INTERNET STVARI

POROČILO PROJEKTA:

SmartFan – ventilator upravljan preko oblačne storitve

Maj Fontana Korošec

3. letnik, Telekomunikacije (UN), UM FERI

Asistent:

Klemen Bravhar, mag. inž. el.

Maribor, 12. 5. 2023

1. Kazalo vsebine

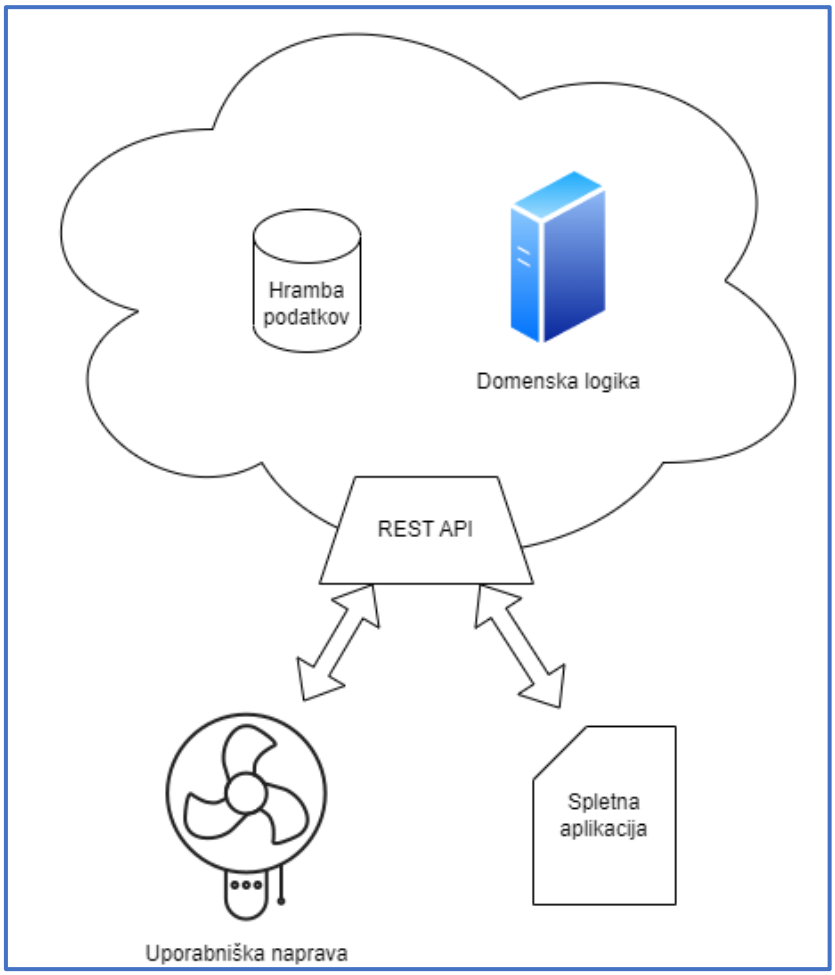
2. Kazalo slik

3. Zasnova

SmartFan je IoT sistem, ki uporabniku omogoča upravljanje ventilatorja preko spletne aplikacije. Uporabnik v aplikaciji nastavi željeno temperaturo v območju pred ventilatorjem, na podlagi katere mikrokrmilnik samodejno vklaplja in izklaplja ventilator. Sistem je možno tudi izklopiti. Uporabnikova naprava vsebuje tudi detektor prisotnosti uporabnika, ter ventilator izklopi, ko uporabnika ni pred napravo.

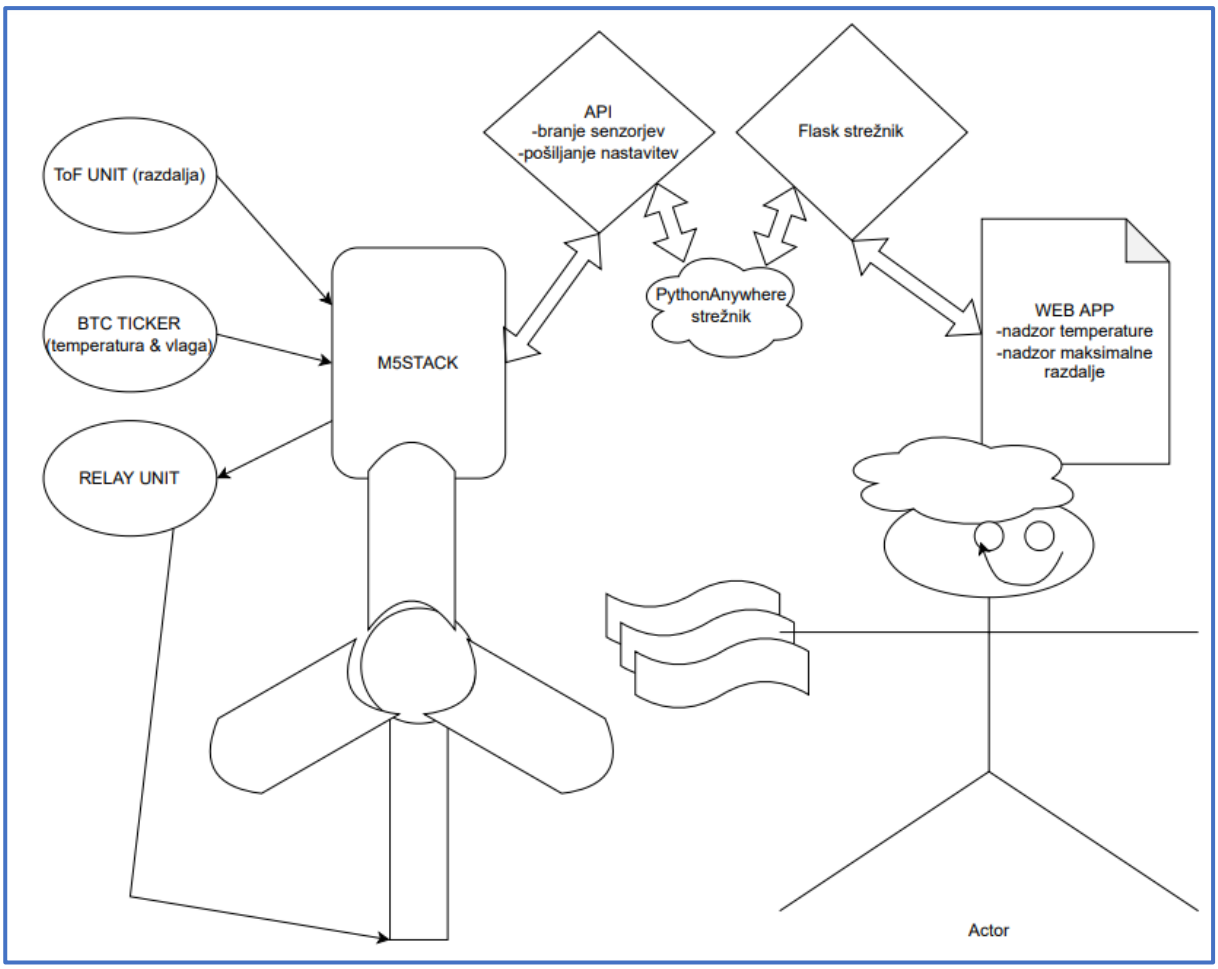
Pri načrtovanju smo sistem razdelili na tri dele:

- Uporabniška naprava – naprava, ki se nahaja pri uporabniku. Sestavljajo jo ventilator, senzorji, rele ter mikrokrmilnik.
- Back-end z REST API – strežnik, ki hrani in obdeluje stanje naprave ter uporabnikove nastavitve, ter jih ponuja uporabniški napravi ter spletni aplikaciji.
- Spletna aplikacija – uporabniški vmesnik, preko katerega uporabnik nadzira delovanje naprave.



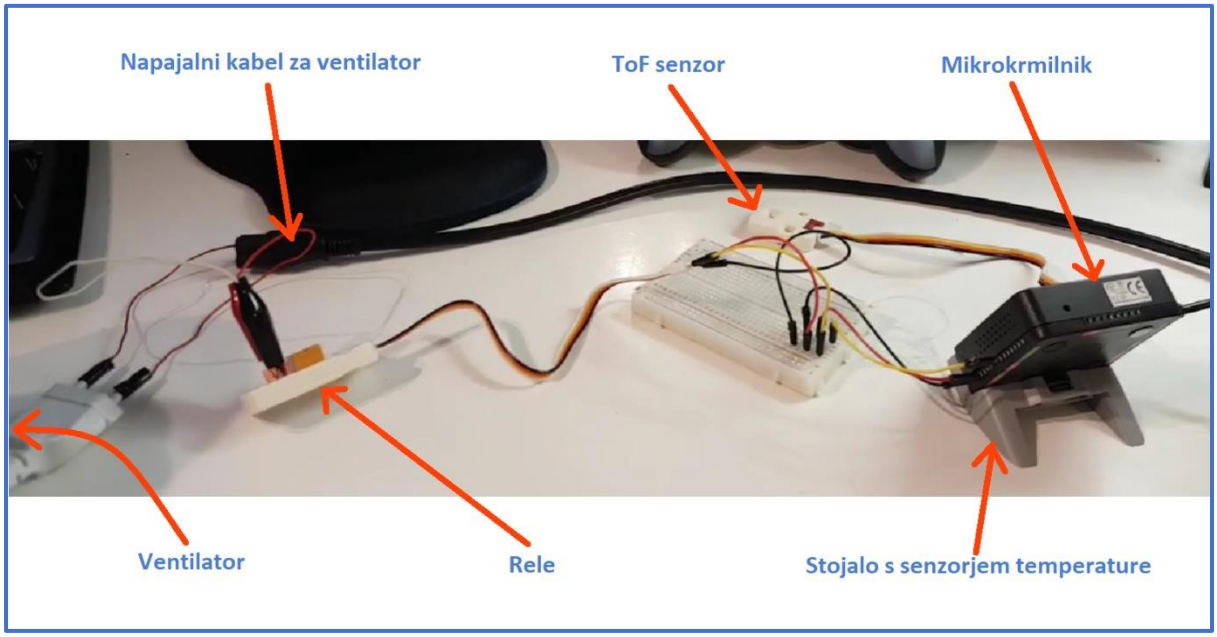
Slika 1: Konceptualna zasnova SmartFan sistema

4. Izvedba

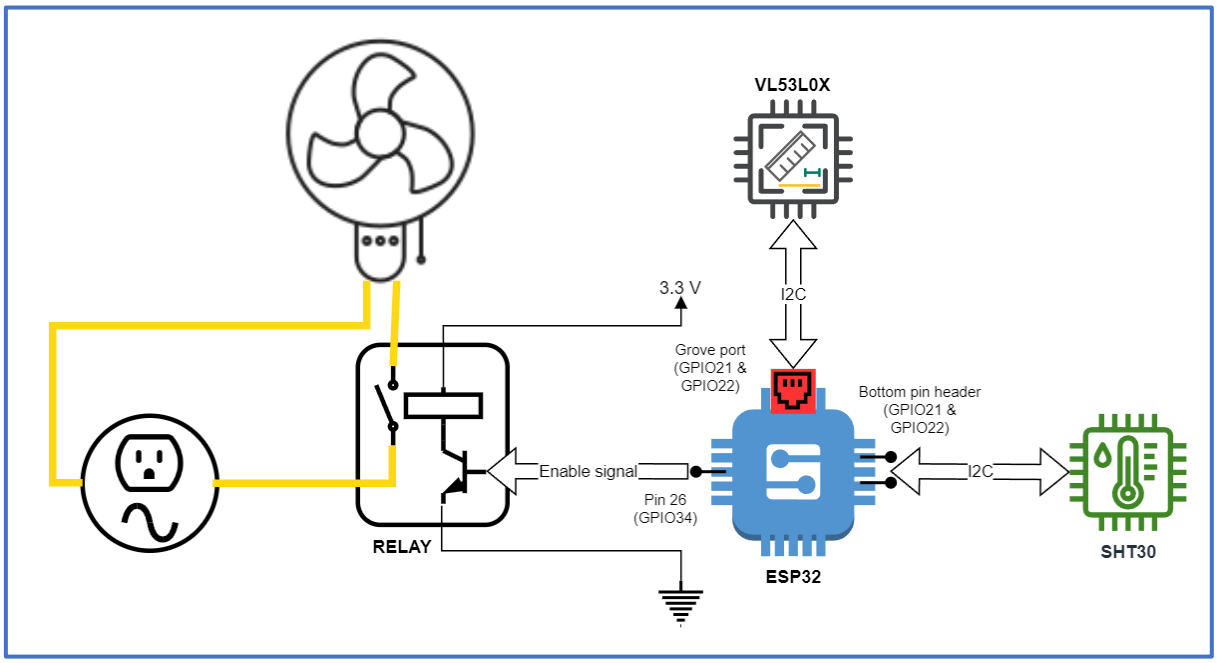


Slika 2: Prvotna verzija načrta implementacije

4.1. Uporabniška naprava



Slika 3: Prototip naprave



Slika 4: Diagram naprave

4.1.1. Senzorji in aktuatorji

Senzorje smo izbirali med M5Stack moduli s plastičnimi ohišji ter vgrajeno I2C komunikacijo. Knjižnice za komunikacijo s senzorji smo naložili preko Library Manager znotraj Arduino IDE.

Za nadzor temperature pred ventilatorjem smo izbrali senzor temperature vgrajen v BTC Ticker stojalo za M5 Basic Core IoT Development Kit. Gre za SHT30 senzor, s katerim smo komunicirali z uporabo knjižnice arduino-sht verzije 1.2.2 razvijalca Johannes Winkelmann. Ko se kit vstavi v stojalo, je senzor z mikrokrmilnikom povezan direktno preko SIP sponk vgrajenih v razvijalni kit.

https://shop.m5stack.com/products/btc-standing-base-for-m5-core-with-sht30

Za detekcijo prisotnosti uporabnika smo uporabili ToF modul za merjenje razdalje. Ta temelji na čipu VL53L0X, s katerim smo komunicirali z uporabo knjižnice VL53L0X verzije 1.3.1 razvijalca Pololu. Senzor je na mikrokrmilnik povezan s kablom preko Grove vtičnice na razvijalnem kitu.

https://shop.m5stack.com/products/tof-sensor-unit

Ventilator smo prižigali in ugašali z uporabo rele modula. Ta nima vgrajene I2C komunikacije, temveč se ga krmili direktno z digitalnim izhodom mikrokrmilnika. Krmilno linijo povezali na pin 26.

https://shop.m5stack.com/products/mini-3a-relay-unit

4.1.2. Mikrokrmilnik

Za krmiljenje smo uporabili M5Stack Basic Core IoT Development Kit, ki temelji na ESP32 mikrokrmilniku.

https://shop.m5stack.com/products/basic-core-iot-development-kit

Koda na mikrokrmilniku v rednih intervalih bere vrednosti iz senzorjev, se odloča o vklopitvi ali izklopitvi ventilatorja, posreduje stanje naprave na strežnik, ter iz strežnika pridobi trenutne nastavitve. Trenutno stanje naprave si hrani v strukturi Status, trenutne nastavitve pa v strukturi Settings. Za sledenje intervalom je implementiran razred Schedule, z metodo setInterval, ki nastavi želeno trajanje intervala, ter metodo poll, ki vrne true samo enkrat znotraj intervala. Redno se preverja tudi stanje WiFi povezave, ki se v primeru izpada poskuša ponovno vzpostaviti.

Mikrokrmilnik ventilator vklopi, ko so izpolnjeni naslednji pogoji:

1. sistem je v nastavitvah vključen,
2. izmerjena razdalja je manjša od konstantne mejne vrednosti ter
3. izmerjena temperatura je večja od željene temperature v nastavitvah.

Pri mejni vrednosti željene temperature se dodatno uporablja še histereza, prepogostim preklopom pa se izognemo tudi z dovolj velikim trajanjem intervalov.

4.2. Oblačna storitev

Za oblačno storitev je uporabljen Python Flask strežnik, postavljen v brezplačnem spletnem razvojnem okolju PythonAnywhere. Spletna aplikacija in back-end REST API sta v glavno Flask aplikacijo vgrajena z Blueprint objekti.

V času pisanja tega dokumenta je spletna aplikacija dosegljiva na spletnem naslovu:

https://lair.pythonanywhere.com/feri-is/smart-fan/dashboard/

Za namene testiranja in demonstracije je bila Arduino C++ koda prevedena v Python skripto, ki emulira delovanje mikrokontrolerja, in se lahko po potrebi zažene na običajnemu računalniku ali pametnemu telefonu.

4.2.1. Back-end z REST API

Back-end si stanje naprave in uporabnikove nastavitve hrani v objektih SmartFanStatus in SmartFanConfiguration, in omogoča branje ali prepis vsebine teh objektov preko REST API. Končni točki za vmesnik sta:

1. /device – branje in pisanje stanja naprave ter
2. /device/configure – branje in pisanje uporabnikovih nastavitev.

V času pisanja dokumenta sta končni točki dosegljivi na naslovih:

https://lair.pythonanywhere.com/feri-is/smart-fan/device/

https://lair.pythonanywhere.com/feri-is/smart-fan/device/configure/

Strežnik ob GET zahtevi vrne vsebino objekta, ob POST zahtevi pa vsebino zahteve razčleni, in z njo posodobi objekt. Ob zahtevah mora odjemalec podati tudi polje X-Api-Key z avtorizacijskim ključem. Posameznim ključem so dodeljene različne pravice za branje in modifikacijo omenjenih objektov.

Vsebina objektov se prenaša v JSON formatu. Struktura uporabniških nastavitev je:

```
{
    "target_temperature": number,
    "fan_enabled": boolean
}
```

Struktura stanja naprave pa:

```
{
    "temperature": number,
    "fan_active": boolean,
    "seconds_since_last_connection": number,
    "configuration":
    {
        "target_temperature": number,
        "fan_enabled": boolean
    }
}
```

seconds_since_last_connection je vrednost, ki jo izračuna strežnik za potrebe spletne aplikacije. Zato je to polje prisotno le ob branju stanja iz strežnika, ne pa tudi pri pisanju na strežnik.

4.2.2. Spletna aplikacija

Za spletno aplikacijo je bil izdelan minimalistični design v želji, da se s čim manj kompleksnosti doseže uporabniški vmesnik z modernim izgledom. Pri tem smo želeli uporabniku podati naslednje informacije:

- ali je naprava povezana na strežnik,
- trenutna temperatura,
- ali je ventilator vklopljen ter
- trenutne nastavitve.

Uporabnik mora imeti nadzor nad dvema nastavitvama:

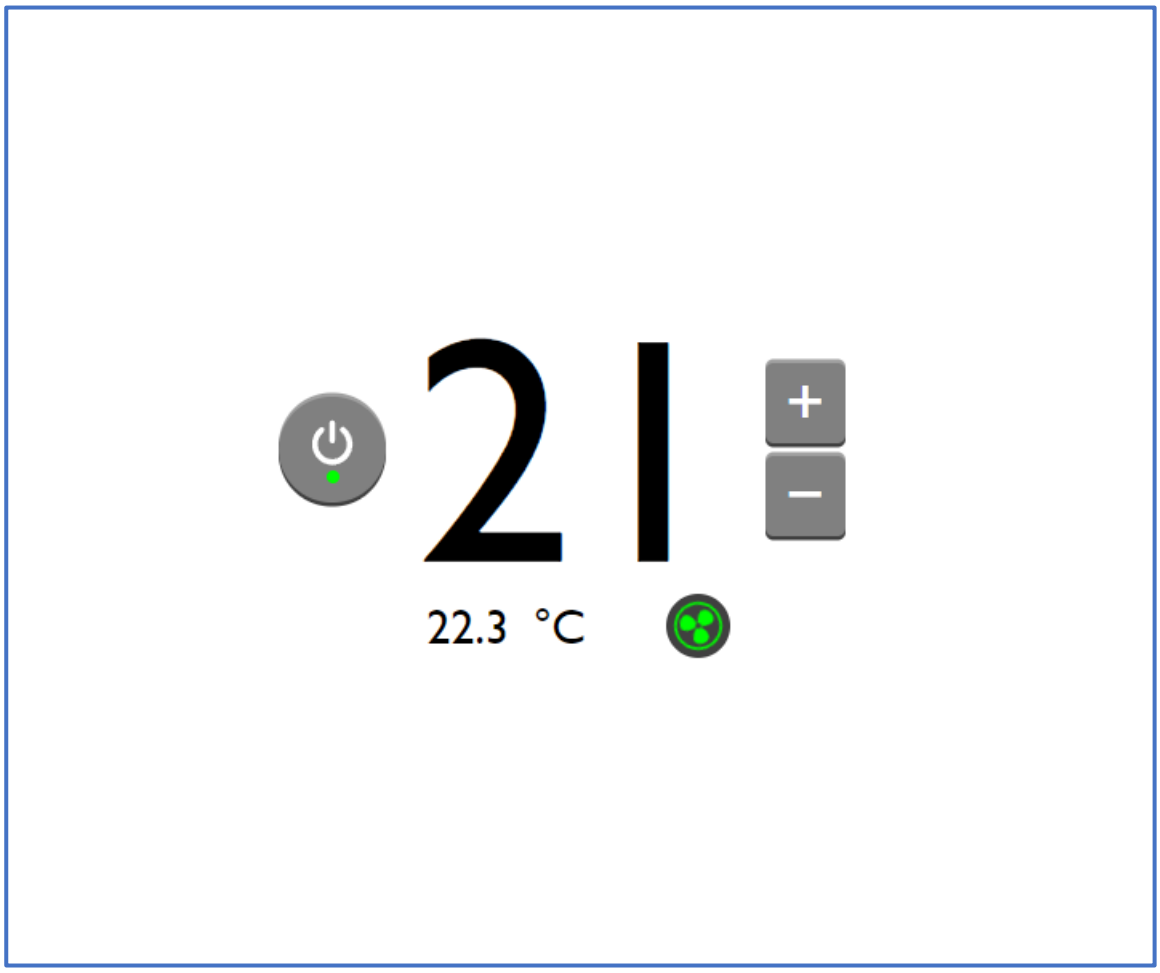
- željena temperatura ter
- ali je sistem omogočen.

Zamislili smo si grafično nezahteven vmesnik. Temperature izrisujemo tekstovno, stanje ventilatorja pa prikazujemo s preprostim indikatorjem z dvema stanjema. Indikatorju za omogočenost sistema smo dodali še tretjo stanje, ki nakazuje, da se nastavitev na krmilniku še ni spremenila, saj se uporabniku v nasprotnem primeru gumb za vklop oziroma izklop sistema zaradi omrežne latence ter intervala na mikrokrmilniku zdi neodziven. Nastavljanje željene temperature poteka z gumboma za inkrement in dekrement temperature. Da bi dosegli minimalistični izgled, smo namesto s tekstovnimi opisi pomen vrednosti in gumbov poskusili predstaviti s simboli, velikostjo in pozicijo posameznih elementov. Postavitev elementov so navdihnile kontrolne površine električnih grelcev. Elemente, ki so gumbi, smo senčili, da je zaradi njihovega izstopanja njihova funkcija bolj jasna.

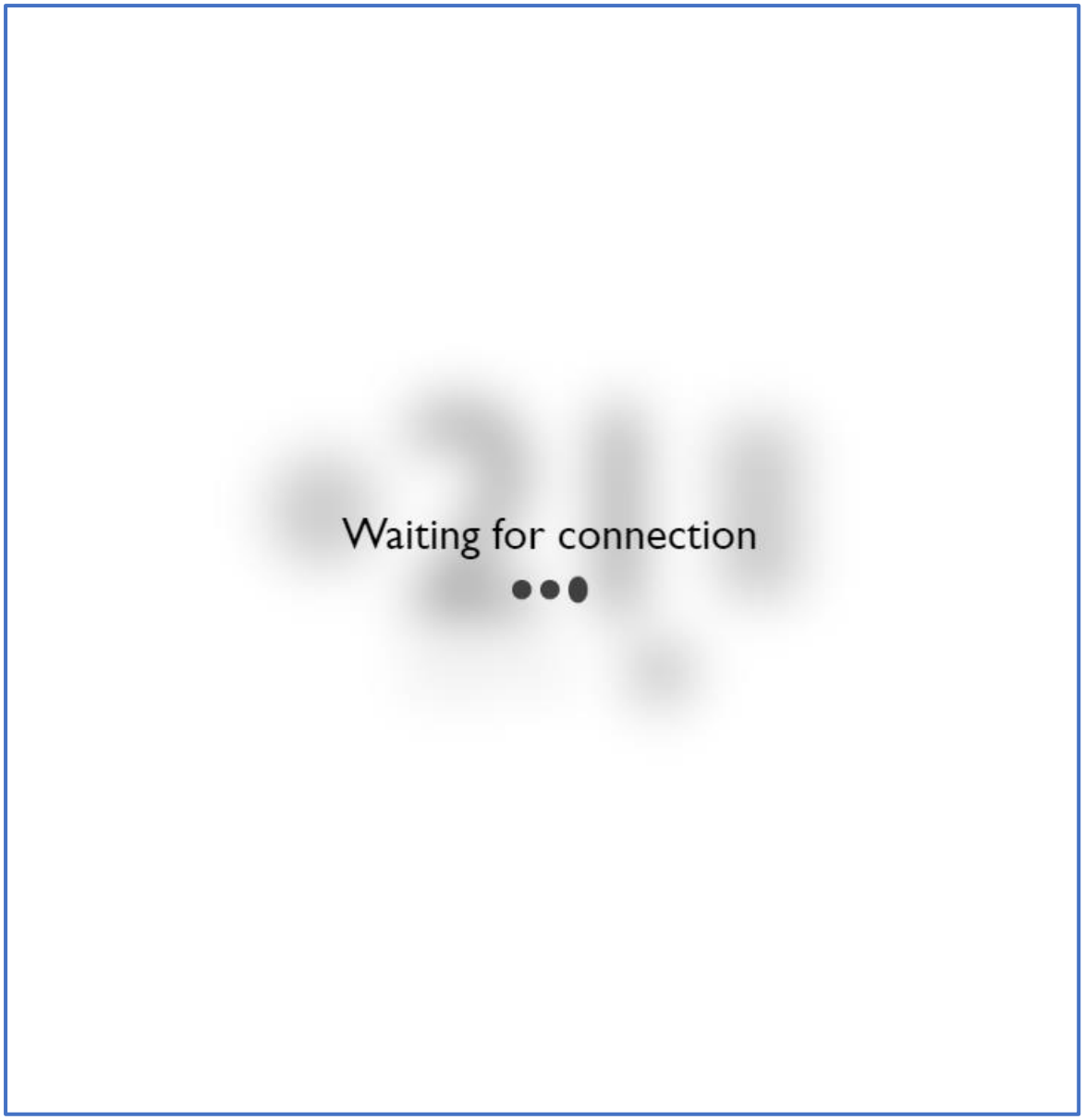
Ali ima uporabniška naprava trenutno povezljivost na strežnik določamo na podlagi sekund od zadnje povezave, ki nam jih sporoči strežnik, ko preberemo stanje naprave. Z izračunom sekund na strežniku se izognemo potrebi po sinhronizaciji ure odjemalca in strežnika. Če se naprava določen čas ni povezala na strežnik, uporabniški vmesnik prekrijemo s prekrivnim slojem, ki uporabniku sporoči, da trenutno naprava ni povezana, ter onemogoči spreminjanje nastavitev.

Izgled vmesnika je napisan v HTML in CSS, funkcionalnost pa v JS. Uporabniški vmesnik je napisan kot ločen element, tako da se lahko vgradi v obstoječo spletno stran v poljubnem številu. Objekt vmesnika hrani vse potrebne podatke za povezavo do končne točke REST API, ter samodejno periodično vzpostavlja komunikacijo s strežnikom, posodablja grafične elemente, ter obdeluje klike oziroma pritiske gumbov. Ker se trenutne nastavitve ves čas berejo iz strežnika, je brez težav možen nadzor naprave iz več naprav brez potrebe po osvežitvi strani. Da preprečimo izpis starih vrednosti nastavitev po tem, ko jih uporabnik spremeni, vrednosti nastavitev po modifikaciji zamrznemo za kratek čas, da informaciji damo čas, da prispe do strežnika.

Spletna aplikacija na začetku prebere avtorizacijski ključ podan v zahtevi, nato pa ustvari novo instanco vmesnika, ter jo vstavi v DOM. Za enostavno testiranje se ključ prebere iz GET parametra.



Slika 5: Izgled uporabniškega vmesnika



Slika 6: Prekrivni sloj, ki se prikaže, ko strežnik neha prejemati sporočila od naprave

5. Evalvacija

5.1. Uporabniška izkušnja

Uporabniška izkušnja je dobra. Spletno aplikacijo je testiralo več ljudi, in nihče ni potreboval razlage kontrol. Z dovolj kratkimi intervali med sinhronizacijami nastavitev s strežnikom na mikrokrmilniku je odzivnost zadovoljiva, se pa s krajšanjem intervalov veča količina proizvedenega internetnega prometa, ter najverjetneje poraba energije.

5.2. Razvoj sistema

Povezovanje posameznih delov sistema (uporabniške naprave, spletne aplikacije in back-end vmesnika) je bilo enostavno in brez težav, kar verjetno pomeni, da je bil vmesnik dobro zasnovan.

Največ časa pa je bilo porabljenega na načrtovanju in programiranju uporabniškega vmesnika za spletno aplikacijo, kar pomeni, da se bi nam za nadaljnje delo na področju razvoja IoT storitev najbrž najbolj splačalo prvo izboljšati seznanjenost s tehnologijami spletnih uporabniških vmesnikov.

5.3. Varnost

Trenutno koda na mikrokrmilniku uporablja HTTP odjemalec, kar pomeni, da so tako stanje naprave, kot nastavitve uporabnika, vidne vsem mrežnim napravam med strežnikom in napravo. To pri današnjih varnostnih standardih za končni produkt ne bi bilo sprejemljivo, zato bi HTTP odjemalec bilo potrebno zamenjati s HTTPS odjemalcem.

Varnost bi lahko izboljšali tudi v uporabniškem vmesniku. Namesto, da avtentikacijski ključ podajamo kot GET parameter, ki ga brskalnik shrani v predpomnilnik, bi na uporabnikovi strani v piškotek morali shraniti ID seje po tem, ko se bi uporabnik overil z ustreznimi poverilnicami, na primer na drugi strani naše spletne aplikacije. To bi zahtevalo tudi dodatno delo na strani strežnika, ki bi moral upravljati z ID-ji sej, ter varno hraniti poverilnice.

5.4. Razširljivost

Trenutni sistem podpira zgolj eno napravo, podatke za katere hrani zgolj v glavnem pomnilniku. Za podporo več naprav, in persistenco, bi bilo podatke verjetno najbolj optimalno hraniti v podatkovni bazi. Ker pa bi si z več napravami najbrž želeli tudi več uporabnikov z dostopom do zgolj določenih naprav, bi bilo potrebno v bazi hraniti tudi uporabnike, ter njihove poverilnice.