Nadzor klimatskih uvjeta poslužiteljske prostorije putem mobilne aplikacije

Paranus, Ivana

Master's thesis / Diplomski rad

2018

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:200:760666

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-05**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij računarstva

NADZOR KLIMATSKIH UVJETA POSLUŽITELJSKE PROSTORIJE PUTEM MOBILNE APLIKACIJE

Diplomski rad

Ivana Paranus

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 19.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Ivana Paranus
Studij, smjer:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 683 R, 03.10.2018.
OIB studenta:	90583840230
Mentor:	Prof.dr.sc. Goran Martinović
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Doc.dr.sc. Josip Balen
Član Povjerenstva:	Dr. sc. Dražen Bajer
Naslov diplomskog rada:	Nadzor klimatskih uvjeta poslužiteljske prostorije putem mobilne aplikacije
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak diplomskog rada:	U radu treba opisati uvjete rada i izraditi model klimatskih uvjeta poslužiteljskih prostora. Nadalje, treba ostvariti sustav za prikupljanje podataka o temperaturi i vlažnosti, a u slučaju prekoračenja željenih vrijednosti obavijestiti korisnika preko mobilne aplikacije koju treba izraditi. Razvijeni sustav treba ispitati u stvarnoj okolini i analizirati njegovu funkcionalnost. (Span, Vedran Vučetić)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
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: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	19.09.2018.

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, 10.10.2018.

Ime i prezime studenta:	Ivana Paranus
Studij:	Diplomski sveučilišni studij Računarstvo, smjer Procesno računarstvo
Mat. br. studenta, godina upisa:	D 683 R, 03.10.2018.
Ephorus podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Nadzor klimatskih uvjeta poslužiteljske prostorije putem mobilne aplikacije**

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:

Sadržaj

1. UVOD

Učinkovito hlađenje poslužiteljskih prostorija može biti veliki izazov. Podatkovna središta rade dvadeset četiri sata, sedam dana u tjednu omogućavajući protok i pohranu podataka, te zbog toga moraju imati učinkovit sustav hlađenja. Pregrijavanje poslužiteljskih prostorija i podatkovnih središta utječe i na postotak kvarova računalne opreme, pa je zbog toga nužno temperaturnu vrijednost održavati u dozvoljenim granicama. Visoka vlažnost može dovesti do kondenzacije na opremi pri čemu može doći do korozije opreme, a zbog niske vlažnosti dolazi do viška statičkog elektriciteta što može oštetiti ili uništiti elektroniku.

Da bi se spriječio nastanak kvarova i smanjila potrošnja energije bitno je pratiti vrijednosti temperature i vlažnosti, te na temelju tih podataka djelovati u okolini. Zagrijavanje poslužiteljske prostorije nije jednako u svim dijelovima prostorije, pa je zbog toga potrebno ispitati vrijednost temperature u različitim dijelovima poslužiteljske prostorije da bi se povećala učinkovitost hlađenja, a smanjila velika potrošnja energije. Smanjena potrošnja energije može se postići i predviđanjem rasta i pada temperature i vlažnosti na način da se prate prethodne vrijednosti i izdaje nekakvo upozorenje ili pokreće klima uređaj ukoliko vrijednosti kontinuirano rastu ili padaju.

Drugo poglavlje je posvećeno problemima klimatizacije poslužiteljskih prostorija i podatkovnih središta. Navedene su dozvoljene i preporučene vrijednosti za vlagu i temperaturu, te prikazan je način hlađenja podatkovnog središta s vrućim i hladnim prolazima. U trećem poglavlju navedena su načela, primjeri i izazovi kibernetsko-fizikalnih sustava i opisan je način nadzora poslužiteljskih prostorija i podatkovnih središta pomoću bežičnih senzorskih mreža, kao i važnost predviđanja rasta i pada temperature i vlažnosti. Četvrto i peto poglavlje donosi prijedlog sklopovske i programske podrške, kao i dizajn sustava, a na kraju i konačno rješenje. Šesto poglavlje opisuje način korištenja sustava, kao i njegovo ispitivanje u stvarnoj okolini.

2. PROBLEMI KLIMATIZACIJE POSLUŽITELJSKIH PROSTORIJA I PODATKOVNIH SREDIŠTA

Kako su podatkovna središta u konstantnom pogonu, potrebno im je osigurati učinkovit rashladni sustav, kao i neprekidno napajanje, a isto tako pratiti vrijednosti temperature, vlažnosti zraka i nestanka struje. Različiti čimbenici utječu na vrijednosti temperature i vlažnosti zraka, pa je najučinkovitije mjeriti i pratiti vrijednosti pomoću senzora. Ako postoje samo senzori, aplikacija može podići upozorenje da je potrebno poduzeti potrebne akcije da bi se regulirala temperatura ili vlažnost. Ako postoje senzori i aktuatori, aplikacija sama kontrolira rashladni sustav. Učinkovit nadzor omogućava sprječavanje kvarova opreme i smanjuje potrošnju energije.

Tablica 2.1. Preporučene i dopuštene vrijednosti za temperaturu i vlažnost za ASHRAE razrede A1, A2, A3, A4, B i C

RAZRED	Temperatura °C	Vlažnost RH
Preporučene vrijednosti		
A1-A4	18-27	8%-60%
Dopuštene vrijednosti		
A1	15-32	8%-80%
A2	10-35	8%-80%
A3	5-40	8%-85%
A4	5-45	8%-90%
B	5-35	8%-80%
C	5-40	8%-80%

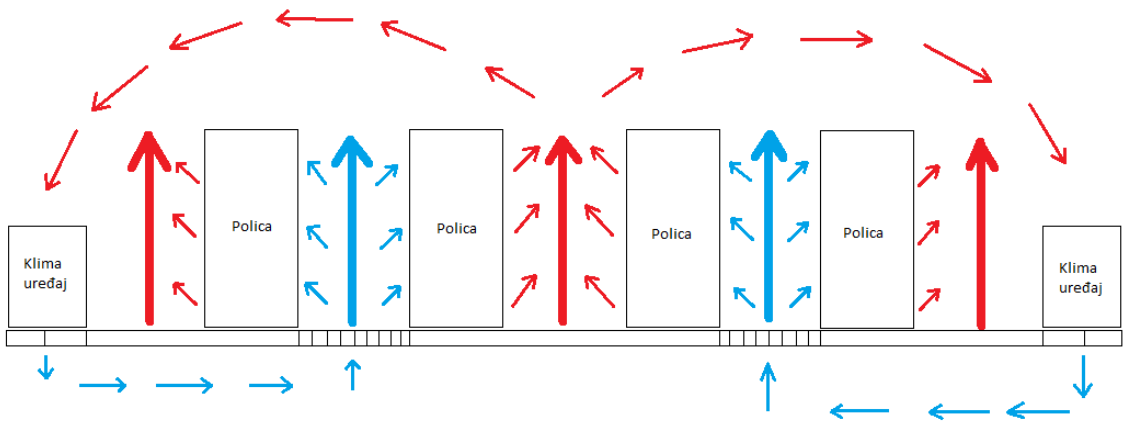
Prema tablici 2.1 po uzoru na [3], definicija ASHRAE razreda:

- Razred A1 – obično podatkovni centar s čvrsto kontroliranim parametrima okoliša (točka rosišta, temperatura i relativna vlaga) i kritičnim operacijama; vrste proizvoda obično dizajnirane za ovo okruženje su poslovni poslužitelji i proizvodi za pohranu.
- Razredi A2, A3 i A4 – obično prostor s računalnom opremom s nekom kontrolom parametara okoliša (točka rosišta, temperatura i relativna vlaga); vrste proizvoda obično dizajnirane za ovo okruženje su proizvodi za pohranu podataka, osobna računala i radne stanice. Između ova tri razreda, A2 ima najuže zahtjeve temperature i vlažnosti, a A4 ima najšire zahtjeve.
- Razred B – obično uredski, kućni ili prijenosni okoliš s minimalnom kontrolom parametara okoliša (samo temperatura); vrste proizvoda obično dizajnirane za ovo okruženje su osobna računala, radne stanice, prijenosnici i pisači.

- Razred C – obično prodajno mjesto ili lagano industrijsko ili tvorničko okruženje s vremenskom zaštitom i dostatnim zimskim grijanjem i ventilacijom; vrste proizvoda obično dizajnirane za ovo okruženje su oprema za prodajno mjesto, robusni kontroleri ili robusna računala i osobni digitalni pomoćnici.

2.1. Pregrijavanje poslužiteljskih prostorija i podatkovnih središta

Jedan od najvažnijih dijelova podatkovnog središta je rashladni sustav. Jako je važno implementirati sustav hlađenja koji učinkovito rashlađuje podatkovno središte, a istovremeno je energetski učinkovit.



Slika 2.1. Primjer vrućeg i hladnog prolaza za prostoriju s izdignutim podom i hlađenjem ispod izdignutog poda

Prema [2], slika 2.1 pokazuje konfiguraciju rasporeda vrućeg prolaza/hladnog prolaza. U vrućem prolazu/hladnom prolazu, police su smještene u nizu redova, stojeći na izdignutom podu. Prednja strana polica okrenuta je jedna prema drugoj i postala hladni prolaz, a stražnja strana polica, isto okrenuta jedna prema drugoj, postala je vrući prolaz. Klima uređaji isporučuju hladni zrak ispod izdignutog poda, a taj hladni zrak ulazi u prostoriju kroz rupičasti izdignuti pod i hladi poslužitelje. Vrući zrak koji izlazi iz poslužitelja vraća se na usisni dio klima uređaja.

2.2. Problemi vlažnosti poslužiteljskih prostorija i podatkovnih središta

Iako većina smatra da je održavanje optimalne vrijednosti temperature najbitnija stvar u održavanju idealnih uvjeta u podatkovnim središtima, vlažnost ima veliku ulogu. Prevelika vlažnost može dovesti do kondenzacije, što dalje može izazvati koroziju opreme. Zbog premale vlažnosti dolazi do viška statičkog elektriciteta što može oštetiti ili uništiti električnu opremu.

Najbitnije je konstantno nadzirati razinu vlažnosti i djelovati u okolini na temelju tih vrijednosti da bi se osigurali preporučeni uvjeti za poslužiteljske prostorije i podatkovna središta. Najbolji način nadzora temperature i vlažnosti je pomoću senzora koji se postave u poslužiteljske prostorije i podatkovna središta. Može se postaviti više senzora po prostoriji i na taj način dobiti uvid u to koji se dio više zagrijava, a koji manje. Prikupljene vrijednosti sa senzora moguće je samo pregledavati i analizirati i na temelju njih djelovati, ili se može razviti pametni sustav koji će pokretati klima uređaj prema potrebi tj. na temelju vrijednosti dobivenih sa senzora. Takav način omogućuje učinkovito hlađenje i smanjenje potrošnje energije jer se hladi samo onaj dio područja koji je previše zagrijan, a ne cijela prostorija. Najbolje je postaviti jedan senzor na svaku poslužiteljsku policu koju vidimo na slici 2.1, te bežično slati vrijednosti temperature i vlažnosti.

3. MODEL NADZORA MIKROKLIME POSLUŽITELJSKIH PROSTORIJA I PODATKOVNIH SREDIŠTA

3.1. Kibernetsko-fizikalni sustavi

Prema [4], kibernetsko-fizikalni sustavi uključuju bliske interakcije između računalnih sustava i fizikalnih objekata (poput automobila ili ljudi). Može ih se promatrati kao integraciju ugradbenih sustava, senzora i sustava kontrole.

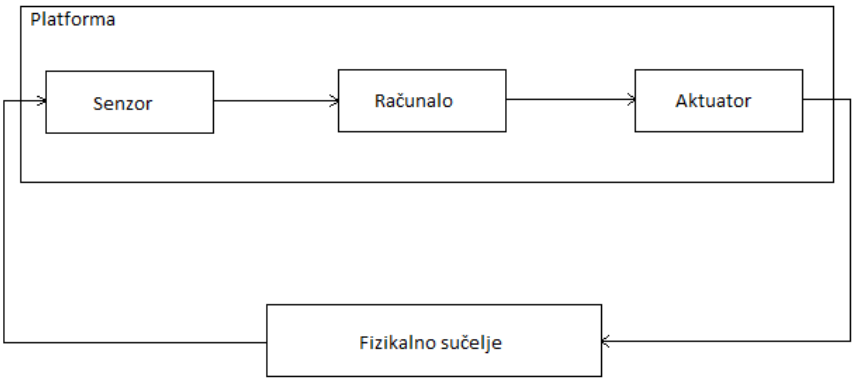
3.1.1. Osnovni pojmovi kibernetsko-fizikalnih sustava

3.1.1.1. Načela kibernetsko-fizikalnih sustava

Računala upravljaju mnogim stvarima oko nas, a neki načini upravljanja su manje vidljivi, kao na primjer upravljanje zračnim jastucima, perilicom posuđa, pisačima, semaforima, zračnim prometom i slično. Takve sustave zovemo ugradbeni sustavi.

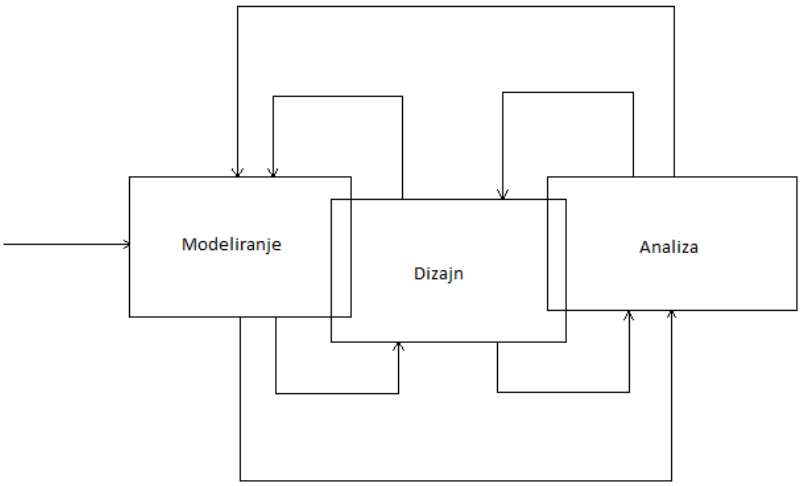
Prema [4], kibernetsko-fizikalni sustavi su sklop kibernetskih objekata (senzori, aktuatori i računala) i fizikalnih objekata (sustavi koje je načinio čovjek ili priroda). Dvije osnovne zadaće kibernetsko-fizikalnih sustava su nadziranje i kontrola. U kibernetsko-fizikalnim sustavima ugradbena računala i mreže prate i kontroliraju fizikalne procese, a fizikalni procesi utječu na računanje i obrnuto. Zbog toga što se u kibernetsko-fizikalnom sustavu događa mnogo stvari u isto vrijeme bitno je koliko je vremena potrebno za izvođenje zadatka, jer ponekad je vrijeme ključno da bi sustav ispravno funkcionirao. Fizikalni svijet je skup akcija koje se događaju u isto vrijeme, pa je zato nužno da ugradbeni sustavi učinkovito nadziru akcije.

Slika 3.1 prikazuje glavne dijelove strukture kibernetsko-fizikalnog sustav, a to su fizikalno sučelje, senzor, aktuator i računalo. Fizikalno sučelje je fizikalni dio kibernetsko-fizikalnog sustava, a može uključivati mehaničke dijelove, ljude i slično. Kibernetski dio kibernetsko-fizikalnog sustava čine senzori, aktuatori i računala. Na ovoj slici senzor, računalo i aktuator čine jednu platformu. Kibernetsko-fizikalni sustavi mogu imati više platformi koje su umrežene i koje međusobno utječu jedna na drugu. Isto tako, jedna platforma može sadržavati više senzora, računala i aktuatora.



Slika 3.1. Struktura kibernetsko-fizikalnog sustava

Prema [5], slika 3.2 prikazuje da postoje tri glavna procesa kod dizajniranja i implementiranja kibernetsko-fizikalnih sustava, a to su modeliranje, dizajn i analiza. Modeliranje je oponašanje sustava zbog dubljeg razumijevanja sustava, a modeli određuju što sustav radi. Dizajn navodi kako sustav čini to što čini. Analiza navodi zašto sustav čini to što čini ili zašto ne uspijeva učiniti ono što model kaže da bi trebao učiniti.



Slika 3.2. Procesi kod kreiranja kibernetsko-fizikalnih sustava

U ovom radu proces modeliranja se odnosi na razumijevanje problema klimatizacije poslužiteljskih prostorija i podatkovnih središta. Dizajn se odnosi na sastavljanje svih komponenata u jednu cjelinu tj. na prijedlog i ostvarenje sklopovske i programske podrške. Proces analize se provlači kroz cijeli rad, odnosno pojavljuje se kod modeliranja i dizajna analizirajući probleme koji se pojavljuju, ali i na kraju rada kod ispitivanja rada sustava.

3.1.1.2. Primjeri kibernetsko-fizikalnih sustava

Postoji puno primjera kibernetsko-fizikalnih sustava koji se koriste svakodnevno. Dobri primjeri kibernetsko-fizikalnih sustava su oni koji sprječavaju nastanak nesreća, zrakoplovnih i automobilskih. Dobro osmišljen sustav može ranije ukazati na neke kvarove i uzroke koji bi kasnije mogli dovesti do katastrofalnih nesreća. Kibernetsko-fizikalni sustavi se jako puno koriste za zaštitu imovine na način da kibernetsko-fizikalni sustav aktivira sigurnosni sustav kada pomoću senzora pokreta uoči uljeza. U velikim gradovima prikupljaju se podaci o prometu u stvarnom vremenu i na taj način se računaju najbrže rute za to doba dana. Kibernetsko-fizikalni sustavi se najviše koriste u kućama i velikim podatkovnim središtima. U kućama ljudi pomoću nadzora temperature i vlažnosti održavaju zdrav okoliš za obitelj. U podatkovnim središtima postavljaju se senzori za temperaturu, vlažnost, pokret, dim, itd. Pomoću podataka sa senzora se kontrolira klimatizacija i sprječava nastanak kvarova i neovlaštenih upada. Ljudi ih danas koriste i da bi nadzirali svoje zdravlje tj. da bi pratili i održavali svoju fizičku aktivnost.

3.1.1.3. Izazovi kibernetsko-fizikalnih sustava

Veliki problem kod kibernetsko-fizikalnih sustava je nesigurnost. Računalima kod izvođenja programa nije problem da li će se program izvesti do kraja, nego je problem što vrijeme izvođenja nije uvijek isto, a kod fizikalnih sustava vrijeme je jako važno. Da bi se postigla fizikalna kontrola u stvarnom vremenu kašnjenje između senzora i aktuatora mora biti minimizirano, jer kod kibernetsko-fizikalnih sustava kašnjenje od jedne sekunde može izazvati katastrofu.

Prema [4], jedna od najvažnijih karakteristika u fizikalnom svijetu je dinamika, tj. stanje sustava je stalno promjenjivo tijekom vremena, dok u kibernetskom svijetu te dinamike su definirane kao niz sljedova koji nemaju vremensku semantiku, i tu se javlja problem kibernetsko-fizikalne neusklađenosti. Postoje dva osnovna pristupa za rješavanje ovog problema: kibernetiziranje fizikalnog (kibernetska sučelja i svojstva se nameću na fizikalni sustav) i fizikaliziranje kibernetskog (kada su program i kibernetske komponente dinamički prikazane u stvarnom vremenu).

Fizikalni procesi su često nestabilni i nepredvidljivi, pa zbog toga kibernetski objekti imaju problem s praćenjem i kontroliranjem fizikalnog svijeta.

Prema [4], ugradbeni sustavi imaju različite razine apstrakcije što je korisno jer to dopušta stručnjacima rad na jednom dijelu sustava bez da razumiju ili mijenjaju ostatak sustava. Na primjer, programer može promijeniti svoj kod bez brige o elektroničkim komponentama koje

izvršavaju naredbe iz tog koda jer su kodovi dizajnirani s određenom apstrakcijom tj. oni su prikladni za sve strojeve sve dok stroj razumije kod. Problem koji se ovdje javlja je gubitak informacija tijekom komunikacije između skupina. Skupine svoje probleme prosljeđuju hardverskim inženjerima i tako gube osjećaj koliki utjecaj ima mala promjena na rad sustava u cjelini. Nedostatak komunikacije ometa napredak proizvodnje kibernetsko-fizikalnih sustava.

3.1.2. Razlika između kibernetsko-fizikalnog i ugradbenog sustava

Prema [5], kibernetsko-fizikalni sustav je integracija računanja s fizikalnim procesima čije je ponašanje definirano kibernetskim i fizikalnim dijelovima sustava. Ugradbena računala i mreže nadgledaju i kontroliraju fizikalne procese, obično s povratnim petljama gdje fizikalni procesi utječu na računanje i obrnuto. Kod kibernetsko-fizikalnih sustava bitna je interakcija između kibernetskog i fizikalnog dijela sustava.

3.1.3. Mobilni kibernetsko-fizikalni sustavi

Prema [4], u mobilnim kibernetsko-fizikalnim sustavima mobilni senzori i računala su povezani s kontrolom fizikalnog svijeta pomoću komunikacijskih mrežnih sustava (obično bežičnom mrežom).

Mobilni kibernetsko-fizikalni sustavi imaju više prednosti od običnih kibernetsko-fizikalnih sustava zato što mogu pratiti fizikalne objekte koji su u pokretu i mjeriti ih. Oni mogu biti primjenjivi u više područja kao što su mobilni inteligentni roboti, praćenje stanja okoliša, inteligentni transportni sustavi, praćenje srčanih bolesnika i slično.

3.1.3.1. Primjeri mobilnih kibernetsko-fizikalnih sustava

Mobilni kibernetsko-fizikalni sustavi se mogu koristiti za praćenje, dijagnosticiranje, sprječavanje i liječenje zdravstvenih problema. Isto tako mogu se koristiti i kao osobni savjetnici za održavanje zdravog načina života prateći putem senzora fizičku aktivnost korisnika.

Sustav mobilnog nadzora omogućuje kompanijama uštedu novca uvođenjem kamera koje se mogu daljinski kontrolirati, pa zbog toga nije potrebno imati čovjeka na svakoj udaljenoj lokaciji da prati i kontrolira stanje. Na udaljenim lokacijama mogu se ugraditi i dodatni izvori energije za sustave mobilnog nadzora, kao što su solarni paneli.

3.1.3.2. Problemi mobilnih kibernetsko-fizikalnih sustava

Zbog sve veće popularnosti mobilnih kibernetsko-fizikalnih sustava privatnost ljudi je sve više narušena, pa zbog toga mobilni kibernetsko-fizikalni sustavi moraju posvetiti puno pažnje sigurnosti osobnih podataka korisnika. Mobilni kibernetsko-fizikalni sustavi se moraju zaštititi od zlonamjernih napada jer oni mogu dovesti do neovlaštenih upada i krađe podataka korisnika, kao i lažnih unošenja podataka.

Kod kibernetsko-fizikalnih sustava ne postoji problem potrošnje energije jer su oni neprestano priključeni na izvor energije. Mobilni kibernetsko-fizikalni sustavi su pokretni pa zbog toga obično rade na baterije koje se mogu brzo potrošiti uslijed potrebe za obradom velike količine podataka i prijenosom iste. Zbog toga je jako bitno pronaći učinkovite načine za prijenos podataka i obavljati što više računanja u oblaku.

Izdržljivost i stabilnost sustava su jako bitni kod sustava koji prate, dijagnosticiraju, sprječavaju i liječe zdravstvene probleme, kao i kod inteligentnih transportnih sustava.

3.1.4. Djelovanje na fizičke objekte na temelju podataka sa senzora

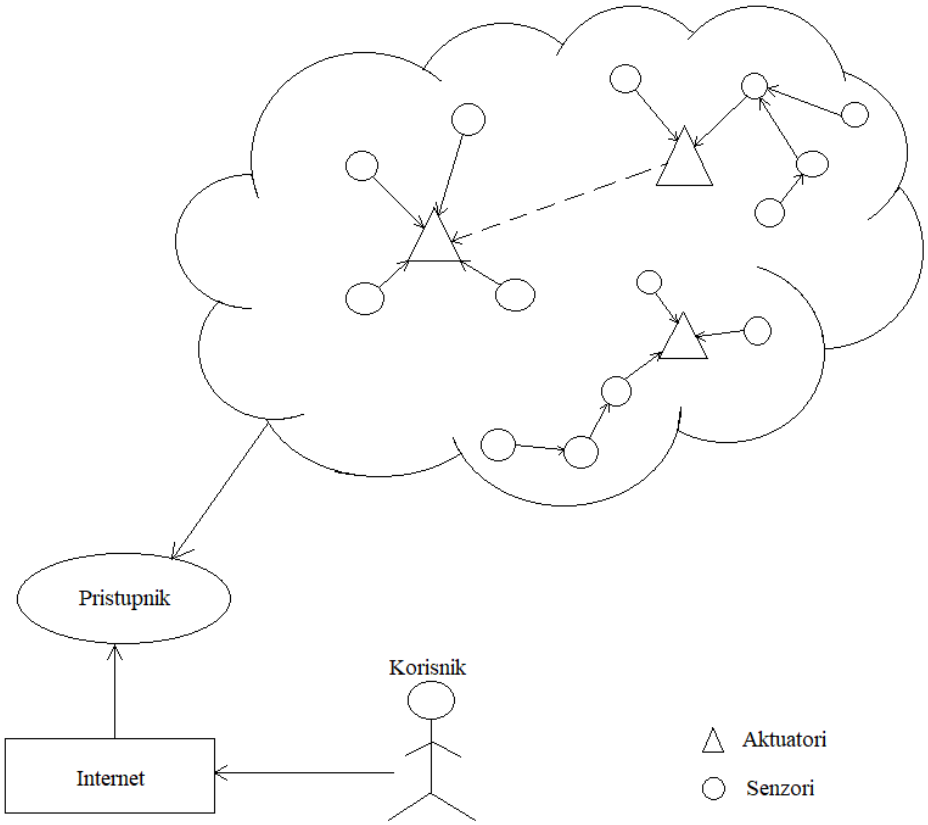
Kod djelovanja na fizičke objekte na temelju podataka sa senzora najbitnije je da se komunikacija između senzora i aktuatora odvija s minimalnim kašnjenjem. Senzori prikupljaju podatke iz stvarnog svijeta, a zatim se podaci šalju na obradu. Nakon obrade aktuatoru se šalju informacije o tome kakve akcije treba poduzeti s obzirom na dobivene podatke sa senzora. Djelovanje aktuatora može biti ovisno i o podacima s drugih aktuatora.

Prema [4], cilj je da u budućnosti bežične senzorske i aktuatorske mreže povežu velik broj mobilnih računalnih i ugradbenih uređaja i da se povežu s internetom s ciljem globalnog dijeljenja informacija, te na taj način postanu ono što povezuje fizikalni svijet s kibernetskim sustavom.

3.1.4.1. Bežične senzorske i aktuatorske mreže

Bežične senzorske i aktuatorske mreže (slika 3.4) se sastoje od više čvorova senzora i aktuatora koji međusobno komuniciraju putem bežične mreže. Mogu biti pokretne i nepokretne. Kod takvih mreža senzori prikupljaju podatke o okolišu, a aktuatori djeluju na osnovu tih podataka. Djelovanje pojedinog aktuatora može biti ovisno ne samo o podacima s jednog ili više senzora, nego i o podacima s jednog ili više drugih aktuatora. Bežične senzorske i aktuatorske mreže su skuplje od bežičnih senzorskih mreža jer senzori su jeftini, a aktuatori, poput robota, su jako skupi. Kod

bežičnih senzorskih i aktuatorskih mreža, za razliku od bežičnih senzorskih mreža, jako je bitno da senzori i aktuatori komuniciraju u stvarnom vremenu.



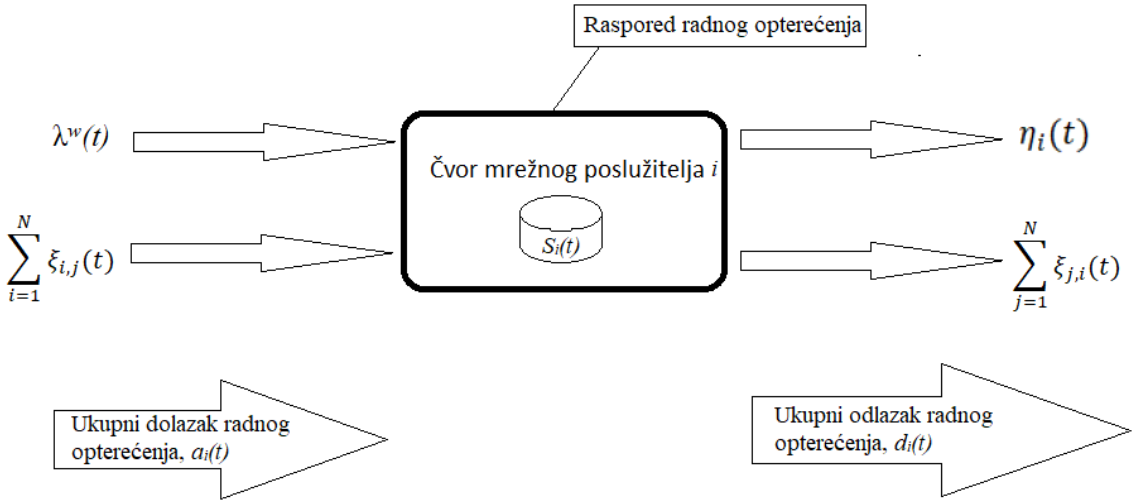
Slika 3.4. Arhitektura bežične senzorske i aktuatorske mreže

U ovom radu nije ostvaren aktuatorski dio. Senzorski dio je ostvaren i, kao i kod bežičnih senzorskih i aktuatorskih mreža, bežične senzorske mreže mogu biti pokretne i nepokretne. Aktuatorski dio bi mogao biti ostvaren ako se u aplikaciji napravi dio za kontrolu klima uređaja. Klima uređaj bi mogao biti kontroliran od strane korisnika putem mobilne aplikacije ili bi se klima uređaj pokretao na temelju podataka sa senzora bez sudjelovanja korisnika.

3.1.4.2. Klimatizacija podatkovnih središta

Prema [4], kontroliranje podatkovnih središta vrti se oko tri općenita modela, a to su: poslužiteljska razina, razina grupe i kontrola na razini podatkovnog središta. Kao što i samo ime kaže, kontrola na poslužiteljskoj razini odvija se na svakom pojedinom stroju poslužitelja, a uključuje kontrolu potrošnje energije i proizvodnju topline kako bi se postigla bolja energetska učinkovitost. Kontrola na razini grupe primjenjuje se na „čvorove“ koji opisuju jednu aplikaciju koja koristi više poslužitelja, a da bi se radno opterećenje, kao i potrošnja energije, smanjili u cijeloj grupi, poslužitelji se smještaju u virtualne strojeve koji se pak mogu prenijeti na različit hardver. Time

se dobiva uravnoteženo opterećenje i smanjenje potrošnje energije u cijeloj grupi tako što se obrada distribuira na više platformi. U kontroli na razini podatkovnog središta integrirano je nekoliko aspekata iz prethodnih razina, a to su optimizacija na poslužiteljskoj razini i migracija na razini grupe. Ova razina implementira i prethodna rješenja razina za učinkovitu potrošnju energije i distribuciju topline, a to može uključivati ujedinjenje opterećenja poslužitelja u manji podskup poslužitelja da bi se ostvarila energetska učinkovitost. Za klimatizaciju podatkovnih središta koristi se model kontrole podatkovnog središta, gdje su mrežni poslužitelji prikazani kao čvorovi kojima dolaze podaci na izračun i odlaze po izvršenju ili migracijom na drugi čvor, a to se može vidjeti na slici 3.3.



Slika 3.3. Prikaz klimatizacije poslužiteljske sobe

Grafički prikaz (slika 3.3.) izveden je iz sljedećih jednadžbi:

$$a_i(t) = \lambda^w(t) S_i(t) + \sum_{j=1}^{N} \xi_{i,j}(t) \tag{3-1}$$

$$d_i(t) = \eta_i(t) + \sum_{j=1}^{N} \xi_{j,i}(t) \tag{3-2}$$

$$\upsilon_i(t) = \mu_i(t) + \sum_{j=1}^{N} \delta_{i,j}(t) \tag{3-3}$$

gdje je:

- $a_i(t)$ – ukupni dolazak radnog opterećenja,
- $\lambda^w(t)$ – brzina dolaska radnog opterećenja u podatkovno središte,
- $S_i(t)$ – čvor mrežnog poslužitelja *i*,
- $\xi_{i,j}(t)$ – brzina migracije radnog opterećenja od *j* do *i*,
- $d_i(t)$ – ukupni odlazak radnog opterećenja,

- $\eta_i(t)$ – odlazak radnog opterećenja nakon izvršenja. Definirano u bilo kojem trenutku bilo kao $\mu_i(t)$ (ako je ukupna brzina dolaska a veća od ukupne brzine odlaska d) ili inače kao brzina dolaska $a_i(t)$,
- $\upsilon_i(t)$ – željena brzina odlaska na određeni čvor,
- $\mu_i(t)$ – željena brzina izvršenja,
- $\delta_{i,j}(t)$ – potrebna brzina migracije.

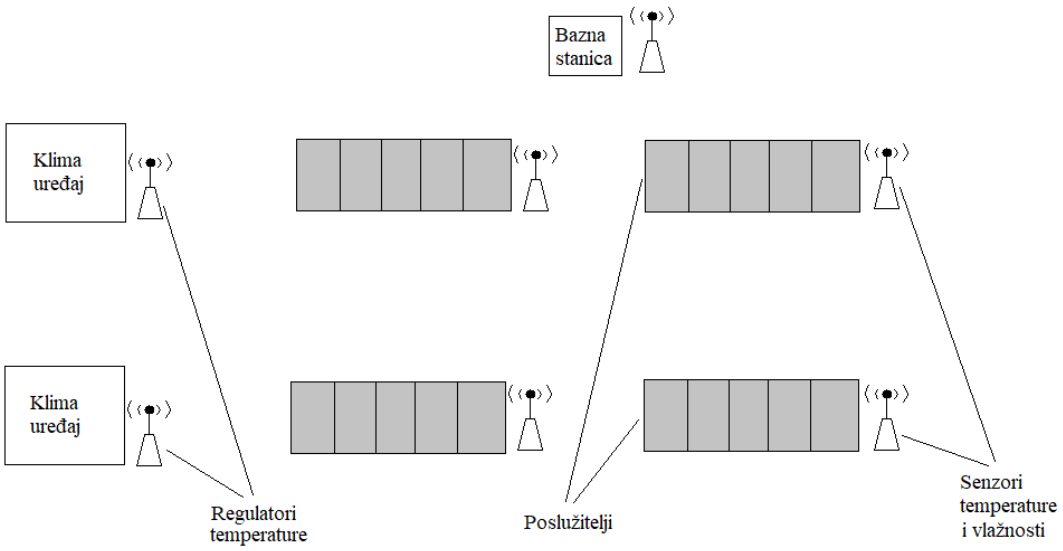
3.2. Nadzor poslužiteljskih prostorija i podatkovnih središta pomoću bežičnih senzorskih mreža

Nadzor pomoću bežičnih senzorskih mreža omogućuje jednostavnu instalaciju bez žica, niske troškove održavanja i učinkovitu potrošnju energije tako što se temperatura u poslužiteljskim sobama i podatkovnim središtima održava na optimalnoj razini.

3.2.1. Arhitektura sustava za nadzor poslužiteljskih prostorija i podatkovnih središta

Bežična senzorska mreža sastoji se od više bežičnih senzora koji prikupljaju podatke o temperaturi i vlažnosti. Glavna prednost ovog tipa nadzora je što se štedi energija na način da se hladi ili zagrijava samo određeno područje poslužiteljskih prostorija i podatkovnih središta, tj. hladi ili zagrijava se područje na kojem je očitana visoka ili niska temperatura, pa na taj način ne rade svi klima uređaji i štedi se energija.

Ova arhitektura na slici 3.5 se sastoji od senzora temperature i vlažnosti, regulatora temperature i vlažnosti i bazne stanice. Senzori temperature i vlažnosti ne pričvršćuju se na svakog poslužitelja, nego se jedan senzor stavlja na svaku poslužiteljsku policu, a oni periodično šalju podatke o temperaturi i vlazi baznoj stanici. Bazna stanica prikuplja podatke o temperaturi i vlazi sa svih senzora i šalje poruku regulatoru temperature ako je potrebno promijeniti temperaturu u bilo kojem području.



Slika 3.5. Sustav za nadzor poslužiteljskih prostorija i podatkovnih središta pomoću bežičnih senzorskih mreža

Bežične senzorske mreže su temelj ovog rada. Pomoću njih najbolje se nadziru klimatski uvjeti poslužiteljskih prostorija i podatkovnih središta. Sklopovsko rješenje koje će biti napravljeno moći će se pričvrstiti na svaku poslužiteljsku policu i prikupljati podatke sa senzora. Sklopovski dio će se moći napajati preko adaptera ili pomoću baterija, a pomoću bežičnog modula moći će se spojiti na mrežu i slati podatke koji će kasnije biti dostupni preko mobilne aplikacije. Dio koji se odnosi na kontroliranje klima uređaja u ovom radu neće biti ostvaren, napravljen će biti samo sustav za nadzor vrijednosti temperature i vlažnosti pomoću mobilne aplikacije.

3.3. Predviđanje rasta i pada temperature i vlažnosti

Predviđanje rasta i pada temperature i vlažnosti u poslužiteljskim prostorijama i podatkovnim središtima omogućava učinkovitu potrošnju energije na način da uspoređuje trenutnu i jednu ili više prethodnih vrijednost temperature i vlažnosti i na osnovu rezultata donosi odluku o tome kakve akcije je potrebno provesti da bi se dobilo idealno stanje okoline. Isto tako omogućava praćenje promjena vrijednosti temperature i vlažnosti kroz različito doba dana. Dakle, predviđanje ne omogućuje da vrijednost temperature i vlage dođe do kritične vrijednosti pa da se onda aktivira sustav, nego omogućava da sustav poduzme akcije i prije nego vrijednost dođe do kritične, te na taj način uštedi energiju i spriječi štetu na opremi.

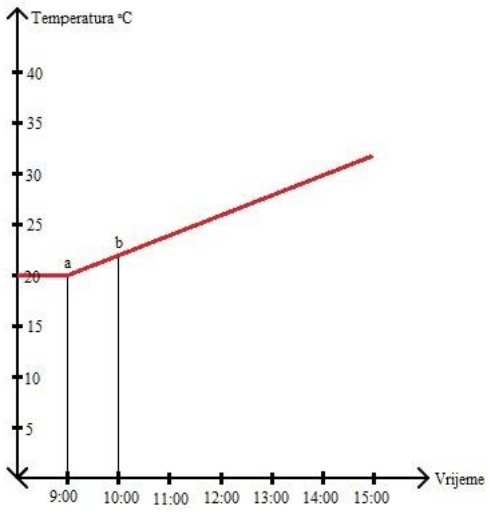
Slika 3.6 prikazuje pseudokod za predviđanje rasta i pada temperature i vlažnosti zraka. Vidljivo je da su na početku inicijalizirane varijable koje će se koristiti u ostatku koda. Nakon toga su u varijable *temp0* i *temp1* spremljene predzadnja i zadnja vrijednost temperature koje su pročitane

sa senzora, a u varijable *vlaga0* i *vlaga1* su spremljene predzadnja i zadnja vrijednost vlažnosti zraka koje su pročitane sa senzora. Nakon toga se uspoređuju zadnje dvije vrijednosti temperature i rezultat se sprema u varijablu *tempRez.* Ako je *tempRez<0* to znači da temperatura raste, a ako je *tempRez>0* to znači da temperatura pada. U slučaju da je *tempRez=0* to znači da temperatura niti raste niti pada. Poslije toga se uspoređuju zadnje dvije vrijednosti vlažnosti zraka i rezultat se sprema u varijablu *vlagaRez*. Ako je *vlagaRez<0* to znači da vlažnost zraka raste, a ako je *vlagaRez>0* to znači da vlažnost zraka pada. Ako je *vlagaRez=0* to znači da vlažnost zraka niti raste niti pada.

```
početak;
ulaz temp0, temp1, vlaga0, vlaga1, tempRez, vlagaRez;
temp0 = predzadnja vrijednost temperature sa senzora;
temp1 = zadnja vrijednost temperature sa senzora;
vlaga0 = predzadnja vrijednost temperature sa senzora;
vlaga1 = zadnja vrijednost temperature sa senzora;
tempRez = usporedi(temp0, temp1);
ako je (tempRez < 0) onda:
    izlaz "Temperatura raste";
ako je (tempRez > 0) onda:
    izlaz "Temperatura pada";
inače:
    izlaz "Temperatura je konstantna";
vlagaRez = usporedi(vlaga0, vlaga1);
ako je (vlagaRez < 0) onda:
    izlaz "Vlažnost zraka raste";
ako je (vlagaRez > 0) onda:
    izlaz "Vlažnost zraka pada";
inače:
    izlaz "Vlažnost zraka je konstantna";
kraj;
```

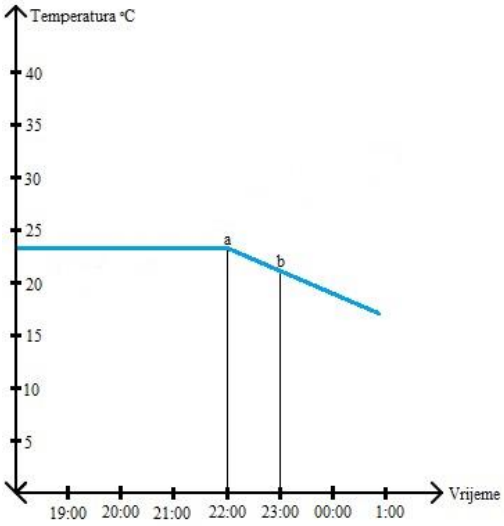
Slika 3.6. Pseudokod za predviđanje rasta i pada temperature i vlažnosti zraka

Na slici 3.7 vidi se porast temperature u određeno doba dana. Pametni sustav će uspoređivati zadnje dvije vrijednosti temperature i ako ustanovi da temperatura raste (ako je b-a>0) i blizu je vrijednosti koja aktivira sustav hlađenja, javiti će operateru zaduženom za kontrolu da temperatura raste i, ako je sustav tako podešen, aktivirati sustav hlađenja da temperatura ne dođe do gornje granice. Zbog pregrijavanja poslužiteljskih prostorija i podatkovnih središta dolazi do kvarova računalne opreme.



Slika 3.7. Graf porasta temperature

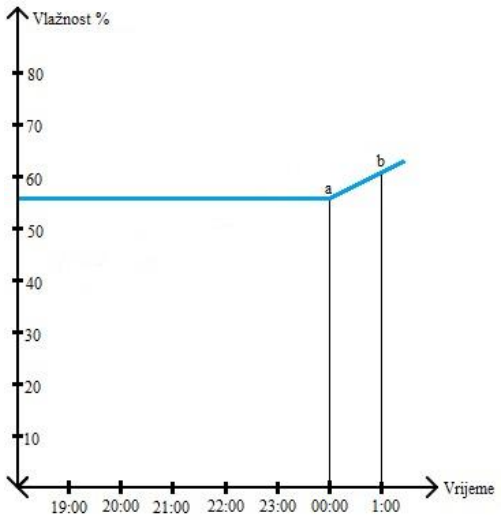
Na slici 3.8 prikazan je pad temperature u određeno doba dana. Sustav će uspoređivati zadnje dvije vrijednosti temperature i ako ustanovi da temperatura pada (ako je b-a<0) i blizu je vrijednosti koja aktivira klima uređaj za grijanje, javiti će operateru zaduženom za kontrolu da temperatura pada i, ako je sustav podešen da održava optimalnu temperaturu, aktivirati sustav grijanja da temperatura ne dođe do donje granice.



Slika 3.8. Graf pada temperature

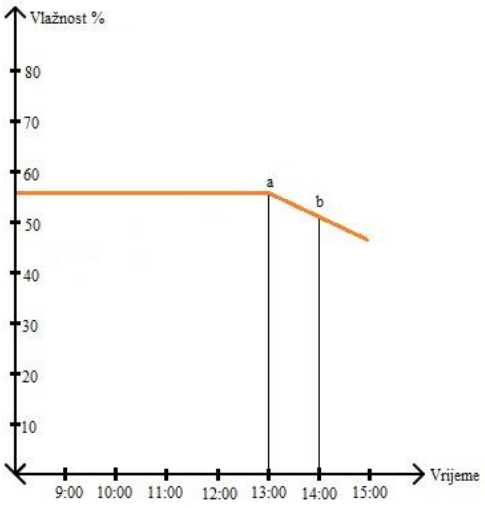
Slika 3.9 prikazuje porast vlažnosti u određeno doba dana. Sustav će uspoređivati zadnje dvije vrijednosti vlažnosti i ako ustanovi da vlažnost raste (ako je b-a>0) i blizu je vrijednosti koja aktivira uređaj za odvlaživanje zraka, javiti će operateru zaduženom za kontrolu da je vlažnost prostorije u porastu i, ako je sustav podešen da održava optimalnu vlažnost prostorije, aktivirati

sustav za odvlaživanje zraka prije nego vrijednost vlažnosti zraka dođe do gornje granice. Zbog prevelike vlažnosti zraka može doći do kondenzacije, što dalje može izazvati koroziju opreme.



Slika 3.9. Graf porasta vlažnosti

Slika 3.10 prikazuje pad vrijednosti vlage u prostoriji u određeno doba dana. Sustav uspoređuje zadnje dvije vrijednosti vlažnosti i ako ustanovi da vlažnost pada (ako je b-a<0) i blizu je vrijednosti koja aktivira uređaj za ovlaživanje zraka, javiti će operateru zaduženom za kontrolu da je vlažnost prostorije u padu i, ako je sustav podešen da održava optimalnu vlažnost prostorije, aktivirati sustav za ovlaživanje zraka prije nego vrijednost vlažnosti zraka dođe do donje granice. Zbog premale vlažnosti može doći do viška statičkog elektriciteta što može oštetiti ili uništiti električnu opremu.



Slika 3.10. Graf pada vlažnosti

4. PRIJEDLOG SKLOPOVSKE I PROGRAMSKE PODRŠKE

4.1. Sklopovska okolina sustava

4.1.1. Upravljač Arduino

Prema [10], Arduino je upravljač otvorenog koda koji omogućava programiranje i interakciju. Programiran je u C/C++ programskom jeziku što omogućuje fleksibilniju programabilnost i sposobnost korištenja elektronike koja se može povezati s Arduinom.

4.1.1.1. Povijest Arduina

Prema [10], 2005. godine Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino i David Mellis željeli su napraviti jednostavan programabilni uređaj koji će biti cjenovno dostupan svima, jednostavan za upotrebu, te se lagano povezivati sa senzorima, motorima i raznim drugim stvarima. Odabrali su 8-bitne mikrokontrolere iz Atmel-a i osmislili samostalnu pločicu i jednostavnu razvojnu okolinu koja koristi programe nazvane skice (engl. *sketches*). Budući da je Arduino otvorenog koda, sheme sklopova su dostupne besplatno i svatko tko želi izraditi vlastitu pločicu na temelju sheme to može i učiniti.

4.1.1.2. Vrste Arduina

Postoji mnogo Arduina koji se razlikuju po veličini, mogućnostima, značajkama i dizajnu, ali postoje samo dva modela koja koriste potpuno drugačije čipove. Standard koristi Atmega8/168/328 čipove, a Mega koristi Atmega1280 čip.

Postoje neke osnovne vrste Arduina koje su najbolji izbor za početnike npr. Arduino Uno, Arduino Nano, Arduino Leonardo itd. Naprednije pločice (npr. Arduino Mega 2560, Arduino Zero, Arduino Due itd.) koriste se za zahtjevnije projekte jer posjeduju naprednije funkcionalnosti.

4.1.1.3. Arduino Uno R3

Arduino Uno R3, prikazan na slici 4.1, je pločica temeljena na Atmega328P čipu. Može se spojiti na računalo pomoću USB kabela, a napajati se može i pomoću AC-DC adaptera i baterije. Specifikacije ovog Arduina prikazane su u tablici 4.1.

Tablica 4.1. Specifikacije upravljača Arduino Uno R3 [11]

Mikrokontroler:	ATmega328P
Radni napon:	5V
Ulazni napon (preporučeni):	7 – 12V
Ulazni napon (ograničeni):	6 – 20V
Digitalni ulazno/izlazni pinovi:	14 (od kojih se 6 može koristiti kao PWM izlaz)
PWM digitalni ulazno/izlazni pinovi:	6
Analogni ulazni pinovi:	6
DC struja po ulazno/izlaznom pinu:	20mA
DC struja za 3.3V pin:	50mA
Brza memorija:	32KB (Atmega328P)
SRAM:	2KB (Atmega328P)
EEPROM:	IKB (Atmega328P)
Brzina sata:	16MHz
LED_BUILTIN:	13
Dužina:	68.6mm
Širina:	53.4mm
Težina:	25g

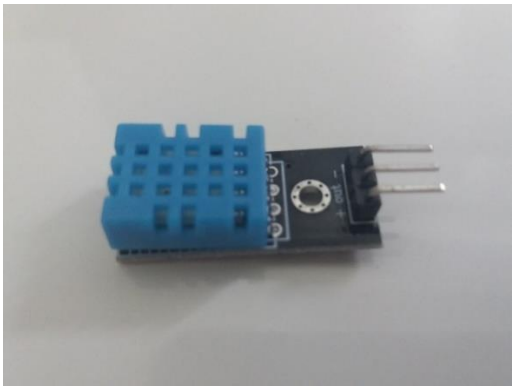
Slika 4.1. Arduino Uno R3

4.1.2. Senzori

Prema [10], senzor je uređaj koji osjeća promjenu u ulaznoj energiji da bi proizveo promjenu u drugom ili istom obliku energije. Dakle, senzor mjeri neku fizikalnu veličinu (temperaturu, vlažnost, tlak itd.) i pretvara ju u signal koji je čitljiv čovjeku i/ili instrumentu.

4.1.2.1. Senzor temperature i vlažnosti

Postoje dva senzora za temperaturu i vlagu, DHT11 i DHT22. DHT11 senzor može imati tri ili četiri pina. DHT11, prikazan na slici 4.2, senzor s tri pina, za razliku od onog s četiri pina, ima ugrađen otpornik od 10000 Ω. On može mjeriti temperaturu od 0 do 50°C i relativnu vlažnost od 20 do 95% i radi na 3.3 – 5V. Jednostavan je za implementaciju, malih je dimenzija s niskom potrošnjom energije, jeftiniji je od DHT22 senzora, ali manje precizan.



Slika 4.2. Senzor temperature i vlažnosti DHT11 s tri pina

4.1.3. Komunikacija

4.1.3.1. Bežični modul

Prema [12], bežični mrežni modul ESP8266, prikazan na slici 4.3, može raditi kao stanica, tako da ga je moguće povezati s bežičnom mrežom, a može djelovati i kao *soft-AP* za uspostavljanje vlastite bežične mreže, pa je moguće povezati ostale stanice s takvim ESP modulom. Maksimalan broj povezanih stanica sa *soft-AP* je pet. ESP8266 može raditi istovremeno na stanica i *soft-AP* načinu tako da može djelovati kao čvor umrežene mreže (engl. *mesh network*).

Slika 4.3. Bežični mrežni modul ESP8266 ESP-01 s adapterom

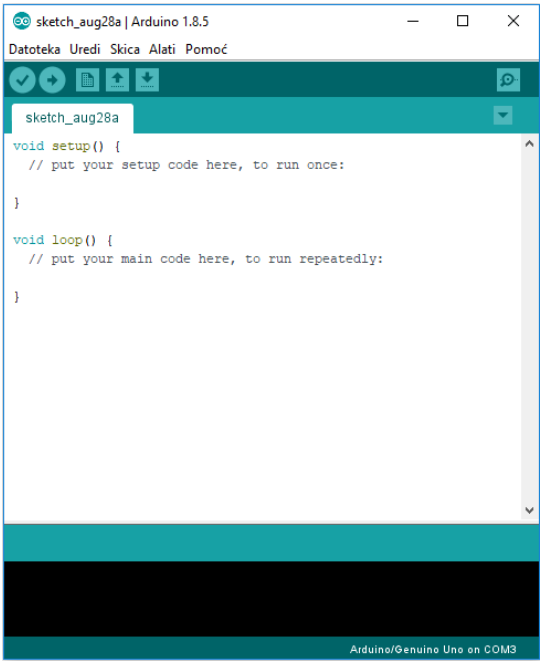
4.2. Programske razvojne okoline, jezici i alati

4.2.1. Arduino okolina

Kada je izumljen Arduino upravljač, izumitelji su osmislili i jednostavnu Arduino integrirau razvojnu okolinu koja koristi programe nazvane skice (engl. *sketches*). Arduino integrirana razvojna okolina programirana je u C/C++ programskom jeziku s velikom Arduino bibliotekom koja omogućuje jednostavnije korištenje elektronike koja se može povezati s Arduinom. Program napisan u Arduino programu lako se prenosi na Arduino pločicu.

Prema [10], Arduino integrirana razvojna okolina ima nekoliko značajki:

- može se izvoditi na Windowsu, Macintoshu i Linuxu,
- lagana je za korištenje,
- program se na Arduino pločicu prenosi putem USB kabela,
- sklopovska i programska podrška su otvorenog koda.

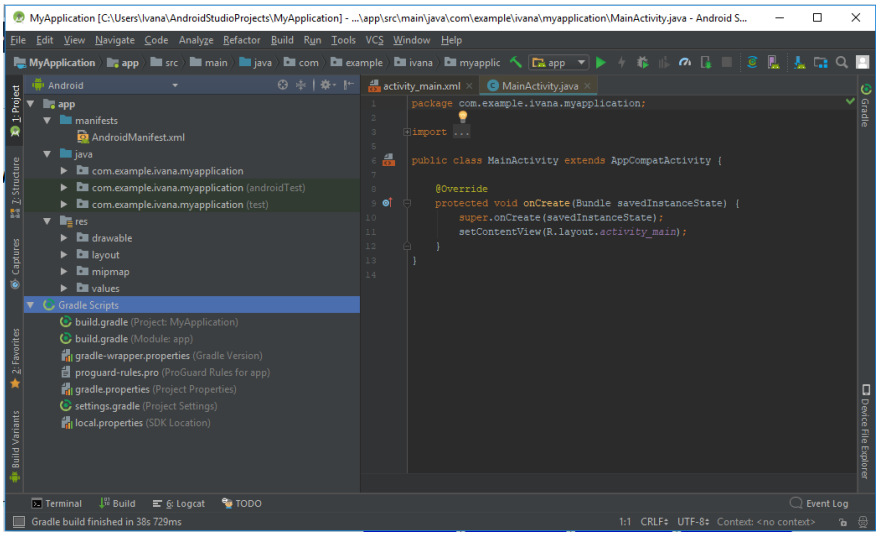


Slika 4.4. Sučelje Arduino integrirane razvojne okoline

4.2.2. Android razvojna okolina

Prema [13], Android Studio je službeno integrirano razvojno okruženje (engl. *Integrated Development Environment*) za razvoj Android aplikacija, temeljeno na IntelliJ IDEA koje je zamijenilo Eclipse. Svaki projekt u programu Android Studio sadrži jedan ili više modula s

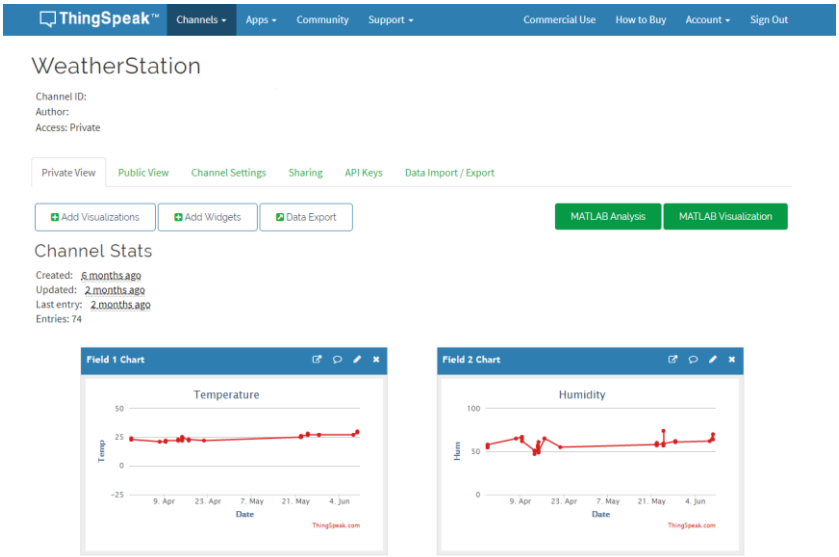
datotekama izvornog koda i datotekama resursa. Besplatan je za skidanje s interneta i korištenje i ima bogato korisničko sučelje.



Slika 4.5. Android Studio sučelje

4.2.3. Platforma ThingSpeak

Prema [14], ThingSpeak je otvorena platforma koja omogućava prikupljanje, vizualizaciju i analizu trenutnih podataka u oblaku. ThingSpeak pruža trenutne vizualizacije podataka postavljene od strane naših uređaja na ThingSpeak. Pomoću MATLAB koda u ThingSpeaku može se izvršiti online analiza i obrada podataka koji trenutno dolaze.



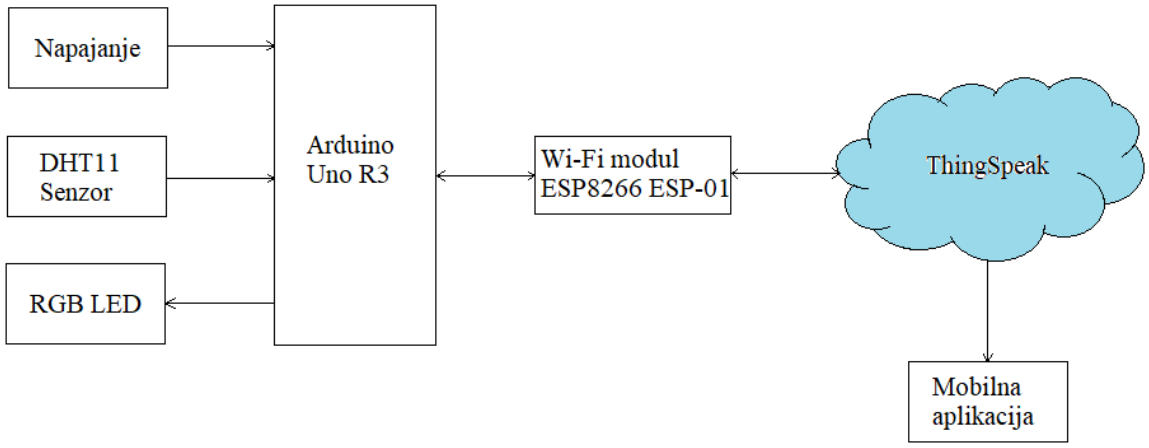
Slika 4.6. Primjer kanala na ThingSpeaku

4.3. Dizajn sustava

4.3.1. Strukturni dijagram za nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta

Nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta uglavnom se sastoji od Arduina, modula za bežično slanje i primanje podataka, napajanja i senzora za pokret, propuštanje vode, protok zraka, temperature, vlažnosti, dima i pokreta.

Slika 4.7 prikazuje način nadzora klimatskih uvjeta u poslužiteljskim prostorijama i podatkovnim središtima. Senzor temperature i vlage DHT11 mjeri vrijednost temperature i vlage i šalje podatke Arduino upravljaču. Na temelju dobivenih podataka Arduino upravljač podešava boju LED diode koja je povezana na Arduino upravljač. Arduino upravljač u određenom vremenskom intervalu šalje podatke o temperaturi i vlazi na ThingSpeak. Korisnik pritiskom na određenu tipku u aplikaciji pokreće skidanje novih podataka o temperaturi i vlazi s ThingSpeaka

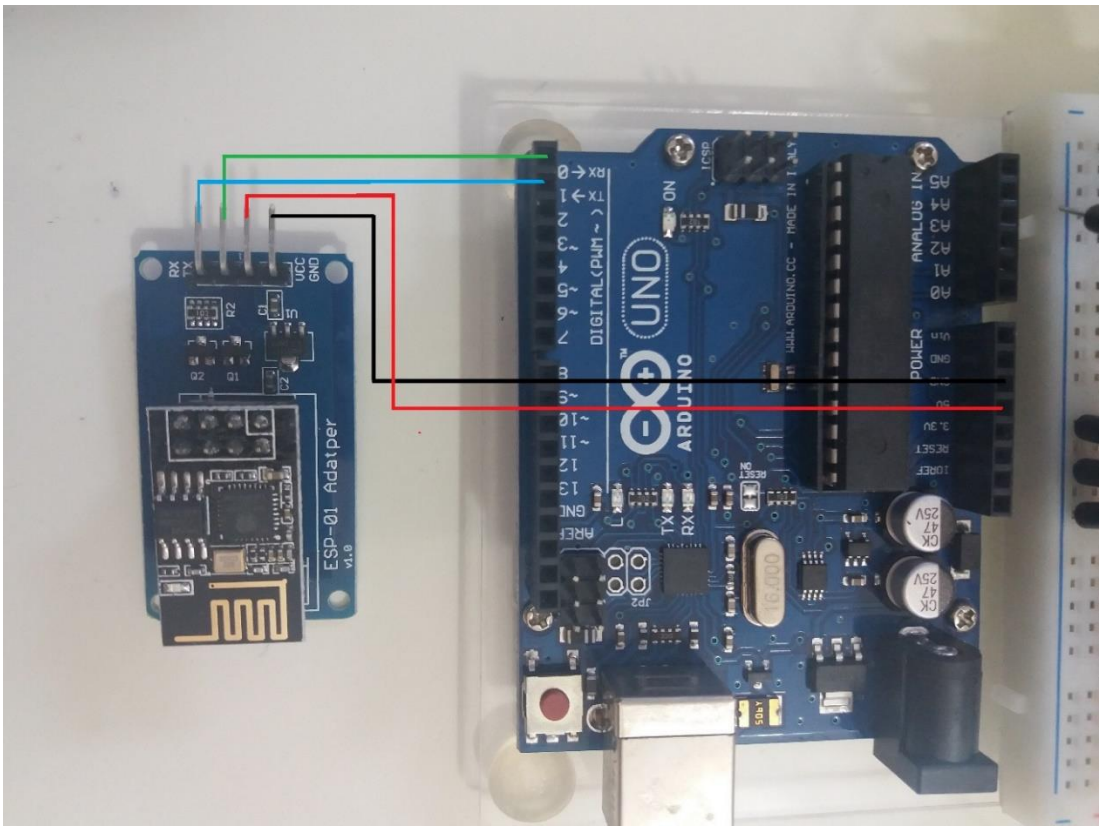


Slika 4.7. Strukturni dijagram za nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta

4.3.2. Dijagram spajanja komponenti

4.3.2.1. Spajanje Arduina Uno i bežičnog mrežnog modula

Bežični mrežni modul ESP8266 ESP-01 se spoji na adapter, a adapter se poveže s Arduino upravljačem kako je prikazano u tablici 4.2 i na slici 4.8.



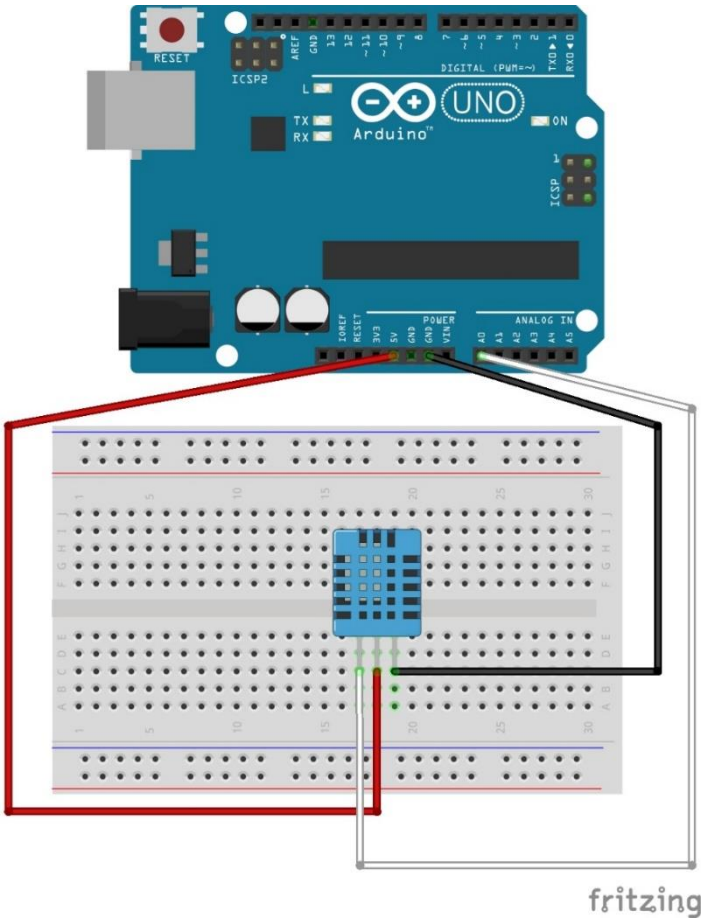
Slika 4.8. Spajanje bežičnog mrežnog modula s adapterom na upravljač Arduino Uno R3

Tablica 4.2. Spajanje bežičnog mrežnog modula ESP8266 ESP-01 na upravljač Arduino Uno R3

ESP8266 ESP-01	ARDUINO UNO R3
GND	GND
VCC	5V
TX	RX
RX	TX

4.3.2.2. Spajanje Arduina Uno i senzora temperature i vlažnosti

Spajanje DHT11 senzora s tri pina s upravljačem Arduino Uno R3, prikazano na slici 4.9, je jednostavno jer nema potrebe za korištenjem otpornika. DHT11 senzor s tri pina dolazi s montiranim otpornikom od 10000 Ω.



Slika 4.9. Shema spajanja senzora temperature i vlažnosti DHT11 s upravljačem Arduino Uno R3

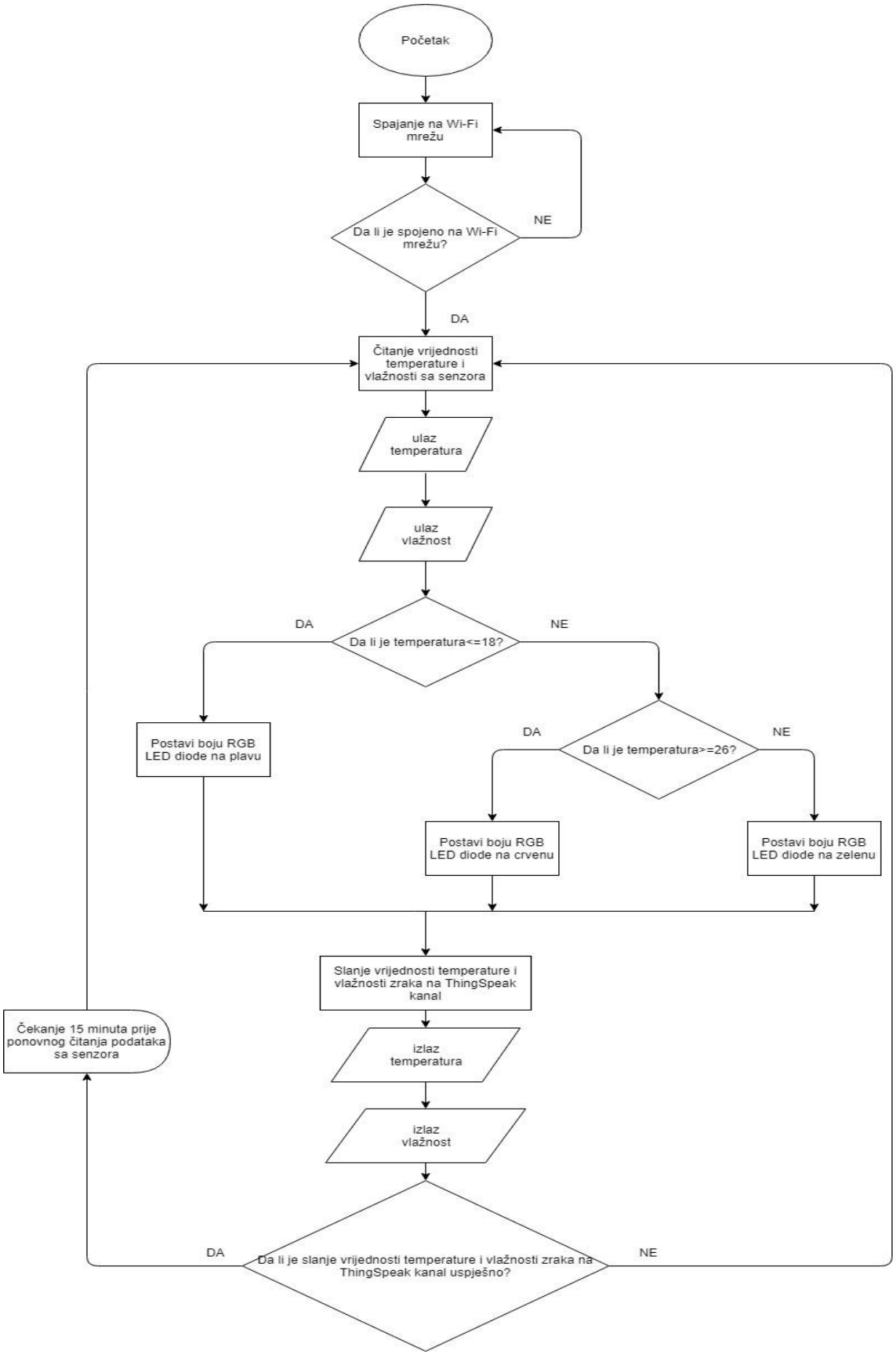
4.3.3. Algoritam upravljanja

Slike 4.10 i 4.11 prikazuju način na koji se vrši nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkvnih središta. Program napisan u Arduino integriranoj razvojnoj okolini na početku definira varijable i postavlja podatke za spajanje na bežičnu mrežu. Nakon toga dolazi do spajanja na bežičnu mrežu. Ako spajanje nije uspješno, ponovo se pokreće funkcija za spajanje na bežičnu mrežu. Ako je spajanje uspješno, prelazi se na dio koji se odnosi na čitanje podataka sa senzora. Kada se pročitaju vrijednosti sa senzora spreme se u varijable *temperatura* i *vlažnost.* Nakon spremanja pročitanih vrijednosti u odgovarajuće varijable provjerava se vrijednost temperature. Ako je vrijednost temperature manja ili jednaka 18°C RGB LED dioda će svijetliti plavo, a ako je vrijednost temperature veća ili jednaka 26°C RGB LED dioda će svijetliti crveno. U slučaju da je vrijednost temperature između te dvije vrijednosti RGB LED dioda će svijetliti zeleno. Nakon toga slijedi slanje vrijednosti temperature i vlažnosti zraka na ThingSpeak kanal. Ako slanje vrijednosti temperature i vlage na ThingSpeak nije uspjelo, ponovo se čitaju vrijednosti sa senzora, postavlja

odgovarajuća boja RGB LED diode i šalju vrijednosti temperature i vlage na ThingSpeak. Ako je slanje uspješno, čeka se petnaest minuta prije nego što se opet čitaju podaci sa senzora, postavlja odgovarajuća boja RGB LED diode i šalju vrijednosti temperature i vlage na ThingSpeak.

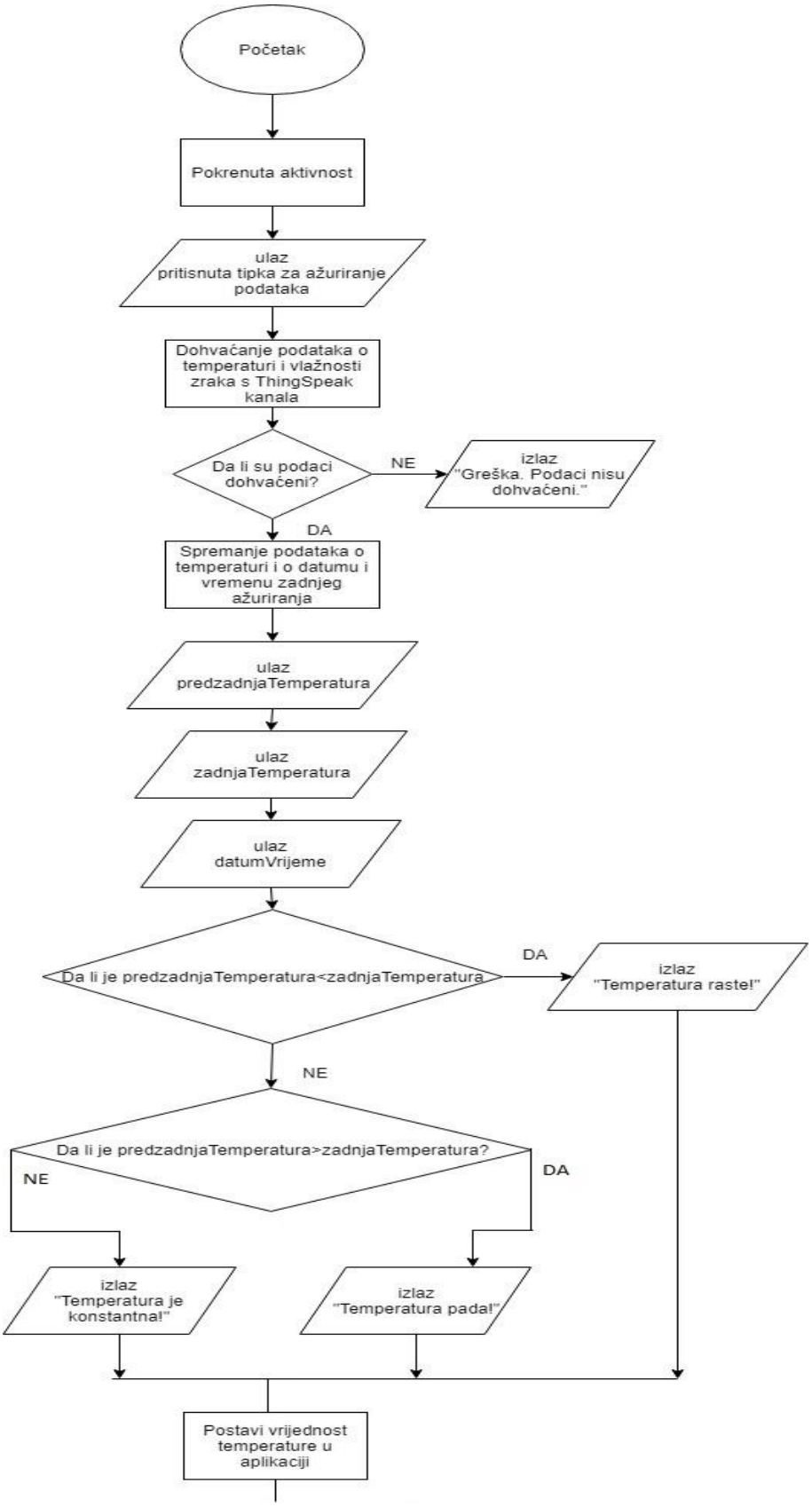
```
početak;
ulaz temperatura, vlažnost, greška;
postavi pinove za RGB LED diodu;
void setup():
    postavi pinove RGB LED diode da budu izlaz;
    spojiWiFi();
    ako je spojiWiFi() onda:
        izlaz "Spojeno na Wi-Fi";
void loop():
    start:
        greška = 0;
        pročitaj podatke sa senzora;
        temperatura = podatak o temperaturi sa senzora;
        vlažnost = podatak o vlažnosti sa senzora;
        ako je (temperatura <= 18) onda:
            postavi boju RGB LED diode na plavu;
        ako je (temperatura >= 26) onda:
            postavi boju RGB LED diode na crvenu;
        inače:
            postavi boju RGB LED diode na zelenu;
        ažurirajTemVlag();
        ako je (greška = 1) onda:
            izlaz "Vrijednosti nisu ažurirane";
            idi na start;
        pričekaj 15 minuta;
void ažurirajTempVlag():
    ažuriraj vrijednosti temperature i vlažnosti na ThingSpeaku;
    ako je uspješno ažurirano onda:
        izlaz je naredba s kojom su poslane vrijednosti na ThingSpeak;
    inače:
        izlaz je naredba za zatvaranje veze;
        greška = 1;
boolean spojiWiFi():
    spajanje na WiFi mrežu;
    ako je spajanje na mrežu uspješno onda:
        vrati true;
    inače:
        vrati false;
kraj;
```

Slika 4.10. Pseudokod za nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta za program napisan u Arduino integriranoj razvojnoj okolini

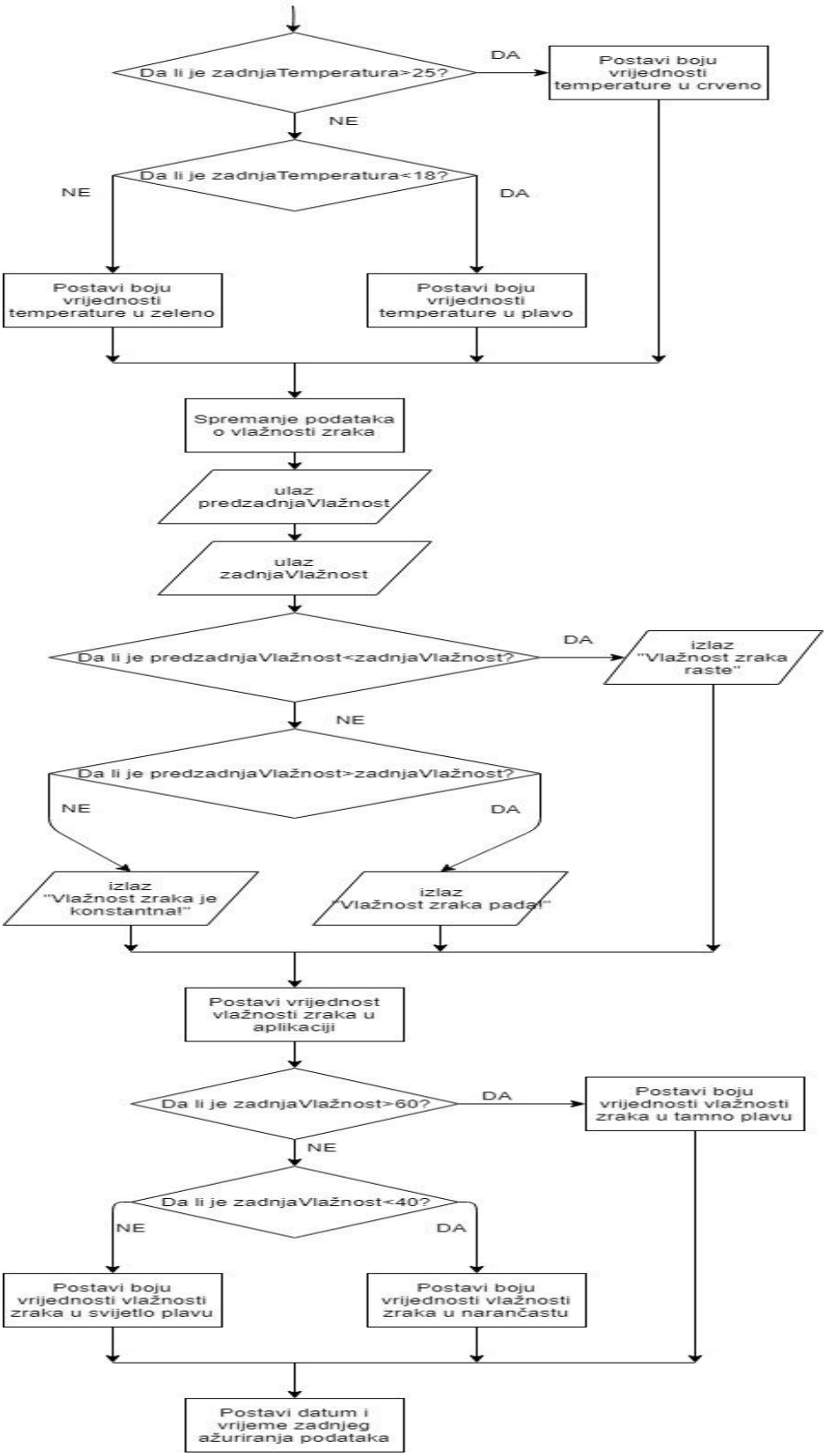


Slika 4.11. Dijagram toga za nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta za program napisan u Arduino integriranoj razvojnoj okolini

Slike 4.12 i 4.13 prikazuju način na koji se vrši nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta putem mobilne aplikacije. Program je napisan u Android Studio integriranoj razvojnoj okolini. Na početku se definiraju varijable koje će se koristiti kroz program i postavlja se izgled aplikacije. Nakon toga se osluškuje da li je korisnik pritisnuo tipku za ažuriranje podataka. Ako je korisnik pritisnuo tipku za ažuriranje podataka pokreće se dohvaćanje podataka o temperaturi i vlažnosti zraka s ThingSpeak kanala. Ako podaci nisu uspješno dohvaćeni program javlja korisniku da je došlo do greške i da podaci nisu preuzeti. U slučaju da su podaci uspješno dohvaćeni, podaci o zadnje dvije vrijednosti temperature i datum i vrijeme zadnjeg ažuriranja se spremaju. Nakon toga program ispituje da li je zadnja vrijednost temperature veća od predzadnje vrijednosti temperature. Ako je zadnja vrijednost temperature veća od predzadnje vrijednosti temperature u aplikaciji se ispisuje da je temperatura u porastu. U slučaju da zadnja vrijednost temperature nije veća od predzadnje vrijednosti temperature ispituje se da li je predzadnja vrijednost temperature veća od zadnje vrijednosti temperature. Ako je predzadnja vrijednost temperature veća od zadnje vrijednosti temperature u aplikaciji se ispisuje da je temperatura u padu. Ako temperatura nije ni u porastu, a ni u opadanju, onda se u aplikaciji ispisuje da je vrijednost temperature konstantna. Nakon toga program ispituje zadnju vrijednost temperature. Ako je zadnja vrijednost temperature veća od 25°C u aplikaciji se vrijednost temperature ispisuje crvenom bojom koja označava da je temperatura visoka, a u slučaju da je manja od 18°C vrijednost temperature u aplikaciji se ispisuje plavom bojom koja označava da je temperatura niska. Kada je vrijednost temperature između 18°C i 25°C u aplikaciji se vrijednost temperature ispisuje zelenom bojom što znači da je temperatura u dozvoljenim granicama. Na isti način kao što se ispituje rast i pad temperature, tako se ispituje i rast i pad vlažnosti zraka. Kod postavljanja određenih boja za vrijednost vlažnosti zraka u aplikaciji ograničenja su drugačija. Vrijednost vlažnosti zraka se postavlja u tamno plavu boju ako je zadnja vrijednost vlažnosti zraka iznad 60%. Kada je vrijednost vlažnosti zraka u tamno plavoj boji to znači da je prevelika vlažnost zraka. Ako je vlažnost zraka ispod 40% boja vrijednosti vlažnosti zraka u aplikaciji se postavlja u narančastu koja označava da je vlažnost zraka preniska. Idealna vrijednost vlažnosti zraka, a to je između 40% i 60%, biti će postavljena u svijetlo plavu boju. Na kraju se postavlja datum i vrijeme zadnjeg ažuriranja temperature i vlažnosti zraka.



Slika 4.12. Dijagram toga za nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta za program napisan u Android Studio integriranoj razvojnoj okolini

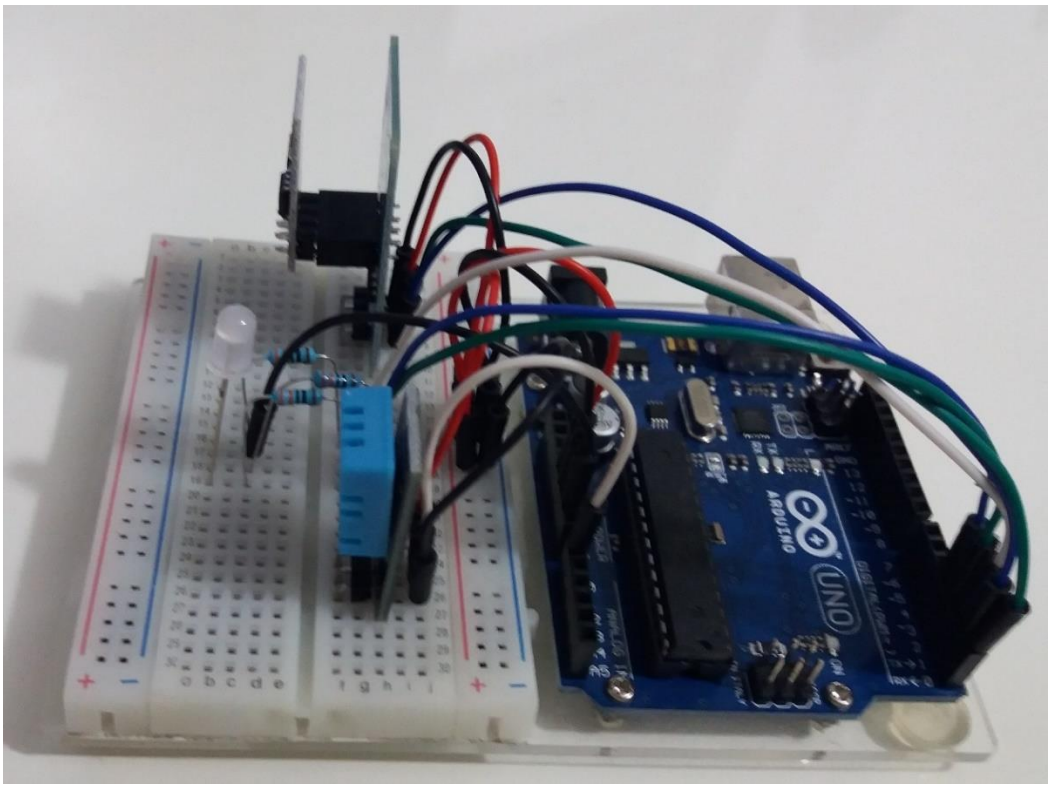


Slika 4.13. Dijagram toga za nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta za program napisan u Android Studio integriranoj razvojnoj okolini

5. OSTVARENJE SKLOPOVSKOG I PROGRAMSKOG RJEŠENJA SUSTAVA

5.1. Sklopovsko rješenje sustava

Na Arduino Uno R3 se spaja bežični mrežni modul ESP8266 ESP-01 pomoću adaptera prema shemi na slici 4.8, senzor temperature i vlažnosti DHT11 prema shemi na slici 4.9 i RGB LED dioda koja prikazuje je li trenutno očitana temperatura normalna, niska ili visoka. Kompletno sklopovsko rješenje je prikazano slikom 5.1.



Slika 5.1. Prikaz skopovskog rješenja

5.2. Programsko rješenje sustava

5.2.1. Programsko rješenje na Arduinu

5.2.1.1. Povezivanje s bežičnom mrežom

Na slici 5.2 funkcija „*connectWiFi()*“ omogućuje povezivanje bežičnog modula ESP8266 ESP-01 s bežičnom mrežom. AT naredba „*AT+CWMODE=1*“ postavlja bežični način rada modula ESP8266 ESP-01 na način rada stanice (klijent), a pomoću „*AT+CWJAP*“ postavlja se pristupna

točka na koju će se bežični modul spojiti. Na pristupnu točku se bežični mrežni modul spaja pomoću naziva bežične mreže i lozinke.

```
boolean connectWiFi(){
  Serial.println("AT+CWMODE=1"); //To set the current Wi-Fi mode of ESP8266. 1 is Station mode.
  delay(2000);
  String cmd="AT+CWJAP=\""; //AT+CWJAP  to set the AP to which the ESP8266 Station needs to be connected.
  cmd+=SSID;
  cmd+="\",\"";
  cmd+=PASS;
  cmd+="\"";
  Serial.println(cmd);
  delay(5000);
  if(Serial.find("OK")){
    return true;
  }else{
    return false;
  }
```

Slika 5.2. Funkcija za povezivanje bežičnog mrežnog modula na bežičnu mrežu

5.2.1.2. Slanje vrijednosti varijabli na ThingSpeak

Na slici 5.3 funkcija „*updateTempHum()*“ služi za slanje vrijednosti temperature i vlage na ThingSpeak kanal. AT naredba „*AT+CIPSTART = tip, adresa, port*“ uspostavlja TCP vezu kao klijent, gdje „*tip*“ može biti TCP ili UDP, a „*adresa*“ i „*port*“ se odnose na udaljenu IP adresu i port. AT naredba „*AT+CIPSEND = duljina*“ šalje podatke, gdje je „*duljina*“ duljina podataka koji se šalju, a naredba „*AT+CIPCLOSE*“ zatvara TCP ili UDP vezu, u ovom slučaju TCP vezu.

```
void updateTempHum(){
  String cmd = "AT+CIPSTART=\"TCP\",\""; //Establishes TCP Connection
  cmd += IP;
  cmd += "\",80";
  Serial.println(cmd);
  delay(2000);
  if(Serial.find("Error")){
    return;
  }
  cmd = msg ;
  cmd += "&field1="; //field 1 for temperature
  cmd += tempC;
  cmd += "&field2="; //field 2 for humidity
  cmd += String(hum);
  cmd += "\r\n";
  Serial.print("AT+CIPSEND="); //Start sending data
  Serial.println(cmd.length());
  if(Serial.find(">")){
    Serial.print(cmd);
  }
  else{
    Serial.println("AT+CIPCLOSE"); //Closes the TCP Connection.
    //Resend...
    error=1;
  }
}
```

Slika 5.3. Funkcija koja šalje podatke o temperaturi i vlažnosti sa senzora na ThingSpeak kanal

5.2.2. Mobilna aplikacija

Za obradu i dohvaćanje podataka s ThingSpeak kanala korištena je *AsyncTask* klasa.

Prema [15], *AsyncTask* klasa omogućuje izvršavanje pozadinskih operacija i objavljivanje rezultata na niti korisničkog sučelja (engl. *UI thread*) bez manipuliranja nitima i/ili rukovateljima (engl. *handlers*). Asinkroni zadatak definiran je od tri generička tipa, a to su:

- *Params* – tip parametara poslanih zadatku po izvršenju,
- *Progress* – tip jedinica napretka objavljenih tijekom pozadinskog računanja,
- *Result* – tip rezultata pozadinskog računanja.

i četiri koraka, kroz koje se prolazi kada se izvrši asinkroni zadatak, a to su:

1. *onPreExecute()* – pozvana na nit korisničkog sučelja prije nego što se zadatak izvrši,
2. *doInBackground(Params...)* – pozvana na pozadinsku nit neposredno nakon što *onPreExecute()* završi izvršavanje. Ovaj se korak koristi za izvođenje pozadinskog računanja koje može potrajati dugo. U ovom koraku se može koristiti *publishProgress(Progress...)* za objavljivanje jedne ili više jedinica napretka. Ove vrijednosti se objavljuju na niti korisničkog sučelja u koraku *onProgressUpdate(Progress...),*
3. *onProgressUpdate(Progress...)* – pozvana je na nit korisničkog sučelja nakon poziva za *publishProgress(Progress...)*. Ova metoda služi za prikaz bilo kojeg oblika napretka u korisničkom sučelju dok se računanje u pozadini i dalje izvodi,
4. *onPostExecture(Result)* – pozvana na nit korisničkog sučelja nakon završetka pozadinskog računanja.

Slika 5.4 prikazuje izgled *AsyncTask* klase u aplikaciji ovog rada. Korak „*doInBackground(URL... urls)“* obavlja dohvaćanje podataka s ThingSpeak kanala i vraća parametar „*result“* koji je rezultat pozadinskog računanja. U koraku „*onPostExecute(String[] result)“* prima se rezultat pozadinskog računanja i ispisuju vrijednosti u aplikaciji. Tip *Progress* nije bio korišten pa je označen kao *Void*.

```
class GetData extends AsyncTask<URL, Void, String[]> {
    protected String[] doInBackgroud(URL... urls) {
        .
        .
        .
        return result;
    }
    protected void onPostExecute(String[] result){
        .
        .
        .
    }
}
```

Slika 5.4. Klasa *AsyncTask* u aplikaciji

5.2.2.1. Dohvaćanje podataka s ThingSpeaka

Podaci s ThingSpeak kanala se dohvaćaju u JSON formatu. JSON podaci su napisani kao ključ/vrijednost parovi i odvojeni su zarezima. Objekti su u vitičastim zagradama, a nizovi u uglatim. Slika 5.5 prikazuje ThingSpeak podatke u JSON formatu. *Channel* i *feeds* su objekti, a objekt *feeds* sadrži niz od dva objekta. U objektu *channel* sadržane su sve informacije o ThingSpeak kanalu, a u objektu *feeds* nalaze se zadnje dvije vrijednosti temperature i vrijeme kada su postavljene na ThingSpeak kanal.


```
{"channel":
    {"id":431197,
    "name":"Weather Station",
    "latitude":"0.0",
    "longitude":"0.0",
    "field1":"Temperature",
    "field2":"Humidity",
    "created_at":"2018-02-20T14:06:47+01:00",
    "updated_at":"2018-06-09T19_12_32+02:00",
    "last_entry_id":76}
 "feeds":[{"created_at":"2018-08-31T17:51:49+02:00", "entry_id":75, "field1":"26.0"},
        {"created_at":"2018-08-31T17:53:26+02:99", "entry_id":76, "field1":"26.0"}]
}
```


Slika 5.5. ThingSpeak podaci u JSON formatu

Dio programskog koda na slici 5.6 prikazuje način dohvaćanja zadnje dvije vrijednosti temperature s ThingSpeaka. Pozivajući „*url1.openConnection*“ dobit će se nova *HttpURLConnection* veza, a pomoću „*urlConnection.disconnect()*“ se prekida *HttpURLConnection* veza. Podatke koji se dobivaju moguće je pročitati pomoću „*urlConnection1.getInputStream()*“ i „*BufferedReader*“, a u „*StringBuilder*“ objekt se spremaju podaci koji su pročitani. Na isti način se dohvaćaju i zadnje dvije vrijednosti vlažnosti zraka, samo se u poveznici broj polja postavi na 2.

```
URL url1 = new URL( spec: "https://api.thingspeak.com/channels/431197/fields/1.json?api_key" +
        "=IKKHFPXVQ2WTM6NO&results=2&timezone=Europe%2FZagreb");
HttpURLConnection urlConnection1 = (HttpURLConnection) url1.openConnection();

InputStream stream1 = new BufferedInputStream(urlConnection1.getInputStream());
BufferedReader bufferedReader1 = new BufferedReader(new InputStreamReader(stream1));
StringBuilder builder1 = new StringBuilder();

while((inputString1 = bufferedReader1.readLine()) != null){
    builder1.append(inputString1);
}

result[1] = builder1.toString();
bufferedReader1.close();
urlConnection1.disconnect();
```

Slika 5.6. Dio koda koji služi za dohvaćanje zadnje dvije vrijednosti temperature s ThingSpeaka

5.2.2.2. Prikaz vrijednosti temperature i vlažnosti u aplikaciji

Slika 5.7 pokazuje način na koji se iz JSON-a dobiva vrijednost iz željenog objekta pomoću ključa. Na isti način se dobiva i vrijednosti vlažnosti zraka. Kako je i prikazano na slici 5.5 *feeds* je objekt koji sadrži niz u kojem se nalaze dva objeka. Da bi se pristupilo dvjema zadnjim vrijednostima temperature potrebno je doći do niza koji se nalazi u objektu *feeds,* a to je moguće postići pomoću „*JSONArray array = root.getJSONArray("feeds")*. Da bi se došlo do svakog objekta u nizu potrebno je proći kroz cijeli niz pomoću *for* petlje. Dio „*DT = object.getString("created_at")*“ odnosi se na dobivanje datuma i vremena zadnjeg slanja podataka na ThingSpeak.

```
JSONObject root = new JSONObject(result[1]);
JSONArray array = root.getJSONArray( name: "feeds");
for(int i=0;i<array.length();i++)
{
    JSONObject object= array.getJSONObject(i);
    TempValue[i] = object.getDouble( name: "field1");
    DT = object.getString( name: "created_at");
}
```

Slika 5.7. Dobivanje vrijednosti temperature iz JSON-a

Dio koda prikazan na slici 5.8 osim što služi za ispisivanje vrijednosti temperature u aplikaciji, također služi za postavljanje određene boje vrijednosti temperature ovisno o tome da li je temperatura visoka, niska ili normalna. Na isti način se postavlja i vrijednost vlažnosti zraka.

```
t1.setText(String.valueOf(TempValue[1]) + "°C ");
if(TempValue[1] > 25){
    t1.setTextColor(Color.parseColor( colorString: "#ff0000"));
}
else if(TempValue[1] < 18){
    t1.setTextColor(Color.parseColor( colorString: "#0000ff"));
}
else{
    t1.setTextColor(Color.parseColor( colorString: "#00ff00"));
}
```

Slika 5.8. Dio koda koji postavlja vrijednosti temperature u aplikaciji u određenu boju

5.2.2.3. Predviđanje rasta i pada temperature i vlažnosti

Kako je prikazano na slici 5.9, kada se dobiju vrijednosti temperature iz JSON-a (što je objašnjeno uz sliku 5.7), uspoređuju se zadnje dvije vrijednosti temperature pomoću metode *compareTo* i u ovisnosti o rezultatu u aplikaciji se ispisuje da li vrijednost temperature raste, pada ili je konstantna.

```
JSONObject root = new JSONObject(result[1]);
JSONArray array = root.getJSONArray( name: "feeds");
for(int i=0;i<array.length();i++)
{
    JSONObject object= array.getJSONObject(i);
    TempValue[i] = object.getDouble( name: "field1");
    DT = object.getString( name: "created_at");
}
Integer TempResult = new Double(TempValue[0]).compareTo(new Double(TempValue[1]));
if(TempResult < 0){
    t6.setText("Temperatura raste!");
}
else if(TempResult > 0){
    t6.setText("Temperatura pada!");
}
else{
    t6.setText("Temperatura je konstantna!");
}
```

Slika 5.9. Dio koda koji uspoređuje zadnje dvije vrijednosti temperature

Kako je prikazano na slici 5.10, predviđanje rasta i pada vlažnosti odvija se isto kao i kod predviđanja rasta i pada temperature. Pomoću metode *compareTo,* i na temelju rezultata u aplikaciji, se ispisuje da li vrijednost vlažnosti zraka raste, pada ili je konstantna.

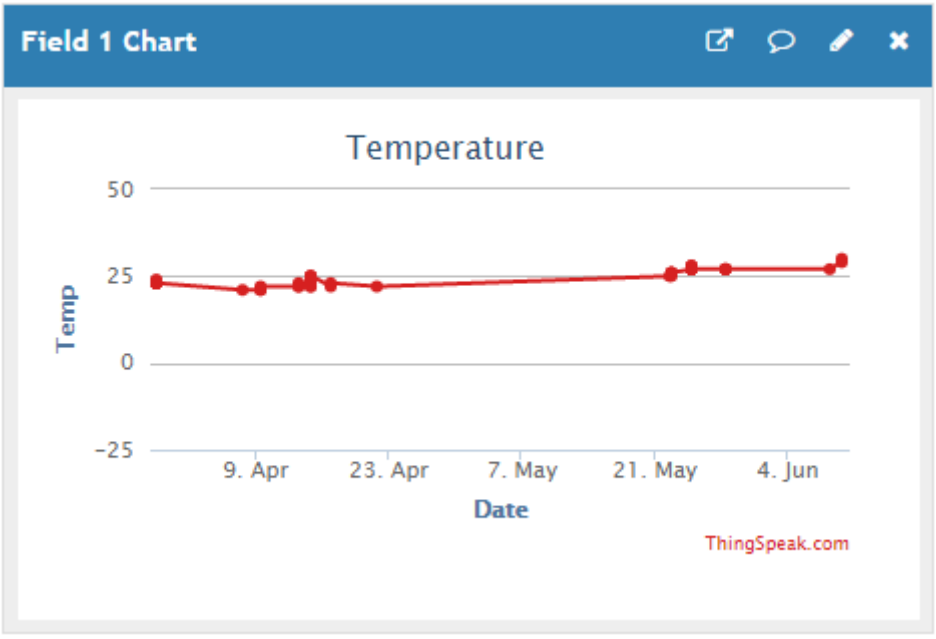
```
JSONObject root2 = new JSONObject(result[2]);
JSONArray array2 = root2.getJSONArray( name: "feeds");
for(int i=0;i<array2.length();i++)
{
    JSONObject object= array2.getJSONObject(i);
    HumValue[i] = object.getDouble( name: "field2");
}
Integer HumResult = new Double(HumValue[0]).compareTo(new Double(HumValue[1]));
if(HumResult < 0){
    t7.setText("Vlaznost raste!");
}
else if(HumResult > 0){
    t7.setText("Vlaznost pada!");
}
else{
    t7.setText("Vlaznost je konstantna!");
}
```

Slika 5.10. Dio koda koji uspoređuje zadnje dvije vrijednosti vlažnosti zraka

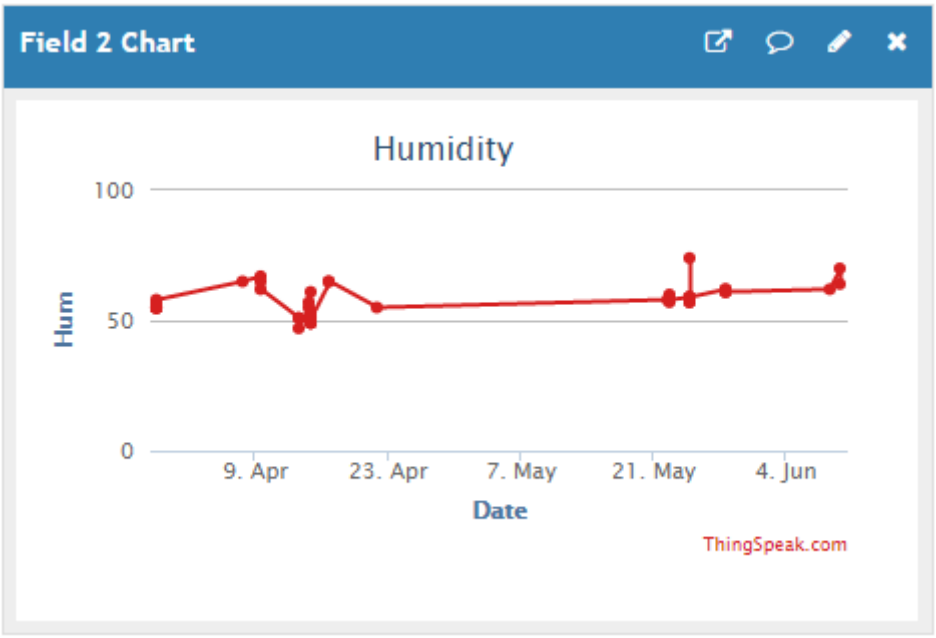
Na početku, aplikacija je sadržavala tri poveznice od kojih je jedna dohvaćala zadnje podatke o temperaturi i vlazi i vrijeme zadnjeg ažuriranja informacija, a druge dvije su dohvaćale po dvije zadnje vrijednosti temperature i vlažnosti zraka. Mobilna aplikacija je zbog toga bila bespotrebno usporena, pa se zbog toga izbacila prva poveznica i željene su se informacije uspjele dohvatiti iz preostale dvije poveznice.

5.2.3. Grafički prikazi na platformi ThingSpeak

Slike 5.11 i 5.12 pokazuju kako su vrijednosti temperature i vlažnosti, koje su poslane na ThingSpeak, grafički prikazane na ThingSpeak kanalu. Sustav će u stvarnim uvjetima slati vrijednosti temperature i vlažnosti na ThingSpeak kanal svakih petnaest minuta, pa će ovaj graf izgledati drugačije tj. na X osi neće biti prikazani datumi, nego vrijeme kada je koja vrijednost poslana na ThingSpeak kanal. Ako se ThingSpeak kanal postavi kao javni, preko internet preglednika, na mobitelu ili računalu, će biti moguće pristupiti ThingSpeak kanalu i pregledati kretanje vrijednosti temperature i vlage kroz jedno vremensko razdoblje.



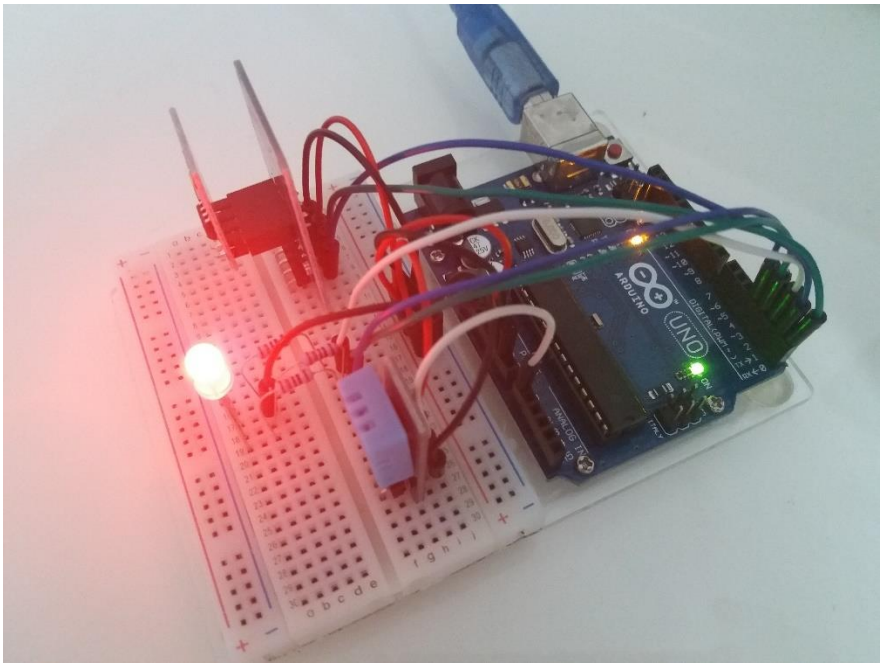
Slika 5.11. Graf prikazuje primljene vrijednosti temperature kroz duže vremensko razdoblje



Slika 5.12. Graf prikazuje primljene vrijednosti vlažnosti zraka kroz duže vremensko razdoblje

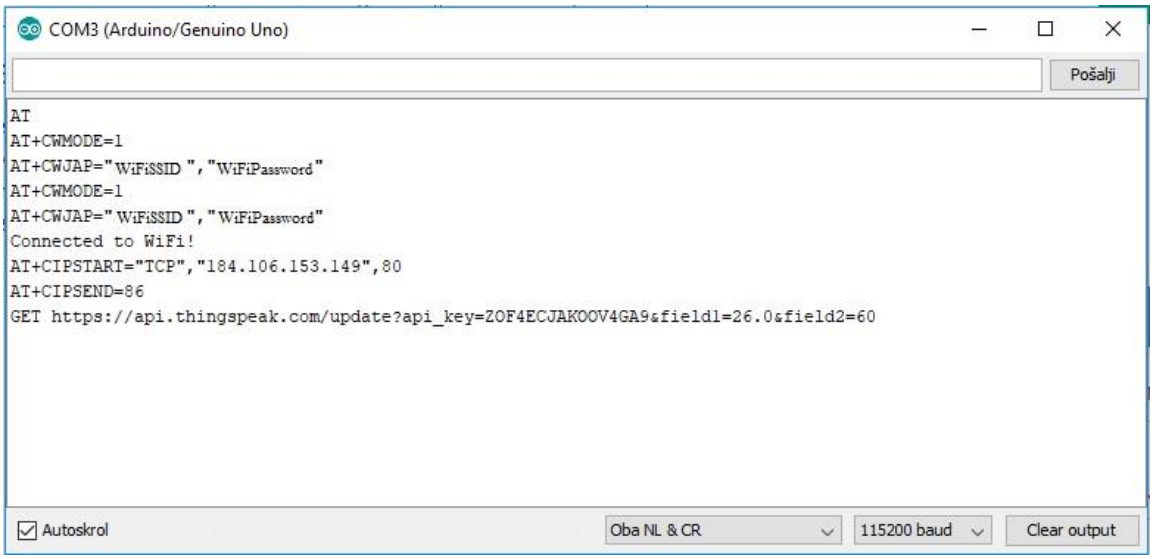
6. NAČIN KORIŠTENJA I ISPITIVANJE RADA SUSTAVA

Sklopovski dio rješenja sustava, prikazan na slici 6.1, se priključi na baterije ili pomoću USB kabela na računalo. Kada se pročitaju vrijednosti sa senzora, RGB LED dioda zasvijetli u određenoj boji. Na slici je vidljivo da dioda svijetli u crvenoj boji što znači da je vrijednost temperature veća ili jednaka 26°C. Potrebno je oko trideset i pet sekundi da se učitaju vrijednosti sa senzora, pošalju na ThingSpeak kanal i da aplikacija može dohvatiti podatke s ThingSpeak kanala.



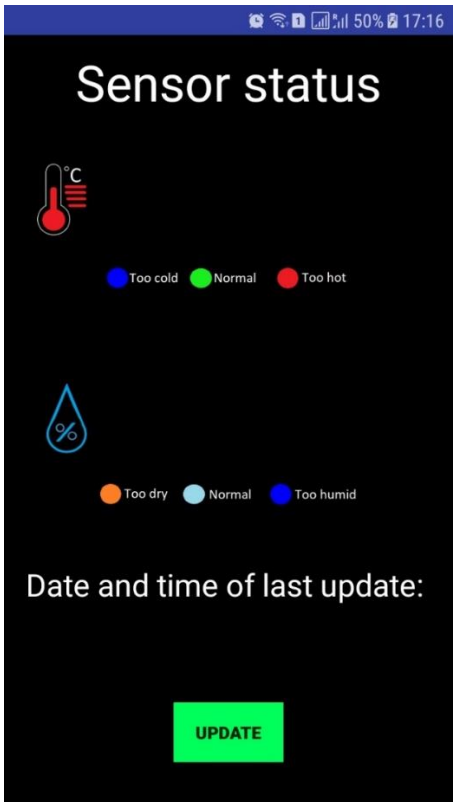
Slika 6.1. Prikaz sklopovskog dijela sustava u radu

Slika 6.2 je ispis kada se upali *Serial Monitor* u Arduino integriranoj razvojnoj okolini. Vidljivo je da je prva naredba *AT* kojom se provjerava da li je bežični mrežni modul dobro spojen. Kako je povratna informacija *OK* dolazi se do funkcije za spajanje na mrežu. *„AT+CWMODE=1“* postavlja bežični način rada modula ESP8266 ESP-01 na način rada stanice. Taj način se koristi kada se želi da se bežični mrežni modul spoji na neku pristupnu točku. *„AT+CWJAP=“WiFiSSID“,“WiFiPassword““* spaja bežični mrežni modul na određenu pristupnu točku pomoću naziva mreže i lozinke. Spajanje je uspjelo pa se odvija slanje podataka na ThingSpeak kanal. Prvo se uspostavlja TCP veza, a to se odvija u koraku *„AT+CIPSTART=“TCP“,“184.106.153.149“,80“,* a zatim se šalju podaci. Ako je slanje uspješno ispisuje se cijela poveznica pomoću koje se šalju podaci na ThingSpeak kanal.



Slika 6.2. Prikaz spajanja na bežičnu mrežu

Slika 6.3 je prikaz početnog zaslona aplikacije za nadzor klimatskih uvjeta. Na slici su prikazane oznake za temperaturu i vlažnost, kao i objašnjenja što znači svaka boja za vrijednost temperature i vlažnosti. Objašnjenja služe da bi se razumjele boje vrijednosti temperature i vlažnosti kada se dohvate s ThingSpeak kanala i ispišu u aplikaciji. Normalna vrijednost temperature bit će ispisana u zelenoj boji, a normalna vrijednost vlažnosti zraka će biti ispisana u svijetloplavoj boji. U slučaju prekoračenja normalne vrijednosti temperature, vrijednost temperature će biti ispisana crvenom bojom, a u slučaju prekoračenja normalne vrijednosti vlažnosti zraka, vrijednost vlažnosti zraka će biti ispisana u tamnoplavoj boji. Ako je vrijednost temperature dobivene sa senzora niža od normalne, vrijednost temperature će biti ispisana tamnoplavom bojom, a ako je vrijednost vlažnost zraka niža od normalne, vrijednost vlažnosti zraka biti će ispisana narančastom bojom. Tablice 6.1 i 6.2 prikazuju kako se mijenjaju boje vrijednosti temperature i vlažnosti zraka u aplikaciji na temelju podataka sa senzora. Također postoji informacija o datumu i vremenu kada su zadnji put poslane vrijednosti temperature i vlažnosti na ThingSpeak kanal. Na dnu aplikacije nalazi se tipka na čiji pritisak zahtijevamo dohvaćanje zadnjih vrijednosti temperature i vlažnosti s ThingSpeak kanala. Ako se vrijednosti nisu uspjele dohvatiti s ThingSpeak kanala, aplikacija će ispisati upozorenje da je došlo do pogreške i sa podaci nisu preuzeti. Arduino programski kod obavlja čitanje podataka o temperaturi i vlažnosti zraka sa senzora i šalje ih na ThingSpeak kanal, a mobilna aplikacija dohvaća te podatke i obrađuje ih.



Slika 6.3. Izgled početnog zaslona aplikacije

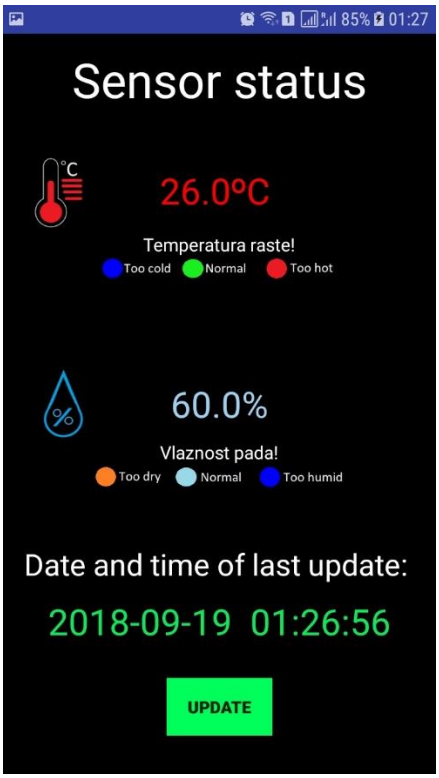
Tablica 6.1. Tablica koja prikazuje kako se mijenja boja vrijednosti temperature u mobilnoj aplikaciji na temelju vrijednosti temperature sa senzora

TEMPERATURA t	MOBILNA APLIKACIJA
t<18°C	Vrijednost temperature je označena plavom bojom
t>25°C	Vrijednost temperature je označena crvenom bojom
17°C<t<26°C	Vrijednost temperature je označena zelenom bojom

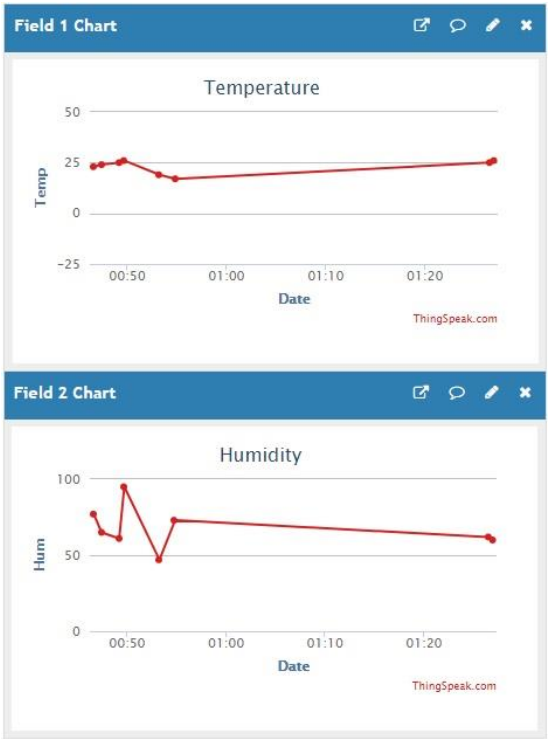
Tablica 6.2. Tablica koja prikazuje kako se mijenja boja vrijednosti vlažnosti zraka u mobilnoj aplikaciji na temelju vrijednosti vlažnosti zraka sa senzora

VLAŽNOST ZRAKA h	MOBILNA APLIKACIJA
h<40%	Vrijednosti vlažnosti zraka označena narančastom bojom
h>60%	Vrijednost vlažnosti zraka označena tamnoplavom bojom
39%<h<61%	Vrijednost vlažnosti zraka označena svijetloplavom bojom

Slike 6.4 i 6.5 prikazuju izgled mobilne aplikacije i ThingSpeak kanala kada je vrijednost temperature iznad dozvoljene granice. U aplikaciji je vrijednost temperature prikazana crvenom bojom koja označava da je temperatura previsoka. Usporedno s tim, na grafičkom prikazu je vidljivo da je temperatura previsoka. Također, u aplikaciji je prikazano da temperatura raste što možemo isto tako vidjeti i na grafičkom prikazu. Vlažnost zraka je unutar dozvoljenih granica pa je zbog toga u aplikaciji prikazana svijetloplavom bojom. U aplikaciji je prikazano da je vlažnost zraka u padu što se može potvrditi gledajući grafički prikaz.

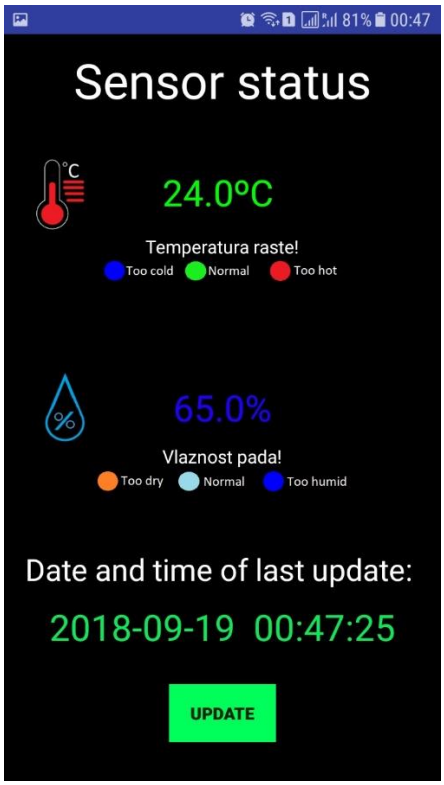


Slika 6.4. Izgled aplikacije kada je vrijednost temperature iznad dozvoljene granice

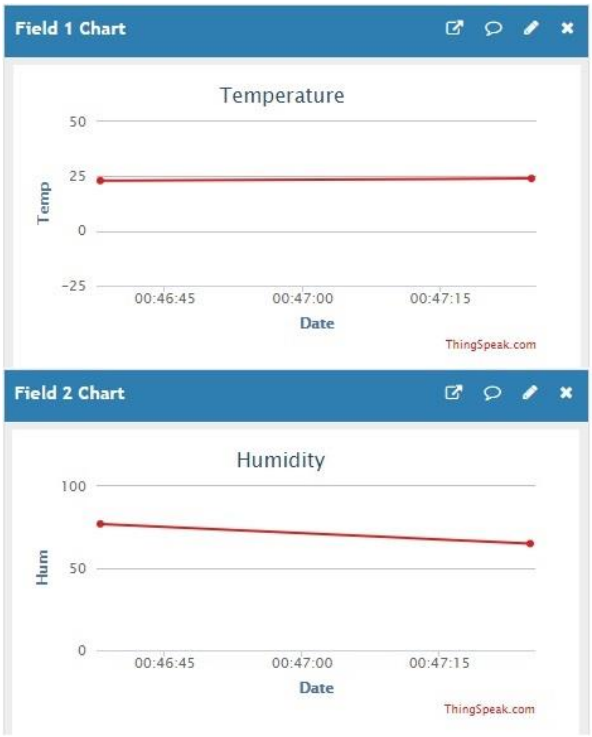


Slika 6.5. Grafički prikazi na platformi ThingSpeak kada je zadnja vrijednost temperature iznad dozvoljene granice

Slike 6.6 i 6.7 prikazuju izgled mobilne aplikacije i ThingSpeak kanala kada je vrijednost temperature unutar dozvoljenih granica. U aplikaciji je vidljivo da je vrijednost temperature označena zelenom bojom što znači da je temperatura unutar dozvoljenih granica, a to je moguće potvrditi gledajući grafički prikaz na platformi ThingSpeak. Na grafu koji prikazuje vrijednosti temperature vidljivo je da je temperatura u porastu, a to se postiže na način da program uspoređuje zadnje dvije vrijednosti temperature i u aplikaciji ispisuje da je temperatura u porastu. S grafa, a i u aplikaciji, je vidljivo da je vlažnost zraka iznad dozvoljene granice, ali da je u padu, pa je zbog toga vrijednost vlažnosti zraka označena tamnoplavom bojom. Vrijeme potrebno da mobilna aplikacija dohvati podatke s ThingSpeak kanala je oko dvije sekunde. Kod nadzora poslužiteljskih prostorija i podatkovnih središta u mobilnoj aplikaciji bi bilo više podataka o temperaturi i vlažnosti zraka koji su prikupljeni s više senzora, pa bi aplikacija morala sadržavati podatke o tome s kojeg je senzora prikupljen koji podatak, tako da bi klima uređaj hladio ili zagrijavao samo određeni dio prostorije i time bi bila smanjena potrošnja energije.

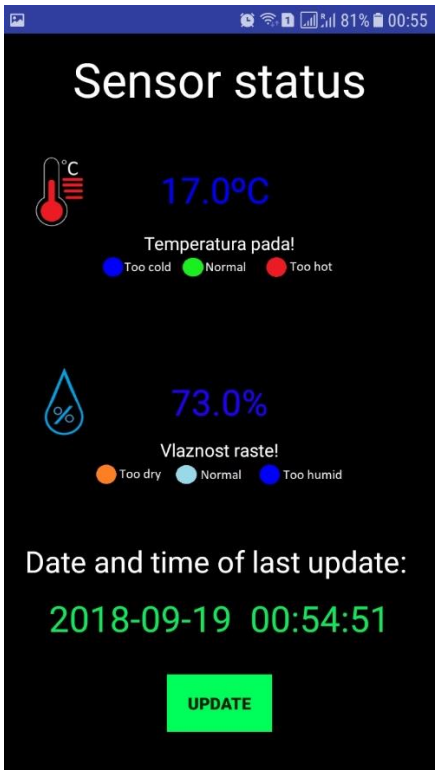


Slika 6.6. Izgled aplikacije kada je temperatura unutar dozvoljenih granica

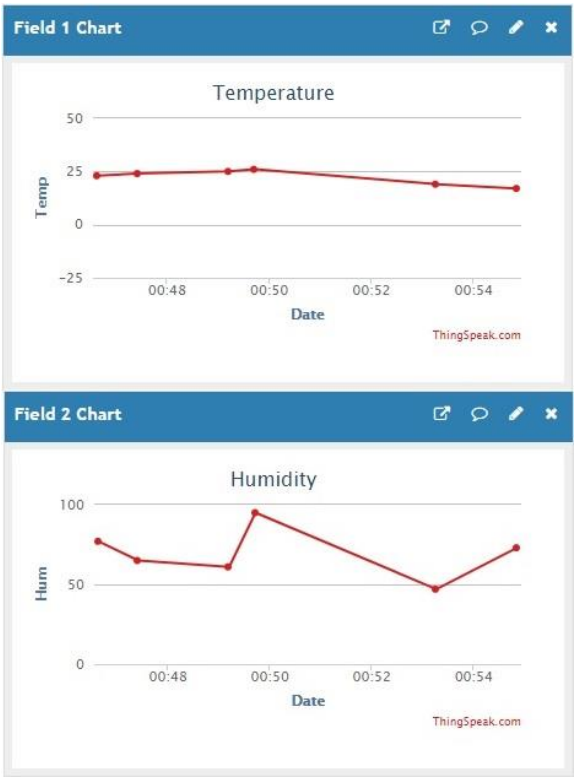


Slika 6.7. Grafički prikazi na platformi ThingSpeak kada je zadnja vrijednost temperature unutar dozvoljenih granica

Slike 6.8 i 6.9. prikazuju izgled mobilne aplikacije i ThingSpeak kanala kada je vrijednost temperature ispod dozvoljene granice. Aplikacija prikazuje da je vrijednost temperature 17°C, odnosno da je previše hladno, pa je zbog toga vrijednost temperature označena plavom bojom. Na grafu se može vidjeti da je temperatura u padu i da je ispod dozvoljene granice. Kada se pogleda vrijednost vlažnosti zraka vidi se da je iznad dozvoljene granice jer je označena tamnoplavom bojom. Prema grafu vlažnost zraka je u porastu, a aplikacija to i prikazuje. U proširenoj verziji ovog rada bilo bi moguće aplikaciju nadograditi da pokreće klima uređaj, pa bi na temelju ovih podataka aplikacija pokrenula klima uređaj kako bi se zagrijala prostorija jer je vrijednost temperature ispod dozvoljene granice, te smanjila vlažnost zraka jer je iznad dozvoljene granice. Uspoređujući podatke o temperaturi i vlažnosti zraka na ThingSpeak kanalu i one prikazane u aplikaciji vidljivo je da aplikacija točno prikazuje navedene podatke. Isto tako vidljivo je da predviđanje rasta i pada temperature i vlažnosti ispravno funkcionira, kao i mijenjanje boja vrijednosti temperature i vlažnosti zraka na temelju podataka dobivenih sa senzora.



Slika 6.8. Izgled aplikacije kada je temperatura ispod dozvoljene granice



Slika 6.9. Grafički prikazi na platformi ThingSpeak kada je zadnja vrijednost temperature ispod dozvoljene granice

7. ZAKLJUČAK

U radu su obrađeni problemi klimatizacije poslužiteljskih prostorija i podatkovnih središta, objašnjeni su neki osnovni pojmovi kibernetsko-fizikalnih sustava, te navedena i razlika između kibernetsko-fizikalnih i ugradbenih sustava. Objašnjeni su i neki osnovni načini klimatizacije poslužiteljskih prostorija i podatkovnih središta. Najbolji način nadzora poslužiteljskih prostorija i podatkovnih središta je pomoću bežičnih senzorskih mreža, tako da se senzori temperature i vlažnosti postave na svaku poslužiteljsku policu i na taj način se promatraju vrijednosti temperature i vlažnosti u pojedinom dijelu poslužiteljske prostorije. Takav način je dobar jer se povećava učinkovitost hlađenja i smanjuje se velika potrošnja energije. Za smanjenje velike potrošnje energije pametno je predvidjeti rast i pad temperature i vlage i na taj način spriječiti da vrijednosti temperature i vlage dosegnu kritičnu vrijednost. Za rješenje sklopovskog dijela rada korišten je upravljač Arduino Uno R3, DHT11 senzor temperature i vlažnosti i bežični mrežni modul ESP8266 ESP-01 s adapterom pomoću kojeg je bilo omogućeno bežično spajanje na mrežu. Za programski dio rada korištena je Arduino integrirana razvojna okolina da bi se učitale vrijednosti sa senzora i poslale na ThingSpeak kanal, Android Studio integrirano razvojno okruženje da bi se napravila aplikacija za dohvaćanje podataka s ThingSpeak kanala i ThingSpeak platforma. Aplikacija je jednostavno napravljena u svrhu da korisniku daje samo potrebne informacije i da je lagana za korištenje.

Ispitivanje rada sustava pokazalo je da mobilna aplikacija točno prikazuje podatke dohvaćene s ThingSpeak kanala i da su vrijednosti temperature i vlažnosti zraka točno prikazane u određenim bojama na temelju dohvaćenih vrijednosti. Predviđanje rasta i pada temperature i vlažnosti zraka radi ispravno, ali kada bi se sustav koristio u nekoj poslužiteljskoj prostoriji ili podatkovnom središtu, bilo bi potrebno predviđanje provoditi na temelju više vrijednosti kako bi rezultati bili što točniji i pouzdaniji. Postoji više načina poboljšanja ovog rada. Moguće je dodati više senzora, npr. senzor za dim ili pokret i postaviti više ovakvih sklopovskih rješenja u poslužiteljsku prostoriju ili podatkovno središte i pratiti vrijednosti iz različitih dijelova prostorije da bi se dobio uvid gdje se prostorija najviše zagrijava. Također, moguće je nadograditi sustav da se preko aplikacije može upravljati klima uređajem ili da se klima uređaj samostalno pokreće kada je potrebno.

LITERATURA

[1] P. Narkhede, B. Kiratkar, B. Suryawanshi, Physical Conditions Monitoring in Server Rooms Internet of Things, International Journal of Electrical and Electronics Research, br.3, str.237 – 239, listopad – prosinac 2015.

[2] N.M.S. Hassan, M.M.K. Khan, M.G.Rasul, Temperature monitoring and CFD Analysis of Data Centre, 5th BSME International Conference on Thermal Engineering, br.56, str.551 – 559, 2013.

[3] ASHRAE Datacom Series, Thermal Guidelines for Data Processing Environments, ASHRAE, Atlanta, 2015.

[4] F. Hu, Cyber-Physical Systems: Integrated Computing and Engineering Design, CRC Press, Boca Raton, FL, SAD, 2014.

[5] E. A. Lee, S. A. Seshia, Introduction to Embedded Systems - A Cyber-Physical Systems Approach, Second Edition, MIT Press, 2017.

[6] A. Seo, J. Jeong, Y. Kim, Cyber Physical Systems for User Reliability Measurements in a Sharing Economy Environment, Sensors, br. 17, str.1 – 16, kolovoz 2017.

[7] Y. Guo, X. Hu, B. Hu, J. Cheng, M. Zhou, R. Y. K. Kwok, Mobile Cyber-Physical Systems: Current Challenges and Future Networking Applications, Security and Privacy for Vehicular Networks, br. 6, str.12360 – 12368, prosinac 2017.

[8] S. Choochaisri, V. Niennattrakul, S. Jenjaturong, C. Intanagonwiwat, C. A. Ratanamahatana, SENVM: Server Environment Monitoring and Controlling System for a Small Data Center Using Wireless Sensor Network, The 1st International Computer Science and Engineering Conference, str.23 – 28, Bangkok, Tajland, 2010.

[9] Z. Afroz, GM Shafiullah, T. Urmee, G. Higgins, Prediction of Indoor Temperature in an Institutional Building, 9th International Conference on Applied Energy, sv. 142, str. 1860 – 1866, Cardiff, UK, 2017.

[10] T. Pan, Y. Zhu, Designing Embedded Systems with Arduino: A Fundamental Technology for Makers, Springer Nature, Singapore, 2018.

[11] ARDUINO UNO REV3, Arduino, 2018, dostupno na: https://store.arduino.cc/arduino-uno-rev3 (29.8.2018.)

[12] I. Grokhotkov, ESP8266WiFi library, 2017, dostupno na: https://arduino-esp8266.readthedocs.io/en/latest/esp8266wifi/readme.html#esp8266wifi-library (29.8.2018.)

[13] Meet Android Studio, 2018, dostupno na: https://developer.android.com/studio/intro/

[14] Learn More About ThingSpeak, The MathWorks, Inc., 2018, dostupno na: https://thingspeak.com/pages/learn_more

[15] AsyncTask, 2018, dostupno na: https://developer.android.com/reference/android/os/AsyncTask

[16] M. Schwartz, S. Buttigieg, Arduino Android Blueprints, Packt Publishing Ltd., Birmingham, UK, 2014.

[17] B. Cruz Zapata, Android Studio Essentials, Packt Publishing Ltd., Birmingham, UK, 2015.

SAŽETAK

Učinkovit rashladni sustav poslužiteljskih prostorija i podatkovnih središta velik je izazov. Iako je najbitnije održavati vrijednost temperature u preporučenim granicama, vlažnost isto tako može oštetiti opremu ako prijeđe dozvoljene granice. Zbog toga je bitno vršiti nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta. Cilj rada je bio napraviti sklopovsko i programsko rješenje za nadzor klimatskih uvjeta poslužiteljskih prostorija i podatkovnih središta. Rješenje problema učinkovitog hlađenja poslužiteljskih prostorija i podatkovnih središta temelji se na bežičnim senzorskim mrežama bez dijela za kontrolu klima uređaja. Za sklopovski dio rješenja korišten je upravljač Arduino Uno R3 sa senzorom za temperaturu i vlagu DHT11 i bežičnim mrežnim modulom ESP8266 ESP-01 s adapterom. Ostvareno sklopovsko i programsko rješenje prikuplja podatke i šalje ih na ThingSpeak kanal. Mobilna aplikacija, razvijena u Android Studio integriranoj razvojnoj okolini, na zahtjev korisnika dohvaća podatke i prikazuje ih u aplikaciji. Razvijeni sustav je ispitan i utvrđen je njegov ispravan rad.

Ključne riječi: Android mobilna aplikacija, bežične senzorske mreže, kibernetsko-fizikalni sustavi, podatkovna središta, ugradbeni sustav, upravljanje klimom.

ABSTRACT

Monitoring the climatic conditions of the server room via mobile application

Efficient cooling system of server rooms and data centers is a great challenge. Although it is most important to maintain the value of the temperature within the recommended limits, humidity can also damage the equipment if it exceeds the permissible limit. It is therefore essential to monitor the climatic conditions of server rooms and data centers. The aim of the paper was to create a circuit and software solution for monitoring the climatic conditions of server rooms and data centers. The solution to the problem of efficient cooling of server rooms and data centers is based on wireless sensor networks without part for controling air conditioner. The Arduino Uno R3 controller with temperature and humidity sensor DHT11 and Wi-Fi ESP-8266 ESP-01 with adapter is used for the hardware part of the solution. The completed hardware and software solution collects data and sends them to the ThingSpeak channel. The mobile app, developed in the Android Studio integrated development environment, retrieves the data and submits them to the application at the request of the user. The developed system has been tested and its proper operation has been established.

Key words: Android mobile application, wireless sensor networks, cyber-physical systems, data centers, embedded system, climate control.

ŽIVOTOPIS

Ivana Paranus rođena je 22.11.1991. god. u Mostaru (BiH). Osnovnu školu završila u Širokom Brijegu 2006., a iste te godine upisuje Gimnaziju fra Dominika Mandića u Širokom Brijegu. Srednju školu završava 2010., te iste te godine upisuje preddiplomski studij informatike na Fakultetu prirodoslovno-matematičkih i odgojnih znanosti u Mostaru. 2013. godine završava preddiplomski studij i upisuje diplomski studij informatike. 2014. godine se ispisuje s Fakulteta prirodoslovno-matematičkih i odgojnih znanosti u Mostaru i upisuje diplomski studij računarstva, smjer procesno računarstvo, na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

PRILOZI (na CD-u)

PRILOG 1. Nadzor klimatskih uvjeta poslužiteljske prostorije putem mobilne aplikacije.docx

PRILOG 2. Nadzor klimatskih uvjeta poslužiteljske prostorije putem mobilne aplikacije.pdf

PRILOG 3. Programski kod android mobilne aplikacije za nadzor klimatskih uvjeta poslužiteljske prostorije

PRILOG 4. Arduino programski kod za nadzor klimatskih uvjeta poslužiteljske prostorije putem mobilne aplikacije