Pametni sustav upravljanja ogrijjevnim sustavom temeljen na ESP IoT

Štrasser, Fran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni preddiplomski studij računarstva

PAMETNI SUSTAV UPRAVLJANJA OGRJEVNIM SUSTAVOM TEMELJEN NA ESP IOT

Završni rad

Fran Štrasser

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 14.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Fran Štrasser
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3993, 26.07.2016.
OIB studenta:	67274883786
Mentor:	Izv.prof.dr.sc. Tomislav Keser
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Pametni sustav upravljanja ogrijjevnim sustavom temeljen na ESP IoT
Znanstvena grana rada:	**Procesno računarstvo (zn. polje računarstvo)**
Predložena ocjena završnog 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: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	14.09.2021.
Datum potvrde ocjene Odbora:	22.09.2021.

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, 25.09.2021.

Ime i prezime studenta:	Fran Štrasser
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R3993, 26.07.2016.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Pametni sustav upravljanja ogrijjevnim sustavom temeljen na ESP IoT**

izrađen pod vodstvom mentora Izv.prof.dr.sc. Tomislav Keser

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

Čovjek teži olakšanju obavljanja svakodnevnih obaveza. Svima je jedan od primarnih ciljeva uštedjeti vrijeme koje gubimo na svakodnevne poslove, a jedan od načina je povezivanje kućanskih uređaja na Internet te njihovo upravljanje pomoću pametnih telefona. Bilo da se radi o sustavu navodnjavanja biljaka, otvaranju prozora, otvaranju ulaznih ili garažnih vrata ili, u ovom slučaju, upravljanju ogrjevnim sustavom. Kao rješenje za dane probleme postavlja se udaljeno upravljanje takvim sustavima pomoću platforme Internet stvari (engl. *Internet of things*, IoT) . IoT se odnosi na milijarde fizičkih uređaja po cijelom svijetu koji su spojeni na internet u cilju prikupljanja i razmjene podataka te udaljenog upravljanja. Svaki fizički objekt se može pretvoriti u IoT uređaj ukoliko se može spojiti na internet. Pojam udaljenog upravljanja odnosi se na upravljanje nekog tijela ili objekta na daljinu, gdje udaljenost između upravljačkog sustava i objekta kojim se upravlja nije strogo definirana.

Postupnim razvitkom načina grijanja kroz povijest dolazimo do uporabe centralnog grijanja u kućanstvima. Centralno grijanje podrazumijeva proizvodnju topline centralno u zajedničkom uređaju ili postrojenju te se pomoću nekog prijenosnika topline dovodi u određene prostorije. Kao prijenosnik topline najčešće se koristi voda, vodena para i zrak.

U ovome radu predstavljena je primjena ESP32 komunikacijskog modula u udaljenom upravljanju ogrjevnim sustavom. LED dioda je reprezentacija statusa ogrjevnog sustava. Ogrjevnim sustavom će upravljati mikrokontroler koji će korisniku prikazati trenutnu vrijednost mjerenih veličina. Budući da će se za ostvarivanje sustava koristiti Croduino, za programiranje njega će se koristiti platforma otvorenog koda, Arduino integrirano programsko okruženje (engl. *Arduino Integrated Development Environment*, Arduino IDE). Arduino IDE podržava jezike C i C++ uz korištenje posebnih pravila strukture koda. Takav dizajn ima prednost dostupnosti, niske cijene, reprogramiranja, lakog dodavanja novih funkcija, te jednostavnosti u programiranju i održavanju. Glavni nedostaci su nepouzdanost i održavanje od strane krajnjih korisnika koji nisu inženjeri.

1.1. Zadatak i struktura rada

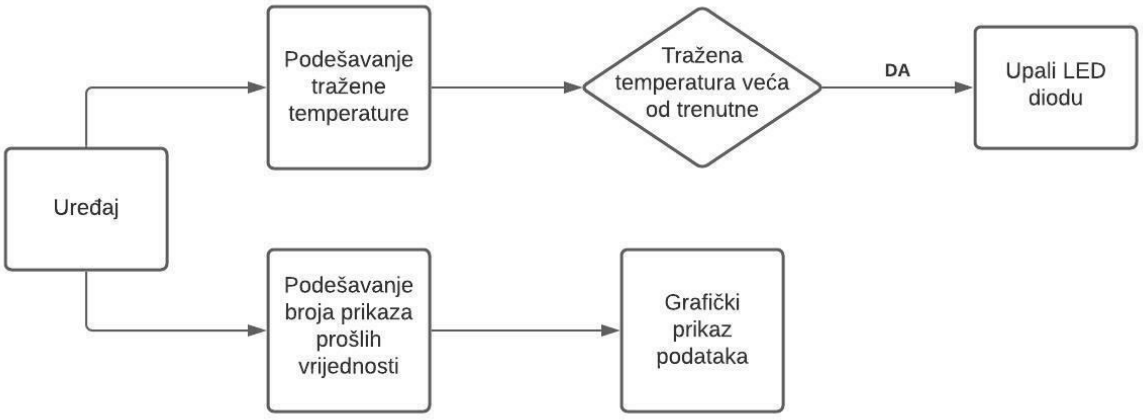
Zadatak završnog rada je izraditi sustav udaljenog upravljanja ogrjevnim sustavom koji će pomoću senzora prikupljati trenutne vrijednosti traženih parametara te ih obraditi i prikazati korisniku na mobilnom uređaju. Za realizaciju ovog sustava koristi se ESP32 komunikacijski modul i Croduino mikrokontroler. Korisniku će se jedino prepustiti odabir željene unutarnje temperature i odabir broja prošlih vrijednosti koje će se prikazivati na grafu. Primijenit će se tehnologija IoT. Programiranje mikrokontrolera te uspostava servera odrađena je u Arduino IDE-u, a izgled web servera napravljen je korištenjem HTML-a (engl. *HyperText Markup Language*) i CSS-a (engl. *Cascading Style Sheets*)

2. TEORIJSKI OSVRT NA PAMETNI SUSTAV UPRAVLJANJA OGRJEVNIM SUSTAVOM

Kroz povijest vidimo napredak u načinu na koji je čovjek održavao svoju okolinu toplom. Otkrićem vatre dolazi do znatnog napretka u grijanju. Vatru je čovjek palio u hladnim vremenima kako bi se grijao. Najstariji oblik grijanja jest grijanje drvima čiji je prvi oblik otvoreno ložište. Ono podrazumjeva postavljanje ložišta u sredinu prostorije koju treba zagrijati. Nedostatak ovakvog načina grijanja je zadimljenost prostorije u kojoj se ložište nalazi. Nakon toga dolazi do razvitka kamina koji je preko dimnjaka osiguravao da dim izlazi.

Daljnim napretkom dolazimo do otkrića hipokausta. Prema [1] hipokaust je antički uređaj za grijanje koji radi na način da se ispod prostorija, koje su se trebale ugrijati, nalazio posebni podrum sa stupovima. Za izradu stupova se koristila opeka ili glina. Ložište je bilo pored zgrade, a kao gorivo koristila se drvo ili drveni ugljen. Ovaj oblik grijanja radi na principu da su vrući dimni plinovi, dovedeni iz ložišta u podrum, zagrijavali stupove i strop podruma. Kad su strop i stupovi bili dovoljno zagrijani gasilo se ložište te se zbog toga ovaj sustav smatra prvim sustavom centralnog grijanja. Centralno grijanje je vrsta grijanja u kojem se toplina proizvodi centralno u uređaju te se u prostorije dovodi nekim prijenosnikom topline koji su najčešće topla voda, para ili zrak.

2.1. Sustav upravljanja ogrjevnim sustavom



Slika 2.1. *Blok dijagram upravljanja sustavom*

Ovaj rad je zamišljen kao upravljanje centralnom jedinicom u sustavu centralnog grijanja u kojem će prijenosnik topline biti voda. Ideja ovog sustava upravljanja je da njegovom korisniku omogući podešavanje željene temperature prostorije i podešavanje broja prikaza prošlih vrijednosti koje će mu biti prikazane putem grafa. (Slika 2.1.) Na temelju željene temperature i trenutne vrijednosti temperature u prostoriji, koju dobivamo pomoću senzora, ogrjevni sustav se pali i gasi po potrebi.

2.1.1. Mjerene procesne veličine

Procesne veličine koje će se mjeriti i prikazivati korisniku su: temperatura prostorije, vlaga u prostoriji, temperatura vode i protok vode. Za mjerenje željenih procesnih veličina potrebni su nam senzori, pomoću čijih ćemo očitanja dobiti trenutnu vrijednost mjerenih parametara.

2.1.2. Teorija prikaza podataka

Nakon što centralna jedinica izmjeri trenutnu vrijednost parametara, njih će biti potrebno prikazati korisniku. Za taj dio će biti zadužen web server i platforma Thingspeak. Putem web servera korisnik će imati mogućnost pregleda trenutnih vrijednosti mjerenih veličina kao i uvid u graf koji će prikazivati promjenu vrijednosti u unaprijed određenom proteklom vremenskom intervalu.

2.1.3. Teorija mjerenja temperature

Senzori temperature su u upotrebi već dugi niz godina. Ljudskim i tehnološkim napretkom oni postaju sve točniji i efikasniji. Prema [2] prvi uređaj za mjerenje temperature je bio termometar koji sadrži živu. Kako temperatura raste tako raste i obujam žive te se temperatura očitava sa skale.

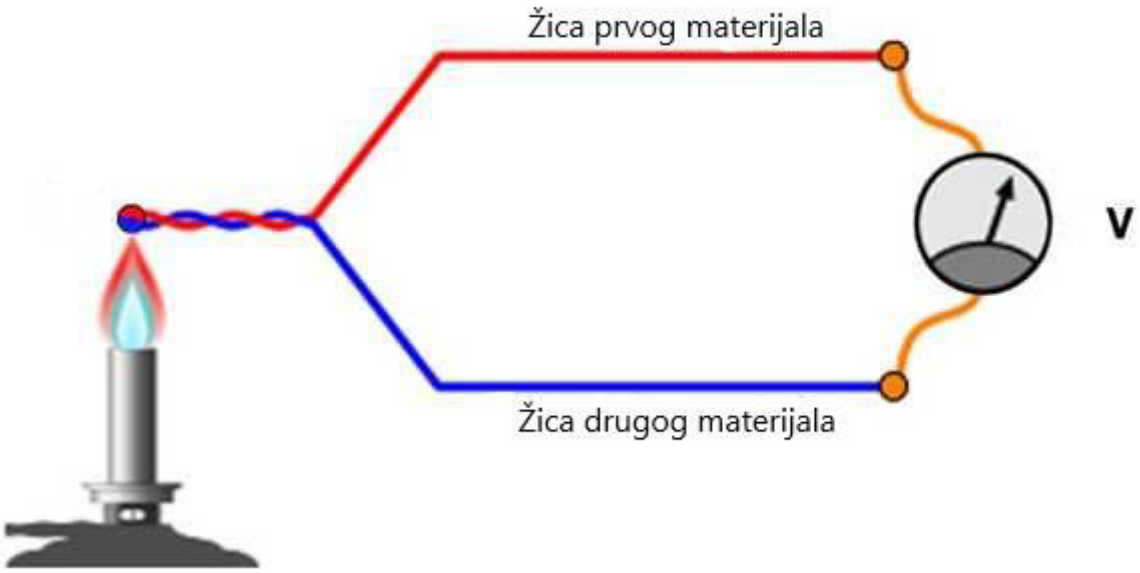
Napredak u mjerenju temperature donosi izum temperaturnih senzora. Prema [3] temperaturni senzori su sposobni mjeriti temperaturu no zahtijevaju dodatan električni sklop kako bi omogućili korisniku uvid u vrijednosti mjerenja. Generalno, sve vrste temperaturnih senzora su elektronički uređaji koji se oslanjaju na otpor, napon i struju koji su potom interpretirani uređajem koji upravlja senzorom.

2.1.4. Metode mjerenja temperature zraka

Postoje tri vrste senzora za mjerenje temperature koji su najčešće korišteni u modernoj elektronici.

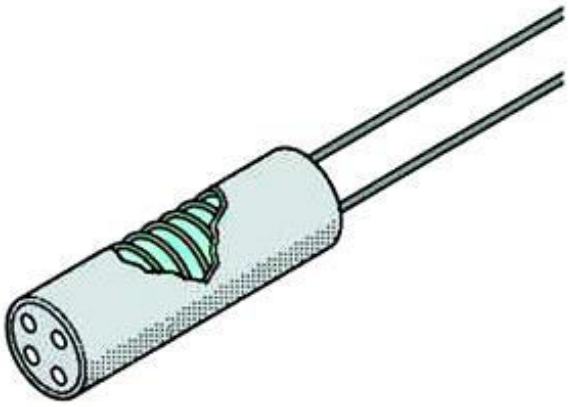
To su:

- Termoparovi
- Senzori otpornikom
- Termistori



Slika 2.2. *Prikaz rada termoparova prema izvoru [4]*

Termoparovi su najčešće korišten tip temperaturnih senzora. Prema slici 2.2. njih čine dvije žice različitih materijala koje su spojene na jednom zajedničkom kraju. Princip rada temelji se na termoelektričnom efektu. Razlika u temperaturi između žica različitih materijala stvara napon. Mjerenjem tog napona izračunava se vrijednost temperature. [4]



Slika 2.3. *Prikaz strukture senzora otpornikom prema izvoru [5]*

Senzori otpornikom temeljeni su na principu da ako se temperatura mijenja, mijenja se i otpor metala. Ovoj vrsti senzora dulje treba da reagira na promjenu temperature od termoparova. Senzori otpornikom su često korišteni u slučajevima kada je najbitnija preciznost mjerenja.

Slika 2.4. *Različiti oblici i veličine termistora prema izvoru [6]*

Prema [6] termistori su slični senzorima otpornikom u smislu načina rada. Promjena u temperaturi uzrokuje promjenu u otporu. Razlika među njima je da su termistori jeftiniji, ali i neprecizniji od senzora otpornikom. Dijele se na dvije vrste: NTC (engl. *Negative Temperature Coefficient*) i PTC (engl. *Positive Temperature Coefficient*). NTC termistorima se otpor smanjuje kako temperatura raste te su oni ćešće korišteni. PTC termistorima se otpor povećava kako temperatura raste.

2.1.5. Metoda mjerenja temperature vode

Za mjerenje temperature vode unutar upravljačke jedinice centralnog grijanja zadužen je termostat.[7] Dizajniran je na način da održava određenu temperaturu vode uključivanjem i isključivanjem grijaćeg elementa. Ukoliko dođe do kvara na grijaćem elementu, termostat ga isključuje u nuždi.

2.1.6. Teorija mjerenja vlage

Vlaga predstavlja količinu vodene pare u zraku. Vlaga raste proporcionalno s temperaturom te senzori za mjerenje vlažnosti zraka često dolaze zajedno sa senzorima za mjerenje temperature. Prema [8] postoje 3 glavne mjere vlažnosti:

- Apsolutna vlažnost
- Relativna vlažnost
- Specifična vlažnost

Apsolutna vlažnost je mjera količine vodene pare u zraku bez obzira na temperaturu zraka. Što je više vodene pare u zraku, to je veća relativna vlažnost.

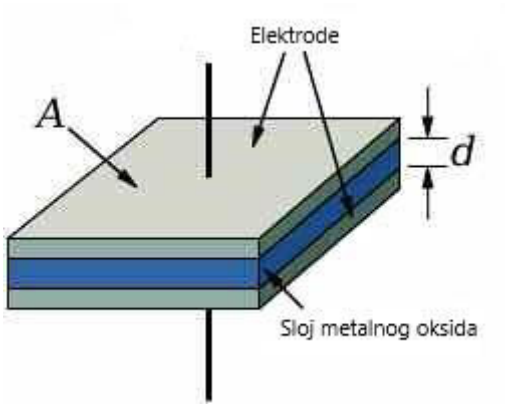
Relativna vlažnost uspoređuje izmjerenu količinu vodene pare u zraku sa iznosom vlage koju zrak može držati pri toj temperaturi. Izražava se u postotcima.

Specifična vlažnost predstavlja omjer težine vodene pare sadržane u masi jedinice zraka. Specifična i apsolutna vlažnost su slične po konceptu.

2.1.7. Metode mjerenja vlage

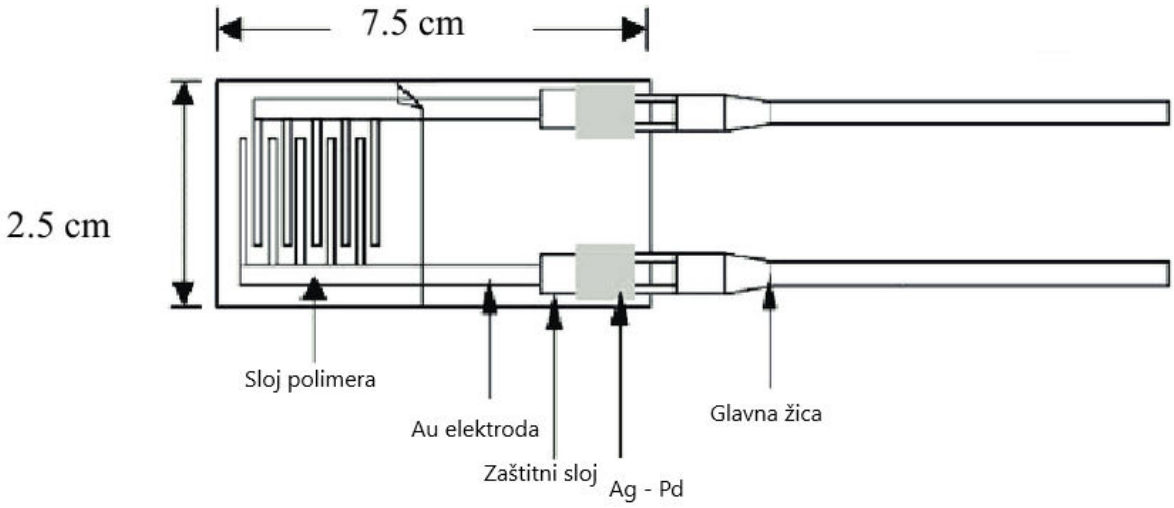
Postoje 3 glavna tipa senzora vlažnosti zraka:

- Kapacitivni
- Otporni
- Termalni



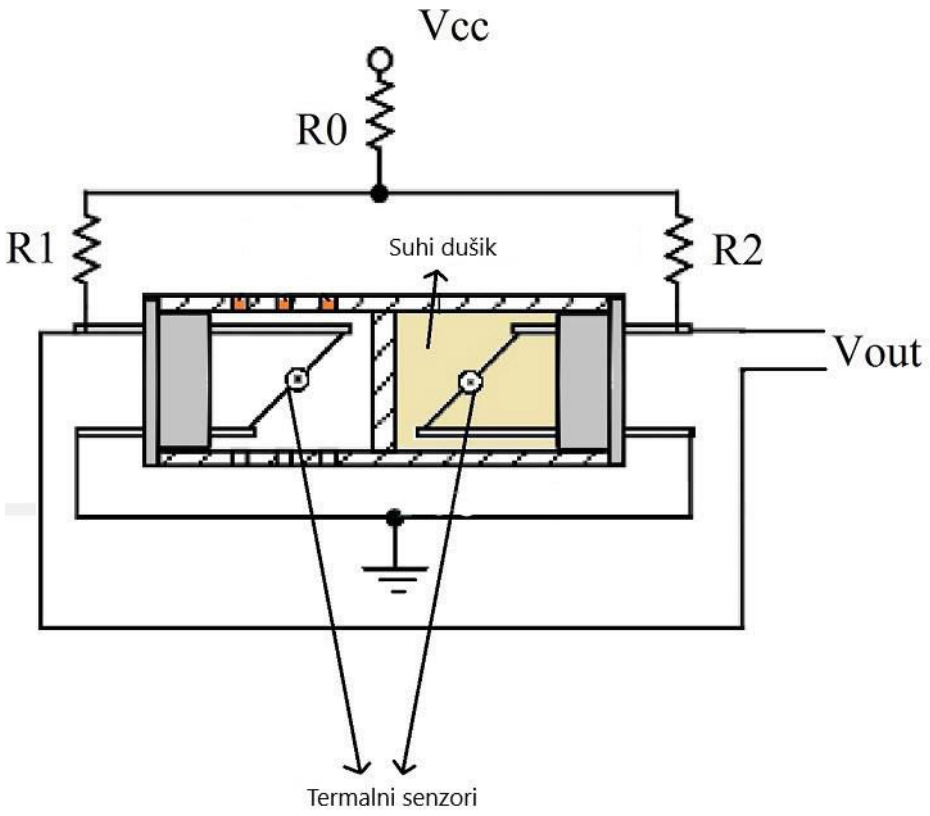
Slika 2.5. *Komponente kapacitivnog senzora vlažnosti prema izvoru [9]*

Kapacitivni senzor vlažnosti mjeri relativnu vlažnost stavljanjem tankog sloja metalnog oksida između dvije elektrode. (Slika 2.5.) Električni kapacitet metalnog oksida se mijenja s obzirom na relativnu vlažnost okolnog zraka. [9] Ovakav tip senzora je linearan i može izmjeriti relativnu vlažnost u intervalu od 0% do 100%.



Slika 2.6. *Shematski prikaz otpornog senzora vlažnosti prema izvoru [10]*

Otporni senzori vlažnosti koriste ione u soli da bi izmjerili električnu impedanciju atoma. (Slika 2.6.) Kako se vlažnost mijenja, tako se i otpor elektroda mijenja. [11]



Slika 2.7. *Shematski prikaz termalnog senzora vlažnosti prema izvoru [11]*

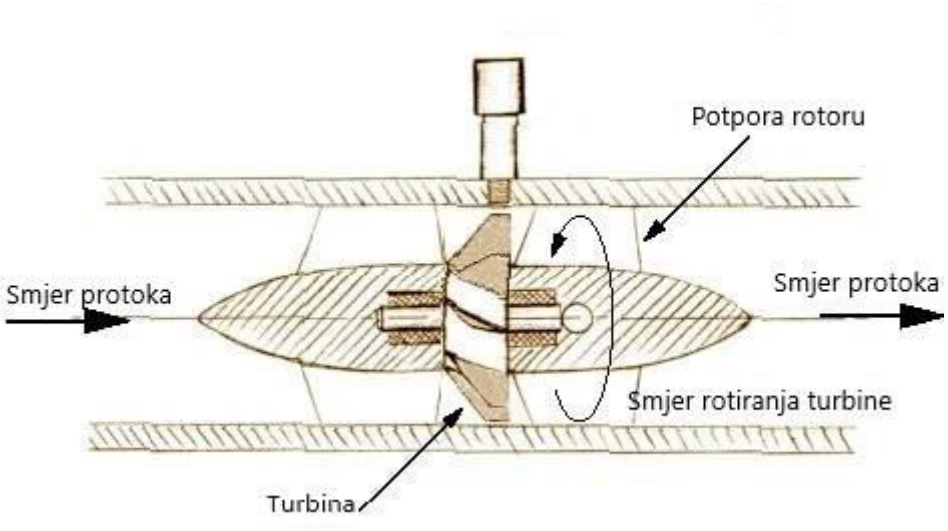
Termalni senzori vlažnosti rade na principu da dva termalna senzora provode struju u odnosu na vlažnost zraka koji ih okružuje. (Slika 2.7) Jedan od senzora je umotan u suhi dušik dok drugi mjeri vlažnost okolnog zraka. Razlika među njima je vrijednost vlažnosti zraka. [11]

2.1.8. Teorija mjerenja protoka fluida

Protok fluida predstavlja volumni protok fluida kroz neku točku u jedinici vremena. Protok fluida se mjeri pomoću senzora za protok fluida.[12] Postoje četiri tipa senzora protoka fluida:

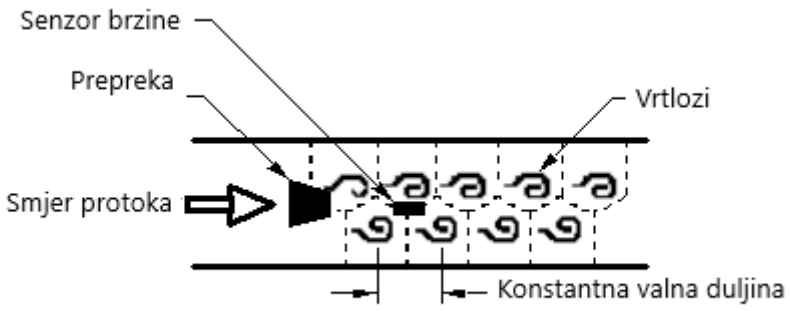
- Mehanički senzor protoka fluida
- Vrtložni senzor protoka fluida
- Ultrazvučni senzor protoka fluida

• Magnetski senzor protoka fluida



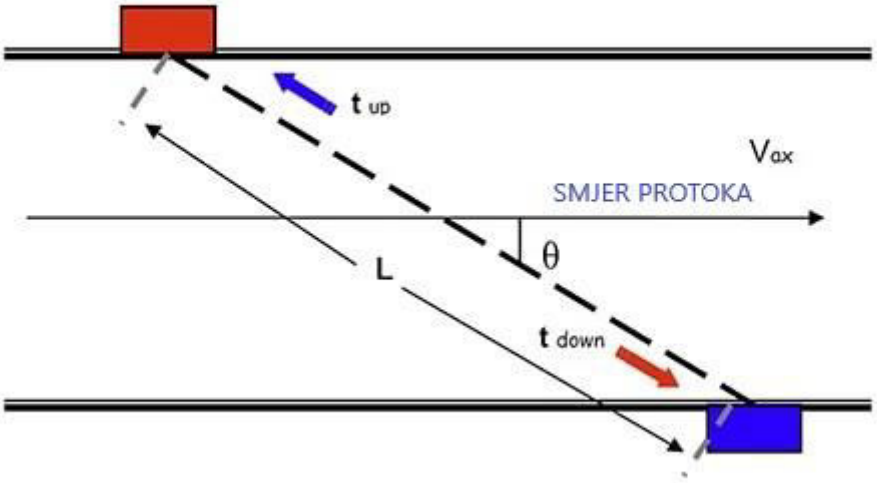
Slika 2.8. *Prikaz rada mehaničkog tipa senzora protoka fluida prema izvoru [12]*

Mehanički tip je najčešći i najekonomičniji senzor protoka fluida. Prema slici 2.7. mehanički tip radi tako što fluid koji prolazi kroz cijev pokreće turbinu te se na temelju brzine rotacije turbine mjeri brzina protoka fluida.



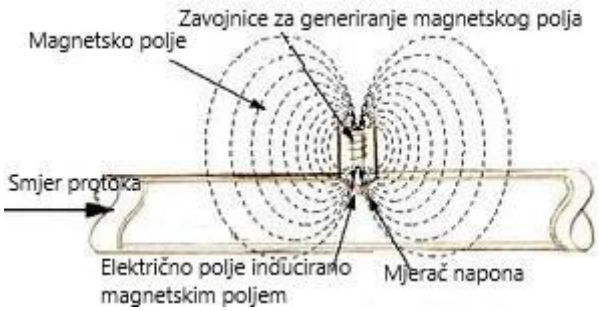
Slika 2.9. *Prikaz rada vrtložnog tipa senzora protoka fluida prema izvoru [13]*

Prema slici 2.8. vrtložni tip radi na principu da fluid u protoku naleti na prepreka stavljenu ispred senzora koji mjeri vrtloge koji nastanu nailaskom fluida na prepreku. Na temelju tih mjerenja se računa količine protoka fluida.



Slika 2.10. *Prikaz rada ultrazvučnog tipa senzora protoka fluida prema izvoru [12]*

Ultrazvučni tip mjeri brzinu protoka fluida koristeći ultrazvukove. Prema slici 2.9. ultrazvuk je odaslan u smjeru protoka te je još jedan ultrazvuk odaslan suprotno smjeru protjecanja fluida. Vrijeme potrebno ultrazvuku koji putuje uzvodno je uspoređeno s vremenom potrebnim ultrazvuku koji putuje nizvodno. Na temelju te usporedbe izračunava se brzina protoka fluida.



Slika 2.11. *Prikaz rada magnetskog tipa senzora protoka fluida prema izvoru [12]*

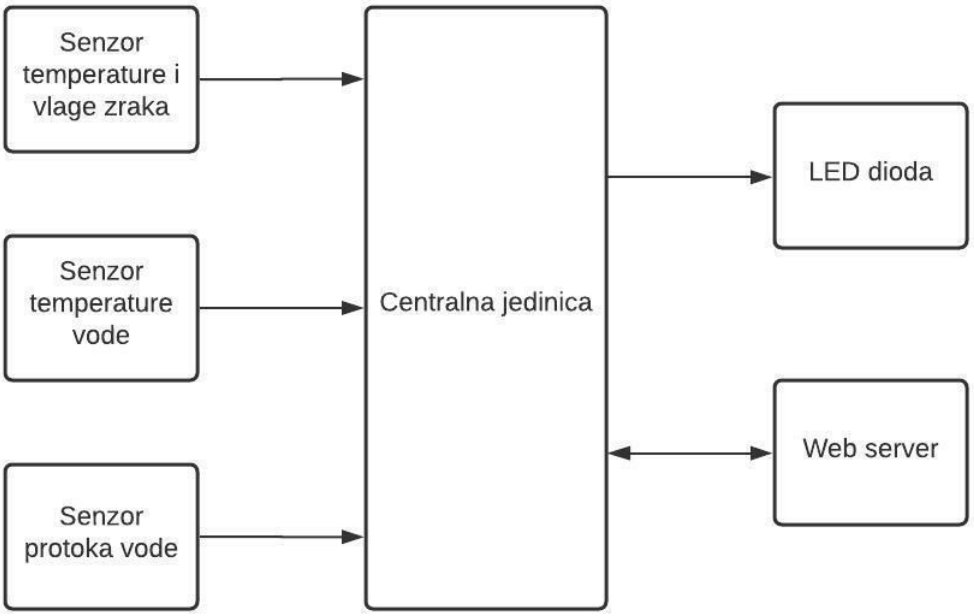
Prema slici 2.10. Magnetski tip mjeri brzinu protjecanja fluida koja prolazi kroz cijev korištenjem magnetskog polja. Ovaj tip zasnovan je na Faraday-evom zakonu elektromagnetske indukcije prema kojem fluid stvara napon prolaskom kroz magnetsko polje.

2.1.9. Metode mjerenja protoka fluida

Senzor protoka fluida korišten u ovom radu je mehaničkog tipa. Kada voda prolazi kroz senzor, ona pokreće rotor te Hallov senzor očitava brzinu rotora i generira impulse.

2.2. Prijedlog sklopovskog rješenja

Dizajnirati će se sustav senzora, LED diode i centralne jedinice koja će ostvariti komunikaciju s uređajem na IoT principu. Sustav reprezentira ogrjevni sustav te upravljanje istime.



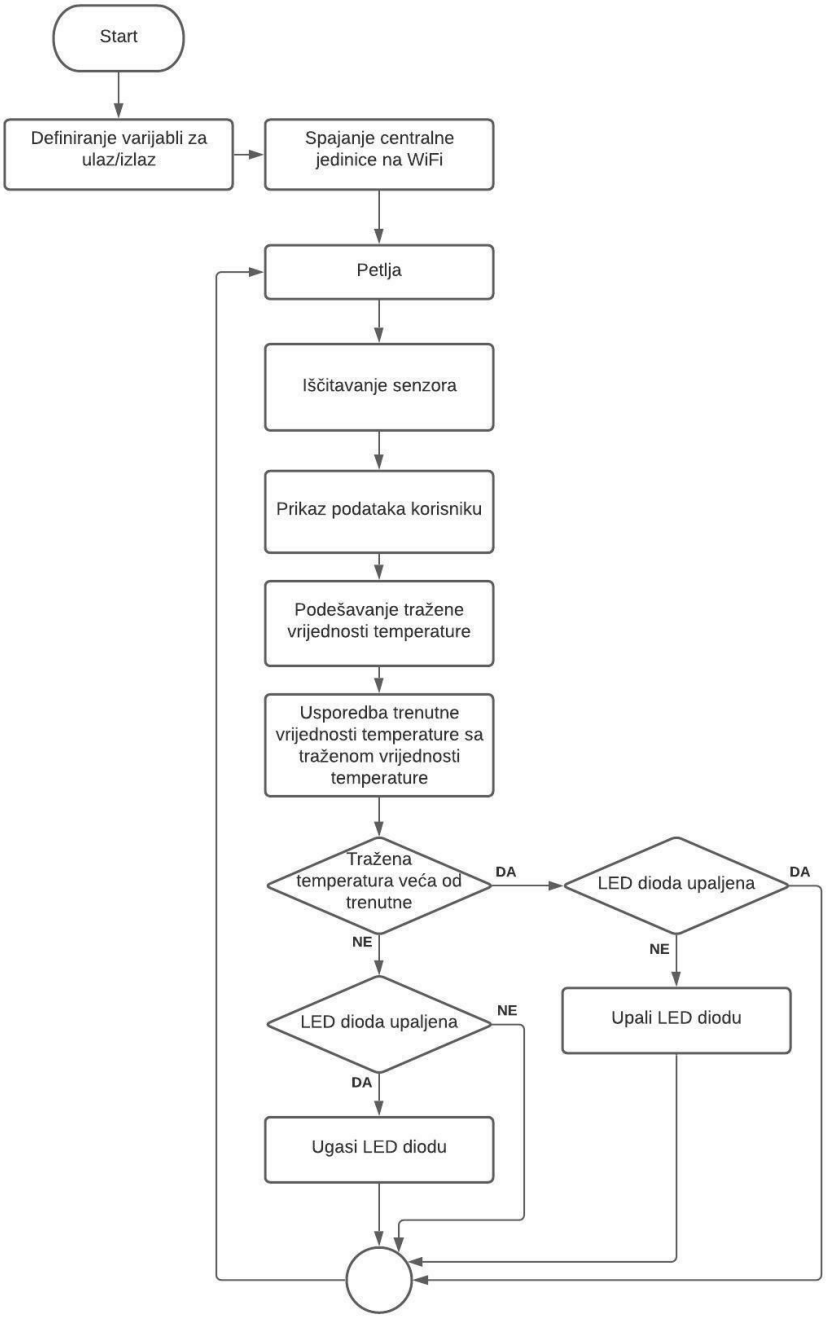
Slika 2.12. *Blok dijagram strukture*

Prema slici 2.12. u centralnu jedinicu potrebno je spojiti 3 senzora. Senzor za mjerenje temperature i vlage zraka, senzor za mjerenje temperature vode i senzor protoka vode. Centralna jedinica treba upaliti i održati ogrjevni sustav upaljenim ako je željena temperatura niža od trenutne. Korisnik upravlja željenom temperaturom te na taj način upravlja i

ogrjevnim sustavom. Upravljanje se odvija putem web server-a čije će datoteke biti spremljene na flash memoriju centralne jedinice. Izbor arduino razvojne ploče ovisi o broju ulaza i izlaza potrebnih za sve senzore i LED diodu te mora imati ESP mikrokontroler kako bi uopće bilo moguće ga spojiti na WiFi i pokrenuti web server na njemu. Grafički prikaz prethodnih i trenutnih vrijednosti sustava korisniku će biti omogućen u stvarnom vremenu ukoliko je uređaj s kojeg korisnik pristupa podacima i upravlja ogrjevnim sustavom spojen na istu mrežu kao i centralna jedinica.

2.3. Prijedlog algoritamskog rješenja

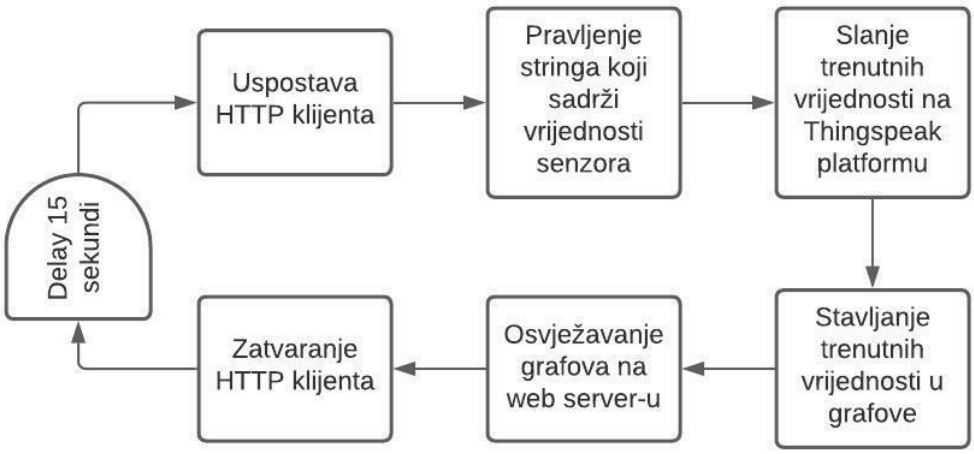
Algoritamsko rješenje je zamišljeno na principu programiranja neovisnog rada 3 senzora.



Slika 2.13. *Blok dijagram algoritamskog rješenja*

Svaki senzor mora izmjeriti trenutni parametar za koji je zadužen i predati tu vrijednost centralnoj jedinici na obradu. Centralna jedinica je zadužena za prosljeđivanje podataka putem IoT platforme na uvid korisniku. Ogrjevni sustav se pali ukoliko je trenutna temperatura manja od tražene. Potrebno je isprogramirati web server putem kojeg će se korisniku dati trenutne i prethodne vrijednosti mjerenih veličina na pregled i omogućiti korisniku podešavanje željene temperature i broja vrijednosti koje će se prikazivati na grafovima. (Slika 2.13.)

2.4. Prijedlog upravljačkog i komunikacijskog sučelja



Slika 2.14. *Blok dijagram algoritma komunikacije s Thingspeak platformom*

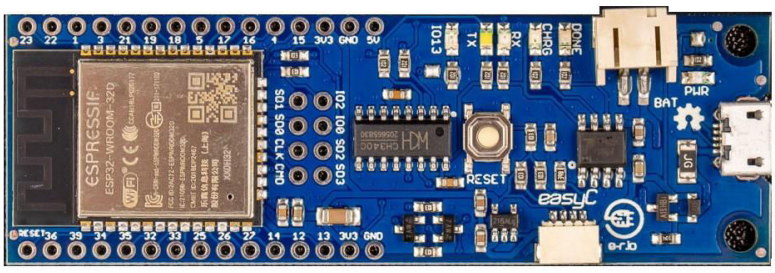
Korisnik upravlja ogrjevnim sustavom putem web server-a. Senzore je potrebno postaviti i upaliti centralnu jedinicu. Za optimalan rad centralne jedinice potrebno joj je pružiti stabilnu internetsku vezu. Web server je ostvaren uporabom HTML prezentacijskog jezika i CSS programskog jezika. Korisniku je omogućen pristup web server-u putem IP adrese ako je spojen na istoj mreži na koju je spojena i centralna jedinica.

3. REALIZACIJA SUSTAVA

3.1. Korišteni alati i razvojna okruženja

3.1.1. Croduino NOVA32 mikrokontroler

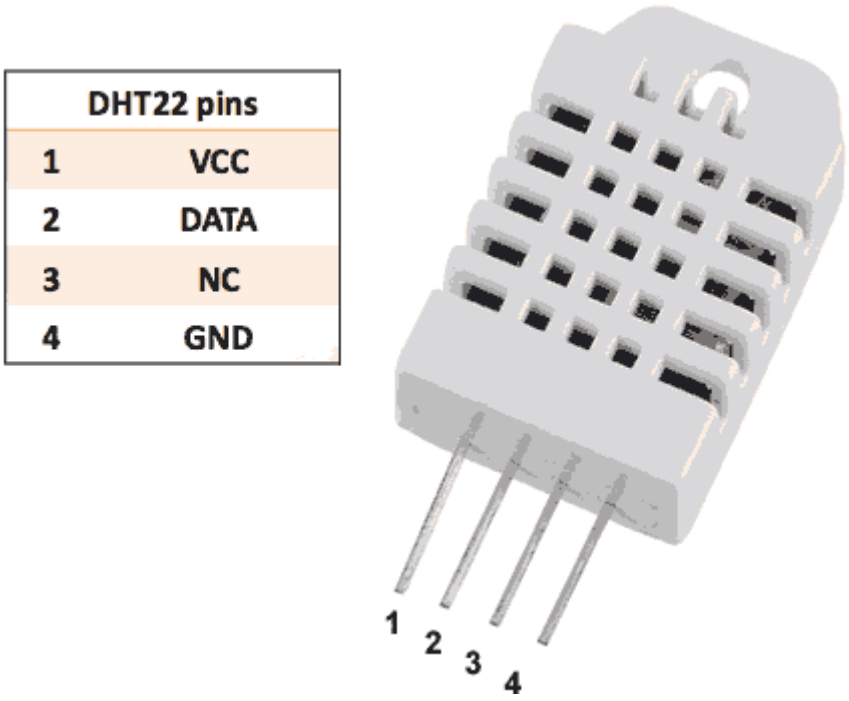
Croduino NOVA32 je mikrokontrolerska pločica koja se temelji na ESP32 mikrokontroleru. Ima mogućnosti spajanja na mrežu putem WiFi modula te mogućnost spajanja putem Bluetooth-a. Programiranje ove pločice odvija se u Arduino integriranom programskom okruženju koje je dostupno na Windows, Mac i Linux operacijskim sustavima. Za programiranje u ovom programskom okruženju koriste se C ili C++ programski jezici te se programiranje mikrokontrolera odvija direktno preko ugrađenog USB konektora. Tehničke karakteristike: **P.1.**



Slika 3.1. *Prikaz Croduino NOVA32 mikrokontrolera prema izvoru [14]*

3.1.2. DHT22 senzor temperature i vlage

DHT22 je digitalni senzor za mjerenje temperature i vlažnosti. On koristi kapacitivni senzor vlažnosti i termistor za mjerenje vlažnosti i temperature okolnog zraka. Jednostavan je za upotrebu ali zahtjeva oprez kod vremena između očitavanja vrijednosti. Svaki put kada ovaj senzor očita iznos procesnih veličina, mora proći barem dvije sekunde prije očitavanja novih vrijednosti. Prednosti nad DHT11 senzorom su preciznija i točnija očitavanja. Tehničke karakteristike: **P.2.**



Slika 3.3. *Prikaz DHT22 senzora s imenima pinova prema izvoru [15]*

Prema slici 3.3. DHT22 senzor se spaja na način da je prvi pin spojen na napon od 5V, drugi na IO pin Croduina, treći pin se ne spaja i zadnji pin se spaja na uzemljenje. Potrebno je još spojiti pull-up otpornik od 10000Ω između prvog i drugog pina kako bi osigurali da jačina signala ostane visoka.

```
#include "DHT.h"//Biblioteka za rad s DHT senzorima

DHT dht(3, DHT22);//Inicijalizacija DHT senzora

float temperature;//Varijabla za spremanje vrijednosti temperature zraka
float humidity;//Varijabla za spremanje vrijednosti vlažnosti zraka

void setup() {
  Serial.begin(115200);
  dht.begin();
}

void loop() {
  temperature = dht.readTemperature();
  humidity = dht.readHumidity();
  Serial.print("Temperatura: ");
  Serial.print(temperature);//Ispis temperature
  Serial.println(" degrees Celsius ");
  Serial.print("Vlaznost: ");
  Serial.print(humidity);//Ispis vlažnosti
  Serial.println("%");
  delay(2000);//Delay prije iščitavanja novih vrijednosti
}
```

Slika 3.4. *Primjer koda za DHT22 senzor*

3.1.3. DS18B20 vodootporni senzor temperature

DS18B20 je vodootporni senzor za mjerenje temperature. Zbog toga što je vodootporan pogodan je za mjerenje temperature vode. Sa DS18B20 senzorom se lako očitava temperatura pomoću biblioteka otvorenog koda namjenjenih za upotrebu sa DS18B20 senzorom. Za komunikaciju ss Arduinom koristi jednu žicu. Tehničke karakteristike: **P.3.**

Slika 3.5. *Prikaz DS18B20 senzora temperature prema izvoru [16]*

Prema slici 3.5. DS18B20 senzor se spaja crvenom žicom na 5V, crnom žicom na uzemljenje i žutom žicom na IO pin Croduina. Između crvene i žute žice se stavlja pull-up otpornik od 4750Ω.

```
#include "OneWire.h"//Uključivanje biblioteka za rad sa DS18B20 senzorom
#include "DallasTemperature.h"

OneWire oneWire(4);//Pravljenje instance OneWire za komunikaciju sa bilo kojim OneWire uređajem
DallasTemperature sensors(&oneWire);//Prosljedivanje OneWire reference DallasTemperature-u

void setup() {
  Serial.begin(115200);
  sensors.begin();//Pokretanje biblioteke
}

void loop() {
  sensors.requestTemperatures();//Zahtjev za trenutnim vrijednostima mjerenih veličina
  Serial.print(sensors.getTempCByIndex(0));//Ispis temperature vode
  Serial.println(" degrees Celsius");
  delay(1000);//Delay prije iščitavanja novih vrijednosti
}
```

Slika 3.6. *Primjer koda za DS18B20 senzor*

3.1.4. YF-S201B senzor protoka vode

Sastoji se od plastičnog ventila, rotora i hall-efekt senzora. Voda prolazi preko rotora i time mu mjenja brzinu dok hall-efekt senzor na izlazu daje odgovarajući signal. Tehničke karakteristike: **P.4.**

Slika 3.7. *Prikaz YF-S201B senzora protoka vode prema izvoru [17]*

```
volatile int impulses;//Broji impulse signala
int waterFlow;
int flowPin = 25;//Broj pina na koji je senzor spojen

void setup() {
  pinMode(flowPin, INPUT);//Postavlja pin 25 kao ulaz
  Serial.begin(115200);
  attachInterrupt(flowPin, rpm, RISING);//Postavlja interrupt na pin 25
}

void loop() {
  impulses = 0;//Postavljanje impulsa u 0 da je spreman za mjerenje
  sei();//Omogućuje interrupt
  delay (1000);
  cli();//Onemogućuje interrupt
  waterFlow = (impulses * 60 / 5.5);//Jednadžba za računanje protoka pomoću impulsa
  Serial.print(waterFlow, DEC);//Ispis količine protoka
  Serial.println(" L/h");
}

void rpm ()//Funkcija koju interrupt poziva
{
  impulses++;//Funkcija mjeri rastući brid signala Hall-effect senzora
}
```

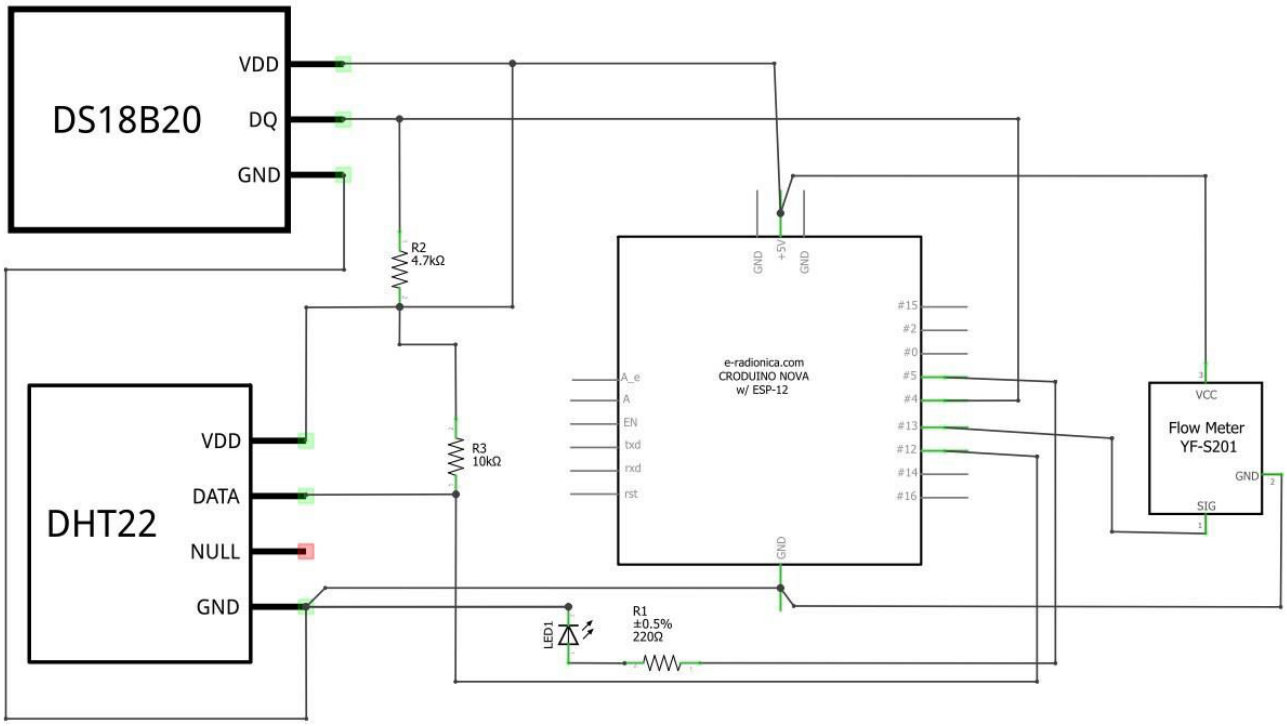
Slika 3.8. *Primjer koda za YF-S201B senzor*

3.2. Realizacija sklopovskog rješenja

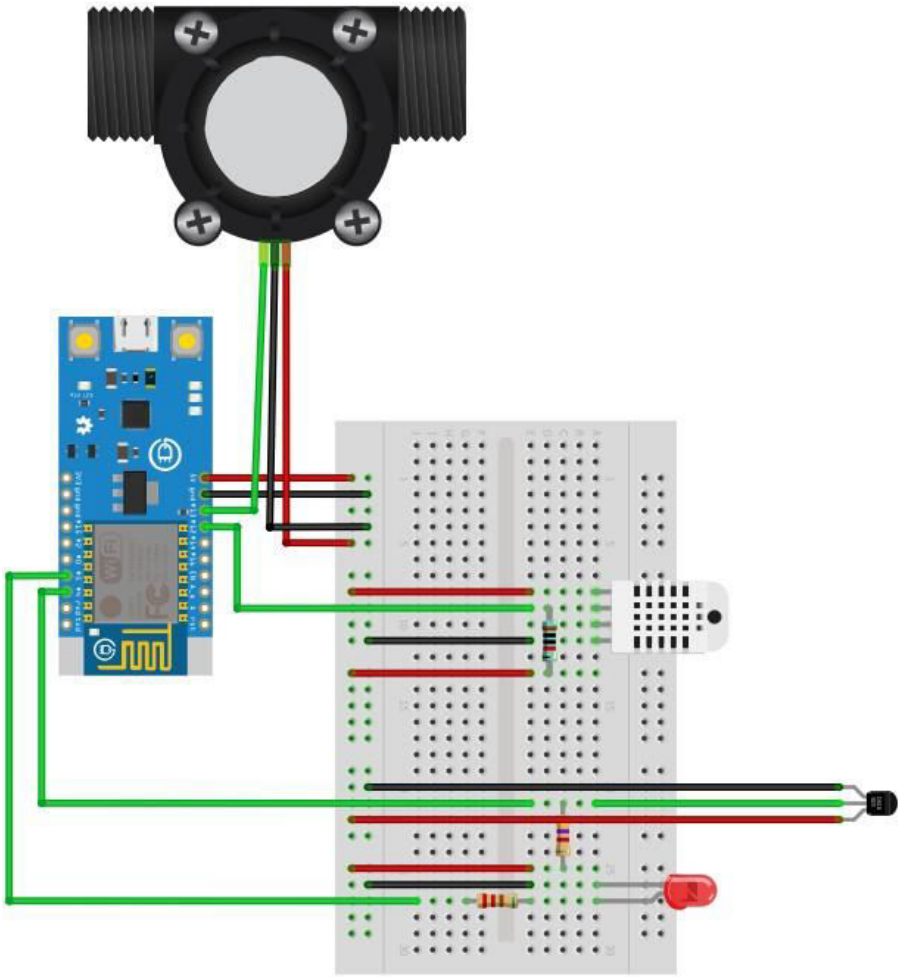
Realizacija sklopovlja ovisi o brojnim faktorima. Potrebno je odabrati odgovarajuće senzore. Na temelju istraživanja i prijedlogu mentora odabrani su senzori koji mogu izmjeriti iznose procesnih veličina. U obzir su ulazili senzori koji odgovaraju namjeni te se korišten senzor odabrao uspoređivanjem dostupnosti, cijene i preciznosti pri mjerenju željene veličine. Zatim odabrati prikladnu Arduino pločicu. Za odabir Arduino pločice u obzir su ulazile pločice sa ESP čipovima te se među njima biralo na temelju broja ulaznih i izlaznih pinova i brzine čipa. Na temelju istraživanja i osobnog iskustva odabrana je Croduino NOVA32 koja ima 28 ulazno-izlaznih pinova i ESP32 čip. ESP32 čip ima bolje performanse od ESP8266 čipa. Dvojezgreni procesor u ESP32 čipu ima taktnu frekvenciju do 240MHz dok jednojezgreni procesor u ESP8266 čipu ima taktnu frekvenciju od 80MHz.

Analizom karakteristika senzora utvrđeno je da svi senzori i LED dioda zahtijevaju napon od 5V za rad. Centralna jedinica također zahtijeva napon od 5V za rad te se ona napaja

pomoću USB konektora koji je spojen u AC-DC strujni adapter. Korišteni adapter daje izlazni napon od 5V i struju od 2A te je njegov odabir ovisio o karakteristikama centralne jedinice.



Slika 3.9. *Električna shema sustava*



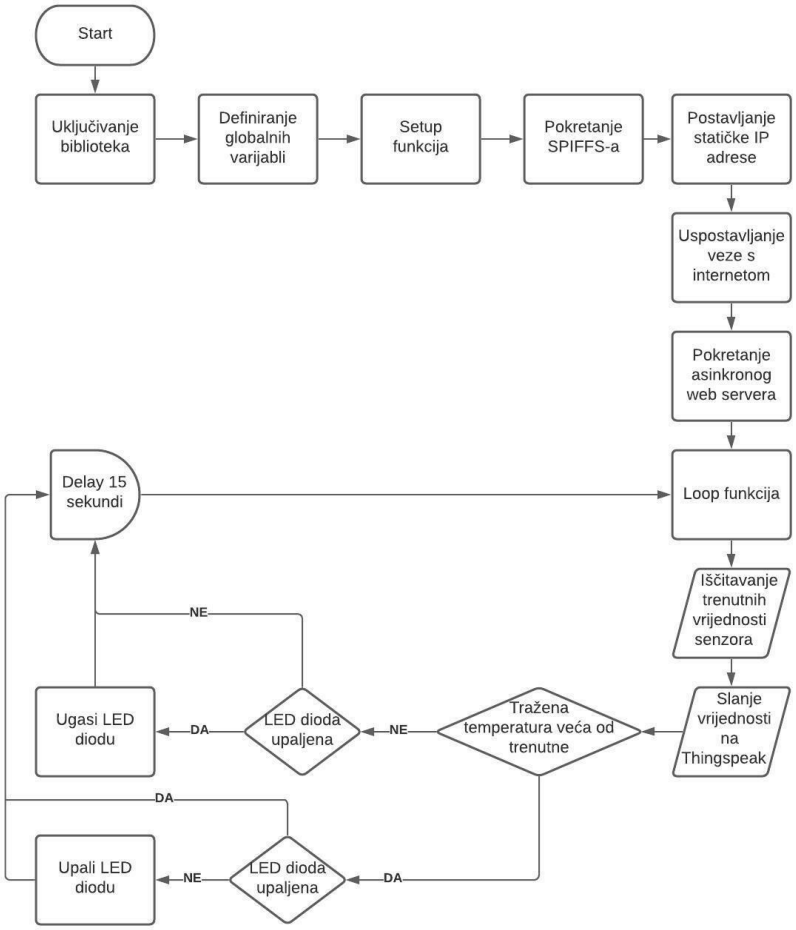
Slika 3.10. Prikaz spojenog sustava

3.3. Realizacija programskog rješenja

Nakon spajanja svih senzora i LED diode s centralnom jedinicom programiran je rad sustava. Napravljeno algoritamsko rješenje radi kako je zamišljeno i potrebno za pravilan rad, kako svakog dijela sustava zasebno, tako i sustava kao cjeline. Centralna jedinica odgovorna je mjerenje trenutnih vrijednosti traženih veličina i njihovo slanje na Thingspeak platformu te za uspostavu, održavanje asinkronog web servera sa statičkom IP adresom i obradu korisničkih zahtjeva koji se zadaju na web server-u.

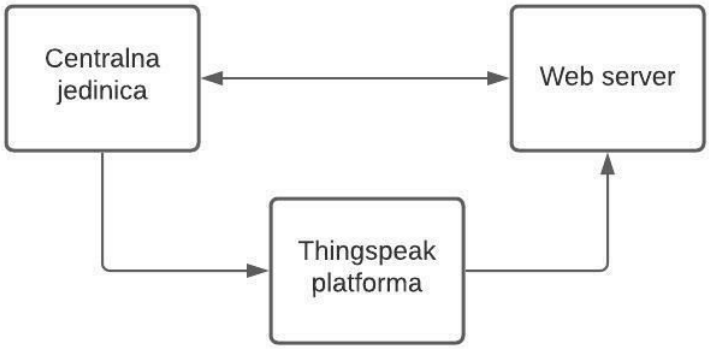
Programski kod za DS18B20 i DHT22 koristi vanjske biblioteke za lakše programiranje tih senzora dok za programiranje YF-S201B senzora i LED diodu se ne koriste vanjske biblioteke. Za spremanje HTML i CSS datoteka lokalno na uređaj koristi se SPIFFS (engl. *Serial Peripheral Interface Flash File System*). Korisnik klikom na određeni gumb na web server-u šalje različit zahtjev centralnoj jedinici na obradu. Centralna jedinica obrađuje taj zahtjev i iz svoje lokalne memorije prikazuje HTML datoteku određenu zahtjevom.

Trenutne vrijednosti mjerenih veličina se šalju Thingspeak platformi pomoću HTTP klijenta. Nakon toga slijedi delay od 15 sekundi jer na besplatnoj verziji Thingspeak-a minimalno vrijeme koje mora proći za slanje novih podataka iznosi 15 sekundi. Za vrijeme trajanja delay-a konstantno se uspoređuju tražena i trenutna temperatura te ako uvjet bude ispunjen ogrjevni sustav se pali ili gasi ovisno o tome koji je uvjet ispunjen. Za detaljan pregled programskog koda centralne jedinice pogledati prilog **P.5.**



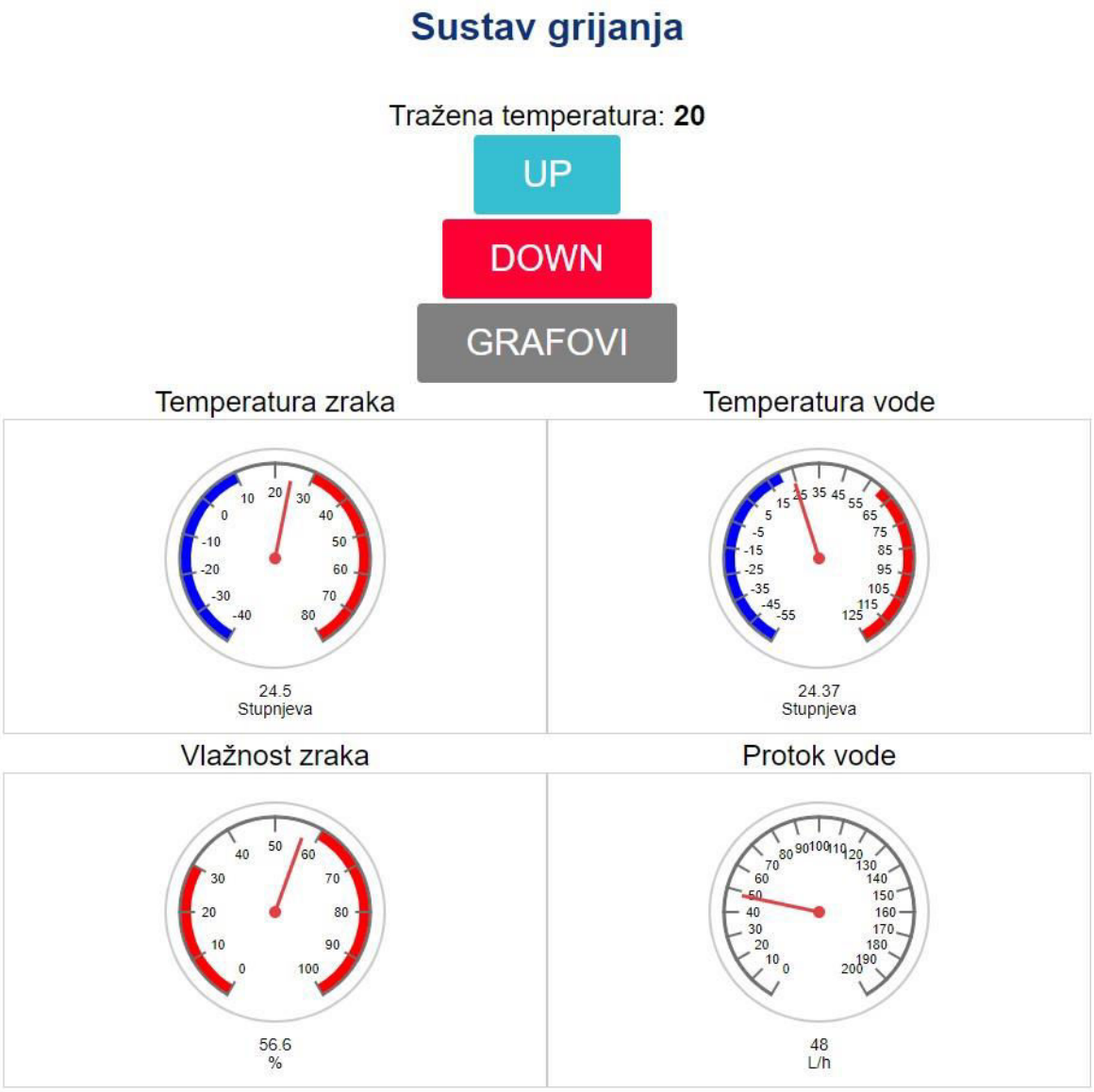
Slika 3.11. *Dijagram toka algoritma centralne jedinice*

3.4. Realizacija upravljačkog i komunikacijskog sučelja

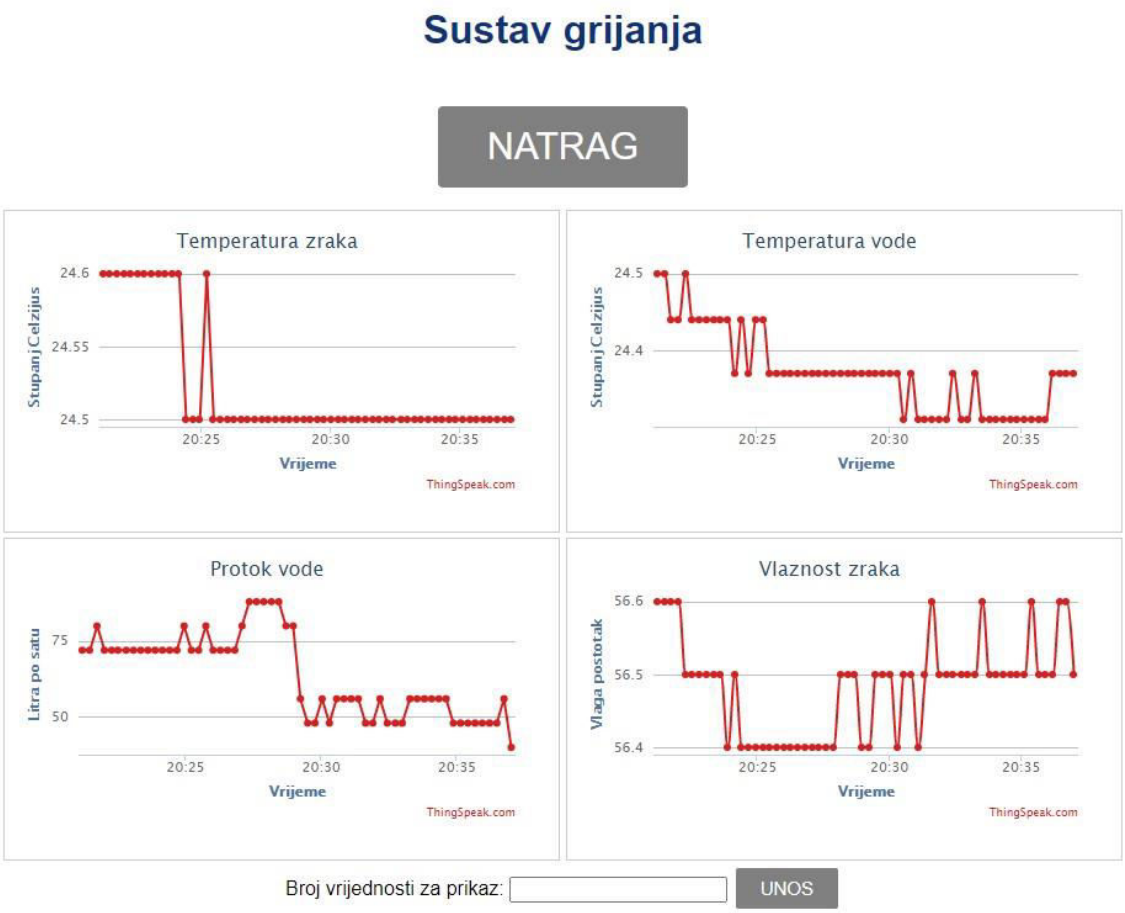


Slika 3.12. *Blok dijagram komunikacije*

Web IoT platforma koja služi kao sučelje kojim se korisniku prikazuju trenutni i prijašnji iznosi procesnih veličina je asinkroni web server. Asinkroni web server razmjenjuje podatke s centralnom jedinicom. Korisnik klikom na prva dva gumba smanjuje i povećava traženu temperaturu u centralnoj jedinici. Centralna jedinica se brine o tome koja stranica će se prikazati u kojem određenom trenutku. Centralna jedinica ostvaruje jednosmjernu komunikaciju s Thingspeak platformom slanjem podataka svakih 15 sekundi. Web server prikazuje grafove s Thingspeak platforme. (Slika 3.12.)



Slika 3.13. *Izgled početne stranice web server-a*



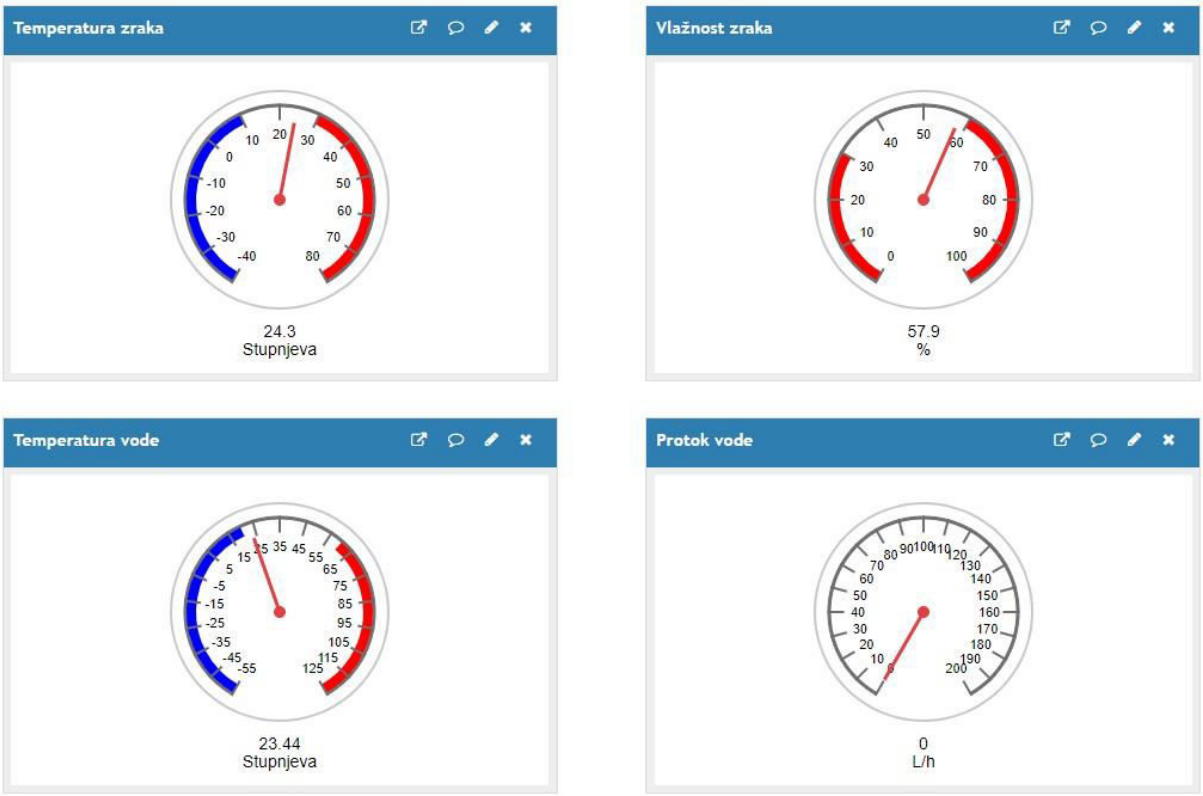
Slika 3.14. *Izgled stranice web server-a s grafovima*

Web server ima dvije stranice. Na početnoj stranici se korisniku predaju na pregled trenutne vrijednosti mjerenih veličina u obliku mjerača i podešava se tražena temperatura s prva dva gumba. (Slika 3.13.) Treći gumb služi za odlazak na drugu stranicu na kojoj je gumb za vraćanje nazad, grafički prikaz prijašnjih vrijednosti procesnih veličina i polje za korisnički unos broja te gumb za unos tog broja koji ujedno i osvježava stranicu kako bi primijenio unešeni broj. (Slika 3.14.) Za detaljan pregled HTML i CSS datoteka pogledati prilog **P.6.**

Za kreiranje grafova zaslužna je platforma Thingspeak. Thingspeak platforma je besplatna do određene mjere te služi za vizualizaciju i analiziranje prosljeđenih podataka. Na navedenoj platformi stvoren je jedan kanal s 4 polja. Svako polje predstavlja jednu mjerenu veličinu. Zbog korištenja besplatne verzije Thingspeak platforme slanje nove vrijednosti je ograničeno na 15 sekundi od zadnje poslane vrijednosti.

Slika 3.15. *Prikaz kanala na Thingspeak platformi*

Svako polje unutar kanala je moguće podesiti po željenim postavkama. Odabrana platforma automatski vizualizira mjerene veličine koje joj je centralna jedinica proslijedila i prikazuje ih u poljima koja su namijenjena za njih. Svako polje je imenovano, imenovane su obje osi grafova i namješten je broj vrijednosti koje svako polje može primiti. (Slika 3.15.) Broj vrijednosti koje polje može primiti je podešen na 60 radi preglednosti grafa.



Slika 3.16. *Prikaz mjerača na Thingspeak platformi*

Za svako polje napravljen je po jedan mjerač koji prikazuje trenutne vrijednosti mjerenih veličina na temelju zadnje vrijednosti poslane sa strane centralne jedinice platformi Thingspeak. Svaki mjerač je podešen na mjerno područje iz karakteristika senzora te je bojom označeno svaki interval vrijednosti koji je izvan intervala optimalnih uvjeta. (Slika 3.16.) Optimalni uvjeti za svaki mjerač su dobiveni istraživanjem.

4. TESTIRANJE I REZULTATI

4.1. Metodologija testiranja

Za potrebe testiranja se koristi Serial monitor Arduino integriranog programskog okruženja koji ispisuje trenutne vrijednosti procesnih veličina i Matlab koji nam na temelju vrijednosti ispisanih na Serial monitoru pravi graf. Za svrhe testiranja testira se svaki senzor pojedinačno i vrijeme potrebno da se centralna jedinica spoji na interneti uspostavi web server.

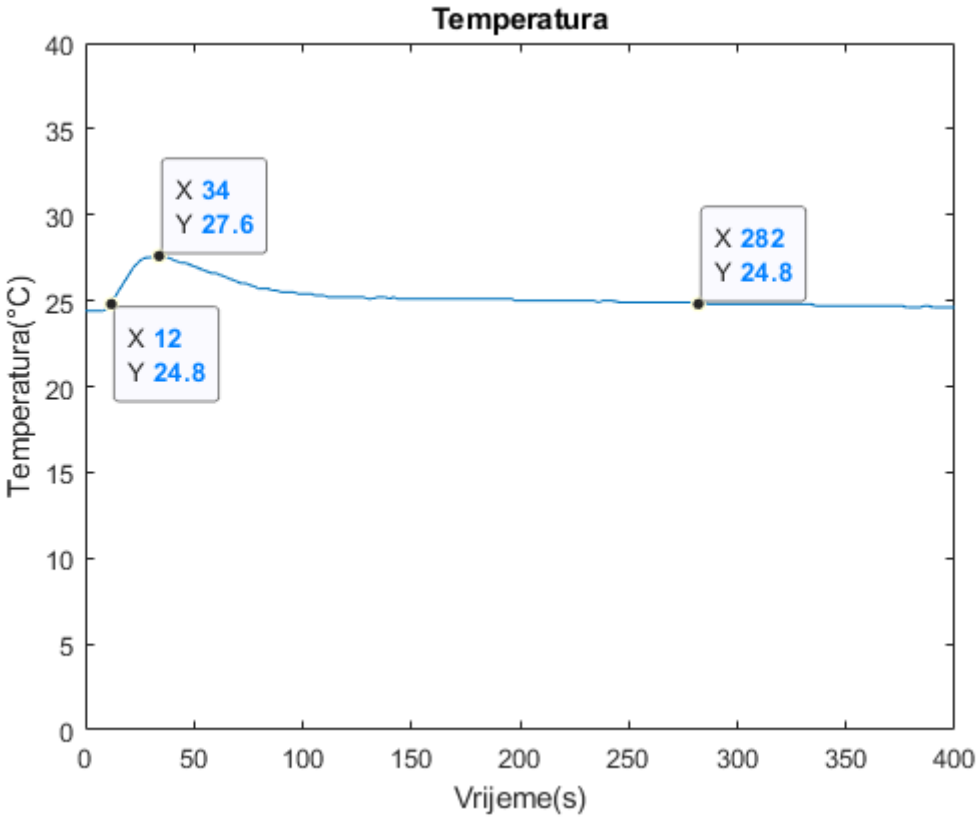
Testiranje će bit napravljeno tako što će se DHT22 i DS18B20 senzorima izmjeriti trenutna veličina, dovesti neki poremećaj i pratiti kada je mjerna veličina približno jednaka probitnom stanju. Promatra se vremenski interval od 400 sekundi, a trenutna vrijednost se uzimati svake 2 sekunde što bi značilo da je ukupno 200 mjerenja. Za YF-S201B senzor protoka voda testiranje je obavljeno na način da se kroz njega održava konstantan protok vode te se na uzorku od 20 mjerenja analiziraju dobiveni rezultati. Za vrijeme spajanja centralne jedinice i uspostavu web server-a se mjeri vrijeme te se također na uzorku od 20 mjerenja analiziraju dobiveni rezultati.

Očekivani rezultat testiranja DHT22 i DS18B20 senzora je da vrijeme potrebno da se njihova temperatura i vlažnost DHT22 senzora vrate približno prvobitnom stanju bude što kraće. Za YF-S201B senzor protoka vode očekivani rezultat je da nemamo preveliko odstupanje od srednje vrijednost skupa 20 vrijednosti, a vrijeme da se centralna jedinica spoji na internet i uspostavi web server bude što kraće.

4.2. Rezultati testiranja

DHT22 senzor smo postavili na isto mjesto s kućnim digitalnim termostatom te im dali neko vrijeme da se prilagode. Nakon toga iščitane su vrijednosti temperatura koje su iznosile 24.40°C za DHT22 senzor te 24.73°C za kućni digitalni termostat. Razlika u vrijednostima temperature zraka DHT22 senzora i kućnog digitalnog termostata se poklapa s intervalom greške DHT22 senzora od ±0.5^{o}C. Zatim u arduinu smo pokrenuli program koji

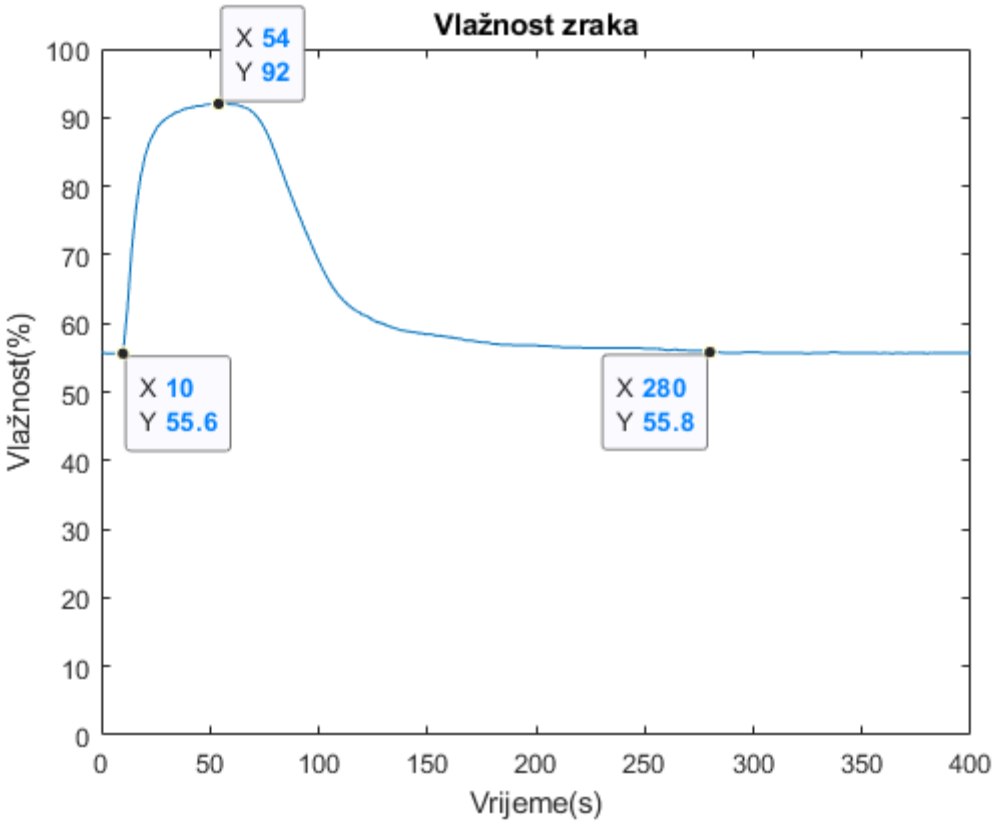
svake dvije sekunde iščitava i ispisuje vrijednost temperature zraka preko DHT22 senzora. Nakon 10 sekundi rada u senzor je puhano kako bi temperatura porasla. Putem ispisanih vrijednosti u Serial monitoru u Matlab-u je napravljen graf te je isti analiziran.



Slika 4.1. *Grafički prikaz temperature zraka mjereno DHT22 senzorom*

Analizom slike 4.1. vidimo kako je senzoru trebalo 248 sekundi od kraja poremećaja da bih se vrijednost temperature vratila u prvobitno stanje.

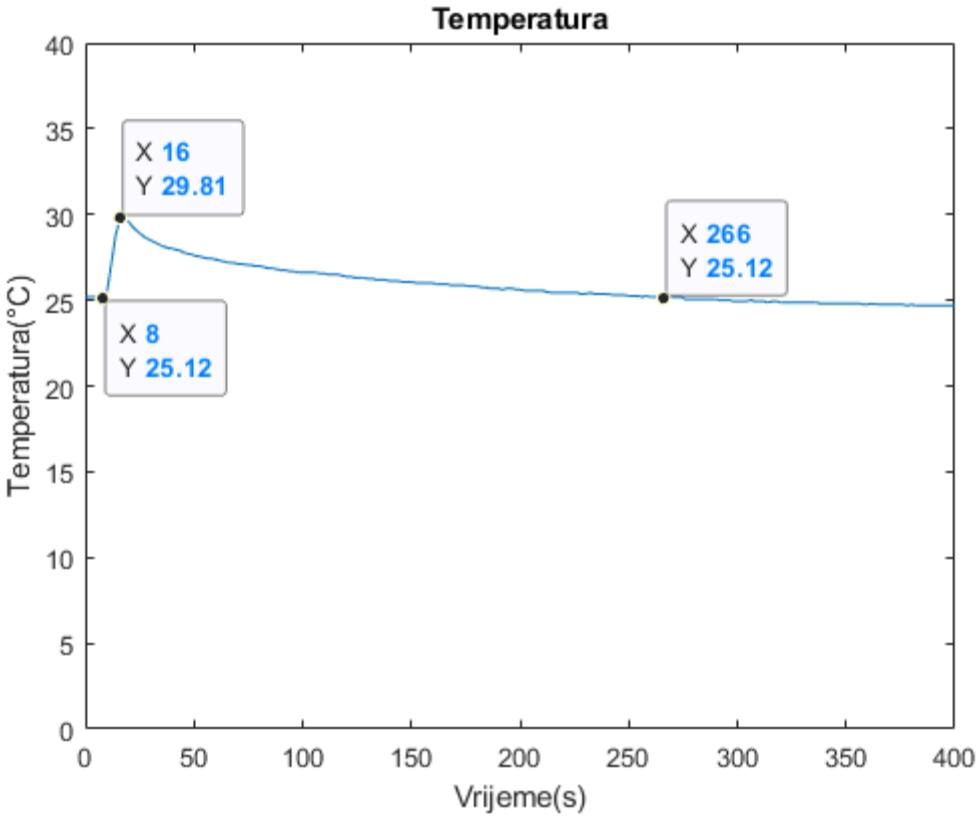
Testiranje vlažnosti je napravljeno na sličan način kao i kod testiranja temperature. Senzor je prvo ostavljen da se neko vrijeme prilagodi uvjetima u prostoriji te je zatim pokrenut program koji je na Serial monitor ispisivao trenutnu vrijednost vlažnosti svake dvije sekunde te je nakon 10 sekundi puhano u senzor što je naglo povećalo vlažnost.



Slika 4.2. *Grafički prikaz vlažnosti zraka mjereno DHT22 senzorom*

Iz slike 4.2. vidimo kako je senzoru trebalo 226 sekundi nakon djelovanja poremećaja da se vrati približno početnoj vrijednosti. U grafu se također vidi da je poremećaj trajao 44 sekunde.

Testiranje DS18B20 senzora se odvilo na identičan način kao i testiranje DHT22 senzora. U svrhu testiranja njime je mjerena temperatura zraka, a ne vode da bi se vidilo vrijeme potrebno da se senzor vrati na početnu vrijednost nakon kraja poremećaja. Navedeni senzor i digitalni kućni termostat su stavljeni na isto mjestu te im je dano neko vrijeme da se prilagode okruženju. DS18B20 je izmjerio 25.05°C, a digitalni kućni termostat 24.92°C. Razlika u ova dva mjerenja se uklapa u odstupanje senzora koje iznosi ±0.5^{o}C. Pokrenut je kod i pratio se vremenski interval od 400 sekundi dok se nova vrijednost uzimala svake dvije sekunde te zatim ispisivala na Serial monitor Arduino programskog okruženja. Pomoću ispisanih vrijednosti se u Matlab-u crtao graf.



Slika 4.3. *Grafički prikaz temperature zraka mjereno DS18B20 senzorom*

Iz slike 4.3. vidimo kako je senzoru trebalo 250 sekundi da se vrati na početnu vrijednost od kraja poremećaja koji je trajao 8 sekundi. Po zakrivljenosti krivulje u trenutku trajanja poremećaja vidimo kako je DS18B20 senzor puno brže reagirao na poremećaj nego DHT22 senzor te mu je vrijednost brže porasla.

Za potrebe testiranja YF-S201B senzora protoka navedeni senzor je spojen na slavinu te je iz slavine puštena konstantna količina vode. Uzeli smo 20 mjerenja u intervalima od jedne sekunde te nad njima vršili analizu.

Redni broj	Rezultat

mjerenja	u litrama po satu
1.	152
2.	141
3.	130
4.	152
5.	141
6.	141
7.	141
8.	152
9.	152
10.	130
11.	141
12.	130
13.	130
14.	130
15.	141
16.	130
17.	141
18.	141
19.	130
20.	141

Tablica 4.1. *Rezultati testiranja YF-S201B senzora*

Iz tablice testiranja YF-S201B senzora možemo izračunati aritmetičku sredinu po formuli:

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n} = 139.35\ L/h \tag{4-1}$$

Prosječno apsolutno odstupanje od aritmetičke sredine računamo po formuli:

$$PAO = \frac{|x_1 - \bar{x}| + |x_2 - \bar{x}| + \cdots + |x_n - \bar{x}|}{n} = 6.545\ L/h \tag{4-2}$$

Vrijeme spajanja centralne jedinice na internet i uspostavu web servera se testira putem Arduino integriranog programskog okruženja. Pravi se 20 mjerenja i analizira se dobiveni rezultat.

Redni broj	Rezultat

mjerenja	u milisekundama
1.	3001
2.	3001
3.	3001
4.	3000
5.	3001
6.	3001
7.	3000
8.	3001
9.	3000
10.	3001
11.	3001
12.	3001
13.	3001
14.	3001
15.	3001
16.	3000
17.	3000
18.	3001
19.	3001
20.	3001

Tablica 4.2. *Rezultati testiranja vremena potrebnog centralnoj jedinici da ostvari vezu s internetom i uspostavu web server-a*

Iz rezultata testiranja vidimo kako je centralnoj jedinici potrebno 3001 milisekunda za spajanje na internet i uspostavu web server-a u 75% slučajeva, 3000 milisekundi u 25% slučajeva. Razlika među mjerenjima je zanemariva jer iznosi jednu milisekundu.

5. ZAKLJUČAK

U završnom radu dizajniran je i izrađen sustav upravljanja ogrjevnim sustavom temeljen na ESP IoT. U radu je prikazano kako pomoću Croduino NOVA32 mikrokontrolera možemo upravljati ogrjevnim sustavom te izmjerene vrijednosti slati na Thingspeak platformu pomoću HTTP klijenta. Također je prikazano kako poslane podatke na IoT web platformu Thingspeak prezentirati i prikazati prošle vrijednosti mjerenih veličina korisniku putem asinkronog web server-a sa statičkom IP adresom. Na kraju rada se vidi obavljeno testiranje koje uključuje vrijeme potrebno senzorima da se vrate na početnu vrijednost nakon utjecaja poremećaja. Velika prednost sustava je veliki broj pinova na mikroupravljaču što nam omogućuje dodavanje još različitih senzora. Za potrebu rada upaljeno stanje ogrjevnog sustava je reprezentirala LED dioda. Izrađeni sustav zahtijevao je znanje većeg broja kolegija tijekom studiranja. Neki od njih su elektronika, komunikacijske mreže i arhitektura računala. Najzahtjevniji dio ovoga rada je bio napraviti, u isto vrijeme, na jednom Croduino uređaju klijenta za Thingspeak platformu i poslužitelja za asinkroni web server korisniku.

LITERATURA

[1] History of heating timeline, dostupno na: https://www.qssupplies.co.uk/history-of-heating-timeline.html , pristup: 10.07.2021.

[2] Temperatura, dostupno na: https://hr.wikipedia.org/wiki/Temperatura , pristup: 10.07.2021.

[3] Temperature Measurement, dostupno na: https://web.mst.edu/~cottrell/ME240/Resources/Temperature/Temperature.pdf , pristup: 10.07.2021.

[4] Thermocouple Techincal Reference Information, https://www.sterlingsensors.co.uk/thermocouples , pristup: 10.07.2021.

[5] Wire Wound RTD Detectors, dostupno na: https://instrumentationforum.com/t/wire-wound-rtd-detectors/5764 , pristup: 10.07.2021.

[6] Definicija termistora Konstrukcija i primjene, dostupno na: https://riverglennapts.com/hr/sensors/784-thermistor-definition-construction-and-applications.html , pristup: 10.07.2021.

[7] Za što je namijenjen termostat u bojleru i kako to radi, dostupno na: https://giropark.ru/bs/truboprovod/zachem-prednaznachen-termostat-v-vodonagrevatele-i-kak-on.html , pristup: 10.07.2021.

[8] Humidity, dostupno na: https://en.wikipedia.org/wiki/Humidity , pristup: 10.07.2021.

[9] The capacitive humidity sensor, dostupno na: https://www.rotronic.com/en-jp/humidity_measurement-feuchtemessung-mesure_de_l_humidite/capacitive-sensors-technical-notes-mr , pristup: 10.07.2021.

[10] Schematic view of resistive type humidity sensor, dostupno na: https://www.researchgate.net/figure/Schematic-view-of-resistive-type-humidity-sensor_fig2_271892038 , pristup: 10.07.2021.

[11] Humidity Sensor – Types and Working principle, dostupno na: https://www.electronicshub.org/humidity-sensor-types-working-principle/ , pristup: 10.07.2021.

[12] Water Flow Meters – How They Work, https://www.azom.com/article.aspx?ArticleID=15058 , pristup: 10.07.2021.

[13] S. Rouse, How a Vortex Flow Meter Works [online], Sierra Instruments, 2018., dostupno na: https://www.sierrainstruments.com/blog/?vortex-flow-meter-works , pristup: 10.07.2021.

[14] Croduino NOVA32, https://e-radionica.com/hr/croduino-nova32.html , pristup: 11.09.2021.

[15] Temperature & Humidity Monitoring, dostupno na: https://steve.fi/hardware/temperature/ , pristup: 11.09.2021.

[16] DS18B20 Waterproof Temperature Sensor, dostupno na: https://www.researchgate.net/figure/DS18B20-waterproof-temperature-sensor_fig2_337490897 , pristup: 11.09.2021.

[17] M. Kumar, Measuring water Flow Rate and Volume using Arduino and Flow Sensor, Circuit Digest, 2020. dostupno na: https://circuitdigest.com/microcontroller-projects/arduino-based-water-flow-sensor , , pristup: 11.09.2021.

SAŽETAK

Cilj rada je bio objasniti i dizajnirat pametni sustav upravljanja ogrjevnim sustavom temeljen na ESP IoT principu. Prikazano je kako, uz pomoć Croduino NOVA32 mikrokontrolera, upravljati sustavom putem web server-a. Korisnik, putem web server-a, dobiva na pregled trenutne i prijašnje vrijednosti mjerenih veličina te mijenja tražena temperaturu. Na kraju rada se može vidjeti testiranje svih senzora i vremena uspostave veze s internetom i podizanja web server-a od strane mikrokontrolera.

ABSTRACT

The goal of the project was to explain and design smart system for heating managment which is based on ESP IoT principle. It is shown how, with the help of Croduino NOVA32 microcontroller, manage the system using a web server. User, using the web server, is given the results of past and present values of the measured values and he changes the required temperature. At the end of the project testing of all of the sensors and the time it takes to connect the microcontroller to the internet and to run a web server is shown.

ŽIVOTOPIS

Fran Štrasser rođen je 28.07.1997. u Osijeku. Osnovnu školu je završio u Višnjevcu. 2016. godine završava III. gimnaziju u Osijeku. Iste godine upisuje preddiplomski sveučilišni studij Računarstva na Fakultetu Elektrotehnike, Računarstva i Informacijskih tehnologija u Osijeku.

PRILOZI

1. Karakteristike Croduino NOVA32 mikrokontrolera su:

- Procesor ima dvije jezgre i njegova frekvencija rada je 240MHz
- 520kB RAM-a, 4MB flash memorije
- 28GPIO pinova
- Ima punjač za litijsku bateriju kao dodatnu opciju napajanja
- Ulazni napon: 3.6V – 5.5V
- Ulazni napon na bateriji: max. 4.2V
- Dimenzije: 78 x 25.6 mm
- mala potrošnja u sleep-u

2. Karakteristike DHT22 senzora su:

- Ulazni napon: 3V – 5V
- Maksimalna struja: 2.5mA
- Raspon mjerenja temperature: od -40^{o}C do 80^{o}C s preciznošću ±0.5^{o}C
- Raspon mjerenja vlažnosti: od 0 do 100% s preciznosću ±2-5%
- Ima 4 pina
- Frekvencija uzimanja uzoraka ne smije biti veća od 0.5Hz

3. Karakteristike DS18B20 senzora su:

- Ulazni napon: 3.3V – 5V
- Maksimalna struja: 1.5mA
- Raspon mjerenja temperature: -55^{o}C do 125^{o}C s preciznošću ±0.5^{o}C

- Na jedan pin moguće postaviti 127 senzora i mjeriti temperaturu na 127 mjesta s jednom žicom

4. Karakteristike YF-S201B senzora protoka su:

- Ulazni napon: 3.5V – 12V
- Raspon mjerenja: 1-30L/min
- Maksimalna radna struja: 15mA

5. Programski kod centralne jedinice:

```
//Uključivanje biblioteka
#include "WiFi.h"
#include "ESPAsyncWebServer.h"
#include "SPIFFS.h"
#include "DHT.h"
#include "HTTPClient.h"
#include "OneWire.h"
#include "DallasTemperature.h"

//Varijabla za slanje podataka na Thingspeak platformu
String serverName = "http://184.106.153.149/update?api_key=59YE1405JW36HG9V";
//Varijable za prijavu na WiFi mrežu
const char* ssid = "A1_812EC8";
const char* password = "3C9EF730E4";
//Pokretanje biblioteka
OneWire oneWire(4);
DallasTemperature sensors(&oneWire);
DHT dht(3, DHT22);
//Varijable za broj vrijednosti koje će se prikazati na grafovima
int numberOfValues = 60;
const char* PARAM_INPUT_1 = "input1";
//Varijable za rad s senzorom protoka
long long impulses;
long long pastImpulses;
unsigned long freq;
unsigned int waterFlow;
int hallSensor = 25;
unsigned long currentTime;
unsigned long pastTime;
//Varijable za uspoređivanje trenutne i tražene temperature
int temperature = 20;
float temp;
float humidity;
//Varijable za delay
const unsigned long interval = 15000;
unsigned long previousTime = 0;
//Podizanje web server-a na portu 80
AsyncWebServer server1(80);
//Podešavanje parametara za statičku IP adresu
IPAddress local_IP(192, 168, 1, 150);
IPAddress gateway(192, 168, 1, 1);
IPAddress subnet(255, 255, 0, 0);
//Setup funkcija
void setup() {
  //Pokretanje senzora
  sensors.begin();
```

```
dht.begin();
pinMode(hallSensor, INPUT);
pinMode(5, OUTPUT);
Serial.begin(115200);
attachInterrupt(25, Rpm, RISING); //Interrupt za mjerenje protoka
sei(); //Pokretanje interrupt-a
currentTime = millis();
pastTime = currentTime;
pastImpulses = impulses;
//Pokretanje SPIFFS-a
if (!SPIFFS.begin(true)) {
  Serial.println("Greška prilikom mount-anja SPIFFS-a!");
  return;
}
//Konfiguriranje statičke IP adrese
if (!WiFi.config(local_IP, gateway, subnet)) {
  Serial.println("Neuspješno konfiguriranje statičke IP adrese!");
}
Serial.print("Spajanje na ");
Serial.println(ssid);
WiFi.begin(ssid, password); //Spajanje na WiFi
while (WiFi.status() != WL_CONNECTED) {
  delay(1000);
  Serial.print(".");
}

Serial.println("");
Serial.println("WiFi spojen.");
Serial.println("IP adresa: ");
Serial.println(WiFi.localIP()); //Printanje IP adrese
//Rješavanje korisničkih zahtjevi sa web server-a
server1.on("/", HTTP_GET, [](AsyncWebServerRequest * request) {
  request->send(SPIFFS, "/index.html", String(), false, processor);
});

server1.on("/style.css", HTTP_GET, [](AsyncWebServerRequest * request) {
  request->send(SPIFFS, "/style.css", "text/css");
});

server1.on("/graphs", HTTP_GET, [](AsyncWebServerRequest * request) {
  request->send(SPIFFS, "/graphs.html", String(), false, iframeProcessor);
});

server1.on("/graphstyle.css", HTTP_GET, [](AsyncWebServerRequest * request) {
  request->send(SPIFFS, "/graphstyle.css", "text/css");
});
```

```
  server1.on("/up", HTTP_GET, [](AsyncWebServerRequest * request) {
    temperature++;
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  server1.on("/down", HTTP_GET, [](AsyncWebServerRequest * request) {
    temperature--;
    request->send(SPIFFS, "/index.html", String(), false, processor);
  });

  server1.on("/get", HTTP_GET, [] (AsyncWebServerRequest * request) {
    String inputMessage;
    String inputParam;
    if (request->hasParam(PARAM_INPUT_1)) {
      inputMessage = request -> getParam(PARAM_INPUT_1)->value();
      inputParam = PARAM_INPUT_1;
      if (inputMessage.toInt() > 0)
        numberOfValues = inputMessage.toInt();
    }
    Serial.println("inputMessage");
    request->send(SPIFFS, "/graphs.html", String(), false, iframeProcessor);
  });

  server1.begin(); //Pokretanje web server-a
}
//Loop funkcija
void loop() {
  getSensorReadings();
  sendToThingspeak();
  temperatureCheck();
}
//Funkcija za očitavanje trenutnih vrijednosti mjerenih veličina
void getSensorReadings() {
  temp = dht.readTemperature();
  humidity = dht.readHumidity();
  sensors.requestTemperatures();
  currentTime = millis();
  freq = (impulses - pastImpulses) / ((currentTime - pastTime) / 1000);
  waterFlow = freq * 60 * 60 / 450;
  pastImpulses = impulses;
  pastTime = currentTime;
  Serial.println(waterFlow);
}
```

```
//Funkcija koju poziva interrupt
void Rpm ()
{
  impulses++;

}
//Funkcija za slanje trenutnih vrijednosti mjerenih veličina na Thingspeak platformu
void sendToThingspeak() {
  if (WiFi.status() == WL_CONNECTED) {
    WiFiClient client;
    HTTPClient http;
    //String u kojem su trenutne vrijednosti mjerenih veličina
    String serverPath = serverName + "&field1=" + String(temp) + "&field2=" + String(humidity) + "&field3=" + String(sensors.getTempCByIndex(0)) + "&field4=" + String(waterFlow);

    http.begin(client, serverPath.c_str()); //Pokretanje HTTP klijenta

    int httpResponseCode = http.GET(); //Odgovor na slanje zahtjeva

    if (httpResponseCode > 0) {
      Serial.print("HTTP odgovor: ");
      Serial.println(httpResponseCode);
      String payload = http.getString();
      Serial.println(payload);
    }
    else {
      Serial.print("Error kod: ");
      Serial.println(httpResponseCode);
    }
    http.end();
  }
  else {
    Serial.println("WiFi isključen");
  }
}
```

```
//Funkcija za paljenje LED diode unutar delay-a
void temperatureCheck() {
  previousTime = millis();
  unsigned long currentTime1 = millis();
  while (currentTime1 - previousTime < interval) {
    temp = dht.readTemperature();
    if (temperature > temp && digitalRead(5) == LOW) {
      digitalWrite(5, HIGH);
    }
    if (temperature <= temp && digitalRead(5) == HIGH) {
      digitalWrite(5, LOW);
    }
    currentTime1 = millis();
  }
}
//Funkcija za ispis vrijednosti tražene temperature na web server-u
String processor(const String& var) {
  Serial.println(var);
  String returnString;
  if (var == "PLACEHOLDER") {
    Serial.println(temperature);
    returnString = String(temperature);
    return returnString;
  }
  return String();
}
```

```
//Funkcija za ispis grafova na web server-u
String iframeProcessor(const String& var) {
  Serial.println(var);
  const String quote = "\"";
  String returnString = "";
  if (var == "IFRAMEPLACEHOLDER") {
    Serial.println(numberOfValues);
    returnString += "<iframe width=" + quote + "450" + quote + " height=" + quote + "260" + quote + " style="
    + quote + "border: 1px solid #cccccc;" + quote + " src=" + quote + "https://thingspeak.com/channels/1479412/charts/1?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results="
    + numberOfValues + "&title=Temperatura+zraka&type=line&xaxis=Vrijeme&yaxis=Stupanj+Celzijus" + quote + "></iframe> ";

    returnString += "<iframe width=" + quote + "450" + quote + " height=" + quote + "260" + quote + " style="
    + quote + "border: 1px solid #cccccc;" + quote + " src=" + quote + "https://thingspeak.com/channels/1479412/charts/3?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results="
    + numberOfValues + "&title=Temperatura+vode&type=spline&xaxis=Vrijeme&yaxis=Stupanj+Celzijus" + quote + "></iframe> ";

    returnString += "<iframe width=" + quote + "450" + quote + " height=" + quote + "260" + quote + " style="
    + quote + "border: 1px solid #cccccc;" + quote + " src=" + quote + "https://thingspeak.com/channels/1479412/charts/4?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results="
    + numberOfValues + "&title=Protok+vode&type=line&xaxis=Vrijeme&yaxis=Litra+po+satu" + quote + "></iframe> ";

    returnString += "<iframe width=" + quote + "450" + quote + " height=" + quote + "260" + quote + " style="
    + quote + "border: 1px solid #cccccc;" + quote + " src=" + quote + "https://thingspeak.com/channels/1479412/charts/2?bgcolor=%23ffffff&color=%23d62020&dynamic=true&results="
    + numberOfValues + "&title=Vlaznost+zraka&type=spline&xaxis=Vrijeme&yaxis=Vlaga+postotak" + quote + "></iframe> ";

    return returnString;
  }
  return String();
}
```

6. HTML i CSS datoteke

```
<!DOCTYPE html>
<html>
<head>
  <title>Sustav grijanja</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
  <link rel="stylesheet" type="text/css" href="style.css">
  <meta charset="UTF-8" />
</head>
<body>
  <h1>Sustav grijanja</h1>
  <p>Tražena temperatura: <strong> %PLACEHOLDER%</strong></p>
  <p><a href="/up"><button class="button">UP</button></a></p>
  <p><a href="/down"><button class="button button2">DOWN</button></a></p>
  <p><a href="/graphs"><button class="button button3">GRAFOVI</button></a></p>
  <div class="float-container">
    <div class="float-child">
        <p>Temperatura zraka</p>
        <iframe  width="450" height="260" style="border: 1px solid #cccccc;"
        src="https://thingspeak.com/channels/1479412/widgets/349154"></iframe>
        <p>Vlažnost zraka</p>
        <iframe width="450" height="260" style="border: 1px solid #cccccc;"
        src="https://thingspeak.com/channels/1479412/widgets/350145"></iframe>
    </div>

    <div class="float-child">
        <p>Temperatura vode</p>
        <iframe  width="450" height="260" style="border: 1px solid #cccccc;"
        src="https://thingspeak.com/channels/1479412/widgets/351988"></iframe>
        <p>Protok vode</p>
        <iframe width="450" height="260" style="border: 1px solid #cccccc;"
        src="https://thingspeak.com/channels/1479412/widgets/354021"></iframe>
    </div>
  </div>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<head>
  <title>Sustav grijanja</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" href="data:,">
  <link rel="stylesheet" type="text/css" href="graphstyle.css">
</head>
<body>
  <h1>Sustav grijanja</h1>
  <p><a href="/"><button class="button button4">NATRAG</button></a></p>
  %IFRAMEPLACEHOLDER%
  <form action="/get">
    Broj vrijednosti za prikaz: <input type="text" name="input1">
    <input type="submit" value="UNOS">
  </form><br>
</body>
</html>
```

```
html {
  font-family: Helvetica;
  display: inline-block;
  margin: 20px auto;
  text-align: center;
}
h1{
  color: #0F3376;
  padding: 2vh;
}
p{
  font-size: 1.5rem;
  margin:auto;
  display:block;
}
.button {
  display: inline-block;
  background-color: #36BED3;
  border: none;
  border-radius: 4px;
  color: white;
  padding: 16px 40px;
  text-decoration: none;
  font-size: 30px;
  margin: 2px;
  cursor: pointer;
}
.button2 {
  background-color: #FF0035;
}
.button3 {
  background-color: #808080;
}
.float-container {
    display: inline-block;
    padding: 0px;
}

.float-child {
    width: auto;
    float: left;
    padding: 0px;

}
```

```
html {
  font-family: Helvetica;
  display: inline-block;
  margin: 20px auto;
  text-align: center;
}
h1{
  color: #0F3376;
  padding: 2vh;
}

.button4 {
  display: inline-block;
  background-color: #808080;
  border: none;
  border-radius: 4px;
  color: white;
  padding: 16px 40px;
  text-decoration: none;
  font-size: 30px;
  margin: 2px;
  cursor: pointer;
}

input[type=submit] {
  background-color: #808080;
  border: none;
  border-radius: 2px;
  color: white;
  padding: 8px 20px;
  font-size: 15px;
  margin: 2px;
  cursor: pointer;
}
```