Bacica Florin

Ce am lucrat la proiect:

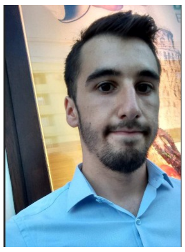
- M-am ocupat cu montarea componentelor electronice
- Testarea aplicatiei
- Realizarea modulului de citire a intensitatii luminoase
- Realizarea proiectului de pe hackster impreuna cu filmuletele de prrezentare

Atomei Georgiana

Ce am lucrat la proiect:

- Realizarea interfetei web + apelul componentelor hardware in functie de setarile din interfata

Ciulei Virgil



Turcas Irina-Maria

Ce am lucrat la proiect:

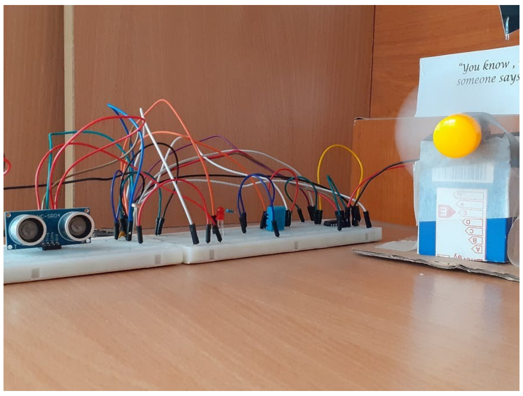
- Realizarea modulului de alarma

Smart Desk

O aplicatie in care folosim un raspberry pi pentru controlul lampii la un birou, a unui ventilator si a unei mici alarme care va porni atunci cand ar trebui sa iei pauza de la lucru. Toate astea vor fi controlate printr-o interfata web, gestionata de un server scris in python cu ajutorul Flask.
Lampa se va aprinde in caz de se detecteaza miscare la birou, ventilatorul se va aprinde atunci cand e prea cald, iar alarma va fi setata de utilizator si pornita din interfata web.

Name: **Smart Desk**
Elevator Pitch: Great project for people who want to have a great experience working at the desk for long hours.

Cover Image:



Story:

As the technology is in a continuous development and more and more people are working at a desk, the need of having a comfortable environment and, why not, also enjoyable, becomes more natural.
Our project comes as a help for anyone working long hours at a desk, offering support for:

- a desk lamp that works in two modes: manual(turned on/off by the user) and auto(turning on when motion is detected, and turning off otherwise to save light when it is not needed)
- a fan that also works in two modes: manual((turned on/off by user) and auto(the fan starts when the temperature goes above 21 degrees Celsius, and stops when the temperature drops below this limit)

- a timer, as in the Pomodoro technique for those of us struggling with being productive.

All these functionalities are accessible from a web interface.

The user can enable the working mode for the lamp and fan by pushing on buttons.

To set the pomodoro alarm we need to specify the time limit, in minutes.When it is set, a count-down timer will be visible on the interface, so the user can see the remaining time.

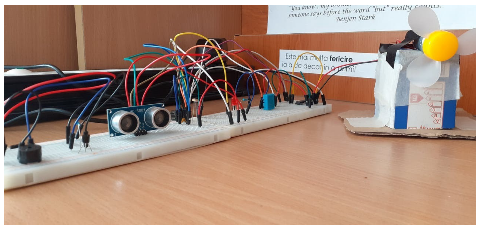
Here it is a video which demonstrate each functionality. Unfortunately, in this video we were not able to show the automatic mode of the fan due to the DHT11 sensor that failed. But we can say for sure that the module works fine with a new sensor, the tests we did initially prove it.

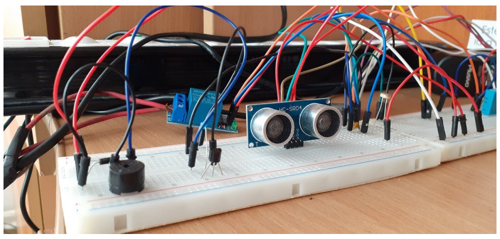
Video:

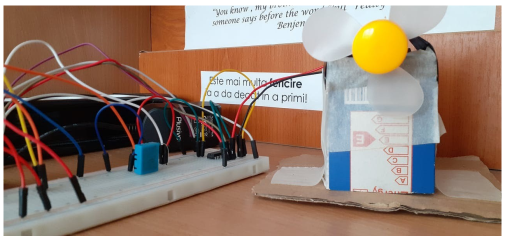
https://www.youtube.com/watch?v=FX02lu9bSzM&feature=emb_title

https://www.youtube.com/watch?v=CWs1zxWbvh4&feature=emb_title

Photos:







Components and apps:

Componente Hardware:

->Raspberry Pi Zero Wireless

-> Ultrasonic Sensor - HC-SR04 (Generic)

-> Buzzer

-> Rezistor 1kOhm

-> General Purpose Transistor NPN

-> DHT11 Temperature & Humidity Sensor (4 pins)

-> Texas Instruments Dual H-Bridge motor drivers L293D

-> DC motor(generic)

-> Male/Female Jumper Wires

-> Breadboard

-> Capacitor 100 nF

Software apps:

->Raspberry Pi Raspbian

Schematics:

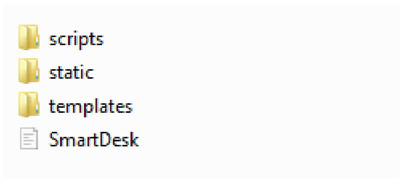
Project Structure

The files in this project must have a structure to ensure that application works.

In folder scripts you will need to put all the python scripts, other than SmartDesk.py.

In folder templates you will need to put the index.html file.

And in static you will need to put the javascript code.



```
scripts
static
templates
SmartDesk
```

Buzzer and ultrasonic:

Transistor:

- emitter: ground (black)
- base: pin 12 (blue)
- collector: to negative pin buzzer (black)

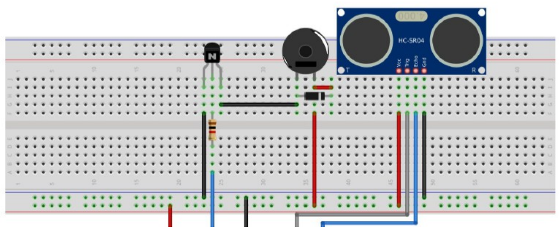
Buzzer:

- positive pin: to 5V power (red)

Ultrasonic:

- VCC: to 3.3V power(red)
- trigger: pin 36 (gray)
- echo: pin 38 (blue)
- ground: to the ground (black)

Diode: parallel with the buzzer



Light sensor schematic:

Light sensor:

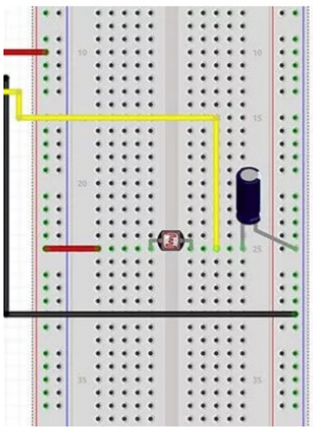
- pin 1: to 3.3V power (red)

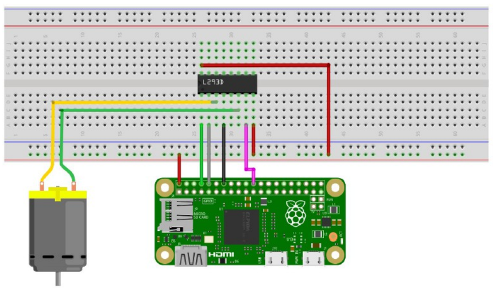
Capacitor:

- One wire to the ground, we use a ceramic capacitor, so, the way we put the capacitor in the circuit doesn't matter

Yellow wire:

- to pin 40



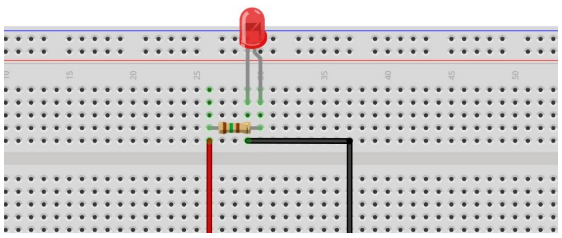


DC motor schematic:

L293D:

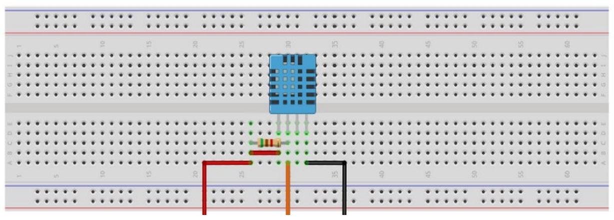
- VSS: to 5V power (red)
- EN1: to pin 8 (green)

- IN1: to pin 10 (gray)
- OUT1: to positive wire DC motor (yellow)
- GND: to ground (black)
- OUT2: to negative wire DC motor (green)
- IN2: to pin 22 (purple)
- VS: to 5V power (red)



LED Schema:

Relay:
- black to pin 11
- red to VCC



DHT11 temperature sensor:

DHT11:
- VCC: to 5V power (red)
- Data: to pin 37 (orange)
- GND: to ground (black)

Code:

Alarm_Driver.py

This module starts an alarm which will make a sound after a custom number of seconds.

```
# Proiect SM Smart Desk
# grupa 1309B
# membri: Atomei Georgiana, Bacica Florin, Ciulei Virgil, Turcas Irina
# autor modul: Turcas Irina

import time
import pigpio
import sys

pi = pigpio.pi()
# pin-ul folosit de buzzer este unul care genereaza
# un cloc$ caz acesta fiind GPIO18, adica pinul 12
buzzer = 18

def suna(minute):
    try:
        print("S-a pornit timerul pentru alarma")
        time.sleep(minute)
        print("Suna alarma!")
        while True:
            # buzzerul este activ cat timp nu s-au scurs 5.$ arg2=700Hz (f mai mare ->
            # sunet mai inalt) arg3=50%FU (FU mai mic -> buzzerul primeste s$ un
            # interval mai mic al clockului)
            pi.hardware_PWM(buzzer, 700, 500000)
            time.sleep(0.13)
            pi.hardware_PWM(buzzer, 0, 0)
            time.sleep(0.1)
            pi.hardware_PWM(buzzer, 900, 500000)
            time.sleep(0.15)
            pi.hardware_PWM(buzzer, 0, 0)
            time.sleep(1.5)
    except KeyboardInterrupt:
        pi.hardware_PWM(buzzer, 0, 0)
```

Opreste_alarma.py

This modules is used if the alarm can't be stopped from interface. It is not used at the runtime of the application but is useful when something goes wrong and the alarm still is on.

```
import signal
import os
import subprocess

try:
    pids = bytes(subprocess.check_output('pgrep -f \'Alarma_Driver.py\'', shell=True))
    pids = pids.split(b'\n')
    print(pids)
    for pid in pids:
        try:
            os.kill(int(pid), signal.SIGINT)
        except Exception as e:
            print(e)
except KeyboardInterrupt:
    print('Alarma a fost oprita')
```

FAN_Driver.py

This module controls the FAN.

```
# Proiect SM Smart Desk
# grupa 1309B
Poze

# Membri: Atomei Georgiana, Bacica Florin, Ciulei Virgil, Turcas Irina
# Autor modul: Ciulei Virgil
# Fan management module

import RPi.GPIO as GPIO
import time
```

```
FAN_SPEED = 14
DIRA = 15
DIRB = 25

pwmPIN = any

def init_FAN_driver():
   global pwmPIN
   pwnPIN = any
   GPIO.setmode(GPIO.BCM)
   GPIO.setwarnings(False)
   GPIO.setup(FAN_SPEED, GPIO.OUT)
   GPIO.setup(DIRA, GPIO.OUT)
   GPIO.setup(DIRB, GPIO.OUT)
   GPIO.setwarnings(False)
   pwmPIN = GPIO.PWM(FAN_SPEED, 100)
   pwmPIN.start(0)



def deinit_FAN_driver():
   GPIO.cleanup()  # clean driver

def stop_fan():
   change_fan_power(0,0)



def change_fan_power(fan_pwm, direction):
   global pwmPIN
   pwmPIN.ChangeDutyCycle(fan_pwm)
   if direction == 1:
      GPIO.output(DIRA, GPIO.HIGH)
      GPIO.output(DIRB, GPIO.LOW)
      GPIO.setwarnings(False)
   else:
      GPIO.output(DIRA, GPIO.LOW)
      GPIO.output(DIRB, GPIO.HIGH)
      GPIO.setwarnings(False)
```

LED_Driver.py

This module controls the relay in which is connected a lamp

```
# Proiect Smart desk
# grupa 1309B
# membri: Atomei Georgiana, Bacica Florin, Ciulei Virgil, Turcas Irina
# autor modul: Ciulei Virgil
# Led on/off module
# Pin led: 17 bcm, 11 board

import RPi.GPIO as GPIO
import time

led = 17

def init_led_driver():
    global led
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(led, GPIO.OUT)
    GPIO.output(led, GPIO.HIGH)

def turn_on_led():
    global led
    GPIO.output(led, GPIO.LOW) # aprind LED
    GPIO.setwarnings(False)

def turn_off_led():
    global led
    GPIO.output(led, GPIO.HIGH) # opresc LED
    GPIO.setwarnings(False)
```

Light_sensor_Driver.py

This modules reads the value on light sensor.

```
# Proiect SM Smart Desk
# grupa 1309B
# Membri: Atomei Georgiana, Bacica Florin, Ciulei Virgil, Turcas Irina
# Autor modul: Bacica Florin
# Light sensor management module
# Pini alesi: 21 bcm sau 40 board

import RPi.GPIO as GPIO
import time

light_sensor = 21

# pe circuit o sa avem senzorul si capacitorul si rezistenta lui se masoara in functie de
# cat de repede se descarca capacitorul
def init_light_sensor_driver():
    GPIO.setmode(GPIO.BCM)

def rc_time():
    count = 0
    GPIO.setwarnings(False)
    GPIO.setup(light_sensor, GPIO.OUT)
    GPIO.output(light_sensor, GPIO.LOW)

    time.sleep(0.1)

    GPIO.setwarnings(False)
    GPIO.setup(light_sensor, GPIO.IN)

    while (GPIO.input(light_sensor) == GPIO.LOW):
        count += 1
        #cand e foarte intuneric valoarea count devine foarte mare asa
        # ca ii vom pune o valoare maxima, peste care daca trece e
        # intuneric sigur
        if count == 50:
            break
    return count
```

Motion_Sensor_Appl.py

This modules uses a ultrasonic sensor to detect if is something moving on the desk, and if it is, turn on the lamp if it is too dark in the room.

```
# Proiect SM Smart Desk
# grupa 1309B
# membri: Atomei Georgiana, Bacica Florin, Ciulei Virgil, Turcas Irina
# autor modul: Ciulei Virgil, Bacica Florin
# Moving sensor Management
# pini: 16(bcm) 36(board) trigger, 20(bcm), 38(board) echo

import RPi.GPIO as GPIO
import time
from scripts import LED_Driver as led
from scripts import Light_sensor_Driver as light

TRIGGER_PIN = 16
ECHO_PIN = 20

def init_motion_appl():
    GPIO.setmode(GPIO.BCM)
    GPIO.setwarnings(False)
    GPIO.setup(TRIGGER_PIN, GPIO.OUT)
    GPIO.setwarnings(False)
    GPIO.setup(ECHO_PIN, GPIO.IN)

def trigger_conversion():
    start_t = 0
    stop_t = 0
    GPIO.output(TRIGGER_PIN, GPIO.HIGH)
    time.sleep(0.00001)
    GPIO.output(TRIGGER_PIN, GPIO.LOW)

    start_t = time.time()
    stop_t = time.time()
    while GPIO.input(ECHO_PIN) == 0:
        start_t = time.time()
    while GPIO.input(ECHO_PIN) == 1:
        stop_t = time.time()
    durata = stop_t - start_t
    dist = durata * 342 / 2 * 100
    return dist

def init_listen_state():
    print("Distante primite in starea de init")
    start_time_listen = time.time()
    stop_time_listen = time.time()
    min_distance = trigger_conversion()
    max_distance = min_distance
    while (stop_time_listen - start_time_listen) < 5:  # 5 secunde pentru calibrare
        time.sleep(0.5)
        trigger = trigger_conversion()
        if trigger < min_distance:
            min_distance = trigger
        if trigger > max_distance:
            max_distance = trigger
        stop_time_listen = time.time()
    print("Valorea minima : ", min_distance)
    print("Valorea maxima : ", max_distance)
    return min_distance, max_distance

def active_state(min_d, max_d):
    trig = trigger_conversion()
    eroare = 15.0
    if trig < (max_d - eroare):
        print("Distanta:"+str(trig))
        return True
    return False

def run():
    # init and listen to some datas
    led.init_led_driver()
    led.turn_off_led()
    init_motion_appl()
    (min_dist, max_dist) = init_listen_state()
    while 1:
        time.sleep(1)
        try:
            lighting = light.rc_time()
            # se intra in if doar daca se detecteaza miscare si este intuneric in camera
            if active_state(min_dist, max_dist) and lighting == 50 :
                print("A aparut un obiect si e intuneric in camera")
                print("Ledul s-a aprins!")
                # turn on the led
                led.turn_on_led()
                start_time_listen = time.time()
                stop_time_listen = start_time_listen
                while (stop_time_listen - start_time_listen) < 30.0:
                    #daca inca avem ceva in raza de actiune atunci nu se cronometreaza nimic, timpii
raman la fel si
                    #se va sta in acest while pana nu e nimic in raza de actiune si au trecut 10 sec
                    time.sleep(1)
                    if active_state(min_dist, max_dist):
                        start_time_listen = time.time()
                        stop_time_listen = start_time_listen
                    else:
                        stop_time_listen = time.time()
                    #iese din while doar daca nu e nimic in fata senzorului si au trecut 10 secunde
#decand nu e nimic acolo
                print("Au trecut 30 secunde decand nu s-a mai detectat miscare!")
                print("Ledul s-a stins!")
                # turn of the led
                led.turn_off_led()
        except KeyboardInterrupt:
            break
```

Temp_Sensor_Appl.py

This module read the temperature on the sensor DHT11.

```
# Proiect SM Smart Desk
# grupa 1309B
# membri: Atomei Georgiana, Bacica Florin, Ciulei Virgil, Turcas Irina
# autor modul: Ciulei Virgil
# Temperature sensor management module
# pini: 26 bcm, 37 board

import RPi.GPIO as GPIO
import Adafruit_DHT
from scripts import FAN_Driver as fan
import time

dht11 = 26
type = Adafruit_DHT.DHT11

def init_temp_driver():
   global dht11
   GPIO.setmode(GPIO.BCM)
   GPIO.setup(dht11, GPIO.IN)

def deinit_temp_driver():
   GPIO.cleanup()

def trigger_conv_temp():
   global dht11
   global type
   tmp, temperature = Adafruit_DHT.read_retry(type, dht11)
   return  temperature

def run():
   init_temp_driver()
   print("Temperatura de dupa care se porneste sezorul este de 20 de grade")
   try:
      while 1:
         try:
            time.sleep(0.5)
            temperature = trigger_conv_temp()
            if temperature is not None:
               print("Temperatura:"+str(temperature))
               if temperature  > 20:
                print("Ventilator pornit!")
                fan.change_fan_power(50, 0)
            else:
                print("Ventilator oprit!")
                fan.change_fan_power(0,0)
        except RuntimeError:
            continue
    except KeyboardInterrupt:
        deinit_temp_driver()
```

SmartDesk.py

This is the module which controls the communication with the web interface. The app is started when this module is run.

```
from flask import Flask, render_template, request, Response
from scripts import LED_Driver, FAN_Driver, Light_sensor_Driver, Motion_Sensor_Appl,
Temp_Sensor_Appl, Alarma_Driver
import multiprocessing

app = Flask(__name__)

fan_init = False
temp_init = False
motion_init = False

automated_air = None
pomodoro_alarm = None
motion_light = None

# initializam drivere pentru obiectele folosite
LED_Driver.init_led_driver()
Motion_Sensor_Appl.init_motion_appl()
Temp_Sensor_Appl.init_temp_driver()
FAN_Driver.init_FAN_driver()

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

```
@app.route('/automated-light',  methods=['POST'])
def automatedLight():
    # initializam modulul de functionare automata
    global motion_init
    global motion_light

    if not motion_init:
        motion_light = multiprocessing.Process(target=Motion_Sensor_Appl.run)
        motion_init=True
        #pornim thread cu functionarea automata
        motion_light.start()
    return Response(status=200)

@app.route('/manual-light-start',  methods=['POST'])
def manualLightStart():
    #daca este pornita functionarea automata, o oprim
    #senzor de miscare
    global motion_init
    global motion_light

    if motion_init:
        motion_light.terminate()
        motion_light=None
        motion_init=False

    #pornim functionarea manuala
    LED_Driver.turn_on_led()
    return Response(status=200)

@app.route('/manual-light-stop',  methods=['POST'])
def manualLightStop():
    # oprire led daca acesta functioneaza
    LED_Driver.turn_off_led()

    return Response(status=200)

@app.route('/manual-air-start',  methods=['POST'])
```

```
def manualAirStart():
    global temp_init
    global automated_air
    global fan_init
    #daca este pornita functionarea automata, trebuie inchisa
    if temp_init:
        automated_air.terminate()
        automated_air = None
        temp_init = False

    # pornire ventilator
    FAN_Driver.change_fan_power(50,1)
    fan_init = True

    return Response(status=200)

@app.route('/manual-air-stop',  methods=['POST'])
def manualAirStop():
    global fan_init
    # oprire ventilator
    if fan_init:       #functionarea manuala poate fi oprita doar daca functioneaza
        FAN_Driver.stop_fan()
        fan_init = False
    return Response(status=200)

@app.route('/automated-air',  methods=['POST'])
def automatedAir():
    #oprirea functionarii manuale, daca aceasta este pornita
    global temp_init
    global automated_air
    manualAirStop()

    if not temp_init:
        #pornirea functionarii automate
        automated_air = multiprocessing.Process(target = Temp_Sensor_Appl.run)
        automated_air.start()
        temp_init = True
    return Response(status=200)

@app.route('/start-alarm',  methods=['POST'])
def pomodoroAlarm():
    global pomodoro_alarm
    # daca avem alta alarma setata, o oprim
    if pomodoro_alarm:
        pomodoro_alarm.terminate()

    # extragem timpul setat in interfata
    time = request.json['oraAlarma']
    if time.isnumeric() and int(time)>0:     #verificam timpul sa fie introdus corect
        time=int(time)                       #pornim thread nou cu alarma
        pomodoro_alarm = multiprocessing.Process(target=Alarma_Driver.suna, args=[time*60,])
        pomodoro_alarm.start()

    return Response(status=200)

@app.route('/stop-alarm',  methods=['POST'])
def stopPomodoroAlarm():
    global pomodoro_alarm
    # daca avem alta alarma setata, o oprim
    if pomodoro_alarm is not None:
        pomodoro_alarm.terminate()
        pomodoro_alarm = None

    return Response(status=200)

if __name__ == '__main__':
    app.run(host='0.0.0.0')
    if automated_air:
        automated_air.terminate()
    if pomodoro_alarm:
        pomodoro_alarm.terminate()
    if motion_light:
        motion_light.terminate()
    # deinit drivers
    FAN_Driver.deinit_FAN_driver()
    Temp_Sensor_Appl.deinit_temp_driver()
```

script.js

This javascript code sends the requested to the server and controls the timer display on the interface.

```
function startManualLight() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
        }
    };

    xhttp.open("POST", "/manual-light-start", true);
    xhttp.send();
}

function startAutoLight() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
        }
    };

    xhttp.open("POST", "/automated-light", true);
    xhttp.send();
}

function stopManualLight() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
        }
    };
    xhttp.open("POST", "/manual-light-stop", true);
    xhttp.send();
}

function startManualAir() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
        }
    };

    xhttp.open("POST", "/manual-air-start", true);
    xhttp.send();
}
function stopManualAir() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
        }
    };

    xhttp.open("POST", "/manual-air-stop", true);
    xhttp.send();
}
function automatedAir() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
        }
    };

    xhttp.open("POST", "/automated-air", true);
    xhttp.send();
}
function startPomodoroAlarm() {
    var xhttp = new XMLHttpRequest();
    let time=document.getElementById("oraAlarma").value
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            startCount(time)
        }
    };

    xhttp.open("POST", "/start-alarm", true);
    xhttp.setRequestHeader("Content-Type", "application/json;charset=UTF-8");
    xhttp.send(JSON.stringify({ "oraAlarma":time}));

}
function stopPomodoroAlarm() {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            stopCount();
        }
    };

    xhttp.open("POST", "/stop-alarm", true);
    xhttp.send();
}
var interval;
var alarm_set=false;
function startCount(time)
{
    if(alarm_set)
    {
        stopCount();
    }
    alarm_set=true;
    var start_date= new Date();
    var countDownDate = new Date(start_date.getTime()+time*60000)
    x = setInterval(function() {

        // Get today's date and time
        var now = new Date().getTime();

        // Find the distance between now and the count down date
        var distance = countDownDate - now;

        // Time calculations for days, hours, minutes and seconds
        var minutes = Math.floor((distance % (1000 * 60 * 60)) / (1000 * 60));
        var seconds = Math.floor((distance % (1000 * 60)) / 1000);

        // Display the result in the element with id="demo"
        document.getElementById("count_down").innerHTML = "Au mai rămas: "+ minutes + "m " +
seconds + "s "+"până la alarma.";

        // If the count down is finished, write some text
        if (distance < 0) {
        stopCount();
        }
      }, 1000);
}

function stopCount()
{
    clearInterval(x);
    document.getElementById("count_down").innerHTML = "";
    alarm_set=false;
}
```

Index.html

This is the web interface.

```html
<!DOCTYPE html>
<html lang="ro-RO">
<head>
  <meta charset="UTF-8">
  <title>Smart Desk</title>
  <!-- <script src="{{ url_for('static', filename='script.js') }}"></script> -->
  <script src="/static/script.js"></script>
 </head>

<body style="margin:0px; padding-left: 5px;padding-right: 5px;" >

<div style="text-align:center;background-color: #C9DC87; margin: 0%;">

  <h1 style="color:#006400;margin-top: 0px; padding-top: 5px;">Smart desk</h1>
  <br/>
</div>

<hr>
<main style="padding: 3%;padding-top: 0%;">
<section class="light" style="color:#006400; padding-left: 20px">
  <h2>Iluminare birou:</h2>

  <div class="manuala" style="padding-left: 20px;">
    <p style="font-family: 'Times New Roman'; font-size: 20px">Manual:</p>
    <button style="height: 30px; margin-right: 20px" onclick="startManualLight()"
>Aprindere</button>
    <button style="height: 30px; margin-left: 20px" onclick="stopManualLight()"
>Stingere</button>

  </div>

  <div class="automata" style="padding-left: 20px;">
    <p style="font-family: 'Times New Roman'; font-size: 20px">Automat:</p>
    <button onclick="startAutoLight()" style="height: 30px;">Pornire</button>

  </div><br/>

</section>
```

```html
<hr>

<section class="air" style="color:#006400; padding-left: 20px">
   <h2>Ventilaie birou:</h2>
    <div class="manuala" style="padding-left: 20px;">
      <p style="font-family: 'Times New Roman'; font-size: 20px">Manual:</p>
     <button onclick="startManualAir()" style="height: 30px; margin-right: 20px">Pornire</button>
      <button onclick="stopManualAir()" style="height: 30px;  margin-left: 20px">Oprire</button>

   </div>

   <div class="automata" style="padding-left: 20px;">
      <p style="font-family: 'Times New Roman'; font-size: 20px">Automat:</p>
      <button onclick="automatedAir()" style="height: 30px;">Pornire</button>

   </div><br/><br/>
</section>

<hr>

<section class="alarm" style="color:#006400; padding-left: 20px">
   <h2>Fii productiv! Folosete tehnica pomodoro i seteaz o alarm</h2>
     <label for="oraAlarma" style="font-family: 'Times New Roman'; font-size: 18px">Introducei
timpul (in minute) dupa care va suna alarma:</label>
      <input type="number" name="oraAlarma" id="oraAlarma" value="25"><br/><br/>

      <button onclick="startPomodoroAlarm()"  style="height: 30px; margin-right: 20px" >Pornete
alarma</button>



   <button onclick="stopPomodoroAlarm()" style="height: 30px; margin-left: 20px" >Oprete
alarma</button>
<br/><br/>
<div id="counter">
   <h2 id="count_down" name="count_down"></h1>
</div>


</section>
```

```
<hr>
</main>
<footer style="background-color: #C9DC87;color:#006400;font-size: 18px;padding-left:
3%;padding-right: 3%;margin: 0%;">
   <p>Proiect realizat de: Atomei Georgiana, Bacica Florin, Ciulei Virgil, Turcas Irina</p>
</footer>
</body>
</html>
```

Link proiect hackster