SVEUČILIŠE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 4507

SUSTAV ZA PRIKAZ, KRONOLOŠKU REGISTRACIJU I STATISTIČKU ANALIZU MJERNIH PODATAKA

Nikola Vrebčević

Zagreb, lipanj 2016.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA
ODBOR ZA ZAVRŠNI RAD MODULA

Zagreb, 16. ožujka 2016.

ZAVRŠNI ZADATAK br. 4507

Pristupnik: **Nikola Vrebčević (0036479059)**
Studij: Računarstvo
Modul: Računalno inženjerstvo

Zadatak: **Sustav za prikaz, kronološku registraciju i statističku analizu mjernih podataka**

Opis zadatka:

U okviru završnog rada potrebno je realizirati mrežno podržani ugradbeni računali sustav za prikaz, kronološku registraciju i statističku obradu mjernih podataka. Sustav temeljiti na ugradbenoj platformi RaspberryPi pod Linux operacijskim sustavom. Mjerni podatci generiraju se u stvarnom vremenu u namjenskoj aplikaciji koja je razvijena za konkretnu mjernu primjenu. Potrebno je definirati način razmjene mjernih podataka između mjerne aplikacije i sustava. Sustav treba omogućiti numeričku i/ili grafičku prezentaciju izmjerenih veličina u stvarnom vremenu, njihovu pohranu u bazu podataka s kronološkom oznakom mjerenja korištenjem sata stvarnog vremena, te primjere vremenski kratkotrajnih statističkih analiza izmjerenih podataka. Sustav treba omogućiti i preuzimanje pohranjenih podataka iz kronološke baze za željeno mjerno razdoblje. Upravljanje sustavom i prikaz mjernih i obrađenih podataka realizirati korištenjem dinamičke web stranice i web servera na ugradbenom računalu čime se omogućava udaljeni mrežni pristup do mjernog sustava. Kao pokaznu aplikaciju sustava demonstrirati primjenu za udaljeno mjerenje brzine i smjera vjetra u automatiziranoj meteorološkoj postaji. Za dodatne informacije obratiti se mentoru.

Zadatak uručen pristupniku: 18. ožujka 2016.
Rok za predaju rada: 17. lipnja 2016.

Mentor:
Prof. dr. sc. Davor Petrinović

Djelovođa:
Prof. dr. sc. Danko Basch

Predsjednik odbora za završni rad modula:
Prof. dr. sc. Mario Žagar

Sadržaj

1 Uvod

Na račun tvrtki koje su godinama ulagale u svoje inžinjere kako bi pratili stopu povećanja broja tranzistora na čipu opisanu Mooreovim zakonom, danas imamo priliku živjeti u vremenu gdje računala možemo modificirati za gotvo svaku potrebu koju imamo u našem životu. Zbog relativno male proizvodne cijene i velikog broja mogućnosti, današnji procesori i mikroprocesori, kontroleri i mikrokontroleri mogu poslužiti kao jedinice za mjerenje i odrađivanje podataka u stvarnom vremenu na mjestu samog događaja kojeg želimo promatrati ili neke pojave koja ima određenu važnost za korisnika. Takve mogućnosti dovele su do razvoja ugradbenih računalnih sustava, odnosno računala koja su specijalizirana za točno određenu vrstu posla.

Promatramo li meteorologiju kroz povijest, njeni korijeni sežu u antičko doba kad su se počele prvi put proučavati meteorološke pojave, a od tada pa do dan danas ona svojom nepredvidivošću, pozitivno ili negativno, fascinira sve stanovnike Zemlje. Danas je meteorologija interdiscpilinarna znanost koja proučava atmosferu na Zemlji. Budući da promatrajući atmosferu možemo zabilježiti različite vrste podataka, za tu svrhu možemo iskoristiti ugradbena računala koja će korisniku omogućiti prikupljanje, obradu i izdvajanje relevantnih podatka.

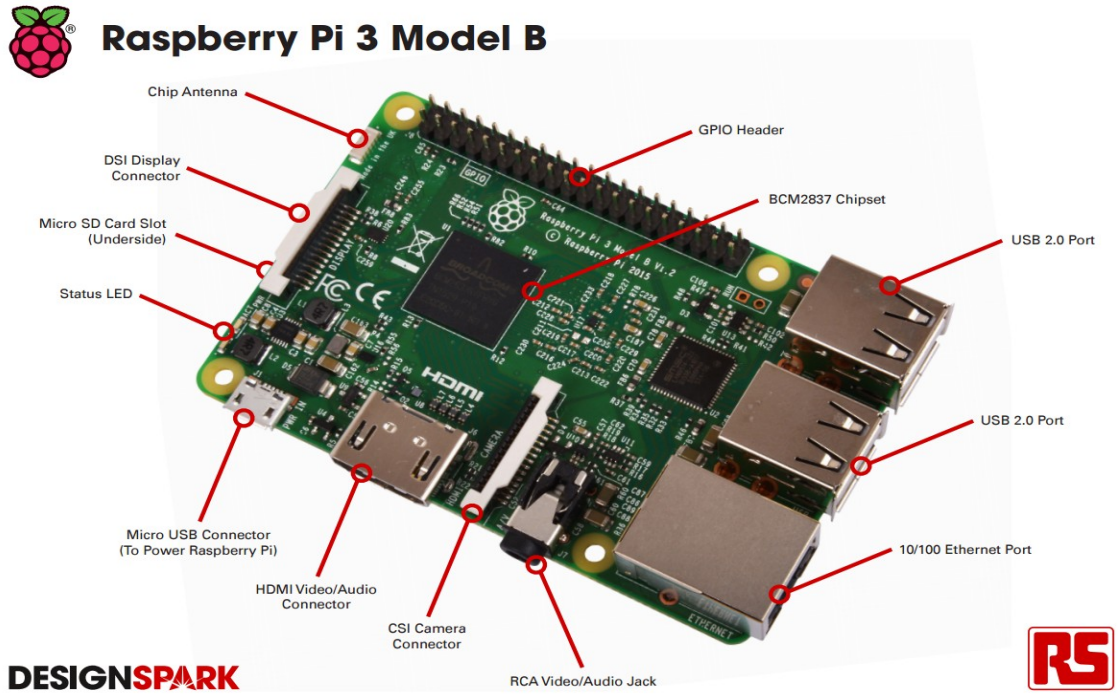
Upravo na takvoj ideji počiva ovaj završni rad u kojem je cilj pomoću ugradbnog računala Raspberry Pi 3 i prikupljenih podataka sa senzora (mjernje i prikupljanje podata nije tema ovog završnog rada) napraviti statističku obradu podataka, izdvojiti relevantna mjerenja od onih koja to nisu te prikazati ih korisniku na njemu razumnjiv način i sve to u stvarnom vremenu.

2 Sklopovska i programska podrška

Za uspješno rješavanje nekog problema pomoću računala, nepohhodno je skladno funkcioniranje sklopovlja i programa koji se izvode na sklopovlju. Pošto nisu sva sklopovlja idealana za sve programe, kao niti svi programi za sva sklopovlja, u nastavku su raspravljene tehnologije koje su korištene pri realizaciji zahtjeva ovog završnog rada.

2.1 Raspberry Pi 3

Raspberry Pi je računalo koje je razvila istoimena organizacija, *Raspberry Pi Foundation*, kako bi se čari digitalnog svijeta približile svima onima koji su za to zainteresirani na način da samostalno nauče kako to računalo primjenit za određene osobne potrebe. Uz to, *Raspberry Pi Foundation* potiče korištenje Raspberry Pi računala pri edukaciji djece koja uz pomoć nastavnika i određenih programskih modula mogu napisati programe koje pokreću na računalu te tako stječu znanja iz programiranja i dobivaju ideju kako računala funkcioniraju.



Slika 1. Raspberry Pi 3 (Preuzeto s *RS Components Ltd: Raspberry Pi 3 Model B*)

Raspberry Pi 3 je model treće generacije Rasaspberry Pi računla koji je stavljen na tržište u veljači 2016. godine. Kao što se vidi na prethodnoj slici (*Slika 1.*), računalo se sastoji od svih perifernih komponenti koje danas nalazimo u osobnim ili prijenosnim računalima: USB priključci, HDMI priključak, Ethernet priključak, RCA zajednički priključak za zvuk i sliku, DSI priključak za paralelnu vezu s prikaznikom i CSI priključak za paralelnu vezu s kamerom, izvodi opće namjene te antena. Centralna upravljačka jedinica je unaprijeđena u odnosu na prethodne modele te ima slijedeće karakteristike:

- 1.2 GHz 64-bitni četverojezgreni ARMv8 procesor
- 802.11n Wireless LAN modul za bežičnu vezu
- Bluetooth 4.1
- Bluetooth Low Energy za primjene s koje zahtjevaju nisku potrošnju

Iz navedenih podataka može se zaključiti da Raspberry Pi 3 ima zavidnu procesnu moć s obzirom na to da još uvijek spada u kategoriju ugradbenih računalnih sustava te da s takvom snagom može upravljati većim brojem senzora ili aktuatori (broj vanjskih jednicia ovisi samo o broju izvoda opće namjene) i uz sve to korisnik je u mogućnosti kontrolirati procese preko grafičkog ili mrežnog sučelja.

2.2 Raspbian

Svako računalo koje može izvoditi puno procesa, računalo za koje je potreban složeniji način upravljanja memorijom i koje ima primarnu i sekundarnu memoriju poželjno je da kao glavni program koji upravlja svim aktivnostima na računalu ima operacijski sustav. Ako je taj operacijski sustav optimiziran i namjenjen posebno za jednu porodicu računala, onda u svijetu Raspberry Pi računala govorimo o Raspbianu. Unatoč tome, postavlja se pitanje zašto treba malom račinalu kao Raspberry Pi operacijski sustav? Budući da četverojezgreni procesor koji radi na frekvenciji 1.2 GHz mora uspješno i pouzdano rukovoditi procesima koje izvodi, zamjenama stranica u priručnim memorijama, kao i u

radnoj memoriji, mora imati pouzdanu programsku podršku za takve operacije jer uspješnost programa koje korisnici izvode računalu ovisi o uspješnosti izvođenja prethodno navedenih operacija. Kako bi se izbjeglo da svaki korisnik mora samostalno pisati programsku podršku za osnovne operacije nad procesima i memorijom, pouzdanije je koristiti već gotove i testirane operacijske sustave koji su razvijani dugi niz godina. Osim pouzdanosti, operacijski sustav nudi korisniku mogućnost da se fokusira na ono što je njemu bitno, a to je programiranje računala za izvođenje željene operacije.

Raspbian je operacijski sustav otvorenog koda na bazi Debian distribucija Linux sustava optimiziran za sklopovlje Raspberry Pi računala. Razvijen je 2012. godine, međutim njegovo unapređivanje nije stalo, već se želi optimizirati za što više paketa iz Debian sustava kako bi Raspbian nudio širok spektar mogućnosti koji već sada broji oko 35000 paketa koji dolaze s instalacijom operacijskog sustava i mogu se jednostavno instalirati. Prilikom izrade ovog završnog rada koristila se Raspbian Jessie distribucija koja se odlikuje time što se nakon pokretanja računala, pokreće grafičko sučelje, a ne komandna linija, omogućuje se da se preko bilo koje Python skripte može pristupiti izvodima opće namjene bez posebne dozvole operacijskog sustava, ondosno kao običan korisnik, sustav je dodatno optimiziran za sklopovlje te sadrži LibreOffice paket i Java alate za učenje programiranja u Javi (BlueJ i Greenfoot).

2.3 Apache HTTP Server

Apache HTTP Server je programska podrška otvorenog koda koja pruža korisnicima višenamjenski i pouzdan HTTP poslužitelj koji se može primjeniti u širokom spektru zahtjeva, od malih privatnih poslužitelja preko komercijalnih poslužitelja do poslužitelja koji sudjeluju u razmjeni podataka pri nekom istraživanju. Poslužitelj je razvijen iz potrebe da se standardizira osnovna funkcionalnost poslužitelja te na taj način oslobodi inženjere razvoja osobnih

poslužitelja, što znači da inženjeri mogu više vremena utrošiti na dizajn sustava, a ne trebaju puno brinuti o osnovnoj funkcionalnosti elemenata njihovih sustava. Budući da je zadatak ovog završnog rada omogućiti prikaz podataka te budući da se u suštini radi o maloj meteorološkoj postaji koja nije u blizini korisnika, sustav će se postaviti u mrežu, a pomoću Apache HTTP poslužitelja će posluživati internet stranicu na kojoj će korisnik moći analizirai i preuzimati podatke.

2.4 Python, PHP, JavaScript

Python je viši programski jezi koji podržava objektno orijentiranu paradigmu, skriptno programiranje te funkcijsko programiranje koje podržava razne matematičke funkcije i prikazivanje rezultata u grafičkom obliku. Osim što je moćan programski jezik, Python ima skup biblioteka koje izvođenje operacija, poput komunikacije sa periferijom preko izvoda opće namjene na Raspberry Pi računalu čine jednostavnim i raznumljivim za korisnika. Uz to što su sklopovlje i Python u simbiozi, veza između Pythona i Raspbiana je također jako dobra, iz razloga što je Raspbian implementiran u Pythonu. Zbog toga Python programi i skripte mogu jednostavno koristiti operacijski sustav kao partnera u izvođenju zadataka, a ne da operacijski sustav predstavlja ograničenje u oblikovanju programske podrške.

PHP je skriptni jezik koji se izvodi na poslužiteljskoj strani. Može se upotrebljavati uz HTML kod za izradu internetskih stranica, ali i za izradu samostalnih skripti koje se mogu izvoditi na zahtjev klijenata ili mogu generirati odgovor na pobudu izazvanu određenim događajem. Zbog svoje interoperabilnosti i prenosivosti, PHP je jedan od najpoznatijih progamskih jezika, dok u svijetu poslužitelja predstavlja glavni programski jezik pomoću kojeg se mogu oblikovati sve funkcionalnosti potrebne na poslužiteljima. Kao većina poslužitelja, tako i Apache HTTP poslužitelj omogućuje korištenje PHP jezika. Zbog sintkse slične kao u C/C++ jeziku, to je osigurao brzo savladavanje sintakse PHPa te relativno brz i učinkovit razvoj funkcionalnosti poslužitelja.

JavaScript je jezik koji uz HTML i CSS čini osnovnu tehnologiju za izdradu internetskih stranica. Iako njegova sintaksa nije intuitivna kao što je slučaj kod Pythona ili PHPa, JavaScript pruža klijentskoj strani mogućnost komunikacije sa poslužiteljem bez ponovnog slanja zahtjeva za cijelom stranicom. Takvi zahtjevi oslobađaju mrežu od prijenosa redundantnih podataka te omogućavaju ažuriranje sadržaja stranice samo na mjestima gdje je to potrebno. Uz procese komunikacije, JavaScript je tehnologija koja je pogodna za prikazivanje podataka u obliku dijagrama kojima se može dodati dinamičko ažuriranje te na taj način korisniku prikazivati podatke u stvarnom vremenu. Konačno, podršku za JavaScript ostvaruje svaki moderniji pretraživač interneta i na taj način dizajner internetske stranice ne mora brinuti o interoperabilnosti svojeg koda.

2.5 Baza podataka RRDtool

RRDtool je alat za praćenje zapisa podataka te omogućava njihov grafički prikaz. Razvila se kao ideja da prati podatke koji se mijenjaju u vremenu (propusnost mreže, temperatura procesora, opterećenost procesora) kako bi se mogao generirati grafički prikaz koji bi se kasnije analizirao. Strukturirana je kao ciklički spremnik kojem je moguće preodrediti dimenzije. To znači da će se podaci koji su stariji od vremena kojeg je korisnik definirao automatski pobrisati iz baze što će omogućiti upis novih podataka u bazu. Baza može imati više razina čuvanja podataka, odnosno po isteku određenog vremena podaci se prebacuju u drugu bazu, međutim, opet nakon isteka određenog vremena će se podaci pobrisati. Zbog toga što je jedan od zahtjeva sustava prikaz podataka u grafičkom ili numeričkom obliku te budući da se radi o mjerenju koje se izvodi konstantno, nije moguće prikazati sva mjerenja, zato je ova baza dobra struktura za pohranu podataka za potrebe ovog završnog rada.

3 Pokretanje i uhodavanje sustava

Sadržaj idućih potpoglavlja bit će opisani načini kako su se prethodno navedene tehnologije ukomponirale zajedno te koje su sve predradnje trebale biti ispunjene da bi sustav funkcionirao kako je od njega očekivano.

3.1 Pokretanje operacjiskog sustava

Kao što je prethodno navedeno, operacijski sustav Raspbian je distribucija Linux operacijskih sustava koja je posebno optimirana za pokretanja na sklopovlju Raspberry Pi računala, a verzija Raspbiana koja se koristila pri izradi ovog završnog rada je Raspbian Jessie. Operacijski sustav na Raspberry Pi računalu se može pokretiti preko sekundarne memorije koja je ostvarena pomoću *microSD* memorijske kartice (8GB, 16GB ili više). Uz operacijski sustav, preporučljivo je priključiti na Raspberry Pi monitor, tipkovnicu i miš kako bi korisnik mogao ostavriti komunikaciju s računalom.

Za uspješnu instalaciju i pokretanje ovog operacijskog sustava najbitnije je pravilno zapisati binarnu datoteku (eng. *image*) na *microSD* karticu koja sadrži Raspbian Jessie te omogućiti da se operacijski sustav pokreće s kartice. Za tu operaciju u ovom završnom radu koristio se program *Win32 Disk Imager* koji je besplatno dostupan na internetu i koristi se bez licence. Korištenje *Win32 Disk Imager* programa je jednostavno i intuitivno. Nakon preuzimanja distribucije Raspbian Jessie sa službenih internetskih stranica *Raspberry Pi Foundaton*, potrebno je umetnuti *microSD* karticu u osobno računalo te učitati preuzetu datoteku u *Win32 Disk Imager* i odabrati *microSD* karticu po imenu pristupne točke u datotečnom sustavu (eng. *mount point*) i pokrenuti proces pisanja na karticu. Po završetku procesa, karticu je potrebno prebaciti u Raspberry Pi te uključiti napajanje i nakon učitavanja operacijskog sustava na ekranu bi se trebalo pojaviti

grafičko sučelje pomoću kojeg korisnik jednostavno može koristiti računalo. Nakon uhodavanja sustava, preporučeno je da korisnik proširi glavnu particiju (*root*) jer prilikom zapisivanja Raspbiana na karticu sustav je zauzeo onoliko memorije koliko je potrebno samo za sustav. Proširivanje particije moguće je ostvariti pomoću programa *GPartited* u kojem je potrebno desnim klikom pokazivača kliknuti na particiju koju korisnik želi proširiti te odabrati opciju *Resize/Move*. Odabirom te opcije otvorit će se prozor u kojem treba upisati novu željenu veličinu particije. Za izvršavanje postupka proširivanja particije, korisnik mora u alatnoj traci odabrati opciju za izvođenje proširivanja te ponovno pokrenuti računalo.

3.2 Konfiguracija i pokretanje procesa poslužitelja

Jednostavani poslužitelj koji će posluživati internetsku stranicu i omogućiti preuzimanje vrijednosti mjerenja u obliku datoteke, omogućit će program Apache HTTP poslužitelj. Paket Apache HTTP poslužitelja nije prethodno instaliran s operacijskim sustavom. Zbog toga potrebno je instalirati paket. Preduvjet za uspješno instaliranje paketa je pouzdana veza s internetom. Na Raspberry Piu, veza za internet može se ostvariti žično pomoću mrežnog kabela ili pomoću bežične veze. Nakon upsostavljene veze, korisnik mora u komandnu liniju (*Terminal*) upisati naredbu za dohvaćanje i instaliranje paketa:

```
sudo apt-get install apache2 php5 libapache2-mod-php5
```

U prethodnoj naredbi, uz Apache HTTP poslužitelj, zatražila se instalacija paketa za PHP te paketa koji omogućuju poslužitelju izvođenje PHP skripti. Nakon završene instalaicje, u početnom direktoriju datotečnog sustava (“/”) kreiran je novi direktorij *var* te poddirektorij *www* iz kojeg Apache poslužitelj dohvaća resurse na početku komunikacije s klijentom. Korisniku se preporuča da unutar *www* direktorija napravi novi poddirektorij u kojem će se nalaziti osnovni resursi poput internetskih stranica, CSS datoteka, skripte koje ostvaruju određenu funkcionalnost.

Idući korak je konfiguracija Apache HTTP poslužitelja. Unutar poddirektorija */etc/apache2* nalaze se konfiguracijske datoteke u kojima je potrebno napraviti nekoliko izmjena kako bi omogućili ispravno pokretanje poslužitelja. Unutar datoteke *apache2.conf* potrebno je dodati novi blok s oznakama *<Directory></Directory>* te unutar tog bloka potrebno je označiti poslužitelju da se unutar novog direktorija koji je korisnik kreirao u */var/www* nalaze resursi:

```
<Directory /var/www/windhost/>
        Options Indexes FollowSymLinks
        AllowOverride None
        Require all granted
</Directory>
```

Nužno je datoteku otvoriti preko komandne linije te pristupati datoteci kao administrator (*root user*). Datoteka se može urediti iz uređivača teksta koji je već instaliran na operacijskom sustavu:

```
sudo gedit /etc/apache2/apache2.conf
```

Ili preko programa *Vim* za uređivanje teksta preko komandne linije kojeg je potrebno prethodno instalirati naredbom:

```
sudo apt-get install vim
sudo vim /etc/apache2/apache2.conf
```

Za uređivanje teksta pomoću *Vim* uređivača, potrebno je znati naredbe za uređivanje.

Završni korak konfiguracije poslužitelja je dodavanje nove konfiguracijske datoteke u direktorij */etc/apache2/sites-available* koja određuje poslužitelju kako dohvatiti početne resurse (npr. početnu stranicu za internetski izvor). Korisno je konfiguracijskoj datoteci dodijeliti isto ime koje ima direktorij na lokaciju */var/www* u kojoj se nalaze resursi koji se poslužuju. Sadržaj konfiguracijske datoteke *windhost.conf* je:

```
<VirtualHost *:80>
        ServerAdmin webmaster@windhost
        ServerName windhost
        ServerAlias windhost.fer
        DirectoryIndex index.html
        DocumentRoot /var/www/windhost
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Prethonda konfiguracija određuje ime datoteke koju će poslužitelj poslati korisniku kao resurs prilikom ostaviranja HTTP veze, u ovom slučaju to je datoteka *index.html*. Dodatno, poslužitelju se definira direktorij (*DocumentRoot*) u kojem se nalazi datoteka *index.html*. Nakon konfiguracije potrebno je omogućiti posluživanje zadanih datoteka pomoću naredbe u komandnoj linij:

```
sudo a2ensite windhost.conf
```

Po završetku upsješnog izvođenja naredbe, potrebno je ponovno pokrenuti poslužitelj kako bi mogao učitati novu konfiguraciju:

```
sudo service apache2 restart
```

Dodatno, naredba za pokreta je Apache HTTP poslužitelja je:

```
sudo service apache2 start
```

Odnosno za zaustavljanje procesa poslužitelja:

```
sudo service apache2 stop
```

3.3 Konfiguracija RRDtool baze podataka

Svaka baza podataka sastoji se od skupina podataka koji su u određenoj logičkoj vezi. Međutim, ovaj sustav nema potrebe za većim brojem struktura i tablica u bazi jer prati trenutak uzorkovanja, srednju brzinu vjetra, brzinu udara vjetra, srednji smjer vjetra i standardnu devijaciju tog smjera. Za te potrebe jednostavno je kreirati RRDtool bazu podataka sljedećim naredbama koje se unose u komandnu liniju:

```
rrdtool create data.rrd \
--step 60 \
DS:time:GAUGE:120:U:U \
DS:speed:GAUGE:120:0:U \
DS:speed_gust:GAUGE:120:0:U \
DS:direction:GAUGE:120:0:360 \
DS:direction_gust:GAUGE:120:0:360 \
RRA:LAST:0.5:1:60
```

Nakon naredbe *rrdtool create* dolazi naziv baze podataka. Iduće svojstvo *step* označava koliko često korisnik želi puni bazu podataka, a izražava se u sekundama. Oznake *DS* označavaju jedan stupac u strukturi tablice, odnosno jednu vrstu podatka. Svakom podatku pridodaje se svojstvo *GAUGE* koje označava da se podatak spremi u obliku kakvom je. Iduće svojstvo je sigurnosni interval vremena u kojem baza očekuje upis u varijablu, a inače stavlja da varijabla ima nedefiniranu vrijednost.. Iduća dva svojstva opisuju minimalnu vrijednost, odnosno maksimalnu vrijednost koju varijabla može poprimiti. Ako je definirano 'U', tada varijabla poprima billo koje vrijednosti. Konačno se definira baza pomoću oznake *RRA* koja sprema zadnji upis (*LAST*) određeni broj puta u određenom vremenu. Za potrebe testiranja, baza je postavljena da se u nju upisuje jednom svaku minutu, a varijable mogu čekati do dvije minute za promjenu što je postavljeno kao sigurnosi interval.

3.4 Pristup s udaljenog računala

Kako bi se olakšalo upravljanje računalom koje se nalazi na nekoj lokaciji koja administratoru računala nije blizu ili put do računala predstavlja ekonomski trošak koji dovodi u pitanje isplativost upotrebe računala za određenu funkciju ili je administrator u nemogućnosti doći do računala zbog osobnih nemogućnosti ili fizaklnih prepreka, postoje protokoli komunikacije između dva računala koja su spojena u mrežu, koji omogućuju udaljeno administriranje određenog računala. Jedan od protokola je *telnet*, međutim, zbog rizika od preusmeravanja ili krađe podataka na internetu koji danas postoji, sigurnije je upotrijebiti protokol *SSH* koji

omogućuje enkriptiranu komunikaciju između dva računala.

Komunikacija između Raspberry Pi računala i računala preko kojeg administrator želi pristupiti Raspberry Piju može se ostvariti uz pomoć programa *PuTTY*. Programu se zada IP adresa Raspberry Pija, odabere se SSH protokol te se pokrene komunikacija. Nakon ostvarne veze potrebno je prijaviti se pomoću korisničkog imena i lozinke korisnika preko kojeg želimo pristupati Raspberry Piju. Nakon uspješne prijave, administrator može pomoću naredbenog retka obaviti sve operacije na udaljenom računalu kao da se nalazi u njegovoj neposrednoj blizini. Osim zadavanja naredbi, pomoću SSH protokola moguće je prenositi datoteke između dva povezana računala. Ukoliko je računalo koje koristi administrator za povezivanje na Raspberry Pi također Linux računalo, dovoljno je da administrator u naredbeni redak svojeg računala unese naredbu *scp* koja kao argumente prima datoteke ili direktorije koji želimo kopirati te IP adresu računala i lokaciju u datotečnom sustavu na koju želimo spremiti datoteke ili direktorije. Ukoliko je računalo koje koristi admnistrator Windows računalo, postoji poseban program *WinSCP* koji omogućuje prijenos datoteka i direktorija između ta dva računala. Opisani programi, protokoli i naredbe omogućuju administratorima potpunu kontrolu nad računalom iako je ono na nekoj udaljenoj lokaciji. Uz to, omogućeno je unaprijeđivanje programa koji se izvode na računalu kao njegova tražena funkcionalnost te dodavanje novih funkcionalnosti.

4 Programska podrška

Zadatak ovog završnog rada je dizajnirati i implementirati programsku podršku koja ima zadatak izvršiti statističku analizu nad mejerenim podacima koji se generiraju u namjenskoj aplikaciji. Programska podrška obavllja kronološku registraciju nad analiziranim podacima te ih pohranjuje u bazu podataka. Konačno, sustav omogućava prikaz podataka preko internetske stranice koju sam poslužuje.

4.1 Robusne statistike

Ukoliko želimo obrađivati mjerenja koja ne podliježu normalnim razdiobama, nužno je primjeniti tehnike računanja statističkih elemenata pomoću robusnih statistika. Podaci poput brzine ili smjera vjetra nisu jednoliko raspoređeni oko neke srednje vrijednosti te bi računanje srednje brzine vjetra ili smjera vjetra dali pogrešnu srednju vrijednost, a standardna devijacija bi također bila lažna. Zbog tih činjenica u nastavku će biti opisane statistike i metode kako bi se iz mjerenih podataka moglo pouzdano odrediti srednju vrijednost i standardnu devijaciju.

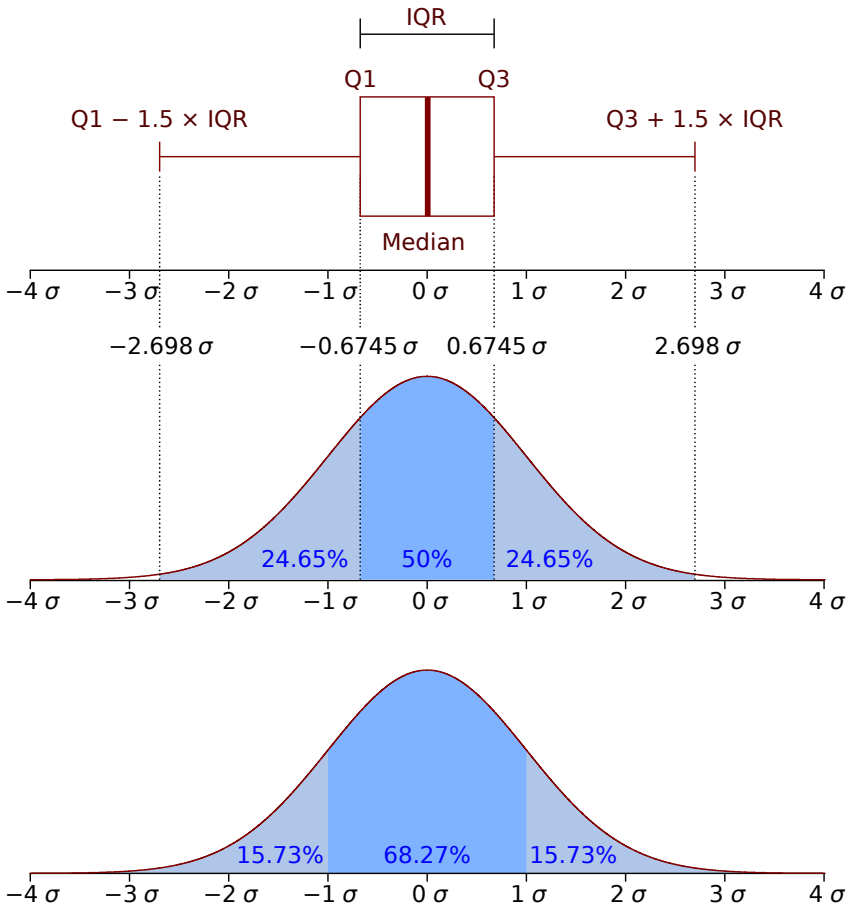
Interkvartilni raspon

Interkvartilni raspon (eng. *interquartile range*, *middle fifty*) je mjera za disperziju slučajnog procesa, a određuje se kao razlika trećeg i prvog kvartila kako je opisano jednadžbom (1).

$$IQR = Q_3 - Q_1 \tag{1}$$

Kvartili su vrijednosti mjerenja koje se nalaze na svakoj četvrtini danog raspona, uzlazno sortiranog niza po vrijednostima mjerenja, a na polovici raspona nalazi se srednja vrijednost (eng. *median*, *mean*) kao jedna od mogućih estimacija

slučajnog procesa. Interkvartilni raspon ima mjeru robusnosti 25% što znači da na nekom velikom rasponu mjerenja, prije nego što ova statistika počne davati pogrešne procjne, dopušta da 25% mjerenja budu pogrešna. Standardan devijacija normalne razdiobe povezana je s ovom statistikom preko faktora 1.349 na način da IQR iz (1) podijelimo s faktorom 1.349.



Slika 2. Usporedba normalne razdiobe i interkvartilnog raspona (Preuzeto s *Wikipedia: Jhguch, Interquartile range)*

Osim što je interkvartilni raspon u određenoj mjeri otporan na pogrešna mjerenja, pomoću njega se mogu aproksimirati razdiobe neke slučajne varijable. Na prethodnoj slici (*Slika 2.*) može se uočiti kako je normalna razdioba (donji graf) skoro u potpunosti aproksimirana interkvartilnim rasponom (prva dva grafa). Implementacija statistike takvih karakteristika za analizu podataka osigurava pouzdane procjene statističkih elementa te otpornost slučajnog procesa na pogrešne izmjerene podatke.

Srednja vrijednost odstupanja

Srednja vrijednst odstupanja (eng. m*edian absolute deviation, MAD*) predstavlja robusnu statistiku koja mjeri disperziju slučajnog procesa pomoću srednje vrijednosti svih odstupanja od estimacije očekivanja nekog mjerenja. Za estimaciju očekivanja može se koristiti median kao robusna estimacija očekivanja, kao što je ranije opisano.

$$MAD = median(|X_i - \overline{X}|) \quad (2)$$

Pri procjenjivanju standardne decvijacije pomoću MAD faktora iz jednadžbe (2) koristila su se mjerenja koja ne ulaze u raspon vrijednosti između prvog i trećeg kvartila te veza između standardne devijacije i MAD faktora ostvarena je pomoću faktora 1.483. Množenjem MAD faktora iz jednadžbe (2) s faktorom 1.483, dobit ćemo kvalitetnu procjenu standardne devijacije s obzirom da se analiziraju podaci koji se ne mogu izravno aproksimirati normalnom razdiobom.

Programska implementacija robusnih statistika

U nastavku slijedi programska implementacija prethodno opisanih robusnih statistika u Pythonu te obajšnjenje implementacije nakon koda.

```
# sigmaEstimation method estimates standard deviation
with IQR factor and MAD factor.
# Result is robust standard deviation.
# Returns median and robust standard deviation
def sigmaEstimation(data, sort_key):
    # q1 and q2 are positions of data in sorted list used
in sigma estimation
    q1 = int(0.25 * len(data))
q2 = int(0.75 * len(data))
    # Sorting input data from lowest value to highest
    sorted_data = sorted(data, key=sort_key)
    # Mean value of data is middle element in sorted list
    median = sorted_data[int(0.5 * len(sorted_data))]
```

```
        # Estimatig sigma with inter quartile range
        iqr_sig = (sorted_data[q2] - sorted_data[q1]) / 1.349
        # calculating ranges to extract data from input data
    list which are not in range [q1, q2]
        ranges = [(0, q1 + 1), (q2, len(sorted_data) + 1)]
        # generating list from ranges for MAD estimation
        mad_list = [abs(num - median) for num in sorted_data]
        # Sorting data for mad estimation
        mad_list = sorted(mad_list, key=float)
        # Estimating sigma with MAD
        mad_sig = mad_list[int(0.5 * len(mad_list))] * 1.483
        # Robust sigma si mean value of iqr and mad
        rob = float((iqr_sig + mad_sig) / 2)
        return [median, rob]
```

Funkcija *sigmaEstimation* prima polje podataka *data* te ključ sortiranja *sort_key* tog polja kao argumente funkcije. Na početku funkcije definijraju se varijable *q1* i *q2* koje predstavljaju prvi i treći kvartil polja *data.* Varijable sadrže oznaku elementa polja (eng. *index*) na jednoj četvrtini duljine polja, odnosno na tri četvrtine duljine polja. Iduća naredba je uzlazno sortiranje podataka u polju *data* po vrijednosti brojeva s pomičnim zarezom zapisanih u tom polju. Vrijednost elementa koji se nalazi na polovici sortirane liste predstavlja estimaciju očekivanja podataka zapisanih u polju *data* te se čuva u posebnoj varijabli *median*. Nakon što je određeno očekivanje slučajnog procesa, potrebno je odrediti standardnu devijaciju. Varijabla *irq_sig* čuva vrijednost aproksimacije standardne devijacije koja je izračunata na prethodno objašnjen način, oduzimanjem vrijednosti elmenta polja na mjestu *q2* od one vrijednosti koja se nalazi na mjestu *q1* te dijeljenje razlike s 1.349. U novo polje, *mad_list*, sprema se apsolutna razlika mjerenja i očekivanja slučajnog procesa. To polje se također sortira po uzlaznim vrijednostima decimalnih brojeva zapisanih u tom polju i pristupa se procjenom standardne devijacije pomoću srednje vrijednosti odstupanja. Standardna devijacija predstavlja element koji se nalazi na polovici sortirane liste *mad_list* te množenje s faktorom 1.483. Za potpunu sigurnost, računa se srednja vrijednost između procjene standardnih devijacija određenih pomoću interkvartilnog raspona i srednje vrijednosti odstupanja kako bi robusna aproksimacija standardne devijacije bila bolja. Funkcija na kraju vraća srednju vrijednost i standardnu devijaciju podataka koji su poslani funkciji.

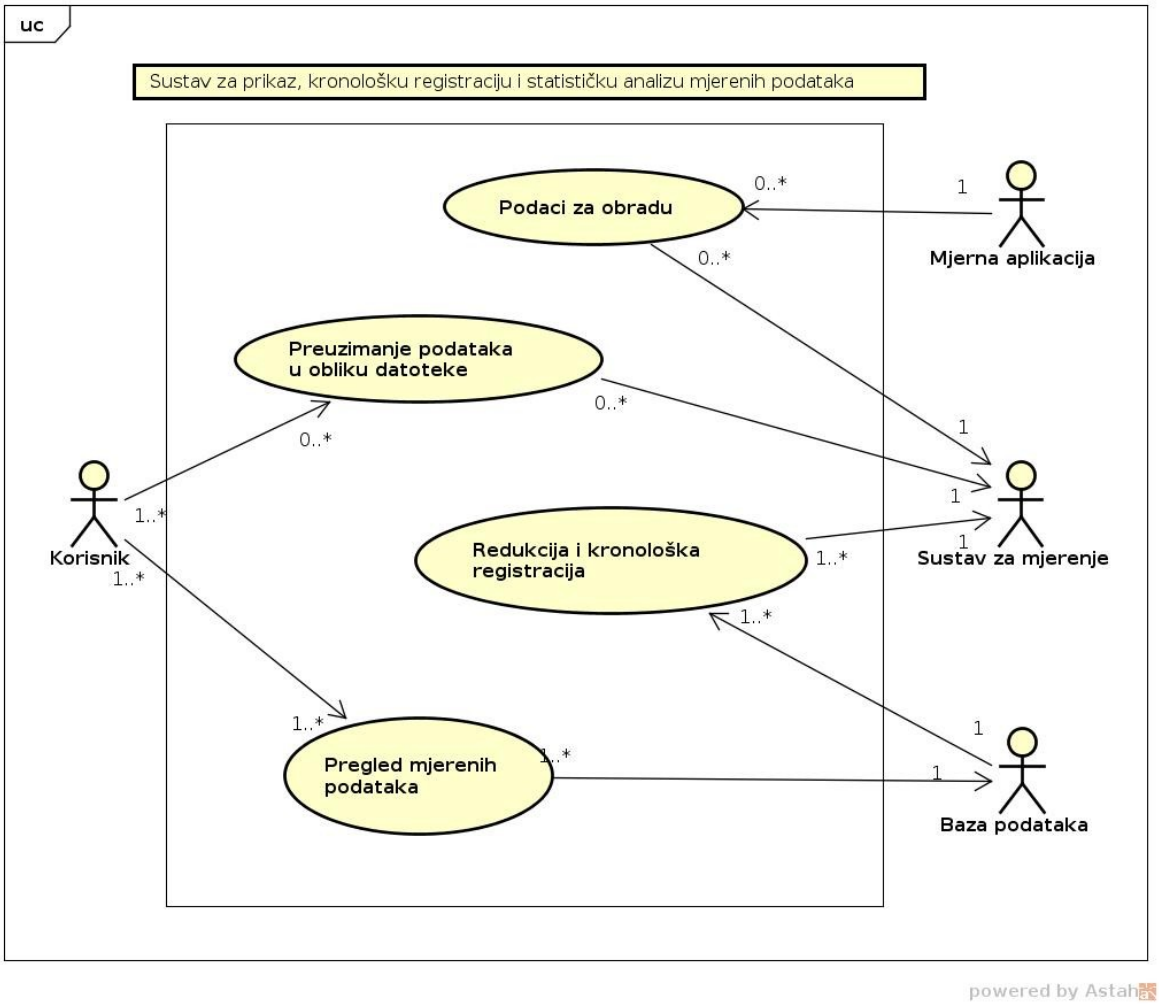
Za interval od 525 realnih brojeva u domeni od 1 do 3 koji predstavljaju brzinu vjetra te isti broj podataka u intervalu od 60 do 80 stupnjeva, statistički izračun programa je sljedeći:

```
mean: 1.73264039739
rob: 0.869950924639
----------------
direction mean: 67.0693391867
direction sigm: 10.3508831825
Trenutna brzina vjetra je 1.732640 km/h.
Trenutni smjer vjetra je 67.0693391867 te varira +/-
10.3508831825
```

Dobiveni rezultat je u skladu s očekivanjima jer aplikacija koja je u svrhu testiranja generirala ove brojeve radi to na način da izračuna brojeve iz određene linearne kombinacije sinusa različitih amplituda, faza i frekvencija što nije simulacija stvarnosti.

4.2 Osnovna funkcionalnost sustava

Osnovnu funkcionalnost sustava predstavaljaju procesi koji izvode programe u kojima su implementirani prethodno navedeni zahtjevi. Osim samog izvođenja programa, za pouzdno funkcioniranje sustava potrebno je uskladiti određene procese. Usklađivanje procesa omogućeno je uz pomoć operacijskog sustava. Svaki operacijski sustav ima određeni broj funkcija pomoću kojih korisnik može iz svog programa upravljati procesom tog programa ili nekim drugim procesom koji se izvodi ili komunicirati s ostalim procesima.



Dijagram 1. Dijagram obrazaca uporabe

Na prikazu obrazaca uporabe (*Dijagram 1.*) nalazi se osnovni scenariji iz

perspektive korisnika koji se predviđaju korištenjem ovog sustava. Dijagram prikazje nekoliko sudionika sustava (aktori). To su jedan ili više korisnik, baza podataka, mjerna aplikacija te sustav za prikza, kronološku registraciju i statističku analizu mjerenih podataka. Sustav konstantno prikuplja mjerenja od mjerne aplikacije te ih obrađuje i pohranjuje odrađene podatke u bazu podataka. Korisnik pregledava analizirane podatke na internetskoj stranici koju sustav poslužuje. Dodatno, korisnik može preuzeti podatke u obliku datoteke. U nastavku bit će prikazane implementacije prethodno navedenih obrazaca uporabe.

Sustav i mjerna aplikacija pisani su u dva različita jezika (već je poznato da je sustav pisan u Pythonu, a mjerna aplikacija pisana je u C jeziku). Unatoč tome, nakon pokretanja mjerne aplikacije i sustava, moguća je komunikacija između ta dva procesa. Implementacija komunikacije s mjernom aplikacijom:

```
# Global variables for interrupet hanlder for signal
SIGUSR1
speedp = []
directionp = []
timep = []
# Method is interrupt handler for signal SIGUSR1
def open_file(signum, stack):
   while True:
      try:
         podaci = open('podaci.txt', 'r')
         flock(podaci, LOCK_EX | LOCK_NB)
         # Reads file line by line and extracts
relevant data
         for item in podaci:
            line = item.rstrip('\n').split(';')
            timep.append(float(line[0]))
            speedp.append(float(line[1]))
            directionp.append(float(line[2]))
         flock(podaci, LOCK_UN)
         podaci.close()
         break
      except IOError:
         print "Neuspjelo otvaranje datoteke"
         continue

if __name__ == '__main__':
    signal(SIGUSR1, open_file
```

```
start = time()
# Receiving data for 10min = 600s.
while (time() - start) < 600:
    print str(time() - start)
    pause()     #Waiting for SIGUSR1 signal
```

Komunikacija između procesa ostvarena je pomoću tekstualne datoteke u kojoj se nalaze podaci, programskog prekida *SIGUSR1* i zaključavanja datoteke pomoću operacijskog sustava. Sustav u glavnom dijelu skripte (*main*) prikuplja podatke deset minuta. To je meteorološki stnadard za prikupljanje podatka o smjeru i brzini vjetra kako bi procjena bila što bolja i otpornija na pogrešna mjerenja. Kada mjerna aplikacija postavi prekid *SIGUSR1*, sustav izvodi prekidnu rutinu *open_file* u kojoj pristupa datoteci te čita podatke iz nje. Na početku prekidne rutine pokušava se pristupiti datoteci s podacima, ukoliko je datoteka zauzeta, čeka se dok se ne oslobodi. Nakon uspješnog pristupanja datoteci, sustav traži od operacijskog sustava da zaključa pristup datoteci pomoću funkcije *flock*. Zatim se čitaju podaci i odvajaju u globale liste ovisno o tome radi li se o vremenskoj oznaci, brzini ili smjeru. Na kraju sustav traži od operacijskog sustava da oslobodi datoteku te ju zatvara.

Idući korak je obrada mjerenja brzine i smjera vjetra. Za svaku vrstu obrade postoji poseban razred kojeg izvodi po jedan proces koji kreira glavna skripta. Za tu potrebu, ponovno se treba omogućiti komunikacija između procesa koji izvodi glavnu skriptu te procesa koji izvode obrade podataka. Između dva Python procesa najjednostavnije je koristi red (*queue*) kao skrukturu podataka preko koje mogu dva procesa komunicirati:

```
if __name__ == '__main__':
    speed_q = Queue()
    speed_lock = Lock()
    dir_q = Queue()
    dir_lock = Lock()
```

U glavnoj skripti definirani su redovi za podatke o brzini (*speed_q*) i smjeru (*dir_q*) s pripadnim strukturama podataka za zaključavanje pristupa redu (*speed_lock*, *dir_lock*) kako bi se izbjeglo čitanje i pisanje u isti red, odnosno gubitak

informacije.

```
speed_lock.acquire()
speed_q.put(speedp)
speed_lock.release()
dir_lock.acquire()
dir_q.put(directionp)
dir_lock.release()
speed_inst = Speed(speed_q, speed_lock)
direction_inst = Direction(dir_q, dir_lock)
speed_processor = Process(target=speed_inst.main())
direction_processor =
     Process(target=direction_inst.main())
speed_processor.start()
direction_processor.start()
```

Nakon čitanja datoteke i izdvajanja podataka o brzini i smjeru vjetra potrebno je te podatke smjestiti u redove iz kojih će procesi koji će obrađivati podatke dohvaćati ulazne podatke. Pri stavljanju određenih podataka potrebno je zatražiti zaključavanje reda (*speed_lock.acquire()*), odnosno ako je red zaklkjučan tada se čeka dok se ne otključa. Nakon zaključavanja, u red se pomoću naredbe *put* stavlja podatak. Redovi se pune na principu *FIFO* (*first in, first out*). Onaj podatak koji se prvi stavio u red, taj podatak će se prvi čitati iz reda. Nakon stavljanja podataka u red, potrebno je osloboditi resurse reda (otključati) za druge procese. Nakon što su se u redove stavili potrebni podaci, kreiraju se primjerci razreda *Speed* i *Direction* koji primaju pokazivače na pripadne redove i ključeve. Početak obrade podataka počinje s kreiranje procesa za svaki primjerak razreda pomoću naredbe *Process* kojoj se prosljeđuje funkcija koju je potrebno pokrenuti u novom procesu. Procesi se pokreću naredbom *start*.

Prilikom pokretanja procesa za primjerak razreda *Speed*, počne se izvršavti sljedeći kod:

```
# main method for class Speed
   def main(self):
      statistics = sigmaEstimation(self.data_speed,
         float)

      # this condition checks if the wind has relevant
mean.
```

```
        # EXPLANATION: wind mean that is too low cannot be
    treated as wind and it can be estimated to 0
        if(abs(statistics[0]) > 0.5):
            if (statistics[1] > 0):
                self.reduction(statistics[0] - (2.5 *
                  statistics[1]))
            statistics = sigmaEstimation(self.data_speed,
                 float)
        else:
            statistics[0] = 0
        if (statistics[1] > 0):
            gust = self.windGust(statistics[0] + (2 *
                 statistics[1]))
        else:
            gust = 0
        if (gust):
            self.lock.acquire()
    # Putting speed mean in queue
            self.speed_q.put(statistics[0])
    # Putting gust flag in queue
            self.speed_q.put(1)
    # Putting speed gust in queue
            self.speed_q.put(gust)
            self.lock.release()
        else:
            self.lock.acquire()
    # Putting speed mean in queue
            self.speed_q.put(statistics[0])
    # Putting gust flag in queue
            self.speed_q.put(0)
            self.lock.release()
```

Pri kreiranju primjerka razreda, u njegovom konstruktoru, preuzeli su se podaci iz priradnog reda. U glavnoj metodi razreda poziva se funkcija *sigmaEstimation* koja je prethodno objašnjena, a povratna vrijednost sadrži srednju vrijednost prosljeđenih podataka te odsupanje. Ukoliko je srednja vrijednost jako mala, može se reći da u tom slučaju ne puše vjetar te je njegova srednja vrijednost nula. U suprotnom slučaju pristupa se eliminiranju podataka koji su manji od vrijednosti koja se prosljeđuje metodi za redukciju podataka *reduction*. Ovaj korak omogućuje da se u procjenu srednje vrijednosti uključe samo ona mjerenja koja su relevantna. Nakon elimiranja nerelevantnih podataka, ponovno se pristupa računanju srednje vrijednosti i standardne devijacije preostalih podataka. Osim procjene srednje vrijednosti, ovaj razred ima zadaću izračunati koliko su jaki

udari vjetra ukoliko ih ima. Gledajući postoji li određena standardna devijacija, proces pristupa izvođenju metode *windGust* sljedećeg sadržaja:

```
    # windGust method determines if there is gust wind.
    # Returns gust speed or 0 if there is no gust
        def windGust(self, limit):
            counter = 0
            gust = []
            for i in range(0, len(self.data_speed)):
                if(self.data_speed[i] >= limit):
                    counter += 1
                    gust.append(self.data_speed[i])
                else:
                    continue
    # condition checks if there is gust
            if(counter >= self.measurements_number):
                gust = sorted(gust, key=float)
                return gust[int(0.5 * len(gust))]
            else:
                return 0
```

Metoda određuje jačinu udara vjetra na način da broji koliko puta se pojavila vrijednost veća od određene vrijednosti *limit* te sprema tu vrijednost u novu listu. Ukoliko je vrijednost varijable *counter* (sadrži broj podataka čija je vrijednost veća od vrijednosti zapisane u varijablu *limit*) veća od vrijednosti zapisane u konstantu *measurements_number*, tada postoji udar te metoda vraća srednju vrijednost udara. Konstanta *measurements_number* sadrži broj mjerenja koji odgovara vremenu od tri sekunde jer se prema meteorološkim standardima udar vjetra prepoznaje tako da moraju postojati vrijednosti koje su veće od srednje vrijednosti brzine vjetra, a traju tri sekunde. Konstanta ovisi o sklopovlju koje izvodi mjerenje, a konstanta za sklopovlje koje se korisiti u ovom završnom radu iznosi 1578. Nakon određivanja srednje brzine vjetra i udara vjetra, izračunati podaci stavljaju se u red pomoću kojeg se vraćaju u proces koji izvodi glavnu skriptu.

Izvođenje glavne metode za razred *Direction* obuhvaća sljedeći kod:

```
# This is main method for object type Direction
    def main(self):
        statistics = sigmaEstimation(self.data_dir, float)
        self.lock.acquire()
        self.dir_q.put(statistics[0])
        self.dir_q.put(statistics[1])
        self.lock.release()
```

Nakon što su se u konstruktoru ovog razreda preuzeli podaci o smjeru vjetra, računa se njegova srednja vrijednost smjera te odstupanje od srednje vrijednosti. Izračunati podaci se vraćaju u red, kako bi mogli koristiti u glavnoj skripti.

Nakon izvršavanja procesa *speed_processor* i *direction_processor*, glavna skripta nastavlja s upisom podataka u bazu. Baza podakata je ostvarena s programom otvorenog koda *RRDtool*, čija je specifičnost da se može definirati koliko dugo se žele čuvati podaci u bazi prije nego što se pobrišu. Takva baza podataka ne zahtjeva posebno upravljanje bazom jer se podaci sami brišu. Postoje paketi za Python koji omogućavaju direktan upis u bazu, samo je potrebno prethodno formatirati tekstualni zapis (*string*) podataka u određeni oblik.

```
db_keys = db_data.keys()

rrd_input = 'N:' + str(time_stamp) + ':'
     +str(db_data['speed']) + ':'
if 'speed_gust' in db_keys:
    rrd_input += str(db_data['speed_gust']) + ':'
else:
    rrd_input += 'U:'
rrd_input += str(db_data['direction']) + ':'
if 'direction_gust' in db_keys:
    rrd_input += str(db_data['direction_gust'])
else:
    rrd_input += 'U'
ret =
rrdtool.update('/home/nikola/meteo_app/db/data.rrd',
rrd_input)
if ret:
    print "Greska pri upisu u bazu! " + rrdtool.error()
```

Unutar mape (*dictionary*) *db_data* nalaze se prethodno izračunate srednje vrijednosti smjera i brzine vjetra te ukoliko postoje, brzina udara vjetra i standardna devijacija smjera vjetra. Lista *db_keys* je lista ključeva mape *db_data* pomoću koje možemo provjeravti postoje li udari vjetra ili odstupanje smjera vjetra od srednje vrijednosti. Podaci koji se upisuju u bazu odvojeni su dvotočkom, a ukoliko neki podatak ne postoji, postavlja se vrijednost 'U'. Oznaka 'N' na početku tesktualnog zapisa označava da se zabilježi stavarno vrijeme uspisa podatka u bazu. Upis u bazu podataka izvodi se naredbom *rrdtool.update*. Nakon izvrašavanja upisa u bazu, cijeli postupak izračuna se ponavlja u beskonačnoj petlji.

4.3 Upravitelj korisnčkih zahtjeva

Kao potpora korisniku za prikazivanje podataka te dohvaćanje datoteka s mjerenjima, postoje određene skripte koje se pokreću po isteku određenog vremena ili na akciju korisnika. Uobičajeni odgovor na neki akciju sadrži pokretanje upita prema poslužitelju pomoću AJAX tehnike (*Asynchronus JavaScript and XML*), izvršavanje određene PHP skripte na strani poslužitelja te vraćanje rezultata izvođenja skripte. Primjer jedne takve komunikacije može biti ažuriranje podataka o smjeru i brzini vjerta:

```
function getValuesForChart(){
     if (window.XMLHttpRequest) {
       xhttp = new XMLHttpRequest();
       } else {
       // code for IE6, IE5
       xhttp = new ActiveXObject("Microsoft.XMLHTTP");
     }
     xhttp.onreadystatechange = function() {
    if (xhttp.readyState == 4 && xhttp.status == 200) {
     responseVal = xhttp.responseText.split(":");
```

```
            values.push(responseVal[0]);
            values.push(responseVal[1]);
            values.push(responseVal[3]);
        }
        };
        xhttp.open("GET", "readData.php", false);
        xhttp.send();
    }
```

Funkcija *getValuesForChart* poziva se svakih nekoliko minuta kako bi osvježila vrijednosti mjerenja koje se nalaze u bazi. Funckcija prestavlja AJAX tehniku komunikacije s poslužiteljem. Pri svakom pozivu ove funkcije kreira se primjerak razreda *XMLHTTPRequest* koji služi za sinkronizaciju komunikacije s poslužiteljem. Ostvarivanje upita omogućuje se pozivom funkcija *open* i *send*. Tijekom komunikacije poslužitelj mijenja stanja. Pri svakoj promjeni stanja poziva se metoda *onreadystatechange* te u trenutku kada poslužitelj bude u stanju 4 i kada status bude 200, skripta je dobila odgovor na svoj upit i u ovom slučaju parsira niz znakova koji je dobila kao odogovr. Skripta *readData.php* je skripta koja se poziva *GET* metodom, što znači da se od poslužitelja zahtjevaju resursi:

```
<?php
    $last =
rrd_lastupdate('/home/nikola/meteo_app/db/data.rrd');
    echo implode(array_values($last['data']), ":");
?>
```

Skripa pomoću funkcije *rrd_lastupdate* iz paketa koji omogućuju pristup *RRDtool* bazi podataka, čita zadnji upis podataka u bazu te tih podataka kreira tekstualni zapis koji vraća kao odgovor na zatražene resurse. Ovakav način komunikacije između klijena i poslužitelja omogućava osvježavanje podataka na stranici bez učitavanja cijele stranice što za posljedicu ima manji mrežni promet i bržu komunikaciju.

Kao što je prethodno bilo navedeno, korisnik ima mogućnost preuzeti datoteku s podacima u *excel* formatu. Ta funkcionalnost se ostvaruje pomoću PHP skripte koja se počne izvršavati u trenutku kada korisnik zatraži preuzimanje datoteke.

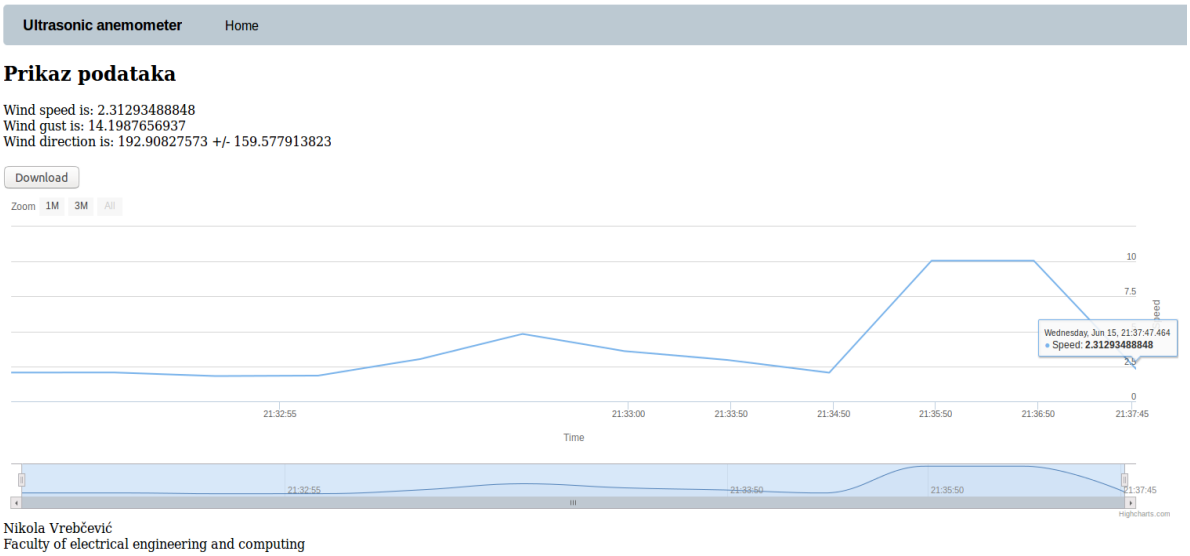
```
<?php
$data = rrd_fetch('/home/nikola/meteo_app/db/data.rrd',
     array("LAST", "-r", "5", "-s", "-360"));
$column_names = array_keys($data['data']);
$time = array_values($data['data']['time']);
$speed = array_values($data['data']['speed']);
$speed_gust = array_values($data['data']['speed_gust']);
$direction = array_values($data['data']['direction']);
$direction_gust = array_values($data['data']
     ['direction_gust']);

for($i=0; $i < count($time); $i++){
     $rows[] = array( $column_names[0] => $time[$i],
     $column_names[1] => $speed[$i],
     $column_names[2] =>  $speed_gust[$i],
     $column_names[3] => $direction[$i],
     $column_names[4] => $direction_gust_from[$i],
     $column_names[5] => $direction_gust_to[$i] );
}

function cleanData(&$str) {
     $str = preg_replace("/\t/", "\\t", $str);
     $str = preg_replace("/\r?\n/", "\\n", $str);
     if(strstr($str, '"')) $str = '"' . str_replace('"',
'""', $str) . '"';
}

$filename = "website_data_" . date('Ymd') . ".xls";
```

```
header("Content-Disposition: attachment;
     filename=\"$filename\"");
header("Content-Type: application/vnd.ms-excel");
$flag = false;
foreach($rows as $row) {
     if(!$flag) {
          echo implode("\t", array_values($column_names))
. "\r\n";
           $flag = true;
     }
     array_walk($row, 'cleanData');
     echo implode("\t", array_values($row)) . "\r\n";
}
?>
```

Skripta dohvaća podatke u određenom vremenu koje se zadaje unutar funckije *rrd_fetch* (za potrebe testiranja to su bile zadnje tri minute mjerenja). Podaci se spremaju u zasebne liste ovisno o tome koju vrstu informacije predstavljaju. Nakon parsiranja svakog retka, klijentu se vraća datoteka te mu internetski pretraživač nudi mogućnost da je preuzme.

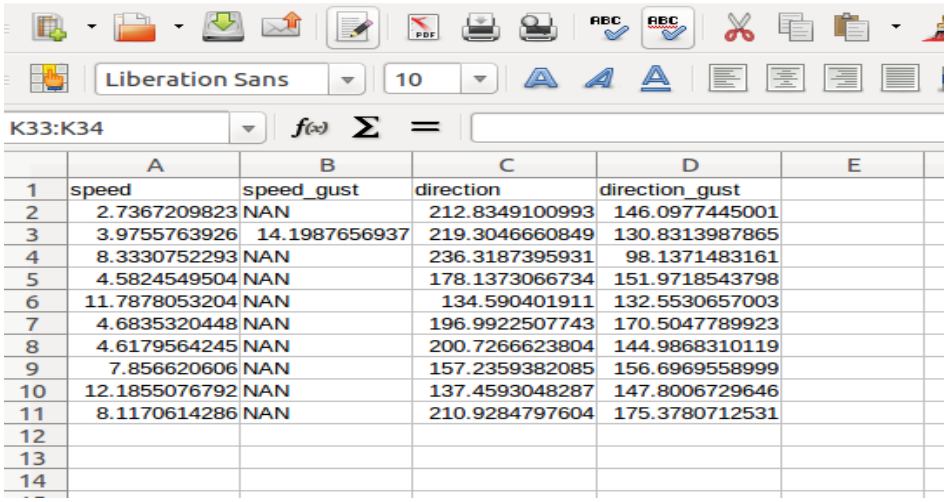
4.4 Prikaz podataka

Korisnik pristupa na internetsku stranicu unoseći adresu poslužitelja u željeni internetski preglednik. Kao odgovor na zahtjev za resursima, klijentski program dobiva sadržaj stranice i JavaScript program koji ima prethodno objašnjenu funkciju, a to je pristupanje poslužitelju te ažuriranje sadržaja internetske stranice bez ponovnog učitavanja stranice.



Slika 3. Prikaz internetske stranice s podacima o brzine i smjeru vjetra

Prethodna slika (*Slika 3*) prikazuje kako izgleda stranica na kojoj korisnik može pratiti brzinu i smjer vjetra. Prikaz podatakak sastoji se od grafičkog i numeričkog dijela gdje se u grafičkom dijelu prikazuje brzina vjetra, a u numeričkom se prikazuje brzina udara i srednji smer vjetra s devijacijom. U budućem radu omogućit će se grafički prikaz u obliku grafa srednje vrijednosti smjera puhanja vjetra, kao i grafički prikaz u obliku vektora trenutne srednje vrijednosti smjera puhanja vjetra. Pritiskom na gumb *Download* korisnik je u mogućnosti preuzeti datoteku u *excel* formatu te ju analizirati. Prikaz preuzete datoteke prikazan je na sljedećoj slici (*Slika 4*).



Liberation Sans 10

K33:K34

	A	B	C	D	E
1	speed	speed_gust	direction	direction_gust	
2	2.7367209823	NAN	212.8349100993	146.0977445001	
3	3.9755763926	14.1987656937	219.3046660849	130.8313987865	
4	8.3330752293	NAN	236.3187395931	98.1371483161	
5	4.5824549504	NAN	178.1373066734	151.9718543798	
6	11.7878053204	NAN	134.590401911	132.5530657003	
7	4.6835320448	NAN	196.9922507743	170.5047789923	
8	4.6179564245	NAN	200.7266623804	144.9868310119	
9	7.856620606	NAN	157.2359382085	156.6969558999	
10	12.1855076792	NAN	137.4593048287	147.8006729646	
11	8.1170614286	NAN	210.9284797604	175.3780712531	
12					
13					
14					

Slika 4 Prikaz *excel* datoteke nakon preuzimanja

5 Zaključak

Dostupnost složene, a relativno jeftine tehlonogije omogućila je veću automatizaciju, ali i unaprijeđenje određenih procesa. Primjerice, proces mjerenja brzine i smjera vjetra bio je automatiziran i prije, međutim ostvarivanje istog tog procesa pomoću računla tipa Raspberry Pi je uvelike smanjilo iznos početnog troška koji korisnik mora uložiti za jednu takvu mjernu stanicu, bez degradicije performansi računala koje upravlja procesom mjerenja. Također, unaprjeđenje sklopovlja omogućava izvođenje složenijih algoritama koji mogu proširiti trenutnu funkcionalnost ili je moguće optimirati postojeće algoritme za brže izvođenje. Moguće je uključiti takva računala u mrežu uređaja koji imaju funkciju upravljati i nadzirarati određene pojave koja se zove Internet stvari (eng. *Internet of things*). U tom slučaju povezalo bi se više mjernih stanica u jednu mrežu, podaci bi se mogli prikupljati na jednom mjestu te bi se pomoću nekog sustava za analizu velike količine podataka (eng. *big data processing*) mogli donositi određeni zaključci o karakteristikama vjetra na pojedinim geografskim lokacijama. Raspberry Pi se pokazao kao računalo koje je poželjno postaviti kako ulaznu točku za mjerenje i obradu izmjerenih podaka koja se može proširiti širokim spekrtrom tehnologija te može naći primjenu za gotovo sve potrebe koje ne dovode ljudski život u opasnost.

(Nikola Vrebčević)

6 Literatura

Raspberry Pi Foundation (https://www.raspberrypi.org/). Lipanj 2016.

Raspbian (https://www.raspbian.org/). Lipanj 2016.

Python Software Foundation (https://www.python.org/). Lipanj 2016.

PHP Group (http://php.net/). Lipanj 2016.

Wikipedia: *JavaScript* (https://en.wikipedia.org/wiki/JavaScript). Lipanj 2016.

Wikipedia: *Interquartile range* (https://en.wikipedia.org/wiki/Interquartile_range). Lipanj 2016.

Wikipedia: *Median absolute deviation* (https://en.wikipedia.org/wiki/Median_absolute_deviation). Lipanj 2016.

RS Components (http://uk.rs-online.com/web/). Lipanj 2016.

Apache Software Foundation (http://www.apache.org/). Lipanj 2016.

RRDtool (http://oss.oetiker.ch/rrdtool/). Lipanj 2016.

7 Sažetak

Naslov

Sustav za prikaz, kronološku registraciju i statističku analizu mjernih podataka

Sažetak

Potreba za unapređivanjem procesa mjerenja bila je inspiracija za temu ovog završnog rada. Svaki proces mjerenja generira velike količine podakata među kojima je većina podataka relevantna, ali postoje i oni koji to nisu. Velike količine podataka zahtjevaju analizu tih podataka jer ti podaci pojedinačno ne sadrže onu količinu informacije koju sadrži određeni skup podataka. Statističkim analizama nad skupina podataka izvlače se informacije koje prezentiraju stvarnu sliku određenog mjerenja, a bitne su korisniku. Osim analize podataka, protrebno je omogućiti korisniku pristup uređaju koji vrši mjerenje i analizu s udaljene lokacije. U tu svrhu računalo koje vrši mjerenja i analize podataka, potrebno je povezati na mrežu te na taj način omogućiti korisniku pristup podacima putem internetske stranice koja prikazuje korisniku podatke na njemu razumljiv način.

Ključne rječi

Interkvartilni raspon, srednja vrijednost odstupanja, Raspberry Pi, Python, JavaScript, PHP, Apache HTTP poslužitelj

8 Abstract

Title

System for Presentation, Chronological Registration and Statistical Analysis of Measurements

Abstract

Inspiration for this prediploma thesis has been found in demand for better measurement process. Every measurement process generate vast amount of data where great majority of data is relevant but some of it is not relevant. Generally, data needs to be processed because individual measurements does not carry as much information as aggregate of measurements. With statistic analysis under that aggregate of measurements it is possible to extract information that represents real picture of some measurements which is relevant to user. Beside statistical analysis, user should have access to device that performs measuring and analysis from remote location. For that purpose, device should be placed in network and it should have ability to serve web site that user could access data and read it in way understandable for user.

Key Words

Interquartile range, median absolute deviation, Raspberry Pi, Python, JavaScript, PHP, Apache HTTP Server