Rīgas 64. vidusskola

CO2 datu padošana ar E-pastu

Eksāmena piekļuves darbs programmēšanā

Darba autors: Ralfs Varess

Rīga, 2023

Saturs

Ievads

Mūsdienās daudziem ir nācies saskarties ar sajūtām, ka iekštelpās trūkst gaisa, nav ko elpot, parādās noguruma vai smaguma sajūta un tādēļ rodas grūtības produktīvi strādāt vai ilgi atrasties tādās telpās. Bieži vien šo sajūtu cēlonis ir nepietiekama telpu ventilācija, vai arī šajā telpā nesen ir uzturējušies daudz cilvēku, kuri izelpojuši skābekli un atstājuši daudz oglekļa dioksīdu jeb CO2. Pastāv vairāki CO2 mērīšanas sensori, kur viens no populārākajiem ir NDIR (nondispersive infared), bet arī elektroķīmiskie un pusvadītāju sensori tiek plaši izmantoti.

Viens no populārākajiem viedokļiem sabiedrībā ir tāds, ka telpas pēc iespējas vairāk jāvēdina, jo neventilētās telpās rodas mitruma, pelējuma un bezgaisa koncentrācija. Šis veids tomēr nenodrošina pietiekamu un regulāru gaisa apmaiņu un gaisa mitruma stabilitāti, turklāt tas ir energoneefektīvs risinājums. Telpu vēdināšana patērē lielu daudzumu siltumenerģijas, līdz ar to, apkures izmaksas ir lielākas.

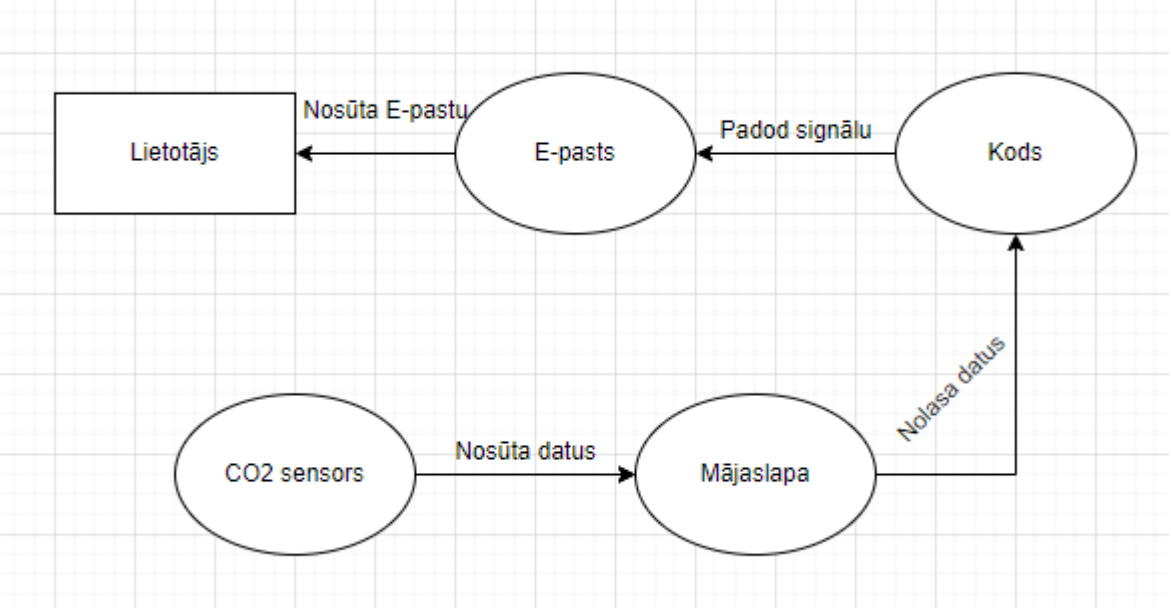
Viens no svarīgākajiem uzdevumiem skolās ir panākt, lai bērni neizjustu svaiga gaisa trūkumu. Pašreizējā situācija nešķiet pārlieku optimistiska. Aukstajā gadalaikā dažādu iemeslu dēļ skolās bieži netiek nodrošināta pietiekama ventilācija. Vecajām skolām problēma bieži slēpjas nepietiekami izbūvētās vai slikti funkcionējošās ventilācijas sistēmās, kur ventilācija tiek veikta tikai starpbrīžos, atverot logus, ar ko nepietiek nepieciešamajai gaisa apmaiņai un svaigā gaisa pieplūdes daudzuma nodrošināšanai. Siltajā gadalaikā ir iespējams logus turēt vaļā daudz vairāk vai pat pastāvīgi, tādējādi vismaz daļēji problēmu atrisinot. Savukārt jaunuzbūvētajās vai renovētajās skolās mēdz gadīties situācijas, kad uzstādītās ventilācijas iekārtas netiek darbinātas to dārgo izmaksu dēļ.

1. Problēmas izpēte un analīze

Skolā tika veikti novērojumi, ņemot datus no mājaslapas, kur tika izsekoti rādījumi CO2 izmaiņām telpā.

Gatavojoties darba uzsākšanai, tIka apskatīti skolas uzstādītie CO2 sensoru mājaslapas dati un tika konstatēts veselībai bīstamais CO2 līmenis. Ņemot vērā konstatētos datus padziļināti izpētījām to ietekmi uz cilvēka veselību. Pārmērīgs CO2 līmenis telpā var negatīvi ietekmēt cilvēku veselību un labklājību, tāpēc ir svarīgi būt spējīgiem precīzi mērīt CO2 līmeni iekštelpās un nodrošināt pareizu gaisa ventilāciju. Ja nebūtu pieeja šāda veida informācijai, tas nemotivētu cilvēkus veikt nepieciešamās izmaiņas un pievērst uzmanību savai veselībai.

2. Programmatūras prasību specifikācija.



1. attēls. Signāla padošanas diagramma

Noteiktās problēmas mērķauditorija būtu skolēni un skolotāji, jo ir svarīgi nodrošināt drošu un veselīgu skolas vidi, jo daudziem ir nācies saskarties ar sajūtu, ka iekštelpās trūkst gaisa, nav ko elpot, parādās noguruma vai smaguma sajūta un tādēļ rodas grūtības produktīvi strādāt vai ilgi atrasties tādās telpās.

Izveidotais kods nolasa datus no mājaslapas un nosūta e-pasta paziņojumu lietotājam, ka CO2 līmenis telpā ir pārsniedzis uzstādīto normas līmeni. Ir nepieciešami sekojoši līdzekļi: uzstādīts CO2 sensors telpā, mājaslapa, no kuras tiek nolasīti dati un dators, kurš pēc noteikta laika palaiž kodu un brīdina lietotāju par augsto CO2 līmeni.

E-pasta apstrādes modelis sastāv no divām funkcijām – datu pārbaudes un e-pasta sūtīšanas funkcijām.

Datu pārbaudes funkcijas apraksts

Tabula 1.

Mērķis:
Nolasīt vajadzīgos datus no mājaslapas.
Ievaddati:
Kods filtrē nepieciešamos datus.
Apstrāde:
Kods no mājaslapas nolasa tikai CO2 datus un pārējiem datiem nepievērš uzmanību (temperatūra, mitrums).
Izvaddati:
Pārbauda CO2 līmeņa uzstādīto robežu.

E-pasta sūtīšanas funkcijas apraksts

Tabula 2.

Mērķis:
Atlasīt gadījumus, kuros tiek nosūtīts e-pasts.
Ievaddati:
CO2 Datu nolasīšana ar kodu.
Apstrāde:
Kods pārbauda vai nav pārsniegta uzstādītā robeža.
Izvaddati: • **Ja netiek pārsniegta robeža:** Nesūta e-pastu. • **Ja tiek pārsniegta robeža:** Nosūta e-pastu lietotājam, kur ir teikts, lai atver logus.

3. Programmatūras izstrādes plāns

3.1. Izstrādātā koda uzturēšana

Galvenais uzdevums ir brīdināt lietotāju par to, ka telpā ir konstatēts bīstams CO2 līmenis. Programmas kodam ir jābūt labi strukturētam, komentētam un viegli uztveramam.

3.2. Izvietojamība

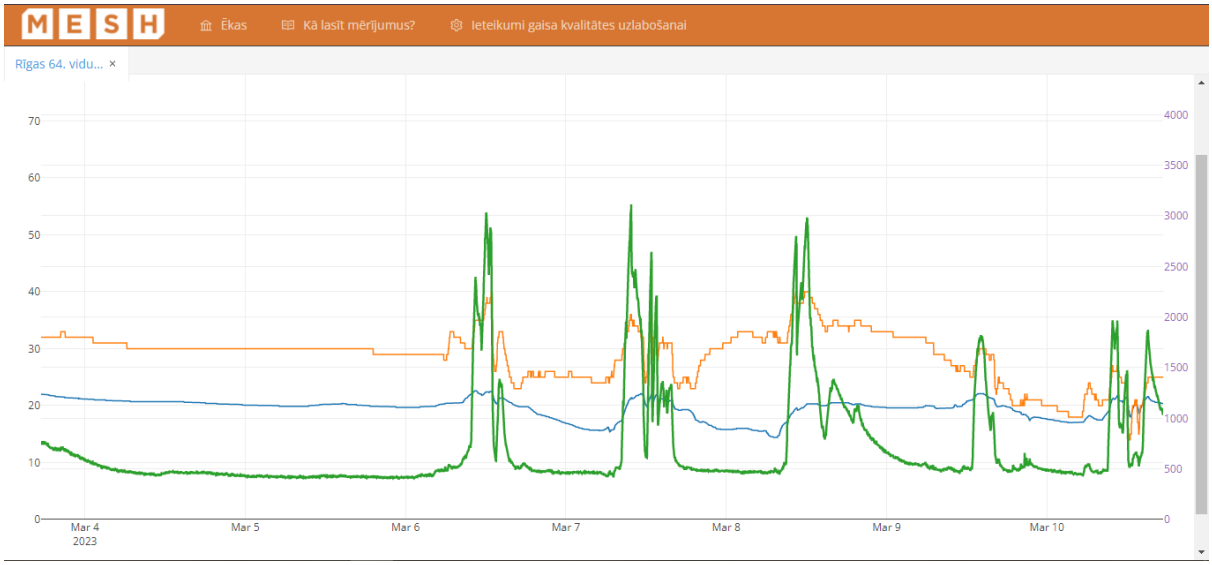
Kodu var rediģēt vai modificēt pielāgojot to citiem nolūkiem.

3.3. CO2 sensors:

Galvenais uzdevums ir mērīt CO2 līmeni, mitrumu un temperatūru telpā. Vidējais sensora dzīves ilgums ir 5 gadi.

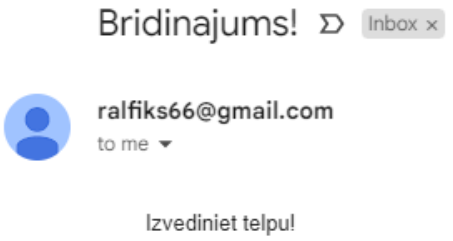
4. Atkļūdošanas un akcepttestēšanas pārskats

Testējot, vai izveidotais kods ir spējīgs nosūtīt e-pasta paziņojumu laikā, kad tiek pārsniegts CO2 bīstamības līmenis, tika izvērtēts kādu CO2 līmeni jāuzliek, lai pārliecinātos, ka izveidotais kods funkcionē.



2. attēls. Mājaslapas grafiks

Kad pārliecinās, ka kods nostrādā testa režīmā, tad tiek samainīts uz īsto CO2 līmeni.

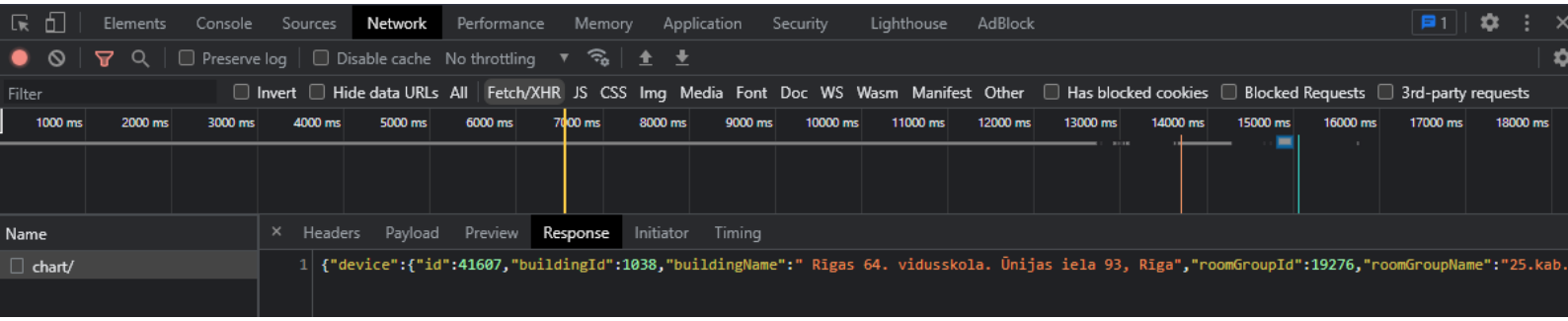


3. attēls. Saņemtais E-pasts

5. Lietotāja ceļvedis.

5.1. Network response iegūšana

Lai nolasītu no mājaslapas attēlotos ierīces datus, ir jāpiekļūst pie “Network response”, kurā glabājās ierīces veiktie mērījumi.



4. attēls. Mājaslapas "Network response"

Tiek ievadīta mājaslapa, no kuras tiek vākti dati, un jāgaida piecas sekundes (sleep(5)), lai veiktu “Xhr” pieprasījumu, kad tiek pārsūtīti dati starp tīmekļa pārlūkprogrammu un serveri.

```
driver.get("https://co2.mesh.lv/home/device-charts/41607")
sleep(5)
```

5. attēls, XHR

5.2. Jaunāko datu atlase

Izprintējot “Network response”, tiek iegūtas divas vārdnīcas, no kurām viena ir ar nosaukumu “device”(base64Encoded), kura satur ierīces informāciju, ierīces id, kurā kabinetā un stāvā atrodas, kas nav vajadzīgs. Otrajā vārdnīcā ar nosaukumu “data” glabājas ierīces, veiktie mērījumi, kā arī temperatūra, CO2, mitrums un laiks, kuri tiek atzīmēti kā “key” vērtības. Šajās “key” vērtībās tiek glabāti dati, kurus atzīmējam kā “value”. Izvēlamies vajadzīgās “key” vērtības un atlasam visjaunākos ierīces mērījumus, kuri vēlāk tiek apstrādāti:

```
new_data = data_dict
new_data.pop('base64Encoded', None)
for key, value in new_data.items():
    res_data = value

# Dictionary iteration
res_data = json.loads(res_data)
for key, value in res_data.items():
    for k, v in value.items():
        if k == 'co2':
            latest_co2 = v[-1]
        if k == 'timestamp':
            latest_time = v[-1]

print(latest_co2, latest_time)
```

6. attēls Vārdnīcas

5.3. Signāla nodošana

Lai informētu par norādītā CO2 līmeņa pārsniegšanu, izvēlējāmies izveidot kodu, kas automātiski izsūta e-pasta vēstuli. Tas tiek panākts, kodam nolasot, vai jaunākā CO2 vērtība pārsniedz uzstādīto vērtību, ja pārsniedz konkrēto vērtību, kura šajā gadījumā ir 2250 ppm, jo intervālā no 2000-3000 ppm gaisa kvalitāte ir vērtējama kā pasliktinājusies, un, iespējams, var samazināties cilvēku kognitīvās spējas, tiek aizsūtīts e-pasts ar ziņu atvērt logus.

```
if latest_co2 >= 2250:
    email_sender = 'ralfiks66@gmail.com'
    email_password = 'biqysqfaxijvsjon'

    email_receiver = 'ralfiks6@gmail.com'

    subject = "Bridinajums!"
    body = """
    Izvediniet telpu!
    """

    em = EmailMessage()
    em['subject'] = subject
    em['From'] = email_sender
    em['To'] = email_receiver
    em.set_content(body)

    context: SSLContext = ssl.create_default_context()

    with smtplib.SMTP('smtp.gmail.com', 587) as smtp:
        smtp.starttls()
        smtp.login(email_sender, email_password)
        smtp.sendmail(email_sender, email_receiver, em.as_string())
    print("Apskaties epastu")
```

7. attēls E-pasta nosūtīšana

5.4. Cikla izveide

Lai kods nepārtraukti strādātu tika izvedots cikls, kur ik pēc divām minūtēm palaistos atkal, ar iespēju lietotājam manuāli apstādināt ciklu ievadot “stop”, kas apstādinātu ciklu:

```
input_value = input("Enter 'stop' to end the loop: ")
#Delay
if input_value == 'stop':
    break
sleep(120)
```

8. attēls Cikls

6. Piemērotās licences pamatojums

Manuprāt, piemērojamā licence būtu EULA (End-User License Agreement) t.i Galalietotāja licences līgums, kurš paredz, ko lietotājs drīkst un ko nedrīkst darīt ar programmatūru un uzskaita nosacījumus, kad piekļuve var tikt ierobežota vai pārtraukta, jo ir svarīgi aizsargāt īpašnieka tiesības.

Secinājumi

Rakstot darbu, tika apgūta metodika par datu “Webscrapping”, e-pastu sūtīšanu un ciklu izveidi ar “Python” palīdzību. Autors guva pārliecību, ka izveidotais kods palīdzēs nodrošināt kvalitatīvu gaisa stāvokli skolēniem un skolotājiem, tādējādi nodrošinot ērtu un drošu mācību vidi.

Literatūras saraksts

1. Bīstams CO2 un mitruma līmenis telpās. [Tiešsaiste] Pieejams: https://ecovent.lv/bistams-co2-un-mitruma-limenis-telpas/
2. Svaiga gaisa trūkums un palielināts CO2 līmenis telpās – kā cīnīties? [Tiešsaiste] Pieejams: https://majaelpo.lv/blogs/ieteiktie-blogi/svaiga-gaisa-trukums-un-palielinats-co2-limenis-telpas-ka-cinities
3. Informācijas un komunikācijas tehnoloģiju pamatjēdzieni [Tiešsaiste] Pieejams: https://profizgl.lu.lv/mod/book/view.php?id=22319&chapterid=6896

Pielikums

```
       from ssl import SSLContext
from time import sleep
import json
from selenium import webdriver
from selenium.webdriver import DesiredCapabilities
import smtplib
from email.mime.text import MIMEText
from email.message import EmailMessage
import ssl

# make chrome log requests
while True:
    capabilities = DesiredCapabilities.CHROME
    capabilities["goog:loggingPrefs"] = {"performance": "ALL"}  # newer:
goog:loggingPrefs
    driver = webdriver.Chrome(
        desired_capabilities=capabilities, executable_path="./chromedriver"
    )
    # fetch a site that does xhr requests
    driver.get("https://co2.mesh.lv/home/device-charts/41607")
    sleep(5)

    # extract requests from logs
    logs_raw = driver.get_log("performance")
    logs = [json.loads(lr["message"])["message"] for lr in logs_raw]

    def log_filter(log_):
        return (
            # is an actual response
                log_["method"] == "Network.responseReceived"
                # and json
                and "json" in log_["params"]["response"]["mimeType"]
        )
    for log in filter(log_filter, logs):
        request_id = log["params"]["requestId"]
        resp_url = log["params"]["response"]["url"]
        print(f"Caught {resp_url}")
        # print(driver.execute_cdp_cmd("Network.getResponseBody",
{"requestId": request_id}))
        data_dict =
driver.execute_cdp_cmd("Network.getResponseBody",{"requestId": request_id})

    # df = pd.DataFrame(data = data dict, index =[0])
    # df = (df.T)
    print('izdarīts')

    new_data = data_dict
    new_data.pop('base64Encoded', None)
    for key, value in new_data.items():
        res_data = value

    # Dictionary iteration
    res_data = json.loads(res_data)
    for key, value in res_data.items():
        for k, v in value.items():
            if k == 'co2':
                latest_co2 = v[-1]
            if k == 'timestamp':
                latest_time = v[-1]
```

```
print(latest_co2, latest_time)
#Alert the user
if latest_co2 >= 2250:
    email_sender = 'ralfiks66@gmail.com'
    email_password = 'biqysqfaxijvsjon'

    email receiver = 'ralfiks6@gmail.com'

    subject = "Bridinajums!"
    body = """
    Izvediniet telpu!
    """

    em = EmailMessage()
    em['subject'] = subject
    em['From'] = email_sender
    em['To'] = email_receiver
    em.set_content(body)

    context: SSLContext = ssl.create_default_context()

    with smtplib.SMTP('smtp.gmail.com', 587) as smtp:
        smtp.starttls()
        smtp.login(email_sender, email_password)
        smtp.sendmail(email_sender, email_receiver, em.as_string())
    print("Apskaties epastu")
input_value = input("Enter 'stop' to end the loop: ")
#Delay
if input_value == 'stop':
    break
sleep(120)
```