Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matjaž Dolgan

# Vremenska postaja na platformi Raspberry Pi

diplomsko delo

visokošolski strokovni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Patricio Bulić

Ljubljana, 2013

*Besedilo je oblikovano z urejevalnikom besedil LaTeX.*

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

*Tržaška 25*
*1001 Ljubljana, Slovenija*
*telefon: 01 476 84 11*
*01 476 83 87*
*faks: 01 426 46 47*
*01 476 87 11*
*www.fri.uni-lj.si*
*e-mail: dekanat@fri.uni-lj.si*

Št. naloge: 00459 / 2013
Datum: 15.4.2013

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MATJAŽ DOLGAN**

Naslov: **VREMENSKA POSTAJA NA PLATFORMI RASPBERRY PI**

**RASPBERRY PI WEATHER STATION**

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Izdelajte vremensko postajo na platformi Raspberry Pi. Vremenska postaja naj omogoča beleženje temperature, vlage, zračnega pritiska ter koordinat GPS. V ta namen uporabite ustrezne senzorje, ki jih preko komunikacijskih adapterjev I2C, single-wire ter USART priključite na platformo Raspbery Pi. Izmerjene podatke beležite v podatkovni bazi SQLite. Vremenska postaja naj omogoča dostop do podatkov v bazi preko spletnega vmesnika.

Mentor:
izr. prof. dr. Patricio Bulić

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

# Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matjaž Dolgan, z vpisno številko **63060073**, sem avtor diplomskega dela z naslovom:

*Vremenska postaja na platformi Raspberry Pi*

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Particija Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 25. avgust 2013 Podpis avtorja:

*Rad bi se zahvalil mentorju izr. prof. dr. Patriciju Buliću za koristne nasvete in usmerjanje dela.*
*Zahvalil bi se rad tudi moji mami Ireni, sestri Tini, babici Danici in punci Urški za moralno podporo skozi celotno študijsko obdobje.*
*Posebna zahvala gre tudi mojemu bratrancu Črtu, ki mi je vedno pomagal in me spodbujal v kočljivih programerskih situacijah.*
*Najlepša hvala tudi sošolcem Primožu, Tomažu in Anji za družbo in popestritev študijskih let.*

# Kazalo

# Povzetek

Cilj diplomske naloge je bil narediti vremensko postajo na podlagi računalnika Raspberry Pi. V prvem delu diplomske naloge smo se osredotočili na predstavitev računalnika in opis uporabljenih tehnologij. Strojne tehnologije so nam pomagale pri meritvi temperature in zračnega tlaka (senzor MPL115A2), vlage (senzor DHT11) in koordinat GPS (sprejemnik MAX-6). Programski del tehnologij sestavljajo programski jezik Python in razne knjižnice: Crontab, SQLAlchemy, SQLite, CherryPi, Mako, nginx. Drugi del diplomske naloge pa nam predstavi samo implementacijo celotnega projekta. Vremensko postajo poganja lasten spletni strežnik, ki s pomočjo spletne aplikacije zajema podatke, jih shranjuje v podatkovno bazo in skrbi za prikaz. Uporaba vmesnika WSGI omogoča enostavno zamenjavo spletnega strežnika.

**Ključne besede:** Raspberry Pi, vremenska postaja, programski jezik Python, vmesnik WSGI, senzor MPL115A2, senzor DHT11, sprejemnik MAX-6

# Abstract

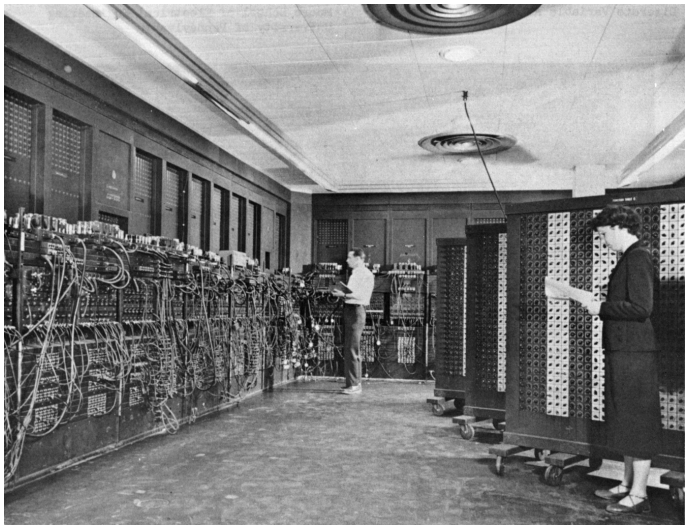
The goal of this project was to make a weather station on the computer Raspberry Pi. In the first part of the thesis we focused on the history of the computer's development and a presentation of technologies we used. Hardware technologies enabled us to measure air temperature and pressure (sensor MPL115A2), humidity (sensor DHT11) and coordinates GPS (receiver MAX-6). For the programming technologies we used Python and some other tools and libraries: Crontab, SQLAlchemy, SQLite, CherryPi, Mako, nginx. In the second part of the thesis we describe implementation of the whole concept. The weather station runs on its own web server and measures values from sensors and stores them into a database. With a simple web interface we displayed the measured results. WSGI enables us the web display and public access with computer or smart phone.

**Key words:** Raspberry Pi, weather station, Python programming language, interface WSGI, sensor MPL115A2, sensor DHT11, receiver MAX-6

# Poglavje 1

# Uvod

Računalništvo se razvija izjemno hitro. Če se ozremo v preteklost, se je prvi namenski računalnik pojavil šele leta 1946. Računalnik so poimenovali ENIAC 1.1. Njegov osnovni namen je bil računanje balističnih tabel za ameriško vojsko. Temeljil je na Turingovemu modelu stroja [1], cena je bila približno 4.5 mio €, velik je bil 167 $m^2$.



Slika 1.1: ENIAC - prvi računalniški sistem, ki je bil sposoben računanja in preprogramiranja za reševanje problemov. Vir: [2]

Že leta 1965 je soustanovitelj podjetja Intel Gordon E. Moore v članku [3] razmeroma točno napovedal hitrost razvoja računalniških sistemov. Po njegovi napovedi se bo število tranzistorjev na integriranih vezjih podvojilo vsaki dve leti. Podoben trend rasti je bilo zaznati tudi na drugih področjih računalništva (npr. kapaciteta trdega diska na enoto površine [4], število zaslonskih pik na dolar). Posledica takega razvoja je pojav,

ko danes v rokah prenašamo računalniške sisteme, ki so neprimerno zmogljivejši, cenejši od predhodnikov, ki so včasih tehtali nekaj 100 kg in zavzemali površino celih pisarn.


Slika 1.2: Primerjava prenosnega računalnika Osborne Executive iz leta 1982 s hitrostjo procesorja 4 MhZ in mobilnega telefona iPhone iz leta 2007 s hitrostjo procesorja 412 MHz. Vir: [2]

Tehnologija postaja vse bolj poceni, manjša in vse bolj dostopna javnosti. Skupaj s tehnologijo za osebne računalnike se razvija tudi veja mikrokrmilnikov, mikroprocesorjev in manjših računalnikov. Mikrokrmilniki so integrirana vezja, ki v enem čipu združujejo centralno procesno enoto, delovni pomnilnik, statični pomnilnik ter različne vmesnike za povezavo med vhodno/izhodnimi napravami. Na trgu je dostopnih več mikrokrmilniških arhitektur (npr. AVR (angl. Advanced Virtual RISC), ARM (angl. Acorn RISC Machine), PIC (angl. Peripheral Interface Controller)). Podjetja večinoma za vsako družino ponujajo več mikrokrmilnikov, ki se razlikujejo v zmogljivosti ter perifernih napravah. Mikrokrmilniki so namenjeni za poganjanje namenskih aplikacij, nekateri pa so celo tako zmogljivi, da lahko na njih namestimo celo operacijski sistem. Iz mikrokrmilnikov so se tako razvili majhni in zmogljivi računalniki.

Za diplomsko nalogo smo se odločili izdelati svojo lastno vremensko postajo. Za jedro projekta smo uporabili računalnik Raspberry Pi. Nanj smo priključili senzorje za temperaturo, zračni tlak in vlago. Poleg teh, smo uporabili tudi modul GPS (angl. Global Positioning System). Ta nam služi za določitev točne lokacije, nadmorske višine in ure. Vse vrednosti meritev se periodično shranjujejo v podatkovno bazo. Računalnik

poganja operacijski sistem, znotraj katerega je spletni strežnik. Podatki naše vremenske postaje so uporabnikom dostopni preko spletnega vmesnika. Računalnik je mobilen, saj ga napaja baterija, za povezavo z internetom pa uporablja brezžični vmesnik. Za naš projekt smo uporabili računalnik Raspberry Pi [5], ki sloni na arhitekturi ARM [6]. Ta je trenutno eden izmed novejših in popularnejših manjših računalnikov na trgu. Povpraševanje zanj se hitro veča, saj je zaradi svoje širše uporabnosti primeren za reševanje raznovrstnih problemov (npr. avtomatizacija hiše, vmesnik za digitalno televizijo, avtoračunalnik).

V prvem delu diplomske naloge so opisane in predstavljene strojne in programske tehnologije, ki smo jih uporabili v projektu. V drugem delu je opisana praktična implementacija vremenske postaje skupaj z vsemi razvitimi programskimi sklopi ter primer praktične uporabe vremenske postaje. V zaključku smo opisali krajši pregled opravljenega dela ter predloge za nadaljno delo. V prilogah se nahaja programska koda za ključne sklope vremenske postaje.

# Poglavje 2

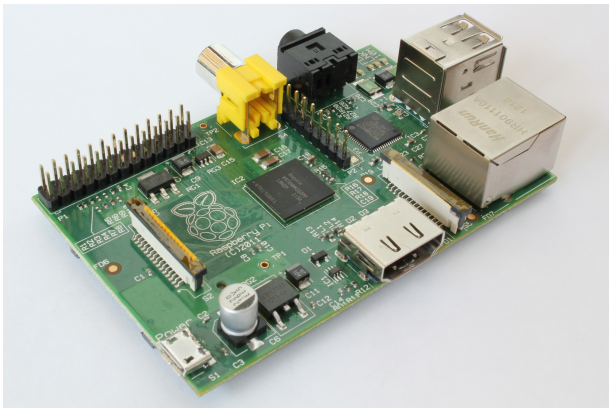
# Predstavitev uporabljenih tehnologij

V tem poglavju vam bomo predstavili tehnologije, ki smo jih uporabili pri izdelavi diplomskega dela.

## 2.1 Strojne tehnologije

### 2.1.1 Računalnik Raspberry Pi

Računalnik Raspberry Pi (slika 2.1) je malo večji od kreditne kartice in ga je iznašla foundacija Raspberry Pi z namenom popularizacije računalništva v šolah. Računalnik je cenovno dostopen, vsestransko uporaben in zelo dobro dokumentiran [7], zato je se pokazalo zanimanje s strani širše javnosti.



Slika 2.1: Računalnik Raspberry Pi. Vir slike: [5]

### Zgodovina

Ideja za razvoj računalnika, baziranega na mikrokrmilniku Atmel ATmega644, se je porajala že leta 2006. Ustanovitelj Eben Upton je zbral ekipo, ki je zajemala ljudi z različnih področij, kot so: učitelji, profesorji in računalniški navdušenci. Za cilj so si zastavili ustvariti računalnik, ki bo pritegnil pozornost mladih in jih navdušil za računalništvo. Najprej so ustvarili prototip računalnika v velikosti spominskega ključa na arhitekturi ARM.

Avgusta 2011 so bile ustvarjene prve alfa verzije Raspberry Pi. Te so bile identične načrtovanemu modelu B, le da so bile zaradi razhroščevalnika izdelane na večjem vezju. Demonstracija samega računalnika je bila uspešna, saj je lahko poganjal namizje LXDE (angl. Lightweight X11 Desktop Environment) na operacijskem sistemu Debian in igro Quake 3 v ločljivosti 1080p ter celo video v visoki ločljivosti, kompresiran s kodekom MPEG-4 preko vmesnika HDMI (angl. High-Definition Multimedia Interface).

Prvi teden leta 2012 je bilo na strani eBay prodanih prvih 10 računalnikov, ki so jih večinsko kupila večja podjetja za višjo ceno in tako finančno podprla projekt. Skupaj so zbrali skoraj 19.000 €, povpraševanje pa je preseglo ponudbo.

### Specifikacije

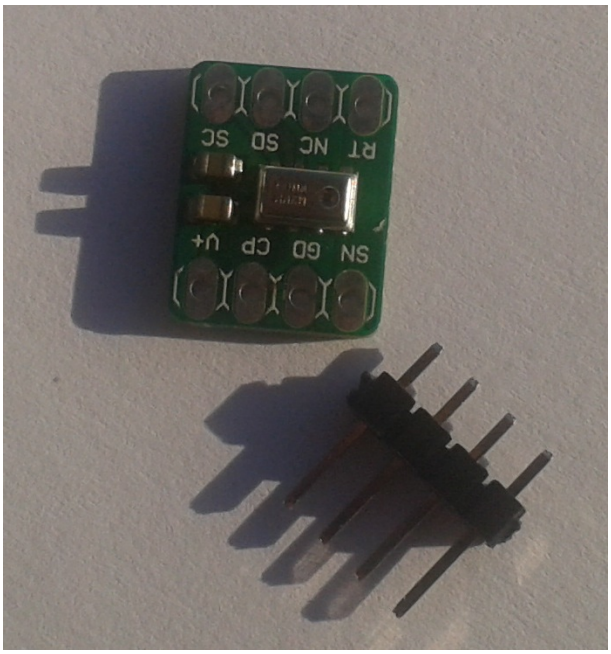
Trenutno sta na tržišču dostopni dve različici računalniških sistemov Raspberry Pi. Šibkejša različica A je omejena le z enim vmesnikom USB in je brez mrežnega vmesnika. V tem primeru lahko do interneta dostopamo preko vmesnika USB ali z brezžičnim adapterjem. Močnejša različica B ima v primerjavi z različico A podvojeno velikost glavnega pomnilnika (512 MB namesto 256 MB), kar se pozna tudi na porabi električne energije (3 W namesto 1.5 W). Nobena različica nima lastne baterije, zato jo je potrebno napajati preko vmesnika USB, baterije ali adapterja. Podrobnejša primerjava med obema modeloma je prikazana v tabeli 2.1.

| specifikacije | Model A | Model B |
|---|---|---|
| Cena: | 19 € | 27 € |
| Procesor: | 700 MHz jedro ARM1176JZF-S (družina ARM11) | |
| GPU: | Broadcom VideoCore IV OpenGL ES 2.0 (24 GFLOPS) MPEG-2 in VC-1, 1080p, h.264/MPEG-4 AVC | |
| Spomin: | 256 MB | 512 MB |
| USB 2.0: | 1x (na samem čipu) | 2x (integriran razdelilec USB) |
| Integrirani disk: | SD / MMC / SDIO reža za kartice (napajanje le 3.3 V) | |
| Mrežna kartica | Brez | 10/100 omrežna kartica preko vodila USB |
| Ostali izhodi: | 8 × GPIO, UART, vodilo $I^2C$, vodilo SPI in vodilo $I^2S$ za audio +3.3 V, +5 V, zemlja | |
| Video izhod: | Sestavljena RCA, HDMI, LCD Panels preko DSI, resolucija HDMI od 640×350 do 1920×1200 z standardi PAL in NTSC | |
| Audio izhod: | 3.5 mm vtič, HDMI in $I^2S$ audio (tudi primeren za audio) | |
| Poraba: | 300 mA (1.5 W) | 700 mA (3.5 W) |
| Napajanje: | 5 voltov preko vmesnika USB ali preko vmesnika GPIO | |
| Velikost: | 85.60 mm × 53.98 mm | |
| Teža: | 45 g | |
| Podprti operacijski sistemi: | Debian GNU/Linux, Raspbian OS, Fedora, Arch Linux ARM, RISC OS, FreeBSD, Plan 9 | |

Tabela 2.1: Podrobnejša primerjava različic Raspberry Pi A in B.

### 2.1.2 Senzor za merjenje temperature in zračnega tlaka

Za merjenje temperature in zračnega tlaka smo uporabili integrirano vezje MPL115A2 [8] proizvajalca Freescale, ki ima že integrirana oba senzorja (slika: 2.2). Zaradi vmesnika $I^2C$ in nizke cene je čip kot nalašč narejen za nizko-proračunske projekte. Čip meri le 5x3 mm, njegova normalna poraba pa znaša 5 mA. Do meritev dostopamo preko vmesnika $I^2C$ (angl. Inter-Integrated Circuit). Senzor je že kalibriran in primeren za takojšnjo rabo.



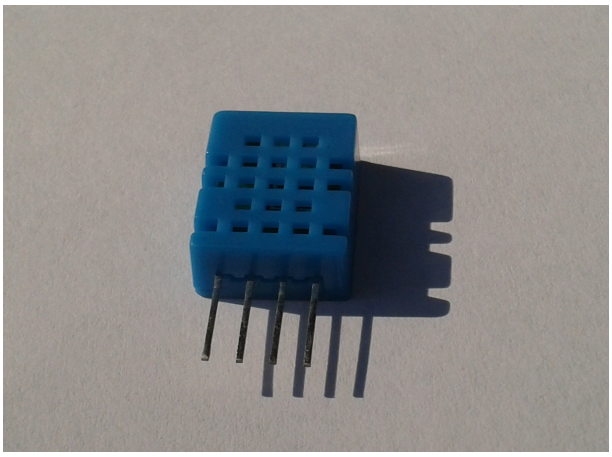

Slika 2.2: Senzor MPL115A2 vgrajen na tiskanem vezju.

### 2.1.3 Senzor za merjenje vlage

Vremenska postaja meri tudi vlažnost zraka. Za meritve uporablja senzor DHT11 (slika: 2.3), ki se tipično uporablja v klimatskih sistemih (angl. HVAC). Čip vsebuje senzor za vlago in temperaturo, oddaja pa kalibriran digitalni signal.

Senzor je posebno kalibriran za natančnejše merjenje vlage. Senzor je sila enostavno zgrajen, saj zahteva le serijski vmesnik iz ene žice (angl. single-wire serial interface). Zaradi velikosti in majhne porabe napajanja (3 - 5.5 V DC) je vmesnik primeren za naš projekt. Senzor za vlago meri v razponu od 20% do 90% relativne vlažnosti zraka.

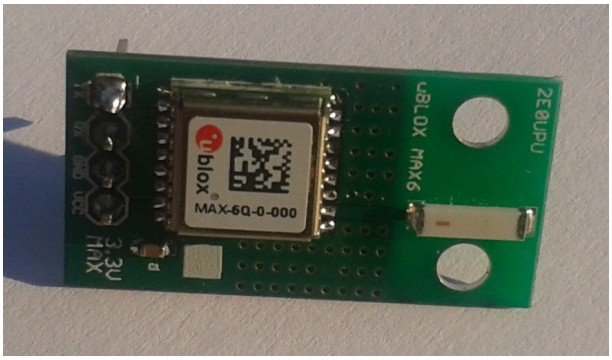
Senzor za vlago vsebuje tudi senzor za temperaturo. Ker ima omenjeni senzor za temperaturo kar visoko odstopanje (2 °C), bomo temperaturo merili s senzorjem MPL115A2.



Slika 2.3: Senzor DHT11 za merjenje vlage.

### 2.1.4 Sprejemnik GPS

Za določanje položaja in nadmorske višine smo na računalnik Raspberry Pi priključili tudi vmesnik GPS MAX-6 [9] proizvajalca u-blox. Sprejemnik, ki ga vidimo na sliki 2.4, podpira komunikacijo preko vmesnika UART. Slednji služi za sprejem podatkov ter konfiguracijo sprejemnika. Sprejemnik MAX-6 je možno optimizirati preko vmesnika UART (angl. Universal Asynchronous Receiver/Transmitter) za specifično uporabo (npr. pri veliki hitrosti, v višjih nadmorskih višinah). Za svoje delovanje sprejemnik potrebuje enosmerno napetost 3.3 V, pri kateri je njegova poraba 0.12 W. Za potrebe vremenske postaje smo se odločili za nakup modula s sprejemnikom GPS in vgrajeno anteno.




Slika 2.4: Sprejemnik MAX-6, vgrajen na tiskanem vezju, skupaj s sprejemno anteno.

### 2.1.5 Brezžični vmesnik WiFi

Za povezavo z internetom (ter posledično odjemalcem) lahko uporabimo kar mrežni kabel. Za izboljšanje mobilnosti smo na Raspberry Pi preko vodila USB priključili brezžični vmesnik. Vmesnik je zelo majhen (1.5 cm v širino in 2 cm v dolžino) in deluje na sistemu brez dodatnih konfiguracij gonilnika. S pomočjo sodobne tehnologije brezžičnih omrežij N je povezava pri daljših razdaljah zanesljivejša. Podatke lahko prenašamo s hitrostjo 150 Mbps. Vmesnik deluje na frekvenci 2.4 - 2.4835 GHz in podpira vse vrste kriptiranja.

## 2.2 Programske tehnologije

### 2.2.1 Operacijski sistem Raspbian wheezy

Raspbian [5] je brezplačni operacijski sistem, ki temelji na distribuciji Debian. Optimiziran je za delovanje na računalniku Raspberry Pi. Sam operacijski sistem sestavljajo osnovna orodja in pripomočki (npr. orodja za delo z vodilom $I^2C$), ki omogočajo optimalno uporabo računalnika. Poleg samega operacijskega sistema pa Raspbian vsebuje čez 35.000 paketkov različnih programov, ki so prilagojeni za hitro namestitev in samo delovanje. Operacijski sistem se še vedno razvija s poudarkom na stabilnosti in učinkovitosti.

### 2.2.2 Programski jezik Pyhon 2.7

Za programiranje aplikacije smo uporabili visoko-nivojski programski jezik Python [10]. Ta je podoben programskim jezikom Pearl, Ruby in shematično tudi Javi. Za razliko od ostalih ima Python preglednejšo in lažje berljivo sintakso. Je objektno orientiran, podpira hierarhijo paketov, ima visoko podporo implementacije in sinhronizacije z drugimi programskimi jeziki. Sintaksa Pythona omogoča programerju, da prikaže svoj koncept v krajši programski kodi, kot bi mu jo omogočal programski jezik C. Ne glede na velikost programa nam konstrukcija omogoča enostavno berljivost programa.

Kot pri ostalih visoko nivojskih programskih jezikih je Python pogosto uporabljen kot skriptni jezik. S pomočjo orodij drugih izdelovalcev lahko programsko kodo implementiramo v samostojno izvedljiv program. Pythonovi prevajalniki so enostavno naložljivi na večini operacijskih sistemov.

### 2.2.3 Podatkovna baza SQLite

SQLite [11] je podatkovna baza, napisana v jeziku C, namenjena izdelavi in upravljanju računalniških podatkovnih baz. Njena posebnost je majhna velikost (350 KB) ter podprtost na številnih platformah. V sami knjižnici je implementirana večina programskega jezika SQL (angl. Structured Query Language). Priljubljena je tudi kot podatkovna baza, integrirana v namizne aplikacije, kot na primer brskalnik. Dandanes je ena najbolj razširjenih pogonov za podatkovne baze, saj jo uporabljajo različni brskalniki in vgrajeni sistemi.

Za naš projekt smo uporabili različico SQLite3. Uporabili smo jo za kreiranje same baze ter za izvrševanje poizvedb med samim razvojem. SQLite3 lahko pošilja poizvedbe in vzdržuje datoteko baze SQLite. Celoten program je napisan v eni sami izvršilni datoteki, ki se nahaja na strežniškem računalniku. Uporaba SQLite je možna v 30-ih programskih jezikih.

### 2.2.4 Spletno ogrodje CherryPy

CherryPy [12] je minimalistično spletno ogrodje, napisano v programskem jeziku Python. Razvijalcem omogoča ustvarjanje spletnih aplikacij, enako kot bi ustvarili objektno usmerjen program. Posledično je za to porabljenega manj časa in koda je preglednejša. Ogrodje CherryPy nam ponuja veliko gradnikov, ki poenostavijo predpolnjenje, kodiranje, vzpostavljanje seje, avtorizacije in prikazovanje statične vsebine. Dodatno ima ogrodje CherryPy vgrajen lastni spletni strežnik, ki omogoča testiranje aplikacij v času razvoja. Hkrati lahko posreduje željeno vsebino večim odjemalcem (na različnih vratih). Končna spletna aplikacija, izdelana v ogrodju CherryPy, je v obliki knjižnice Python. Spletni strežnik do nje dostopa preko vmesnika WSGI (angl. Web Server Gateway Interface). Zaradi take oblike lahko spletne aplikacije CherryPy enostavno povežemo z drugimi relacijskimi bazami (SQLite, MySQL, idr.), knjižnicami s predlogami (npr. Mako, Cheetah, idr.) in spletnimi strežniki.

### 2.2.5 Spletni strežnik nginx

Nginx [13] je odprtokodni spletni strežnik, ki posreduje podatke preko protokolov HTTP (angl. Hypertext Transfer Protocol Hypertext Transfer Protocol), SMTP (angl. Simple Mail Transfer Protocol) in IMAP (ang. Internet Message Access Protocol) z visoko stopnjo vzporednosti (sočasnosti), izjemno hitrostjo in majhno porabo pomnilnika. Program je izdan pod licenco FreeBSD, kar pomeni, da lahko program uporabljamo zastonj.

Nginx lahko poganjamo iz različnih operacijskih sistemov, kot so: Unix, Linux, variacije BSD, Mac OS X, Solaris, AIX, HP-UX in Microsoft Windows.

Dinamični prikaz HTTP vsebine se pri nginx-u izvaja preko FastCGI (ang. Fast Common Gateway Interface), modula za izvajanje skript za SCGI (angl. Simple Common Gateway Interface) ali WSGI (angl. Web Server Gateway Interface) aplikacijskih serverjev. Program služi kot razdelilnik, saj v primerih večjega navala zahtevanih informacij razporedi delo na več računalnikov ali pa razdrobi zahteve in jih tako lažje in hitrejše posreduje naprej. Nginx uporablja nesinhron dogodkovno voden dostop do obravnavanja zahtevkov za razliko od konkurenčnega strežnika Apache HTTP, ki uporablja niti in procesno-orentiran dostop. Bistvena razlika pri dogodkovno vodenih pristopih je ta, da je ob velikih naporih bolj učinkovit.

### 2.2.6 Vmesnik WSGI

Ideja za razvoj se je pojavila zaradi težav novih uporabnikov pri uporabi Pythonovih spletnih aplikacij. Tako je nastal WSGI [14], ki nudi uporabnikom univerzalni vmesnik med spletnimi strežniki in spletnimi aplikacijami ali drugimi spletnimi ogrodji za programski jezik Python.

WSGI se deli na dva dela: strežniški in aplikacijski del. Za procesiranje samega zahtevka mora strežnik spletni splikaciji poslati informacijo o seji in zahtevku (npr. spletni naslov, argumente obrazca, podatke o piškotkih) in funkcijo za posredovanje odgovora aplikacijskemu delu. Spletna aplikacija najprej pokliče posredovano funkcijo za odgovor HTTP, nato pa vrne niz znakov z vsebino HTML (angl. HyperText Markup Language).

Osnovni primer spletne aplikacije skladne z vmesnikom WSGI:

```
1 def application(environ, start_response):
2     start_response('200 OK', [('Content-Type', 'text/plain')])
3     yield 'Hello World\n'
```

Vmesnik WSGI je podprt s strani številnih spletnih strežnikov (npr. nginx, Apache, idr.) in ogrodij za izdelavo spletnih strani (npr. CherryPy, Django, web.py).

### 2.2.7 Orodje Crontab

Za osveževanje vnosov v podatkovni bazi uporabljamo orodje Crontab [15]. Uporabnikom omogoča periodično izvajanje programov ali lupinskih skript ob določenem času. V praksi se uporablja za avtomatizacijo vzdrževanja sistemov, lahko pa ga uporabimo tudi za ostale namene, kot npr.: povezava na internet, prenos elektronske pošte, periodično varnostno kopiranje podatkov.

### 2.2.8 Knjižnica s predlogami Mako

Knjižnica Mako [16] je ogrodje, ki je napisano v programskem jeziku Python. Ta omogoča uporabo predlog pri ustvarjanju spletnih strani v formatu HTML. Cilj uporabe predloge je ločevanje med obliko in programsko logiko. Programska logika se tako nahaja v spletni aplikaciji, kjer se vnaprej pripravijo vse potrebne dinamične vsebine za izpis na spletni strani. V predlogi sta definirana oblika in izgled spletne strani, predpripravljeno je mesto za vsebino, vendar brez vsebine. Končni korak je vstavljanje vsebine na določeno mesto v predlogi, za kar poskrbi knjižnica sama.

# Poglavje 3

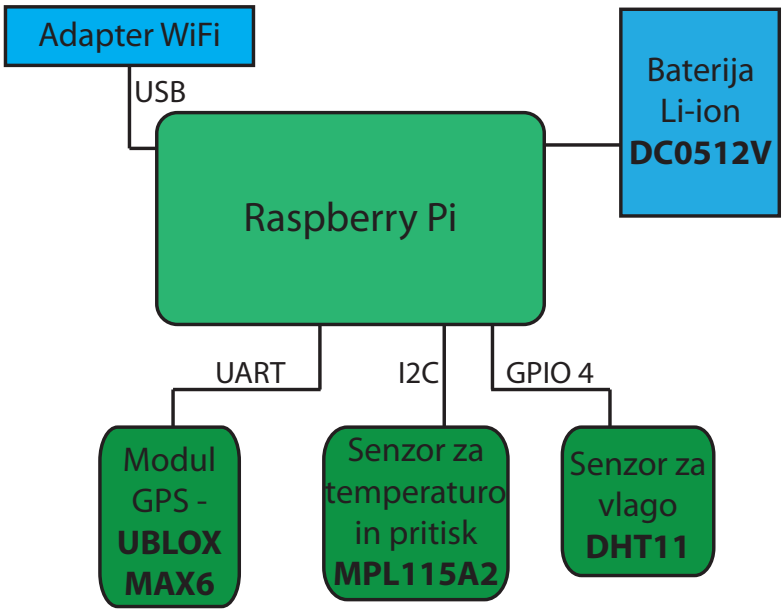
# Implementacija vremenske postaje

V tem poglavju bomo podrobneje opisali postopek implementacije strojnega dela in priklop senzorjev z njihovimi povezavami. V nadaljevanju pa bomo opisali programsko plat vremenske postaje.

## 3.1 Strojni del

Na sliki 3.1 je prikazana shema strojnih tehnologij, vrste povezav in interakcij med njimi.

Osnovo predstavlja računalnik Raspberry Pi. V fazi razvoja vremenske postaje smo računalnik napajali preko adapterja mikro USB. Ob uporabi pa smo ga priključili na baterijo z enako napetostjo ali na napajanje preko mrežnega kabla (angl. Power over Ethernet (PoE)). Vmesnik HDMI (angl. High-Definition Multimedia Interface) nam omogoča enostavnejši prikaz nastavitev nekaterih konfiguracij. V času razvoja smo računalnik priključili na splet preko mrežnega kabla, ob delovanju pa smo nanj priklopili brezžični adapter USB.




Slika 3.1: Shema strojnih tehnologij.

### 3.1.1 Priklop sprejemnika GPS

Na računalnik smo priključili tudi različne senzorje. Vodilo UART nam je omogočilo priklop modula GPS. Pri tem smo morali biti posebej pozorni, ker tega modula ne smemo priključiti na napetost višjo od 3 V. Modul smo povezali na ustrezeno nožico, podatkovne poti pa je bilo potrebno povezati z nožicami TXD (angl. Transmit Data - za prenos bajtov iz računalnika) in RXD (angl. Receive Data - za prenos bajtov v računalnik).

### 3.1.2 Priklop senzorja za vlago

Senzor za vlago smo priklopili na 5 V. Informacije, ki jih oddaja, pa prejemamo na nožico GPIO 4 (angl. General-purpose input/output - nožica za splošne potrebe). Med njima je bilo potrebno povezati upor vrednosti 4.7 kΩ.

### 3.1.3 Priklop senzorja za pritisk in temperaturo

Za merjenje temperature in pritiska smo uporabili senzor MPL115A2. Tega smo povezali na vodilo $I^2C$. To smo povezali na preostalo nožico s 5 V in ga priklopili na nožici

GPIO 2(SDA - ang. Serial Data Line - nožica za prenos podatkov) in GPIO 3(SCL - ang. Serial Clock - nožica za serijsko uro).

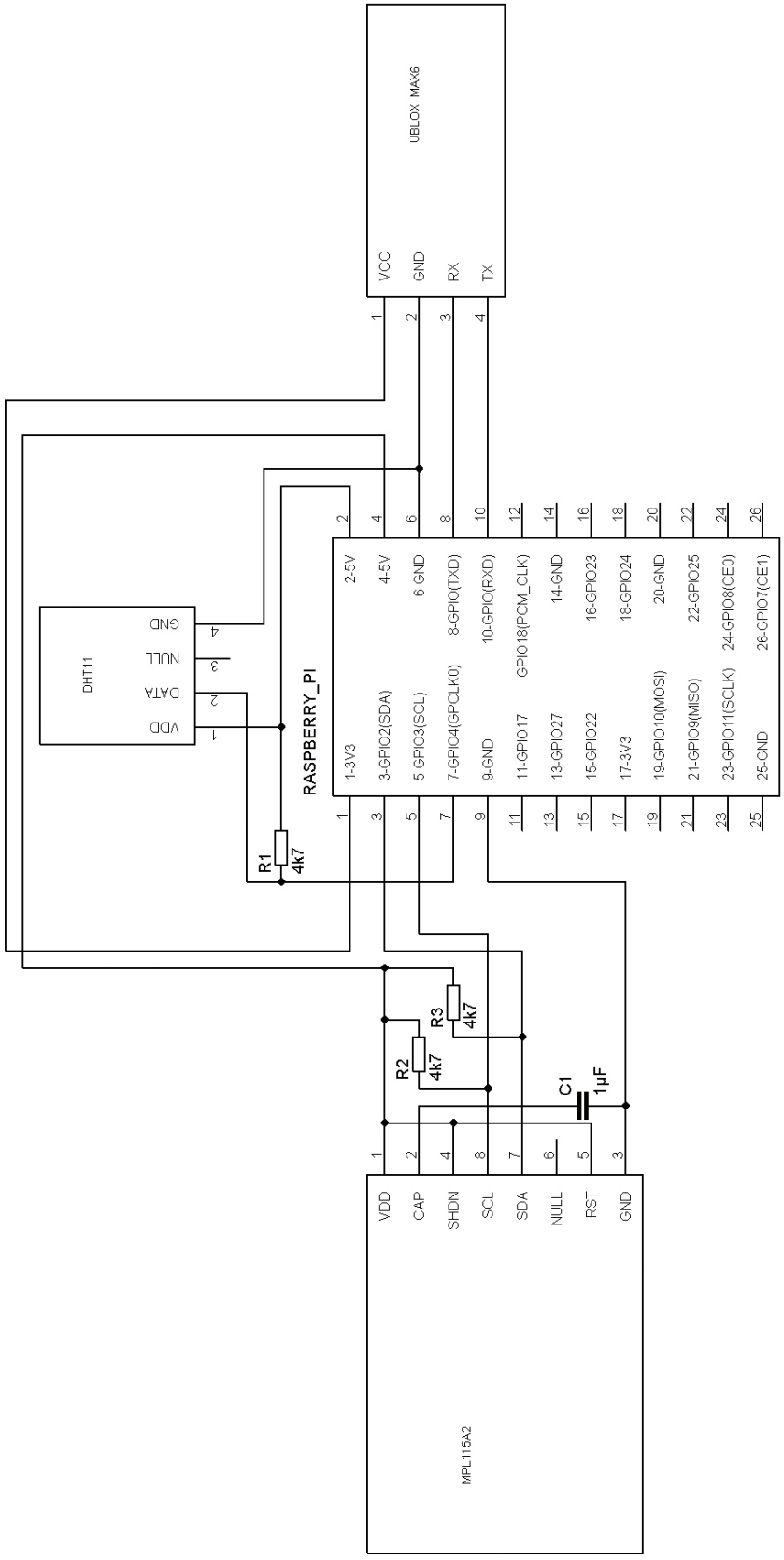
Vse povezave je bilo potrebno ozemljiti (nožica GND (angl. Ground)). Tako smo priklopili vse senzorje, ostale nožice pa bi lahko uporabili za nadgradnjo števila senzorjev. Povezanost senzorjev lahko vidimo iz shematske slike 3.2.

### 3.1.4 Vgradnja konstrukcije v vodotesno ohišje

Celotno konstrukcijo smo vgradili v vodotesno elektro razdelilno dozo in jo s tem zaščitili pred vremenskimi vplivi. Senzorje za temperaturo, vlago in pritisk smo priklopili na posebno tiskano vezje, ki smo ga priključili preko vodnika (8-žilni mrežni kabel) na konstrukcijo v dozi. Senzorji tako merijo fizikalne parametre zunanjega zraka.

### 3.1.5 Modifikacija mrežnega adapterja

Za alternativni dostop do vremenske postaje smo naredili dodatno napajanje preko mrežnega kabla. Mrežni kabel potrebuje za komunikacijo le 4 od skupno 8-ih žic. Modificirali smo adapter za podaljšek mrežnega kabla, tako da smo nanj priklopili adapter za napajanje. Potrebno je bilo izmeriti aktivne in pasivne žice ter jih ustrezno povezati z napajalnimi. Pri priklopu PoE adapterja moramo biti še posebej previdni, da je ta pravilno usmerjen. Ob napačnem priklopu bi prišlo do kratkega stika. Iz ZDA smo naročili poseben vodotesen vmesnik za priklop mrežnega kabla, ki ima na izhodni strani že ločen napajalni in mrežni kabel.




Slika 3.2: Slika sheme povezav med senzorji in krmilnikom Raspberry Pi

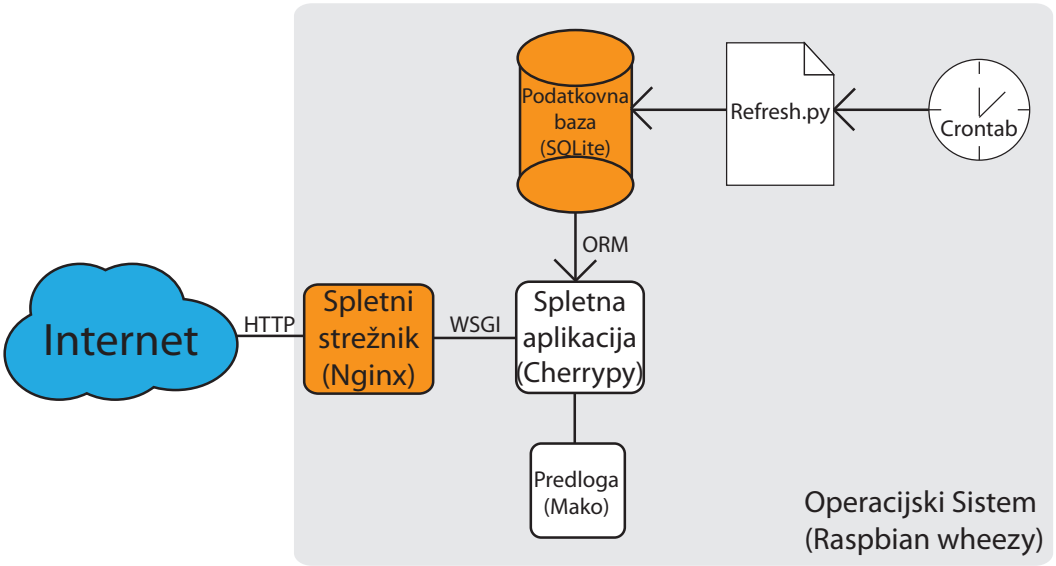
## 3.2 Programski del

Večino časa je bilo posvečenega programskemu delu projekta. Slika 3.3 prikazuje uporabljene programske tehnologije in njihove interakcije.

Glavnina projekta je bila napisana v programskem jeziku Python, ki je že implementiran ob namestitvi operacijskega sistema. S pomočjo Python knjižnice SQLAlchemy smo ustvarili novo podatkovno bazo, ki zajema vse namerjene podatke.

Napisati je bilo potrebno nov program, ki zajame meritve iz senzorjev, jih ustrezno obdela in jih shrani v podatkovno bazo. V orodju Crontab (tudi del operacijskega sistema) smo morali nastaviti pogostost izvajanja in pot do naše datoteke, ki skrbi za polnjenje baze. Tako smo dobili konsistentnost in avtomatizacijo poganjanja našega programa.

Prikaz spletne strani nam je omogočila spletna aplikacija CherryPy. Potrebno jo je bilo sprogramirati, da s pomočjo modificirane predloge Mako vstavi podatke baze na ustrezna mesta. Ob zagonu program generira uporabniku prijazen spletni vmesnik. Za prikaz na spletu je bilo potrebno namestiti program nginx in mu nastaviti pravilne sistemske poti do naše aplikacije. Aplikacije se zaporedno izvedejo in pošljejo vsebino končnemu uporabniku, ko ta pošlje zahtevek za prikaz spletne strani.




Slika 3.3: Shema programskih tehnologij.

### 3.2.1 Zajem podatkov iz senzorjev

Za branje podatkov GPS z vodila UART smo morali implementirati posebno knjižnico Serial, ki nam to omogoči. Na vratih AMA0 in frekvenci 9600 nam modul ciklično pošilja podatke. Te podatke smo morali zajeti in iz njih izluščiti zemljepisno širino in dolžino. Obe števili sta izmerjeni v stopinjah, zato smo morali uporabiti formulo (3.1) za pretvorbo v decimalni števili.

$$n = d + \frac{m}{60} + \frac{s}{3600} \tag{3.1}$$

V enačbi predstavlja seštevanec d stopinje (zemljepisna dolžina se meri v stopinjah od -180° na vzhodni polobli do +180° na zahodni polobli z ničtim poldnevnikom v Greenwichu. Zemljepisna širina pa se meri v stopinjah od južnega tečaja pri -90° do severnega tečaja pri +90° z ničtim vzporednikom na ekvatorju). Ko seštejemo vrednosti, dobimo longitudo in latitudo, ki ju uporabimo za določitev pozicije. Poleg omenjenih nam sprejemnik GPS posreduje tudi nadmorsko višino.

Za merjenje vlage smo morali namestiti gonilnike [17] iz uradne spletne strani. Ti imajo že delujočo knjižnico, ki je napisana v programskem jeziku C. Knjižnici je potrebno preko argumentov podati številko nožice za sporočanje podatkov in model senzorja (DHT11 v našem primeru). Knjižnica prebere podatke in jih posreduje na izhod. Naša naloga je bila zajem in prilagoditev izmerjenih vrednosti za vnos v podatkovno bazo. Senzor ne more brati podatkov na manj kot pet sekund. Na to moramo biti še posebej pozorni pri ročnem vnosu meritev v podatkovno bazo.

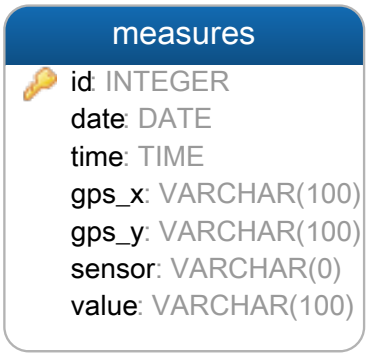
Za merjenje pritiska in temperature s senzorjem MPL115A2 smo morali namestiti neuradni program [18], ki je bil posebej prilagojen za računalnik Raspberry Pi. Ob zagonu sistema smo morali omogočiti dostop do vodila $I^2C$. S tem smo sistemu dovolili dostop do zajemanja podatkov preko tega vodila. Izmerjene podatke smo preoblikovali v pravilno obliko zapisa, ki ustreza pravilom za shranjevanje podatkov v podatkovni bazi (glej poglavje 3.2.1).

### 3.2.2 Podatkovna baza

Podatkovno bazo smo ustvarili s pomočjo knjižnice SQLite. V njej smo shranili podatke, ki so jih izmerili senzorji. Ob vnosu v bazo smo morali decimalna števila najprej množiti

s koeficientom 100 zaradi natančnosti in kompatibilnosti s knjižnico SQLAlchemy. Tako smo dobili podatkovni model, ki je prikazan na sliki 3.4. Imena in obrazložitev atributov:

- ID (zaporedni vnos),
- date (datum meritve),
- time (čas meritve),
- gps_x (gps koordinatna vrednost x),
- gps_y (gps koordinatna vrednost y),
- sensor (vrsta senzorja - [temperature] za merjenje temperature, [moist] za merjenje vlage, [pressure] za merjenje pritiska in [height] za merjenje nadmorske višine),
- value (številčna vrednost senzorja).




Slika 3.4: Model podatkovne baze narejen v programu SQLite

### 3.2.3 Nastavitev podatkov za prikaz

Implementacija Googlovega zemljevida je enostavno predstavljena na njihovi uradni strani za razvijalce [19]. Zemljevidu je bilo potrebno predhodno nastaviti pozicijo izrisa na spletni strani, vnesti koordinate GPS in nastaviti željeno povečavo. Skripta sama centrira zemljevid na določene koordinate, vendar smo zaradi lažje predstave na isto mesto postavili tudi zaznamek (v primeru premika zemljevida ostane zaznamek vedno na določenih koordinatah).

Za pridobitev meritev je bilo potrebno bazo najprej razčleniti na vsak senzor posebej. Po razčlenitvi smo izvedli funkcije za pridobitev minimuma, maksimuma in povprečja z vrednostmi, ki smo jih pridobili s posameznimi senzorji. Za število vseh meritev smo uporabili kar nazadnje vnešen ID, za meritev posameznega senzorja pa to številko delili s številom senzorjev v bazi. Za poizvedbo časa delovanja sistema smo uporabili sistemsko funkcijo Uptime, ki vrne čas od zadnjega sistemskega zagona dalje. Za pridobitev časa prve meritve je bilo potrebno podatke v bazi sortirati po ID-ju naraščajoče in od prvega vnosa združiti datum in čas.

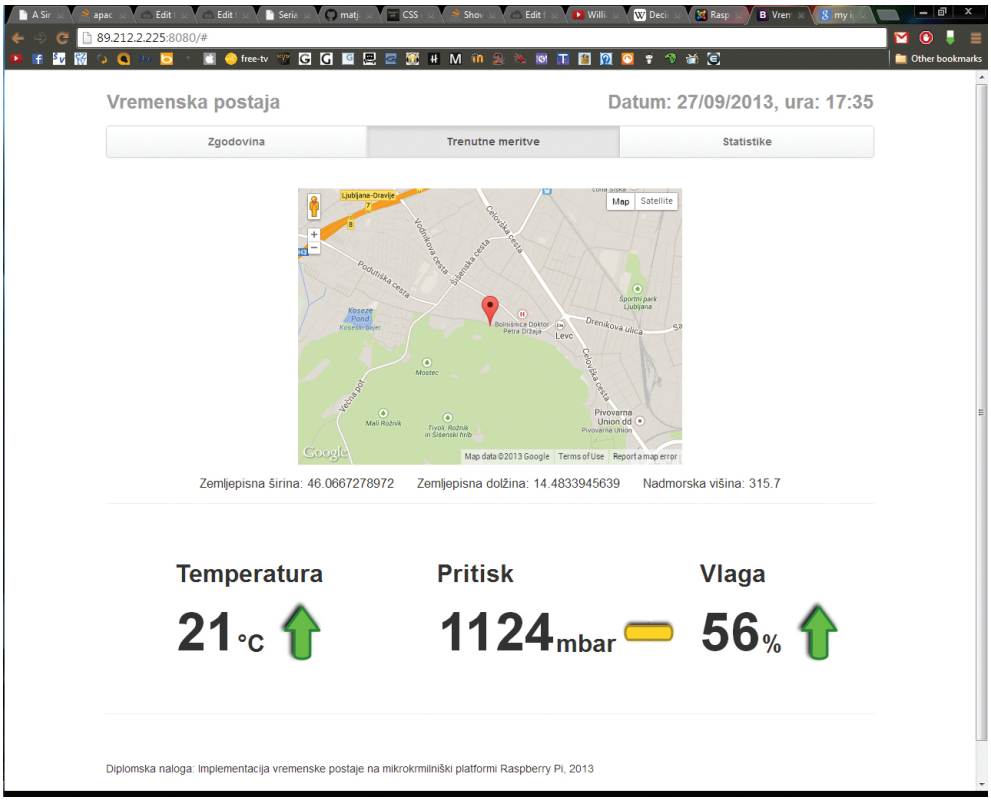
Za izris grafov smo uporabili knjižnico JavaScript [20]. Potrebno je bilo zajeti 1152 zadnjih meritev iz podatkovne baze. Meritve smo razčlenili po senzorjih in tako dobili 288 meritev vsakega senzorja. Seznam meritev je bilo potrebno združiti v skupine po 12 (število meritev na uro), izračunati njihovo povprečje in jih ponovno vnesti v seznam. Tako smo dobili 24 povprečnih meritev za minulih 24 ur. Podobno smo naredili tudi s časovnimi poizvedbami. Ure in minute je bilo potrebno pretvoriti v sekunde za lažje računanje povprečja. Rezultate smo iz sekund ponovno pretvorili v ure in minute. Vse sezname je bilo na koncu potrebno še zrcaliti (zaradi sortiranja - od zadnje meritve padajoče). Vrednosti se posredujejo predlogi mako, ki jih postavi na pravilno mesto v datoteko HTML.

### 3.2.4 Spletna aplikacija

Spletna aplikacija nam omogoča lepši prikaz podatkov na spletni strani. Podatke meritev nam posreduje naš program webapp.py. Ob branju decimalnih števil smo zaradi estetike prikazali cela števila brez decimalk. To smo storili množenjem s 100, nato pa število še zaokrožili. Predloga Mako jih ustrezno umesti na primerno mesto. Iz namenjenih poizvedb v podatkovno bazo nam ta pošlje zahtevane podatke vključno z zadnjimi izmerjenimi. Tako skrbimo za prikaz ažurnih vrednosti.

Za lepši prikaz spletnega vmesnika smo uporabili knjižnico Bootstrap. Ta nam je omogočila lažje oblikovanje spletne strani. Za delovanje strani je bilo potrebno vkjučiti tudi knjižnice programskega jezika jQuery, JavaScript in oblikovalskega jezika CSS (angl. Cascading Style Sheets). S pomočjo teh smo omogočili knjižnici Bootstrap prikaz integriranih komponent. Bootstrap [21] omogoča enostavnejšo implementacijo elementov HTML, lažjo vizualno postavitev spletne strani in že vključuje lasten CSS, ki skrbi za oblike, barve, senčenje in velikosti. JavaScript in jQuerry [22] sta nam poenostavila programiranje grafov, aktivacijo in preklapljanje gumbov, skrivanje in prikaz komponent, prikaz zemljevida GPS in obračanje puščic. Spletna knjižnica Ajax (angl. Asynchro-

nous JavaScript and XML - razširitev programskega jezika JavaScript) nam omogoča asinhrono osveževanje podatkov na spletni strani. Knjižnica vsakih petnajst sekund zamenja podatke dinamičnih spremenljivk. Uporabili smo jo za osvežitev podatkov iz senzorjev, ki so prikazani na spodnjem delu spletne strani. To je razvidno na sliki 3.5.




Slika 3.5: Prikaz privzete strani trenutnih meritev, kot ga vidi končni uporabnik.

## 3.3 Grafični vmesnik

Grafični vmesnik sestavljajo tri glavne komponente: glava, srednji del in noga.

V glavi imamo na levi strani naslov, na desni pa datum in uro. Ta je nastavljena sistemsko in se shranjuje z vnosom v bazo.

V srednjem delu se vsebina spreminja glede na zahtevo uporabnika. Podrobneje bomo vsebino opisali v naslednjih poglavjih.

Noga je namenjena za prikaz trenutnih meritev. Te so izmerjene vsakih pet minut. Poleg izmerjenih vrednosti meritev pa imamo prikazane še slike puščic, ki ponazarjajo dviganje ali spuščanje vrednosti posameznih parametrov. Puščice se spremenijo le, če imajo zadosten odklon od povprečja in so v sklopu zadnjih petih meritev. Za prikaz

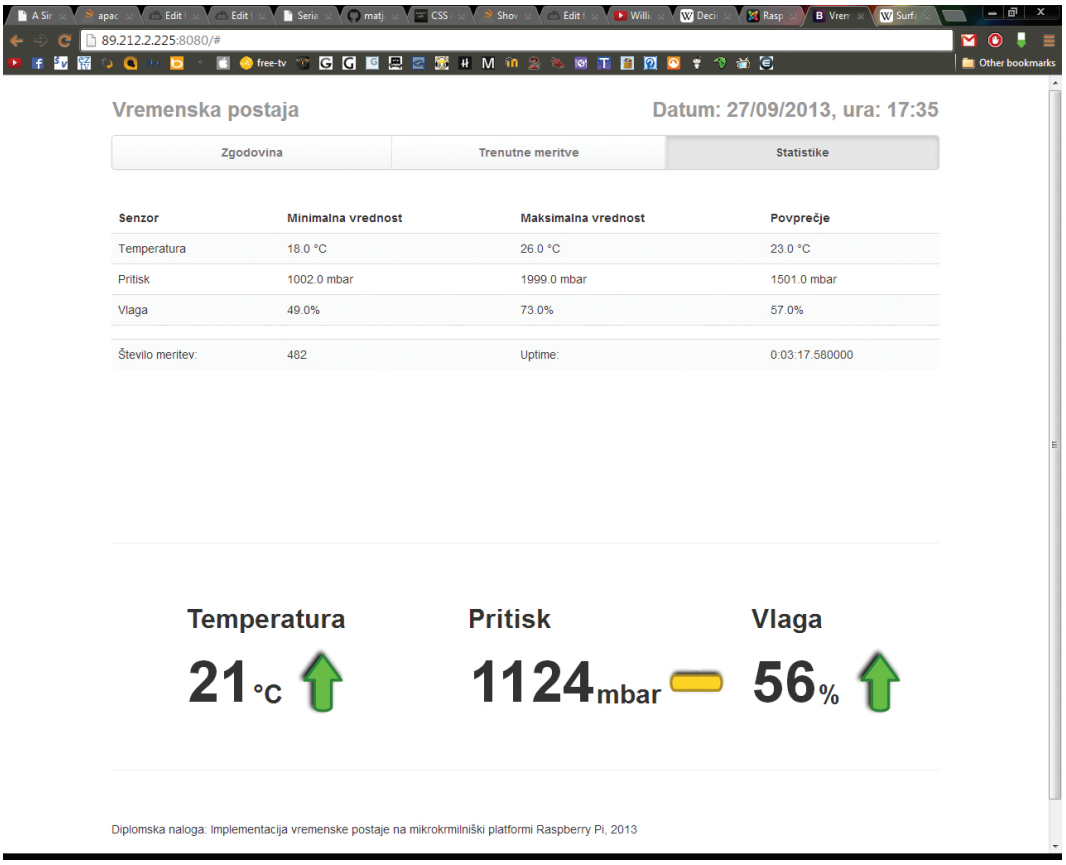
padca oziroma rasti temperature zadošča odklon za 1 °C, za prikaz spremembe pritiska zadošča 50 mbarov, za vlago pa 1%.

### Trenutne meritve

Obisk spletne strani nam namenoma prikaže privzeto stran vseh trenutnih meritev. Tako imamo večino informacij že na prvi strani. Pred nami se pojavi Google zemljevid, kot ga vidimo na sliki 3.5. Prikazuje nam trenutno izmerjene koordinate GPS. Te so izpisane pod zemljevidom vključno z nadmorsko višino, ki jo prav tako pridobimo preko sprejemnika GPS.

### Statistika

Na strani statistike smo prikazali nekaj zanimivih statističnih podatkov, kar je razvidno iz slike 3.6.



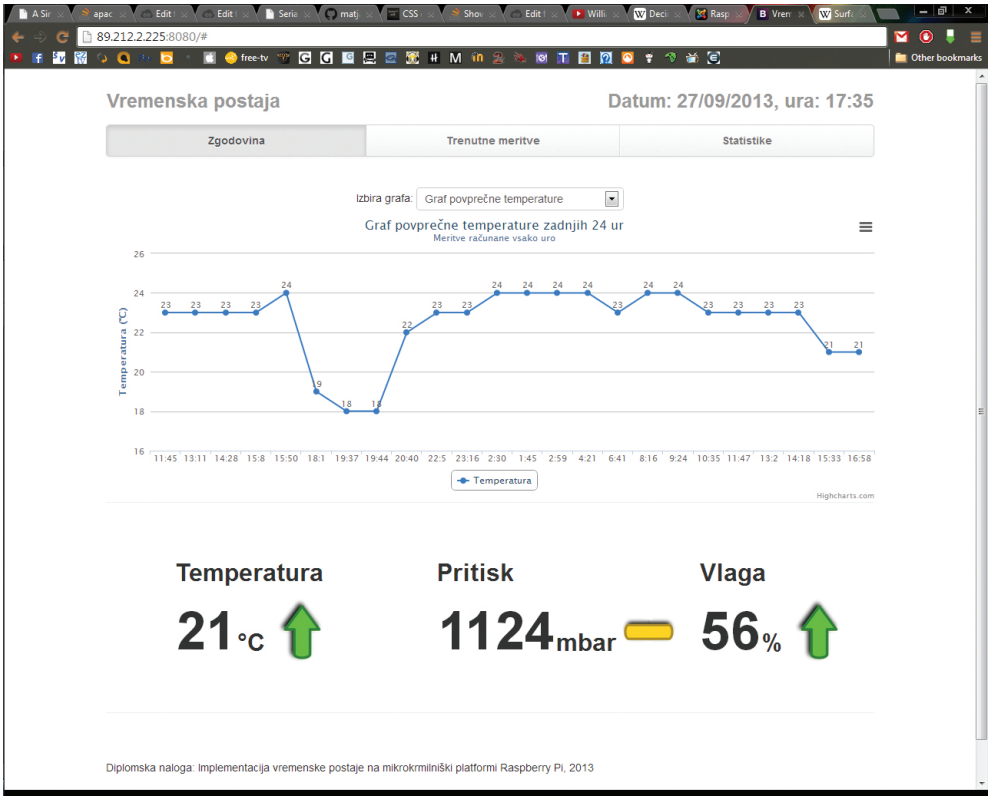

Slika 3.6: Prikaz strani s statističnimi podatki.

Vse poizvedbe smo predstavili v tabeli, ki smo jo uvozili iz Bootstrapa. V njej so prikazani naslednji podatki:

- maksimalne in minimalne vrednosti,
- število vseh meritev (število vnosov v bazo),
- število meritev posameznega senzorja,
- čas delovanja sistema (pove nam, koliko časa sistem aktivno zbira podatke iz senzorjev),
- čas prve meritve.

### Zgodovina meritev

Stran zgodovine meritev vsebuje grafe povprečnih meritev v zadnih 24-ih urah. Na sliki 3.7 lahko vidimo primer grafa povprečne temperature.




Slika 3.7: Stran zgodovine meritev s prikazanim grafom povprečne temperature v zadnjih 24-ih urah.

Za izbiro izrisa posameznega grafa smo ustvarili tudi padajoči menu, ki nam ponudi izbiro različnih grafov. Ta deluje tako, da skrije vse grafe in prikaže le tistega, ki je trenutno izbran v menuju. Vsi grafi so linijski. Na osi y je prikazana lestvica izbranih parametrov (temperatura, pritisk, vlaga). Os x pa predstavlja časovno premico. S klikom na znak v desnem zgornjem kotu grafa ga lahko shranimo na računalik. Izbiramo lahko med različnimi formati slike (jpeg, tiff, pdf in vektorski zapis). V kolikor v podatkovni bazi ni dovolj vnosov, nam stran to sporoči.

## 3.4 Namestitev in konfiguracija

Iz uradne strani je bilo potrebno prenesti sliko operacijskega sistema Raspbian "wheezy" [23]. Kartico SD je bilo potrebno predhodno formatirati in prenesti program Win32-DiskImager, s katerim smo na kartico naložili zagonski operacijski sistem. Naslova IP (angl. Internet Protocol) ni bilo potrebno nastavljati, saj ima novejša verzija operacijskega sistema že integriran protokol DHCP (angl. Dynamic Host Configuration Protocol). Tako smo ob priklopu mrežnega kabla že imeli delujočo internetno povezavo. Zaradi praktičnosti smo si namestilil tudi program VNC (angl. Virtual Network Computing) [24], ki nam omogoča oddaljen dostop do grafičnega vmesnika. V operacijskem sistemu smo nastavili konfiguracijo za zagon VNC-ja ob zagonu sistema [25].

Do konzole smo dostopali preko protokola SSH (angl. Secure Shell) ali z orodjem PuTTY [26]. Za urejanje tekstovnih datotek smo si pomagali z integriranim orodjem Vim [27]. Prenos datotek iz Raspberry Pi-ja na stacionarni računalnik nam je omogočil ukaz SCP (angl. Secure copy) protokola SSH. Za več prenosov in gradnjo strukture drevesa pa smo uporabili grafično orodje Cyberduck [28].

Samega programskega okolja Python ni bilo potrebno nastavljati, saj je že vključen v operacijski sistem.

### 3.4.1 Kreiranje baze

V poglavju 3.5 smo opisali, iz katerega repozitorija si lahko prenesemo datoteke za delovanje vremenske postaje. Ko imamo datoteke na svojem računalniku, lahko zaženemo program init_db.py. Ta nam ustvari novo podatkovno bazo. V kolikor podatkovna baza že obstaja, jo lahko spraznemo s pogonom programa clr_db.py.

### 3.4.2 Nastavitev orodja Crontab

Orodje Crontab skrbi za cikličen pogon datoteke, ki bere podatke in jih shranjuje v podatkovno bazo. Zaradi lažjega pogona smo napisali svojo skripto v programskem jeziku bash z imenom "run_script.sh". To moramo odpreti z urejevalnikom teksta in spremeniti pot do datoteke "refresh.py"(v kolikor njena pot ni /usr/share/nginx/www). Za lažjo predstavitev kopiramo datoteko run_script.sh z ukazom: "cp run_script.sh /usr/bin/". Do Crontab-a dostopamo z ukazom: "sudo crontab -e", vpišemo geslo in na koncu dodamo vrstico: " */5 * * * * /usr/bin/run_script.sh". S tem smo omogočili sistemu pogon zahtevane datoteke.

### 3.4.3 Postavitev nginx strežnika

Za postavitev lastnega spletnega strežnika moramo najprej pregledati ali imamo vse zadnje sistemske posodobitve. To naredimo z ukazom: "sudo apt-get update". Strežniški paket naložimo z ukazom: "sudo apt-get install nginx". Nginx se tako naloži v sistem in ustvari privzeto pot do bodoče javno dostopne mape. Za serviranje vsebine moramo vse naše datoteke skopirati v to mapo z ukazom: "cp ./* /usr/share/nginx/www -r". V nastavitvah nginx-a moramo nastaviti vrata in pot do naše aplikacije. To spremenimo z ukazom "vim /etc/nginx/sites-available/mysite" in vnesemo naslednjo vsebino:

```
1 server {
2         listen 80;
3         root /usr/share/nginx/www;
4         server_name 44.92.213.266 #IP naslov
5         index index.html index.htm;
6 }
```

Za aktivacijo dostopa do interneta, moramo najprej spremeniti trenutno lokacijo z ukazom: "cd /etc/nginx/sites-enabled" in nato ustvariti mehko povezavo z ukazom: "sudo ln -s ../sites-available/mysite". S tako nastavitvijo smo aktivirali našo spletno stran, vnesli naslov za dostop, številko vrat in serviranje datotek. Strežnik lahko nato zaženemo z ukazom: "sudo service nginx start".

## 3.5 Seznam in opis datotek

Vse datoteke so dostopne na spletnem portalu GitHub na naslovu: https://github.com /matjaxy/Vremenska_postaja. S klikom na gumb "download ZIP" lahko prenesemo celoten projekt na svoj računalnik. Prenešen paket vsebuje naslednje datoteke:

- webapp.py - jedro programa, ki iz baze bere podatke in jih posreduje predlogi Mako,
- refresh.py - program za ustvarjanje seje in pravila za vnos podatkov v podatkovno bazo (s pogonom te datoteke ročno vnesemo trenutne meritve),
- index.htm - osnova html strani na podlagi predloge Mako (skrbi za prikaz elementov na spletni strani),
- i2c.py - program za branje iz vodila $I^2C$ (branje temperature in pritiska),
- init_db.py - skrbi za kreiranje podatkovne baze (v kolikor hočemo na novo kreirati podatkovno bazo),
- alchemy_interface.py - vsebuje lastnosti in pravila vnosov v podatkovno bazo,
- measures.db - že obstoječa podatkovna baza (v kolikor želimo novo, jo lahko zbrišemo in na novo kreiramo tako, da poženemo program clr_db.py),
- prod.conf - vsebuje nastavitve za prikazovanje statičnih slik na spletni strani,
- clr_db.py - program za avtomatski izbris in kreiranje nove podatkovne baze,
- run_script.sh - skripta za posredovanje zagona datoteke za branje iz senzorjev,
- mapa css - vsebuje Bootstrap datoteke za lepši izgled spletne strani,
- mapa js - nekatere Bootstrap datoteke za poganjanje JavaScript in jQuery knjižnic (nekatere so dostopane preko interneta),
- mapa img - v tej mapi so shranjene vse slike, ki so prikazane na spletni strani,
- mapa eagle - v tej mapi se nahaja shema vseh povezav, narejena v programu eagle.

# Poglavje 4

# Zaključek

V diplomski nalogi smo predstavili postopek postavitve vremenske postaje na računalniku Raspberry Pi. Z rezultati smo zelo zadovoljni, saj smo v kratkem času postavili svoj sistem z delujočimi senzorji za temperaturo, vlago in pritisk ter sprejemnikom GPS. Glede na funkcionalnost same postaje nas lahko navdušuje dejstvo, da tehnologija postaja vse bolj dostopna in si jo lahko privošči vsak. Ljudje iz dneva v dan predstavljajo različne ideje uporabe manjših računalnikov. Tudi tisti, ki imajo le osnove programiranja, si lahko sami prilagodijo krmilnike za lajšanje vsakodnevnih opravil.

Vse komponente vremenske postaje so bile naročene preko internetnih trgovin dobaviteljev in niso presegle 100 €. Velikost in mobilnost vremenske postaje nam omogočata prosto prenašanje. Če želimo vremensko postajo postaviti nekam, kjer je na razpolago le električno napajanje in nimamo dostopa do interneta, nam bo ta še vedno zbirala podatke, ki jih lahko pogledamo kasneje. V primeru, da imamo na voljo računalnik, interneta pa ne, lahko aplikacijo še vedno poženemo lokalno in tako dostopamo do podatkov.

Naš projekt bi lahko tudi izboljšali in dopolnili s številnimi posodobitvami, kot na primer: svoje integrirano vezje za direkten priklop na Raspberry Pi, boljša baterija za daljši čas delovanja sistema brez elektrike, sončne celice za napajanje baterije v sončnih dneh, dodatni senzorji (za merjenje onesnaženosti zraka, svetlobe v UV spektru, števca strel, magnetnega polja, zvočnih frekvenc) in z zunanjo anteno za sprejemnik GPS.

# Literatura


[1] D. Kodek, *Arhitektura in organizacija računalniških sistemov*. Bi-tim, 2008.

[2] W. Foundation. (2013, Jul.) Image archive. [Online]. Available: http://www.wikimedia.org/

[3] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics Magazine*, p. 4, 11 2006.

[4] W. Chip, "Kryder's law," *Scientific American*, 7 2005.

[5] E. Upton *et al.* (2013, Jul.) The raspberry pi foundation. [Online]. Available: http://www.raspberrypi.org/

[6] ARM. (2013, Jul.) Dokumentacija arhitekture arm. [Online]. Available: http://infocenter.arm.com/help/index.jsp

[7] T. Ankur. (2013, Jul.) Raspberry pi documentation. [Online]. Available: http://www.element14.com/community/docs/DOC-42993/l/raspberry-pi-single-board-computer

[8] F. Inc. (2013, Jul.) Mpl115a2, miniature i2c digital barometer - data sheet. [Online]. Available: http://cache.freescale.com/files/sensors/doc/data_sheet/MPL115A2.pdf

[9] U-blox. (2013, Jul.) Max-6 data sheet. [Online]. Available: http://www.u-blox.com/images/downloads/Product_Docs/MAX-6_Data-Sheet_%28GPS.G6-HW-10106%29.pdf

[10] P. S. Foundation. (2013, Jul.) Python standard library. [Online]. Available: http://docs.python.org/2/library/

[11] R. E. Hipp. (2013, Jul.) Sqlite documentation. [Online]. Available: http://www.sqlite.org/docs.html

[12] T. CherryPy. (2013, Jul.) Cherrypy documentation. [Online]. Available: http://docs.cherrypy.org/stable/index.html

[13] N. Inc. (2013, Jul.) Nginx documentation. [Online]. Available: http://nginx.org/en/docs/

[14] W. org. (2013, Jul.) Functions of wsgi. [Online]. Available: http://wsgi.readthedocs.org/en/latest/genindex.html

[15] D. M. K. and W. Matt, *Running Linux, 5th Edition*. O'Reilly Media, 2005.

[16] M. Bayer. (2013, Jul.) Dokumentacija mako. [Online]. Available: http://docs.makotemplates.org/en/latest/

[17] A. learning system. (2013, Sep.) Dht11 drivers. [Online]. Available: http://learn.adafruit.com/dht-humidity-sensing-on-raspberry-pi-with-gdocs-logging/software-install

[18] G. Filippini. (2013, Sep.) I2c driver. [Online]. Available: hhttp://www.brainworks.it/rpi-environmental-monitoring/index.php?id=read-pressure-from-mpl115a2

[19] Google. (2013, Sep.) Implementation of google map. [Online]. Available: https://developers.google.com/maps/documentation/javascript/examples/map-simple

[20] H. S. AS. (2013, Sep.) Highchart. [Online]. Available: http://www.highcharts.com/products/highcharts

[21] Twitter. (2013, Sep.) Bootstrap usage. [Online]. Available: http://getbootstrap.com/2.3.2/

[22] T. jQuery Foundation. (2013, Sep.) jquery documentation. [Online]. Available: http://api.jquery.com/

[23] T. R. P. Foundation. (2013, Jul.) Guide to raspbian wheezy instalation. [Online]. Available: http://www.raspberrypi.org/downloads

[24] R. Ltd. (2013, Jul.) Download vnc software. [Online]. Available: http://www.realvnc.com/download/

[25] M. Simon. (2013, Jul.) Running vncserver at startup. [Online]. Available: http://learn.adafruit.com/adafruit-raspberry-pi-lesson-7-remote-control-with-vnc/running-vncserver-at-startup

[26] P. Tartarus. (2013, Jul.) Putty documentation page. [Online]. Available: http://www.chiark.greenend.org.uk/šgtatham/putty/docs.html

[27] M. Bram. (2013, Jul.) Vim documentation. [Online]. Available: http://vim.sourceforge.net/docs.php

[28] iterate GmbH. (2013, Jul.) Cyberduck homepage. [Online]. Available: http://cyberduck.ch/

# Priloge

## Priloga A: Program za zajem podatkov in osveževanje podatkovne baze

```
1  import datetime
2  import random
3  from alchemy_interface import *
4  import subprocess
5  import serial
6  import string
7  import math
8
9  #data format
10 now = datetime.datetime.now()
11 date = now.date()
12 #time = now.time()
13 gpsout = []
14
15 #branje iz UART-a
16 port = serial.Serial("/dev/ttyAMA0", baudrate=9600, timeout=5.0)
17 rcv = str(port.read(250))
18 tmpgps = string.split(rcv,"\n")
19 for i in tmpgps:
20     if "GPGGA" in i and len(i)>15:
21         gps_out = i
22 print gps_out
23
```

```
24 gps_list = gps_out.split(',')
25
26 session = Session()
27
28 #list of sensors
29 sensors =["temperature", "pressure", "moisture", "height"]
30
31 #gps coordinates
32 if len(gps_list[2]) < 2:
33     x = "99.99" #max 90 -> invalid info, send false
34     y = "99.99"
35     h = "99.99"
36     print "nogps"
37 else:
38     x = float(gps_list[2])
39     y = float(gps_list[4])
40     h = float(gps_list[9])
41     xsplit = str.split(str(x),".")
42     ysplit = str.split(str(y),".")
43     tmp = xsplit[0]
44     secondsx = float(xsplit[1]) / (10**(len(xsplit[1])))
45     secondsy = float(xsplit[1]) / (10**(len(ysplit[1])))
46     if len(xsplit[0]) > 4:
47         deg = tmp[:3]
48         minutes = tmp[3:]
49     else:
50         deg = tmp[:2]
51         minutes = tmp[2:]
52     x = float(deg) + float(minutes)/60 + float(secondsx)/3600
53     #print deg, minutes, seconds
54     #print x
55
56     #zemljepisna sirina
57     tmp = ysplit[0]
58     if len(ysplit[0]) > 4:
```

```
59          deg = tmp[:3]
60          minutes = tmp[3:]
61      else:
62          deg = tmp[:2]
63          minutes = tmp[2:]
64      y = float(deg) + float(minutes)/60 + float(secondsy)/3600
65
66      tmp = gps_list[1]
67
68  #system time
69  cajt = datetime.time((now.hour+2)% 24,
70      now.minute, now.second, now.microsecond)
71
72  ##parsanje cifer iz izhoda driverja za zaznavanje senzorjev
73  args = ['sudo', '/home/pi/Adafruit_DHT_Driver/Adafruit_DHT', '11', '4']
74
75  proc = subprocess.Popen(args, stdout=subprocess.PIPE)
76  output = proc.stdout.read()
77  counter = 0
78  tmpmeasures = []
79  for splitter in output.split(" "):
80      if counter == 10 or counter == 14:
81           tmpmeasures.append(int(splitter))
82      counter += 1
83
84
85  #for every sensor add a new line
86  for i in sensors:
87      value = round(random.randint (1000, 2000), 1)
88
89      ###################
90      if i == "height":
91          value = h
92      if i == "temperature":
93          value = tmpmeasures[0]
```

```
94      if i == "moisture":
95          value = tmpmeasures[1]
96
97
98      new_entry = Measure(date, cajt, x, y, i, value)
99
100     # Add the record to the session object
101     session.add(new_entry)
102
103
104 # commit the record the database
105 session.commit()
106 session.close()
```

# Priloga B: Spletna aplikacija

```
1  import cherrypy
2  import os.path
3  import datetime
4  from datetime import timedelta
5  from decimal import *
6  from mako.template import Template
7  from alchemy_interface import *
8  from cherrypy.lib.static import serve_file
9  import math
10 import itertools
11 import time
12
13 current_dir = os.path.dirname(os.path.abspath(__file__))
14 session = Session()
15
16 mytemplate = Template(filename = 'index.htm')
17
18 class Root(object):
```

```
19      @cherrypy.expose
20      def index(self):
21          #pretvorba iz sekund v ure
22          def sec_to_time(sec):
23
24              days = sec / 86400
25              sec -= 86400*days
26
27              hrs = sec / 3600
28              sec -= 3600*hrs
29
30              mins = sec / 60
31              sec -= 60*mins
32              m = "{0}:{1}".format(hrs,mins)
33              return m
34
35          #dict za spremenljivke, ki gredo na stran
36          l = {"sensors":{}}
37
38          #modifikacija datuma za prikaz grafa vlage
39          now = datetime.datetime.now()
40          l["date"] = now.strftime('%d/%m/%Y')
41          newnow = now + timedelta(hours=2)
42          l["time"] = newnow.strftime('%H:%M')
43          l["grafyear"] = now.year
44          l["grafmonth"] = now.month - 2
45          l["grafday"] = now.day
46
47          #izpis tabele zadnjih vnosov
48          sensor_types = session.query(distinct(Measure.sensor)).all()
49          for i in sensor_types:
50              qry = session.query(Measure).filter(Measure.sensor == i[0]).
51              order_by(Measure.id.desc()).first()
52              l["sensors"][qry.sensor] = qry.value
53              l["gps_x"] = qry.gps_x
```

```
54              l["gps_y"] = qry.gps_y
55              l["date2"] = qry.date
56              l["time2"] = qry.time
57              l["displaygps"] = 1
58              if int(qry.gps_x) == 99:
59                  l["displaygps"] = 0
60
61          #uptime
62          with open('/proc/uptime', 'r') as f:
63              uptime_seconds = float(f.readline().split()[0])
64              uptime_string = str(timedelta(seconds = uptime_seconds))
65          l["uptime"] = uptime_string
66
67          #vsi zapisi in stevilo vnosov posameznega senzorja
68          all = session.query(Measure).
69          order_by(Measure.id.desc()).first()
70          l["id_by_sensor"] = all.id / len(sensor_types)
71          l["id"] = all.id
72
73          #izpis casa prve meritve
74          fst_time = session.query(Measure).order_by
75              (Measure.id.asc()).first()
76          l["fst_time"] = datetime.datetime.combine(first_time.date,
77              first_time.time)
78
79          ###branje podatkov za izris grafa
80
81          graftemperature = []
82          grafpritiska = []
83          grafvlage = []
84          grafcasa = []
85          strcasa = []
86
87          tmpgraftemperature = []
88          tmpgrafpritiska = []
```

```
89          tmpgrafvlage = []
90          tmpgrafcasa = []
91          tmpcasvsekundah = []
92          if l["id_by_sensor"] < 12*24:
93              l["graf"] = 0
94              l["graftemperature"] = graftemperature
95              l["grafpritiska"] = grafpritiska
96              l["grafvlage"] = grafvlage
97              l["grafcasa"] = grafcasa
98          else:
99              l["graf"] = 1
100             k = 12
101
102
103         for sens2 in sensor_types:
104             qry3 = session.query(Measure).filter(
105             Measure.sensor == sens2[0]).
106             order_by(Measure.id.desc()).limit(12*24)
107             for k2 in qry3:
108                 if sens2[0] == 'temperature':
109                     x = time.strptime(str(k2.time).
110                     split('.')[0],'%H:%M:%S')
111
112                     total_sec = datetime.timedelta(hours=x.tm_hour,
113                     minutes=x.tm_min,
114                     seconds=x.tm_sec).total_seconds()
115                     tmpgrafcasa.append(int(total_sec))
116                     tmpgraftemperature.append(int(k2.value))
117
118                 if sens2[0] == 'pressure':
119                     tmpgrafpritiska.append(int(k2.value))
120
121                 if sens2[0] == 'moisture':
122                     tmpgrafvlage.append(int(k2.value))
123         temp_by_12 = itertools.imap(None,
```

```
124
125         *[iter(tmpgraftemperature)]*12)
126         pres_by_12 = itertools.imap(None, *[iter(tmpgrafpritiska)]*12)
127         moist_by_12 = itertools.imap(None, *[iter(tmpgrafvlage)]*12)
128         time_by_12 = itertools.imap(None, *[iter(tmpgrafcasa)]*12)
129
130         for x in range(24):
131             k = temp_by_12.next()
132             graftemperature.append(sum(k)/len(k))
133
134         for x in range(24):
135             k = pres_by_12.next()
136             grafpritiska.append(sum(k)/len(k))
137
138         for x in range(24):
139             k = moist_by_12.next()
140             grafvlage.append(sum(k)/len(k))
141
142         for x in range(24):
143             k = time_by_12.next()
144             grafcasa.append(sum(k)/len(k))
145
146         graftemperature = graftemperature[::-1]
147         grafpritiska = grafpritiska[::-1]
148         grafvlage = grafvlage[::-1]
149         grafcasa = grafcasa[::-1]
150
151
152         for i in range(24):
153             strtime = sec_to_time(grafcasa[i])
154             strcasa.append(strtime)
155
156         l["graftemperature"] = list(graftemperature)
157         l["grafpritiska"] = list(grafpritiska)
158         l["grafvlage"] = list(grafvlage)
```

```
159         l["grafcasa"] = strcasa
160
161
162 ######################################################
163         #branje zadnjih 5-ih za izris puscic
164
165         i = 5
166         if l["id_by_sensor"] < i:
167             i = l["id_by_sensor"]
168
169
170         ####tabele zadnjih vnosov
171
172         povtemperatura = []
173         povpritisk = []
174         povvlaga = []
175
176         #init rezultatov
177         smertemperature = 0
178         smerpritiska = 0
179         smervlage = 0
180
181
182
183         for sens in sensor_types:
184             qry2 = session.query(Measure).
185             filter(Measure.sensor == sens[0]).
186             order_by(Measure.id.desc()).limit(i)
187
188             #seznam za  graf
189             for j in qry2:
190
191                 #vnos valut v tabele
192                 if sens[0] == 'temperature':
193                     povtemperatura.append(j.value)
```

```
194
195                 if sens[0] == 'pressure':
196                     povpritisk.append(j.value)
197
198                 if sens[0] == 'moisture':
199                     povvlaga.append(j.value)
200
201         povprecnatemperatura = sum(povtemperatura)/len(povtemperatura)
202         povprecnipritisk = sum(povpritisk)/len(povpritisk)
203         povprecnavlaga = sum(povvlaga)/len(povvlaga)
204         #print povprecnatemperatura, povprecnipritisk, povprecnavlaga
205
206
207         for count in range(0,i):
208             smertemperature+=povtemperatura[count]-povprecnatemperatura
209             smerpritiska += povpritisk[count] - povprecnipritisk
210             smervlage += povvlaga[count] - povprecnavlaga
211         l["smertemperature"] = smertemperature
212         l["smerpritiska"] = smerpritiska
213         l["smervlage"] = smervlage
214
215
216
217
218     #izpis max
219         for k in sensor_types:
220             aver = session.query(func.avg(Measure.value).
221             label('average')).filter(Measure.sensor==k[0])
222             maxi = session.query(func.max(Measure.value).
223             label('maximum')).filter(Measure.sensor==k[0])
224             mini = session.query(func.min(Measure.value).
225             label('minimum')).filter(Measure.sensor==k[0])
226
227             #if k[0] == 'temperature':
228             l[k[0]+"Average"] = math.ceil(aver[0].average)
```

```
229             l[k[0]+"Max"] = math.ceil(maxi[0].maximum)
230             l[k[0]+"Min"] = math.ceil(mini[0].minimum)
231 
232             if k[0] == 'height':
233                 l["hAverage"] = aver[0].average
234                 l["hMax"] = maxi[0].maximum
235                 l["hMin"] = mini[0].minimum
236 
237 
238 
239     session.close()
240     return mytemplate.render(**l)
241 
242 #zagon serverja in nastavitve staticnih slik
243 cherrypy.config.update({'server.socket_host': '0.0.0.0' })
244 conf = {'/img': {'tools.staticdir.on': True,
245         'tools.staticdir.dir': '/home/pi/cherryws/img'}}
246 cherrypy.quickstart(Root(), '/', config=conf)
```