Dokumentacija implementacije

Uvod

Ovaj dokument opisuje implementaciju sistema za evidenciju prisustva studenata korišćenjem Raspberry Pi Pico W mikrokontrolera. Sistem uključuje generisanje Wi-Fi mreže, web interfejs za evidentiranje prisustva, tajmer za praćenje predavanja, LED indikatore za status tajmera, i upravljanje prisustvom studenata putem JSON datoteka.

Biblioteke

Kod koristi sledeće biblioteke:

- `network`: Za postavljanje Wi-Fi mreže.
- `machine`: Za interakciju sa hardverom kao što su GPIO pinovi i tajmeri.
- `time`: Za praćenje vremena i kašnjenja.
- `socket`: Za postavljanje servera koji sluša HTTP zahteve.
- `ure`: Za parsiranje HTTP zahteva.
- `json`: Za rad sa JSON podacima.
- `os`: Za rad sa sistemskim fajlovima.

Definicija Pinova

```
button_start_pin = machine.Pin(0, machine.Pin.IN,
machine.Pin.PULL_DOWN)
button_stop_pin = machine.Pin(1, machine.Pin.IN,
machine.Pin.PULL_DOWN)
timer = machine.Timer()

red_led_pin = machine.Pin(14, machine.Pin.OUT)
green_led_pin = machine.Pin(12, machine.Pin.OUT)
blue_led_pin = machine.Pin(13, machine.Pin.OUT)
```

Pinovi su definisani za start i stop dugmad, kao i za crvenu, zelenu i plavu LED diodu.

Konfiguracija Wi-Fi mreže

```
ssid = 'PRISUSTVO'
password = '123456789'
```

Konfiguriše se Wi-Fi mreža sa SSID-om `PRISUSTVO` i lozinkom `123456789`.

Inicijalizacija Podataka

```
connected_clients = set()
stop_time = False
start_time = None
elapsed_time = 0
timer_running = False
users = {
    "20230001": {"ime": "Marko", "prezime": "Marković", "lozinka":
"pass123", "index": "20230001", "ip_adresa": None, "prisustvo": False,
"vrijeme_osluskivanja": None, "ukupno_vrijeme_osluskivanja": None},
...
}
```

Promenljive za praćenje stanja sistema i evidenciju korisnika (studenta) se inicijalizuju.

Funkcija za Ažuriranje LED Dioda

```
def update_leds():
    global timer_running
    global stop_time
    global users

    if not timer_running:
        red_led_pin.on()
        green_led_pin.off()
        blue_led_pin.off()
    elif timer_running and not stop_time:
        red_led_pin.off()
        green_led_pin.on()
        blue_led_pin.off()
    elif timer_running and stop_time:
        red_led_pin.on()
        green_led_pin.on()
        blue_led_pin.off()
```

Ova funkcija ažurira stanje LED dioda u zavisnosti od stanja tajmera i prisustva.

Postavljanje Wi-Fi Access Point-a

```
ap = network.WLAN(network.AP_IF)
ap.config(essid=ssid, password=password)
ap.active(True)
```

Wi-Fi mreža se postavlja i aktivira.

Funkcije za Rukovanje Tajmerom

```
def timer_callback(timer):
    global elapsed_time
    elapsed_time += 1
```

Funkcija koja se poziva svakog sekunda za ažuriranje proteklog vremena.

Rukovanje Dugmadima

Start Dugme

```
def button_start_handler(pin):
    global timer_running
    global start_time
    global stop_time

    if not timer_running:
        timer_running = True
        start_time = time.time()
        timer.init(freq=1, mode=machine.Timer.PERIODIC,
callback=timer_callback)
    elif not stop_time:
        stop_time = True
        timer.deinit()
    elif timer_running:
        stop_time = False
        start_time = time.time()
        timer.init(freq=1, mode=machine.Timer.PERIODIC,
callback=timer_callback)
    update_leds()
```

Ova funkcija rukuje start dugmetom. Ako tajmer nije aktivan, počinje brojanje vremena. Ako je pauziran, nastavlja brojanje. Ako je aktivan, zaustavlja brojanje.

Stop Dugme

```
def button_stop_handler(pin):
    global timer_running
    global elapsed_time
    global users
    update_leds()

    for user_id in users:
        if users[user_id]["prisustvo"]:
            user_time = users[user_id]["vrijeme_osluskivanja"]
            if user_time is not None:
                time_diff = int(elapsed_time) - int(user_time)
                diff_hours = time_diff // 3600
                diff_minutes = (time_diff % 3600) // 60
                diff_secs = time_diff % 60
                users[user_id]["ukupno_vrijeme_osluskivanja"] =
"{:02d}:{:02d}:{:02d}".format(diff_hours, diff_minutes, diff_secs)
            else:
                users[user_id]["ukupno_vrijeme_osluskivanja"] = "N/A"


    timestamp = time.time()
    json_filename = "users_" + str(int(timestamp)) + ".json"


    json_data = json.dumps(users)

    formatted_json_data = json_data.replace(',', ',\n').replace('{',
'{\n').replace('}', '\n}')

    with open(json_filename, 'w') as json_file:
        json_file.write(formatted_json_data)

    for user_id in users:
        users[user_id]["ip_adresa"] = None
        users[user_id]["prisustvo"] = False
        users[user_id]["vrijeme_osluskivanja"] = None
        users[user_id]["ukupno_vrijeme_osluskivanja"] = None
    timer_running = False
    timer.deinit()
    elapsed_time = 0
```

Ova funkcija rukuje stop dugmetom. Zaustavlja tajmer i ažurira evidenciju prisustva studenata, čuvajući podatke u JSON fajlu.

Generisanje HTML Sadržaja

```
def generate_html():

    ...
```

Ova funkcija generiše HTML sadržaj koji se prikazuje na web interfejsu, uključujući listu prisutnih studenata i trenutno proteklo vreme.

Funkcije za Parsiranje POST Zahteva

Parsiranje Dodavanja Novih Korisnika

```
def parse_new_user_request(request):
    match =
ure.search(r"sifa_unosa_studenta=([^&]*)&new_name=([^&]*)&new_last_nam
e=([^&]*)&new_index=([^&]*)&new_password=([^&]*)", request)
    if match:
        return match.group(1), match.group(2), match.group(3),
match.group(4), match.group(5)
    return None, None, None, None, None
```

Funkcija za parsiranje POST zahteva za dodavanje novih korisnika.

Parsiranje Brisanja Korisnika

```
def parse_errase_user_request(request):
    match =
ure.search(r"sifa_brisanja_studenta=([^&]*)&errase_index=([^&]*)",
request)
    if match:
        return match.group(1), match.group(2)
    return None, None
```

Funkcija za parsiranje POST zahteva za brisanje korisnika.

Postavljanje i Slušanje na Socket-u

```
addr = socket.getaddrinfo('0.0.0.0', 80)[0][-1]
s = socket.socket()
s.bind(addr)
s.listen(1)
```

Socket se postavlja na port 80 i sluša zahteve.

Parsiranje Zahteva

```
def parse_request(request):
    match = ure.search(r"index=([^&]*)&password=([^&]*)", request)
    if match:
        return match.group(1), match.group(2)
    return None, None
```

Funkcija za parsiranje POST zahteva za prijavljivanje korisnika.

Provera Da li je IP Adresa U Upotrebi

```
def is_ip_in_use(ip):
    for index, user in users.items():
        if user["ip_adresa"] == ip:
            return True
    return False
```

Funkcija koja proverava da li je IP adresa već u upotrebi od strane nekog korisnika.

Glavna Petlja za Slušanje Zahteva

```
while True:

    ...
```

U glavnoj petlji, server prihvata klijentske zahteve, parsira ih i odgovara generisanim HTML sadržajem.

Funkcionalnost Projekta

1. **Wi-Fi mreža**: Generisanje Wi-Fi mreže sa SSID-om `PRISUSTVO` i lozinkom `123456789`.
2. **Web interfejs**: Prikaz login forme i lista prisutnih studenata.
3. **Upravljanje prisustvom**:
    - Start: Dugme za početak brojanja vremena.
    - Pauza: Dugme za zaustavljanje tajmera.
    - Kraj predavanja: Dugme za završetak brojanja vremena.
4. **LED indikatori**:
    - Crvena LED: Indikuje da predavanje nije započelo.
    - Zelena LED: Indikuje da je predavanje započelo.
    - Plava LED: Indikuje da je predavanje pauzirano.

5. **Evidencija prisustva**:
    - Prijava studenata putem login forme.
    - Merenje vremena prisustva.
    - Čuvanje podataka u JSON fajlove.
6. **Upravljanje podacima**:
    - Dodavanje novih studenata.
    - Brisanje studenata.
    - Sigurnost podataka.

Ova implementacija omogućava efikasno praćenje prisustva studenata tokom predavanja, omogućavajući profesorima jednostavno upravljanje podacima putem web interfejsa.