Mobilni sustav nadzora okoline u stvarnom vremenu za Android platformu

Gregurić, Ivan

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:884244

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-25**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni diplomski studij

MOBILNI SUSTAV NADZORA OKOLINE U STVARNOM VREMENU ZA ANDROID PLATFORMU

Diplomski rad

Ivan Gregurić

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 12.05.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Ivan Gregurić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina	D-491, 16.10.2014.
OIB studenta:	98887911357
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Nenadić
Član Povjerenstva:	Bruno Zorić
Naslov diplomskog rada:	Mobilni sustav nadzora okoline u stvarnom vremenu za Android platformu
Znanstvena grana rada:	**Arhitektura računalnih sustava (zn. polje računarstvo)**
Zadatak diplomskog rada:	U teorijskom dijelu rada treba opisati zahtjeve okoline koja se nadzire s gledšta mogućnosti korištenih senzora, osmisliti sustav za nadzor u stvarnom vremenu i opisati njegove sklopovske komponente. U praktičnom dijelu rada potrebno je sklopovski obaviti prilagodbu komponenti sustava okolini, programski ostvariti Android aplikaciju za prikaz i pohranu podataka, te sustav prikladno ispitati.
Prijedlog ocjene pismenog dijela ispita (diplomskog	Vrlo dobar (4)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 2 razina
Datum prijedloga ocjene mentora:	12.05.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.05.2017.

Ime i prezime studenta:	Ivan Gregurić
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D-491, 16.10.2014.
Ephorus podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Mobilni sustav nadzora okoline u stvarnom vremenu za Android platformu**

izrađen pod vodstvom mentora Prof.dr.sc. Goran Martinović

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

1. UVOD 1
2. PRIKUPLJANJE I OBRADA PODATAKA U STVARNOM VREMENU 2
2.1. Zahtjevi za prikupljanje i obradu podataka u stvarnom vremenu 3
2.2. Osnovni pojmovi i podjela sustava za rad u stvarnom vremenu 3
2.3. Upravljanje sustavom 5
2.4. Fuzija podataka 6
2.5. Kratki uvod u fuziju podataka 7
2.6. Višesenzorski sustav 7
2.7. Sinergija podataka 7
2.8. Tipovi fuzije podataka 8
2.8.1. Najčešće korištene raspodjele fuzije podataka 8
2.9. Internet stvari (IoT) i njegov razvoj 12
2.10. Upotreba Interneta stvari 13
2.11. Komunikacijski modeli Interneta stvari 15
2.12. Veliki skupovi podataka 17
2.13. Veliki skupovi podataka i fuzija podataka 17
2.14. Veliki skupovi podataka i Internet stvari 18
3. ZAHTJEVI NA RAZVIJENI SUSTAV PRIKUPLJANJA I OBRADE PODATAKA 19
3.1. Funkcijski zahtjevi 20
3.2. Vremenski zahtjevi na izrađeni sustav 20
4. SKLOPOVSKO RJEŠENJE SUSTAVA 21
4.1. Sklop za obradu podataka 21
4.2. Građa i mogućnosti Arduino Mega 2560 rev3 21
4.3. Razvojna pločica Arduino Mega 2560, mikroupravljač Atmel ATmega 2560 23
4.4. Napajanje i komunikacija 24
4.5. Digitalni i analogni ulazi i izlazi 24
4.6. Arduino razvojno okruženje 25
4.7. Sklop za prikupljanje podataka 25
4.8. Senzor za praćenje temperature i vlage zraka 26
4.9. Senzor za praćenje tlaka zraka 26
4.10. Senzor za mjerenje temperature zemlje 27
4.11. Senzor za praćenje vlage zemlje 28
4.12. Komunikacijski dio 28
4.13. Usmjerivač TP-Link 703n 29

4.14. 3G stick za mobilni internet 29
5. PROGRAMSKO RJEŠENJE SUSTAVA 31
5.1. Programska podrška prikupljanja podataka 31
5.2. Programski kod za Arduino platformu 31
5.3. Podrška na poslužitelju 32
5.4. Programski kod PHP za prikupljanje podataka 32
5.5. Programski kod za pohranu podataka u bazu MySQL 33
5.6. Operacijski sustav OpenWRT 33
5.7. Programsko rješenje za prikaz rezultata mjerenja na operacijskom sustavu Android 34
5.8. Operacijski sustav Android 34
5.9. Programski jezik Java 34
5.10. Android IDE 35
5.11. Biblioteka za mrežnu komunikaciju 35
5.12. Programsko rješenje mobilne aplikacije 35
5.13. Korisničko sučelje 36
5.14. Upravljačko sučelje 38
5.15. Fuzija podataka koristeći dobivene podatke 43
6. PRIMJENA SUSTAVA I TESTIRANJE S ANALIZOM REZULTATA 45
6.1. Postavke komunikacijskih uređaja 45
6.2. Unos podataka s Arduina u bazu podataka u oblaku 49
6.3. Prikaz podataka iz baze podataka u oblaku računala na Android uređaj 50
6.4. Testiranje rada sustava 53
7. ZAKLJUČAK 56
LITERATURA 57
POPIS SLIKA 59
SAŽETAK 62
ABSTRACT 59
ŽIVOTOPIS 63

1. UVOD

Razvoj tehnologije doveo je ljude u položaj da ako se prepozna pojedinačni problem i postoji želja za njegovim rješenjem, ono vjerojatno postoji negdje na Internetu. Djelomično ili potpuno neovisno, ali negdje je netko prepoznao isti problem i ponudio svoju inačicu rješenja.

Tema diplomskog rada javlja se kao potreba da se u stvarnom vremenu može imati nadzor nad udaljenom okolinom poput staklenika ili poljoprivrednih površina i biti u mogućnosti pravovremeno reagirati (ako se pokaže da uvjeti okoline nisu u razinama željenog) te smanjiti rizik i eventualne gubitke. Pritom potreba za automatiziranjem reakcije otvaranja staklenika ili navodnjavanja ovisno o dobivenim podacima nije postojala, ali ju je moguće vrlo jednostavno implementirati. Također, cilj ovog diplomskog rada je daljnji razvoj platforme za nadzor i upravljanje udaljenom okolinom budući da je cijeli projekt osnovan na otvorenom kodu i vidljiv je svima koje zanima rad na njemu. Uporabom lako dostupnih dijelova poput Arduina, senzora za isti i mrežnog usmjerivača za sklopovlje te besplatnih poslužitelja u oblaku pružena je mogućnost vrlo jednostavnog upuštanja u svijet otvorenog koda, Interneta stvari i razvoja Android aplikacija.

U nastavku diplomskog rada prikazana je ideja izrade i što se od rada očekivalo. Tako se u drugom poglavlju nalaze tehnologije koje se koriste. Tehnologije su ukratko opisane i navedene su neke njihove glavne podjele te područja gdje ih se može naći u primjeni. U trećem poglavlju navedeni su zahtjevi koje bi sustav trebao ispunjavati u svojem radu. U četvrtom poglavlju prikazano je sklopovlje koje se koristi za izradu diplomskog rada. U petom poglavlju navedene su tehnologije koje su korištene za programsko rješenje diplomskog rada. Također, dani su primjeri programskog koda. U šestom poglavlju prikazan je način korištenja sustava i rezultati dobiveni korištenim sklopovskim i programskim rješenjem uz komentare mogućih poboljšanja.

2. PRIKUPLJANJE I OBRADA PODATAKA U STVARNOM VREMENU

Uvođenjem tehnologije u sve više sfera ljudskog života dolazi do porasta količine prikupljenih podataka koje korisnici dijele ili šalju neovisno jesu li oni toga svjesni ili ne. Krene li se samo od industrije mobilnih uređaja koja ima skoro najveći utjecaj na ljudski način komuniciranja i planiranja svakodnevnice pa sve do raketa koje se lansiraju za razne istraživačke svrhe, može se vidjeti isti cilj prikupljanja i obrade podataka, ali s različitim zahtjevima na vrijeme izvršavanja operacije.

Razvoj sustava za rad u stvarnom vremenu usko je povezan s razvojem računalne tehnologije i njezinih kapaciteta. Prema [1] uzme li se za primjer vojno zrakoplovstvo ili neki sustavi u medicini, može se primijetiti kako su sustavi postali sve složeniji, ali neke temeljne karakteristike nisu se promijenile još iz 1960-ih.

Iako je ideja sustava u stvarnom vremenu postavljena i ranije te prezentirana kao simulacija, prema [1] svoj pravi oblik dobiva kasnih 1940-ih. Računanje u stvarnom vremenu (engl. *real-time computing*) svoj početak u praktičnom smislu kreće na projektu američke mornarice 1947. pod nazivom Whirlwind MIT. Radi se o simulatoru letenja koji je trebao pomoći pri obučavanju američkih pilota. Ostavlja svoj trag u povijesti jer prvi u svojoj građi koristi vakumske cijevi i feritnu memorijsku jezgru, a za njihovo upravljanje koristi se kompajler koji je preteča Fortrana. Nedugo nakon toga započinje se i na radu s projektom SAGE (engl. *Semiautomatic Ground Environment*) u suradnji s IBM-om, a imao je zadaću upravljanja protuzrakoplovnom obranom.

Početkom 50-ih godina 20. stoljeća dolazi do povećanja složenosti samih računalnih sustava, ali ne i do fizičkog zauzimanja prostora računala. U listopadu 1953. ekipa predvođena Seymourom Crayom predstavlja Univac 1103A (engl. *Univac Scientific*) koji za razliku od svojih prethodnika koristi memoriju s magnetskom jezgrom, sklopovlje za pomični zarez (engl. *floating point*) te sklopovlje za asinkroni prekid (engl. *asynchronous interrupt*).

Kraj 20. stoljeća donosi naglu ekspanziju mikroprocesora, a samim tim i lakšu implementaciju sustava koji rade u stvarnom vremenu ne samo u znanstvene, vojne i industrijske već i u komercijalne svrhe. Nakon dolaska drugih tehnologija izrade bila je potrebna prilagodba i samog programskog jezika te se u to vrijeme implementiraju programski jezici tipa Fortran, CMS-2 i JOVIAL. Zbog pojave sve većeg broja programskih jezika Ministarstvo obrane Sjedinjenih Američkih Država izdaje zahtjev za izradu jednog programskog jezika koji bi uspješno pokrio rad svih servisa i standardizirao rad na programiranju sustava za rad u stvarnom vremenu. Kao odgovor na taj zahtjev 1983. godine nastaje Ada programski jezik koji nakon prepoznatih i

popravljenih pogrešaka te određenih nadogradnji 1995. godine dobiva verziju Ada 95.

U 21. stoljeću sustav u stvarnom vremenu zbog dostupnosti i brzine razvoja dobiva veću pozornost od novo-nastalih industrija. Veliki dio autoindustrije smanjuje broj zaposlenika i zamjenjuje ih robotskim rukama koje bez zamora i gubitka pažnje obavljaju isti posao brže i učinkovitije. Sukladno s porastom tehnologija koje koriste sustave stvarnog vremena raste i broj tehnologija za obradu skupljenih podataka, prepoznavanje uzorak te učenje iz skupljenih informacija.

2.1. Zahtjevi za prikupljanje i obradu podataka u stvarnom vremenu

Da bi se za sustav moglo reći da je sustav koji radi u stvarnom vremenu prema [1], od njega se traži da upravljačko računalo na kojem se ono pokreće u svakom trenutku radi ispravno, bez zastoja te neovisno o poslovima i zadacima koji se trenutno postavljaju na izvođenje u sustavu. Većina sustava koji se danas globalno koriste nisu sposobni za takvu vrstu zadaće jer da bi se postiglo navedeno, potrebno je uskladiti sve elemente sustava (sklopovlje, operacijski sustav, programa i potprograma), a potrebno je i obučiti samog korisnika kako upravljati cijelim sustavom.

2.2. Osnovni pojmovi i podjela sustava za rad u stvarnom vremenu

Ako se pobliže promotre neki od procesa koji nas okružuju, može se jednostavno zaključiti da se stanje sustava tijekom vremena mijenja. Do promjene u sustavu dolazi bez obzira na ljudski utjecaj. Kako bi se što točnije mogla izmjeriti ili promotriti promjena koja se događa u sastavu, tu zadaću prenosimo s čovjeka na računalo. Računalo će navedenu promjenu izmjeriti preciznije i brže, a i puno je jednostavnije prilagoditi računalo za određenu zadaću te vrste nego čovjeka.

Neispravan sustav ili sustav koji je došao u neočekivano stanje dati će pogrešnu informaciju. Ako se sustav vrati u prvobitno stanje, govori se o kratkotrajnoj pogrešci, a ako sustav više ne može doći u operabilno stanje, govori se o zatajenju. U sustav se još u fazi projektiranja unose brojčane vrijednosti i njih nazivamo logičkom ispravnosti. Kako bi se prepoznalo radi li se o neočekivanom stanju ili zatajenju, uspoređuju se vrijednosti logičke ispravnosti i vrijednosti dobivene očitavanjima sa senzora. Ovisno o zahtjevima na sustav postoji mogućnost da logička ispravnost nije dovoljna te kvaliteta sustava ovisi koliko će brzo informacija doći na traženo odredište. Uzimajući u obzir da sustav na temelju dobivene informacije pokreće određene postavne sprave ili aktuatore (motori, ventili i sl.), zadovoljavanje vremenskih zahtjeva uvelike će utjecati na kvalitetu samog sustava.

Kada je sustavu pored logičke ispravnosti važno zadovoljavanje vremenskih zahtjeva, radi se

o sustavu za rad u stvarnom vremenu (engl. *real-time systems*).

Zadaće i procesi koji se ranije spominju mogu se povezati s događajima u sustavima za rad u stvarnom vremenu. Ovisno o pojavljivanju, događaji mogu biti:

1. Periodni– događaji koji se pojavljuju u jednakim vremenskim razmacima (periodima).
2. Aperiodni – događaji koji se pojavljuju u sustavu nasumično ili u nepravilnim vremenskim razmacima. Opisuju se određenim statističkim raspodjelama.
3. Sporadični – događaji koji se rijetko pojavljuju i zbog toga se ne uvrštavaju među periodne ili aperiodne.

Događaji koji se izvršavaju istovremeno s drugim događajima u sustavima za rad u stvarnom vremenu nazivaju se sinkroniziranim skupinama, dok događaji koji se izvršavaju nepovezano nazivaju se asinkronim skupinama događaja.

Budući da se na svaki od sustava za rad u stvarnom vremenu gleda kao na sustav s točno određenom zadaćom, tijekom vremena je nastalo mnogo načina na koje se sustave može kategorizirati. Najvažnija podjela je ona koja govori o očekivanjima za sustav za rad u stvarnom vremenu odnosno o posljedicama na cijeli sustav ako se određena zadaća ne izvrši pravovremeno. Prema toj podjeli postoje sljedeće vrste sustava:

1. Sa strogim vremenskim zahtjevima (engl. *hard RTS*) – sustavi kod kojih se zadana vremenska ograničenja moraju strogo poštovati jer u suprotnom dolazi do zakazivanja cijelog sustava. Kao primjer može se navesti sustav upravljanja u zrakoplovu.
2. Ublaženi (engl. *soft RTS*) – kod ublaženih sustava za rad u stvarnom vremenu ako dođe do kašnjenja sustav neće u potpunosti zakazati iako je moguće da učestalim ponavljanjem pogreške može doći do smanjene kvalitete proizvoda ili usluge. Primjer je prijenos podataka uživo preko mreže.
3. Čvrsti (engl. *firm RTS*) – nastaje kao kombinacija strogih i ublaženih sustava za rad u stvarnom vremenu. Dozvoljen je poneki ispad ali više njih rezultira ispadom sustava.

Može se primijeniti još jedna podjela koja je vezana uz pojavu određenog događaja u sistemu, pa se tako razlikuju sustavi upravljani protokom vremena (engl. *time-triggered*) i sustavi pobuđivani događajima (engl. *event-triggered*).

2.3. Upravljanje sustavom

Prema [1], sustav za rad u stvarnom vremenu mora obavljati tri funkcije: raspoređivanje, prosljeđivanje te usklađivanje skupova zadataka s obavljanjem komunikacije unutar sustava. Dio sustava koji obavlja najuži dio navedenih zadaća nalazi se u jezgri operacijskog sustava (engl. *kernel*). Raspored određuje kada će se određeni zadatak izvršiti, dok dio sustava koji provodi prosljeđivanje vodi računa o tome da zadatak pravovremeno dođe u slijed za izvođenje. Dio zadužen za unutarnju komunikaciju i usklađivanje osigurava suradnju između događaja koji se obavljaju. Uz *kernel* se nalaze još *mikrokernel* i *nanokernel* koji upravljaju ponašanjem zadataka na nitima.

Pristup kojim se upravlja sustavom za rad u stvarnom vremenu prilagođava se ovisno o zahtjevima na sustav. Ovisno radi li se o jednostavnijim ili složenijim sustavima, može se koristiti:

1. Upravljačka petlja – koristi se za najjednostavnije ugrađene SRSV-a. Na mikroupravljaču se izvršava programska petlja koja nakon što očita ulazne podatke pregledava uvjete koji su joj postavljeni i treba li izvršiti određenu akciju ovisno o dobivenim podacima. Sama upravljačka petlja može se dodatno proširiti ovisno koristi li se za upravljanje nekih periodičnih zadataka.
2. Upravljanje zasnovano na događajima – ovaj način upravljanja može se uzeti kao nadogradnja prijašnjeg. Dok upravljačka petlja konstantno prima podatke iz okoline i ovisno o njihovim rezultatima reagira, sustav koji se temelji na događajima ima zadatak reagirati u trenutku pojave događaja. Reagiranje na taj način omogućuje mehanizam prekida. Sustavi s takvim rješenjem moraju imati podsustav za prihvat prekida. Njegova zadaća je povezivanje trenutka kada se pojavio neki događaj s određenom reakcijom sustava zadanom za takav događaj. Reguliranje pojave kada više događaja dođe na red za izvođenje, reakcija na njih obavlja se pomoću postavljanja prioriteta na određeni događaj.
3. Sustav upravljanja alarmima i prekidima – na sustav za rad u stvarnom vremenu upravljan događajima dodaje se još jedan podsustav i on služi za upravljanje vremenom. Sustav se dodatno proširuje mogućnošću postavljanja alarma. Njegova zadaća je aktiviranje brojila koje sinkrono odbrojava i dolaskom do nule izaziva prekid. Nakon pregleda svih prekida alarmi se ponovo aktiviraju. Ovako građeni sustavi mogu kombinirati različite načine upravljanja kako bi se što bolje prilagodili zahtjevima na njih. Asinkronim događajima tako se može upravljati iz prekidnih funkcija dok se ostatak sustava slobodno može pokriti alarmima i prekidima.
4. Kooperativna višenitnost – podsustav za višenitnost uzrokuje promjene na svim ostalim

uvedenim podsustavima. Višenitnost se ne mora nužno uvesti kao podsustav već kao dio programa koje sustav obavlja na sebi. Ako se ipak ugrađuje kao podsustav, potrebno je razviti sustav za pohranu podataka iz prve niti i obnovu podataka iz druge niti. Raspoređivanje zadataka na niti može biti izravno i neizravno. Kod izravnog, u programu se definira kada će nit A prepustiti zadatak niti B, dok se kod neizravnog raspoređivanje zadataka po nitima izvršava pomoću unaprijed određenih prioriteta.

5. Višenitnost – upotrebom različitih naprava koje će imati u sebi ugrađen sustav za raspoređivanje prekida zadataka prema određenoj hijerarhiji može se napraviti višenitni istiskujući sustav (engl. *preemptive*). U takvom slučaju niti mogu biti u aktivnom ili blokiranom stanju. Aktivne niti odrađuju određeni zadatak, a blokirane čekaju na neki mehanizam koji će ih aktivirati. S obzirom na to da se radi o visoko složenim operacijama, neki od jednostavnijih načina upotrebe višenitnosti je i sustav gdje se svakoj niti definira njezin način upravljanja zadacima. Ako se ipak želi nešto složenije, može se iskoristiti hijerarhijski sustav odlučivanja. Niže rangirani zadaci obavljaju se na nižim razinama dok se odluke o daljnjim radnjama obavljaju na višim razinama.
6. Raspodijeljeni sustavi – raspoređivanje sustava na manje komponente (čvorove) koji zajedno komuniciraju. Komunikaciju odrađuje zaseban sustav prema određenim protokolima. Zbog količine čvorova te informacija koje kolaju između problem je uspostaviti vremensku usklađenost i samim time otežano je predvidjeti ponašanje sustava.

U diplomskom radu koristi se inačica upravljanja sustava stvarnog vremena pomoću upravljačke petlje. Dio koji nedostaje da bi bio sustav bio u potpunosti upravljan upravljačkom petljom su uvjeti koje sustav provjerava ovisno o podacima koje dobiva sa senzora te aktuatori koji se aktiviraju ukoliko sustav dođe u stanje da treba ispuniti neki od uvjeta.

2.4. Fuzija podataka

Iz [2] fuziju podataka opisujemo kao postupak uvrštavanja veće količine podataka koje predstavljaju stvarni svijet s ciljem dobivanja njegovog što točnijeg i korisnijeg prikaza. Kako će podaci biti iščitani, ovisi o percepciji samog sustava koji ih koristi. Fuzija podataka može se podijeliti na podatkovnu i na informacijsku. Kod podatkovne fuzije kao rezultat dobivaju se određene vrijednosti koje se mogu direktno upotrijebiti za rješavanje nekog problema dok je kod informacijske fuzije potrebno dodatno proučiti percepciju iz koje su informacije dobivene.

2.5. Kratki uvod u fuziju podataka

Prema [3] i [4], moderan način korištenja fuzije podataka započinje u vojnoj industriji. Širenjem njezinih mogućnosti ubrzo nakon toga svoju svrhu pronalazi i u civilnom društvu. Primitivan način korištenja fuzije podataka može se proučiti koristeći vlastita osjetila. Evolucijom su ljudi i životinje razvili višestruka osjetila koja su im pomogla kako bi preživjeli. Više nije bilo potrebe oslanjati se samo na jedno osjetilo nego korištenjem ostalih doći do zaključka o pojavi koju promatraju.

Fuzija podataka u modernim računalnim sustavima kreće s prvim pokušajima izvedbe umjetne inteligencije početkom 1970. Jedan od prvih pokušaja umjetne inteligencije bilo je računalo SYMBOLIC za koji je napravljen posebno prilagođen programski jezik. Danas se, prema [4], fuzija podataka pronalazi u najnaprednijim vojnim sustavima za navođenje raketa, automatsko prepoznavanje meta, nadzor bojišta, navođenje autonomnih vozila, dok se u civilnoj primjeni pronalazi u nadzorima proizvodnih pogona, medicini, nadzoru i upravljanju kompleksnim strojevima.

2.6. Višesenzorski sustav

Prema [5], višesenzorski sustav fuzije podataka služi kako bi se povećala kvaliteta informacije koja se dobiva ako se svaki od senzora koristi individualno. Podaci se mogu dobiti s više različitih senzora i preko njih dobiti određene spoznaje što ujedno može značiti i korištenje jednog senzora, ali kroz više iteracija mjerenja.

Glavni problem korištenja višesenzorskog sustava je određivanje najprikladnije metode za spajanje podataka dobivenih sa senzora. U današnjim sustavima najviše se koriste statističke metode spajanja podataka te izvlačenje informacija iz dobivenih rezultata. Bitna prednost statističkog pristupa je što je moguće uvrstiti određene veze između senzora te time smanjiti mogućnost pojave pogreške, a i ako dođe do nje, bit će je lakše pronaći. Navedeni pristup naziva se Bayesovim zaključivanjem i omogućuje izračun procjene nesigurnosti u svim nepoznanicama problema.

2.7. Sinergija podataka

Kao glavni cilj fuzije podataka prema [5], može se postaviti zadaća poboljšanja kvalitete izlazne informacije koja se dobiva kroz sinergiju podataka. Kako bi se postigla sinergija podataka, nije potrebno korištenje više senzora. Dovoljno je korištenje jednog senzora, ali s većim brojem

očitanja podataka s istog. Ako bi se željela povećati sinergija sustava, preporučljivo je koristiti veći broj senzora, što povećava robusnost sustava, smanjila mogućnost pojave pogreške na senzorima, smanjila količinu šuma te povećala točnost procjene. Načini na koje višesenzorski sustav poboljšava postojeće sustave u koje se ugrađuje su:

1. Prikaz – dobivanjem informacije iz skupa podataka, informacija dobiva novu razinu apstrakcije koja je viša od početne. Postizanjem viših razina apstrakcije dobiva se bolja relacijska povezanost među dobivenim informacijama.
2. Sigurnost – povezivanjem podataka pridonosi se dobivanju kvalitetnije, a potom i sigurnije informacije.
3. Točnost – sa što više očitavanja sa senzora i podataka, manja je mogućnost pojave šuma u sustavu koja nastaje u procesu popunjavanju vremena između očitanja.
4. Cjelovitost – uključivanjem nove informacije o sustavu proširujemo spoznaju o stanju sustava.

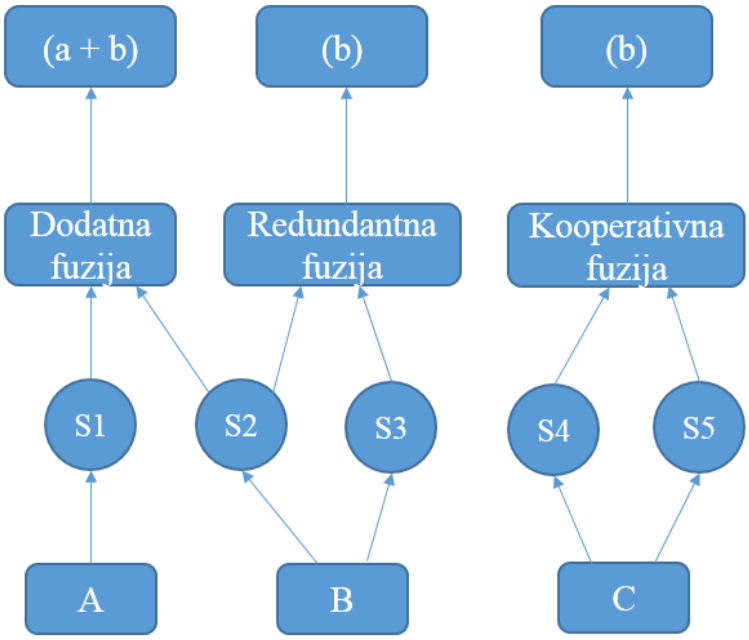
2.8. Tipovi fuzije podataka

Zbog prirode samog sustava koji svaku pojavu gleda kao individualnu, svakoj razradi fuzije podataka pristupa se kao zasebnoj cjelini. Unatoč tome smatra se prema [6], da je najbolju definiciju fuzije podataka dala radna skupina Joint Directors of Laboratories. Oni su je definirali kao višerazinski proces koji se bavi korelacijom, povezivanjem podataka i informacija iz jednostrukih i složenih izvora s ciljem postizanja potpune i pravovremene procjene situacije, procjena prijetnji, te njenog utjecaja na sustav. Zbog svoje multidisciplinarne uključenosti u različita područja djelovanja nailazi se na problem definiranja fuzije podataka. Osim njezine definicije problem predstavlja i njezina raspodjela prema neki zajedničkim značajkama. U nastavku se nalaze neke od postojećih raspodjela koje se najčešće koriste u primjeni.

2.9.Najčešće korištene raspodjele fuzije podataka

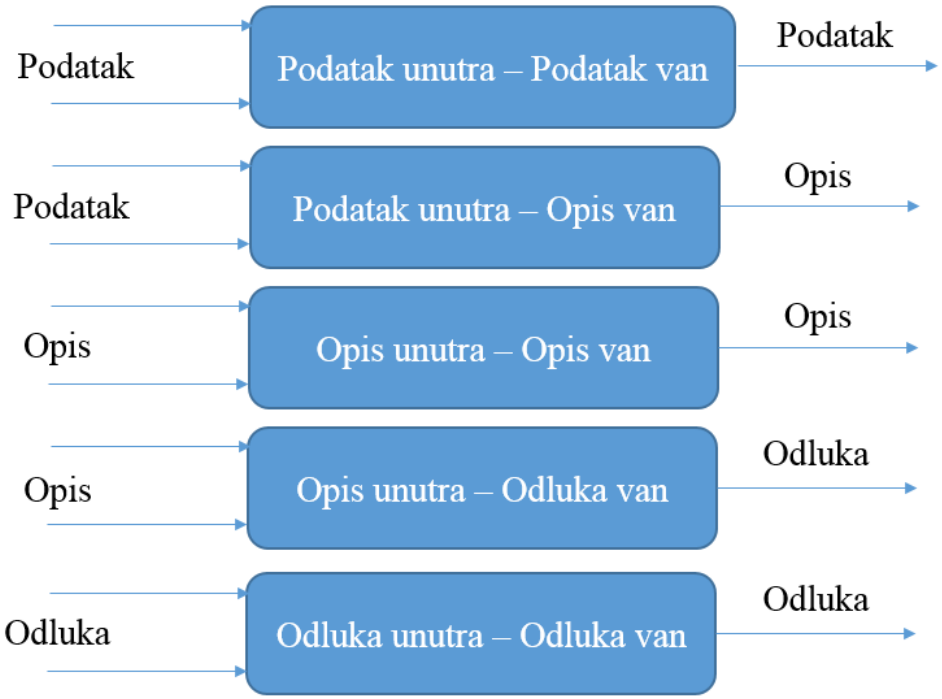
1. Fuzija podataka koja dodaje vezu između izvora podataka slikovito se može pokazati na slici 2.2. koja je napravljena prema [6]. Dijeli se na:
   - Dodatna – kada se do informacije dolazi koristeći podatke s većeg broja ishodišta s ciljem dobivanja što točnije informacije.
   - Redundantna – koristeći ovaj tip dobiva se veća sigurnost u točnost informacije, jer podaci koji se dobivaju dolaze s nekoliko izvora istoga tipa.
   - Kooperativna – kombinacijom nekoliko podataka s različitih izvora dobiva se informacija

šireg spektra nego što se dobiva gledanjem svakog podatka zasebno.



Sl. 2.1. Ilustrativni prikaz rada fuzije podataka.

2. Fuzija podataka prema postavkama ulazno/izlaznih tipova podataka čiji se slikoviti prikaz može vidjeti na slici 2.3 prema [6] dijeli se na:
   - Podaci unutra – podaci van (engl. *data in – data out, DAI-DAO*) – najosnovnija metoda fuzije podataka. Podaci se ne obrađuju već se prosljeđuju dalje u istom obliku u kojem su došli sa senzora.
   - Podaci unutra – značajke van (engl. *data in – feature out,* DAI - FEO) – na ovoj razini proces fuzije podataka uzima podatke sa senzora obrade i odabire samo one koji mu služe kako bi opisali neki entitet u okolini.
   - Značajka unutra – značajka van (engl. *feature in – feature out, FEI - FEO*) – kod ovog tipa fuzije i ulazni i izlazni podaci imaju pridodane određene značajke. Dodane značajke pomažu sustavu da bi dobivena informacija bila što naprednija ili dobila novu značajku.
   - Značajka unutra – odluka van (engl. *feature in – decision out, FEI - DEO*) – sustavi koji koriste ovaj tip fuzije podataka na ulaz dobivaju određeni skup značajki prema kojima sustav donosi odluke te zatim iste te odluke šalje na izlaz.
   - Odluka unutra – odluka van (engl. *decision in – decision out, DEI - DEO*) – ulaz kod ovog tipa fuzije unaprijed donosi već neke utvrđene odluke, dok na izlaz dolazi nova i bolja odluka.

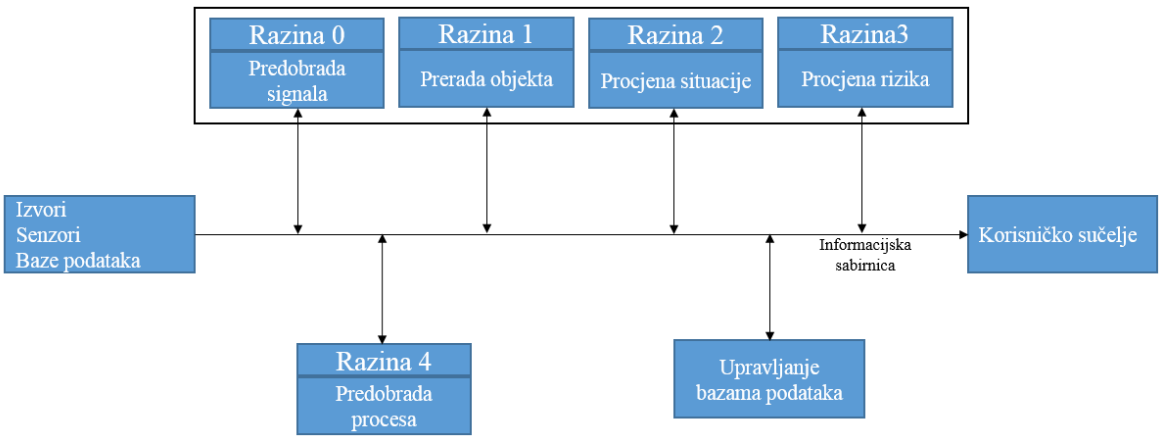


Sl. 2.2. Shema Dasarathy-eve klasifikacija fuzije podataka prema postavkama ulaza i izlaza.

3. Fuzija podataka temeljena na apstrakciji ima nekoliko razina apstrakcije:
   - Signal – signal koji je dobiven sa senzora direktno se prosljeđuje dalje.
   - Piksel – djeluje na razini slike gdje unaprjeđuje zapažanje i obradu podataka na slici.
   - Karakteristika – koristeći podatke dobivene signalom ili slikama proizvodi nove značajke promatrane pojave.
   - Simbol – informacija se dobiva kroz prikaz određenog reda simbola koji se svrstavaju u razinu odluke.

   Neke od ostalih razina apstrakcije su:
   - Niska razina fuzije – na ovoj razini uzimaju se podaci kakvi se dobiju sa senzora. Većim brojem očitanja istog senzora smanjuje se šum te mogućnost pogreške mjerenja.
   - Srednja razina fuzije – na temelju dobivenih podataka stvaraju se neke karakteristike ili značajke.
   - Visoka razina fuzije – na ovoj razini sustav je sposoban samostalno donositi odluke temeljene na podacima dobivenim iz okoline.

4. Klasifikacija fuzije podataka postavljena od strane američkog ministarstva obrane i Joint Directors of Laboratories vidljiva je na slici 2.4 koja je rađena prema [6]. Prema [6], ovo udruženje stvorilo je klasifikaciju fuzije podataka koja se najčešće koristi u današnje vrijeme. Njihova klasifikacija je podijeljena u pet razina obrade podataka, baze podataka i informacijske sabirnice koja povezuje sve navedene komponente.
   - Razina 0 - predobrada signala – najniža razina procesa fuzije podataka uključuje fuziju na razini signala i piksela. Na ovoj se razini dodatno reducira količina podataka i održavaju korisne informacije koje se koriste za procese viših razina.
   - Razina 1 – prerada objekata – u ovom se koraku prethodni dobiveni podaci raznim postupcima pretvaraju u konzistentne strukture podataka. Na izlazu se dobivaju klasificirani i identificirani podaci skupa s njihovim stanjem i orijentacijom.
   - Razina 2 – procjena situacije – na temelju dobivenih podataka i promatranih situacija ova razina fuzije podataka cilja na procjenu vjerojatnih događaja. Taj postupak kreće od uspostavljanja veza između podataka, pridodajući im određenu vrijednost ovisno u kojoj se okolini nalazi. Ova razina uspostavljanja veza pronalazi obrazac ponašanja objekata u okolini.
   - Razina 3 - procjena rizika – koristi predviđanja iz prijašnje razine, stavlja ih u određene situacije te promatra njihovo djelovanje na okolinu. Novonastala situacija se procjenjuje te se izračunavaju određeni rizici, njezine slabosti i prednosti.
   - Razina 4 – predobrada procesa – zadatak joj je preraspodjela zadataka kroz prijašnje četiri razine.



Sl. 2.3. Prikaz procesa fuzije podataka postavljena od strane Američkog Ministarstva obrane i Joint Directors of Laboratories.

Ovisno o potrebni pet komponenti se mogu podijeliti na fuziju visoke i niske razine, a prednosti navedenog koncepta su:

- Izvori – zasluženi su za prikupljanje podataka. Nije nužno da su svi izvori iste vrste već mogu biti sakupljeni iz različitih baza podataka, ručno unesenih informacija, podataka sa senzora i sl.
- Interakcija čovjeka i računala (engl. *human - computer interaction , HCI)* – sučelje koje operatoru omogućuje unose u sustav nakon čega sustav obavlja unesene zadatke.
- Upravljanje bazama podataka – bitan dio sustava jer pohranjuje širok spektar dobivenih podataka i fuzija kao rezultat njihovih fuzija.

5. Klasifikacija fuzije podataka prema tipu postavljene arhitekture
    - Centralizirana arhitektura – sva čvorišta za fuziju podataka nalaze se u središnjoj procesorskoj jedinici. U nju dolaze svi podaci sa senzora koji se zatim obrađuju. Ovaj tip je optimalan jedino uz pretpostavku da je raspored dolaska podataka ispravan kao i njihovo povezivanje te da vrijeme u kojem su podaci stigli nije važno. Najčešće to nije slučaj, jer je za prijenos čistih podataka potrebna velika propusnost i ona postaje usko grlo sustava.
    - Decentralizirana arhitektura – omogućuje uporabu više čvorova u kojem svaki ima svoju središnju procesorsku jedinicu i nema fuzije podataka u samo jednoj točki. Prednost ovakvog sustava je što svaki čvor skuplja i obrađuje podatke za područje za koje je namijenjen. Povećanjem broja čvorova pojavljuju se nedostaci ove arhitekture, a to je međupovezanost i komunikacija između čvorova. Većim brojem čvorova dolazi do otežane skalabilnosti.
    - Raspodijeljena arhitektura – obrada dobivenih mjerenja sa svakog čvora obavlja se odvojeno na svakom čvoru dok se tek obrađeni podaci šalju u čvor koji je zadužen za fuziju podataka. Prednost ove arhitekture je što se podaci s određenog područja obrađuju lokalno i šalju u čvor za fuziju podataka gdje ona povezuje više lokalnih čvorova i dobiva se jedan globalan pogled na cijeli sustav.
    - Hijerarhijska arhitektura – do nje se dolazi povezivanjem prethodno navedenih arhitektura poslaganih prema određenim razinama hijerarhije.

2.10. Internet stvari (IoT) i njegov razvoj

Internet stvari (engl. *Internet od Things, IoT*) se prema [7] najjednostavnije opisuje kao mreža fizičkih uređaja, vozila, zgrada i mnogih drugih objekata povezanih elektroničkim uređajima,

softverom, senzorima i mrežom koja omogućuje objektima skupljanje i izmjenu podataka. Od jednostavnog koncepta koji se je razvijao 15 godina Internet stvari prerastao je u arhitekturni okvir koji omogućuje unos i izmjenu podataka između fizičkog svijeta i računala preko postojeće mrežne infrastrukture.

Prema [7], pojam Interneta stvari prvi upotrebljava britanski pionir s područja tehnologije Kevin Ashton 1999. godine za opis sustava u kojem bi objekti iz fizičkog svijeta mogli biti spojeni na Internet koristeći se senzorima. Ashton je iskoristio termin da bi prikazao mogućnosti RFID (engl. *Radio-Frequency Identification*) oznaka koje su se u to vrijeme koristile u opskrbljivačkim lancima stavljajući na Internet stanje i praćenje pošiljke bez potrebe za ljudskim djelovanjem. Iako je Internet stvari poprilično nov, samo načelo pozivanja računala i mreža kako bi se nadgledali i kontrolirali uređaji postoji već desetljećima. Krajem 1970-ih godina već postoji mogućnost očitanja potrošnje električne struje putem telefonske linije. Od 1990. godine počinje se koristiti i u industrijskim pogonima kao dio opreme za nadzor i upravljanje putem bežičnog načina rada između strojeva (engl. *machine-to-machine, M2M*). Problem kod navedenog načina rada je rad mreže temeljen na industrijskom standardu, a ne na internet protokolu (IP) i njegovim standardima.

Ideja spajanja svakodnevnih objekata koji se koriste na Internet razvijala se neovisno o industriji te je pratila napredak mrežne infrastrukture.

2.11. Upotreba Interneta stvari

Internet stvari, objedinjuje mnoštvo objekata i pojava u stvarnom svijetu. Razvoj koji je postignut na tehnologiji Interneta primjenu je našao u medicini, poljoprivredi i stočarstvu, meteorologiji, a može se naći i u uređajima koji služe za pronalazak i spašavanje te mnogim drugim područjima čovjekovog djelovanja od kojeg su u daljnjem tekstu nabrojani samo neki.

1. Meteorologija

   Nadzor prirodnih uvjeta na nekom području postiže se korištenjem senzora poput onih za praćenje temperature i vlage zraka i zemlje, brzinu i smjer vjetra, količine padalina i razinu sunčeve svjetlosti. Prema [8], jednostavno i vrlo brzo mogu se dobiti informacije koje se mogu iskoristiti u mnoštvu budućih odluka. Unoseći podatke u bazu podataka prate se vremenski uvjeti te se kroz određeno vrijeme mogu imati obrasci vremenskih prilika na tom području.

2. Poljoprivreda i stočarstvo

   Prema [9], preuzimanjem podataka iz okoliša ili čak pregledom obrazaca vremenskih prilika na promatranom području vrlo se jednostavno može doći do zaključaka koje bi

kulture bile najuspješnije odnosno je li već postojeće kulture potrebno tretirati ili dohranjivati.

U stočarstvu Internet stvari postaje jedan od glavnih pokretača modernizacije. Ovisno o potrebama Internet stvari može se upotrijebiti za praćenje i nadzor krda u prirodnom uzgoju do čipiranja životinja i praćenja njihovih tjelesnih funkcija.

3. Sustavi za upravljanje potrošnje električne energije uz nadzor i automatizaciju

   Kako bi se mogla što pametnije iskoristiti, kreće uporaba pametnih kućanskim uređaja. Prema [10], uspješnom komunikacijom između senzora ugrađenih u prostoriju i aktuatora koji djeluju na određene sustave te njihovim razvojem korištenje električne energije moguće je svesti na točno onoliko koliko je potrebno. Ugradnjom senzora koji će prepoznavati razlike u temperaturi i količini svjetla između one vanjske i one u prostoriji, aktuatorima se upravlja jačina svjetla u kući, otvorenost roleta i prozora, otvorenost ventila na grijaćim tijelima ili jačina hlađenja na klimama.

4. Promet

   Nadzorom prometa, prema [11], Internet stvari omogućuje upoznavanje s njegovim ponašanjem. Praćenje stanja prometa, uvjeta na cestama, prometnih gužvi omogućuje uvid u stanje sustava te se može odmah reagirati kako bi se stanje popravilo. Sve je veći broj senzora koji su ugrađeni u vozila i koji imaju mogućnost komuniciranja s okolinom. Mogućnosti izgradnje takvog sustava postaju gotovo nepregledne: od komunikacije između vozila do komunikacije između automobila i sustava za nadzor prometa. Tu se mogu ubrojiti i naplata cestarine, upravljanje parkinzima i njihova naplatu te stvaranje pametne kontrole prometa i prometnica.

5. Medicina i zdravstvena zaštita

   Razvoj senzora za Internet stvari, prema [12], našao je još jednu granu u kojoj se može razvijati: praćenje tjelesnih funkcija u čovjeka kako bi liječnici odnosno specijalisti brzo i sigurno mogli doći do potrebnih informacija. Tu se mogu navesti primjene poput uređaja za mjerenje tlaka i praćenje rada srca. Vrlo se korisnim pokazao za nadzor osoba starije dobi. Neki složeniji primjeri mogu biti bežična kontrola i nadzor rada *pacemakera*.

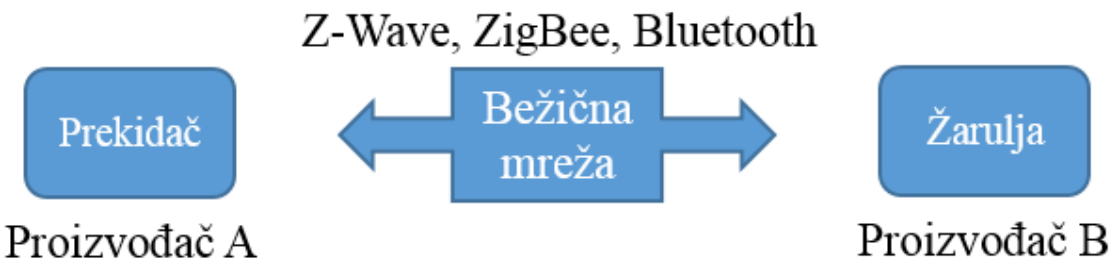
U diplomskom radu proučava se izrada sustava koji će se služiti Internetom stvari za nadzor meteoroloških uvjeta okoline u koju je postavljen. Sustav će podatke skupljen iz okoline bežičnim putem spremati u bazu na oblaku. Uređaj koji će prikazivati podatke iz baze biti će mobilni uređaj sa Android operacijskim sustavom.

2.12. Komunikacijski modeli Interneta stvari

Zbog što jednostavnijeg i bržeg povezivanja stvari 2015. godine *Internet Architecture Board, IAB* izdaje dokument RFC 7452 u kojemu se nalaze upute i pravila kako izraditi mrežu Interneta stvari.

1. Komunikacija među uređajima slikovito je prikazana na slici 2.5. napravljenoj prema [13].

Kod ovog modela su dva ili više uređaja direktno spojena i međusobno komuniciraju bez posrednika poput aplikacijskog poslužitelja. Za povezivanje se često koriste tehnologije poput Bluetootha, Z-Wavea i ZigBeea. Zbog male količine podataka koje uređaji koriste, ovaj oblik komunikacije najčešće se nalazi u sustavima za pametne kuće, odnosno gdje nema potrebe za složenim operacijama već onima poput upaliti i ugasiti prekidač za svjetlo, upravljanje termostatom i slično. Problem najčešće nastaje ako se želi povezati više tehnologija komunikacije koje nisu međusobno kompatibilne. To je i bio jedan od razloga za nastanak RFC 7452.



Sl. 2.4. Komunikacija uređaj - na - uređaj.

2. Komunikacija uređaj – na – oblak prikazan je na slici 2.6 napravljenoj prema uzoru iz [13].

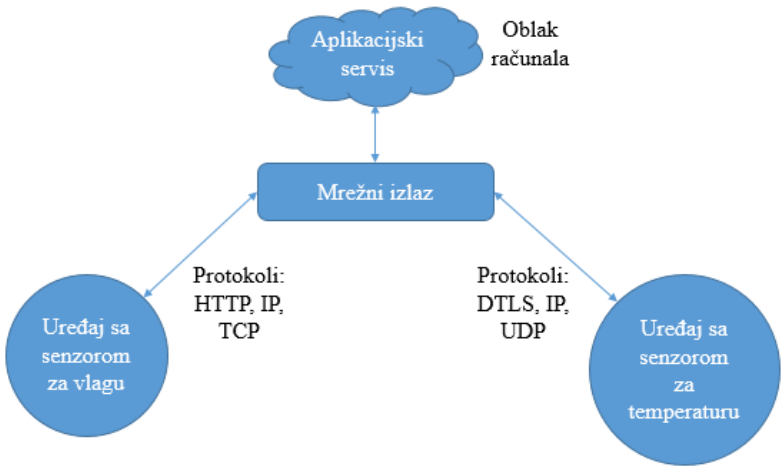
Koristeći neku vrstu usluge oblak računala, uređaji sa senzorima na sebi koriste standardni IP protokol kako bi dostavili pakete koji sadrže informaciju sa senzora. Povezivanje na oblak (engl. *cloud*) izvodi se ovisno o zahtjevima projekta, a ono može biti bežično ili putem mrežnog (engl. *ethernet*) kabela.

Sl. 2.5. Komunikacija uređaj - na - oblak.

Zbog složenosti sustava i načina korištenja dobivenih informacija ponovo se dolazi do problema ako se želi unijeti promjene ili ugraditi vlastito rješenje u postojeći sustav. Proizvođač je svoj proizvod osigurao tako što ga je „zaključao“ te svaka promjena na sustavu koja nije u skladu s pravilima upotrebe rezultira gubitkom jamstva ili podrške.

3. Model uređaj – na – posrednik protokola (engl. *gateway*)

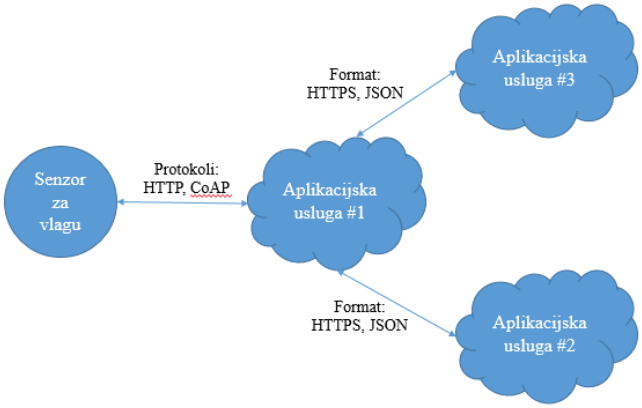
Kako bi se pojačala sigurnost i rasteretile operacije na oblaku, u sustav se uvodi *gateway* koji će poslužiti kao posrednik između uređaja sa senzorima i aplikacijske usluge na oblaku. Na njemu će se skupljati podaci, strukturirati u cjelinu, odrediti način prijenosa na oblak te osigurati dodatne provjere kako bi se povećala sigurnost sustava. Slika 2.7 izrađena prema [13] prikazuje komponente i protokole koji se koriste u tom modelu. Najčešću uporabu ovakvog modela pronalazi se u primjerima gdje uređaji nemaju direktnu vezu s oblakom već koriste posrednički uređaj kao izlaz na Internet. Kao primjer mogu se uzeti senzori vezani uz tjelovježbu. Podaci koji se skupljaju senzorima s tijela putem neke od bežičnih vrsti komunikacije poput Bluetootha šalju se na mobitel. On na sebi ima aplikaciju koja dobivene podatke šalje putem mobilnog interneta na oblak.
Prednost ovog modela je mogućnost dodavanja novih senzora s različitim oblicima komunikacije dok se kao najveći nedostatak navodi izrada aplikacijskog sloja na *gatewayu*.



Sl. 2.6. Model uređaj – na – gateway.

4. Model dijeljenja podataka s pozadinskih usluga

Arhitektura ovog modela prikazana na slici 2.8 izrađena je prema [13] te omogućuje krajnjem korisniku pristup bazi i njeno preuzimanje. Nakon što korisnik preuzme bazu, njome može manipulirati, obrađivati podatke ili jednostavno cijelu bazu implementirati u neku drugu aplikaciju koja će je drugačije obrađivati nego što je prvotno zamišljeno.



Sl. 2.7. Model dijeljenja podataka s pozadinskih servisa.

2.13. Veliki skupovi podataka

Prema [14], veliki skupovi podataka (engl. *big data*) kao pojam obuhvaća sve strukturirane, polustrukturirane i nestrukturirane podatke koji mogu biti korišteni kako bi se došlo do određene informacije. Termin *big data* počinje se koristiti oko 2005. godine kada ga počinje upotrebljavati O'Reilly Media.

U zadnjih nekoliko godina raste broj novih specijaliziranih firmi (engl. *start up*) vezanih uz velike skupove podataka koje pomažu raznim organizacijama kako bi shvatile što su veliki skupovi podataka i kako bi ih mogle implementirati u svoj sustav. Nakon što je sustav implementiran samo ostaje pitanje kako izvući informaciju koja nam je potrebna.

U ovom diplomskom radu veliki skupovi podataka stvaraju se zapisom sa senzora u bazu podataka. Kroz određeno vrijeme baza podataka će se moći uzeti kao referentna za područje sa kojeg se skupljaju podaci o klimatskim uvjetima.

2.14. Veliki skupovi podataka i fuzija podataka

Prema [15] i [16], može se reći da je fuzija podatka prvi uvjet za razvoj koncepta velikih skupova podataka koji se oslanja na veliku količinu različitih tipova podataka i međusobno koreliranih informacija iz dobivenih podataka. Različitim načinima i alatima podaci se organiziraju, strukturiraju, klasificiraju, sortiraju, obrađuju te prikazuju. Razvojem sustava velikih količina podataka razvijaju se i napredni matematički modeli i algoritmi, te alati i metode obrade i dobivanja korisnih informacija iz podataka. Neki od načina obrade i strukturiranja nepovezanih podataka u informativnu cjelinu je skup metoda pod nazivom strojno učenje (engl. *machine learning*) čime je iz skupa nepovezanih, ponekad i stohastičkih podataka moguće dobiti dobre modele koji opisuju prirodu i narav skupa podataka. Time se mogu modelirati i sustavi koji će

predviđati buduća stanja na temelju dostupnih podataka, odnosno služiti kao pomoć u odlučivanju. Budući da većina prikupljenih podataka često predstavlja podatke iz okoline što znači da se, osim periodnih procesa, radi o stohastičkim procesima s često promjenjivom dinamikom, moguće je stvoriti i adaptivne algoritme koji će sami sebe prilagođavati prema podacima koji dolaze. Strojno učenje postaje sve važniji skup metoda i u industrijskom okruženju.

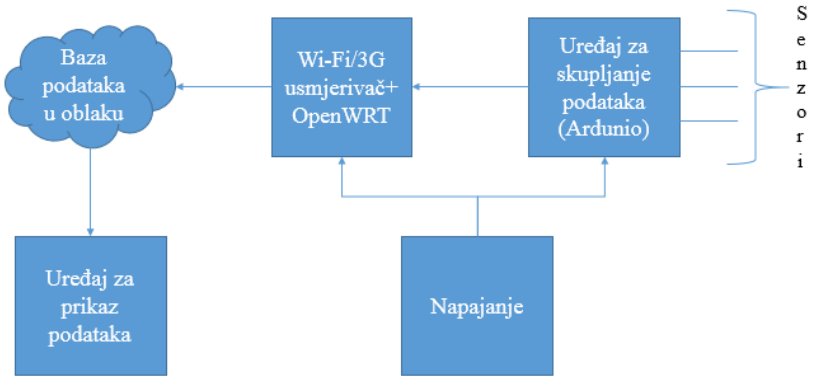
Način na koji se navedena tehnologija može koristiti u diplomskom radu je dobivanje virtualnih senzora. Koristeći unose iz baze podatak i njihovim uvrštavanjem u unaprijed napravljene formule moguće je dobiti dodatne informacije o klimatskim uvjetima okoline.

2.15. Veliki skupovi podataka i Internet stvari

Internet stvari generira veliku količinu različitih podataka i mjerenja što zapravo može činiti veliki skup podataka. Dio Interneta stvari u budućnosti bi mogao biti svaki uređaj koji posjedujemo odnosno uređaj koji će biti priključen na Internet i pružati određena mjerenja ili podatke, omogućavati interakciju s drugim uređajima, a sve u svrhu poboljšanja kvalitete života, informiranja, ubrzavanja financijskih operacija, potpore u odlučivanju, vođenju i slično.

3. ZAHTJEVI NA RAZVIJENI SUSTAV PRIKUPLJANJA I OBRADE PODATAKA

Razvoj sustava koji se razvija u sklopu ovog diplomskog rada treba rezultirati razvojem modularnog uređaja koji će se moći prilagoditi ovisno o tome u koju ga se okolinu postavlja. Njegova je zadaća sakupljati podatke iz okoline u stvarnom vremenu i ovisno na koji način korisnik odabere dobivene podatke i prikazati. Sustav koji će se testirati kao zadatak ovog diplomskog rada sastojat će se od sklopa koji će skupljati podatke iz okoline i omogućiti nam da dobivene podatke dobijemo u stvarnom vremenu kako bismo mogli reagirati ovisno o njihovim vrijednostima. Na slici 3.1 može se vidjeti sklopovska građa sustava.



Sl. 3.1. Blok shema sustava koji se izrađuje.

Dijelovi sustava su:

- Uređaj za skupljanje podataka – Arduino Mega 2560
- Senzori s kojih se skupljaju podaci
- WiFi/3G usmjerivač koji će prenositi podatke u bazu podataka
- MySQL baza podataka u oblaku računala
- Uređaj koji će prikazivati prikupljene podatke

Postupak prikupljanja podataka počinje očitavanjem iznosa sa senzora. Podaci sa senzora dolaze na Arduino Mega 2560 razvojnu pločicu gdje se pripremaju za slanje u bazu podataka. Usmjerivač (engl. *router*) koji je USB-om povezan s Arduinom s njega prima podatke i bežičnom mrežom internetom ih šalje u bazu podataka koja se nalazi u oblaku računala.

3.1. Funkcijski zahtjevi na izrađeni sustav

U funkcijskim zahtjevima na sustav razrađeni su zahtjevi koji se traže od pojedine komponente u sustavu kako bi se olakšao daljnji prolazak kroz zadatak.

1. Senzori – odluka koji će se senzori koristiti ovisi o tome koristi li ih se za dobivanje podataka iz okoline na vanjskom ili unutrašnjem području. U slučaju da se skupljaju iz vanjske okoline, senzori trebaju biti robusniji i bolje zaštićeni. Ako se želi dobiti što preciznije podatke, koristit će se kvalitetniji senzor. Ako će se dobivati podatci iz zatvorenih prostora, sama robusnost i zaštita nisu toliko bitni, ali zahtjevi za preciznost i kvalitetu i dalje postoje.
2. Uređaj za skupljanje podataka – Arduino, uređaj za koje su senzori konstruirani i koji će imati komunikaciju s njima naziva se Arduino Mega 2560. To je razvojna pločica čija je zadaća sakupiti i obraditi podatke u obliku spremnom za spremanje u bazu podataka.
3. WiFi/3G modem – mrežni uređaj koji će, ovisno gdje se nalazi, koje uvjete ima i s kojim podacima raspolaže, održavati povezanost na Internet. Vezu na Internet bit će moguće izvesti putem dostupne bežične Internet mreže ili ako nije dostupna može se koristiti bežični mobilni Internet.
4. Baza podataka u oblaku – MySQL baza nalazi se na oblaku Open Shift koji je napravila američka tvrtka Red Hat i besplatan je za korištenje.
5. Uređaj za prikaz podataka – podaci prikupljeni u bazu mogu se vizualizirati. Kako bi korisnik u svakom trenutku mogao dobiti potrebne podatke, koristit će se aplikacija za operacijski sustav Android.

3.2. Vremenski zahtjevi na izrađeni sustav

S obzirom na to da se radi o sustavima otvorenog koda koji su besplatni za korištenje, vremenski zahtjevi ne bi smjeli biti kritični. Ovisno o tome koja se fizikalna veličina promatra, tako, po važnosti se može odrediti vrijeme očitanja podataka sa senzora i njihovo slanje u bazu podataka.

Kod ostalih zahtjeva na sustav potrebno je znati u kojim će se uvjetima sklopovsko rješenje nalaziti te sukladno tome potrebno je prirediti zaštitu komponenti. Ako na sustav djeluju vanjske oborine, komponente koje se nalaze vani treba primjereno zaštititi.

4. SKLOPOVSKO RJEŠENJE SUSTAVA

Ovaj diplomski rad ujedno je analiza koncepta sustava koji skuplja podatke iz okoline i stavljati ih u bazu podataka kako bi korisnik dobio informaciju na mobilni uređaj o promjena u temperaturi i vlažnosti zraka i tla. Također, podaci koji se skupljaju u bazu podataka mogu se koristiti za praćenje vremena kroz duži period i na temelju dobivenih podataka za previđanje vremenskih uvjeta.

4.1. Sklop za obradu podataka

Prema [17], Arduino je tvrtka, zajednica i projekt koji se bavi proizvodnjom i razvijanjem sklopovlja i programa otvorenog koda. Sklopovlje sa svojim pokretačkim programima omogućuju raznim senzorima i aktuatorima prikupljanje podataka i upravljanje fizičkim objektima. Arduino je razvojna pločica s mikroupravljačem, a isti naziv nosi i IDE (engl. *Integrated Development Environment*) odnosno sučelje koje se koristi na računalu za programiranje mikroupravljača i postavljanje napisanog koda na isti. Pločica na sebi ima ulazno/izlazne nožice koji služe za razna proširenja koja dolaze na tiskanim pločicama i nazivaju se štitovi (engl. *shield*) i senzore/aktuatore koji se spajaju na njega.

Najčešće se primjere Arduino koristi u su projektima amatera koji prototipiraju razne robote, robotske ruke, nadzore, upravljanja raznim uređajima u kući, dvorištu i slične stvari gdje je dovoljno imati ideju, a ostvarenje može biti brzo i povoljno.

Ovisno o potrebama projekta, Arduino ima cijelu paletu proizvoda s nekim unaprijed osmišljenim mogućnostima. Želi li se spojiti Bluetooth, na njemu postoji mikroupravljač s već ugrađenim Bluetoothom na sebi. Isto to vrijedi i za LAN način povezivanja, WiFi povezivanje, ZigBee i sl.

S ciljem dodatne dogradnje postojećeg mikroupravljača bez lemljenja, napravljeni su tzv. shields. Oni proširuju mogućnost mikroupravljača, pa se određenim shield-om Arduino specijalizira za određenu zadaću. Tako postoje štitovi koji služe za upravljanje motorima na 3D pisaču, očitavanje GPS-a (Global Position System), za upravljanje robotskom rukom i razni drugi čiji broj samo raste zahvaljujući sve široj upotrebi Arduina.

4.2. Građa i mogućnosti Arduino Mega 2560 rev3

Mikroupravljač Arduino Mega poslužiti će za obradu podataka dobivenih sa senzora. Nakon dobivanja podataka na Arduino, njihov ispis se oblikuje za slanje u bazu na oblaku računala.

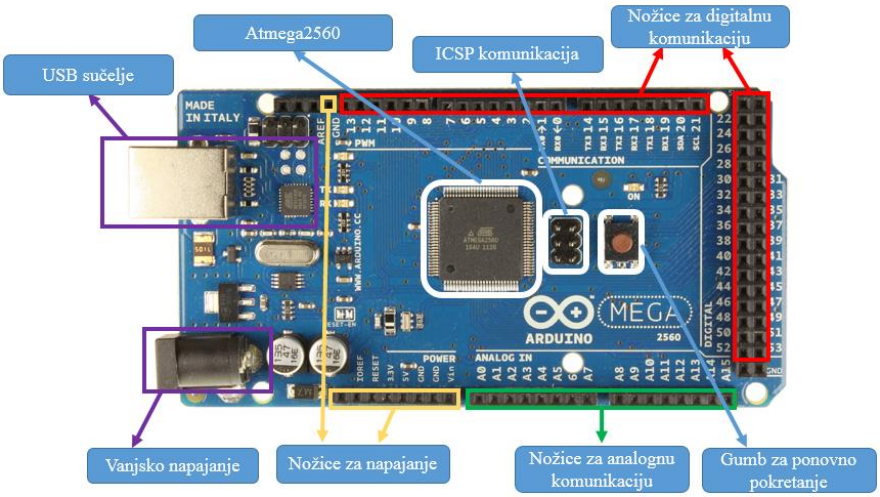
Zbog svoje jednostavnosti i lakoće obavljanja zadaće poput primanja informacija iz okoliša ili upravljanja fizičkim objektima Arduino vrlo brzo stječe popularnost. Informacije iz okoliša skuplja pomoću senzora, a upravljanje fizičkim objektima obavlja koristeći razna pomagala poput motora. Ugradnja programa u Arduino, radi se pomoću okuženja otvorenog koda u kojem se programiraju operacije koje će Arduino izvršavati u jeziku AVR C, a proširiti se može s C++ knjižicama.

U [18] se mogu vidjeti neke od bitnijih komponenti na Arduino Mega 2560, a to su njegova 54 digitalna ulaza/izlaza, 16 analognih ulaza, Atmel ATmega 2560 mikroupravljač, USB sučelje za komunikaciju, izvor vanjskog napajanja, *flash* memorija od 256 kB, tipka za ponovno pokretanje, i LED na digitalnoj nožici 13. Tablica 4.1 dodatno prikazuju neke od detalja preuzetih s [20].

Tab. 4.1. Tablica značajki mikroupravljača ATmega 2560.

Mikroupravljač	Atmel ATmega 2560
Napon	5V
Preporučeni ulazni napon	7 – 12V
Granični ulazni napon	6 – 20V
Broj digitalni ulaza/izlaza	54
Broj analognih ulaza	16
Jakost istosmjerne struje po I/O pinu	10 mA
Jakost istosmjerne struje za 3.3V pin	50 mA
Flash memorija	256
SRAM	8 kB
EEPROM	4kB
Radni takt	16 MHz

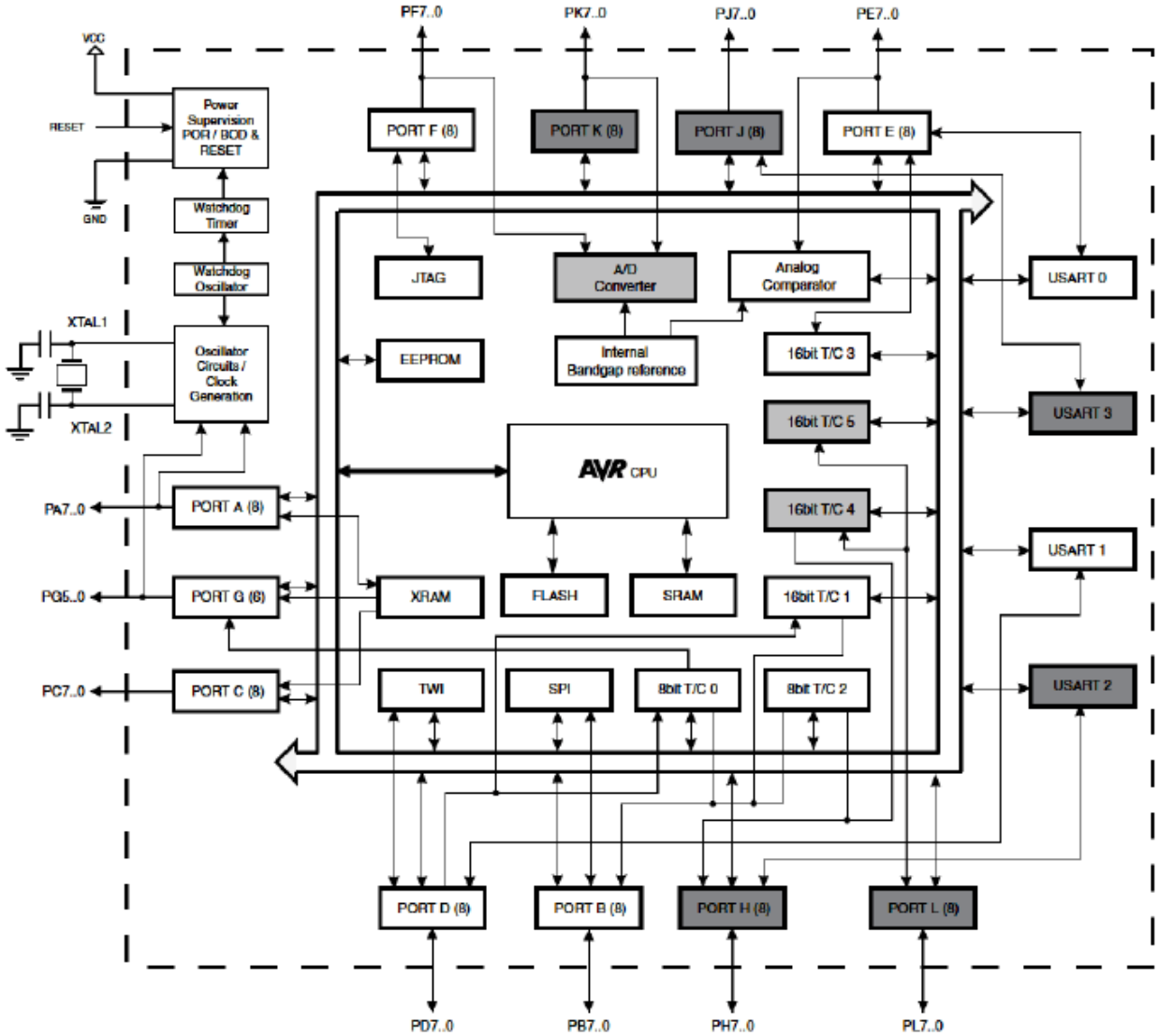
Slika 4.1 prikazuje glavne komponente od kojih se sastoji razvojna pločica. Izgled pločice varira ovisno o tome na kojoj se inačici pločice radi.



Sl. 4.1. Bitniji dijelovi na Arduino Mega 2560 razvojnoj pločici.

4.3. Razvojna pločica Arduino Mega 2560, mikroupravljač Atmel ATmega 2560

Iz [18] je vidljivo da je ATmega 2560 mikroupravljač visokih performansi i male potrošnje koji s Atmel-ovim 8-bitnim *RISC* (engl. *Reduced Intruction Set Computing*) mikroupravljačem koji sadrži 256 kB *ISP* (engl. *In-System programming*) *flash* memorije, 8 kB *SRAM*-a, 4 kB *EPROM*-a, 86 pinova opće namjene, 32 radna registra opće namjene, A/D pretvornik i radi na naponu 4,5 – 5,5V. Na slici 4.2 koja je preuzeta iz [19] može se vidjeti blok dijagram mikroupravljača.



Sl. 4.2. Blok shema mikroupravljača ATmega 2560.

4.4. Napajanje i komunikacija

Prema [18], napajanje Arduina može se izvesti pomoću predodređenog pristupa za vanjsko napajanje ili putem USB-a. Za vanjsko napajanje potreban je 2,1mm konektor sa središnje postavljenim pozitivnom polom. Ako postoji samo konektor bez adaptera koji ide u struju, može se poslužiti i baterijom tako da se spoji na GND i Vin pinove. Arduino može raditi na vanjskom napajanju koje će isporučivati 7 - 20V. Ako napajanje isporučuje manje od 5V napona, Arduino postaje nestabilan, dok u slučaju napajanja iznad12V napona, na njemu može doći do pregrijavanja i oštećenja.

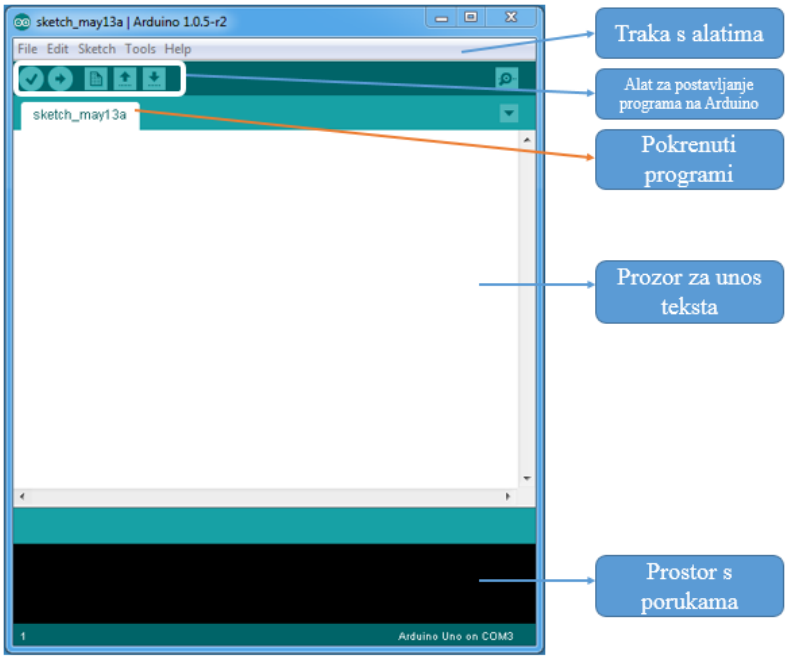
Komunikaciju na Arduino Mega 2560 razvojnoj pločici moguće je uspostaviti s računalom, s drugim mikroupravljačem ili s drugom Ardino pločicom. Na njemu se nalaze četiri UART (engl. *Universal Asynchronous Receiver/Transmitter*) za TTL (engl. *Transistot-transistor Logic*) serijsku komunikaciju. Dok starije revizije Arduino Mega 2560 za komunikaciju koriste USB – to – Serial, novije revizije koriste 16U2 ugrađene programe (engl. *firmware*). Za uspostavljanje veze s računalom koristi se USB protokol koji se na računalu pojavljuje kao virtualni CoM (engl. *Communication port*). Kako bi se osiguralo da Arduino šalje i prima podatke pri postavljanju programa na Arduino, prekidajuće će svijetliti će Tx i Rx LED diode.

4.5. Digitalni i analogni ulazi i izlazi

Arduino Mega 2560 zasnovan je na Atmelovom mikroupravljaču ATmega2560 te je samo jedan u paleti mikroupravljača koje proizvodi Arduino. Prema [18], na pločici se nalaze 53 digitalna ulaza ili izlaza, ovisno o tome kako se definiraju funkcije *pinMode()*, *digitalWrite()* i *digitalRead()*. Svaki od njih radi na 5 V, izdržava jakost električne struje od 20 mA i ima *pull-up* otpornik od 20-50Ω. Također ima specijalizirane nožica koji služe za serijsku vezu, nožice s PWM-om (engl. *Pulse Width Modulation*), nožice za ICSP zaglavlja, nožice na koje se mogu konfigurirati prekidi te nožice za TWI (engl. *Two Wire Communication*) komunikaciju koji koriste I2C sučelje. Osim digitalnih Arduino na sebi ima i 16 nožica koje služe za analognu komunikaciju.

4.6. Arduino razvojno okruženje

Arduino razvojno okruženje se prema [20] naziva Arduino Software IDE (engl. *Integrated Development Environment*) i služi za razvoj i postavljanje programa na Arduino (Sl. 4.3.). Sastoji se od prozora za unos teksta, područja gdje će stizati poruke vezano uz ponašanje programa na Arduinu i trake s alatima gdje Arduino se može dodatno podešavati.



Sl. 4.3. Razvojno sučelje Arduino IDE.

Arduino IDE se zasniva na programskom jeziku AVR C. Služi za pisanje, potvrđivanje i prebacivanje koda na Arduino pločicu. Prije nego što se krene s prebacivanjem potrebno je postaviti na kojem portu nam se nalazi Arduino, o kojoj je inačici Arduina riječ i koji procesor se nalazi na pločici. Nakon toga se u dvije glavne funkcije *setup*() i *loop*() upisuje željeni kod. Funkcija *setup*() pokreće se samo jednom i to pri pokretanju pločice ili njezinom ponovnom pokretanju. Kada se koristi neki od senzora ili štitova za Arduino, uz njih dolazi i njihova programska biblioteka (engl. *library*). U njemu se specificira raspored i uloga pojedine nožice/pina na Arduinu.

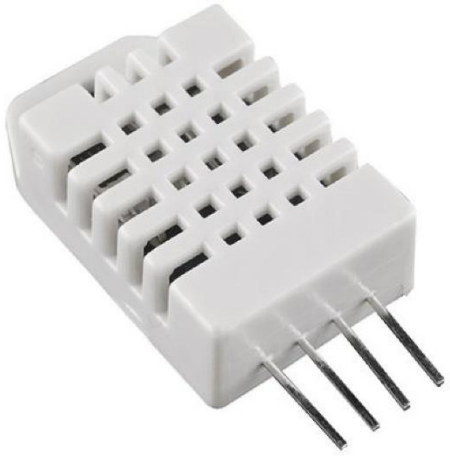
4.7. Sklop za prikupljanje podataka

Sklop za prikupljanje podataka složen je od komponenti dostupnih u trgovinama elektroničkom opremom. Za svaku od komponenti postoji gotova programska podrška koja mjerne

veličine može prikazati na razvojnoj pločici Arduino, te njima rukovati iz koda u Arduino IDE.

4.8. Senzor za praćenje temperature i vlage zraka

Senzor za praćenje temperature zraka je oznake DHT 22 i prikazan je na slici 4.4 preuzete sa [21]. Mjerne veličine ovog senzora su unaprijed kvantizirane gotovim programskim rješenjem koje se poziva unutar koda Arduino projekta. Veličine koje senzor mjeri su temperatura i vlaga zraka.



Sl. 4.4. Senzor za mjerenje temperature i vlage DHT22.

U tablici 4.2. prikazane su značajke senzora DHT22 preuzete sa [22].

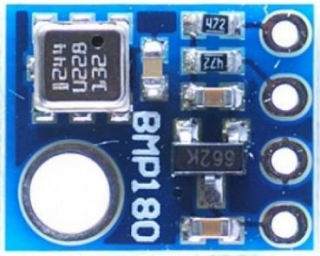
Tab. 4.2. Značajke senzora DHT22.

Model	DHT22
Napajanje	3.3 – 6V
Raspon mjerenja	0 – 100% RH ; -40 - +80°C
Točnost	+/- 2% RH ; +/- 0.5°C
Stabilnost i gubici	+/- 0.5% RH/ godišnje

4.9. Senzor za praćenje tlaka zraka

Senzor za praćenje tlaka zraka prikazan je na slici 4.5 preuzete sa [23] te ima standardnu oznaku BMP180. Programska podrška također već postoji za Arduino razvojno okruženje te je senzor potrebno spojiti pomoću jedne od nekoliko podržanih komunikacijskih protokola i veza na Arduino prototipnu pločicu. U radu je korišten I2C komunikacijski protokol pomoću dvije žice.

Posebnosti ovog senzora su osim mogućnosti da mjeri barometarski tlak i mogućnost mjerenja nadmorske visine.



Sl. 4.5. Senzor za praćenje tlaka zraka BMP180.

Značajke koje su prikazane u tablici 4.3 preuzete su sa [24].

Tab. 4.3. Značajke senzora BMP180.

Model	BMP180
Napajanje	1.8 – 3.6V
Raspon mjerenja	300 – 1100 hPa
Točnost	100% u rasponu 0 – 65 °C

4.10. Senzor za mjerenje temperature zemlje

Senzor za mjerenje temperature zemlje je digitalni senzor Texas Instruments DS18820 i prikazan je na slici 4.6. preuzetoj s [25]. Senzor je vodootporan i rađen prema industrijskim standardima. Za komunikaciju je korišten One Wire komunikacijski protokol uz postojeću programsku podršku digitalnog integriranog kruga senzora i komunikacijskog protokola.

Sl. 4.6. Senzor za mjerenje temperature zemlje DS18B20.

Njegove značajke opisane su u tablici 4.4 i preuzete su sa [26].

Tab. 4.4.Značajke senzora DS18B20.

Model	DS18B20
Napajanje	3.0 – 5.5 V
Raspon mjerenja	-55 - +125 °C
Točnost	+/- 0.5°C u rasponu od -10 do +85°C

4.11. Senzor za praćenje vlage zemlje

Senzor korišten za mjerenje vlage zemlje je analogni senzor proizvođača SparkFun i prikazan je na slici 4.7 preuzete sa [27]. Senzor radi na primitivnom principu mjerenja varijabilnog otpora ovisnog o protoku struje između dvije galvanizirane elektrode. Za ovaj senzor postoji i pločica s integriranim krugom i A/D pretvornikom koji daje digitalni signal na Arduino prototipnu pločicu.

Sl. 4.7. Senzor za praćenje vlage zemlje LM393.

Njegove karakteristike koje su opisane u tablici 4.5 preuzete su sa [28].

Tab. 4.5. Značajke senzora za praćenje vlage zemlje.

Model	Senzor i LM393
Napajanje	3.3 – 5.5 V
Dubina mjerenja	-37 mm
Točnost	+/- 0.5° RH u rasponu od 0 do +40°C

4.12. Komunikacijski dio

U ovom poglavlju opisat ćemo proces komunikacije između uređaja koji prikuplja podatke iz okoline i baze smještene u oblaku računala.

4.13. Usmjerivač TP-Link 703n

TP – Link 703 je povoljni mrežni usmjerivač koji ima podršku za Open WRT Linux operacijski sustav [29]. Također posjeduje jedan USB 2.0. serijski komunikacijski ulaz što omogućava jednostavnu vezu s Arduino razvojnom pločicom. Budući da je kroz Open WRT omogućena izmjena sustavskih postavki, uz postojeće FTDI pogonske programe koji su potrebni zbog korištenja FTDI kontrolera za USB komunikaciju na Arduino Mega pločici, moguće je serijskom vezom dobiti mjerne podatke s Arduino pločice u Linux konzolu. Time je omogućeno da se formatirani niz alfanumeričkih znakova (engl. *string*) jednostavno pretvori u HTTP GET zahtjev prema kreiranom API-ju.

Pristup na Internet omogućen je 3G USB modemom koji se uz Arduino USB mora priključiti na USB sabirnik (engl. *USB Hub)*. Pri tome je potrebno paziti da su Hub, Arduino, te FTDI čip usklađeni prema verziji USB protokola.

```
#!/bin/sh /etc/rc.common
START=91
start() {
cat /dev/ttyACM0 | while read LINE  // Cat znači da otvara serijsku vezu i čita njen stream
do
wget "$LINE" -O /dev/null > /dev/null
done
}
```

Programski kod 4.1. Povezivanje Arduino razvojne pločice na T-Link usmjerivač.

Naredba Cat otvara serijsku vezu i čita njen niz znakova. Svaka linija poziva wget naredbu koja je zapravo GET HTTP zahtjev za preuzimanjem određenog skupa podataka s interneta, te sprema odgovor poslužitelja u /dev/null koji je zapravo terminator odnosno prazan objekt, gdje se gubi svaki zapis odnosno briše iz sustava.

```
chmod +x arduinoStartScript
/etc/init.d/ arduinoStartScript start
/etc/init.d/ arduinoStartScript enable
```

Programski kod 4.2. Pokretanje skripte za spajanje pri svakom paljenju TP-Link usmjerivača.

4.14. 3G stick za mobilni internet

Uređaj se na Arduino TP-Link 703n spaja pomoću USB veze, te je potrebno ispravno podesiti i instalirati pogonske programe na Open WRT uređaj. Kako bi se spajanje na Internet moglo odvijati na OpenWRT operacijskom sustavu, potrebno je podesiti pristupne podatke GPRS ili 3G. Potrebno je provjeriti proizvođače i modele 3G i 4G USB modema jesu li podržani unutar operacijskog sustava Open WRT i postoje li za njih pogonski programi. Konfiguracija mreže

također ovisi o tome je li modem zaključan na određenu mrežu ili nije. Ako nije, potrebno je u *firmware* modema upisati parametre mreže čija se kartica koristi u USB modemu. Ispravnom konfiguracijom i testnim spajanjem dobiva se pristup internetu koji se može provjeriti kroz konzolu Open WRT-a TP Link 703n uređaja te naredbom npr.: ping google.com.

5. PROGRAMSKO RJEŠENJE SUSTAVA

5.1. Programska podrška prikupljanja podataka

Prikupljanje podataka kontrolira programska podrška Arduino razvojnog okruženja. Za svaki senzor je napisana biblioteka ili neka druga programska podrška koja je jednostavno implementirana u program kojeg izvršava Arduino programska pločica.

5.2. Programski kod za Arduino platformu

Program Arduino pločice je skup naredbi koje inicijaliziraju vezu sa svakim od dostupnih senzora korištenjem njihovih biblioteka ili dostupnih naredbi ugrađenih u Arduino razvojno okruženje. Nakon uspješne inicijalizacije komunikacije, podaci se prikupljaju na nekoliko načina. Ako se radi o analognom sklopu, najčešće je potrebno napisati vlastitu jednostavnu programsku biblioteku ili skup naredbi kako bi se veličine kvantizirane A/D programskim sklopom pretvorile u razumljive i standardizirane mjerne veličine. Nakon toga sve prikupljene veličine agregiraju u alfanumerički niz koji označava potpuni URL s podacima koji će biti iskorišten za HTTP GET zahtjev prema API-ju na poslužitelju.

Arduino URL šalje kao niz znakova USB komunikacijom prema TP Linku i Open WRT-u

```
#include <dht.h>
#include <OneWire.h>
#include <DallasTemperature.h>

#define DHTPIN 2
#define DHTTYPE DHT22   // DHT 22  (AM2302)
#define ONE_WIRE_BUS 2

dht DHT;

OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);

int HumMoisturePin = A1;

int i = 0;
int s1; // temperatura zraka
int s2; // vlažnost zraka, C
int s3; // temperatura zemlje
int s4; // vlažnost zemlje

void setup() {
  Serial.begin(9600);
  dht.begin();
  DallasSensors.begin();
}
```

```
void loop ()
{
  // DHT22
  DHT.read11(A0);
  s1 = (int) DHT.temperature; // ocitavanje temperature i spremanje vrijednosti u s1 varijablu
  s2 = (int) DHT.humidity; // ocitavanje vlažnosti i spremanje vrijednosti u s2 varijablu
  s3 = (int) DallasSensors.requestTemperatures();
  s4 = analogRead(HumMoisturePin);

  Serial.print("http://arduino-igreguric.rhcloud.com/?prvi=");
  Serial.print(s1);
  Serial.print("&drugi=");
  Serial.print(s2);
  Serial.print("&treci=");
  Serial.print(s3),
  Serial.print("&cetvrti=");
  Serial.print(s4);
  Serial.print("\n");

}
```

Programski kod 5.1. Arduino kod za skupljanje podataka sa senzora i kreiranje poveznice.

5.3. Podrška na poslužitelju

Za podršku na poslužitelju korištena je Red Hat Cloud Open Shift pohrana u oblaku računala. Kreiran je virtualni računalni stroj iz prethodno konfiguriranih predložaka koji se mogu izabrati iz izbornika nadzornog sučelja Open Shift sustava. Posebnost ovakve konfiguracije virtualnog računalnog stroja je što ima instaliranu Linux poslužiteljsku distribuciju s unaprijed podešenom Apache HTTP programskom podrškom, konfiguriranu MySQL bazu te instaliran program PhpMyAdmin za jednostavno upravljanje MySQL bazama, kao i podešen GIT repozitorij na koji se jednostavno postavljaju razvijeni kodovi. Prilikom instanciranja virtualnog stroja dobivaju se potrebni pristupni podaci te je time olakšan i ubrzan razvoj.

5.4. Programski kod PHP za prikupljanje podataka

Na strani poslužitelja bilo je potrebno napisati aplikacijsko programsko sučelje (engl. *Application programming interface* - *API*) koje će podatke dobivene GET HTTP zahtjevom iz atributa URL-a pročitati i spremiti u za to prethodno kreiranu MySQL bazu.
URL se sastoji od osnovnog URL-a, putanje do .php skripte API-ja i parametara ili atributa URL-a koji su označeni imenima, kao i vrijednostima koje se pohranjuju u bazu.

```
http://arduino-igreguri.rhcloud.com/?prvi=22&drugi=40&treci=20&cetvrti=35
```

Programski kod 5.2. Primjer URL-a za slanje podataka.

5.5. Programski kod za pohranu podataka u bazu MySQL

Nakon što su podaci prikupljeni s URL-a, potrebno ih je spremiti u bazu MySQL. Za to je kreirana baza imena *arduino* s tablicama *devices*. Tablice sadrže atribute: s1, s2, s3, s4 tipa *integer* veličine od -2147483648 do 2147483647.

```
<?php
define('DB_NAME', $_ENV['OPENSHIFT_APP_NAME']);
define('DB_USER', $_ENV['OPENSHIFT_MYSQL_DB_USERNAME']);
define('DB_PASSWORD', $_ENV['OPENSHIFT_MYSQL_DB_PASSWORD']);
define('DB_HOST', $_ENV['OPENSHIFT_MYSQL_DB_HOST'] . ':' .
$_ENV['OPENSHIFT_MYSQL_DB_PORT']);

define('DB_CHARSET', 'utf8');

define('DB_COLLATE', '');

$prvi = $_GET['prvi'];
$drugi = $_GET['drugi'];
$treci = $_GET['treci'];
$cetvrti = $_GET['cetvrti'];

echo $prvi.", ".$drugi.", ".$treci.", ".$cetvrti;


$con=mysql_connect(DB_HOST,DB_USER,DB_PASSWORD) or die("cannot connect");
mysql_select_db(DB_NAME)or die("cannot select DB");
if(!empty($prvi) || !empty($drugi) || !empty($treci) || !empty($cetvrti))
{
	$sql = "INSERT INTO devices (s1, s2, s3, s4) VALUES ('$prvi', '$drugi', '$treci', '$cetvrti')";
}
else
{
	echo "variables are not set";
}
$result = mysql_query($sql);
mysql_close($con);


?>
```

Programski kod 5.3. Pohrana podataka u bazu MySQL.

5.6. Operacijski sustav OpenWRT

OpenWRT je, prema [30], GNU/Linux distribucija prvenstveno namijenjena za ugrađene (engl. *embedded*) uređaje poput usmjerivača. Budući da je rađen na Linux jezgri, u mogućnosti je koristiti širok spektar biblioteka namijenjenih za isti. Tu se dobivaju nove mogućnosti uređaja te se samostalno implementira modularnost uređaja. Isto tako, zbog svoje otvorenosti pruža korisniku

slobodu samostalnog uređivanja *firmwarea*. Iako je sustav kao takav kada ga se preuzme poprilično ogoljen, on pruža mogućnost korisniku da samostalno implementira funkcionalnost koja mu je potrebna od uređaja. Ništa od navedenog ne bi bilo moguće da zajednica nije cijelo vrijeme tijekom razvoja pružala i proširivala dokumentaciju koja je vrlo detaljna i precizna tako da pruža korisniku brz i kvalitetan uvid u samo funkcioniranje sustava.

5.7. Programsko rješenje za prikaz rezultata mjerenja na operacijskom sustavu Android

Za prikaz skupljenih podataka iz okoline koriste se uređaji na kojima se nalazi operacijski sustav Android. Izbor korištenja operacijskog sustava Android je njegova otvorenost i mogućnost testiranja različitih vrsti komunikacije s bazom podataka te sukladno tome izbor najpogodnijeg za promatrani slučaj.

5.8. Operacijski sustav Android

Prema [31], Android nastaje krajem 2003. godine kada se na Linux jezgru dodaju razni alati rađeni u Javi kako bi operacijskom sustavu dodali mogućnost postavljanja korisničkog sučelja i komunikacije između različitih aplikacija zaduženih za zadaće poput uspostave poziva, slanja SMS poruka, pokretanja kamere i ostale. Aplikacije su se izrađivale u razvojnom okruženju Eclipse ubacivanjem dodatnog programskog paketa (engl. *Software development kit, SDK*). SDK je skupina razvojnih alata koji se koriste za razvoj Android aplikacija. Svakom novom inačicom operacijskog sustava Android izlazi novi SDK.

Operacijski sustav Android koristi se u radu kako bi se prikazali dobiveni podaci sa senzora te pobliže upoznao razvoj same aplikacije u Androidu koristeći uputstva Googleovog razvojnog tima.

5.9. Programski jezik Java

Java je objektno orijentirani programski jezik razvijen u američkoj tvrtci Sun Microsystems. Programi pisani u Javi mogu se pokretati na bilo kojem operacijskom sustavu koji na sebi ima instaliran pokretački potprogram Java virtualni stroj (engl. *Java Virtual Machine*, *JVM*). Korisnik ovisno o svojim potrebama ima mogućnost instaliranja Java pokretačke okoline (engl. *Java Runtime Environment, JRE*) i Java razvojne okoline (engl. *Java Development Kit, JDK*).

Razlog zbog kojeg je Android odabrao programski jezik Java je što, unatoč svojoj dostupnosti

pruža visoku razinu sigurnosti i pouzdanosti. Nedostatak koji nastaje u pružanju navedenih usluga je vrijeme koje postaje potrebno za izvršavanje zadataka. U ovom radu, programski jezik Java koristi se kao osnova za razvoj Android aplikacije. Aplikacija se na mobilnom uređaju pokreće preko Java virtualnog stroja.

5.10. Android IDE

Za Android razvojno okruženje (eng. *Integrated Development Environment*, *IDE*) prema [26] do 2014. koristi se Eclipse u koji su se implementirali paketi razvojnih alata. U prosincu 2014. Google izdaje službenu inačicu razvojnog okruženja pod nazivom Android Studio.

Prelazak na novo razvojno okruženje donosi novi pristup samom razvoju aplikacija, a isto tako i upravljanjem upravljačkim potprogramima. Dio razvojnog okruženja zadužen na nadzor samog procesa razvoja aplikacije naziva se Gradle. Zbog velikog raslojavanja verzija operacijskog sustava Android pojavila se potreba za nadzorom korištenja raznih biblioteka koje se koriste. Neke od njih postaju zastarjele, neke druge ih mijenjaju, a odgovoran za njihovo korištenje postaje Lint, alat za nadzor performansi, korištenja i kompatibilnosti inačica biblioteka. Neki od bitnijih alata koji su došli s Android Studiom su ProGuard (nadzor ugradnje dozvola za korištenje aplikacije), alat za izradu korisničkog sučelja s raznim dodacima, podrška za razvoj aplikacija koje će raditi na Android uređajima koji nisu nužno mobilni telefoni te potpuna integracija s Google uslugama u oblaku (engl. *Google Cloud Platform*).

5.11. Biblioteka za mrežnu komunikaciju

Za uspostavljanje komunikacije između mobilnog uređaja s operacijskim sustavom Android i baze u kojoj se nalaze skupljeni podaci koristimo Volley biblioteku. Biblioteka je dostupna u službenom Googleovom repozitoriju i kao takva može se implementirati izravno iz Android Studia.

Prema [32], Volley je Android biblioteka koja se koristi za mrežnu komunikaciju putem HTTP prijenosnog protokola (engl. *Hypertext Transfer Protocol*, HTTP).

5.12. Programsko rješenje mobilne aplikacije

Za prikaz podataka dobivenih iz okoline i spremljenih u MySQL bazu izrađena je u Android Studiu Android aplikacija.

5.13. Korisničko sučelje

Jednostavno korisničko sučelje razvijeno je implementacijom određenog broja rasporeda (engl. *layout*). U svakom pojedinom rasporedu definirane su komponente aplikacije koje će se prikazati na zaslonu (engl. *display*) ovisno u kojem se dijelu aplikacije nalazi. Kao primjer takve vrste rasporeda može se vidjeti programski kod 5.4.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   xmlns:tools="http://schemas.android.com/tools"
   android:orientation="vertical">

   <RelativeLayout
      android:layout_gravity="center"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content">


   <TextView
         tools:text="34"
         android:id="@id/s1"
         android:textSize="40dp"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:paddingTop="80dp" />

      <TextView
         android:layout_toEndOf="@+id/s1"
         android:layout_toRightOf="@+id/s1"
         android:text="@string/celsius"
         android:textSize="40sp"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:paddingTop="80dp" />

   </RelativeLayout>

   <TextView
      android:id="@id/date"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:paddingTop="10dp"/>

   <TextView
      android:text="@string/air_temp"
      android:textSize="20sp"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:paddingTop="80dp"
      android:paddingBottom="60dp" />
```

```
    <Button
        android:id="@id/refresh"
        android:text="@string/refresh"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"/>

</LinearLayout>
```

Programski kod 5.4. Primjer rasporeda spremnog za prikaz na zaslon.

Nije nužno da postoje rasporedi koji će služiti samo za prikaz. Postoje rasporedi koji se koriste kako bi se implementirala drugačija ponašanja i upravljanje aplikacijom. U programskom kodu 5.5 vidi se rješenje izbornika koji se pojavljuje s lijeve strane i definirani prostor u koji će se postavljati rasporedi s izmjerenim podacima. Za orijentiranje po aplikaciji odabrana je navigacijska ladica odnosno izbornik koji je definiran s lijeve strane zaslona koji se pojavljuje dodirom na prečac u alatnoj traci ili jednostavnim pokretnom s lijeve na desnu stranu pokazivača.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/white"
    android:orientation="vertical">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@color/android_green" />

    <android.support.v4.widget.DrawerLayout
        android:id="@id/drawer_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <FrameLayout
            android:id="@id/fragment_container"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />

        <ListView
            android:id="@id/drawer_list"
            android:layout_width="240dp"
            android:layout_height="match_parent"
            android:layout_gravity="start"
            android:background="@color/android_green" />

    </android.support.v4.widget.DrawerLayout>
</LinearLayout>
```

Programski kod 5.5. Primjer rasporeda za proširivanje funkcionalnosti.

Kako bi se napravilo sučelje koje će biti jednostavno za izmijeniti, sav tekst koji se vidi u izbornicima, gumbima i raznim tekstualnim okvirima definira se u izborniku sastava (engl.

strings), koji prikazuje programski kod 5.6.

```
<resources>
    <string name="app_name">Weather_app</string>
    <string name="action_settings">Settings</string>
    <string name="air_temp">Temperatura zraka</string>
    <string name="air_humidity">Vlaznost zraka</string>
    <string name="soil_temp">Temperatura zemlje</string>
    <string name="soil_humidity">Vlaznost zemlje</string>
    <string name="date">Vrijeme ocitanja</string>
    <string name="refresh">Osvjezi podatke</string>
    <string name="celsius">°C</string>
    <string name="humidity">%</string>
</resources>
```

Programski kod 5.6. Primjer sastava za lakše upravljanje izbornicima.

5.14. Upravljačko sučelje

Upravljačko sučelje izrađeno je prema MVP modelu (engl. *model-view-presenter*, MVP) koji je izabran zbog svoje modularnosti i mogućnosti testiranja svakog koraka. Modularnost se odnosi na raslojavanje same aplikacije, a to su:

a) Izgled (engl. *View*) je raspored koji je spomenut u korisničkom sučelju. Služi za raspored komponenti koje će kasnije biti vidljive korisniku.
b) Model (engl. *model*) je aplikacije koji služi za povezivanje s bazom i skupljanjem podataka.
c) Davatelj (engl. *presenter*) se brine da dobiveni i obrađeni podaci pravim putom dođu na svoje mjesto koje je rasporedom predviđeno na korisničko sučelje.

Mogućnost odvajanja pozadinskih procesa od onih na sučelju čini aplikaciju puno neovisnijom na utjecaje koji se pojavljuju dok aplikacija prolazi kroz izvođenje. Održavanje aplikacije izvedene u MVP modelu uvelike je olakšano što zbog razdvojenosti određenih dijelova što zbog količine napisanog koda. Testiranje koda i dijelova same aplikacije se isto puno jednostavnije izvodi, a budući da je model izrade aplikacije široko upotrebljavan, postoje unaprijed razrađeni testovi koji se brzo i pouzdano mogu implementirati.

Rasporedi su obrađeni u prijašnjoj cjelini tako da će se u ovom dijelu prikazati bitni dijelovi koda aplikacije koji u ovom slučaju služe kao modeli i davaoci.

Kao model kreirana je Java datoteka naziva Data koja se može vidjeti u programskom kodu 5.7. U njoj se pripremaju podaci koji dolaze iz Volley komunikacijske biblioteke.

```
package com.ferit.ivan.weather_app.main.model;

import com.google.gson.annotations.SerializedName;

import java.io.Serializable;
```

```
/**
 * Created by ivan on 7/14/16.
 */
public class Data implements Serializable {

    private long id;
    private String date;
    private String s1;
    private String s2;
    private String s3;
    private String s4;

    public long getId() {
        return id;
    }

    public String getDate() {
        return date;
    }

    public String getS1() {
        return s1;
    }

    public String getS2() {
        return s2;
    }

    public String getS3() {
        return s3;
    }

    public String getS4() {
        return s4;
    }
}
```

Programski kod 5.7. Java datoteka koja sprema podatke primljene s poslužiteljske strane.

Za slučaj nedostupnosti poslužitelja, kako ne bi došlo do gašenja aplikacije i kako bi se održao njen životni ciklus, postavljena je i adapter klasa vidljiva u programskom kodu 5.8.

```
package com.ferit.ivan.weather_app.main.adapter;

import android.content.Context;
import android.support.annotation.NonNull;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

import com.ferit.ivan.weather_app.R;
import com.ferit.ivan.weather_app.main.model.Data;

import java.util.ArrayList;
import java.util.List;
```

```
/**
 * Created by ivan on 7/27/16.
 */
public class DataListAdapter extends BaseAdapter {

    private final List<Data> mSensorsData = new ArrayList<>();

    public void setSensors(@NonNull List<Data> sensordata) {
        mSensorsData.clear();
        mSensorsData.addAll(sensordata);
        notifyDataSetChanged();
    }

    @Override
    public int getCount() {
        return mSensorsData.size();
    }

    @Override
    public Data getItem(int position) {
        return mSensorsData.get(position);
    }

    @Override
    public long getItemId(int position) {
        return mSensorsData.get(position).getId();
    }

    @Override
    public boolean hasStableIds() {
        return true;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        ViewHolder holder;

        if (convertView == null) {
            convertView = LayoutInflater.from(parent.getContext()).inflate(R.layout.show_data_item, parent, false);
            holder = new ViewHolder();
            holder.mDataTextView = (TextView) convertView.findViewById(R.id.data);
            holder.mDateTextView = (TextView) convertView.findViewById(R.id.date);
            holder.mDataExplanationTextView = (TextView) convertView.findViewById(R.id.data_explanation);
            convertView.setTag(holder);
        } else {
            holder = (ViewHolder) convertView.getTag();
        }

        Data item = getItem(position);
        holder.mDataTextView.setText(item.getS1());
        holder.mDataTextView.setText(item.getS2());
        holder.mDataTextView.setText(item.getS3());
        holder.mDataTextView.setText(item.getS4());
        holder.mDateTextView.setText(item.getDate());
        holder.mDataExplanationTextView.setText(R.string.air_temp);
        holder.mDataExplanationTextView.setText(R.string.air_humidity);
        holder.mDataExplanationTextView.setText(R.string.soil_temp);
        holder.mDataExplanationTextView.setText(R.string.soil_humidity);

        return convertView;
```

```
    }

    private static class ViewHolder {
        private TextView mDataTextView;
        private TextView mDateTextView;
        private TextView mDataExplanationTextView;
    }
}
```

Programski kod 5.8. Adapter klasa.

Nakon što su podaci primljeni i spremljeni u unaprijed određene varijable, potrebno ih je postaviti rasporedom na određena mjesta kako bi bili vidljivi korisniku. Za to je zadužen fragment koji se ugrađuje kao dio djelovanja (engl. *activity*) u aplikaciji. Fragment je dio *activityja* koji prati njegov životni ciklus, ali omogućuje korištenje više istih u jednom *activityju*. Sukladno s tim smanjuje se količina potrebne memorije za korištenje aplikacije te njezino vrijeme odaziva. Fragment koji se koristi u Android aplikaciji vidljiv je u programskom kodu 5.9.

```
package com.ferit.ivan.weather_app.main.fragment;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ListView;
import android.widget.TextView;

import com.ferit.ivan.weather_app.R;
import com.ferit.ivan.weather_app.base.BaseFragment;
import com.ferit.ivan.weather_app.commons.requests.BackendRequest;
import com.ferit.ivan.weather_app.commons.volley.ResponseListener;
import com.ferit.ivan.weather_app.commons.volley.VolleyErrorHelper;
import com.ferit.ivan.weather_app.main.adapter.DataListAdapter;
import com.ferit.ivan.weather_app.main.model.Data;

import java.io.Serializable;
import java.util.List;

/**
 * Created by ivan on 8/9/16.
 */
public class ShowAirTempFragment extends BaseFragment implements View.OnClickListener {

    public static final String BUNDLE_DATA = "bundle_data";

    private Data mWeatherData;

    private TextView mTempTxt;
    private TextView mDateTxt;

    public static Fragment newInstance(Data data) {
```

```
        Fragment fragment = new ShowAirTempFragment();
        Bundle args = new Bundle();
        args.putSerializable(BUNDLE_DATA, data);
        fragment.setArguments(args);
        return fragment;
    }

    @Override
    public void onActivityCreated(@Nullable Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        prepareData();
    }

    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.fragment_airtemp_data, container, false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        prepareUI(view);
    }

    @Override
    protected void prepareUI(@NonNull View layoutView) {
        mTempTxt = (TextView) layoutView.findViewById(R.id.s1);
        mDateTxt = (TextView) layoutView.findViewById(R.id.date);
        Button mRefreshBtn = (Button) layoutView.findViewById(R.id.refresh);
        mRefreshBtn.setOnClickListener(this);
    }

    @Override
    protected void prepareData(){
        Bundle args = getArguments();
        if (args != null && args.containsKey(BUNDLE_DATA)){
            mWeatherData = (Data) args.getSerializable(BUNDLE_DATA);
        }
        setUiData();
    }

    @Override
    public void onClick(View v) {
        int id = v.getId();
        if (id == R.id.refresh) {
            refreshData();
        }
    }

    private void refreshData() {
        BackendRequest.getsInstance(getActivity()).requestPosts(new ResponseListener<List<Data>>() {
            @Override
            public void onResponse(List<Data> data) {
                mWeatherData = data.get(0);
                setUiData();
            }

            @Override
            public void onError(Object error) {
```

```
            }
        });
    }

    private void setUiData() {
        if (mWeatherData == null) {
            refreshData();
            return;
        }
        mTempTxt.setText(mWeatherData.getS1());
        mDateTxt.setText(mWeatherData.getDate());
    }
}
```

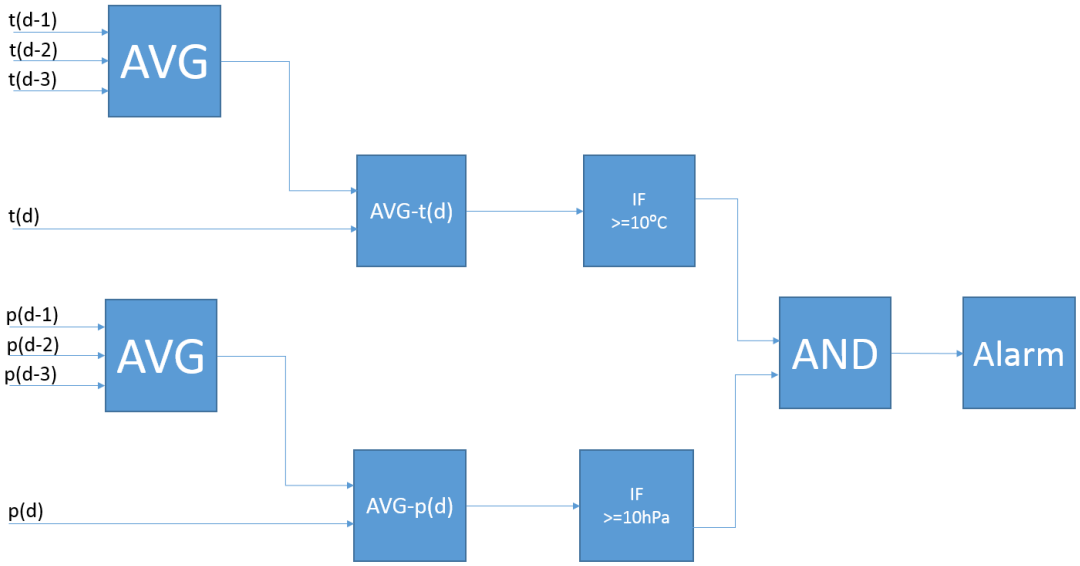
Programski kod 5.9. Fragment izrađen za prikazivanje temperature zraka.

5.15. Fuzija podataka koristeći dobivene podatke

Koristi li se sustav kao meteorološka stanica, podaci s nje mogu se iskoristiti za proučavanje mikro lokacijskih klimatskih uvjeta na prostoru s kojeg sustav skuplja podatke. Spremanjem podataka sa senzora u bazu podataka kroz nekoliko godina nastaju veliki skupovi podatka. Koristeći dobivene velike skupove podataka i njihovom kombinacijom, dobivaju se predviđanja o budućim klimatskim uvjetima. Iz informacija o tlaku zraka i temperaturi može se predvidjeti vrste padalina (kiša, snijeg, tuča) [33]. Ugradi li se senzor koji prati količinu padalina, uz navedene podatke o temperaturi i o tlaku zraka moguće je predvidjeti i količinu prognozirane padaline. Uz kontinuirano praćenje klimatskih uvjeta moguće je i kreirati alarm koji će se aktivirati ukoliko se kroz kombinacije različitih mjernih veličina ustvrdi da dolazi vremenska nepogoda.

Višesenzorsku fuziju podataka niske razine može se koristiti kako s ciljem stalneili povremene provjere rezultata koji se šalju u bazu. Korištenjem višesenzorske fuzije podataka osigurava se dodatna razina sigurnosti i pouzdanosti sustava.

Primjer korištenja fuzije podataka može se vidjeti na shemi sa slike 5.1. koju prikazuje slika 5.1. koja je izrađena prema podacima koji su uvjet za nastanak oborina. Na njoj je prikazan postupak kojim se od podataka dobivenih sa senzora i odgovarajućih uvjeta za vremensku nepogodu poput tuče može generirati alarm koji će upozoriti na moguću pojavu nepogode. Vrijednosti koje se pojavljuju u shemi su *t(d)* i *p(d)* koje predstavljaju trenutnu temperaturu i tlak, te *t(d-1)/p(d-1) - t(d-3)/p(d-3)* koje predstavljaju vrijednosti temperature i tlaka u prethodna tri dana. Vremensku nepogodu poput tuče moguće je detektirati naglom promjenom temperature i tlaka. Oduzimanjem trenutne vrijednosti zraka ili tlaka od prosječnih u zadnja tri dana, ispituju se uvjeti za nastanak tuče. Ako je dobiveni rezultat veći od očekivanih vrijednosti, postavlja se alarm koji upozorava na mogućnost tuče.



Sl. 5.1. Shematski prikaz postupka za aktiviranje alarma za mogućnost tuče.

6. PRIMJENA SUSTAVA I TESTIRANJE S ANALIZOM REZULTATA

6.1. Postavke komunikacijskih uređaja

Nakon nabave svih potrebnih komponenti, slijedi postupak omogućavanja njihovog međusobnog spajanja i prepoznavanja. Prvi koji dolazi na red je komunikacijski uređaj TP-Link 703N. Da bi uopće dobio mogućnosti koje su potrebne za traženi način komunikacije, potrebno je na njega instalirati operacijski sustav OpenWRT. Za početak je potrebno skinuti sliku operacijskog sustava OpenWRT za navedeni mrežni usmjerivač. Njega se preuzima na stranici http://openwrt.org. Nakon toga se u pretraživač upisuje naziv uređaja i odlazi na prvi od ponuđenih rezultata.

Postoji razlika u mogućnosti preuzimanja slika operacijskog sustava. Ako je trenutno na uređaju tvornički operacijski sustav, preuzima se slika s dodatkom u nazivu : *squashfs-factory.bin*. Ako mrežni uređaj na sebi već ima OpenWRT operacijski sustav i želi se njegova nadogradnja ili vraćanje na stariju verziju, preuzima se slika s nastavkom : *squashfs-sysupgrade.bin*. Ako se zamijene slike i pogrešno postave, uređaj postaje nedostupan i teško ga se ponovno vraća u funkciju.

Nakon preuzimanja slike i spajanja mrežnog uređaja na računalo LAN kabelom, u adresnu liniju Internet pregleda upisuje se adresa 192.168.1.1 te prateći postupak na slike nađene na internetu (jer je cijeli izbornik na mandarinskom jeziku) [34] postavlja se *squashfs-factory.bin* slika OpenWRT operacijskog sustava. Mrežni uređaj će obaviti instalaciju novog operacijskog sustava te se ponovno pokrenuti. Na novo sučelje uređaja dolazimo putem komandne linije. Ovisno koristi li se operacijski sustav Windows ili Linux, otvara se Command Prompt ili Terminal. Upisivanjem naredbe *telnet* i mrežne adrese 192.168.1.1 prijavljuje se u sučelje OpenWRT-a koje izgleda kao na slici 6.1.

Sl. 6.1. Početna slika nakon prijave u OpenWRT.

Nakon prijave, postavlja se nova šifra za pristup kako bi se dodatno osigurao uređaj. Tim postupkom će se svako sljedeće prijavljivanje na mrežni uređaj vršiti naredbom *ssh root@<mrežna adresa uređaja>*.

Kako bi se mogao nastaviti daljnji razvoj projekta, bilo je potrebno izvršiti instalaciju potrebnih paketa koji omogućuju ostvarivanje veze s 3G *stickom* za mobilni Internet i Arduinom. Za preuzimanje paketa potrebno je mrežni uređaj konfigurirati na način kako bi se imao pristup internetu. Unošenjem postavki u konfiguraciju mrežnog uređaja implementira se mogućnost mrežnog repetitora. Postavke koje su unesene u konfiguraciju mogu se vidjeti u programskom kodu 6.1. i programskom kodu 6.2.

```
/etc/config/network
config interface 'loopback'
        option ifname 'lo'
        option proto 'static'
        option ipaddr '127.0.0.1'
        option netmask '255.0.0.0'

config globals 'globals'
        option ula_prefix 'fdea:2b85:a9da::/48'
```

Programski kod 6.1. Postavke mreže.

```
/etc/config/wireless
config interface 'lan'
        option ifname 'eth0'
        option type 'bridge'
        option proto 'static'
        option ipaddr '192.168.1.1'
        option netmask '255.255.255.0'
        option ip6assign '60'

config interface 'wan'
        option ifname 'wlan0'
        option proto 'dhcp'

config wifi-device  radio0
        option type     mac80211
        option channel  11
        option hwmode   11ng
        option path     'platform/ar933x_wmac'
        list ht_capab 'SHORT-GI-20'
        list ht_capab 'SHORT-GI-40'
        list ht_capab 'RX-STBC1'
        list ht_capab 'DSSS_CCK-40'
        option htmode   HT20
#       option disabled '0'

config 'wifi-iface'
        option 'ssid' 'ime_mreze_koja_se_prima'
        option 'encryption' 'psk2'
        option 'device' 'radio0'
        option 'mode' 'sta'
        option 'network' 'wan'
        option 'key' 'sifra_mreze_koja_se_prima'
```

```
config wifi-iface
        option device   radio0
        option network  lan
        option mode     ap
        option ssid     ArduinoNET //naziv mreze koja se dijeli
        option encryption none
```

Programski kod 6.2. Postavke bežičnih mreža.

Nakon što su postavke dodane na mrežni uređaj, potrebno ga je ponovno pokrenuti. Kako bi se mreža pokrenula s novim postavkama, potrebno je izvršiti naredbu: /ets/init.d/network reload. Prilikom njezina izvršavanja na popisu bežičnih mreža pojavljuje se i napravljena mreža Arduino net i sami mrežni uređaj ima mogućnost pristupa internetu.

Za instalaciju paketa koji služe za uspostavu komunikacije putem 3G mobilnog *sticka* i za njegovo prepoznavanje, na mrežnom uređaju potrebno je unijeti programski kod 6.3. u konzolu.

```
$ opkg update
$ opkg install comgt kmod-usb-serial kmod-usb-serial-option kmod-usb-serial-wwan usb-modeswitch usb-modeswitch-data ppp
```

Programski kod 6.3. Instalacije paketa potrebnih za korištenje 3G mobilnog sticka.

Komunikacija između mrežnog uređaja i Arduina također zahtijeva svoje postavne pakete, a njih se instalira kao što prikazuje programski kod 6.4.

```
$ opkg install kmod-usb-serial-ftdi kmod-usb-acm kmod-usb-serial-pl2303 kmod-usb-serial-cp210x
```

Programski kod 6.4. Instalacija paketa za komunikaciju mrežnog uređaja i Arduina.

Nakon neuspjelog prepoznavanja 3G *sticka* za mobilni Internet i nemogućnosti spajanja mrežnog uređaja na Internet, počinje traženje mjesta gdje je nastao problem. Navedena situacija vidljiva je u programskom kodu 6.5.

```
root@OpenWrt:~# cat /proc/bus/pci/devices
root@OpenWrt:~# dmesg
[   11.270000] usbserial: USB Serial support registered for FTDI USB Serial Device
[   11.360000] PPP generic driver version 2.4.2
[   11.370000] NET: Registered protocol family 24
[   11.440000] usbcore: registered new interface driver option
[   11.460000] usbserial: USB Serial support registered for GSM modem (1-port)
[   11.490000] ath: EEPROM regdomain: 0x0
[   11.490000] ath: EEPROM indicates default country code should be used
[   11.490000] ath: doing EEPROM country->regdmn map search
[   11.490000] ath: country maps to regdmn code: 0x3a
[   11.490000] ath: Country alpha2 being used: US
[   11.490000] ath: Regpair used: 0x3a
[   11.510000] ieee80211 phy0: Selected rate control algorithm 'minstrel_ht'
[   11.510000] cfg80211: Calling CRDA for country: US
[   11.520000] ieee80211 phy0: Atheros AR9330 Rev:1 mem=0xb8100000, irq=2
[   11.540000] cfg80211: Regulatory domain changed to country: US
[   11.540000] cfg80211:  DFS Master region: FCC
[   11.540000] cfg80211:   (start_freq - end_freq @ bandwidth), (max_antenna_gain, max_eirp), (dfs_cac_time)
```

```
[   11.550000] cfg80211:   (2402000 KHz - 2472000 KHz @ 40000 KHz), (N/A, 3000 mBm), (N/A)
[   11.560000] cfg80211:   (5170000 KHz - 5250000 KHz @ 80000 KHz), (N/A, 1700 mBm), (N/A)
[   11.570000] cfg80211:   (5250000 KHz - 5330000 KHz @ 80000 KHz), (N/A, 2300 mBm), (0 s)
[   11.580000] cfg80211:   (5735000 KHz - 5835000 KHz @ 80000 KHz), (N/A, 3000 mBm), (N/A)
[   11.590000] cfg80211:   (57240000 KHz - 63720000 KHz @ 2160000 KHz), (N/A, 4000 mBm), (N/A)
[   23.580000] IPv6: ADDRCONF(NETDEV_UP): eth0: link is not ready
[   23.580000] device eth0 entered promiscuous mode
[   23.610000] eth0: link up (100Mbps/Full duplex)
[   23.610000] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[   23.620000] br-lan: port 1(eth0) entered forwarding state
[   23.620000] br-lan: port 1(eth0) entered forwarding state
[   25.620000] br-lan: port 1(eth0) entered forwarding state
[ 1002.880000] usb 1-1: new high-speed USB device number 2 using ehci-platform
[ 1003.040000] option 1-1:1.0: GSM modem (1-port) converter detected
[ 1003.050000] usb 1-1: GSM modem (1-port) converter now attached to ttyUSB0
[ 1003.050000] option 1-1:1.1: GSM modem (1-port) converter detected
[ 1003.060000] usb 1-1: GSM modem (1-port) converter now attached to ttyUSB1
[ 1003.080000] option 1-1:1.2: GSM modem (1-port) converter detected
```

Programski kod 6.5. Ispis konzole nakon spajanja 3G sticka za mobilni Internet.

Prepoznavanjem šifre proizvođača i serijskog broja proizvoda otvara se mogućnost ručnog unosa proizvoda kojeg do toga trenutka nije bilo na popisu uređaja koji se mogu prepoznati. Pronalazi se dokument u */etc/usb-mode.json* u koji se unose linije iz programskog koda 6.6.

```
"19d2:0016" : {
        "+" : {
                "t_vendor" : 6610,
                "t_product" : [ 148, 338 ],
                "mode" : "StandardEject",
                "msg" : [   ]
         }
},
```

Programski kod 6.6. JSON kod za prepoznavanje 3G sticka za mobilni Internet.

Prilikom ponovnog pokretanja uređaja i upisivanjem naredbe za ispis procesa koji se događaju prilikom uključivanja 3G *sticka* za mobilni internet, dobiva se ispis iz programskog koda 6.7. 3G *stick* je prepoznat i spreman za uspostavu komunikacije.

```
root@OpenWrt:~# cat /sys/kernel/debug/usb/devices

T:  Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#=  1 Spd=480  MxCh= 1
B:  Alloc=  0/800 us ( 0%), #Int=  0, #Iso=  0
D:  Ver= 2.00 Cls=09(hub  ) Sub=00 Prot=01 MxPS=64 #Cfgs=  1
P:  Vendor=1d6b ProdID=0002 Rev= 3.10
S:  Manufacturer=Linux 3.10.49 ehci_hcd
S:  Product=EHCI Host Controller
S:  SerialNumber=ehci-platform
C:* #Ifs= 1 Cfg#= 1 Atr=e0 MxPwr=  0mA
I:* If#= 0 Alt= 0 #EPs= 1 Cls=09(hub  ) Sub=00 Prot=00 Driver=hub
E:  Ad=81(I) Atr=03(Int.) MxPS=   4 Ivl=256ms

T:  Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#=  2 Spd=480  MxCh= 0
D:  Ver= 2.00 Cls=00(>ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs=  1
P:  Vendor=19d2 ProdID=0016 Rev= 0.00
```

```
S:  Manufacturer=ZTE,Incorporated
S:  Product=ZTE WCDMA Technologies MSM
C:* #Ifs= 3 Cfg#= 1 Atr=c0 MxPwr=500mA
I:* If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=option
E:  Ad=81(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=01(O) Atr=02(Bulk) MxPS= 512 Ivl=4ms
I:* If#= 1 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=option
E:  Ad=82(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=02(O) Atr=02(Bulk) MxPS= 512 Ivl=4ms
I:* If#= 2 Alt= 0 #EPs= 3 Cls=ff(vend.) Sub=ff Prot=ff Driver=option
E:  Ad=83(I) Atr=03(Int.) MxPS=  64 Ivl=2ms
E:  Ad=84(I) Atr=02(Bulk) MxPS= 512 Ivl=0ms
E:  Ad=03(O) Atr=02(Bulk) MxPS= 512 Ivl=4ms
```

Programski kod 6.7. Uspješno prepoznavanje 3G sticka za mobilni Intermet.

Isto kao što je namješteno korištenjebežične mreže potrebno je namjestiti mrežu koja dolazi s 3G *sticka* za mobilnu komunikaciju i njezino ponašanje na mrežnom uređaju prema programskom kodu 6.8. Postavke mreže koja dolazi na 3G *stick* iz programskog koda 6.9 upisuju se također u skriptu koja se nalazi u */etc/chatscripts/3g.chat*.

```
config interface '3g'
        option 'ifname'  'ppp0'
        option device  '/dev/ttyUSB1'
        option 'apn'     'data.vip.hr'
        option 'service' 'umts'
        option 'proto'   '3g'
        option 'pppd_options' 'noipdefault' 'refuse-chap 'refuse-mschap' 'refuse-mschap-v2' 'refuse-eap'
```

Programski kod 6.8. Mrežne postavke 3G mobilne mreže.

```
ABORT   BUSY
ABORT   'NO CARRIER'
ABORT   ERROR
REPORT  CONNECT
TIMEOUT 10
""      "AT&F"
OK      "ATE1"
OK      'AT+CGDCONT=1,"IP","$USE_APN"'
SAY     'Calling UMTS/GPRS'
TIMEOUT 30
OK      "ATD*99***1#"
CONNECT ' '
```

Programski kod 6.9. Unesene postavke 3g.chat skripte.

6.2. Unos podataka s Arduina u bazu podataka u oblaku

Dodavanjem postavki na mrežni uređaj omogućeno je primati podatke s Arduino razvojne pločice s mikroupravljačem i spremanje u bazu podataka putem mobilnog Interneta. Nakon što su podaci dostavljeni u bazu .php skripta iz programskog koda 6.10 oblikuje podatke iz baze na način koji će biti razumljiv Android biblioteci za mrežnu komunikaciju, tzv. JSON.

```
<?php

define('DB_HOST', getenv('OPENSHIFT_MYSQL_DB_HOST'));
define('DB_PORT',getenv('OPENSHIFT_MYSQL_DB_PORT'));
define('DB_USER',getenv('OPENSHIFT_MYSQL_DB_USERNAME'));
define('DB_PASS',getenv('OPENSHIFT_MYSQL_DB_PASSWORD'));
define('DB_NAME',getenv('OPENSHIFT_GEAR_NAME'));

$con=mysql_connect(DB_HOST,DB_USER,DB_PASS) or die("cannot connect");
mysql_select_db(DB_NAME)or die("cannot select DB");
$data = mysql_query("SELECT * FROM `devices` ORDER BY `devices`.`date` DESC LIMIT 1") or die("can not
select table");


$return_arr = array();

        while ($row = mysql_fetch_array($data)) {
        $row_array[date] = $row[date];
        $row_array[s1] = $row[s1];
        $row_array[s2] = $row[s2];
        $row_array[s3] = $row[s3];
        $row_array[s4] = $row[s4];

         array_push($return_arr,$row_array);
        }
header('Content-type: application/json');
        echo json_encode($return_arr);

 mysql_close($con);

 ?>
```

Programski kod 6.10. Skripta za formatiranje JSON requesta.

6.3. Prikaz podataka iz baze podataka u oblaku računala na Android uređaj

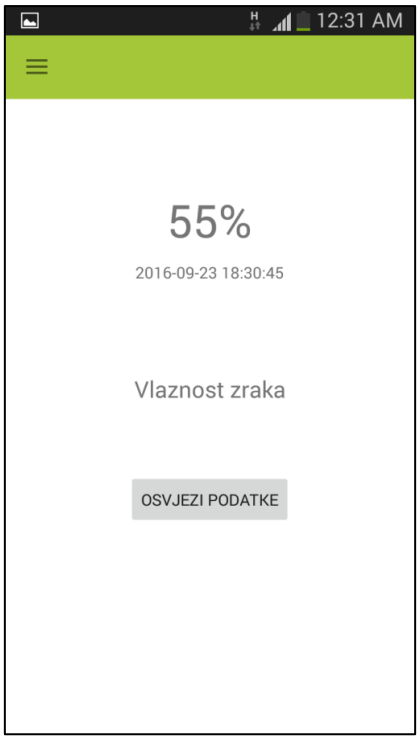
Prilagođavanjem prikaza podataka završava se zadnji dio procesa na poslužiteljskoj strani. Slijedi prikaz podataka na Android uređaju. Zbog toga što se diplomski rad koristi u kontroli klimatskih uvjeta u nastambama za pčele prilikom otvaranja aplikacije otvara se prozor kao na slici 6.2 koji odmah prikazuje temperaturu okoline jer je ona najbitnija informacija koja je u tom trenutku bila potrebna. Ako se aplikacija prije toga nije otvarala ili se čistila pričuvna memorija vjerojatno će aplikaciji trebati malo više vremena dok uspostavi vezu s bazom i počne primati podatke sa senzora.

Sl. 6.2. Početni prozor aplikacije za prikaz podataka sa senzora za mjerenje temperature okoline.

Ako se želi ponoviti dohvat podataka sa poslužitelja, koristi se izbornik *OSVJEZI PODATKE*. Podaci s ostalih senzora dostupni su u izborniku do kojeg se dolazi pritiskom na kućicu smještenu u gornjem lijevom kutu ili potezom prsta po zaslonu. Otvaranje izbornika s popisom senzora s kojih se mogu očitati podaci izgleda kao na slici 6.3.

Sl. 6.3. Prikaz izbornika s listom senzora.

Odabirom mogućnosti za očitavanje vlažnosti zraka odnosno *Vlaznost zraka* otvara se prozor koji izgleda kao na slici 6.4.



Sl. 6.4. Prozor za prikaz senzora za mjerenje vlažnosti zraka.

Ostali senzori koji se mogu koristiti su senzor temperature zemlje i senzor vlažnosti zemlje, a njihov izgled može se vidjeti na slici 6.5 i slici 6.6.

Sl. 6.5. Prozor za prikaz senzora za mjerenje temperature zemlje.

Sl. 6.6. Prozor za prikaz senzora za mjerenje vlažnost zemlje.

6.4.Testiranje rada sustava

Testiranje sustava provedeno je postavljanjem različitih brzina mobilne i internetske komunikacije kako na dijelu za pripremu podataka tako i na dijelu za prikaz podataka. Uređaj za mrežnu komunikaciju u prvom se slučaju spaja na 2,5G (E) mrežu koja postiže najveću brzinu od 40 *Kbps*, dok se na mobilnom uređaju izmjenjuju vrste mobilne i bežične internetske mreže te kontrolira vrijeme prihvaćanja podataka. Uređaj na kojem se obavljalo testiranje rada aplikacije i brzina protoka informacija je Samsung Galaxy S3 i-9300, specifikacija prikazanim u tablici 6.1.

Tab. 6.1. Specifikacije mobilnog uređaja za testiranje aplikacije.

Mreže: EDGE / GPRS (850 / 900 / 1800 / 1900 MHz), GSM 3G, HSDPA, HSDPA+
Procesor: 4 Cortex-A9 jezgre, takta 1,4 GHz, na Exynos 4 čipsetu
Grafički procesor: Mali-400MP
RAM: 1 GB
Interna memorija: 16GB
Operacijski sustav: Android 4.3

U tablici 6.2. prikazani su se rezultati dobiveni mjerenjem dok je mobilna mreža na 3G *sticku* bila postavljena na 2,5G (E).

Tab. 6.2. Testiranje rada sustava s postavljenom 2,5G mrežom na 3G stick-u za mobilni Internet.

	Mobilna 2,5G(E) mreža [s]	Mobilna 3G mreža [s]	Mobilna 3,5G mreža [s]	Bežični Internet [s]
Vrijeme slanja podatka do baze podataka	3,331	3,182	3,174	3,125
Vrijeme prvog prihvaćanja podataka	4,472	3,191	3,017	3,084
Vrijeme osvježavanja podataka	2,196 s	2,064	1,937	1,9788

Nakon toga se brzina mobilnog interneta s 3G *sticka* povećava na 3G mrežu čija najveća brzina je 42 Mbit/s, dok se brzine na mobilnom uređaju za testiranje aplikacije izmjenjuju isto kao i u prošlom testiranju. Rezultati dobiveni u ovom slučaju prikazani su u tablici 6.3.

Tab. 6.3. Testiranje rada sustava s postavljenom 3G mrežom na 3G stick-u za mobilni Internet.

	Mobilna 2,5G(E) mreža [s]	Mobilna 3G mreža [s]	Mobilna 3,5G mreža [s]	Bežični Internet [s]
Vrijeme slanja podatka do baze podataka	2,552	2,704	2,583	3,079
Vrijeme prvog prihvaćanja podataka	4,145	3,011	2,668	3,104
Vrijeme osvježavanja podataka	2,033 s	1,722 s	1,529 s	1,492 s

Posljednje testiranje provedeno je bez 3G mobilnog sticka odnosno s mrežnim uređajem spojenim na bežični internet čija najveća brzina iznosi 25 Mbit/s. Rezultati su vidljivi u tablici 6.4.

Tab. 6.4. Testiranje rada sustava s postavljenom mrežom na bežični Internet.

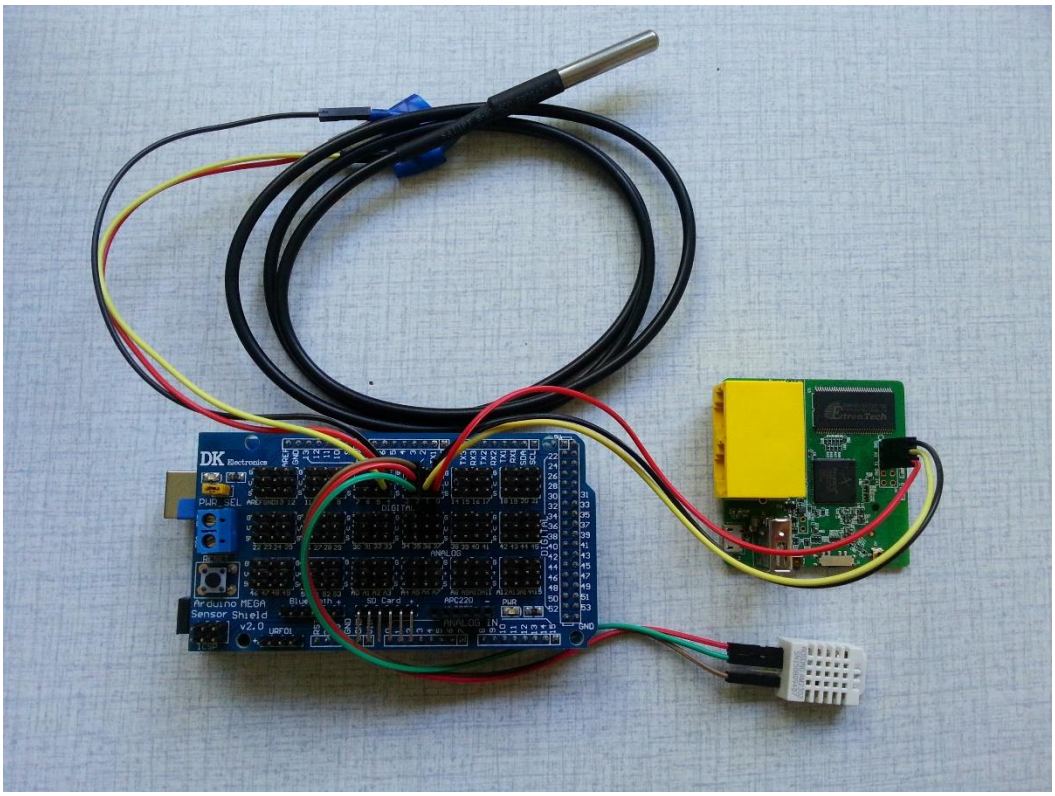
	Mobilna 2,5G(E) mreža [s]	Mobilna 3G mreža [s]	Mobilna 3,5G mreža [s]	Bežični Internet [s]
Vrijeme slanja podatka do baze podataka	3,042	2,952	3,102	3,001
Vrijeme prvog prihvaćanja podataka	3,948	2,428	2,139	3,092
Vrijeme osvježavanja podataka	1,554	2,311	1,475	1,646

Prilikom testiranja uređaja na duže vrijeme primijećeno je povremeno gubljenje veze između mrežnog uređaja i Arduina. Nakon opsežne provjere pronađen je problem koji se nalazio u pokretačkom programu (engl. *bootloader*) operacijskog sustava OpenWRT. Popravak je rezultirao onesposobljavanjem nekolicine mrežnih uređaja. Nakon što je pronađen odgovarajući pokretački

program zbog nepoznavanja procesa onesposobljeno je još nekoliko uređaja. Nakon ispravljanja memorijskih adresa u postupku instalacije novi pokretački program je nadograđen i do pucanja veze između mrežnog uređaja i Arduina više nije dolazilo.

Vremena dobivena prilikom mjerenja dokaz su sposobnosti izrađenog sustava o njegovoj mogućnosti da informaciju dostavi na odredište u kratkom vremenu, a to bi zadovoljilo soft-real time zahtjeve ovog sustava. Informaciju koju sustav treba prenijeti dolazi periodno. Period se ovisno o zahtjevima na sustav može skraćivati ili produžavati. U slučaju skraćivanja perioda treba se pripaziti na stabilnost sustava. Ukoliko informacija ne dođe u određenom rok sustav neće zakazati ali to može biti upozorenje da nešto na sustavu ne radi. Uzimajući to u obzir, sustav se i vremenski zahtjevi na isti može se svrstati u ublaženi sustav za rad u stvarnom vremenu.

Konačni izgled izrađenog sustava vidljivi su na Sl. 6.7.



Sl. 6.7. Konačni izgled izrađenog sustava

7. ZAKLJUČAK

Ostvareni sustav sastoji se od Arduino razvojne pločice s mikroupravljačem koji skuplja podatke sa senzora, te mrežnog uređaja koji mrežom postavlja i sprema skupljene podatke u bazu podataka. Baza podataka nalazi se u oblaku računala, a podaci skupljeni u njoj prikazuju se u Android aplikaciji. Sljedeći korak nakon izrade sustava bio je pokušati izmjeriti vrijeme koje je potrebno za svaku pojedinu aktivnost te vidjeti radi li stvarno sustav u jednom od područja rada stvarnog vremena. Problem koji se u početku javio je ograničenje senzora koji može obaviti određeni broj mjerenja u određenom periodu. Jedino što je još utjecalo na brzinu izvođenja operacija skupljanja, prijenosa i prikaza podataka je stabilnost internetske veze i njezina brzina. Nakon svih pojedinačnih izmjerenih vremena ustanovljeno je da je sustav sposoban raditi u granicama prihvatljivima za sustave stvarnog vremena. Potrebno je odlučiti treba li sustav kao takav raditi u stvarnom vremenu ili mu treba omogućiti da svoju zadaću obavlja kroz određene intervale. Sustav se pokazao koristan za skupljanje podatka sa senzora i stvaranja velikih skupova podataka. Koristeći kombinaciju mjernih veličina sa senzora dobiveni veliki skupovi podataka poslužili bi za predviđanje klimatskih uvjeta. Osim toga, koristeći nisku razinu fuzije može se iskoristiti i provjeriti ispravnosti senzora u sustavu te usporedbom dobivenih rezultata sa senzora osigurati bolju kvalitetu informacije koju sustav daje.

Ovako razrađen sustav podloga je za daljnje nadogradnje i implementiranje raznih mogućnosti ovisno koju zadaću mora ispunjavati osim skupljanja podataka. U sustav se može ugraditi određena automatizacija pomoću određivanja limita te ugradnje sustava za alarmiranje i aktuatora koji će reagirati na praćenu pojavu.

LITERATURA

[1] P. A. Laplante, Real-Time systems and Analysis 3.rd Edittion, ebook, IEEE PressA John Wiley & Sons, INC., New York, USA, 2004, 20.6.2016.

[2] H. Fourati, K. Iniewski, Multisensor Data Fusion, From Algorithms and Architectual Design to Applications, ebook, Taylor & Francis Group, LLC, Boca Raton, USA, 2016, 20.6.2016.

[3] D. Srivastava, Big Data Integration, Morgan & Claypoll Publishers, ebook, 2015, 21.6.2016.

[4] O.Vermesan and P.Friess, Internet of Things – from Research and Innovation to Market Deployment, ebook, River Publishers, Denmark, 23.6.2016.

[5] H.B. Mitchell, Multi-Sensor Data Fusion an introduction, ebook, Springer, New York. 2010, 24.6.2016.

[6] Data Fusion Techniques, Link ,25.6.2016.

[7] IoT-Owerview, Link , 27.6.2016.

[8] IoT Brings New Era of Weather Forecasting, Link, 27.6.2016.

[9] IoT and Precision Agriculture, Link, 27.6.2016.

[10] IoT and Smart Grid, Link, 27.6.2016.

[11] IoT and Driverless Car, Link, 27.6.2016.

[12] IoMT (Internet of Medical Things) or Healthcare IoT, Link, 27.6.2016.

[13] D.Hamliton, The Four IoT Connectivity Models explained, Link, 28.7.2016.

[14] Big Data, Link, 28.6.2016.

[15] D. L. Hall, Handbook of Multisensor Data Fusion, ebook, CRC Press, Boca Raton 2015.

[16] S. Yu, X. Lin, J. Mičić, X. Shen, Networking for Big Data, ebook, Chapman & Hall, CRC, Boca Raton 2016.

[17] Introduction to Arduino, Link, 2.7.2016.

[18] Arduino Mega 2560 & Genuino Mega 2560, Link, 28.6.2016.

[19] ATmega2560, Link, 3.7.2016.

[20] Arduino Environment, Link, 4.7.2016.

[21] DHT22, Link, 5.7.2016.

[22] DHT22 Datasheet, Link, 5.7.2016.

[23] BMP 180 picture, Link, 5.7.2016.

[24] BMP 180 Datasheet, Link, 5.7.2016.

[25] DS18B20 picture, Link, 5.7.2016.

[26] DS18B20 Datasheet, Link, 5.7.2016.

[27] LMV393 picture, Link, 5.7.2016.

[28] LMV393 Datasheet, Link, 5.7.2016.

[29] OpenWRT TP-Link 703 information, Link, 7.5.2016.

[30] OpenWRT, Link, 6.7.2016.

[31] Android Introduction, Link, 6.7.2016.

[32] Volley, Link, 6.7.2016.

[33] Crometeo, Link, 10.5.2017.

[34] OpenWRT TP-Link 703 image transfer, Link, 7.5.2017.

SAŽETAK

U diplomskom radu dan je pregled tehnologija koje omogućuju prikupljanje podataka iz okoline, te njihovo međudjelovanje. Opisani su pristupi poput fuzije podataka, Interneta stvari i velikih skupova podataka, te njihova uloga u praćenju okoline u stvarnom vremenu. Također, opisana je izgradnja navedenog sustava koristeći rješenja otvorenog koda, odnosno OpenWRT. Programski je ostvarena podrška koristeći OpenShift na poslužiteljskoj strani, te je izrađena aplikacija za Android. Poslužitelj se nalazi u oblaku računala, a podatke dobivene sa senzora (temperatura i vlažnost zemlje, temperatura i vlažnost zraka) sprema se u bazu podataka i priprema ih za prikaz u Android aplikaciji. Sustav je testiran na različitim brzinama internetske veze. Uzme li se u obzir količina podataka i način njihovog prijenosa, sustav se može svrstati u sustav s ublaženim vremenskim zahtjevima. Ovisno o zahtjevima na sustav, vremenski razmak između očitanja se može korigirati, ali treba razmotriti stabilnost sustava ukoliko se vrijeme slanja informacije u oblak računala želi smanjiti.

Ključne riječi: Arduino, Internet stvari, mobilna aplikacija, oblak računala.

ABSTRACT

MOBILE SURVEILLANCE SYSTEM OF THE ENVIRONMENT

A review of a part of the technologies which are found in different projects related to the concept of the Internet of things has been published in this diploma thesis. Moreover, the interdependence of the afore-mentioned technologies is also presented, as well as the zones in which they overlap. The very overlap which has been described is shown as part of the project which is used for the real-time environmental monitoring. The data collected from the environment are transferred into the database in the cloud, and are presented on an Android application. Each of the systems has been elaborated separately, with its development being presented at the end of the thesis, as well as the way in which system operates. The system was tested on various mobile Internet speeds. The analysis has shown that the system is capable of transmitting the data in soft real time. When it comes to system requirements, time laps could be modified, but system stability should be considered if we want to decrease cloud uploading time.

Keywords: Arduino, Internet od things, mobile application, cloud.

POPIS SLIKA

Sl. 2.1. Ilustrativni prikaz rada fuzije podataka. 9
Sl. 2.2. Shema Dasarathy-eve klasifikacija fuzije podataka prema postavkama ulaza i izlaza. 10
Sl. 2.3. Prikaz procesa fuzije podataka postavljena od strane Američkog Ministarstva obrane i Joint Directors of Laboratories. 11
Sl. 2.4. Komunikacija uređaj - na - uređaj. 15
Sl. 2.5. Komunikacija uređaj - na - oblak. 15
Sl. 2.6. Model uređaj – na – gateway. 16
Sl. 2.7. Model dijeljenja podataka s pozadinskih servisa. 17
Sl. 3.1. Blok shema sustava koji se izrađuje. 19
Sl. 4.1. Bitniji dijelovi na Arduino Mega 2560 razvojnoj pločici. 22
Sl. 4.2. Blok shema mikroupravljača ATmega 2560. 23
Sl. 4.3. Razvojno sučelje Arduino IDE. 25
Sl. 4.4. Senzor za mjerenje temperature i vlage DHT22. 26
Sl. 4.5. Senzor za praćenje tlaka zraka BMP180. 27
Sl. 4.6. Senzor za mjerenje temperature zemlje DS18B20. 27
Sl. 4.7. Senzor za praćenje vlage zemlje LM393. 28
Sl. 5.1. Shematski prikaz postupka za aktiviranje alarma za mogućnost tuče. 44
Sl. 6.1. Početna slika nakon prijave u OpenWRT. 45
Sl. 6.2. Početni prozor aplikacije za prikaz podataka sa senzora za mjerenje temperature okoline. 51
Sl. 6.3. Prikaz izbornika s listom senzora. 51
Sl. 6.4. Prozor za prikaz senzora za mjerenje vlažnosti zraka. 52
Sl. 6.5. Prozor za prikaz senzora za mjerenje temperature zemlje. 52
Sl. 6.6. Prozor za prikaz senzora za mjerenje vlažnost zemlje 53
Sl. 6.7. Konačni izgled izrađenog sustava 55

POPIS TABLICA

Tab. 4.1. Tablica značajki mikroupravljača ATmega 2560. 22
Tab. 4.2. Značajke senzora DHT22. 26
Tab. 4.3. Značajke senzora BMP180. 27
Tab. 4.4.Značajke senzora DS18B20. 28
Tab. 4.5. Značajke senzora za praćenje vlage zemlje. 28
Tab. 6.1. Specifikacije mobilnog uređaja za testiranje aplikacije. 53
Tab. 6.2. Testiranje rada sustava s postavljenom 2G mrežom na 3G stick-u za mobilni Internet. 54
Tab. 6.3. Testiranje rada sustava s postavljenom 3G mrežom na 3G stick-u za mobilni Internet. 54
Tab. 6.4. Testiranje rada sustava s postavljenom mrežom na bežični Internet. 54

POPIS PROGRAMSKIH KODOVA

Programski kod 4.1. Povezivanje Arduino razvojne pločice na T-Link usmjerivač. 29
Programski kod 4.2. Pokretanje skripte za spajanje pri svakom paljenju TP-Link usmjerivača. 29
Programski kod 5.1. Arduino kod za skupljanje podataka sa senzora i kreiranje poveznice. 32
Programski kod 5.2. Primjer URL-a za slanje podataka. 32
Programski kod 5.3. Pohrana podataka u bazu MySQL. 33
Programski kod 5.4. Primjer rasporeda spremnog za prikaz na zaslon. 37
Programski kod 5.5. Primjer rasporeda za proširivanje funkcionalnosti. 37
Programski kod 5.6. Primjer sastava za lakše upravljanje izbornicima. 38
Programski kod 5.7. Java datoteka koja sprema podatke primljene s poslužiteljske strane. 39
Programski kod 5.8. Adapter klasa. 41
Programski kod 5.9. Fragment izrađen za prikazivanje temperature zraka. 43
Programski kod 6.1. Postavke mreže. 46
Programski kod 6.2. Postavke bežičnih mreža. 47
Programski kod 6.3. Instalacije paketa potrebnih za korištenje 3G mobilnog sticka. 47
Programski kod 6.4. Instalacija paketa za komunikaciju mrežnog uređaja i Arduina. 47
Programski kod 6.5. Ispis konzole nakon spajanja 3G sticka za mobilni Internet. 48
Programski kod 6.6. JSON kod za prepoznavanje 3G sticka za mobilni Internet. 48
Programski kod 6.7. Uspješno prepoznavanje 3G sticka za mobilni Intermet. 49
Programski kod 6.8. Mrežne postavke 3G mobilne mreže. 49
Programski kod 6.9. Unesene postavke 3g.chat skripte. 49
Programski kod 6.10. Skripta za formatiranje JSON requesta. 50

ŽIVOTOPIS

Ivan Gregurić rođen je u Našicama 19. ožujka 1989. Osnovnu školu pohađao je u Donjem Miholjcu i završio je 2004. godine te je nakon toga upisao Opću gimnaziju u Donjem Miholjcu koju je završio 2007. godine. Nakon završene srednje škole 2007. godine upisuje se na Elektrotehnički fakultet u Osijeku, preddiplomski studij Računarstva koji završava 2011. Odmah nakon toga upisuje diplomski studij, smjer Procesno računarstvo, također na Elektrotehničkom fakultetu u Osijeku. Pri završetku diplomskog studija 2016. godine zapošljava se u Siemens Convergenceu u Osijeku u Odjelu za tehničku podršku. Uz obrazovanje aktivni je volonter u Društvu „Naša djeca“ u Donjem Miholjcu, u udruzi MIGRA (udruga za razvoj civilnog društva) te na projektima vezanim uz slobodni internet kao što su Otvorena Mreža i Wireless Osijek.

POPIS PRILOGA (na CD-u)

Prilog 1. Programski kod Android aplikacije

https://github.com/igreguric/WeatherApplication

Prilog 2. Dokument i PDF diplomskog rada

Prilog 3. Programski kod za povezivanje Arduino razvojne pločice na T-Link usmjerivač

Prilog 4. Pokretanje skripte za spajanje pri svakom paljenju TP-Link usmjerivača

Prilog 5. Arduino kod za skupljanje podataka sa senzora i kreiranje poveznice

Prilog 6. Primjer URL-a za slanje podataka

Prilog 7. Pohrana podataka u bazu MySQL

Prilog 8. Postavke mreže

Prilog 9. Postavke bežičnih mreža

Prilog 10. Instalacije paketa potrebnih za korištenje 3G mobilng sticka

Prilog 11. Instalacija paketa za komunikaciju mrežnog uređaja i Arduina

Prilog 12. Json kod za prepoznavanje 3G sticka za mobilni internet

Prilog 13. Uspješno prepoznavanje 3G sticka za mobilni internet

Prilog 14. Mrežne postavke 3G mobilne mreže

Prilog 15. Unesene postavke 3g.chat skripte

Prilog 16. Skripta za formatiranje json requesta