Bezbedna kuća

Janko Vasić 122/2021RI

1. Opis sistema
2. Arhitektura sistema
3. Skladište podataka
4. Web servis
5. Korisnička aplikacija
6. Uredjaj
7. Prilozi

Opis sistema

Šta sistem modeluje:

Sistem modeluje uređaje za detekciju dima i plamena, koji služe za rano otkrivanje požara i obaveštavanje korisnika.

Organizacija i implementacija:

Uređaji za detekciju: Periodično se javljaju serveru da su aktivni i mere koncentraciju dima.
Firebase: Skladišti podatke o očitavanjima senzora.
Web servis: Prikazuje trenutno stanje uređaja i aktivira alarm kada je koncentracija dima veća od praga.
Thonny: Lokalno pali zujalicu ako se prekorači prag koncentracije dima.
Funkcionalnosti:

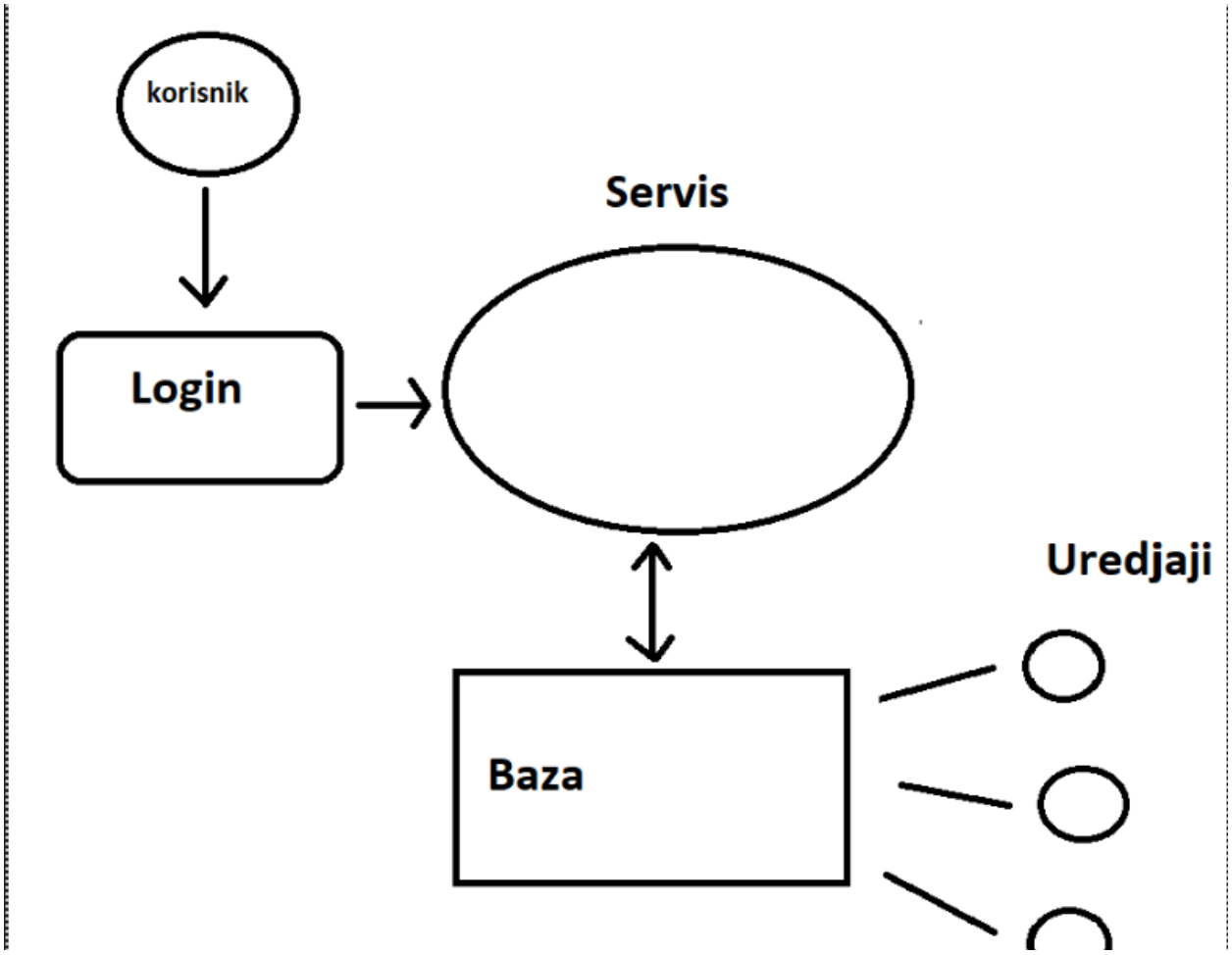
Detekcija požara: Aktiviranje alarma i slanje obaveštenja kada se otkrije požar.

Provera statusa uređaja: Prikaz poslednjeg javljanja uređaja (da li su "živi").

Očitavanje parametara senzora: Prikaz trenutnih očitavanja senzora na zahtev korisnika.

Arhitektura sistema

Od elemenata postoje Korisnik, Login, Servis, Baza, Uredjaji,



korisnik komunicira sa sistemom preko web aplikacije, login je modul za autentifikaciju korisnika, servis komunicira sa bazom i sluzi za prikaz aplikacije, baza skladišti podatke o uredjajima i uredjaji su u suštini senzori dima koji salju podatke.

Cloud: Servis i baza podataka hostovani na cloud platformi

Lokalni računari/uređaji: Uređaji za detekciju dima i zujalica

Programski jezici: Python (MicroPython), JavaScript, HTML

Skladište podataka

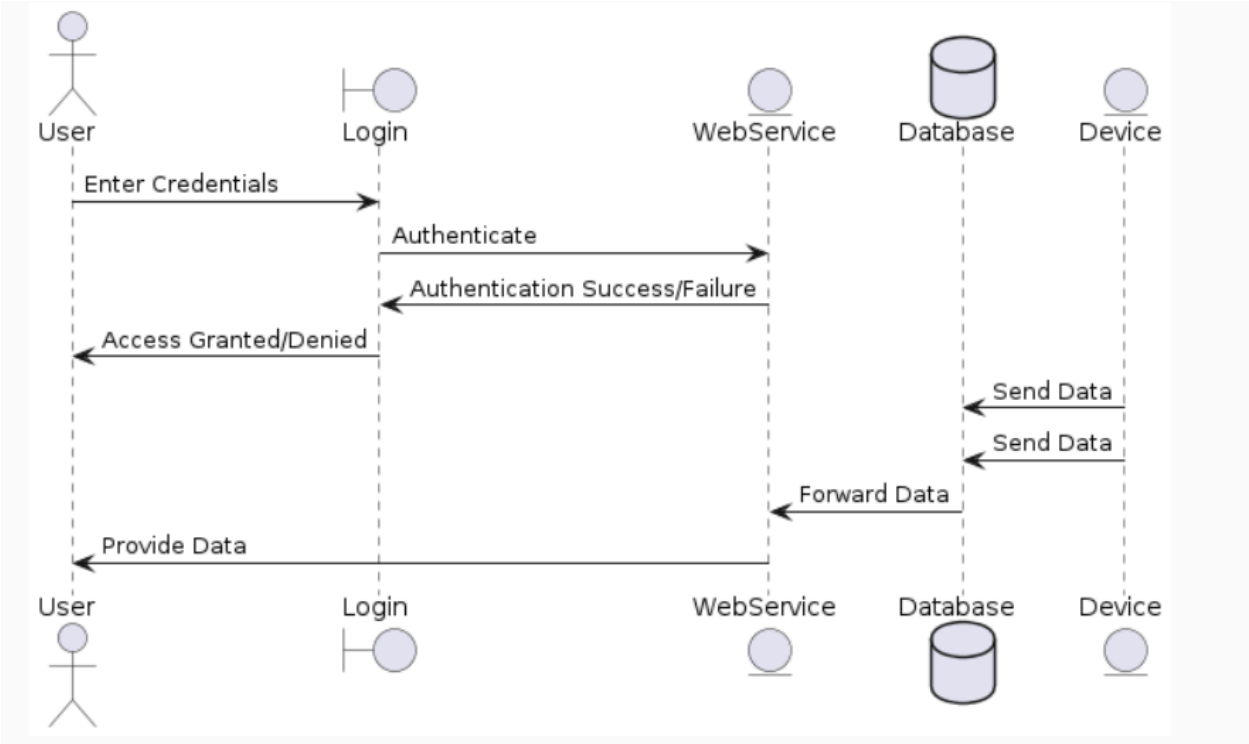
Za skladište podataka sam koristio Firebase Realtime Database.

Najbitniji parametri su URL baze, Auth token i struktura podataka.

URL: https://rpi1-5529c-default-rtdb.europe-west1.firebasedatabase.app/

Auth: 6qU01SWkIeIBhJwmKyn2UHknZXWC0q7CUe4RCWwY

Za strukturu podatka se korsiti JSON format.



Web servis

Što se tehnologija tiče koristio sam firebase za čuvanje podataka, HTML, JS.

Funkcionalnost:

Prikazivanje podataka o količini gasa, nema eksplicitnih ruta, već se podaci periodično čitaju iz Firebase baze. Prikaže se trenutna količina gasa u kući. Ažurira se automatski kada se novi podaci zapišu u Firebase.

Registracija uređaja, /registerDevice, POST zahtev sa validnim `deviceId` koji se šalje na Firebase bazu, uspešna registracija vraća poruku o uspehu, a neuspešna prikazuje grešku.

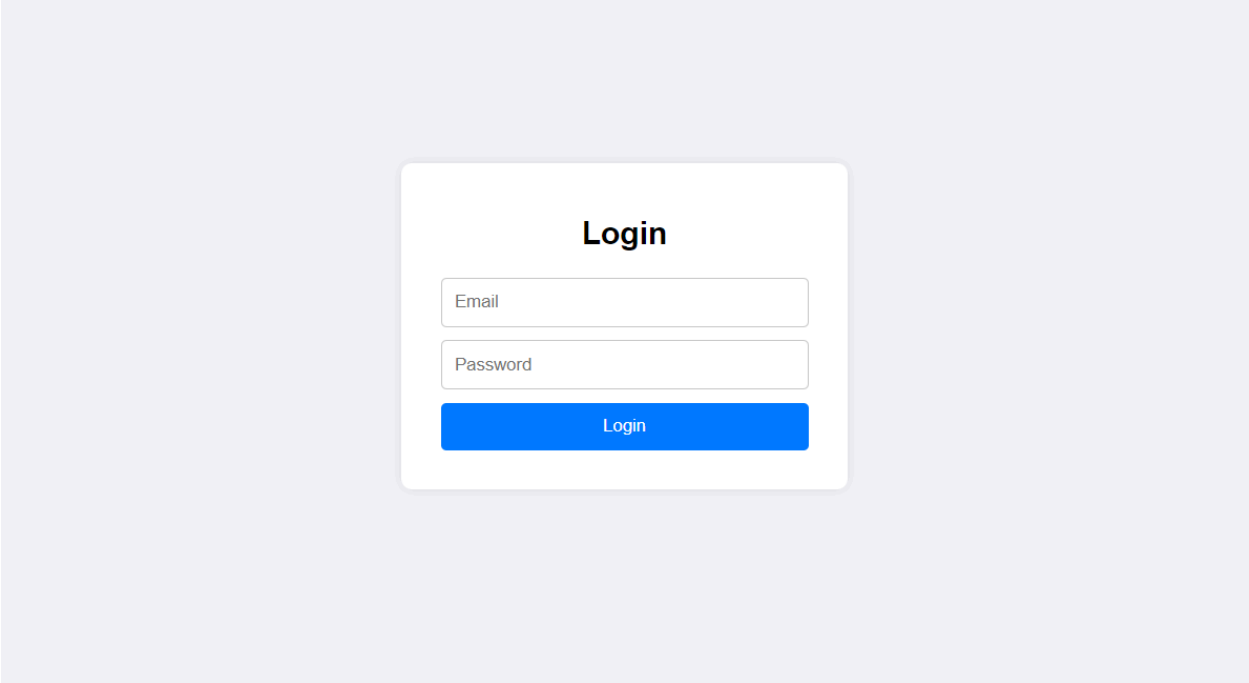
Korisnička aplikacija

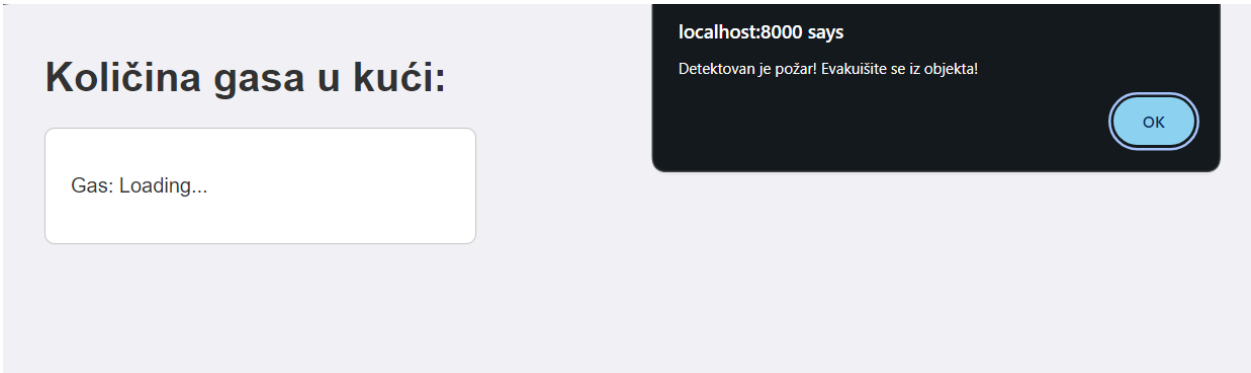
U fetchData() prikupljamo podataka o koncentraciji gasa iz baze podataka i ažuriramo prikaz na osnovu najnovijih podataka.

Ako koncentracija gasa pređe određeni prag, aplikacija šalje notifikaciju korisniku pomoću `sendNotification()`.

`Funkcija updateDisplay() ažurira korisnički interfejs sa najnovijim podacima o koncentraciji gasa.`

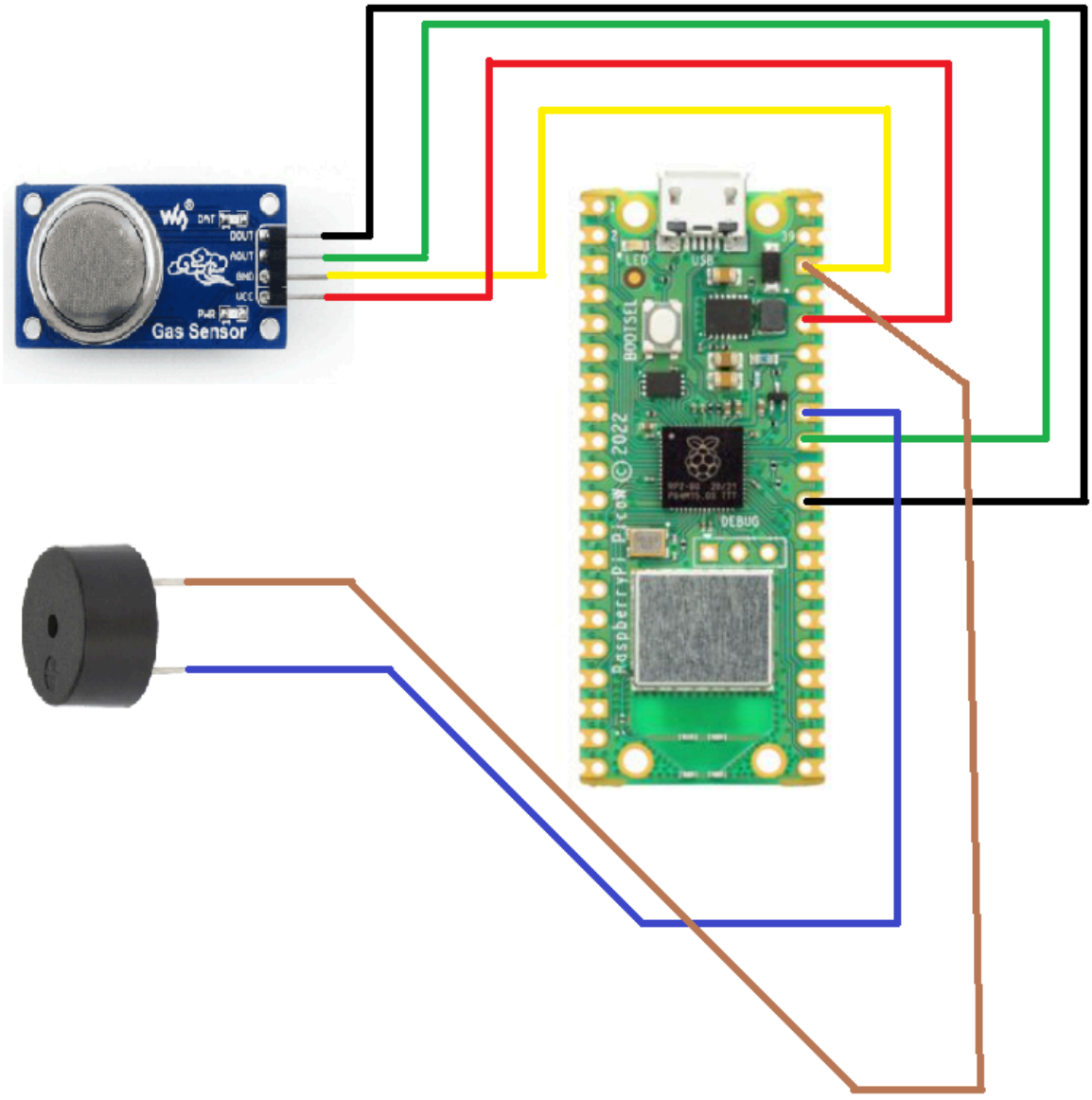
`Funkcija registerDevice() omogućava korisnicima da registruju nove uređaje u sistemu.`





UREĐAJ

Uređaj je dizajniran za merenje koncentracije gasova u vazduhu pomoću MQ-135 senzora. Kada se detektuje visoka koncentracija gasa, aktivira se alarm putem buzzera. Uređaj takođe šalje podatke o koncentraciji gasa na Firebase server radi praćenja u realnom vremenu.



PRILOZI

```
from time import sleep
import urequests as requests
import ujson
import network
import socket
from machine import Pin, ADC
import machine

# Initialize ADC and analog pin for MQ-135 sensor
analog_pin_gas = ADC(Pin(26))

# Initialize buzzer pin
buzzer = Pin(27, Pin.OUT)

ssid = 'balsa'
password = 'dnsz7317'


firebase_url = "https://rpi1-5529c-default-rtdb.europe-west1.firebasedatabase.app/"
auth_token = "6qU01SWkIeIBhJwmKyn2UHknZXWC0q7CUe4RCWwY"

def connect():
    # Connect to WLAN
    wlan = network.WLAN(network.STA_IF)
    wlan.active(True)
    wlan.connect(ssid, password)
    while not wlan.isconnected():
        print('Waiting for connection...')
        sleep(1)
    ip = wlan.ifconfig()[0]
    print(f'Connected on {ip}')
    return ip
```

```
def open_socket(ip, port=8080):
    # Open a socket
    address = (ip, port)
    connection = socket.socket()
    connection.bind(address)
    connection.listen(1)
    print(connection)
    return connection

def get_device_id():
    id = ""
    for b in machine.unique_id():
        id += "{:02X}".format(b)
    return id
device_id = get_device_id()
def mainloop(connection):
    # Your main loop code here
    pass

def read_gas():
    # Read analog value from MQ-135 sensor
    analog_value_gas = analog_pin_gas.read_u16()

    # Convert analog value to gas concentration (adjust conversion formula based on sensor characteristics)
    gas_concentration = analog_value_gas  # Adjust conversion formula based on sensor characteristics

    return int(gas_concentration)  # Ensure the gas concentration is an integer

try:
    ip = connect()
    connection = open_socket(ip)
    mainloop(connection)
except KeyboardInterrupt:
```

```
    machine.reset()


while True:
    # Read gas concentration from sensor
    gas_concentration = read_gas()

    # Data to send to Firebase
    data = { "device_id": device_id,
             "gas_concentration": gas_concentration}
    json_data = ujson.dumps(data)

    # Construct URL for the Firebase endpoint
    endpoint = firebase_url + "baza.json"

    # If authentication token is provided, append it to the URL
    if auth_token:
        endpoint += "?auth=" + auth_token


    if gas_concentration > 8000:
        buzzer.on()  # Turn on the buzzer
    else:
        buzzer.off()  # Turn off the buzzer if gas concentration is below 2000

    # Make a POST request to Firebase
    response = requests.post(endpoint, data=json_data)

    # Check if the request was successful
    if response.status_code == 200:
        print("Gas concentration data sent successfully to Firebase")
```

```
# Check if the request was successful
if response.status_code == 200:
    print("Gas concentration data sent successfully to Firebase")
else:
    print("Failed to send gas concentration data to Firebase:", response.status_code)


# Close the response to free up resources
response.close()


# Wait for some time before sending the next data (adjust as needed)
sleep(1)
```