Pametni sustav za održavanje biljaka temeljen na Raspberry pi-u

Peruško, Daniel

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:262227

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Sveučilišni Preddiplomski Studij Informatika

Daniel Peruško

Pametni sustav za održavanje biljaka temeljen na Raspberry pi-u

Završni rad

Mentor: doc. dr. sc. Miran Pobar

Rijeka, 26.09.2022.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, 30.5.2022.

Zadatak za završni rad

Pristupnik: Daniel Peruško

Naziv završnog rada: Sustav za njegu biljaka temeljen na Raspberry pi-u

Naziv završnog rada na eng. jeziku: Raspberry pi-based plant care system

Sadržaj zadatka: Potrebno je razviti sustav koji će omogućiti udaljeno praćenje stanja biljaka, po potrebi nadolijevati vodu u posudu ili upaliti svjetiljku, uz mogućnost unaprijed definiranog automatskog paljenja i gašenja sustava za navodnjavanje ili rasvjete. Hardverska osnova sustava može biti računalo Raspberry Pi i odgovarajući senzori i aktuatori, dok softverski dio predstavlja web aplikacija koja će pružati pregled stanja biljka i senzorskih podataka te upravljanje aktuatorima. Opisati hardversko i softversko rješenje.

Mentor

Doc. dr. sc. Miran Pobar

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 30.5.2022.

Daniel P.

(potpis pristupnika)

Sažetak

Ideja ovog završnog rada je napraviti web aplikaciju kroz koju ćemo promatrati stanje jedne ili više biljka te po potrebi upravljati sustavom za održavanje na daljinu. Projekt možemo realizirati pomoću Raspberry Pi platforme na koju su spojeni razni senzori i aktuatori. Povezanost web aplikacije i Raspberry Pi-a će omogućiti baza podataka. Raspberry Pi periodički pohranjuje informacije dobivene očitavanjem senzora, pa ih zatim web aplikacija dohvaća i informira korisnika o stanju pojedine biljke. Korisnik nadalje može upravljati aktuatorima putem web aplikacije kako bi utjecao na biljku.

Ključne riječi: Raspberry Pi, web aplikacija, senzori, pametni sustav, održavanje biljka.

Sadržaj

1. Uvod

Ovim projektom se želi riješiti problem održavanja biljka u vremenu kada korisnik nije u blizini biljke. Biljke je potrebno njegovati skoro svaki dan. Treba ih zalijevati, pobrinuti se da imaju dovoljnu količinu svijetla, kontrolirati kvalitetu tla i pratiti fizički izgled biljke, ukoliko želimo da biljka ostane zdrava. Potrebno je osmisliti sustav koji će nam omogućiti praćenje stanja biljka, po potrebi nadolijevati vodu u posudu ili upaliti svjetiljku, mogućnost unaprijed definirati automatsko paljenje i gašenje sustava za navodnjavanje ili svjetiljku i pregled prošlih očitavanja senzora. Ideja se može realizirati pomoću mikroračunala i web aplikacije. Jedna od poznatijih mikroračunala je platforma Raspberry Pi. Putem računala ili pametnog telefona želimo vidjeti te podatke u lijepo uređenom obliku. Potreban nam je jednostavna web aplikacija koja će nam pružati pregled stanja biljka i po potrebi reagirati.

Sustav se sastoji od Raspberry Pi-a koji će pokretati web server i prikupljati informacije biljke pomoću senzora. Našem sustavu ćemo dodati kameru koja se temelji na Arduino platformi. Glavni programski jezik u ovo projektu je Python, te se susrećemo s JavaScripta i C++. Korisnik se spaja na server pomoću web aplikacije koja prikuplja informacije iz baze podataka. Senzorske informacije će se očitavati svakih 60 sekundi i pohraniti u bazu podataka. Klijent će se na server spojiti pomoću web aplikacije.

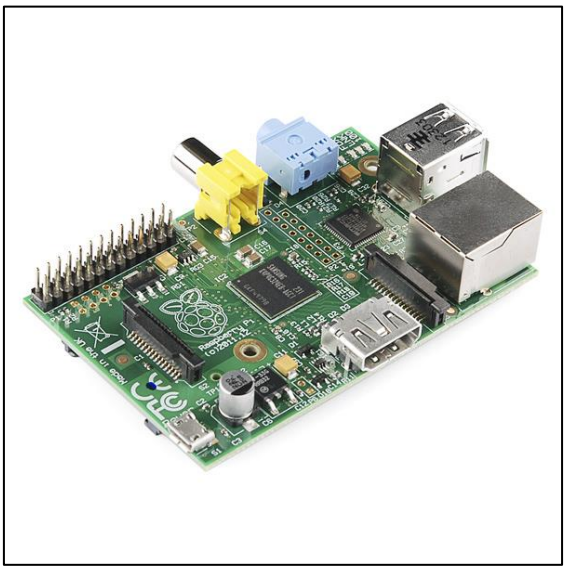
2. Korištene tehnologije

2.1. Osnove Raspberry Pi platforme

Raspberry Pi je grupa malih kompaktnih računala dizajnirana u Velikoj Britaniji. Prvi model *„Raspberry Pi Model B“ (Slika 1.)* se pojavio na tržištu u veljači 2012. godine. Operacijski sustav se zove *„Raspberry Pi OS*“ (32 ili 64-bitni) koji se temelji na Debian operacijskom sustavu iz Linux obitelji.

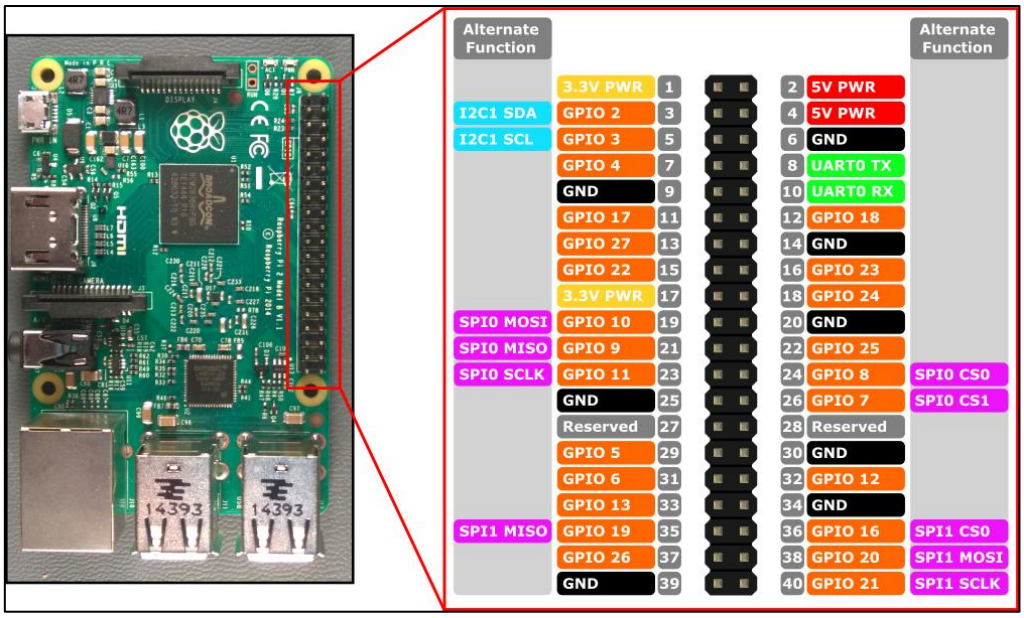
Sadrži sve hardverske komponente kao i tradicionalno računalo kao što su procesor, radna memorija, pohrana podataka.

Velika prednost ovih mikroračunala je njihova kompaktnost i cijena. Najčešća područja primjene su im robotika, Internet stvari (eng. „*Internet of Things*“, skraćeno IoT) i školski projekti.



Slika 1. Raspberry Pi Model B. [14]

Također, ono što ih još čini prikladnim u pojedinim situacijama su tzv. general purpose input-output (*„GPIO“) konektori (Slika 2).*



Slika 2. GPIO oznake Raspberry Pi-a 2. [14]

GPIO su spojevi pomoću kojih Raspberry Pi komunicira s ulaznim i izlaznim uređajima. Jednostavno se implementiraju u programski kod i putem njih komuniciramo s ostalim elektroničnim elementima spojenih na GPIO. Ulazne komponente su najčešće senzori (npr. senzori temperature, vlažnosti, fotootpornika i slično). Izlazne komponente zahtijevaju napon od 3.3V ili 5V, a može se raditi o uređajima kao što su jednostavne LED diode ili kompleksniji sustavi kao releji koji upravljaju raznim naponima.

Tijekom godina tehnologija je napredovala i s time su novi modeli Raspberry Pi-a sposobniji obraditi veću količinu podataka nego prijašnji modeli i imaju više funkcionalnosti koje su utjecale na rast njihove popularnosti.

Trenutno najznačajniji model je *„Raspberry Pi 4 Model B“* koji nudi procesor *„ARM Cortex-A72“* (1.5 GHz) s četiri jezgre i do 8 GB radne memorije. Postoje ostali modeli kao što su:

- Raspberry Pi Pico

Jedan od najmanjih mikrokontrolera, dvojezgreni Arm Cortex-M0+ procesor brzine do 133MHz s 264kB radne memorije i 2MB QSPI pohrane. Idealan je za kompaktne i jednostavne projekte.

- Raspberry Pi Zero

  Nudi procesor s jednom jezgrom do 1GHz, 512 MB radne memorije, Mini-HDMI, Micro-USB priključak i 40-pinsko zaglavlje. Idealan je za projekte kod kojih je potrebna veća brzina obrade podataka, a dodatna prednost mu je mini-HDMI priključak za video signal.

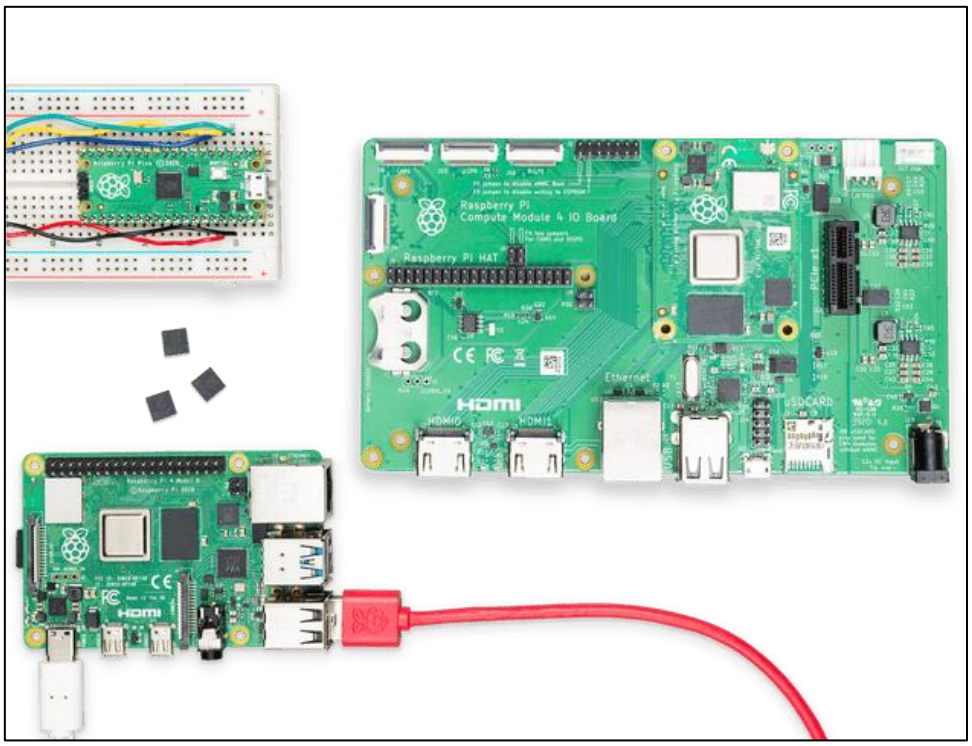
- Raspberry Pi Compute Module

  Model posebno specijaliziran za polje industrije i ugradbenih sustava, degradirani model Raspberry Pi-a 4 s manje priključaka.

- Raspberry Pi Model A

  Jeftiniji od Modela B s limitiranim funkcionalnostima. Sadrži samo jedan USB priključak i nema ugrađen RJ45 priključak u odnosu na Model B koji ima 2 USB i RJ45 priključka.

Modeli se razlikuju po veličini, specifikacijama i ponuđenim funkcionalnostima, te su pojedini prikladniji za specifične projekte od ostalih modela (Slika 3).



Slika 3. Različiti modeli Raspberry Pi-a. [14]

Za ovaj projekt odabran je „Raspberry Pi 3 Model B V1.2“. Posjeduje 40-pinski GPIO konektor koji nam je potreban za povezivanje senzora i aktuatora, četverojezgreni procesor brzine 1.2GHz sa 1GB radne memorije i Wi-Fi protokolom. Idealne specifikacije za obradu podataka i slanje u bazu.

2.2. Osnove Arduino platforme i razlika u odnosu na Raspberry Pi

Arduino je programabilni mikrokontroler za upravljanje digitalnih uređaja. Sastoji se od procesora, radne memorije i pohrane. Neki modeli također imaju ugrađene Wi-Fi, Bluetooth i ostale tehnologije za komunikaciju. Kao Raspberry Pi, Arduino također posjeduje pinsko zaglavlje kojim mu je omogućeno komunicirati s ostalim komponentama. Arduino se programira pomoću njegovog integriranog razvojnog okruženja ili IDE (eng. "*Integrated Development Environment*") u programsko jeziku C/C++. Platforme Arduina i Raspberry Pi su vrlo slični u nekim područjima, ali bitna razlika između njih je da je Arduino mikrokontroler, dok Raspberry Pi spada u mikroprocesore.

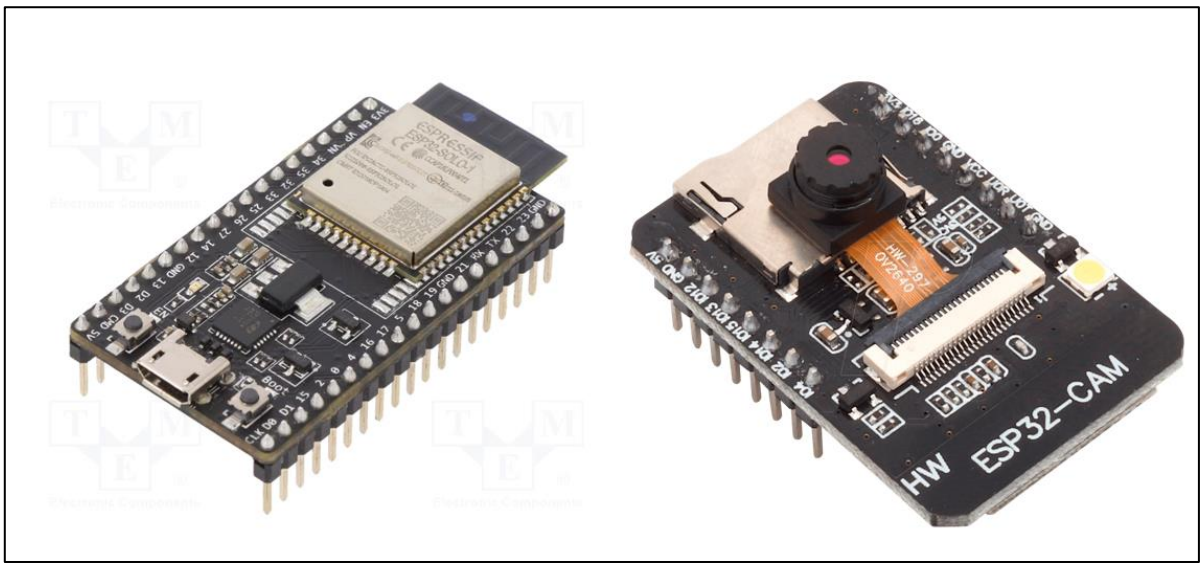
Platforma Arduino:

- Lakše upravljanje senzorima i aktuatorima
- Limitiran na C/C++ programski jezik

Platforma Raspberry pi:

- Potpuno računalo.
- Fleksibilnost u programiranju

Jedan od popularnih obitelji modela Arduina je ESP32 (Slika 4.), nudi jednojezgreni ili dvojezgreni procesor do 240MHz brzine, radna memorija do 16MB i pohranu podataka. Za potrebe projekta koristit ćemo model „ESP32 - CAM". Degradirani model *„ESP32"* obitelji koji je specijaliziran za fotografiranje i snimanje okoline, idealno za ovaj projekt.



Slika 4. Arduino model ESP32 i ESP32 - CAM. [13]

2.3. Quasar / Vue.js

Web aplikacija je izrađena pomoću popularnog Vue.js-a (Quasar) (Slika 5.), *„open-source“* front-end JavaScript framework-a. Radi se o proširenom okruženju od HTML i JS s obogaćenim funkcijama. Idealna je za izradu korisničkog sučelja i aplikacije na jednoj stranici.

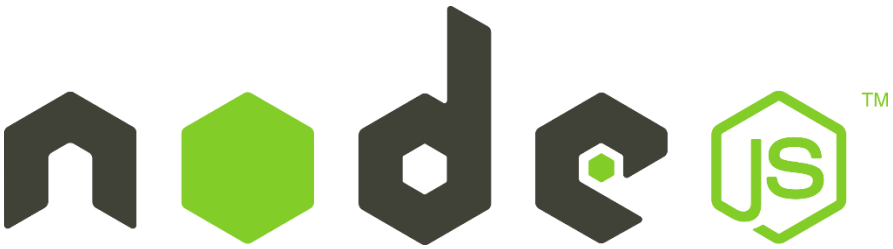
Quasar Framework

Slika 5.Quasar logo. [2]

Glavna struktura jedne vue stranice se dijeli na <template> u kojemu pišemo HTML kod, <script> dio u kojemu se nalaze podaci i funkcije koda, i privremeno <style> koja je zadužen za izgled HTML koda, ali on se najčešće nalazi u zasebnoj datoteci.

2.4. Node.js

Node.js (Slika 6.) je *„open-source“* back-end JavaScript radno okruženje koje omogućuje izvođenje JavaScript koda izvan web preglednika. Omogućuje da se svi zahtjevi provode pomoću jednog procesa, tj ne stvara novi proces za svaki zahtjev. Node.js je izrazito lagan pa se može koristiti na mikroračunalima.



Slika 6. Logo Node.js. [12]

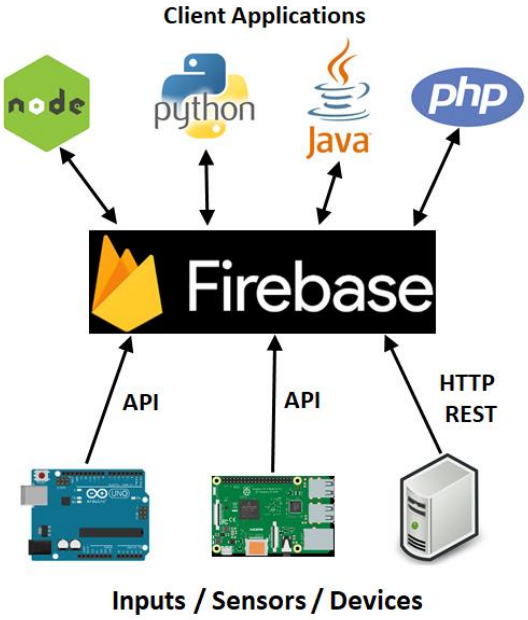
Jedne od popularnih aplikacija koje koriste Node.js su: LinkedIn, Uber, Netflix, Payplay, Mozilla.

2.5. Firebase

Google nudi dvije vrste baze podataka:

- Cloud Firestore
- Realtime Database

Realtime Database je originalna baza podataka, efikasna i brza. Cloud Firestore je nadogradnja Realtime Databasu, suvremenija je, brža, fleksibilna, bolje se skalira i nudi veći izbor modela podataka. Radi se o NoSQL tipu baze podataka koji je idealan za potrebe ovog projekta (spremanje teksta, brojeva i fotografija). U ovom projektu koristit ćemo bazu podataka Cloud Firestore (Slika 7). Sigurna sinkronizira podatke na svim spojenim uređajima ju čini pouzdanom. Baza se dijeli na kolekcije, koje su podijeljene na dokumente. Kolekcija može imati i podkolekcije, šta su kolekcije unutar kolekcije (ugniježđene kolekcije).



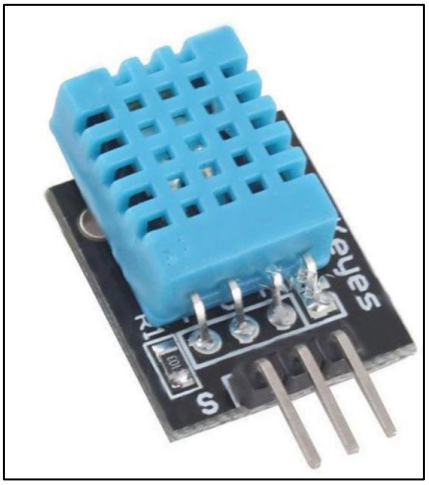
Slika 7. Firebase baza podataka. [1]

3. Korišteni senzori i aktuatori u projektu

U ovom projektu koristit ćemo sljedeće senzore i aktuatore:

3.1. DHT11

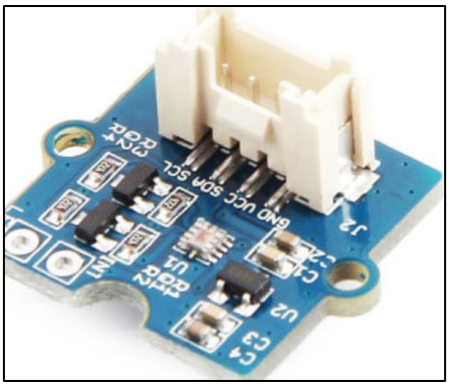
DHT11 je senzor koji ispisuje podatke o trenutnoj temperaturi i vlažnosti zraka u okolini biljke (Slika 2). Granice mjerenja temperature su od 0°C do 50°C, a vlažnosti zraka do 20% do 90%. Tolerancija iznosi ±1°C i ±1%. Tijekom mjerenja potrebno mu je 0.3mA jakosti električne struje, a potrebna voltaža mu je između 3 do 5.5 V. Posjeduje tri izlazna konektora (Slika 8). Dva su namijenjena za napon, a jedan za prijenos podataka.



Slika 8. DHT11 Senzor temperature i vlažnosti zraka. [7]

3.2. Grove Sunlight Sensor

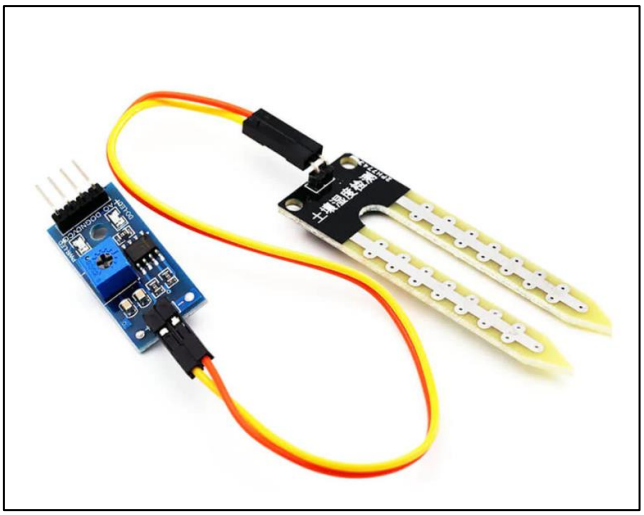
Grove - Sunlight Sensor je senzor koji očitava jačinu svjetlosti u okolini biljke (Slika 9). Ima mogućnost očitavanja ultraljubičaste, vidljive i infracrvene spektra svjetlosti ili valne duljine između 280 – 950 nm. Radna temperatura mu je između -45°C do 85°C. Tijekom mjerenja potrebno mu je 3.5mA jakosti električne struje, a potrebna voltaža mu je između 3 do 5.5 V. Za prijenos podataka koristi I2C protokol. I2C je protokol koji zahtijeva samo dvije signalne linije. Protokol omogućuje komunikaciju između više uređaja pomoću istih signalnih linija.



Slika 9. Grove Sunlight Sensor [5]

3.3. Soil Moisture Sensor

Soil Moisture Sensor je senzor pomoću kojeg očitavamo količinu vlažnosti tla biljke (Slika 10). Sam senzor je spojen na svoj modul koji služi za regulirati granice i pretvorbu signala. Osim dva konektora za dovod napona, postoje još dva konektora, jedan za prijenos informacija u analognom obliku, a drugi u digitalnom. Potrebna voltaža za rad je između 3.3 do 5 V.



Slika 10. Soil Moisture Sensor. [8]

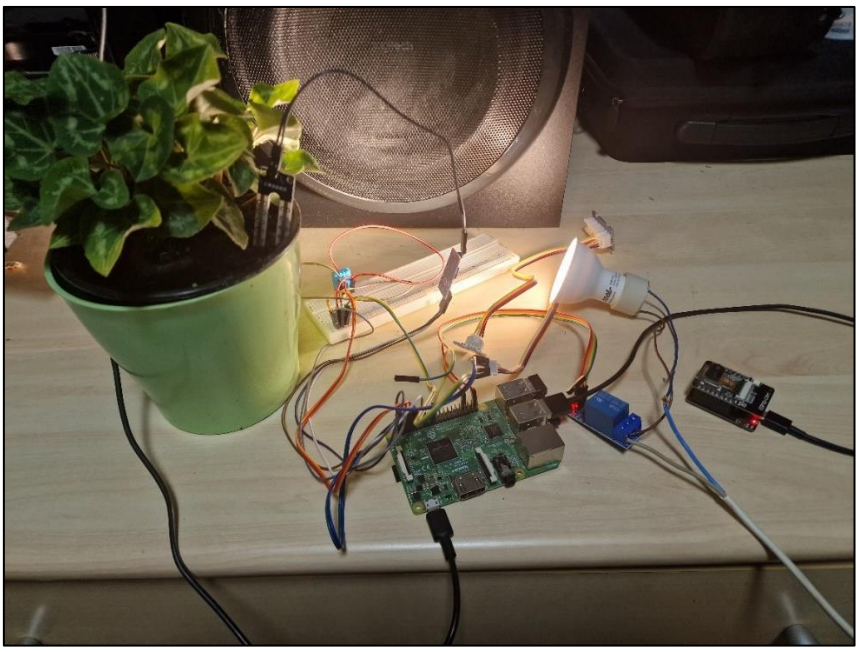
3.4. Relej 5V/230V

Relej je vrsta prekidača koji kontroliramo električnim nabojem (Slika 11). Dovodom električnog napona oko elektromagneta se stvara magnetsko polje koje mjenja stanje sklopke. Upravlja naponom do 250V AC ili 30V DC (10A), što je standardni raspon kojeg zahtijeva većina električnih uređaja. Potrošnja mu je između 50 do 60 mA. Osim konektora za dovod napona, postoji treći konektor putem kojeg se šalje signal za aktivaciju releja. Na izlazu se nalaze 3 spojnice koje

omogućuju dva strujna kruga. Jedan strujni krug je uvijek zatvoren dok drugi nije, djelovanjem releja stanja strujnih krugova se mijenja. Njime ćemo upravljati žarulju za našu biljku.

Slika 11. Relej 5V/230V. [3]

Prednost gore navedenih senzora je njihova jednostavnost i niska cijena na tržištu, zbog kojih su odabrani za ovaj projekt. Određeni senzori kao povratnu informaciju vraćaju u obliku analognog signala. Platforma Raspberry Pi nema načina da očitava analogni signal. Postoji rješenje kojim prevodimo analogni signal u digitalni, npr. pomoću mikročipa „*MCP3008*“. Sami programski kod je napisan tako da bude fleksibilan za korištenje drugih senzora uz minimalne izmjene. Na izlazu se nalazi relej u funkciji aktuatora, za paljenje i gašenje žarulje. Način na koji su spojeni svi senzori i aktuator na Raspberry Pi, s Arduino dodatkom je prikazan na slici 12.



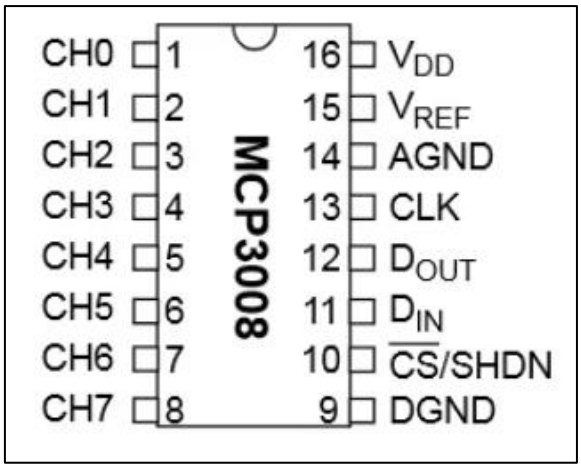
Slika 12. Izgled projekta.

3.5. Mikročip MCP3008

Kao što smo prije spomenuli, Raspberry Pi nema načina da očitava analogni signal. Neki senzori vraćaju povratnu informaciju isključivo u obliku analognog signala, te komunikacija između tih senzora i Raspberry Pi-a je omogućena pomoću analogno – digitalnih pretvarača (eng. *„analog-digital converter*“, skraćeno ADC) (Slika 13).

Osnovne karakteristike pretvarača:

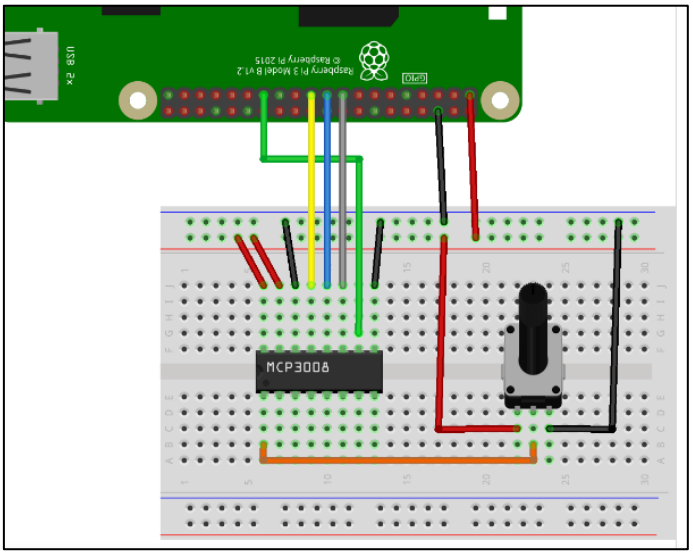
- Ulazni raspon napona: 2.7 – 5.5V
- Broj bitova iznosi: 10
- Brzina uzrokovanja po sekundi: 200 tisuća
- Broj ulaza: 8
- Komunikacijski protokol: SPI (eng. *„Serial Peripheral Interface*“)



Slika 13. Shema MCP3008 pretvarača. [11]

3.6. Primjer pretvaranja analognog signala u digitalni za potenciometar

Na slici 14. vidimo način spajanja MCP3008 pretvarača s pinskim zaglavljem i potenciometrom. Serijsko periferno sučelje ili SPI od 9. do 16. kontakta spojeno je na pinsko zaglavlje Raspberry Pi-a, a podatkovni izlaz potenciometra je spojen na kanal 0 ili na kontakt 1 pretvarača.



Slika 14. Shema pretvaranja analognog signala potenciometra u digitalni. [9]

Nadalje u programskom kodu (Slika 15.) pozivamo odgovarajuće biblioteke.

Biblioteka „busio“ omogućuje SPI komunikaciju. „digitalio“ omogućuje pristup glavnim digitalnim ulazno izlaznim funkcijama. „board“ omogućuje jednostavno korištenje GPIO. „MCP“ i „AnalogIn“ omogućuje radnje sa MCP3008 pretvaračem i „time“ mogućnost mjerenja vremena u kodu.

Program ispisuje stvarnu pretvorenu vrijednost i voltažu potenciometra. Pišemo sljedeći kod:

```
import busio
import digitalio
import board
import adafruit_mcp3xxx.mcp3008 as MCP
from adafruit_mcp3xxx.analog_in import AnalogIn
import time

# create the spi bus
spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)

# create the cs (chip select)
cs = digitalio.DigitalInOut(board.D5)

# create the mcp object
mcp = MCP.MCP3008(spi, cs)

# create an analog input channel on pin 0
chan = AnalogIn(mcp, MCP.P0)

while True:
  print('Raw ADC Value: ', chan.value)
  print('ADC Voltage: ' + str(chan.voltage) + 'V')
  time.sleep(1)
```

Slika 15. Primjer programskog koda analognog čitanja potenciometra.

4. Programski kod na Raspberry Pi-u

Na početku programskog koda pozivamo sve biblioteke koje će se kasnije koristiti. Definiramo jedinstvene postavke Raspberry Pi-a kojima se upućujemo jednoj biljci (Slika 16).

```
#Jedinstvene postavke Raspberry Pi-a
constPlantID = '0006'
constUIDPlant = '77C9Sg60ixljradBT0OQ'
```

Slika 16. Jedinstvene postavke biljke.

Povezujemo se s Google-ovom bazom podataka Firebase pomoću privatnog ključa koji se nalazi u datoteci *„key.json“* (Slika 17).

```
cred = credentials.Certificate("key.json")
firebase_admin.initialize_app(cred)
db=firestore.client()
```

Slika 17. Povezivanje s bazom podataka.

Nakon toga inicijaliziramo senzore i povežemo ih s GPIO pinovima. Sljedeća ponavljajuća petlja je glavni dio programa koja se izvodi svakih 60 sekundi. Na početku očitavamo podatke s odgovarajućih senzora, te ih pohranjujemo u zasebne varijable (Slika 18).

```
light = SI1145.ReadVisible//10
humidity, temperature = Adafruit_DHT.read_retry(11, 4)
if(moisturevalue.value):
    moisture = 0
else:
    moisture = 100
```

Slika 18. Očitavanje senzora.

Zatim dobivene podatke šaljemo bazi podataka dva puta, jednom ažuriramo podatke biljke u kolekciji *„plantlist“* , a drugi put dodajemo novo čitanje u kolekciju *„readings“* (Slika 19).

```
db.collection(u'plantlist').document(constUIDPlant).update(data)
db.collection('readings').add(readingsData)
```

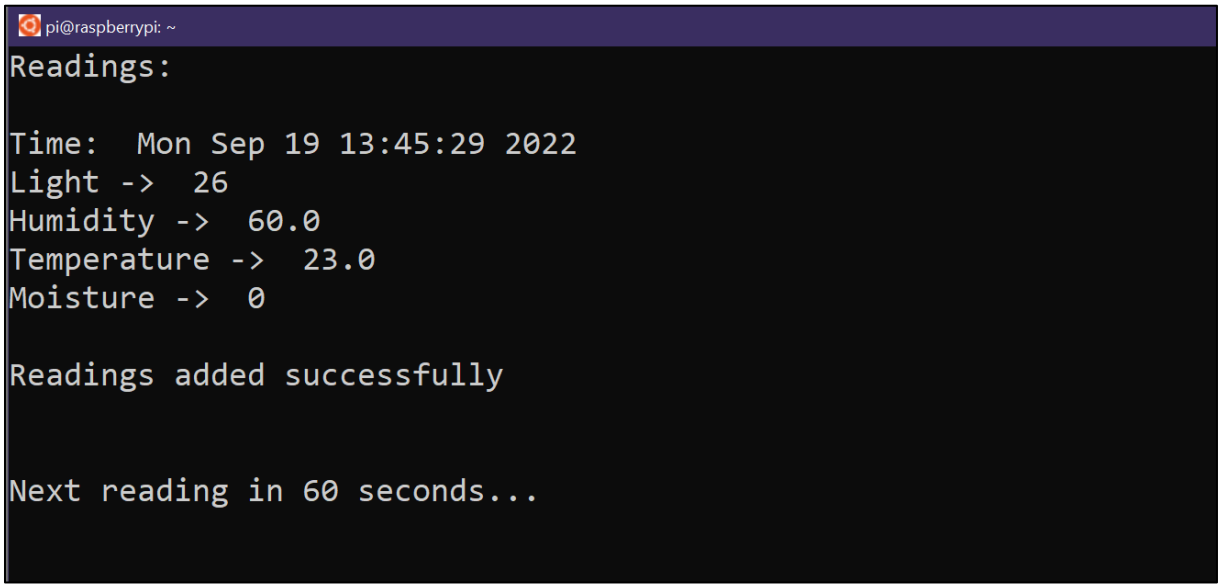
Slika 19. Slanje podataka u bazu.

Zadnji dio programskog koda je zadužen za uključivanje i isključivanje releja ovisno o vremenu, tj. „*tajmer*“. Povlačimo dokumente iz kolekcije „*timers*“ i zatim ih filtriramo. Ukoliko je vremenski interval zadovoljen, odabrani relej će biti uključen (Slika 20).

```
if (y["startTimeUNIX"] and y["stopTimeUNIX"]):
    timeCompareStart = y["startTimeUNIX"]
    timeCompareStop = y["stopTimeUNIX"]
    if (time.time() >= timeCompareStart) and (time.time() <= timeCompareStop):
        if(y["lightBulb"] and flagLight==False):
            print('Light is on!\n')
            lightIsOn = True
            flagLight = True
            GPIO.output(relej, GPIO.LOW)
```

Slika 20. Isječak koda zadužen za paljenje žarulje.

Pokretanjem programa, ispisuje nam se očitavanja vrijednost senzora (Slika 21).

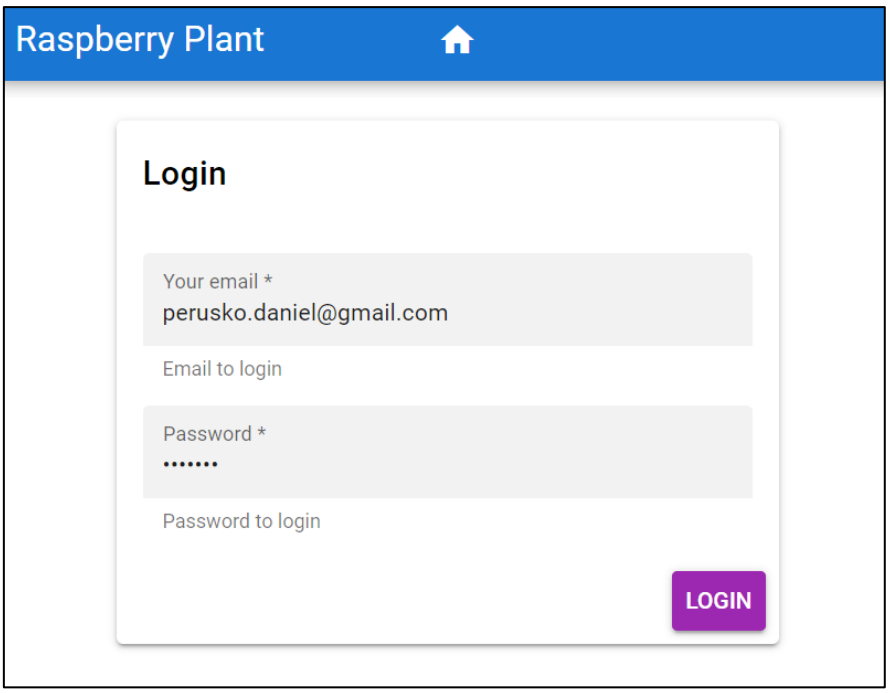


Slika 21. Očitavanja vrijednost senzora.

5. Web aplikacija

5.1. Prijava u sustav

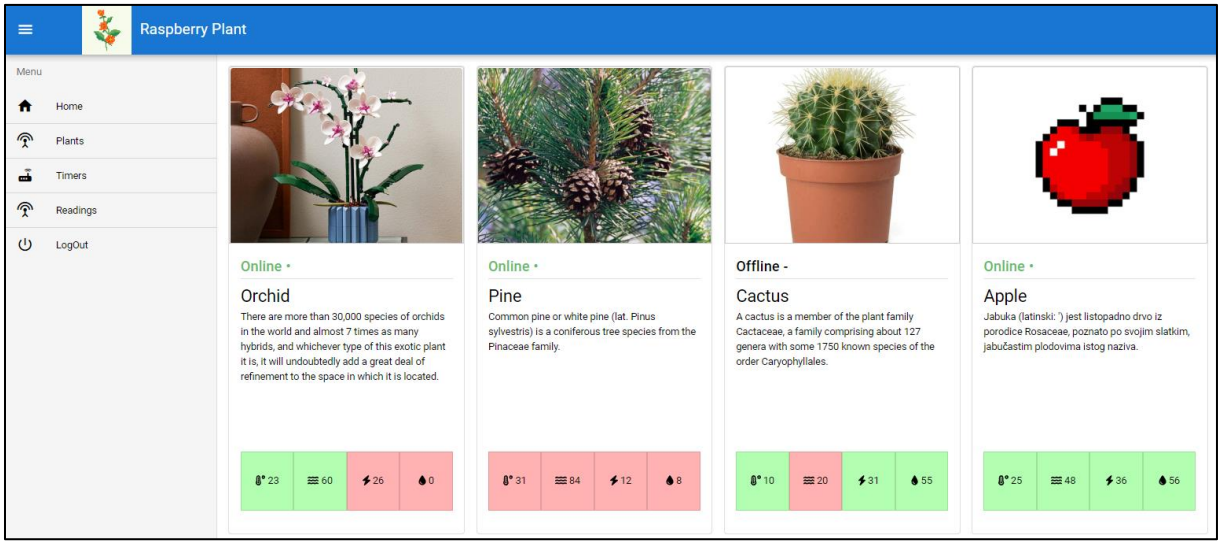
Za ulazak u web aplikaciju potrebno se autentificirati pomoću email adrese i lozinke (Slika 22).



Slika 22. Prijava u aplikaciju.

5.2. Početna stranica (Home)

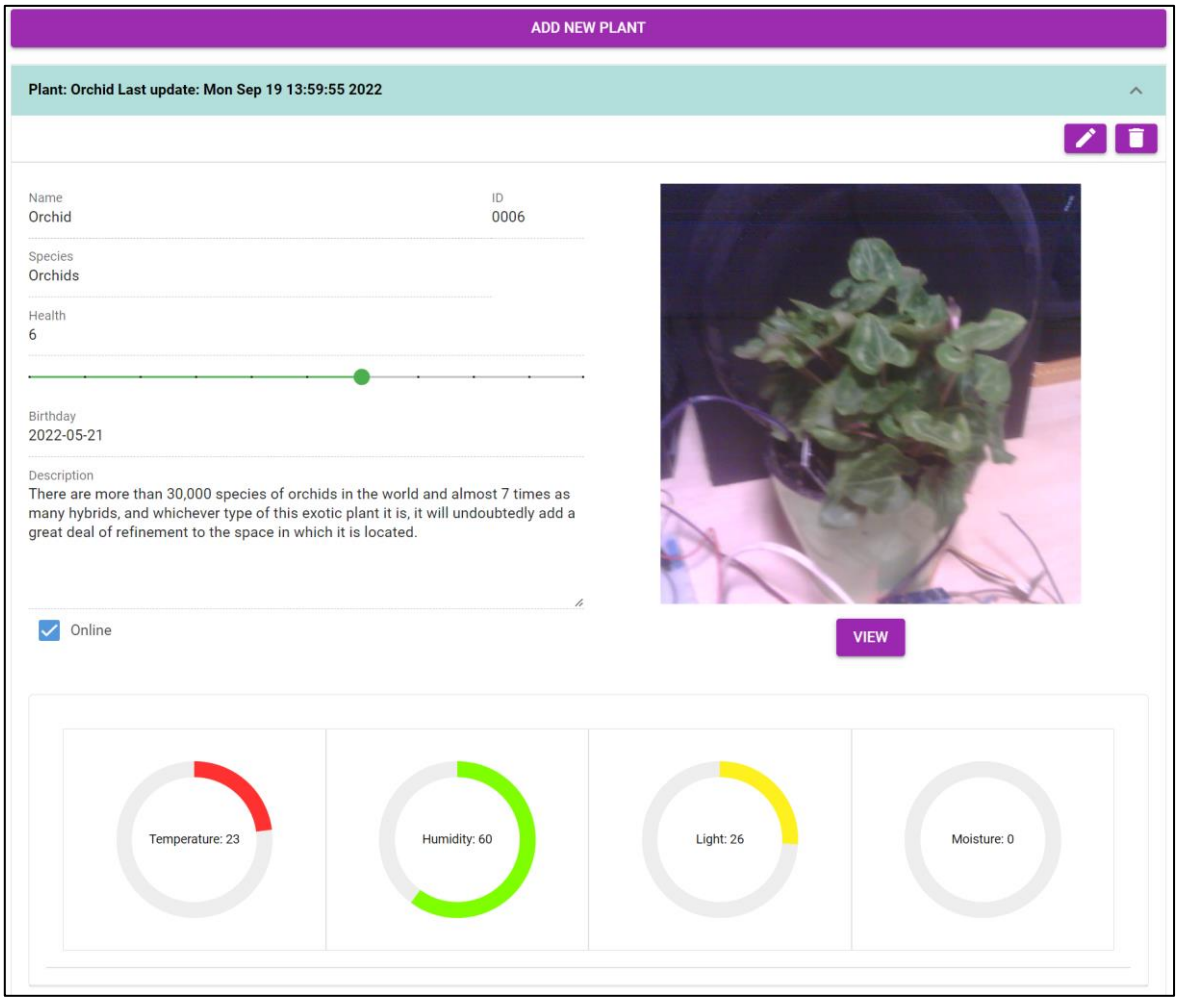
Na početnoj stranici (Slika 23.) se nalazi osnovni pregled informacija svih biljka u obliku grid-a (mreže). Svaka biljka ima svoju fotografiju, osnovne informacije kao ime, opis i vrijednost senzora. Ukoliko je vrijednost senzora niska za potrebe biljke, crvenom bojom je istaknuta informacija, u suprotnom informacija je istaknuta zelenom bojom.



Slika 23. Početna stranica.

5.3. Popis biljka (Plants)

Na ovoj ugniježđenoj stranici se nalazi lista biljka, biljke su podijeljene u obliku kartica. U naslovu kartice se nalazi ime biljke i vrijeme zadnje promijene. Klikom na zaglavlje kartice ona se proširuje ili smanjuje (Slika 24).



Slika 24. Detaljni prikaz biljke.

Proširenjem kartice dobivamo detaljni pogled informacija od pojedine biljke:

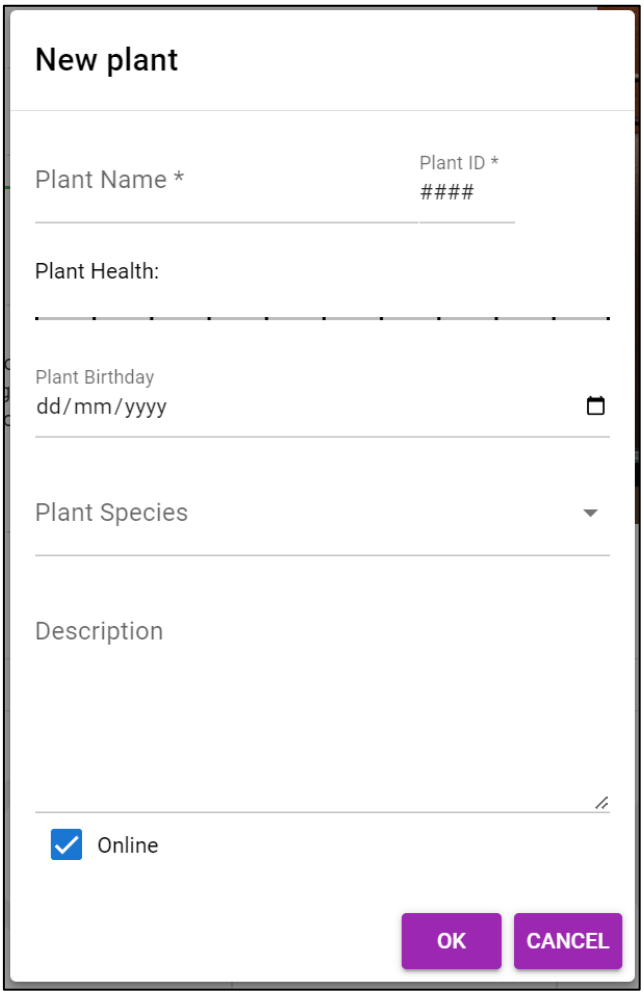
- Ime
- Jedinstvenu oznaku
- Vrstu biljke
- Stanje biljke
- Datum postojanja
- Opis
- Sliku
- Očitavanja
- Stanje

Klikom na gumb „*View*“ otvara nam se novi prozor s uvećanom slikom.

Očitavanja senzora temperature, vlage zraka, jačina svijetla i vlaga tla nam se prikazuje u obliku kružnog grafa (eng. „*Pie chart*“).

Pritiskom na gumb „*add new plant*“, korisniku se otvara novi prozor za upisivanje podatka (Slika 25). Korisnik unosi ime biljke, jedinstvenu oznaku, stanje biljke, datum postojanja, vrstu, opis i stanje.

Također korisniku je omogućeno promjena informacija određene biljke kao i brisanje biljke.



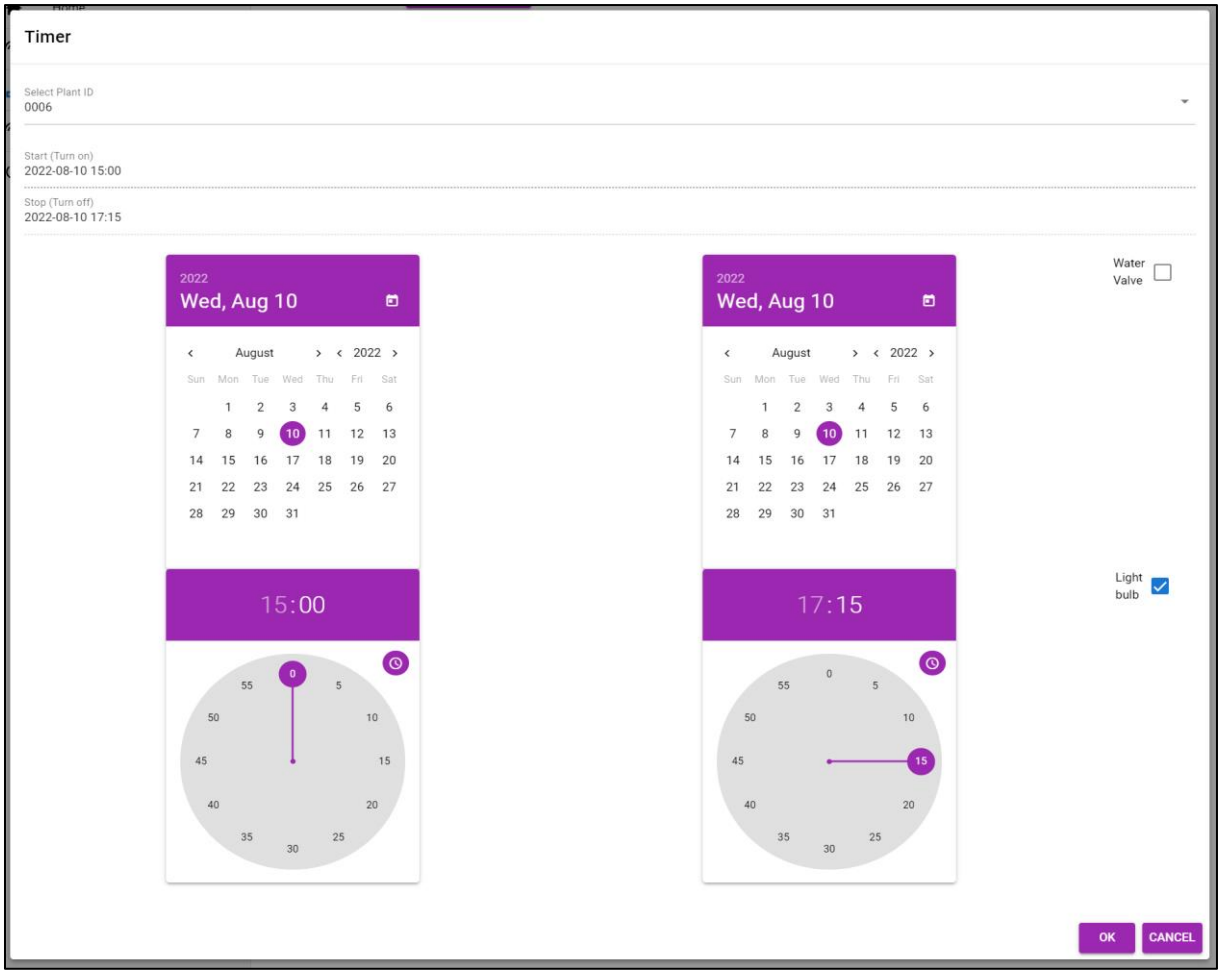
Slika 25. Prozor za dodavanje nove biljke.

5.4. Tajmeri (Timers)

Prilikom otvaranjem ugniježđene stranice „*tajmeri*“, prikazuju se aktivni tajmeri u obliku tablice (Slika 26), svaki redak ima svoji „*plantID*“, tj. jedinstvenu oznaku biljke na koju se on odnosi, početno vrijeme tajmera i krajnje vrijeme tajmera. Tablicu možemo sortirati po vrijednostima.

Slika 26. Tablica tajmera.

Novi tajmer se dodaje pritiskom na gumb „*add timer*“, zatim se otvori novi prozor za unos podataka (Slika 27).



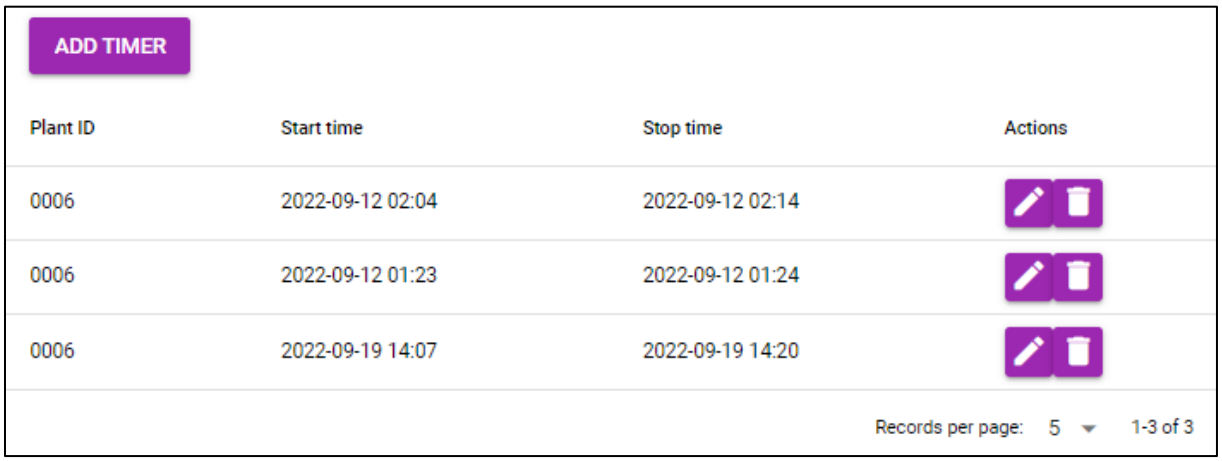
Slika 27. Prozor za dodavanje novog tajmera.

Prvo se označava za koju biljku se tajmer odnosi, zatim se podešava datum i sat paljenja i gašenja releja. Na kraju se bira aktuator kojim se upravlja, vodeni ventil ili žarulja. U ovom primjeru odabrali smo da želimo uključiti žarulju između 15:00 i 17:15 na datum 10.08.2022 za biljku s jedinstvenom oznakom „*0006*“.

Tajmere se također može uređivati i brisati po potrebi.

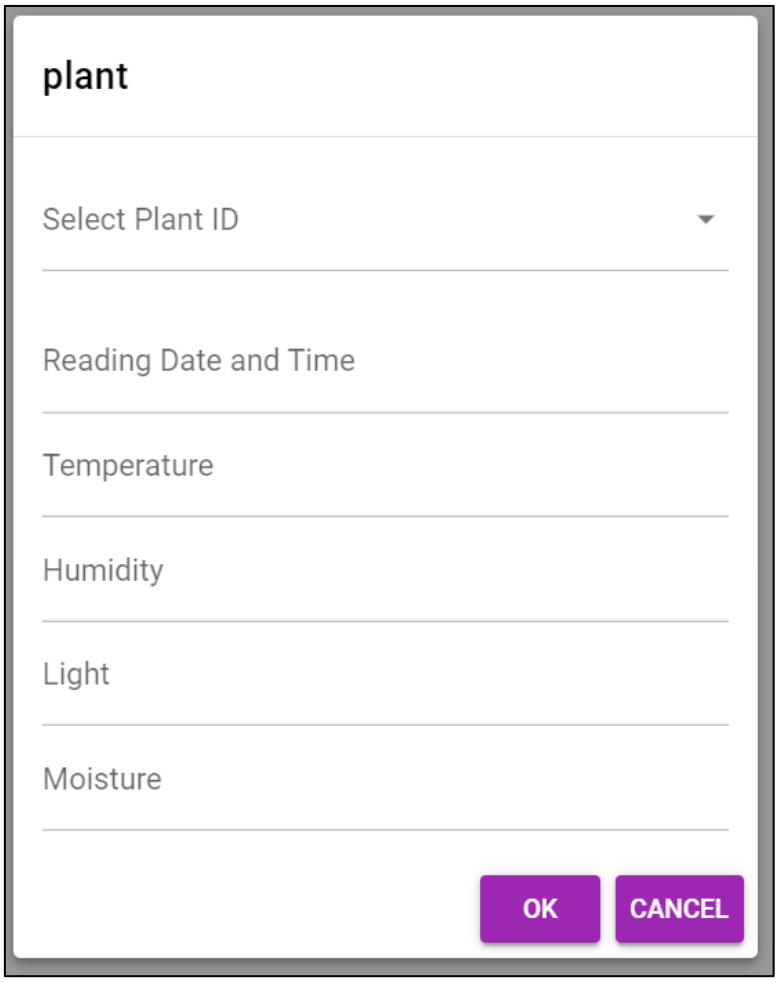
5.5. Očitavanja senzora (Readings)

Pritiskom na tab „*Readings*“, otvara se popis očitavanja svih biljka u obliku tablice (Slika 28). Svaki redak sadrži svoj datum i vrijeme očitavanja, te očitane vrijednosti senzora.



Slika 28. Tablica vrijednosti senzora.

Korisniku je omogućeno ručno unošenje podataka. Pritiskom na gumb *„add timer“* se otvara novi prozor, te se od korisnika traže podatci (Slika 29). Uređivanje i brisanje je također omogućeno.



Slika 29. Prozor za ručno dodavanje očitavanja senzora.

5.6. Odjava iz aplikacije

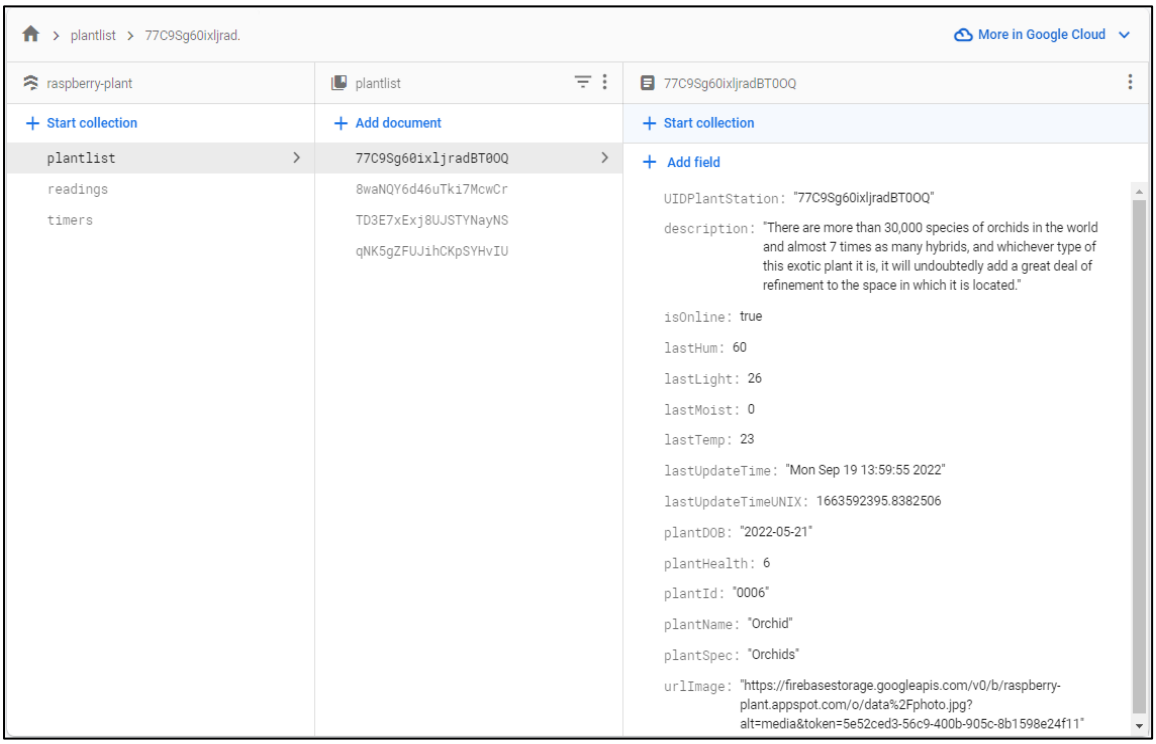
Kada je korisnik gotov korištenja web aplikacije, korisnik se odjavljuje. Pritiskom na gumb *„LogOut“* korisnik se odjavljuje i vrača se na početnu stranicu za prijavu.

6. Struktura baze

Baza se sastoji od kolekcija koja ima dokumente. Svaki dokument ima svoja polja s podatcima. Dokumenti mogu imati i svoje podkolekcije, što se koristi u većim i zahtjevnijim projektima. Bitno je napomenuti da se u većini slučajeva polja dokumenata u jednoj kolekciji ne mijenjaju, već njihova vrijednost. Baze podataka mogu imati jednu ili više kolekcija, a broj dokumenta u jednoj kolekciji ili podkolekciji je u većini slučajeva veći nego jedan. Također je bitno napomenuti da svaki dokument mora imati svoji primaran ključ, tj. jedinstvenu oznaku s kojom se on razlikuje od ostalih dokumenta.

U bazi imamo tri glavne kolekcije:

1. Kolekcija „plantlist“ i dokumenti
   – u ovim dokumentima kolekcije (Slika 30.) ćemo pohraniti biljku i sve njezine podatke polja. Polja dokumenta su jedinstvena oznaka biljke kao i sam naziv dokumenta, zadnje očitavanje senzora, vrijeme zadnjeg očitavanja, podatci o biljci i fotografija biljke.



Slika 30. Kolekcija plantlist i polja dokumenta.

U tablici ćemo detaljno opisati karakteristike polja dokumenta biljke (Tablica 1).

Ime polja	**Tip polja**	**Primjer vrijednosti**	**Opis**
UIDPlantStation	*String*	*4YFh5WAyt7qmrLRrgJie*	*Jedinstvena oznaka*
plantID	*Number*	*9884*	*Jedinstvena oznaka*
plantName	*String*	*Šparuga*	*Ime biljke*
plantSpec	*String*	*Ornamental Grasses*	*Vrsta biljke*
plantHealth	*Number*	*9*	*Stanje biljke*
plantDOB	*String*	*2022-05-23*	*Datum postojanja*
description	*String*	*Sparuga raste u 3 i 4 mj...*	*Opis biljke*
lastHum	*Number*	*87*	*Zadnja vlažnost zraka*
lastLight	*Number*	*41*	*Zadnja svjetlost*
lastTemp	*Number*	*21*	*Zadnja temperatura*
lastMoist	*Number*	*79*	*Zadnja vlažnost tla*
lastUpdateTime	*String*	*Mon May 30 13:52:22 2022*	*Datum očitavanja*
lastUpdateTimeUnix	*Number*	*1653911542.2002113*	*Datum očitavanja Unix format*
urlImage	*String*	*https://firebasestorage...*	*Poveznica* fotografije

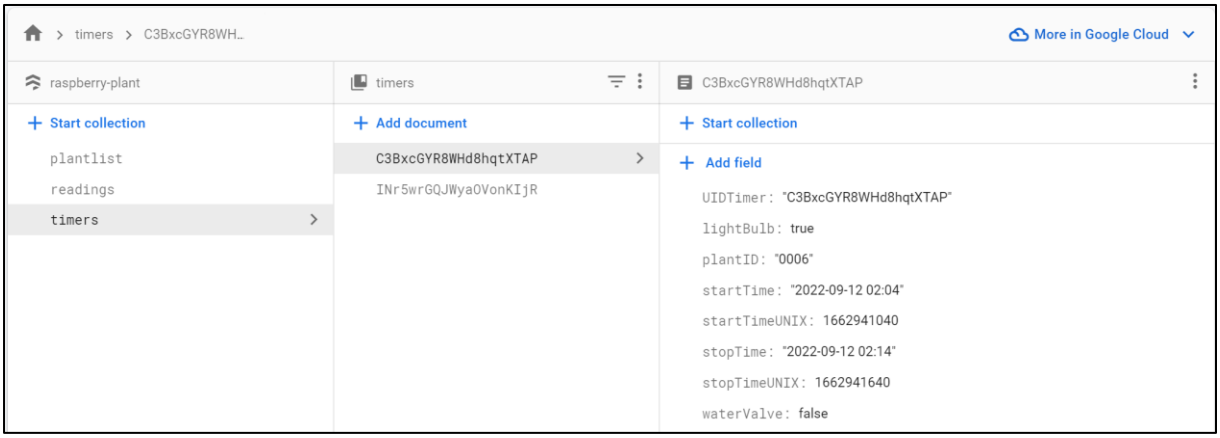
Tablica 1. Karakteristike polja dokumenta biljke.

2. Kolekcija „readings“ i dokumenti

– u ovim dokumentima kolekcije (Slika 31.) ćemo pohraniti očitavanja senzora. Jedinstvenu oznaku očitavanja, oznaku biljke, očitavanje za svaki senzor zasebno i vrijeme očitavanja.

Slika 31. Kolekcija readings i polja dokumenta.

3. Kolekcija „timers“ i dokumenti
   – u ovim dokumentima kolekcije (Slika 32.) ćemo pohraniti podatke tajmera. Jedinstvenu oznaku tajmera i biljke na koju se on odnosi, vrijeme početka i kraja tajmera, te aktuator kojim upravljamo.



Slika 32. Kolekcija timers i polja dokumenta.

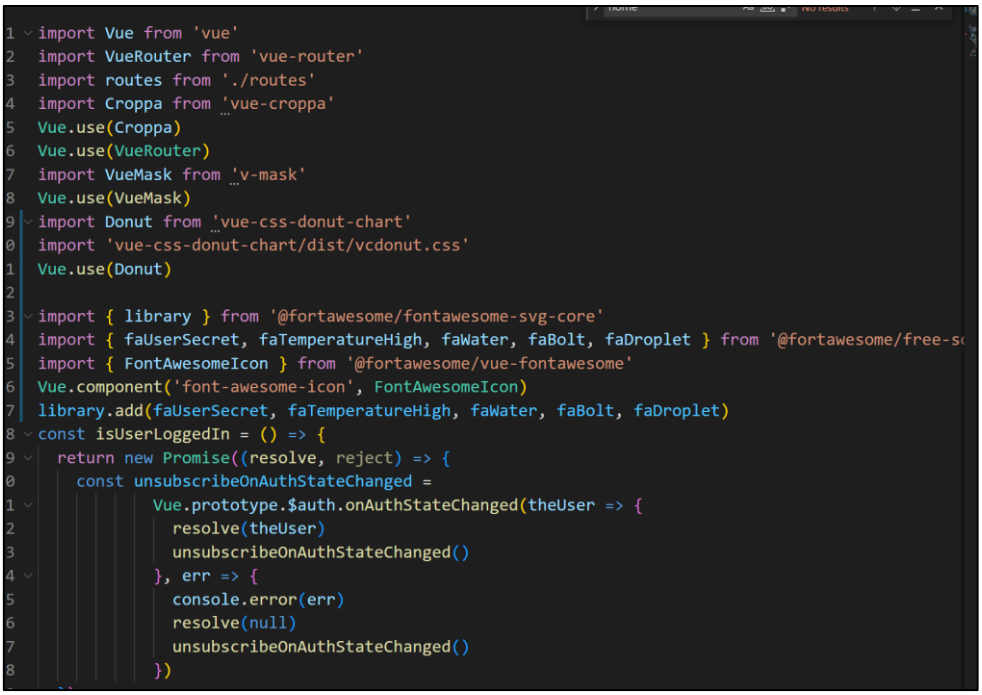
7. Osnovni dijelovi programskog koda web aplikacije

7.1. Index.js

Na samom početku web aplikacije potrebno je definirati koje biblioteke se koriste u daljnjoj aplikaciji.

Iz biblioteke „Vue“ učitavamo glavne funkcije Vue frameworka. „VueRouter“ i „routes“ nam služe kod korištenja putanja. „Croppa“ se koristi za jednostavno upravljanje fotografijama. „VueMask“ služi za definiranje maska u poljima za upis podataka, „Donut“ za korištenje grafova i ostale biblioteke su korištene za font i ikona.

Učitavanje biblioteka i par osnovnih pravila (Slika 33).



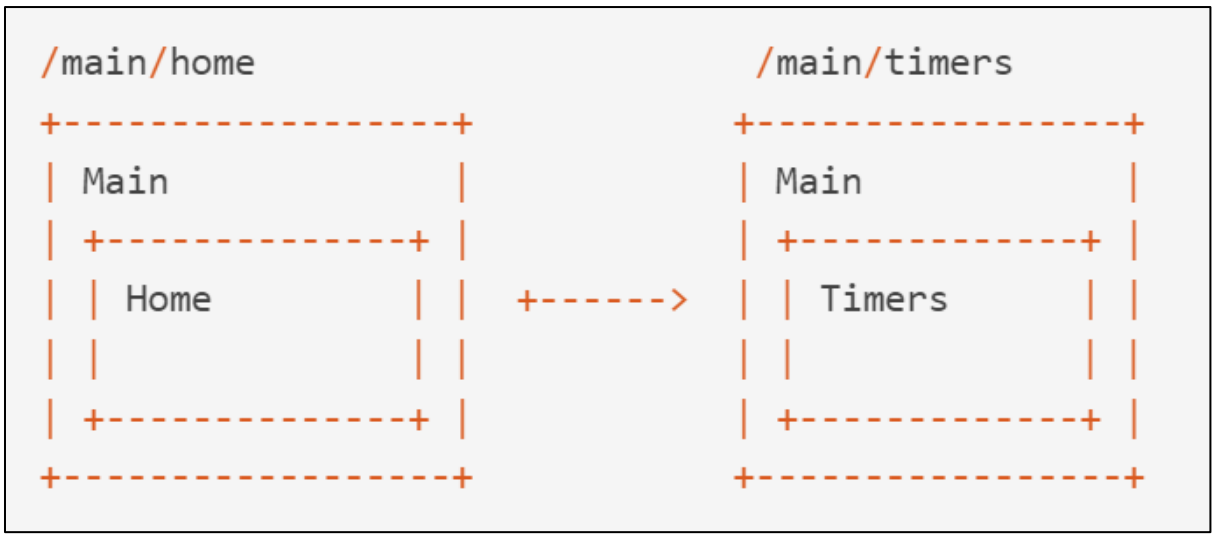
```
import Vue from 'vue'
import VueRouter from 'vue-router'
import routes from './routes'
import Croppa from 'vue-croppa'
Vue.use(Croppa)
Vue.use(VueRouter)
import VueMask from 'v-mask'
Vue.use(VueMask)
import Donut from 'vue-css-donut-chart'
import 'vue-css-donut-chart/dist/vcdonut.css'
Vue.use(Donut)

import { library } from '@fortawesome/fontawesome-svg-core'
import { faUserSecret, faTemperatureHigh, faWater, faBolt, faDroplet } from '@fortawesome/free-s
import { FontAwesomeIcon } from '@fortawesome/vue-fontawesome'
Vue.component('font-awesome-icon', FontAwesomeIcon)
library.add(faUserSecret, faTemperatureHigh, faWater, faBolt, faDroplet)
const isUserLoggedIn = () => {
  return new Promise((resolve, reject) => {
    const unsubscribeOnAuthStateChanged =
            Vue.prototype.$auth.onAuthStateChanged(theUser => {
              resolve(theUser)
              unsubscribeOnAuthStateChanged()
            }, err => {
              console.error(err)
              resolve(null)
              unsubscribeOnAuthStateChanged()
            })
```

Slika 33. Isječak Index.js datoteke.

Bitna metoda u ovoj datoteci je da se nakon svakog preusmjeravanja između stranica verificira da li je korisnik i dalje prijavljen.

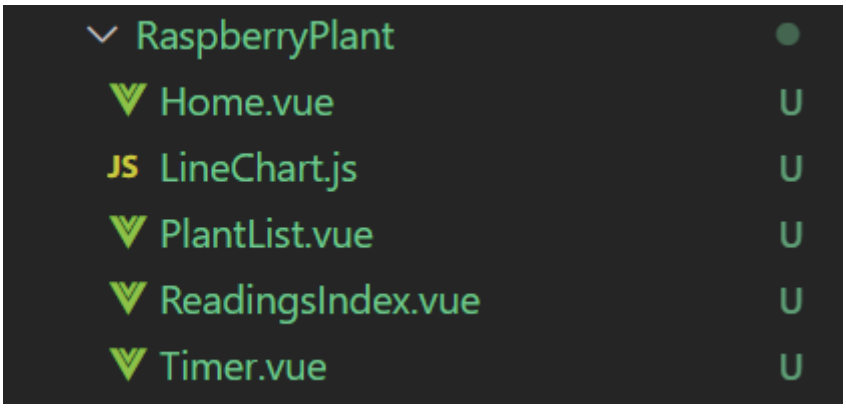
7.2. Putanje i struktura

Web aplikacija je napravljena u obliku gnijezda. To znači da su traka navigacije i bočni izbornik statični, dok se stranice unutar mijenjaju. Slika 34. prikazuje primjer gdje je main traka navigacije i bočni izbornik, a stranice Home i Timers su ugniježđene stranice.



Slika 34. Model gnijezda. [2]

U ovom projektu postoje 4 glavne ugniježđene stranice (Slika 35), od kojih ćemo detaljno objasniti jednu.



Slika 35. Popis 4 glavnih ugniježđenih stranica.

Donji isječak koda (slika 36.) nam govori da nakon uspješne autentifikacije, učitavamo stranicu „*RaspberryPlantLayout.vue*“ (traka navigacije i bočni izbornik), kojemu ćemo ugnijezditi jednu stranicu. U ovom slučaju „*/Home.vue*“ dodatno definirano u „LoginIndex.vue“.

U datoteci „*routes.js*“ postavljamo određena pravila i povezujemo putanje s lokacijom stranica.

```
{
  path: '/Administration',
  component: () =>
    import('layouts/RaspberryPlantLayout.vue'),
  meta: { auth: true },
  children: [
    {
      path: '/PlantList',
      meta: { auth: true },
      component: () =>
        import('pages/RaspberryPlant/PlantList.vue')
    }
```

Slika 36. Isječak koda putanja.

7.3. Autentifikacija s bazom pomoću privatnog ključa

Prije svakog zahtijeva nad bazom podataka, korisnik se treba verificirati. To radi na način da prosljeđuje privatan ključ bazi. Privatni ključ (Slika 37.) ne smije biti javno dostupan. Svatko tko ima prisup privatnom ključu, ima i pristup bazi podataka.

```
const firebaseConfig = {
  apiKey: 'AIzaSyAeG7gjSzNXsEV_yFDYdABMvghYRhuJS0E',
  authDomain: 'raspberry-plant.firebaseapp.com',
  databaseURL: 'https://raspberry-plant-default-rtdb.europe-west1.firebasedatabase.app',
  projectId: 'raspberry-plant',
  storageBucket: 'raspberry-plant.appspot.com',
  messagingSenderId: '94922165141',
  appId: '1:94922165141:web:64e6a0345bb807ada4849c',
  measurementId: 'G-1R3D9CPQTT'
}
```

Slika 37. Privatni ključ za pristup bazi.

7.4. Opis jedne ugniježđene stranice (PlantList.vue)

Vue dokument se dijeli na 2 glavna dijela, „*template*“ koji sadrži HTML strukturu (Slika 38.) i „script“ koji sadrži programski kod.

Stranica je podijeljena na „*kontejnere*“ koji imaju svoje atribute. Klase su atributi pomoću kojih dodjeljujemo izgled kontejneru definirane u vanjskoj datoteci. Koristit ćemo elemente i klase koje su već unaprijed definirane u sklopu Quasar frameworka.

Slika 38. Isječak koda HTML.

U djelu skripta definiramo varijable u koje ćemo kasnije učitavati podatke (Slika 39). Jedna od bitnijih stavki je struktura „*plantStructure*“. Polja dokumenta biljka u bazi podataka i polja strukture su jednaki. Veza između HTML koda i funkcija (metoda) je postignuta interaktivnim elementa kao što su gumbovi, polja za pisanje i klizač.

```
data () {
  return {
    cameraDialog: false,
    myCroppa: {},
    imageSend: null,
    openPlantStationDialog: false,
    PlantStations: [],
    plantGroup: null,
    plantStructure: {
      UIDPlantStation: null,
      urlImage: 'https://images.unsplash.
      plantSpec: null,
      description: null,
      plantDOB: null,
      plantHealth: null,
      plantName: null,
      plantId: null,
      lastTemp: null,
```

Slika 39. Isječak varijabla.

Metode su radnje koje se provode na objektima. Mounted metoda je metoda koja se poziva pri samom učitavanju stranice, izvršava kod koji je definiran unutar mounted funkcije. Async funkcija je funkcija koja će se asinkrono pozvati, što znači da će se funkcija odvijati zasebno od ostatka koda u zasebnoj procesnoj niti. Izraz await je ključna riječ koja se pridodaje funkciji kako bi se ona izvršavala sinkrono i ostatak koda bi čekao na promise kojeg await funkcija vraća.

U mounted metodi definiramo *„collectionRef“* s kolekcijom *„plantlist“*, zatim učitavamo pojedine dokumente *„plantGroup“* i spremamo ih u *„plantStations“* pomoću push funkcije (Slika 40).

```
mounted: async function () {
  const collectionRef = this.$db.collection('plantlist')
  await collectionRef.get().then((PlantStations) => {
    PlantStations.forEach(async (plantGroup) => {
      const PlantStationFromDB = plantGroup.data()
      this.PlantStations.push(PlantStationFromDB)
    })
  })
},
```

Slika 40. Mounted metoda.

Metoda *„onNewPlantStation“* (Slika 41.) učitava strukturu i otvara novi dijaloški prozor za upis podataka o biljci. Podatci su uređeni u JSON formatu. Kada smo upisali sve željene podatke, korisnik stišće na gumb *„OK“* i pohranjuje podatke u bazu tako da poziva funkciju *„onOKPlantStationDialogClick“*.

```
onNewPlantStation () {
  this.plantGroup = JSON.parse(JSON.stringify(this.plantStructure))
  this.openPlantStationDialog = true
},
```

Slika 41. Funkcija gumba nova biljka.

Prvi uvjet će biti izvršena ako definiramo novu biljku. Izrađujemo novi dokument s jedinstvenom oznakom i zatim upisujemo vrijednosti koje je korisnik unio. Ukoliko već postoji biljka s tom jedinstvenom oznakom, tražimo biljku s tom oznakom u skupu „*PlantStation*“ pomoću „*find*“ funkcije. Nakon toga korisnik može mijenjati njezine podatke i pohraniti promijene (Slika 42).

```
onOKPlantStationDialogClick () {
  const collectionRef = this.$db.collection('plantlist')
  if (this.plantGroup.UIDPlantStation === null) {
    collectionRef
      .add(this.plantGroup)
      .then((doc) => {
        var docRef = this.$db
          .collection('plantlist')
          .doc(doc.id)
        docRef
          .update({ UIDPlantStation: doc.id })
          .then((response) => {
            this.plantGroup.UIDPlantStation = doc.id
            this.PlantStations.push(this.plantGroup)
            this.openPlantStationDialog = false
          })
          .catch(function (error) {
            console.error('Error adding document: ', error)
          })
      })
      .catch(function (error) {
        console.error('Error adding document: ', error)
      })
  } else {
    var docRef = this.$db
      .collection('plantlist')
      .doc(this.plantGroup.UIDPlantStation)
    docRef
      .set(this.plantGroup)
      .then((response) => {
        const plantGroup = this.PlantStations.find(
          (plantGroup) =>
            plantGroup.UIDPlantStation ===
            this.plantGroup.UIDPlantStation
        )
        if (plantGroup) {
          for (const attributeName in this.plantGroup) {
            plantGroup[attributeName] = JSON.parse(
              JSON.stringify(this.plantGroup[attributeName])
            )
          }
        }
        this.openPlantStationDialog = false
      })
      .catch(function (error) {
        console.error('Error adding document: ', error)
      })
  }
},
```

Slika 42. Isječak funkcije ok gumba.

Klikom na gumb za uređivanje biljke, poziva se funkcija *„onUpdatePlantStation“* (Slika 43.) s atributom *„plantGroup“*. Učitavamo strukturu i vrijednost varijabli u varijablu *„plantGroup“*. Nadalje otvaramo dijaloški okvir gdje korisnik može mijenjati podatke.

Na sličan način rade ostale ugniježđene stranice kod kojih su samo atributi drukčiji.

```
onUpdatePlantStation (plantGroup) {
  this.plantGroup = JSON.parse(JSON.stringify(this.plantStructure))
  for (const attributeName in this.plantGroup) {
    this.plantGroup[attributeName] = JSON.parse(
      JSON.stringify(plantGroup[attributeName])
    )
  }
  this.openPlantStationDialog = true
},
```

Slika 43. Funkcija gumba uredi.

8. Dodatak za fotografiju pomoću Arduina „ESP32- Cam“

Korisnik možda neće biti stalno u blizini biljke, te ima potrebu provjeriti njezino stanje. Pomoću Arduino *„ESP32-Cam-a“* je omogućen prikaz biljke putem fotografije. Arduino nije u direktnoj komunikaciji s Raspberry Pi-em, već pohranjuje podatke direktno u bazu podataka.

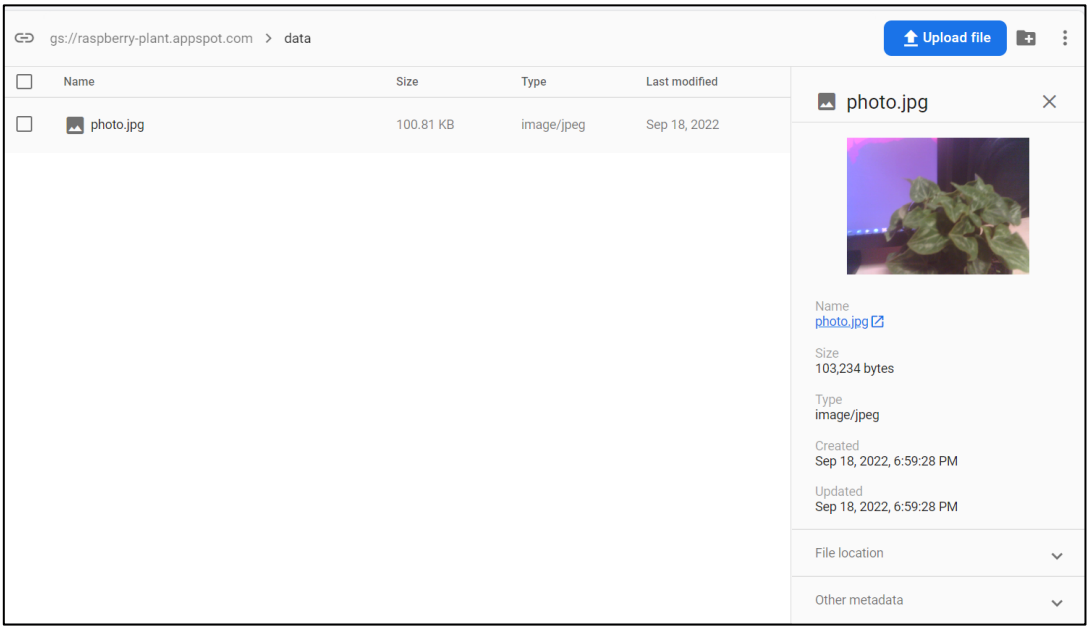
Zadatak mu je da svakih nekoliko minuta napravi novu fotografiju i pohrani ju u bazu podataka. Programski kod je javno dostupan na mreži (poveznica u literaturi *„9.“*) te je dosta trivijalan pa ga nećemo detaljno objašnjavati.

Prvi korak mu je spajanje na Wi-Fi mrežu pomoću funkcije *„initWiFI()“* uz globalne parametre *„SSID“ i „password“* koji su ranije definirani u kodu (Slika 44).

```
void initWiFi(){
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
}
```

Slika 44. Funkcija za povezivanje na Wi-Fi mrežu. [3]

Zatim generira sliku, vrši provjeru pa ju pohrani u oblak i izrađuje poveznicu (*„Download URL“*) za fotografiju koju prosljeđuje bazi podataka (Slika 45).



Slika 45. Mjesto pohrane fotografije u oblak.

Na slici 46. je prikazan isječak programskog koda koji omogućuje slanje podatka u bazu podatka. Definiramo konstantnu putanju biljke. Zatim izrađujemo strukturu s unesenom vrijednost adrese fotografije koje smo prethodno snimili. Ukoliko već postoji adresa fotografije u bazi, zamjenjujemo je adresom nove fotografije i pohranjujemo promijene, u suprotnom pohranjujemo adresu fotografije.

```
String documentPath = "plantlist/77C9Sg60ixljradBT0OQ";
FirebaseJson content;
content.set("fields/urlImage/stringValue", String(fbdo.downloadURL().c_str()).c_str());
if(Firebase.Firestore.patchDocument(&fbdo, FIREBASE_PROJECT_ID, "", documentPath.c_str(), content.raw(), "urlImage")){
  Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
  return;
}else{
  Serial.println(fbdo.errorReason());
}

if(Firebase.Firestore.createDocument(&fbdo, FIREBASE_PROJECT_ID, "", documentPath.c_str(), content.raw())){
  Serial.printf("ok\n%s\n\n", fbdo.payload().c_str());
  return;
}else{
  Serial.println(fbdo.errorReason());
}
```

Slika 46. Isječak koda za unos podataka u bazu. [3]

Pri izvršavanju skripte na Arduinu, u konzoli se ispisuje sljedeće (Slika 47).

```
Connecting to WiFi...
SPIFFS mounted successfully
Token info: type = id token, status = on request
Token info: type = id token, status = ready
Taking a photo...
Picture file name: /data/photo.jpg
The picture has been saved in /data/photo.jpg - Size: 0 bytes
Taking a photo...
Picture file name: /data/photo.jpg
The picture has been saved in /data/photo.jpg - Size: 82432 bytes
Uploading picture...
Download URL: https://firebasestorage.googleapis.com/v0/b/raspberry-plant.a
5 min wait...
```

Slika 47. Konzola Arduina ESP32- CAM.

Bitno je napomenuti da smo umjesto *„ESP32 - CAM“* modula mogli koristiti i kameru koja je namijenjena za platformu Raspberry Pi. Izabrali smo dvije različite platforme tako da možemo dokazati da one mogu raditi neovisno jedna od druge.

9. Zaključak

U ovom završnom radu smo realizirali ideju kako pomoću mikroprocesora nadzirati i upravljati biljkom pomoću aktuatora. Jedan način za to učiniti je pomoću Raspberry Pi platforme s odgovarajućim senzora, aktuatora i web aplikacije. Opisali smo sve korake koji su potrebni za realizaciju sustava, sve od definiranja baze podataka, izgradnje web aplikacije i programiranja skripte na Raspberry Pi-u i Arduino platformi.

U današnje vrijeme sve više stvari se pokušava automatizirati. Brzi dohvat informacija s par klikova na pametnom telefonu ili računalu je potreba.

10. Popis literature

Internet (25.09.2022.):

1. https://firebase.google.com/docs/
2. https://quasar.dev/
3. https://randomnerdtutorials.com/esp32-cam-save-picture-firebase-storage/
4. https://iot-school.veleri.hr/en/home/ (Dokumentacija IoT projekta)
5. https://wiki.seeedstudio.com/Grove-Sunlight_Sensor/
6. https://github.com/Seeed-Studio/Seeed_Python_SI114X
7. https://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-the-raspberry-pi/
8. https://www.techcoil.com/blog/how-to-read-soil-moisture-level-with-raspberry-pi-and-a-yl-69-fc-28-moisture-sensor/
9. https://www.hackster.io/talofer99/analog-input-on-raspberry-pi-with-mcp3008-e64e43
10. https://github.com/mobizt/Firebase-ESP-Client
11. https://cdn-shop.adafruit.com/datasheets/MCP3008.pdf
12. https://nodejs.org/en/docs/
13. https://github.com/espressif/arduino-esp32
14. https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#raspberry-pi-3-model-b

11. Popis slika