Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Lea Jarnjak Levstek

Pametni terarij

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Ljubljana, 2019

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Lea Jarnjak Levstek

Pametni terarij

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: viš. pred. dr. Aljaž Zrnec

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V zadnjih nekaj letih se na številnih področjih pojavlja vedno večja potreba po pametnih napravah, povezanih v internet. V okviru diplomske naloge predstavite pametni terarij in realizirajte sistem, ki bo preko senzorjev zajemal podatke in omogočal avtomatsko in ročno krmiljenje nastavitev za vzdrževanje ustreznih življenjskih pogojev v terariju. Za realizacijo sistema uporabite računalnik Raspberry Pi.

Zahvalila bi se mojemu mentorju, viš. pred. dr. Aljažu Zrnecu, za vso podporo pri izdelavi diplomske naloge. Posebna zahvala gre moji družini in prijateljem za spodbudo skozi vsa leta šolanja.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
ADC	analog-to-digital converter	analogno-digitalni pretvornik
CSS	Cascading Style Sheets	kaskadne stilske predloge
GPIO	General Purpose Input Output	splošno namenski vhod/izhod
HDMI	High-Definition Multimedia Interface	visoko-ločljivostni multimedijski priključek
HTML	Hyper Text Markup Language	označevalni jezik
IoT	Internet of things	medomrežje stvari
I^{2}C	Inter-Integrated Circuit	serijski vmesnik in protokol
PWR	power	napajanje
SPI	Serial Peripheral Interface Bus	sinhrono serijsko vodilo
SSH	Secure Shell	protokol za varno mrežno povezavo
SFTP	SSH File Transfer Protocol	protokol za varen prenos datotek
USB	Universal Serial Bus	vsestransko zaporedno vodilo
UART	Universal asynchronous receiver/transmitter	univerzalni asinhroni sprejemnik/oddajnik
Wi-Fi	Wireless Local Area Networking	lokalno brezžično omrežje
WSGI	Web Server Gateway Interface	univerzalni vmesnik za komunikacijo spletnega strežnika

Povzetek

Naslov: Pametni terarij

Avtor: Lea Jarnjak Levstek

V sklopu diplomske naloge smo razvili prototip sistema pametnega terarija, ki omogoča avtomatski nadzor nad ogrevanjem, hlajenjem in zalivanjem terarija. Pri tem smo uporabili računalnik Raspberry Pi 3 model B. Ta deluje kot spletni strežnik za našo spletno aplikacijo, preko katere lahko nadziramo delovanje zunanjih naprav, kot so luči, ventilatorji, vodna črpalka in toplotni grelec. Uporabnik ima možnost pregleda nad trenutnim stanjem temperature, vlage v zraku in zemlji znotraj terarija ali pa si ogleda zgodovino meritev v določenem časovnem obdobju. Spletna aplikacija in zaledni sistem temeljita na programskem jeziku Python in ogrodju Flask. Podatki o meritvah so shranjeni v podatkovni bazi SQLite, nadzor nad zunanjimi napravami pa upravljamo s pomočjo relejnega modula.

Ključne besede: avtomatizacija, IoT, Raspberry Pi, senzorji, pametni terarij, terarij.

Abstract

Title: Smart terrarium

Author: Lea Jarnjak Levstek

The purpose of this thesis is to develop a prototype system for a smart terrarium with automatic control of heating, cooling and watering. We have used Raspberry Pi 3 Model B computer. It serves as a web server for our web application with which we can control devices such as lights, fans, water pump and heater. User can check status of temperature, humidity and soil moisture in the terrarium and can see a history of readings in a certain time interval. Web application and backend are based on Python programming language and Flask framework. Data readings are stored in a SQLite database, while control over external devices is managed with a relay module.

Keywords: automation, IoT, Raspberry Pi, senzors, smart terrarium, terarrium.

Poglavje 1

Uvod

Terarij je po navadi zaprt prostor, v katerem gojimo rastline ali živali za opazovanje in raziskave [2]. V njem poskušamo simulirati življenjske pogoje, ki jih živi organizmi potrebujejo za preživetje, zato je zelo pomembna pravilna vzpostavitev pravih pogojev za dobro zdravje in rast organizmov. Za vzpostavitev umetnega okolja je potrebne veliko potrpežljivosti in znanja. Najlepši in najtežje vzdržljivi terariji so tropski terariji, saj jih je potrebno ves čas nadzorovati. V terariju moramo uravnavati svetlobo, temperaturo, vlago, paziti moramo na vlažnost zemlje in imeti dober zračni pretok [2]. Težavnost oskrbe pa se bistveno poveča pri plazilcih, saj so ti zelo občutljivi in lahko že majhna sprememba razmer v terariju ogrozi njihovo zdravje [8]. Primer pol puščavskega terarija z živimi rastlinami prikazuje Slika 1.1.

Živimo v času, ko je tehnologija v polnem zagonu in je vedno več povpraševanja po pametnih napravah [4]. Potreba po digitalnem nadzoru in avtomatizaciji se povečuje, saj želimo svoj čas čim bolje izkoristiti. Različne naprave in izdelki za upravljanje vseh pogojev v terariju že obstajajo, vendar so cenovno zelo neugodne, potrebujemo jih več in na žalost ni neke skupne poceni rešitve, ki bi nam hkrati tudi olajšala nadzor nad terarijem. Ker vzdrževanje stalnih pogojev vzame veliko časa, smo se odločili izdelati pametni terarij. Osredotočili se bomo na prototip pametnega terarija s sistemom z nadzorom temperature, kroženja zraka, svetlobe ter vlage v zraku in ze-

Slika 1.1: Dom leopardjega gekona.

mlji. Pri tem bomo poskušali izdelek narediti kar se da cenovno ugodnega. Za jedro projekta smo uporabili Raspberry Pi 3 Model B [40], nanj pa smo priklopili različne senzorje za temperaturo in vlago. Preko spletne aplikacije bomo imeli pregled in nadzor nad pogoji v terariju. Upravljali bomo lahko z zunanjimi napravami, in sicer z lučmi, ventilatorji, vodno črpalko in s toplotnim grelcem.

Poglavje 2

Predstavitev trenutnih rešitev nadzora nad terarijem

Terarij smo do sedaj nadzorovali sami ali pa smo za to imeli različne naprave. Vsako napravo je bilo potrebno ločeno nastavljati in med seboj niso bile povezane. To je pomenilo, da smo potrebovali več takih naprav, poleg tega pa večina tudi ni imela možnosti nadzora preko interneta. Ob kakršni koli potrebni spremembi smo morali biti vedno prisotni ob terariju, hitro pa se je zgodilo, da so se pogoji v terariju rizično poslabšali in ušli izpod nadzora.

Za nadzor luči smo uporabljali digitalna ali analogna časovna stikala (Slika 2.1). Čas vklopa in izklopa je bilo potrebno nastavljati ročno, poleg tega pa smo bili omejeni s priklopom ene luči na eno stikalo. Zaradi spreminjanja dolžine dneva smo morali vedno znova nastavljati čas izklopa in vklopa luči.

Temperaturo smo spremljali z različnimi termometri, nameščenimi v samem terariju. Vzdrževali smo jo s pomočjo ročnega ugašanja in prižiganja luči. Pozimi je nastal večji problem, saj je bilo potrebno temperaturo višati še z dodatnimi napravami, kot so grelni kabli ali toplotni grelci. Ob tem smo morali paziti na temperaturo tal, ki se je lahko hitro preveč povišala. To smo nadzorovali z digitalnimi termostati, ki so odvisno od temperature ugašali ali prižigali talne grelce (Slika 2.2).

Slika 2.1: Digitalno in analogno časovno stikalo.

Poleti, ko se zunanje temperature dvignejo zelo visoko, smo morali poskrbeti za učinkovito hlajenje našega terarija. To smo na enak način reševali z ročnim prižiganjem in ugašanjem ventilatorjev. S tem se nam je zmeraj porušila vlažnost zraka in smo jo morali dvigovati s pršenjem vode po notranjosti terarija. Vodne črpalke za časovno uravnano pršenje vode že obstajajo, vendar je bilo v določenih primerih ob vključitvi pršenja prisotna že zadostna vlažnost. Terarij se je tako po nepotrebnem zalil in vlažnost je narasla. Z nenatančnim nadzorom so se stroški oskrbe za vzdrževanje večali, pogoji v terariju pa so se hitro rizično spremenili, kar je zelo ogrozilo naše rastline in živali.

Nadzor nad terarijem še vedno zahteva veliko naše prisotnosti in potrpežljivosti za pravilno vzdrževanje pogojev v terariju. Na žalost je trenutno najcenejša možnost uporaba časovnih stikal in temperaturnih termostatov, ki pa so nepraktični za učinkovito vzdrževanje pogojev v terariju, trenutno

Slika 2.2: Termostat za nadzor temperature.

dostopni izdelki pametnih terarijev pa so zelo dragi ali pa so še vedno v fazi razvoja. Ravno zaradi tega bomo poskusili sami izdelati prototip poceni sistema, ki bo združil ves nadzor nad pogoji v terariju v celovito rešitev. Cilj diplomske naloge je izdelati cenovno ugoden pametni terarij z nadzorom na daljavo.

2.1 Obstoječe rešitve pametnega terarija

Na trgu že obstajajo produkti pametnih terarijev, ki pa so cenovno zelo neugodni ali pa so še vedno v fazi razvoja.

Za primer lahko vzamemo pametni terarij Biopod [33], ki ga bomo kasneje podrobneje primerjali z našo rešitvijo v Poglavju 6. Z nakupom pametnega terarija Biopod pridobimo steklen terarij z že vgrajenim sistemom za nadzor pogojev znotraj terarija (Slika 2.3). Upravljamo ga lahko preko mobilne aplikacije, na voljo pa je v različnih velikostih. Vse komponente sistema za nadzor ima že vgrajene znotraj terarija. Primeren je za gojenje rastlin in

živali.

Slika 2.3: Pametni terarij Biopod.

Slika 2.4: SmartTerra pametni terarij.

Podoben produkt je pametni terarij SmartTerra [35], ki pa je namenjen predvsem gojenju rastlin (Slika 2.4). Steklen terarij ima vgrajene zvočnike za predvajanje zvočnih posnetkov iz narave. Tudi tega lahko upravljamo preko

mobilne aplikacije, omogoča pa tudi simulacijo nevihte, sončnega vzhoda in zahoda. Produkt bo na voljo predvidoma v prvi četrtini leta 2019.

Slika 2.5: Produkt OrchidBox.

Gojenje rastlin lahko nadzorujemo tudi s pomočjo majhnega pametnega terarija OrchidBox (Slika 2.5) [34]. Vgrajeno ima zaznavanje vlažnosti zemlje, s katerim nas lahko opozori, kdaj je potrebno rastline zopet zaliti. Preko mobilne aplikacije pa lahko uredimo nastavitve za osvetlitev in spremljamo količino vode v terariju na daljavo. Tudi ta produkt bo na voljo v letošnjem letu 2019.

Pri vseh zgoraj naštetih produktih smo omejeni na določene dimenzije pametnega terarija z že vgrajenim sistemom za nadzor nad pogoji v terariju. Nismo pa zasledili, da bi bilo na trgu mogoče kupiti samostojen produkt sistema za nadzor terarija. Na voljo so projekti domače izdelave za nadzor nad terariji ali akvariji, ki so jih ljudje sami izdelali in delili z javnostjo. Za primer lahko vzamemo projekt Raspberry Pi Terrarium Controller [37], s katerim s pomočjo senzorjev in računalnika Raspberry Pi nadzorujemo pametne vtičnice.

Poglavje 3

Opredelitev zahtev

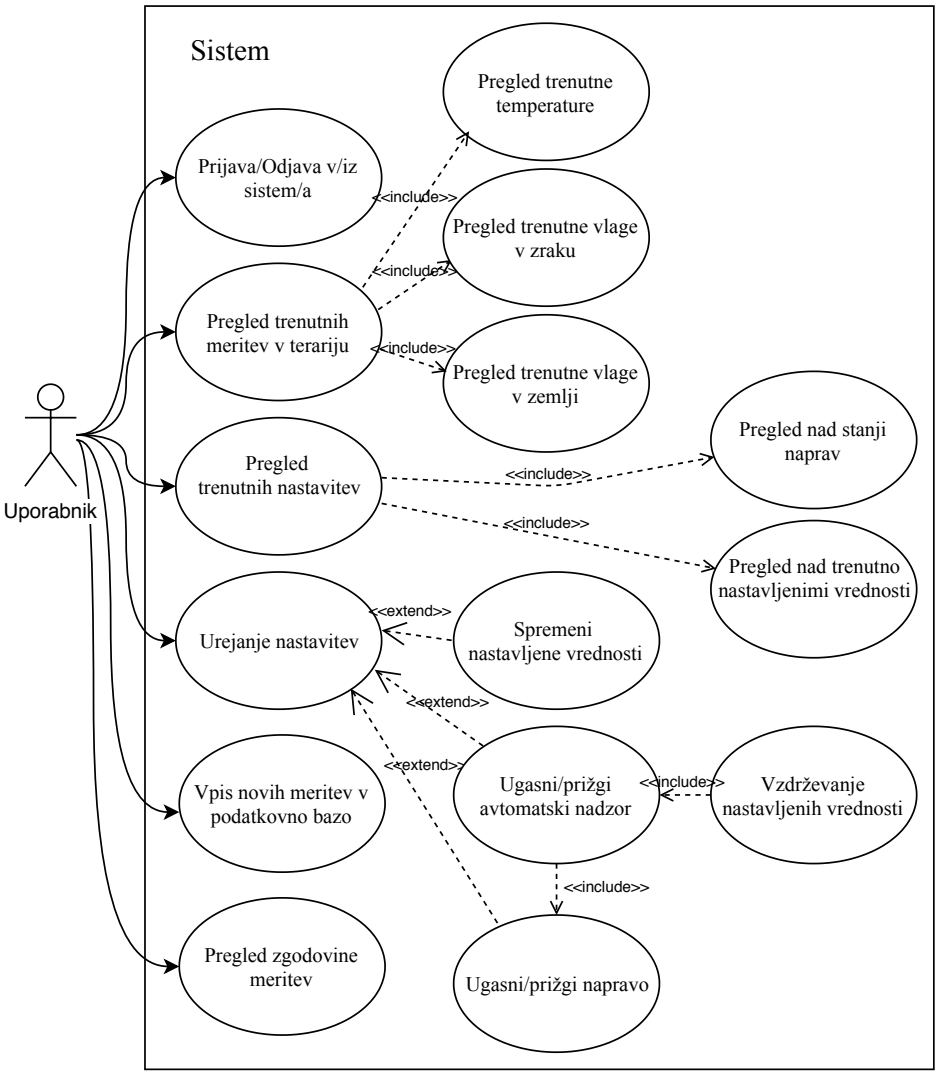
Pred začetkom načrtovanja našega sistema smo naredili analizo zahtev izdelave pametnega terarija. Glede na to, da smo že imeli skoraj vse komponente terarija, kot so luči, toplotni grelec, vodna črpalka, steklen terarij, smo se osredotočili na izdelavo sistema, ki bo nadzoroval in vzdrževal pogoje terarija, katere smo omenili v prejšnjem poglavju.

Naš projekt smo tako razdelili na dva dela. Prvi del bo zajemal vso strojno opremo. Potrebovali bomo računalnik, ki bo skrbel za celoten nadzor nad našim terarijem. Z njim bomo poganjali spletno aplikacijo, skrbel bo za komunikacijo s senzorji ter upravljal s povezanimi zunanjimi napravami, kot so luči, grelec, ventilatorji in vodna črpalka. Na računalniku bomo poganjali celotno programsko kodo sistema, ki predstavlja drugi del našega projekta.

Odločili smo se za izdelavo spletne aplikacije, saj jo bomo potrebovali za komunikacijo med uporabnikom sistema pametnega terarija in jo lahko kasneje prilagodimo tudi za mobilne naprave. Končni uporabnik bo lahko upravljal z napravami, povezanimi v naš sistem. Zasnovati smo morali tudi podatkovno bazo, ki bo poskrbela za shrambo podatkov, s katerimi bomo upravljali med samim delovanjem.

Sistem bo tako omogočal pregled nad trenutnimi meritvami temperature ter vlage v zraku in zemlji znotraj terarija. Prikazane bodo trenutno nastavljene nastavitve pametnega terarija in stanje priklopljenih naprav. Na voljo

pa bo tudi celoten pregled nad zgodovino meritev pametnega terarija znotraj določnega časovnega okvira. Vse nastavljene nastavitve bomo lahko poljubno spremenili, v podatkovno bazo pa bomo lahko zabeležili nove meritve. Sistem bo omogočal regulacijo nastavljenih nastavitev z vklopom avtomatskega načina. Funkcionalnosti sistema prikazuje diagram primerov uporabe na Sliki 3.1.



Slika 3.1: Diagram primerov uporabe našega sistema.

Funkcionalnosti sistema:

- Prijava/Odjava v/iz sistema – Uporabnik sistema se ob prvem dostopu na spletno aplikacijo lahko prijavi z uporabniškim imenom in geslom. Uporabnik se lahko iz sistema kadar koli tudi odjavi.
- Pregled trenutnih meritev v terariju – Prijavljen uporabnik ima pregled nad trenutnimi meritvami temperature, vlage v zraku in zemlji ter časom in datumom zadnjih meritev.
- Pregled trenutnih nastavitev – Prijavljen uporabnik ima možnost pregleda nad trenutnimi nastavitvami pametnega terarija, kot so stanja naprav, trenutno nastavljene vrednosti temperature, vlage v zraku, vlage v zemlji, čas vklopa in čas izklopa luči.
- Urejanje nastavitev – Prijavljen uporabnik lahko spremeni nastavljene vrednosti, kot so temperatura, vlaga v zraku, vlaga v zemlji, čas vklopa in izklopa luči. Na voljo ima tudi nadzor nad priklopljenimi napravami – ugasne/prižge lahko luči, vodno črpalko, toplotni grelec in ventilatorje. Prižge in ugasne lahko avtomatski nadzor nad terarijem, kar sproži vzdrževanje nastavljenih vrednosti terarija – regulacijo temperature, vlage v zraku in zemlji.
- Vpis novih meritev v podatkovno bazo – Prijavljen uporabnik lahko shrani nove meritve temperature, vlage v zraku in zemlji v podatkovno bazo.
- Pregled zgodovine meritev – Prijavljen uporabnik lahko pogleda zgodovino meritev pametnega terarija. Pogleda lahko zgodovino meritev temperature, vlage v zraku in vlage v zemlji v določenem časovnem obdobju.

Poglavje 4

Uporabljena strojna in programska oprema

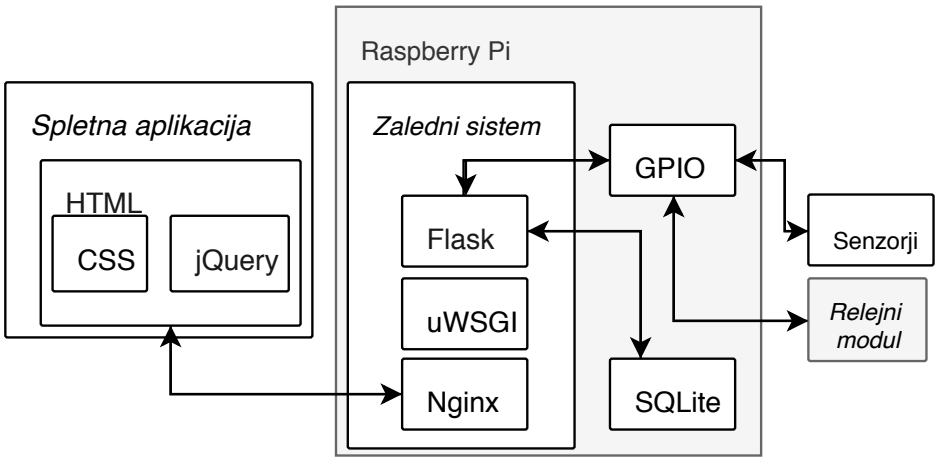
4.1 Izbira strojne platforme in arhitektura sistema

Odločili smo se, da bomo za potrebe implementacije naše rešitve uporabili mini računalnik Raspberry Pi 3 Model B, tretje generacije. Od predhodnih različic ima ta boljši procesor, dodani pa sta WiFi in Bluetooth povezljivosti [5]. Ti dve se nam lahko kasneje izkažeta za koristne, saj nam lahko ponudita lažjo povezavo s senzorji v terariju. Raspberry Pi 3 Model B predstavlja jedro sistema. Imel bo tri glavne naloge, in sicer da bo deloval kot spletni strežnik, da bo upravljal relejni modul za nadzor nad priklopljenimi napravami ter da bo bral podatke iz senzorjev in jih tudi ustrezno prikazoval na spletni strani [6].

Za operacijski sistem smo izbrali Raspbian verzijo Jessie, ki temelji na Debian distribuciji operacijskega sistema Linux [23]. Je uradni operacijski sistem za Raspberry Pi 3 Model B. Vsebuje komplet osnovnih programov in pripomočkov, ki pripomorejo k lažjemu začetku uporabe. Kot eden izmed največkrat uporabljenih operacijskih sistemov ima na spletu na voljo veliko zbirko dokumentacije [25].

Za spletni strežnik smo uporabili Nginx [48], ki zahteva nizko porabo sistemskih sredstev. Ima zelo dobro dokumentacijo in podporo ter velja za enega izmed bolj priljubljenih spletnih strežnikov. Na spletnem strežniku deluje zaledni sistem, ki temelji na Python [36] ogrodju Flask [21], ta pa je povezan s podatkovno bazo SQLite [31]. V njej shranjujemo vse prebrane podatke iz senzorjev, ki se pred vnosom v bazo ustrezno obdelajo. Za uspešno komunikacijo spletnega strežnika s spletno aplikacijo smo namestili vmesnik uWSGI [50].

Spletna stran je zgrajena in prikazana s pomočjo označevalnega jezika HTML [14], oblikovana pa je s pomočjo stilskega jezika CSS [10] ter Javascript knjižnico jQuery [19]. Za enostavno in hitro vizualizacijo preteklih meritev pa uporabljamo ogrodje Google Charts [22]. Zaledni sistem, odvisno od uporabnikove akcije na spletni strani, sproži določene klice funkcije znotraj Flask aplikacije. Arhitekturo ter gradnike našega celotnega sistema lahko vidimo na Sliki 4.1.



Slika 4.1: Arhitektura sistema.

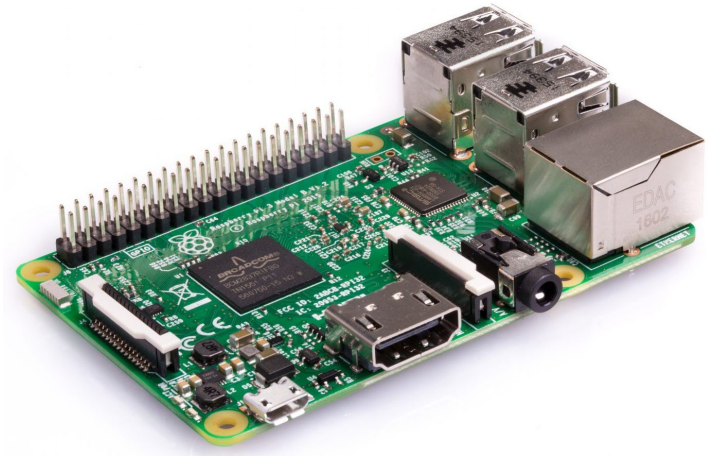
Uporabnik lahko na spletni strani spremeni nastavljene nastavitve pametnega terarija z vnosom novih vrednosti v vnosna polja ali pa določi novo stanje priklopljenih naprav s pritiskom na gumbe. S spremembo nastavitev se novi podatki pred vnosom v podatkovno bazo preverijo. Če preverjanje podatkov uspe, se ti shranijo, s tem pa začnejo veljati tudi nove spremembe

pri delovanju. Spletna stran ima poleg spreminjanja nastavitev na voljo celoten pregled nad trenutnimi in preteklimi meritvami pametnega terarija. Spremljamo lahko zgodovino podatkov o temperaturi, vlažnosti zraka in zemlje, ki so prikazani z grafi. Možno je tudi določanje drugega časovnega okvira zgodovine meritev s senzorjev.

4.2 Strojna oprema

Strojna oprema je naš prvi del, ki smo ga zasnovali. V nadaljevanju so na kratko predstavljene uporabljene strojne komponente.

4.2.1 Računalnik Raspberry Pi 3 Model B

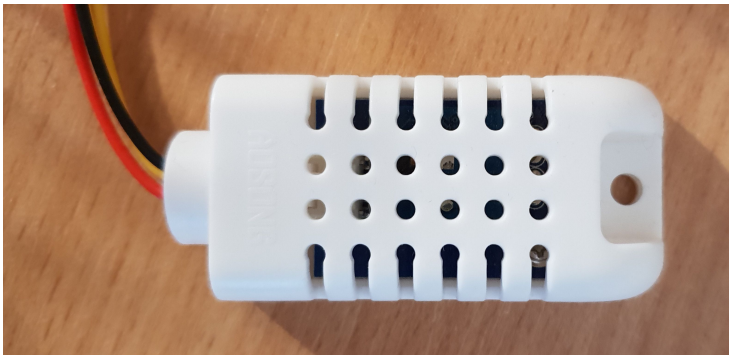


Slika 4.2: Raspberry Pi 3 Model B.

Raspberry Pi 3 model B (v nadaljevanju RPi3) je cenovno ugoden mini računalnik v velikosti bančne kartice (Slika 4.2). Leta 2012 ga je razvila dobrodelna ustanova Raspberry Pi Foundation s ciljem širjenja in izobraževanja ljudi o osnovah računalniškega znanja. Kljub njegovi nizki ceni in majhnim dimenzijam je zmogljiv, saj ga lahko uporabljamo za enake stvari kot običajen računalnik. Omogoča brskanje po spletu, igranje računalniških iger, predvajanje video posnetkov, glasbe in pisanje dokumentov. Odločili smo se za

uporabo računalnika RPi3, ki ima dodani brezžični povezavi WiFi 802.11n [15] in Bluetooth 4.1 [46]. RPi3 je izšla leta 2016, leta 2018 pa so izdali boljšo in dražjo različico Raspberry Pi 3 Model B+ [1]. RPi3 vsebuje 40 GPIO (ang.: General Purpose Input Output) pinov, ki omogočajo komunikacijo z zunanjim svetom [41]. Od vseh pinov jih je 26 splošno namenskih vhodno-izhodnih, dva imamo za napajanje z napetostjo 5 V DC (ang. direct current), dva za napajanje z napetostjo 3 V DC ter 8 za ozemljitev. Preostali pini pa omogočajo priklop z drugimi vodili, kot so UART, SPI in I^2C. Pine lahko enostavno sprogramiramo, da podatke preko njih pošiljamo ali pridobivamo.

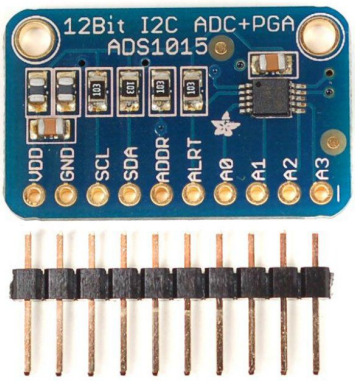
4.2.2 Senzor za merjenje temperature in vlage AM2302



Slika 4.3: Senzor za merjenje temperature in vlage AM2302.

Za merjenje temperature in vlage smo uporabili digitalni senzor AM2302 (Slika 4.3). Uporablja kapacitiven senzor vlage in termoupor za merjenje okoliškega zraka. V primerjavi s cenovno ugodnejšim DHT11 senzorjem je ta bolj natančen in zanesljiv. Njegova natančnost pri merjenju vlage je 2—5 %. Temperaturni obseg senzorja je od -40 °C do +80 °C z natančnostjo +- 0,5 °C. Plastično ohišje pripomore k lažji postavitvi senzorja v terariju. S tem preprečimo stik vezja z živalmi ali rastlinami [44].

4.2.3 Analogno-digitalni pretvornik ADS1015



Slika 4.4: Analogno-digitalni pretvornik ADS1015

Ker RPi3 ne omogoča branja analognih signalov, smo zato uporabili Adafruit 4-kanalni 12-bitni analogno-digitalni pretvornik (Slika 4.4). Tega smo potrebovali za naš senzor vlažnosti, ker smo hoteli še večjo natančnost pri merjenju vlage v zemlji [9].

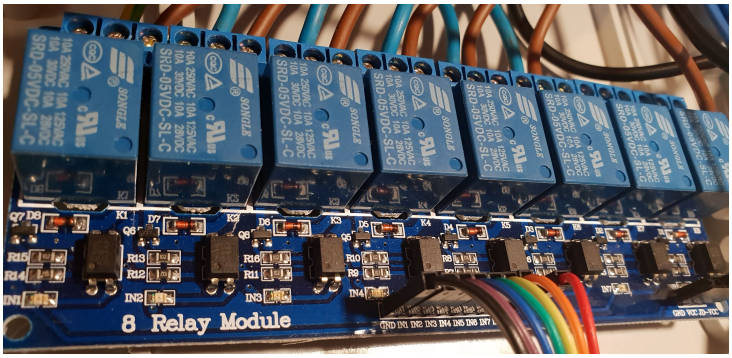
4.2.4 Senzor za merjenje vlažnosti v zemlji SEN-13637

Slika 4.5: Senzor za merjenje vlažnosti v zemlji.

Za merjenje vlažnosti v zemlji smo izbrali analogni senzor SEN-13637 SparkFun Soil Moisture Sensor (Slika 4.5). Princip delovanja senzorja temelji

na sporočanju upornosti zemlje glede na količino vode v njej. Večja kot je upornost, manj je zemlja vlažna. Izbrani senzor ima na merilnih sondah dodatno zaščito pred korozijo [43].

4.2.5 Osemkanalni relejni modul



Slika 4.6: Relejni modul.

V diplomski nalogi nadziramo luči, vodno črpalko ter toplotni grelec, katere je potrebno priključiti na električno napetost med 220 in 240 V. Za nadzor nad priklopljenimi zunanjimi napravami smo uporabili osemkanalni 5 V relejni modul, ki je primeren za uporabo z RPi3 (Slika 4.6). Rele je elektromagnetno stikalo, ki ga krmili tok skozi magnetno navitje, pri čemer sta krmilni in močnostni tokokrog galvansko ločena [7]. Krmilni tokokrog se priključi na RPi3 [28].

4.2.6 Zunanje naprave

Zunanje naprave, ki smo jih uporabili za terarij, so:

- računalniški ventilatorji z napetostjo 12 V – za zračni pretok;
- vodna črpalka – Exo Terra Repti Flo 200 [51], 220 V–240 V – za dovod vode;

- 13 W žarnica – Exo Terra Natural Light [13] in stojalo za luč – Exo Terra Compact top, 230 V – za dobro osvetlitev [11];
- 16 W grelna podloga – Exo Terra Heat mat, 220 V–240 V – za ogrevanje terarija [12].

4.3 Uporabljene tehnologije, orodja in programska oprema

V okviru tega poglavja so predstavljene tehnologije in orodja, ki smo jih uporabili pri izdelavi pametnega terarija.

4.3.1 Jezik Python

Python je eden izmed najbolj priljubljenih objektno orientiranih programskih skriptnih jezikov. Programiranje v Pythonu je zelo hitro, koda pa je berljiva, kar je zelo pomembno pri večjih projektih [39]. Primeren je tako za začetnike, kot tudi za naprednejše programerje. V njem so napisane tudi večje aplikacije (npr. Youtube, Netflix). Jezik je enostavno razširljiv ter podprt z raznovrstnimi knjižnicami, ki nam omogočajo lažjo implementacijo najrazličnejših funkcij. V okviru diplomske naloge smo izbrali Python različice 2.7. Ta ima več podpore kot novejši Python 3.0, ki ni kompatibilen z našimi knjižnicami za nadzor nad senzorji.

4.3.2 Ogrodje Flask

Flask je minimalistično ogrodje za Python, ki se zanaša na pogon za predloge Jinja2 in knjižnico Werkzeug [52] – zbirko WSGI (ang. Web Server Gateway Interface) pripomočkov za programski jezik Python. Je preprost za uporabo in privzeto za delovanje ne zahteva nobenih dodatnih knjižnic. Omogoča poljubno dodajanje razširitev, te pa se posodabljajo bolj redno in hitreje kot samo jedro Flask ogrodja.

4.3.3 Jinja2

Jinja2 je eden izmed najbolj uporabljenih pogonov za predloge za Python. Vsebuje dodaten izrazni jezik, ki ponuja večjo izbiro orodij za kreiranje predlog. Uporablja se kot primarni pogon za predloge za Flask in je kompatibilen z večimi Python različicami. Z njim lahko generiramo oznake, podpira pa tudi vrsto matematičnih funkcij [32].

4.3.4 Spletni strežnik Nginx

Za spletni strežnik smo izbrali Nginx. Je odprtokodni, zastonjski, visoko zmogljiv HTTP strežnik in povratni proxy strežnik. Podpira HTTPS, SMTP, POP3 in IMAP protokole [47]. Za komunikacijo med spletnim strežnikom in spletno aplikacijo pa smo uporabili vmesnik uWSGI.

4.3.5 Orodje virtualenv

V začetni fazi razvoja našega projekta je zelo priporočljiva namestitev virtualenv orodja. Z njegovo namestitvijo se izognemo morebitnim težavam z nameščanjem različnih verzij paketov, saj z njim lahko ustvarimo izolirana okolja. Problemi lahko nastanejo, ko imamo na našem razvojnem okolju več aplikacij, ki uporabljajo isto knjižnico. Kasneje, ko želimo knjižnico nadgraditi na novejšo verzijo, se kakšna od funkcionalnosti knjižnice prejšnje verzije izbriše ali spremeni. To nam lahko povzroči napake v delovanju naših aplikacij in ravno z virtualenv si zagotovimo neodvisnost med verzijami [27].

4.3.6 Podatkovna baza SQLite

SQLite je relacijska podatkovna baza, primerna za manjše aplikacije, za katero ne potrebujemo strežnika ali namestitvenih datotek. Za branje in pisanje se uporablja datoteka, kar pomeni, da je celotna podatkovna baza zapisana v eni sami datoteki. Omogoča izjemno hitro in enostavno prenašanje baze med aplikacijami in strežniki.

4.3.7 HTML in CSS

HTML je označevalni jezik za izdelavo spletnih strani. Zanj so značilne značke (ang. tags) in vsebina med značkami, s katerimi povemo, kako naj se stran prikazuje. Za pisanje HTML dokumentov potrebujemo urejevalnik besedila, kot je npr. beležnica. Značke lahko pišemo z velikimi in malimi črkami. Začetna značka se zapiše z `<IME>` in konča s končno `</IME>`. HTML tako definira strukturo spletne strani. V HTML dokumente lahko vstavimo CSS, s katerim lahko spletne strani še lepše predstavimo z različnimi pravili. Določimo lahko različne lastnosti spletne strani, kot so barva, odmiki, velikost, obrobe, pozicije in vrsto drugih atributov. Z unikatnimi identifikatorji, selektorji in razredi lahko razločimo strukturo strani od njene predstavitve.

4.3.8 Razvojno okolje Pycharm

Pycharm je napredno razvijalsko okolje, specializirano za programerski jezik Python. Izbiramo lahko med plačljivo (Professional) in brezplačno (Community) različico. Razvilo ga je podjetje JetBrains leta 2010 [16]. Ponuja dobro podporo že obstoječih ogrodij, kot so Django [20] ali Flask, ki delujejo v Windows [26], Linux [29] in MacOS [24] okolju. Vsebuje številne funkcionalnosti. Te so številčenje vrstic, kar nam pomaga ugotoviti, v kateri vrstici in stolpcu se nahajamo, samodejno zamikanje kode za lažjo preglednost, omogoča hitro iskanje, označevanje napak, razhroščevanje za uspešnost pri odkrivanju napak, testiranje po sklopih ter pregledno projektno drevo. Je napredno razvojno okolje z zelo pametnim samodejnim dokončevanjem sintakse, kar nam zelo olajša naše delo na projektu [42].

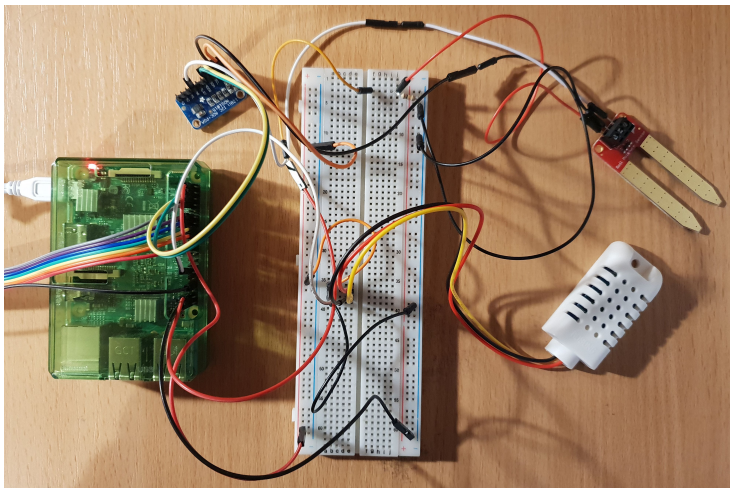
Poglavje 5

Implementacija

5.1 Vezava

Postopek izdelave našega prototipa za pametni terarij smo pričeli z vezavo senzorja za temperaturo in vlago AM2302 s tremi priključki. Za digitalno komunikacijo uporabimo pin številka 29, ki je eden izmed digitalnih vhodov na RPi3. Ostala dva priključka sta namenjena napajanju in ozemljitvi.

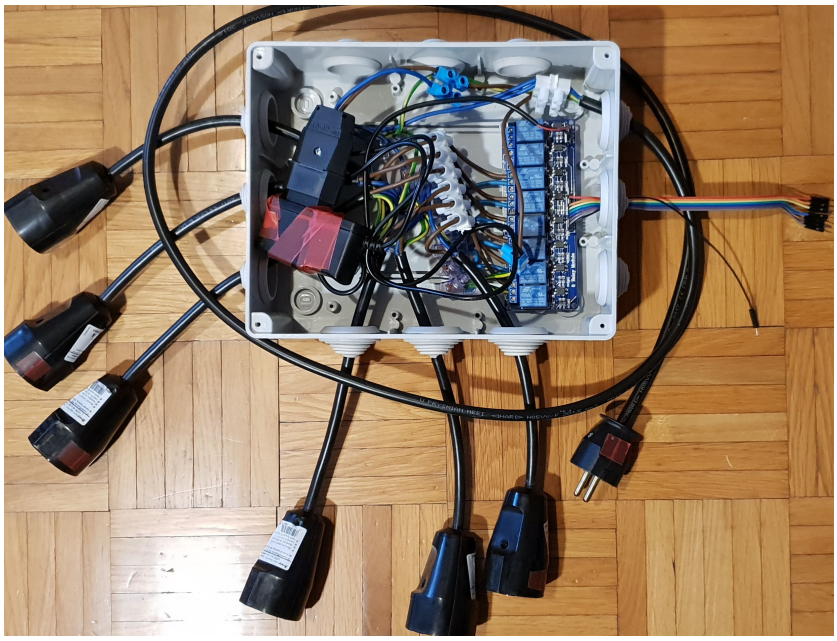
Priključka SCL in SDA analogno-digitalnega pretvornika smo povezali s pinoma 3 in 5 na našem računalniku RPi3.



Slika 5.1: Raspberry Pi 3 Model B s testno ploščo in senzorji.

Naslednja komponenta je analogni senzor za merjenje vlage v zemlji. Ta nam poda vrednost v voltih, s katero lahko določimo bolj natančne meritve vlažnosti. Zato smo potrebovali analogno-digitalni pretvornik (v nadaljevanju AD pretvornik), saj RPi3 nima analognih vhodov in izhodov. Senzor za merjenje vlage v zemlji ima 3 pine. Pin z oznako SIG smo povezali s pinom A0 na AD pretvorniku.

Vse senzorje, razen senzorja za merjenje vlage v zemlji, smo povezali skupaj na isti vir napajanja in ozemljitve na računalnik RPi3. Napajanje senzorja za vlago v zemlji smo priklopili na pin 31 in tako omogočili prižig senzorja samo pred izvedbo branja z njega. Senzor bi sicer bil pod stalno napetostjo, kar bi povzročilo hitrejšo oksidacijo merilnih sond.



Slika 5.2: Relejni modul z vtičnicami.

Sledil je priklop relejnega modula (Slika 5.2). Pri vezavi bi lahko uporabili napajanje z računalnika RPi3, vendar smo želeli zaradi varnosti napajanje ločiti. Tako smo uporabili 5 V napajalni modul in ga zvezali s pinom JD-VCC ter GND. Relejni modul ima 8 priključkov za vsak rele, vendar smo jih

trenutno potrebovali samo 6. Te smo priključili s pini 7, 11, 13, 15, 16 in 18 na računalniku RPi3. Na vsak rele smo vzporedno vezali svojo vtičnico za priklop zunanjih naprav. Vse komponente smo zaradi boljše preglednosti povezali s pomočjo testne ploščice (ang. breadboard) (Slika 5.1).

5.2 Podatkovna baza

Za podatkovno bazo smo izbrali SQLite. Najprej smo razmislili, kakšno podatkovno bazo sploh potrebujemo. Glede na naše podatke, ki jih bomo morali shranjevati, smo se odločili za SQLite. Podprta je s Python 2.7, zato jo je enostavno uporabiti na računalniku RPi3. Ko smo se povezali na naš RPi3, smo z ukazom `sudo apt-get install sqlite3` namestili našo podatkovno bazo. Ustvarili smo datoteko z imenom `app.db`, katere lokacija se nahaja znotraj našega projekta.

Naše podatke predstavljajo vrednosti, pridobljene iz senzorjev, katere je potrebno shranjevati za kasnejšo uporabo. Razvojno okolje Pycharm ponuja lažje urejanje podatkovne baze, zato smo se odločili z njegovo pomočjo ustvariti tabelo, v kateri imamo shranjene podatke. Osnovne meritve bodo tako shranjene v tabeli z imenom *Readings*. Vsaka meritev vsebuje število ID, *sensorID* za lažje medsebojno prepoznavanje senzorjev, časovno značko *datetime* ter vrednosti za vsako meritev posebej, in sicer *temperature* za temperaturo, *humidity* za vlažnost v zraku in *moisture* za vlago v zemlji.

5.3 Razvojno okolje

Ker RPi3 nima integriranega spominskega medija, smo za namestitev sistema uporabili kartico Micro SD. S spleta smo prenesli sliko operacijskega sistema Raspbian in ga naložili na spominsko kartico. Sledila je nastavitev povezave z brezžičnim omrežjem. Na spominski kartici operacijskega sistemu Raspbian smo morali v datoteki `wpa_supplicant.conf`, ki se nahaja v direktoriju `/etc/wpa_supplicant`, dodati ime in geslo našega brezžičnega

omrežja. V datoteki `/etc/network/interfaces` pa smo določili tudi statični IP naslov. Prav tako smo dodali datoteko `.ssh`, ki omogoča SSH povezavo ob prvem zagonu. Po prvem zagonu smo nastavili operacijski sistem, ga posodobili z ukazoma `sudo apt-get update` in `sudo apt-get upgrade` in se lotili priprave razvojnega okolja.

Ker smo si želeli, da bi razvoj potekal enostavno, hitro in pregledno, smo izbrali razvojno okolje PyCharm. Tega smo naložili na svojem računalniku z operacijskim sistemom Windows, kjer bomo tudi pisali našo programsko kodo, ta pa se bo izvajala na oddaljenem Python Interpreterju [38] na računalniku RPi3. Nato smo v nastavitvah razvojnega okolja Pycharm uredili nastavitve za daljavo in dodali naš RPi3 kot oddaljeni strežnik. Vnesli smo njegov statični IP naslov, geslo in uporabniško ime za dostop na daljavo. Komunikacija med samim razvojnim okoljem in Python Interpreterjem tako poteka preko SSH povezave. Prav tako smo omogočili avtomatski prenos izvorne kode, ki jo prenese na RPi3 preko SFTP povezave. To se zgodi ob vsaki shranitvi datoteke, v kateri pišemo našo spletno aplikacijo. Pri razvoju smo Pycharm razvojno okolje povezali tudi s **git** – sistemom za nadzor verzij (VCS –Version Control System) [45]. S tem smo si zagotovili dodatno varnost v primeru lokalne izgube podatkov ter lažje nadaljevanje dela, saj smo diplomsko nalogo razvijali na različnih lokacijah [7].

Ustvarili smo direktorij, v katerem bomo implementirali naš sistem. Dostopen je na poti `/home/pi/Projekt/Sistem_za_nadzor`. Znotraj omenjenega direktorija smo se lotili namestitve **virtualenv** ogrodja. Z ukazom `virtualenv venv` ustvarimo Python virtualno okolje z imenom `venv`. Tega moramo pred uporabo aktivirati z ukazom `. venv/bin/activate`. Sedaj imamo naše okolje pripravljeno za uporabo.

5.4 Zaledni sistem

Po postavitvi podatkovne baze in razvojnega okolja smo nadaljevali s postavitvijo zalednega sistema na računalniku RPi3. Pričeli smo z namestitvijo in uvozom vseh knjižnic ter orodij, potrebnih za delovanje sistema.

Za začetek smo namestili naš spletni strežnik Nginx z ukazom `sudo apt-get install nginx`, katerega je bilo treba nastaviti, da vse prejete zahtevke posreduje vmesniku uWSGI.

Sledila je še namestitev vmesnika uWSGI. To naredimo z ukazom `pip install uwsgi` [30]. Ustvarili smo datoteko `uwsgi.ini`, v kateri bomo imeli zapisane nastavitve.

Nato smo nadaljevali z namestitvijo Python ogrodja Flask. Pomembno je, da pred namestitvijo Flask ogrodja aktiviramo naše **virtualenv** okolje [27], z ukazom `pip install flask` pa ga namestimo. Nato smo ustvarili datoteko z imenom `app.py`, v kateri bo zapisana koda naše aplikacije, ter uredili avtomatski zagon naše Flask aplikacije. V datoteki z imenom `app.service`, ki se nahaja v `/etc/systemd/system` direktoriju, smo dodali podatke o delovnem direktoriju, informacije o tem, kdaj naj se naša storitev zažene, ter nastavili tudi pot do našega virtualnega okolja ter konfiguracijske datoteke uWSGI. Z urejenimi nastavitvami smo datoteko shranili in omogočili avtomatski zagon z ukazoma `sudo systemctl enable app` in `sudo systemctl start app`.

S tem smo dokončali naše nastavitve za serviranje naše spletne aplikacije znotraj virtualnega Python okolja na operacijskem sistemu Raspbian na računalniku RPi3.

5.5 Spletna aplikacija

Po končani namestitvi vseh orodij, knjižnic ter konfiguraciji datotek smo se lotili programiranja. V tem poglavju bomo tako razložili implementacijo našega sistema za pametni terarij. Spletna aplikacija bo namenjena nadzoru nad podatki, kot so temperatura, vlaga in vlažnost zemlje, ter preostalimi komponentami, ki so del pametnega terarija in skrbijo za njegove pogoje.

Programiranje smo razdelili v več manjših sklopov. Ustvarili smo več funkcij, zato bomo opisali vsako posebej.

5.5.1 Globalne spremenljivke

Za začetek smo definirali vse globalne spremenljivke, katere bomo uporabljali čez celoten program. Sem spadajo vse nastavitve glede priklopljenih senzorjev ter informacije, na katerih pinih na računalniku RPi3 se nahajajo. Definirali smo tudi slovar z imenom `settings_dic` z nastavitvami naših priklopljenih komponent. Ker razvijamo prototip, smo se odločili, da zadostuje, če izberemo vnaprej določene nastavitve lokacij pinov, senzorjev ter v kakšnem stanju so priklopljene komponente – ali so prižgane ali ne. Vrednosti slovarja so številke, in sicer "0"pomeni, da je naprava ugasnjena, "1"pa prižgana. V slovarju shranjujemo tudi vrednosti, nad katerimi bo pametni terarij vzdrževal, če je vključen avtomatski nadzor. To so temperatura, vlažnost zemlje in vlažnost zraka, čas za priklop in izklop luči.

5.5.2 Branje senzorjev

Ko smo imeli nastavljene globalne spremenljivke in parametre, smo se lotili branja senzorjev in prikazovanja podatkov. Ustvarili smo funkcijo z imenom `read_sensor` (Slika 5.3). Za to smo uporabili knjižnici `Adafruit_Python_ADS-1x15` [17] in `Adafruit_Python_DHT` [18], da pridobimo podatke iz senzorjev in jih nato ustrezno shranimo v spremenljivko. Prva knjižnica je namenjena branju podatkov iz našega senzorja za vlažnost zemlje preko AD pretvornika, druga pa za AM2302 senzor za branje temperature in vlažnosti zraka. Vse prebrane parametre v funkciji vrnemo kot število, katero prej primerno zaokrožimo na dve decimalki. Kot vhod v funkcijo imamo senzor, katerega želimo prebrati, in pin, na katerem je senzor priklopljen. Urediti je bilo potrebno tudi vrednost vlažnosti zemlje. Kot izhod AD pretvornik vrača določeno število. Senzor je bilo potrebno preveriti, kdaj kaže največjo vlažnost. Vzeli smo kozarec vode in vanj potopili senzor. Prebrana vrednost

je takrat kazala blizu števila 1260, ko pa smo ga izpostavili samo zraku, pa je bila vrednost blizu 0. Pri tem so možna manjša odstopanja zaradi nenatančnosti senzorja, zato smo želeli vračati približno vrednost v odstotkih. To pomeni, da smo za 100-odstotno vlažnost določili vrednost blizu števila 1260, blizu 0 pa 0-odstotno vrednost vlažnosti.

```
def read_sensor(sensor, pin):
    if sensor == Adafruit_DHT.AM2302:
        humidity, temperature = Adafruit_DHT.read_retry(sensor, pin)
        humidity = float(format(humidity, '.2f'))
        temperature = float(format(temperature, '.2f'))
        return humidity, temperature
    if type(sensor) is Adafruit_ADS1x15.ADS1015:
        pin_on(moisture_channel)
        time.sleep(0.1)
        moisture = sensor.read_adc(adc_channel, pin)
        time.sleep(0.1)
        pin_off(moisture_channel)
        return moisture if moisture > 0 else 0
```

Slika 5.3: Funkcija za branje senzorjev.

5.5.3 Nadzor nad relejnim modulom

Sedaj, ko smo imeli urejeno vse v povezavi s senzorji, smo se lotili programiranja nadzora nad relejnim modulom. Ustvarili smo dve funkciji `pin_on` ter `pin_off`. Znotraj vsake smo uporabili `Rpi.GPIO` knjižnico za upravljanje s pini, na katere je priključen relejni modul. Z ukazom `GPIO.output(channel, GPIO.HIGH)` pin in posledično relejni modul prižgemo, z `GPIO.output(channel,GPIO.LOW)` pa ugasnemo. Pred tem moramo določiti po katerem načinu bomo številčili pine na računalniku RPi3, kar smo storili z ukazom `GPIO.setmode(GPIO.BCM)`. Potrebno pa je bilo še določiti, ali so naši pini vhod ali izhod. Ker podatke pridobivamo iz senzorjev, smo nastavili uporabljene pine na izhod z ukazom `GPIO.setup(channel,GPIO.OUT)`. Kot vidimo v programski kodi (Slika 5.3), smo pred branjem senzorja za vlažnost zemlje najprej pin prižgali in nato po branju ugasnili. S tem smo si zagotovili bolj nadzorovano napajanje senzorja, s čimer manj obrabljamo senzorske diode, ki se ob

branjih hitro porabljajo.

5.5.4 Nastavitve in prikaz trenutnih meritev

Sledil je glavni del naše aplikacije. Kot smo omenili v prejšnjem poglavju smo za našo aplikacijo ustvarili datoteko `app.py`, katero smo razdelili na dva Flask pogleda. Prvi pogled smo definirali z dekoratorjem `app.route('/live_conditions')` do katerega dostopamo z dopisom `/live_conditions` k domeni v spletnem naslovu.

Rezultat pogleda je HTML predloga z imenom `live.html`, ki jo uporabljamo za prikaz trenutnih pogojev ter nastavitev pametnega terarija. S pomočjo Flask ogrodja smo uporabili funkcije `render_template`, s katero prikažemo HTML predlogo, prebrane parametre iz senzorjev pa shranimo v argumente in jih prikažemo s pomočjo Jinja2 ukazov. Strani so dostopne samo prijavljenim uporabnikom. Primer klica funkcije `render_template` za prikaz predloge `live.html` lahko vidimo na sliki 5.4.

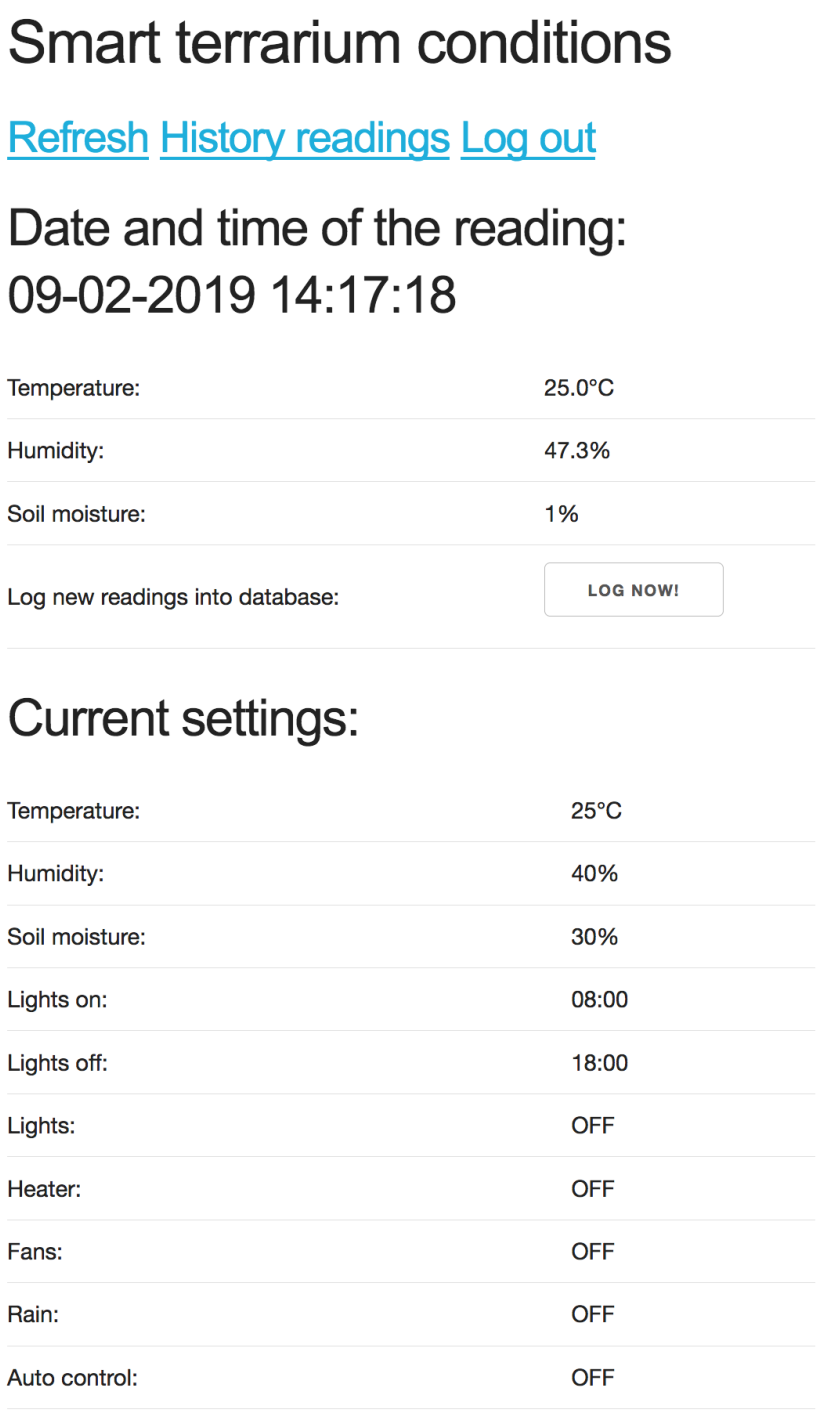
```
return render_template("live.html", temp=temperature, \
            hum=humidity, moisture=moisture, \
            read_time=read_time, settings=settings_new)
```

Slika 5.4: Primer klica Flask funkcije za prikaz HTML predloge.

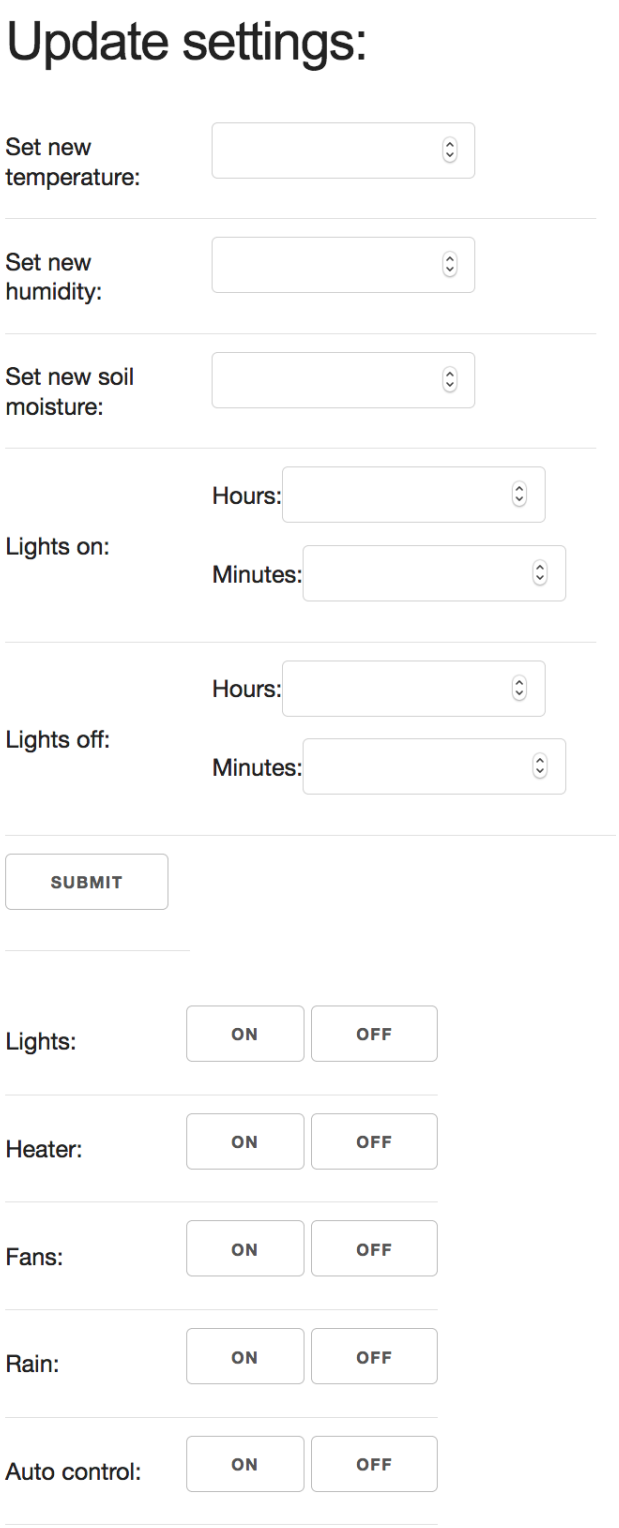
Stran prikazuje trenutne nastavitve priklopljenih komponent, ali so te prižgane ali ne, izpisane pa so tudi trenutno nastavljene vrednosti za temperaturo, vlažnost zraka ter vlažnost zemlje (Slika 5.5). Za spreminjanje nastavitev smo dodali vnosna polja za določanje novih vrednosti ter gumbe za prižig ali izklop luči, ventilatorjev, dežja, toplotnega grelca in način avtomatskega nadzora (Slika 5.6). Vse naprave lahko upravljamo posamezno ali pa jih nadzoruje avtomatski nadzor. Celotno stran smo zgradili s tabelami in obrazci. S klikom na potrditveni gumb se nove vrednosti vnosnih polj posredujejo na strežnik. Vnosna polja je potrebno pred vsakim pošiljanjem pregledati, zato smo v programski kodi opravili validacijo. Z njo preverimo, ali so polja pozitivna števila ter če ustrezajo območju znotraj dovoljene meje

za temperaturo, vlažnosti zraka in zemlje pametnega terarija, katere smo določili že na začetku programa z globalnimi spremenljivkami. Če je validacija uspela, se nove vrednosti posodobijo in shranijo, sicer se vrednosti ne uveljavijo in na spletni strani se prikaže sporočilo za neveljavnost spremembe.

Podobno je z gumbi za vklop ali izklop naprav. Pred vsako spremembo se preveri, v kakšnem stanju je naprava in če se stanje razlikuje od želene spremembe, se nato preveri, ali je ta dovoljena. V primeru previsoke temperature znotraj terarija onemogočimo spremembo za vklop toplotnega grelca, saj bi s tem temperaturo terarija še povečali. Podobno se zgodi v primeru prenizke temperature, ki onemogoči izklop grelca. Na enak način smo preverjali tudi vklop pršenja vode. Če je vlažnost zemlje že nad 90 %, ne moremo vklopiti te možnosti, saj bi s tem lahko terarij preveč namočili. Trajanje pršenja smo za dodatno varnost sistema omejili na določen časovni interval, da se izognemo poplavi terarija. Vlažnost v zraku smo zavarovali z vklopom in izklopom ventilatorjev, in sicer je v primeru prenizke vlažnosti zraka njihov vklop onemogočen ali obratno. To smo implementirali, da ne bi prišlo do primerov, ko bi lahko pogoje v terariju po pomoti poslabšali in s tem ogrozili organizme v njem. Za dodatno varnost smo dodali funkcijo z imenom `check_conditions` za preverjanje vrednosti pogojev v terariju z najvišjimi dovoljenimi vrednostmi. Preverjanje smo morali opravljati na vsakih 5 minut, zato smo ga implementirali z nitenjem. Zaradi stalnega preverjanja pogojev in branja senzorjev smo uporabili modul `threading`. S tem smo ločili preverjanje pogojev in stanja naprav od glavnega programa, da ne bi prišlo do zaustavitve programa. Preverjanje je delovalo v treh delih. Prvi del je skrbel za ugašanje toplotnega grelca ob previsoki temperaturi v terariju, drugi za prižiganje ventilatorjev ob previsoki vlažnosti zraka ter zadnji za izklapljanje pršenja vode ob previsoki vlažnosti zemlje.



Slika 5.5: Zaslonska slika spletne strani s trenutnimi meritvami in nastavitvami pametnega terarija.



Slika 5.6: Zaslonska slika spletne strani za spremembo nastavitev pametnega terarija.

5.5.5 Avtomatski nadzor

Sledila je implementacija avtomatskega nadzora nad pametnim terarijem. Ob vklopu le-tega se sproži nadzor nad vsemi parametri brez ročnega preverjanja in preklapljanja zunanjih naprav. Zažene se funkcija z imenom `auto_control`, ki izvaja branje senzorjev na vsakih pet minut v zanki v posebni niti. Kot vhod v funkcijo imamo globalno spremenljivko slovar `settings`. Ustvarili pa smo tudi še dodatno globalno spremenljivko z imenom `running`, s katero ob pritisku na gumb za izklop avtomatskega nadzora spremenimo stanje na `False`, kar povzroči zaustavitev delovanja naše niti. Avtomatski nadzor ob vsakem branju senzorjev preverja pogoje v terariju, prebrane vrednosti se zapišejo v bazo in na podlagi teh se sprožijo določeni vklopi in izklopi naprav, hkrati pa se posodobijo tudi podatki na spletni strani za pregled nad delovanjem sistema. Prebrane vrednosti se shranijo v trenutne spremenljivke, ki se med delovanjem primerjajo z nastavljenimi globalnimi spremenljivkami in se šele ob koncu potrdijo in prepišejo v globalne.

Prva stvar, ki jo avtomatski nadzor spremlja, je temperatura. Preverjanje je narejeno tako, da se ob previsoki temperaturi toplotni grelec izklopi, za nekaj sekund pa se vklopijo ventilatorji. Ob prenizki temperaturi pa je situacija obratna. Vklop ventilatorjev za nekaj sekund omogoči boljše kroženje zraka v terariju, saj s tem pridobimo bolj realno temperaturo ozračja. Pri tem smo dovolili odstopanje temperature za največ 3 °C, saj se je ob testiranju delovanja to zdela najbolj primerna vrednost za interval preverjanja na vsakih 5 minut. Programsko kodo za preverjanje temperature z avtomatskim nadzorom lahko vidimo na Sliki 5.7.

Na podoben način kot pri temperaturi smo preverjali tudi vlažnost zraka in zemlje. Pri vlažnosti zraka smo dovolili odstopanje za 5 %. Vklop ventilatorjev se zgodi v primeru, ko je trenutna vlažnost v zraku večja od nastavljene, izklop pa, ko pade vlažnost pod najnižjo dovoljeno. Ob zelo nizki vlažnosti v zraku se vklopi pršenje vode, a samo v primeru padca vrednost za 40 % ali več, pod pogojem, da je tudi vlažnost zemlje pod 30 %. S tem se izognemo prekomernemu zalivanju terarija ob vsaki minimalni spremembi

meritev.

```
def auto_control(settings):
  temperature = settings["temperature"]
  while settings["auto_control"]:
      # branje senzorjev
      humidity_current, temperature_current = \
         read_sensor(temperature_sensor, pin)
      # trenutna temperatura je vecja od nastavljene
      if temperature_current > temperature + 3:
         pin_off(heater_channel)
         pin_on(fan_channel)
         time.sleep(5)
         pin_off(fan_channel)
         settings["heater"] = 0
      # trenutna temperatura je manjsa od nastavljene
      elif temperature_current < temperature - 3:
         settings["heater"] = 1
         pin_on(heater_channel)
         if(settings["fans"] == 1):
            pin_off(fan_channel)
            settings["fans"] = 0
   time.sleep(5)
   return settings
```

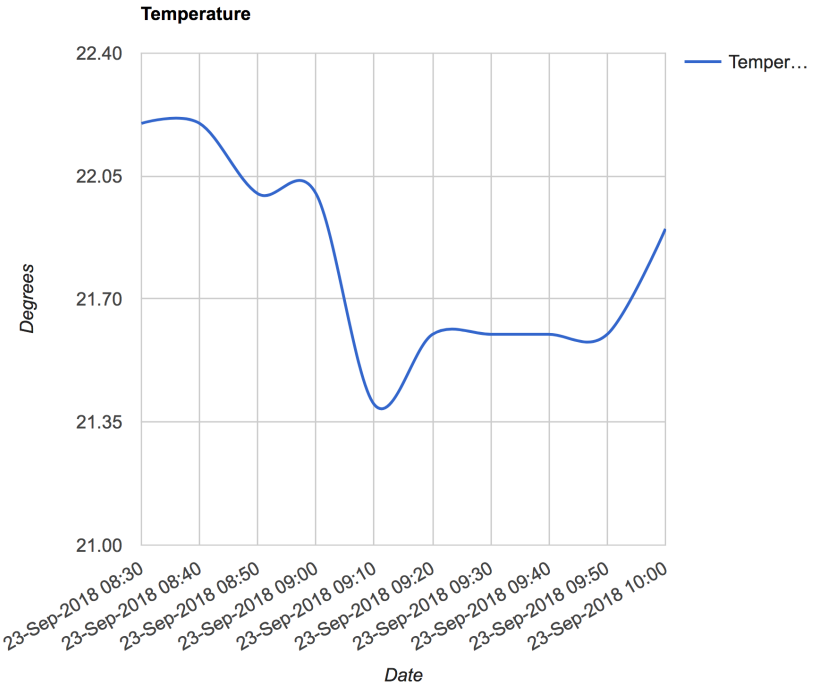
Slika 5.7: Programska koda za avtomatski nadzor temperature.

Avtomatski nadzor preverja tudi vklop in izklop luči. Te se vklopijo in izklopijo dvakrat na dan ob določenih urah, ki smo jih nastavili na začetku programa. Ker smo vnesli dve uri, eno za vklop ter drugo za izklop, jih je potrebno preverjati. Trenutni čas se ves čas primerja s shranjenimi. Dokler se vrednosti ujemajo, se izvede operacija izklopa ali vklopa luči.

5.5.6 Zgodovina meritev pametnega terarija

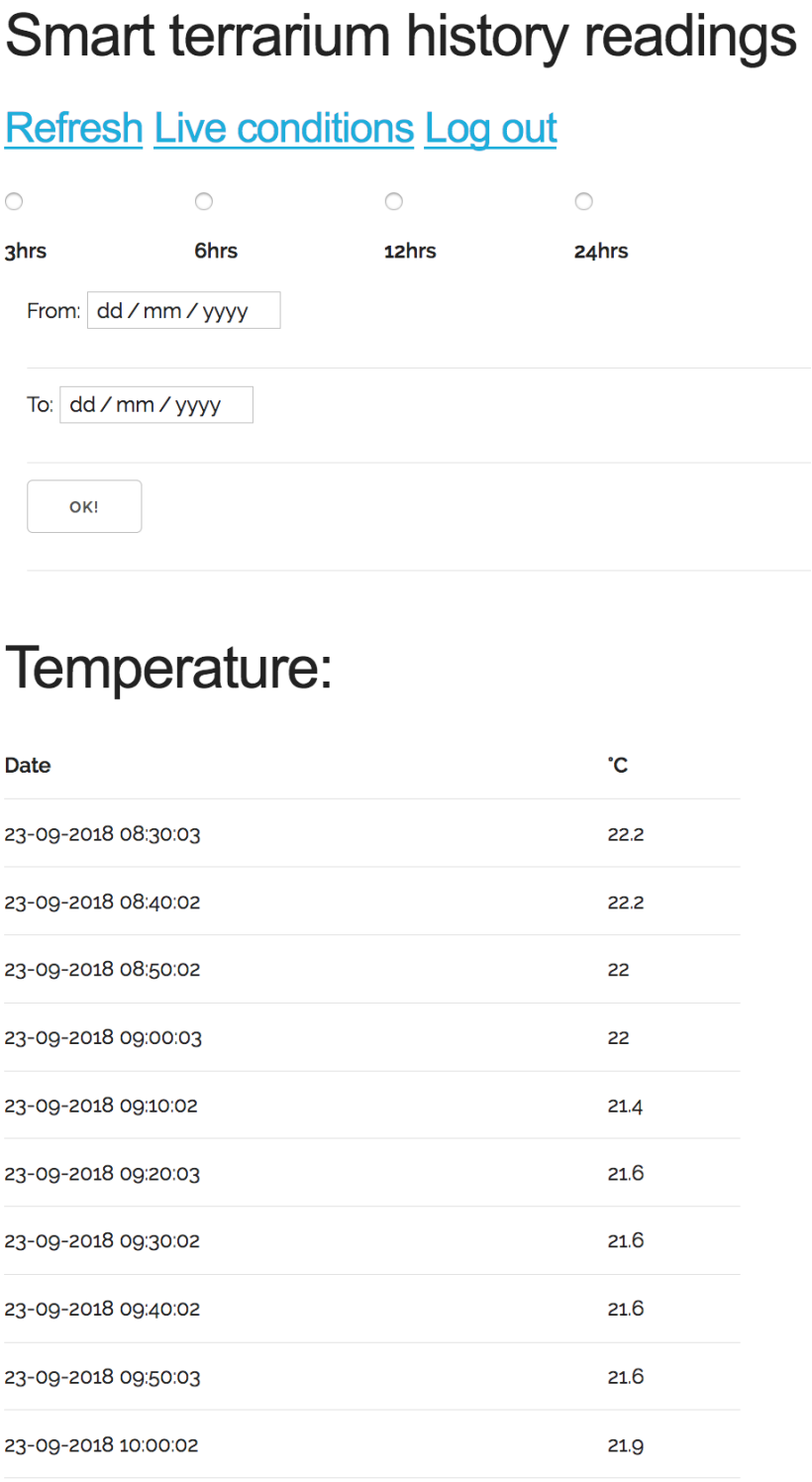
Implementacijo spletne aplikacije smo nadaljevali z drugim Flask pogledom. Definirali smo ga z dekoraterjem `app.route('/history_readings')`. Pri tem smo ustvarili funkcijo z imenom `history_read()`, katere glavna naloga je pridobitev podatkov iz podatkovne baze in ustrezen prikaz podatkov na spletni strani. Rezultat pogleda je torej HTML predloga z imenom `history.html`, ki prikazuje zgodovino meritev pametnega terarija. Prikazanih je zadnjih deset zabeleženih meritev iz podatkovne baze, podatki pa

so razporejeni po tabelah za vsako meritev posebej. Poleg tabel so podatki predstavljeni tudi v obliki grafov, ki so narejeni s pomočjo Google Charts spletnega ogrodja (Slika 5.8).



Slika 5.8: Graf za prikaz zgodovine meritev temperature.

Spletna stran ponuja tudi prikaz meritev v določenem časovnem obdobju, ki ga lahko izberemo sami. Za izbiro datuma smo dodali vnosna polja (Slika 5.9). S klikom na gumb za potrditev se vrednosti vnosnih polj posredujejo naprej za nadaljnjo obdelavo. Novi podatki se shranijo v začasne spremenljivke, ki se uporabijo pri SQL poizvedbi nad podatkovno bazo [49]. Če meritev v zahtevanem obdobju ni, se izpiše sporočilo za neuspešno poizvedbo. Meritve lahko pridobimo tudi s pomočjo vnosa parametrov v spletni naslov, vendar moramo pri tem paziti na pravilen zapis. Pri vpisu datuma in ure se podatki preverijo in šele ob pravilni obliki se zahteva izvede. Zgodovino meritev lahko omejimo tudi z izbirnimi tipkami. Odvisno od izbrane tipke se tudi pri tem ob kliku na eno izmed njih sproži funkcija `history_read()`, ki prikaže zahtevane meritve v določenem časovnem okviru.



Slika 5.9: Nastavitve za prikaz zgodovine meritev in tabela zgodovine meritev temperature.

Poglavje 6

Primerjava in možne nadgradnje pametnega terarija

V tem poglavju se bomo osredotočili na primerjavo naše rešitve z že obstoječo rešitvijo pametnega terarija Biopod. Pogledali bomo, kako se med seboj cenovno razlikujeta, katere funkcionalnosti imata ter kakšne so prednosti in slabosti obeh rešitev. Omenili bomo tudi primere uporabe naše rešitve ter možne nadgradnje.

Pametni terarij Biopod se prodaja od leta 2018 in je eden izmed novejših produktov, ki jih lahko trenutno kupimo. Za najmanjšo velikost terarija z že vgrajenim pametnim sistemom za nadzor moramo odšteti približno 520 EUR. Steklen terarij, ki je vizualno lepo izdelan, ima vgrajene toplotne grelce, pršilnik vode, ventilacijo, luči, sistem za gnojenje ter kamero. Vse komponente lahko nadzorujemo preko mobilne aplikacije. Omogoča nadzor nad temperaturo, vlažnostjo zraka in zemlje, svetlobo, zračnim pretokom ter pršenjem vode. Dodatna funkcionalnost njihovega izdelka je gnojenje zemlje, kar lahko kasneje tudi sami vključimo v nadgradnjo našega prototipa.

Kot lahko vidimo v Tabeli 6.1 so stroški za nakup komponent pri našem izdelku znašali približno 100 EUR. Če prištejemo še ceno steklenega terarija, luči, vodne črpalke, ventilatorjev in talnega grelca, katere smo imeli na zalogi že od prej, pa vse skupaj znaša približno 170 EUR. Za izdelavo končnega

Oprema	**Cena**
Raspberry Pi 3 Model B, spominska kartica MicroSD, ohišje ter napajalnik	50 EUR
Adafruit 4-kanalni analogno-digitalni pretvornik ADS1015	10 EUR
Senzor za merjenje vlažnosti v zemlji SEN-13637	6 EUR
Senzor temperature in vlažnosti AM2302	15 EUR
Osemkanalni relejni modul SRD-05VDC-SL-C	5 EUR
Električna napeljava, ohišje, vtičnice	10 EUR
Testna ploščica ter priključni kabli	5 EUR
Steklen terarij	30 EUR
Stojalo za luč in žarnica	20 EUR
Vodna črpalka in priključki	7 EUR
Toplotni grelec	10 EUR
Računalniški ventilatorji	5 EUR
Skupaj:	**173 EUR**

Tabela 6.1: Tabela kupljene opreme in okvirna cena.

izdelka smo morali vse skupaj dobro premisliti, ustrezno nastaviti ter rešitev sprogramirati, vendar je pri tem mogoča veliko večja svoboda izdelave. Vsako komponento lahko ob okvari lažje zamenjamo, saj so bolj dostopne in cenovno ugodnejše v primerjavi s pametnim terarijem Biopod.

Glede funkcionalnosti sta si izdelka podobna, vendar moramo upoštevati, da je naša rešitev še vedno v fazi razvoja. Trenutno je naš izdelek narejen za vzdrževanje vnaprej določenih pogojev za točno določen terarij. Z vgradnjo dodatnih senzorjev bi lahko možnost uporabe bistveno povečali in s tem zvišali natančnost merjenja pogojev v terariju. Kot je pri pametnem terariju Biopod vgrajenih več senzorjev za merjenje temperature v različnih višinskih nivojih, bi lahko na isti način storili tudi pri našem pametnem terariju. To bi omogočilo boljšo predstavo o trenutni temperaturi terarija na različnih mestih, s čimer bi izboljšali uravnavanje pogojev znotraj terarija. Dodatno bi

z namestitvijo kamere spremljali premike živali v terariju, kar bi pripomoglo k boljšemu razumevanju njihovega vedenja [3, 8]. Sistem bi lahko uporabljali za nadzor akvarijev ali pa za proizvodnjo določenih rastlin za pridelavo hrane. S pravilno nadgradnjo in implementacijo bi naš sistem vključili v večje projekte, kot je upravljanje toplotnih gred, vrtov ali pa za vzrejo večjega števila živali.

Poglavje 7

Sklepne ugotovitve

Pravilno vzdrževanje pogojev v terariju zahteva veliko našega časa in truda. Uravnavati je potrebno svetlobo, temperaturo, vlago v zraku in zemlji, pozorni pa moramo biti tudi na primeren zračni pretok v terariju. Pri ročnem nadzoru terarija so se pogoji v njem lahko hitro poslabšali in ušli iz nadzora, če nismo bili vedno prisotni. To je povzročilo večje stroške vzdrževanja ali pa celo ogrozilo zdravje organizmov v terariju. Trenutno dostopni izdelki pametnih terarijev predstavljajo precejšen strošek ali pa so še vedno v fazi razvoja, zato smo se odločili izdelati svoj sistem pametnega terarija za lažji nadzor nad pogoji.

V naši diplomski nalogi smo izdelali delujoč prototip cenovno ugodnejšega sistema pametnega terarija, katerega lahko upravljamo preko spletne aplikacije. Z različnimi senzorji lahko pogoje terarija nadzorujemo s pomočjo računalnika RPi3, ki deluje tudi kot spletni strežnik. Z uporabo ogrodja Flask v programskem jeziku Python smo izdelali zaledni sistem, ki se odziva na uporabniške akcije. Zunanje priklopljene naprave pa nadzorujemo preko relejnega modula, ki ga upravljamo s pomočjo računalnika RPi3. Implementirali smo sistem za avtomatski nadzor nad temperaturo, svetlobo, kroženjem zraka, vlago v zraku in zemlji. Omogočen imamo nadzor nad priklopljenimi napravami, kot so luči, toplotni grelec, vodna črpalka in ventilatorji. Uporabnik sistema ima popoln pregled nad trenutnimi meritvami temperature,

vlage v zraku in zemlji znotraj terarija. Pogleda lahko trenutno nastavljene nastavitve pametnega terarija in stanje priklopljenih naprav, katere lahko tudi posamezno vklopi ali izklopi ali pa to prepusti avtomatskemu nadzoru. Podatke preteklih meritev si lahko ogleda v točno določenem časovnem okviru in zabeleži nove meritve pogojev terarija.

Naša rešitev še ni končna, saj je še vedno v fazi razvoja. Ugotovili smo, da v nekaterih primerih nakupa cenejših izdelkov hitreje pride do okvare, zato ni vedno dobro kupovati najcenejše komponente. To se je zgodilo pri senzorju za merjenje vlage v zemlji. S cenejšim senzorjem so se merilne sonde pokvarile hitreje, kar je povzročilo napačne meritve, zato smo kupili dražji senzor z dodatno zaščito pred korozijo. Tudi regulacija pogojev naše rešitve še ni popolna in z dodatnim testiranjem bomo lahko v prihodnosti to še izboljšali.

Trenutno je naš izdelek narejen za vzdrževanje temperature ter vlage v zraku in zemlji. Z vgradnjo dodatnih senzorjev bi lahko možnost uporabe bistveno povečali in s tem zvišali natančnost merjenja pogojev v terariju. Z dodatnimi temperaturnimi senzorji bi pridobili natančnejše meritve temperature pametnega terarija. Enako bi to lahko storili s senzorji za merjenje vlage v zemlji in zraku.

V primerjavi s trenutno obstoječimi rešitvami pametnih terarijev je naša rešitev bolj prilagodljiva, saj lahko sistem vgradimo v kateri koli terarij in pri tem nismo omejeni na velikost terarija. Sami se lahko odločimo, katere zunanje naprave bomo nadzorovali z računalnikom RPi3, saj jih lahko priključimo na relejni modul preko posameznih vtičnic. Ob okvari katere koli zunanje naprave imamo možnost hitre zamenjave, saj nam ni potrebno razdirati celotnega pametnega terarija, kot bi to morali pri drugih rešitvah pametih terarijev, ki imajo sistem že vgrajen.

Z diplomsko nalogo smo zadovoljni, saj smo uspeli izdelati cenovno ugodnejši sistem za nadzor pogojev v terariju.

Literatura

[1] T. Brant. Raspberry pi 3 model b+: A delightful next step. *PC Magazine*, pages 70–76, 2018.

[2] E. Bruins. *The Complete Encyclopedia of Terrarium.* Complete Encyclopedia Series. Grange, 2001.

[3] T. Fei, A. K. Skidmore, V. Venus, T. Wang, B. Toxopeus, M. Bian, and Y. Liu. Predicting micro thermal habitat of lizards in a dynamic thermal environment. *Ecological Modelling*, 231:126–133, 2012.

[4] Y. Li, Y. Guo, and S. Chen. A survey on the development and challenges of the internet of things (iot) in china. In *2018 International Symposium in Sensing and Instrumentation in IoT Era (ISSI)*, pages 1–5, 2018.

[5] M. Maksimović, V. Vujović, N. Davidović, V. Milošević, and B. Perišić. Raspberry pi as internet of things hardware: performances and constraints. *design issues*, 3:8, 2014.

[6] A. Pajankar, A. Kakkar, M. Poole, and R. Grimmett. *Raspberry Pi: Amazing Projects From Scratch.* Packt Publishing, 2016.

[7] V. Sandeep, K. L. Gopal, S. Naveen, A. Amudhan, and L. S. Kumar. Globally accessible machine automation using raspberry pi based on internet of things. In *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015.

[8] P. Santos, J. Alves, and P. Carneiro. Monitoring and control of a reptile terrarium. In *2014 11th International Conference on Remote Engineering and Virtual Instrumentation (REV)*, pages 438–439, 2014.

[9] Analogno-digitalni pretvornik ADS1015. Dosegljivo: `https://www.modmypi.com/raspberry-pi/iousbanalogue-expansion-1028/adcs-and-analogue-expansion-1030/adafruit-4-channel-i2c-12-bit-analogue-to-digital-converter-adc`. [Dostopano: 3. 2. 2019].

[10] CSS – user guide. Dosegljivo: `https://developer.mozilla.org/en-US/docs/Learn/CSS`. [Dostopano: 4. 2. 2019].

[11] Exo Terra Compact Top. Dosegljivo: `http://www.exo-terra.com/en/products/compact_top.php`. [Dostopano: 4. 2. 2019].

[12] Exo Terra Heat Mat grelna podloga. Dosegljivo: `http://www.exo-terra.com/en/products/heat_mat.php`. [Dostopano: 4. 2. 2019].

[13] Exo Terra Natural Light žarnica. Dosegljivo: `http://www.exo-terra.com/en/products/natural_light.php`. [Dostopano: 6. 2. 2019].

[14] HTML – user guide. Dosegljivo: `https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML`. [Dostopano: 4. 2. 2019].

[15] IEEE_802.11. Dosegljivo: `https://en.wikipedia.org/wiki/IEEE_802.11`. [Dostopano: 6. 2. 2019].

[16] JetBrains. Dosegljivo: `https://www.jetbrains.com/products.html#`. [Dostopano: 4. 2. 2019].

[17] Knjižnica Adafruit_Python_ADS1x15. Dosegljivo: `https://github.com/adafruit/Adafruit_Python_ADS1x15`. [Dostopano: 10. 2. 2019].

[18] Knjižnica Adafruit_Python_DHT. Dosegljivo: `https://github.com/adafruit/Adafruit_Python_DHT`. [Dostopano: 10. 2. 2019].

[19] Knjižnica jQuery. Dosegljivo: https://api.jquery.com/. [Dostopano: 4. 2. 2019].

[20] Ogrodje Django. Dosegljivo: https://www.djangoproject.com/. [Dostopano: 4. 2. 2019].

[21] Ogrodje Flask. Dosegljivo: http://flask.pocoo.org/. [Dostopano: 23. 1. 2019].

[22] Ogrodje Google Charts. Dosegljivo: https://developers.google.com/chart/. [Dostopano: 4. 2. 2019].

[23] Operacijski sistem Debian. Dosegljivo: https://www.debian.org/. [Dostopano: 6. 2. 2019].

[24] Operacijski sistem MacOS. Dosegljivo: https://www.apple.com/lae/macos/mojave/. [Dostopano: 9. 2. 2019].

[25] Operacijski sistem Raspbian. Dosegljivo: https://www.raspberrypi.org/documentation/raspbian/. [Dostopano: 4. 2. 2019].

[26] Operacijski sistem Windows. Dosegljivo: https://www.microsoft.com/sl-si/windows. [Dostopano: 6. 2. 2019].

[27] Orodje virtualenv. Dosegljivo: https://virtualenv.pypa.io/en/latest/. [Dostopano: 4. 2. 2019].

[28] Osemkanalni 5 V relejni modul. Dosegljivo: http://wiki.sunfounder.cc/index.php?title=8_Channel_5V_Relay_Module. [Dostopano: 10. 2. 2019].

[29] P. Cobbaut. Linux Fundamentals. Dosegljivo: http://linux-training.be/linuxfun.pdf. [Dostopano: 10. 2. 2019].

[30] Pip (package manager). Dosegljivo: https://pip.pypa.io/en/stable/user_guide/. [Dostopano: 6. 2. 2019].

[31] Podatkovna baza SQLite. Dosegljivo: https://www.sqlite.org/docs.html. [Dostopano: 4. 2. 2019].

[32] Pogon Jinja2. Dosegljivo: http://jinja.pocoo.org/. [Dostopano: 4. 2. 2019].

[33] Produkt Biopod. Dosegljivo: https://www.biopod.com/. [Dostopano: 4. 2. 2019].

[34] Produkt OrchidBox. Dosegljivo: https://www.orchidbox.io/. [Dostopano: 4. 2. 2019].

[35] Produkt SmartTerra. Dosegljivo: https://www.kickstarter.com/projects/416958328/smartterra-a-smart-terrarium-with-customizable-env. [Dostopano: 4. 2. 2019].

[36] Programski jezik Python. Dosegljivo: https://www.python.org/about/. [Dostopano: 6. 2. 2019].

[37] Projekt Raspberry Pi Terrarium Controller. Dosegljivo: https://www.carnivorousplants.co.uk/resources/raspberry-pi-terrarium-controller/. [Dostopano: 4. 2. 2019].

[38] Python Interpreter. Dosegljivo: https://www.jetbrains.com/help/pycharm/configuring-python-interpreter.html. [Dostopano: 6. 2. 2019].

[39] Python za programerje. Dosegljivo: http://trac.lecad.si/vaje/raw-attachment/wiki/python/pythonzaprogramerje.pdf. [Dostopano: 4. 2. 2019].

[40] Raspberry Pi 3 Model B. Dosegljivo: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

[41] Raspberry Pi GPIO. Dosegljivo: https://www.raspberrypi.org/documentation/usage/gpio/README.md. [Dostopano: 6. 2. 2019].

[42] Razvojno okolje Pycharm. Dosegljivo: https://www.jetbrains.com/pycharm/. [Dostopano: 4. 2. 2019].

[43] Senzor za merjenje vlage v zemlji SEN-13637. Dosegljivo: https://www.sparkfun.com/products/13637. [Dostopano: 4. 2. 2019].

[44] Senzor za temperaturo in vlago AM2302. Dosegljivo: https://www.adafruit.com/product/393. [Dostopano: 4. 2. 2019].

[45] Sistem za nadzor verzij Git. Dosegljivo: https://git-scm.com/. [Dostopano: 6. 2. 2019].

[46] Specifikacije računalnika Raspberry Pi 3 Model B. Dosegljivo: https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/. [Dostopano: 10. 2. 2019].

[47] Specifikacije strežnika Nginx. Dosegljivo: https://nginx.org/en/docs/. [Dostopano: 4. 2. 2019].

[48] Spletni strežnik Nginx. Dosegljivo: https://www.nginx.com/resources/wiki/. [Dostopano: 4. 2. 2019].

[49] SQL vodič. Dosegljivo: https://www.w3schools.com/sql/. [Dostopano: 10. 2. 2019].

[50] Vmesnik uWSGI. Dosegljivo: https://uwsgi-docs.readthedocs.io/en/latest/. [Dostopano: 4. 2. 2019].

[51] Vodna črpalka Exo Terra Repti Flo 200. Dosegljivo: http://www.exo-terra.com/en/products/repti_flo_200.php. [Dostopano: 6 2. 2019].

[52] Zbirka WSGI pripomočkov Werkzeug. Dosegljivo: http://werkzeug.pocoo.org/docs/0.14/. [Dostopano: 4. 2. 2019].