie la acest proiect, partea cea mai importanta a fost partea în care a trebuit să ne unim munca și să o facem să funcționeze ca un întreg.

Module precum citirea temperaturii, citirea umidității, afisare display, aprindere led, activare alarma au fost realizate separat, apoi au fost legate prin intermediul interfeței. Satisfacția cea mai mare a fost atunci când totul a decurs fără prea mari probleme, dovedind astfel că am comunicat bine și am lucrat corect.

3. Ce a fost cel mai dificil de făcut la proiect?

Cred că cea mai dificilă parte din proiect a fost să ne decidem asupra temei, precum și partea inițială din proiect (ce avem nevoie, cum implementăm, ce interfață folosim, împărțire sarcini).

4. Ce cunoștințe și abilități trebuie să aibă cineva care vrea să replice/refacă/evalueze proiectul?

Cel care vrea să replice proiectul trebuie să aibă cunoștințe Python, HTML, circuite, afisare display, utilizare pini, Flask, să cunoască Raspberry Pi.

5. În lumea reală, cu eventuale completări, la ce s-ar putea utiliza proiectul?

Cu o completare la partea de control a temperaturii și umidității din camera, controlul luminii din toată casa, extinderea conceptului de alarma prin detectia mișcării, putem obține o aplicație care ne va ușura complet viața în căminul propriu. Prin urmare, acest proiect poate fi utilizat cu succes în realizarea unei case inteligente cu ajutorul căreia vom economisi timp și energie.