Sustav za prevenciju zamagljivanja teleskopske optike

Šrempf, Michael

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

Sveučilišni studij

SUSTAV ZA PREVENCIJU ZAMAGLJIVANJA TELESKOPSKE OPTIKE

Završni rad

Michael Šrempf

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:	Michael Šrempf
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4286, 26.07.2018.
OIB studenta:	48434917301
Mentor:	Izv.prof.dr.sc. Tomislav Keser
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Sustav za prevenciju zamagljivanja teleskopske optike
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: 3 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, 28.09.2021.

Ime i prezime studenta:	Michael Šrempf
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4286, 26.07.2018.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Sustav za prevenciju zamagljivanja teleskopske optike**

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:

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

IZJAVA

o odobrenju za pohranu i objavu ocjenskog rada

kojom ja __Michael Šrempf_____, OIB: __48434917301_____, student/ica Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek na studiju _Preddiplomski sveučilišni studij Računarstvo_, kao autor/ica ocjenskog rada pod naslovom: _Sustav za prevenciju zamagljivanja teleskopske optike**,**
dajem odobrenje da se, bez naknade, trajno pohrani moj ocjenski rad u javno dostupnom digitalnom repozitoriju ustanove Fakulteta elektrotehnike, računarstva i informacijskih tehnologija Osijek i Sveučilišta te u javnoj internetskoj bazi radova Nacionalne i sveučilišne knjižnice u Zagrebu, sukladno obvezi iz odredbe članka 83. stavka 11. *Zakona o znanstvenoj djelatnosti i visokom obrazovanju* (NN 123/03, 198/03, 105/04, 174/04, 02/07, 46/07, 45/09, 63/11, 94/13, 139/13, 101/14, 60/15).
Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog ocjenskog rada. Ovom izjavom, kao autor/ica ocjenskog rada dajem odobrenje i da se moj ocjenski rad, bez naknade, trajno javno objavi i besplatno učini dostupnim:

a) široj javnosti

b) studentima/icama i djelatnicima/ama ustanove

c) široj javnosti, ali nakon proteka 6 / 12 / 24 mjeseci (zaokružite odgovarajući broj mjeseci).

**U slučaju potrebe dodatnog ograničavanja pristupa Vašem ocjenskom radu, podnosi se obrazloženi zahtjev nadležnom tijelu Ustanove.*

__Osijek, 28.09.2021._____ _____

(mjesto i datum) (vlastoručni potpis studenta/ice)

SADRŽAJ

1. UVOD

Kondenzacija na teleskopskoj optici predstavlja veliki problem tijekom noćnog promatranja neba. Kondenzacija se stvara na površini optike te otežava ili u nekim slučajevima potpuno sprječava promatranje. Uklanjanje kapljica s papirom ili drugim sredstvima može oštetiti optiku. Cilj ovoga rada je osmisliti pouzdan sustav koji će spriječiti stvaranje kondenzacije.

U prvom poglavlju bit će opisana teorija problema te prijedlog sklopovskog, algoritamskog i komunikacijskog rješenja. U drugom dijelu rada bit će opisana realizacija sustava. Na kraju će biti prikazani rezultati testiranja koji će biti grafički prikazani i objašnjeni.

1.1. Zadatak i struktura rada

Zadatak završnog rada je izraditi sustav za prevenciju zamagljivanja teleskopske optike, tako što će se izraditi sustav koji će mjeriti temperaturu optike, temperature zraka i vlagu zraka te prema potrebi uključivati grijač koji će zagrijavati leću na određenu temperaturu. Za kontrolu sustava koristit će se ESP32 sustav koji će se ponašati kao pristupna točka. Pomoću HTML - a napravit će se korisničko sučelje na koje će se korisnik spajati putem Wi - Fi mreže. Korisniku će biti prikazane vrijednosti temperature i vlage te će moći odabrati jedan od ponuđenih načina rada. Prvi način rada je da sustav automatski odlučuje kada je potrebno uključiti grijač, drugi način rada je da korisnik odabere željenu temperaturu optike koju će sustav održavati, treći način rada je da grijač radi dok ga korisnik ne isključi te četvrti gdje korisnik može odabrati jačinu kojom će grijač raditi.

2. PREVENCIJA ZAMAGLJIVANJA TELESKOPSKE OPTIKE

Vlažnost zraka može se opisati apsolutnom vlažnosti i relativnom vlažnosti [1]. Apsolutna vlažnost predstavlja količnik mase vodene pare i volumena zraka. Relativna vlažnost predstavlja omjer apsolutne vlage koja je u zraku i maksimalne vlage koju zrak može sadržavati pri toj temperaturi. Mjerna jedinica apsolutne vlažnosti je kilogram po kubičnom metru (kg/m^3) i mjeri se apsorpcijskim vlagomjerom. Relativna vlažnost zraka prikazuje se u postotcima i mjeri sa psihometrom.

Rosište je temperatura pri kojoj zrak, uz nepromijenjeni tlak i nepromijenjenu količinu vodene pare, postaje zasićen vodenom parom. Hlađenjem zraka ispod temperature rosišta, višak vodene pare prelazi u tekuće ili čvrsto agregatno stanje [2]. Ta reakcija naziva se kondenzacija.

Na teleskopu dolazi do kondenzacije zato što je staklena optika hladnija od vlažnog zraka, odnosno hladnija od temperature rosišta te se vlažan zrak u kontaktu s optikom naglo hladi i kondenzira. Kapljice na teleskopskoj optici otežavaju ili u nekim slučajevima potpuno sprječavaju korištenje teleskopa.

2.1. Pregled postojećeg stanja i metode prevencije zamagljivanja teleskopske optike

Zamagljivanje nije problem koji se samo pojavljuje na teleskopskoj optici. Zamagljivanje se može dogoditi i na dvogledu i kameri. Kako bi se problem zamagljivanja riješio, koristi se nekoliko metoda [3].

Prva metoda je upotreba štita koji se postavlja na kraj optike. Štit usporava direktni dotok zraka do optike i usporava radijaciju topline.

Druga metoda je korištenje grijača koji se postavlja s vanjske strane optike te zagrijava optiku, no najčešće se koristi zajedno sa štitom kako bi se postigli što bolji rezultati.

U radu će biti prikazana izrada sustava koji će kombinacijom navedenih metoda spriječiti zamagljivanje teleskopske optike. Sustav će se sastojati od mikroupravljača, grijača i senzora za mjerenje temperature, vlage i tlaka zraka. Mikroupravljač će na temelju prikupljenih podataka sa senzora uključivati i isključivati grijač. Kontrolom grijača pokušava se osigurati jednaka temperatura optike i okoline. Takvom kontrolom temperature optike sprječava se naglo hlađenje zraka i uklanja mogućnost kondenzacije vlage.

Postoje tri vrste teleskopa. Refraktorski teleskopi koji koriste leće, reflektorski teleskopi koji koriste zrcala i katadiopterski teleskopi koji koriste kombinaciju leća i zrcala [4]. Kako bi se na refraktorskom teleskopu dobila veća slika, potrebna je veća i deblja leća, za razliku od reflektorskog teleskopa kojemu je samo potrebno povećati zrcalo.

Astronomski opservatoriji koriste zakrivljena zrcala zato što ih je lakše izraditi i održavati [5]. Zrcala su puno lakša od debelih leća pa se koriste i na teleskopima u svemiru.

Amaterski sustavi s manjim lećama i zrcalima mogu koristiti metode za prevenciju zamagljivanja, poput štita ili grijača, dok profesionalni astronomski opservatoriji ne mogu primijeniti iste metode na velika zrcala.

Zagrijavanje velikih zrcala nije praktično, jer njihovim zagrijavanjem mijenja se indeks loma svjetlosti koji može utjecati na točnost prikupljenih podataka [6].

Zato astronomski opservatoriji pri biranju lokacije za izgradnju teleskopa uzimaju u obzir količinu vlage u zraku, atmosferske turbulencije i količinu čestica prašine i soli.

Količina čestica prašine i soli u zraku može znatno utjecati na zamagljivanje optike. Kada je optika čista i u okolnom zraku ima jako malo čestica prašine, teško dolazi do kondenzacije [7].

Vlaga u zraku utječe na zamagljivanje teleskopske optike, ali utječe i na indeks loma svjetlosti. Osim vlage i zagrijavanja zrcala, na indeks loma svjetlosti utječu atmosferske turbulencije [8].

Mjesta koja izbjegavaju navedene probleme većinom se nalaze na velikim nadmorskim visinama.

Jedno takvo mjesto Europski je južni opservatorij koji je smješten u pustinji Atacama u Čileu. Dnevne temperature u pustinji Atacama kreću se između 0 °C i 25 °C te u njoj nije zabilježena kiša već 500 godina. Osim što vremenski uvjeti odgovaraju za izgradnju teleskopa, jako slabo je naseljena i zbog toga nema svjetlosne zagađenosti.

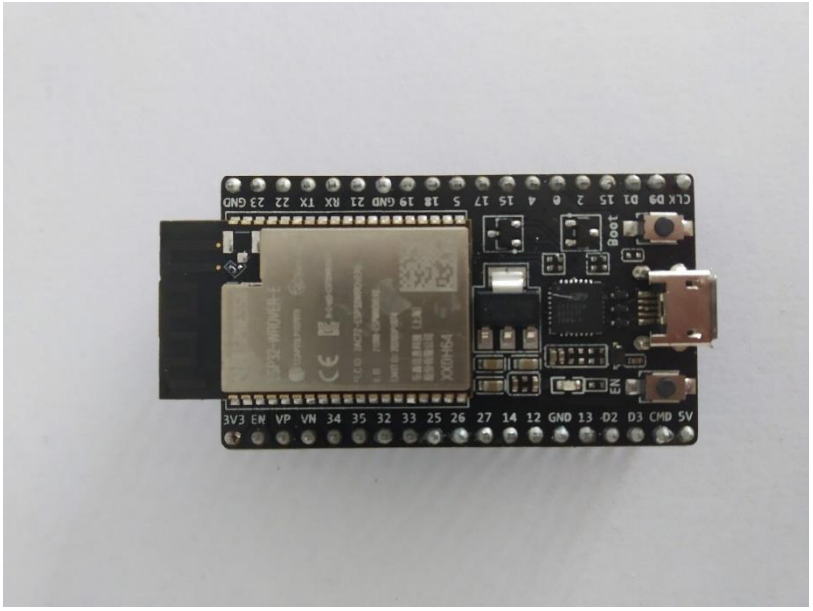
Kako znanstvenici žele sve bolje i bolje slike, traže nova mjesta sa sve boljim uvjetima. Jedno takvo mjesto na najvišoj je ledenoj kupoli na Antarktici [9].

Ali ni na jednom mjestu na Zemlji ne može se zaobići problem koji stvaraju turbulencije zraka i atmosfera. Taj problem riješen je slanjem teleskopa u svemir [10]. Hubble teleskop jedan je od najpoznatijih teleskopa poslanih u svemir. Pomogao je odrediti starost svemira, napravio je 3D kartu tamne materije, otkrio da gotovo svaka galaksija ima crnu rupu u središtu i još mnogo stvari [11].

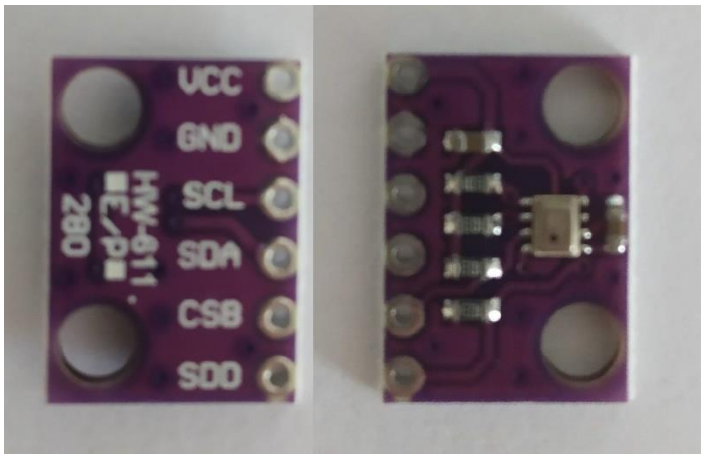
2.2. Prijedlog sklopovskog rješenja

Osnovni dijelovi ovoga sustava bit će ESP32 mikroupravljač, grijač, napajanje i senzori za mjerenje temperature, vlage i tlaka zraka.

Koristit će se ESP32 sustav koji pripada jeftinim programabilnim sustavima male potrošnje s integriranim Wi - Fi i Bluetooth sustavom. Od ponuđenih ESP32 sustava odabrana je DevKitC - VE pločica, prikazana na slici 2.1. s WROVER - E mikroupravljačem. Ona je odabrana zato što dolazi s ugrađenom Wi - Fi antenom i 8 MB brze memorije na koju će se spremiti web stranica.



Slika 2.1. DevKitC - VE razvojna pločica s WROVER - E mikroupravljačem



Slika 2.2. BMP280 senzor za temperaturu i tlak zraka

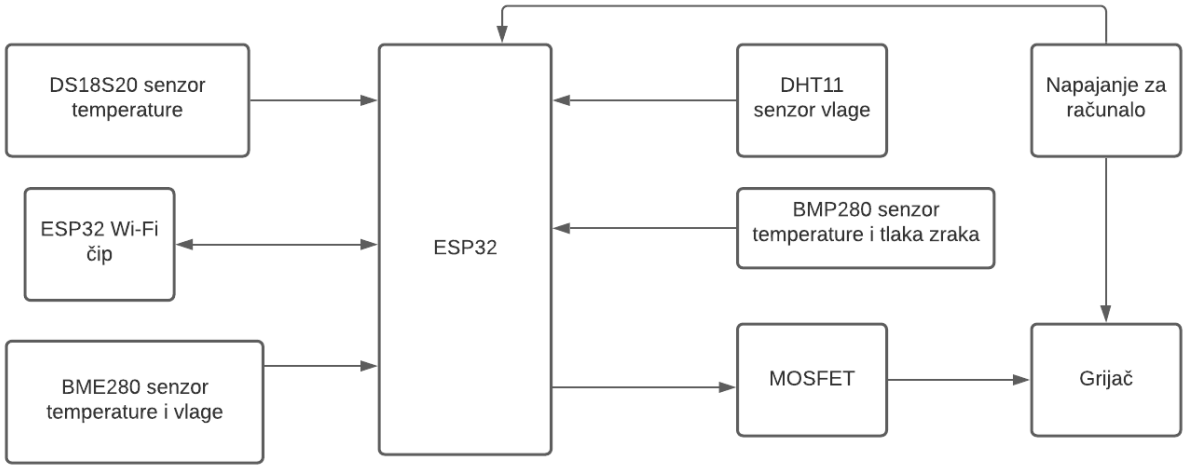
Temperatura optike mjeriti će se DS18S20 senzorom. U okolini bit će smješten DHT11 senzor za mjerenje vlage i BMP280 senzor, prikazan na slici 2.2. za mjerenje temperature i tlaka zraka. U blizini teleskopske optike bit će postavljen BME280 senzor, prikazan na slici 2.3. koji će mjeriti temperaturu, vlagu i tlak zraka.

Slika 2.3. BME280 senzor za temperaturu zraka i vlagu

Budući da mikroupravljač ne može samostalno napajati grijač, koristit će se napajanje za računalo koje ima ugrađen ispravljač da izmjeničnu struju ispravlja u istosmjernu. Napajanje koje će se koristiti ima pretvarače s kojima nam daje pristup istosmjernim stabilnim naponima od 12 V i 5 V koji su potrebni za izvedbu sustava. 12 V je potrebno za grijač, dok je 5 V potrebno za mikroupravljač.

ESP32 mikroupravljač pomoću MOSFET - a kontrolirati će snagu grijača. MOSFET je vrsta unipolarnih tranzistora. Za razliku od bipolarnih tranzistora s kojima se upravlja protokom struje, MOSFET - om se upravlja s naponom. MOSFET ima tri nožice koje predstavljaju izvor, odvod i upravljačku elektrodu. Povećavanjem i smanjivanjem napona na upravljačkoj elektrodi kontrolira se otpor između izvora i odvoda. U ovome radu koristit će se IRFZ44N MOSFET N tipa koji s povećanjem napona na upravljačkoj elektrodi smanjuje otpor.

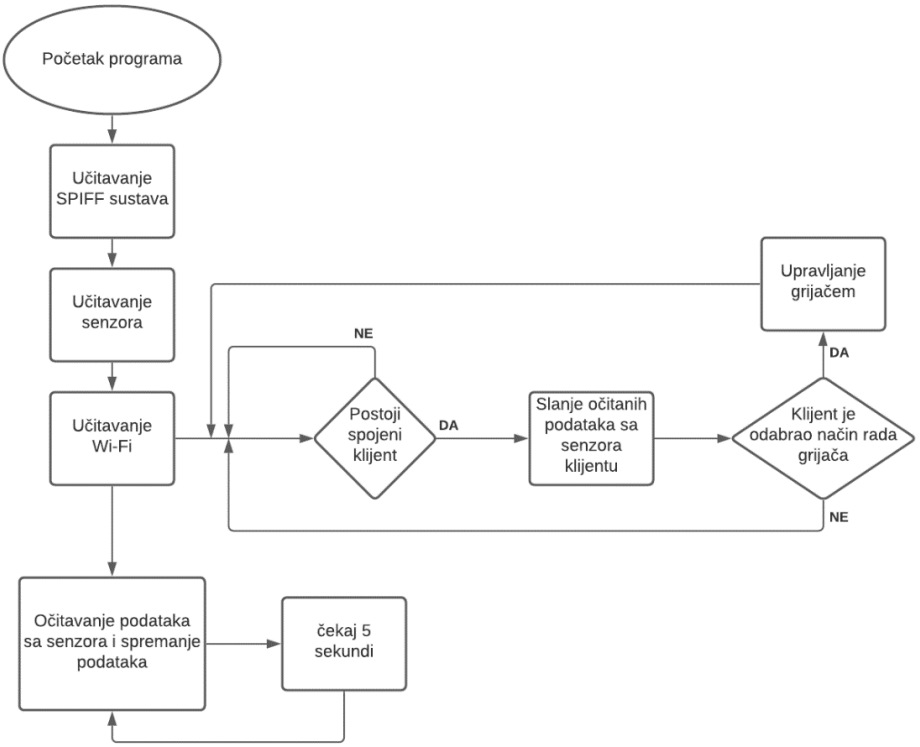
Na slici 2.4. prikazan je odnos svih dijelova sustava. ESP32 mikroupravljač prikupljati će podatke sa senzora te će prikupljene podatke pomoću Wi – Fi - ja slati korisniku. Ta veza s korisnikom bit će dvosmjerna kako bi korisnik mogao slati mikroupravljaču upute za reguliranje temperature teleskopske optike. Mikroupravljač će pomoću svog napona na MOSFET - u kontrolirati grijač. ESP32 mikroupravljač napajati će se sa 5 V, a grijač sa 12 V.



Slika 2.4. Blok dijagram sustava

2.3. Prijedlog algoritamskog rješenja

Kao što se može vidjeti u blok dijagramu 2.5., kada se sustav uključi prvo učitava i pokreće potrebne sustave za rad. Potreban je sustav za rad s datotekama koji se zove SPIFF sustav, sustav za komunikaciju sa senzorima i sustav za Wi - Fi komunikaciju. Ovi sustavi biti će detaljnije objašnjeni u nastavku rada.



Slika 2.5. Blok dijagram algoritma za upravljanje sustava

Nakon što je sve učitano, svakih pet sekundi očitavaju se podaci sa senzora te ako ima spojenih klijenata šalju im se rezultati. Drugi dio sustava istovremeno provjerava ima li spojenih klijenata te obavlja komunikaciju s njima. U slučaju da klijent odabere jedan od četiri načina rada grijača, šalje se poruka mikroupravljaču, nakon čega mikroupravljač kreće sa regulacijom temperature optike.

3. REALIZACIJA SUSTAVA

3.1. Korišteni alati i programska okruženja

ESP32 podržava nekoliko programskih jezika kao što su MicroPython, Arduino, NodeMCU i drugi, ali za ovaj sustav korišten je Arduino. Za pisanje programa korišten je Visual Studio Code tekstualni editor. Kako bi se pomoću odabranog tekstualnog editora mogao programirati ESP32 mikroupravljač, bilo je potrebno instalirati dodatak Platform IO koji omogućuje otklanjanje pogrešaka i učitavanje programa. Spomenuti dodatak omogućuje komunikaciju između računala i mikroupravljača putem USB kabela. Uspostavljena komunikacija između računala i mikroupravljača omogućila je testiranje rada dijelova programa.

Web stranica također je pisana u Visual Studio Code tekstualnom editoru. Za nju korišteni su HTML označni jezik, CSS stilski jezik i JavaScript skriptni programski jezik.

3.2. Realizacija sklopovskog rješenja

Glavni dio ovoga sustava je ESP32 mikroupravljač koji kontrolira sve komponente ovoga sustava. Na mikroupravljaču spremljena je web stranica koja se poslužuje klijentu. Klijent putem web stranice komunicira s mikroupravljačem. Detaljne karakteristike DevKitC - VE mikroupravljača nalaze se u prilogu P.3.1.

Sustav je podijeljen u dva dijela. Prvi dio senzori su koji prikupljaju potrebne podatke, a drugi dio grijač.

Prvi dio sastoji se od četiriju senzora na tri mjesta.

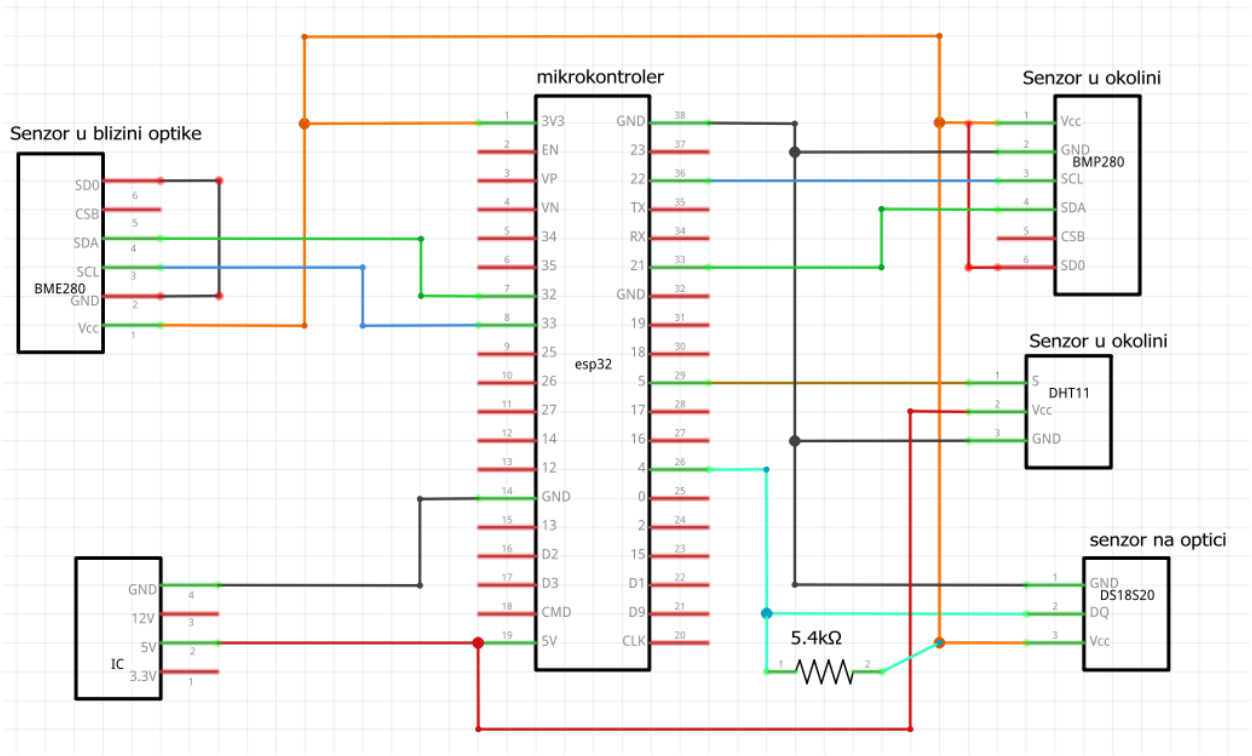
Prvo mjesto mjerenja teleskopska je optika. Na teleskopskoj optici nalazi se DS18S20 senzor koji mjeri njenu temperaturu. Odabrani senzor vrlo je malih dimenzija kako ne bi smetao pri korištenju teleskopa.

Drugo mjesto unutar je štita teleskopa gdje se nalazi BME280 senzor koji mjeri temperaturu i vlagu u blizini teleskopske optike.

Treće mjesto smješteno je izvan teleskopa gdje se nalaze DHT11 senzor koji mjeri vlagu u zraku i BMP280 senzor koji mjeri temperaturu i tlak zraka.

Shema koja prikazuje kako su spojeni senzori može se vidjeti na slici 3.1.

BMP280 i BME280 senzori koriste I2C komunikacijski protokol. I2C je sinkroni komunikacijski protokol koji ima dvije signalne linije. Jedna je za prijenos podataka, a druga služi kao vremenski brojač. I2C komunikacijski protokol radi na način *master* / *slave*. Unatoč tome što pomoću I2C komunikacije na jednome kanalu možemo biti spojeno 128 uređaja, koriste se dva komunikacijska kanala kako bi pozicioniranje senzora bilo lakše. Pinovima 22 i 33 na mikroupravljaču prolazi signal za sinkronizaciju, a pinovima 21 i 32 šalju se podaci. Za razliku od ovih dvaju senzora kojima su potrebne dvije žice, za komunikaciju senzorima DHT11 i DS18S20 potrebna je samo jedna žica. Svi senzori napajaju se preko ESP32 mikroupravljača. DHT11 priključen je na 5 V, dok je ostalim senzorima potrebno 3.3 V.



Slika 3.1. Shema spoja senzora i mikroupravljača

Drugi dio sustava sastoji se od grijača, MOSFET - a i otpornika. Mikroupravljač ne može pružiti grijaču dovoljan napon da dosegne potrebnu snagu pa je grijač spojen na napajanje od 12 V. Da bi mikroupravljač mogao kontrolirati grijač, potreban je MOSFET. Korišten je IRFZ44N MOSFET N tipa zato što ga se može kontrolirati s rasponom napona koji mikroupravljač može pružiti.

Nakon što je odabran MOSFET, bilo je potrebno izračunati treba li mu dodatni hladnjak za rad sa naponom koji se koristi u ovome sustavu. Za računanje toga korišten je otpor MOSFET - a R_{DS}, otpor grijača R, maksimalna temperatura pri kojoj MOSFET može raditi T_J, termalni otpor MOSFET - a $R_{\theta JA}$ i temperatura okoline T_A. Potrebni podaci pronađeni su u tehničkim specifikacijama MOSFET - a [12].

Prvo je izračunata maksimalna struja formulom 3.1., nakon toga snaga sustava formulom 3.2., a na kraju maksimalna snaga za koju ne treba hlađenje 3.3.

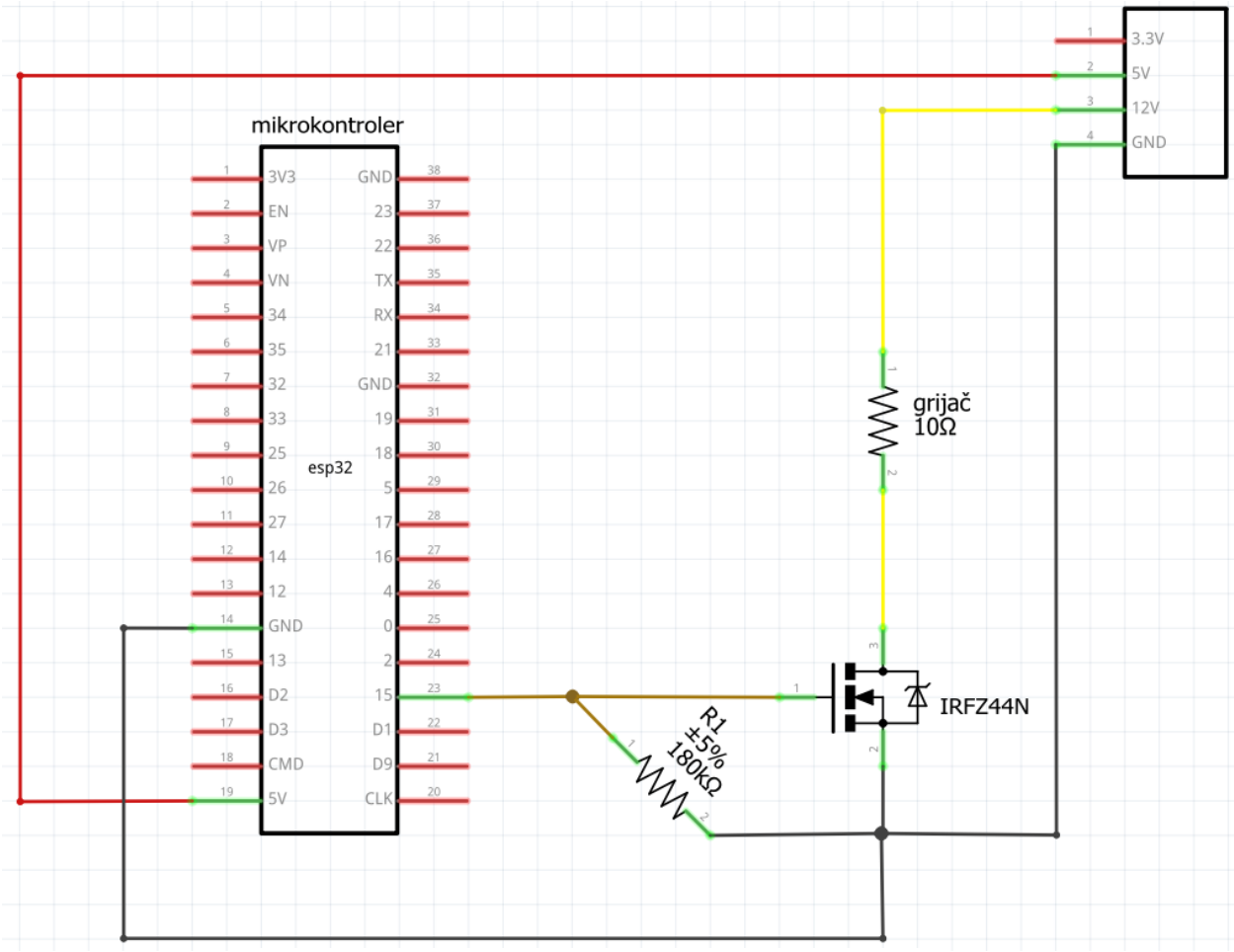
$$I_D = \frac{U}{R_{UK}} = \frac{U}{R + R_{DS}} = \frac{12}{10 + 0.0175} = 1.2\ A \tag{3.1.}$$

$$P_D = {I_D}^2 * R_{DS} = 1.2^2 * 0.0175 = 0.0252\ W \tag{3.2.}$$

$$P_{Dmax} = \frac{T_J - T_A}{R_{\theta JA}} = \frac{175 - 25}{62} = 2.42\ W \tag{3.3.}$$

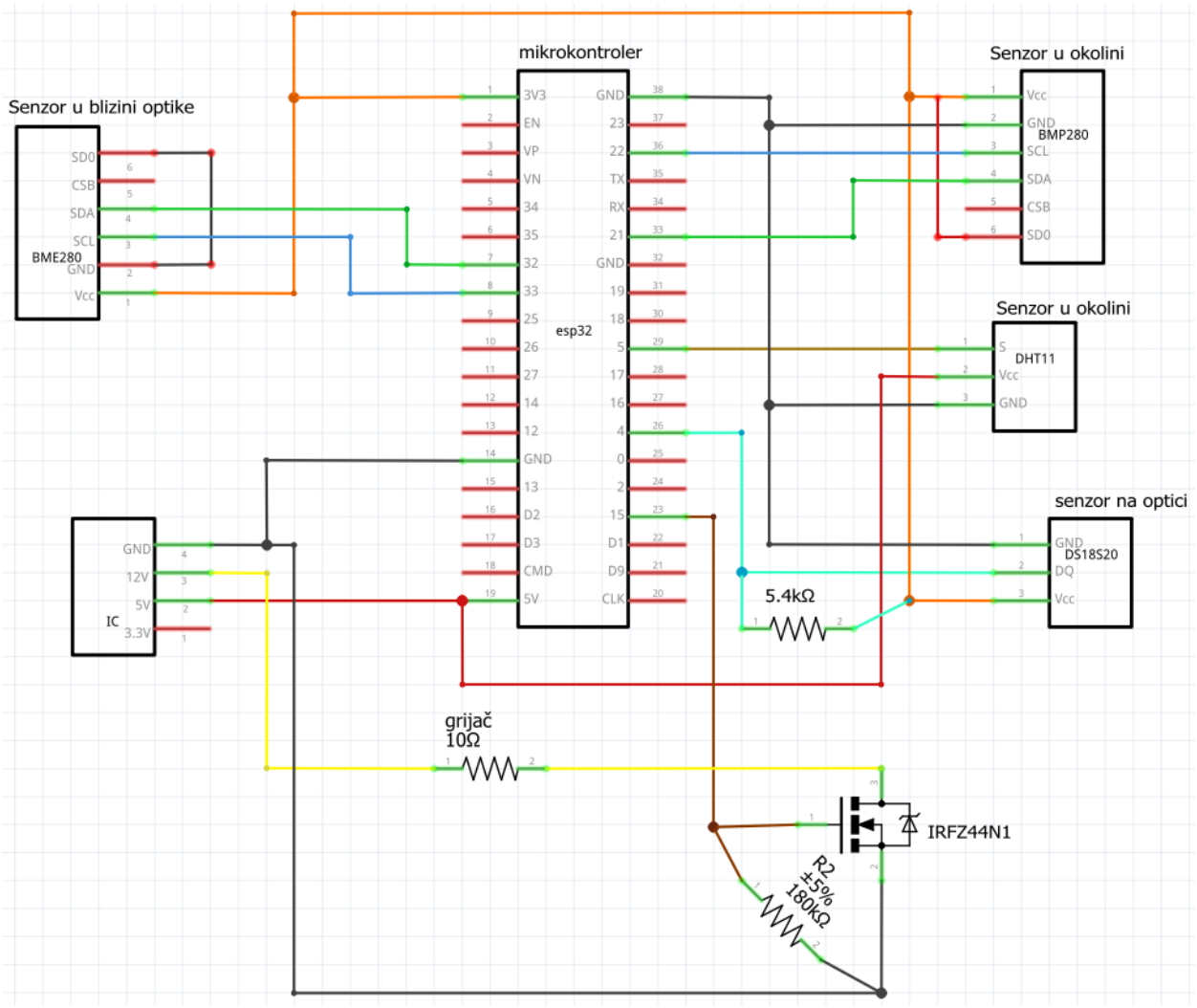
Nakon računanja utvrđeno je da MOSFET - u nije potrebno dodatno hlađenje.

Kako MOSFET već pri vrlo malim naponima na upravljačkoj elektrodi propušta struju, može se dogoditi da nakon što korisnik ugasi grijač, mala količina napona ostane na upravljačkoj elektrodi te i dalje propušta struju. To je spriječeno dodavanjem otpornika između pina 15 i uzemljenja. Shema spoja može se vidjeti na slici 3.2.

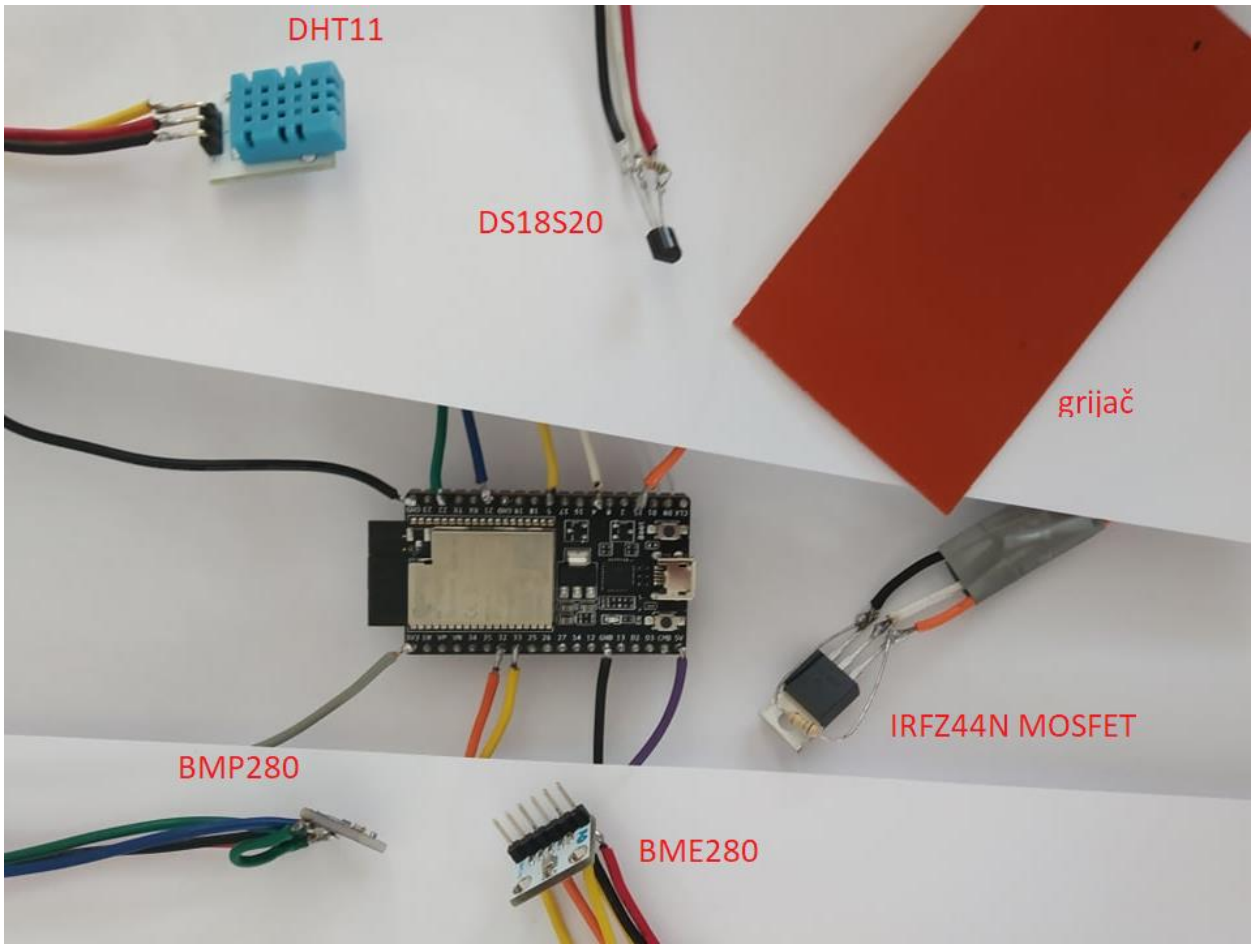


Slika 3.2. Shema spoja grijača s mikroupravljačem

Nakon što je sve izračunato i provjereno, svi dijelovi zalemljeni su prema prikazanoj shemi na slici 3.3. te se može vidjeti dovršen sustav na slici 3.4.



Slika 3.3. Shema cijelog sustava

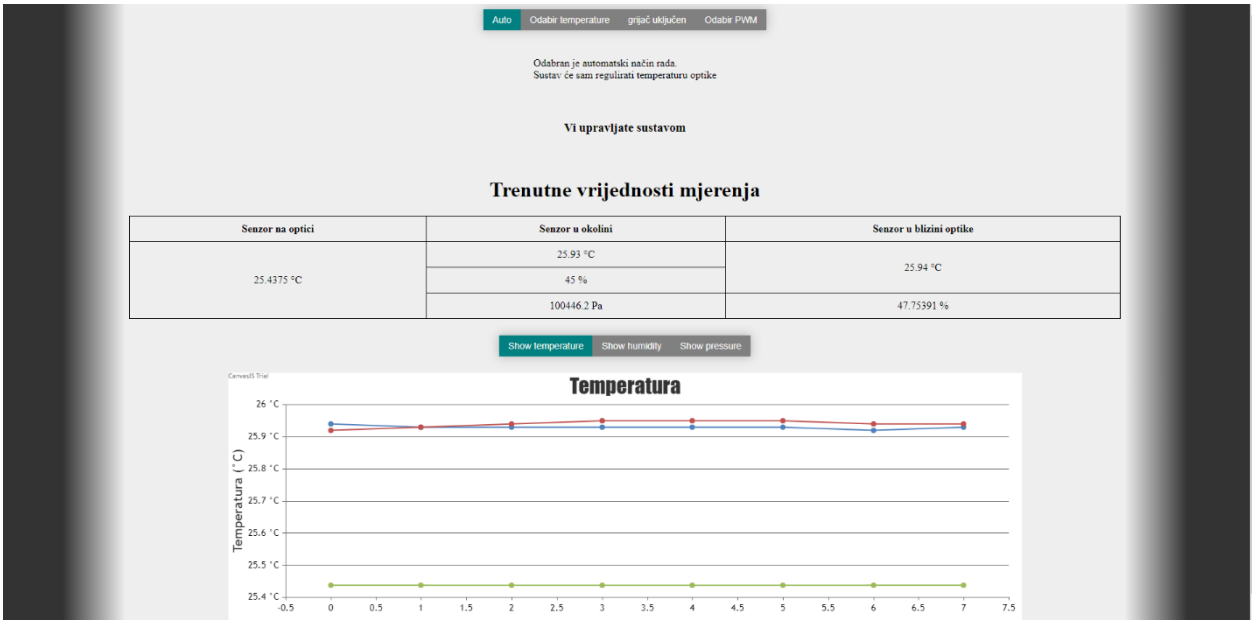


Slika 3.4. Dovršen sustav za prevenciju zamagljivanja teleskopske optike

3.3. Realizacija programskog rješenja

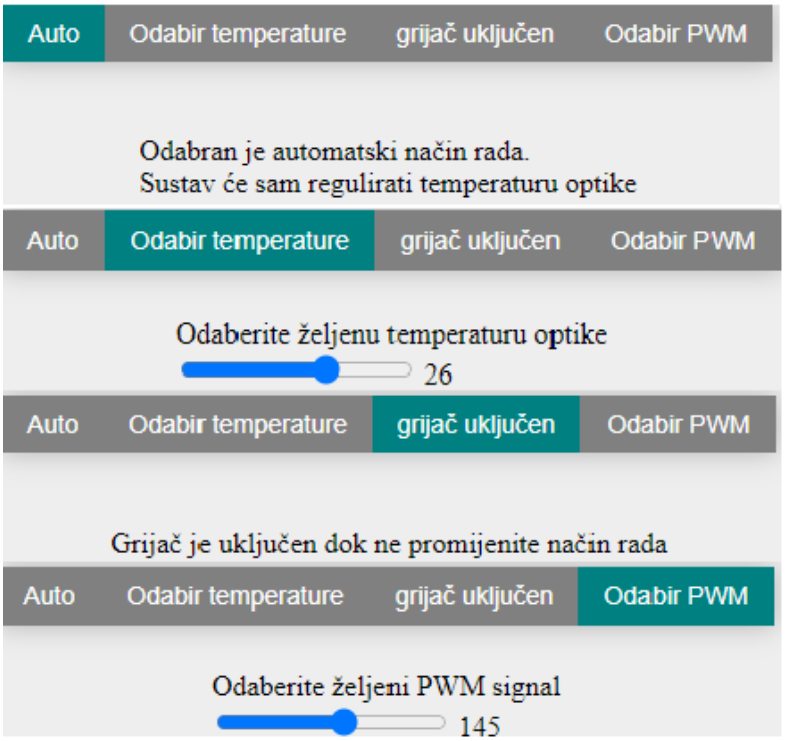
Sustav se sastoji od ESP32 mikroupravljača i web stranice koju otvara klijent na svome uređaju. Mikroupravljač radi kao pristupna točka i nije ovisan o drugim mrežama. Na njemu spremljeni su svi dokumenti potrebni za rad sustava, poput koda za web stranicu i potrebnih biblioteka za rad sa senzorima.

Prvo je isprogramirano web sučelje pomoću kojega se mogu pratiti trenutne vrijednosti na senzorima i odabrati način reguliranja temperature teleskopske optike. Na slici 3.5. prikazan je izgled web sučelja koje korisnik otvara na svome uređaju kada se spoji putem Wi - Fi mreže na mikroupravljač. HTML jezikom definirana je struktura, a CSS jezikom elementi su pozicionirani i uređeni. JavaScript jezikom obavlja se komunikacija sa serverom i sve dinamičke promjene na web stranici. Za grafički prikaz podataka korištena je biblioteka CanvasJS [13] koja je preuzeta i spremljena u memoriju mikroupravljača. Preuzeta biblioteka nalazi se u prilogu P.3.6.



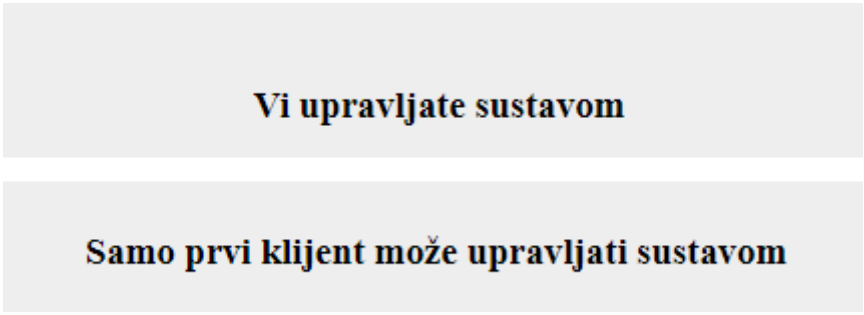
Slika 3.5. Web sučelje

Na samome vrhu web stranice nalazi se izbornik na kojemu klijent bira željeni način rada sustava. Na slici 3.6. prikazane su sve opcije izbornika.



Slika 3.6. Izbornici na vrhu web stranice

Nakon izbornika nalazi se poruka koja klijentu prikazuje može li upravljati sustavom. U slučaju da klijent ne može upravljati sustavom, neće imati izbornik za odabir načina rada koji se može vidjeti na slici 3.6. nego će mu samo biti prikazana druga poruka na kojoj piše da ne može upravljati sustavom. Izgled poruka prikazan je na slici 3.7.



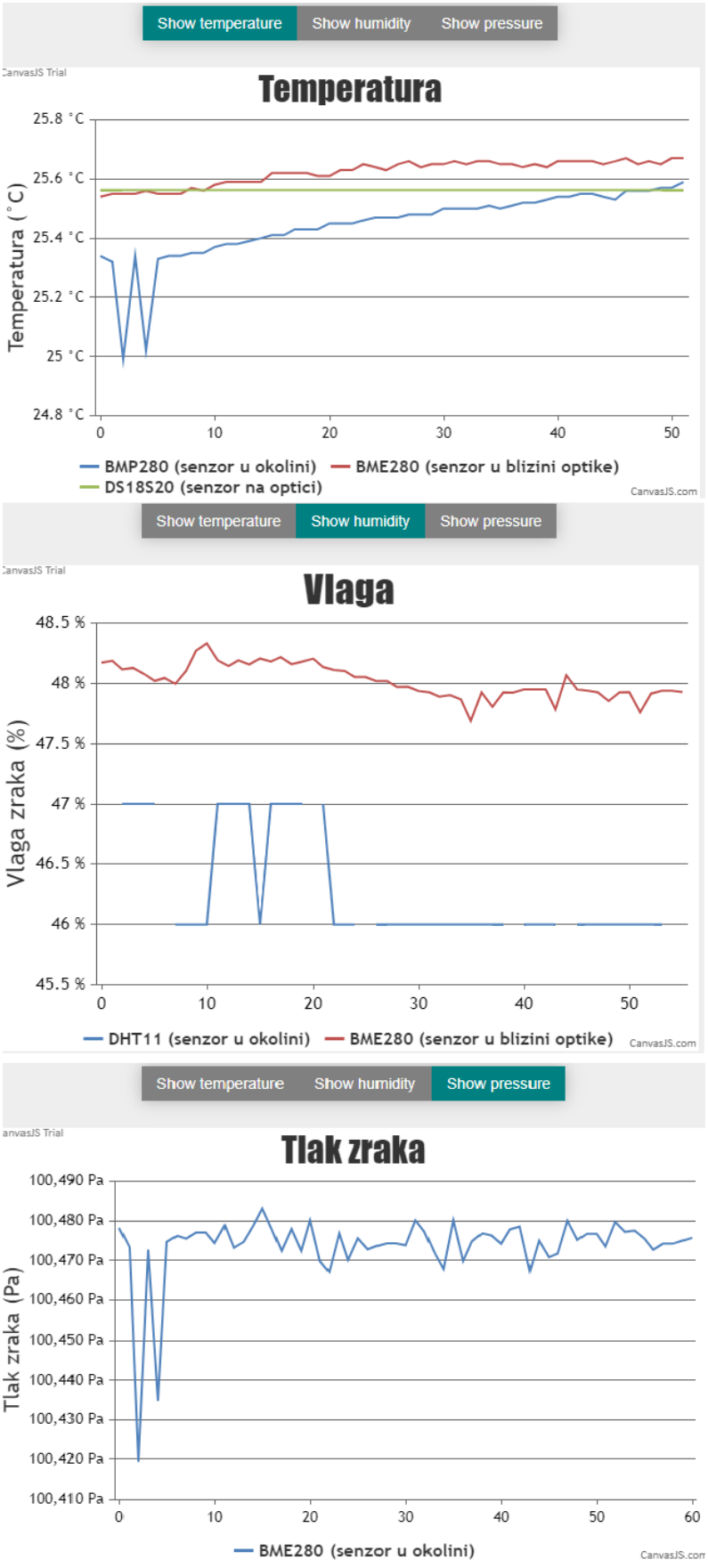
Slika 3.7. Poruke koje ukazuju klijentu može li upravljati sustavom

Ispod poruke nalazi se tablica koja prikazuje trenutne rezultate mjerenja na senzorima. Tablica se može vidjeti na slici 3.8. Nakon tablice slijedi izbornik pomoću kojega klijent bira graf koji želi vidjeti te ispod izbornika prikazan je odabrani graf. Grafovi su napravljeni pomoću preuzete biblioteke CanvasJS. Podijeljeni su tako da prvi prikazuje temperature, drugi vlagu, a treći tlak zraka. Izgled svih grafova prikazan je na slici 3.9.

Trenutne vrijednosti mjerenja

Senzor na optici	Senzor u okolini	Senzor u blizini optike
25.5625 °C	25.64 °C	25.67 °C
	46 %	
	100480.9 Pa	47.92676 %

Slika 3.8. Tablica s trenutnim vrijednostima



Slika 3.9. Prikaz izbornika za grafove i pripadajućih grafova

Na slici 3.10. prikazan je blok dijagram algoritma web sučelja. Kada klijent dobije sve datoteke potrebne za funkcioniranje web stranice, pokreće se funkcija za učitavanje varijabli i početni prikaz. Poslije toga funkcija za početni prikaz poziva funkciju za uspostavu WebSocket komunikacije.

WebSocket dvosmjeran je komunikacijski protokol kojega klijent treba zatražiti od servera [14]. Pojednostavljuje komunikaciju zato što nakon uspostave komunikacijskog kanala više nisu potrebne HTTP _ GET i HTTP _ POST metode koje se obično koriste za komunikaciju između servera i klijenta. Odabrana je WebSocket komunikacija zato što je nakon uspostave lakše slati poruke i zahtijeva manje resursa od servera. Nakon što se uspostavi WebSocket komunikacija, definiraju se funkcije koje će se odvijati kao reakcija na otvorenu WebSocket vezu, na zatvaranje iste, na grešku te najbitnije na primljenu poruku.

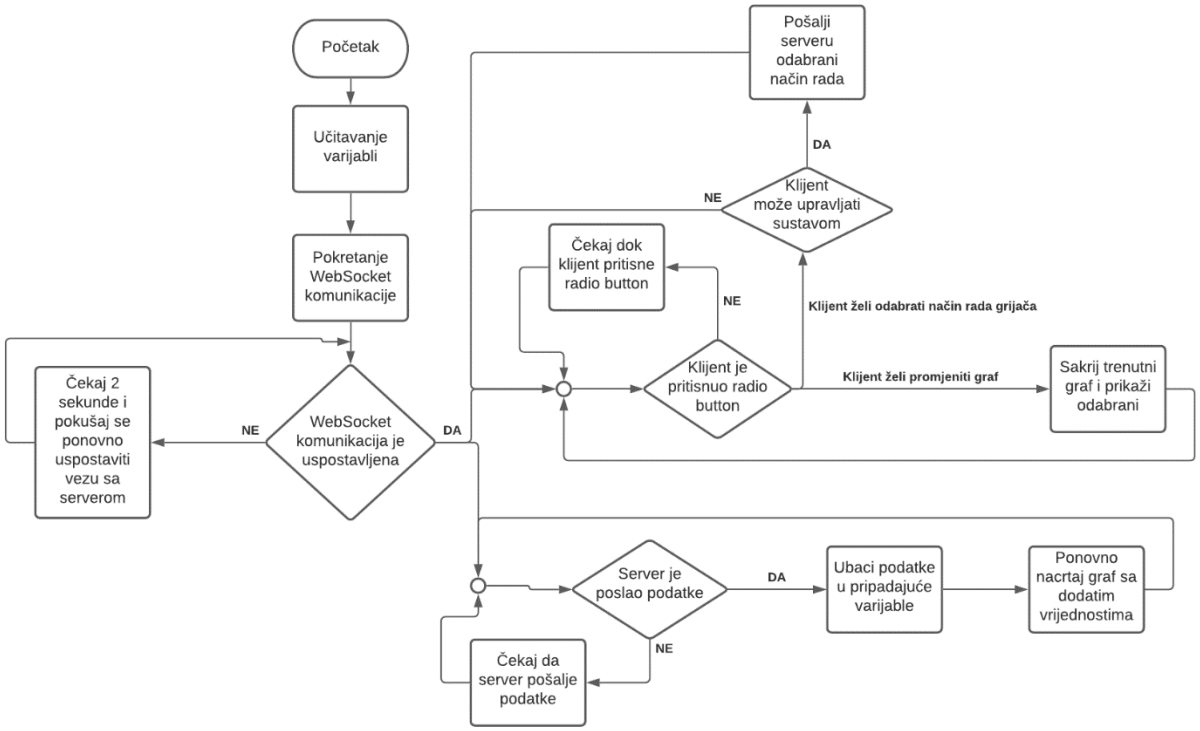
Kada je veza otvorena, server šalje klijentu njegov broj koji određuje može li spojeni klijent upravljati načinom rada grijača. Ako se veza prekine, klijent pričeka dvije sekunde te pokuša ponovno uspostaviti vezu sa serverom.

Server šalje klijentu JSON objekt u tekstualnom obliku. Kada klijent dobije poruku, postavlja podatke u pripadajuće varijable. Pristigle vrijednosti ubacuju se u tablicu gdje su prikazane trenutne vrijednosti na senzorima te se ponovno crtaju grafovi kako bi nove vrijednosti bile vidljive.

U slučaju da je klijentu dopušteno mijenjanje načina rada grijača, to mu se prikazuje na web stranici sustava i nudi mogućnost odabira. Nakon što klijent odabere željeni način rada, šalje se poruka serveru.

Kako bi klijent došao na web stranicu sustava nakon prijave na Wi - Fi, treba otići na web preglednik te upisati adresu 192.168.4.1.

Cijeli HTML kod prikazan je u prilogu P.3.2., njegov CSS kod u prilogu P.3.3. i JavaScript kod u prilogu P.3.4.



Slika 3.10. Algoritam web sučelja

Algoritam mikroupravljača može se vidjeti na slici 3.11. Kada se mikroupravljač uključi, prvo provjerava može li pristupiti SPIFFS – u, odnosno svom serijskom datotečnom sustavu u kojemu su pohranjene datoteke za web stranicu. Nakon toga uspostavlja komunikaciju sa senzorima. Budući da se koristi nekoliko različitih senzora, za svaki potrebna je njegova biblioteka. Sljedeći korak inicijalizacija je Wi – Fi - ja. Potrebno mu je postaviti ime, lozinku i definirati način rada. Ime Wi - Fi mreže je „ESP32sustav“, lozinka je „lozinka123“, a način rada postavljen je kao pristupna točka.

Poslije toga definira se nekoliko zadataka pomoću FreeRTOS [15], čime se postiže da obje jezgre ESP32 mikroupravljača rade istovremeno te se nad njima ima kontrola. Jedan zadatak dodan je jezgri 0. Zadatak je prikupljanje podataka sa senzora i slanje istih klijentu svakih pet sekundi. Nakon njega definirana su još dva zadatka. Jedan je za automatsko reguliranje temperature teleskopske optike, a drugi za reguliranje temperature teleskopske optike prema temperaturi koju

je korisnik odredio. Ta dva zadatka odmah nakon definiranja zaustavljaju se te se pokreću kada korisnik to zatraži.

Nakon što se klijent spoji i posluži mu se web stranica, server čeka da klijent odabere način rada sustava od četiriju ponuđenih. Kada klijent odabere način rada sustava, server zaustavlja trenutnu regulaciju te kreće s novom.

Prvi način rada je da sustav automatski regulira treba li se grijač uključiti ili ne. U slučaju da je razlika između temperature okoline i temperature u blizini teleskopske optike manja od jednog stupnja i da se temperatura okoline nije povećala za više od dva stupnja, temperatura optike regulira se prema temperaturi u blizini optike koju mjeri BME280 senzor. Ako taj uvjet nije zadovoljen, temperatura teleskopske optike regulira se prema temperaturi okoline koju mjeri BMP280 senzor.

Drugi način rada je da klijent odabere željenu temperaturu optike te je grijač uključen dok teleskopska optika ne dosegne željenu temperaturu.

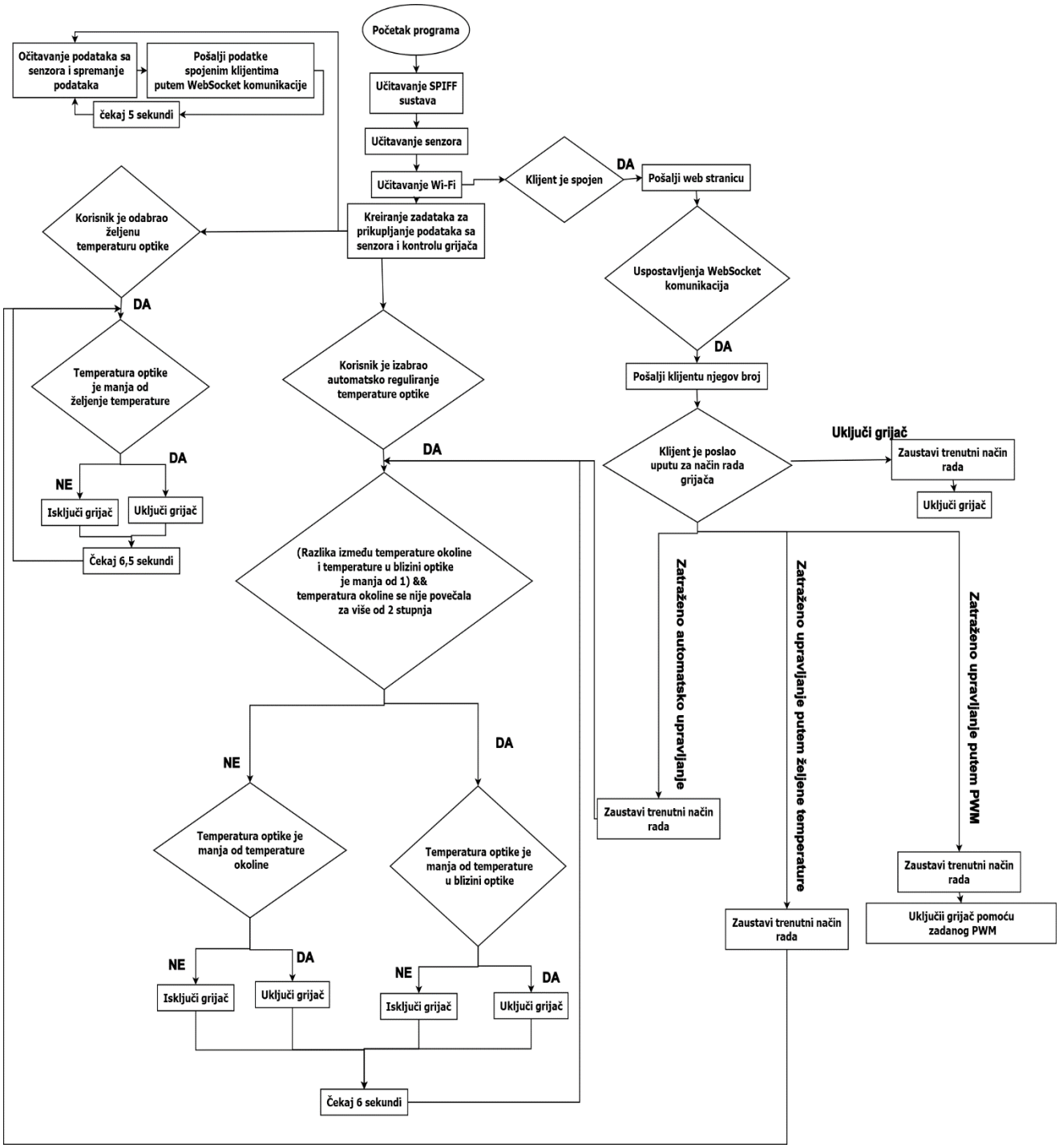
Treći način rada je da grijač bude bezuvjetno uključen, što znači da grijač treba raditi bez obzira na izmjerene vrijednosti temperature na postavljenim mjestima.

Četvrti način rada je da grijač bezuvjetno bude uključen, ali određene snage koju je klijent odabrao pomoću PWM, odnosno modulacije trajanja signala. Napon na upravljačkoj elektrodi ovisi o radnom ciklusu kojega korisnik određuje. Radni ciklus može biti postavljen od 0 do 255. Kada je radni ciklus signala 0, grijač ne radi, odnosno napon na upravljačkoj elektrodi je 0. Kada je radni ciklus 255, na upravljačkoj elektrodi MOSFET - a nalazi se 3.3 V. Toliki napon na upravljačkoj elektrodi IRFZ44N MOSFET - a nije potreban zato što već pri naponu od 2 V, svih 12 V sa napajanja prolazi kroz grijač i on radi punom snagom. Upotrebom drugačijeg modela MOSFET - a mijenja se potreban napon za upravljanje.

Napon od 2 V postiže se kada je radni ciklus 155.

Prva tri načina rada koriste PWM s radnim ciklusom od 133, dok u četvrtom korisnik određuje snagu grijača.

Cijeli kod mikroupravljača prikazan je u prilogu P.3.5.



Slika 3.11. Algoritam mikroupravljača

4. TESTIRANJE I REZULTATI

4.1. Metodologija testiranja

Za promatranje rezultata testiranja pratit će se graf na web stranici. Izvesti će se tri pokusa. Prvi pokus bit će sa staklenom površinom trenutne temperature okoline. Drugi pokus bit će sa staklenom površinom malo hladnijom od trenutne temperature okoline. Treći pokus bit će s dosta hladnijom staklenom površinom u odnosu na trenutnu temperaturu okoline.

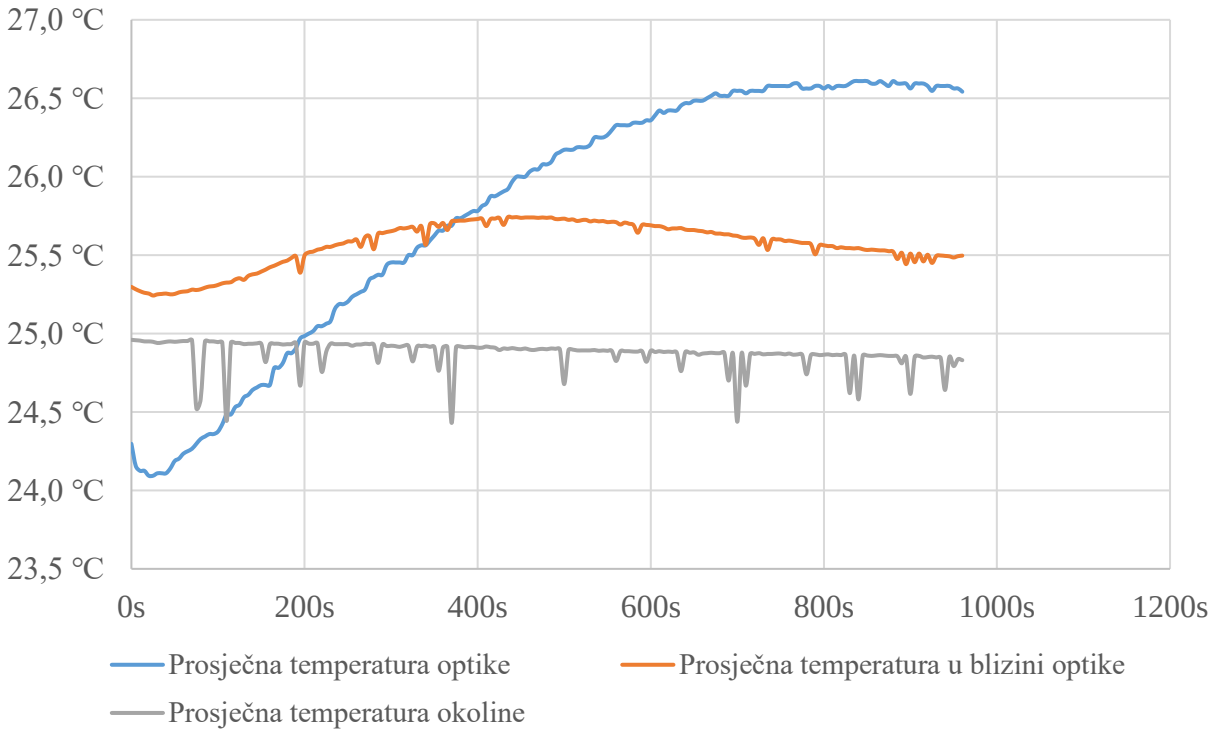
Testiranje će biti obavljeno tako da sustav sam regulira potrebu paljenja grijača te spriječi kondenzaciju na staklenoj površini. Svaki pokus bit će proveden više puta te će se grafički prikazati rezultati. Pratit će se vrijeme potrebno da optika dosegne temperaturu okoline i pojavljuje li se kondenzacija na staklenoj površini.

Očekivani rezultat je da će kod prvoga pokusa temperatura staklene površine vrlo brzo doseći temperaturu okoline te do kondenzacije neće doći. Kod drugog pokusa, koji je najrealniji scenarij, moguće je da se na početku počne kondenzirati vlaga na staklenoj površini, ali vrlo brzo bi se optika trebala ugrijati uključivanjem grijača i otkloniti problem. Kod trećeg pokusa gotovo sigurno kondenzirati će se vlaga na staklenoj površini, ali pratit će se koliko je vremena potrebno da se staklena površina zagrije na temperaturu okoline.

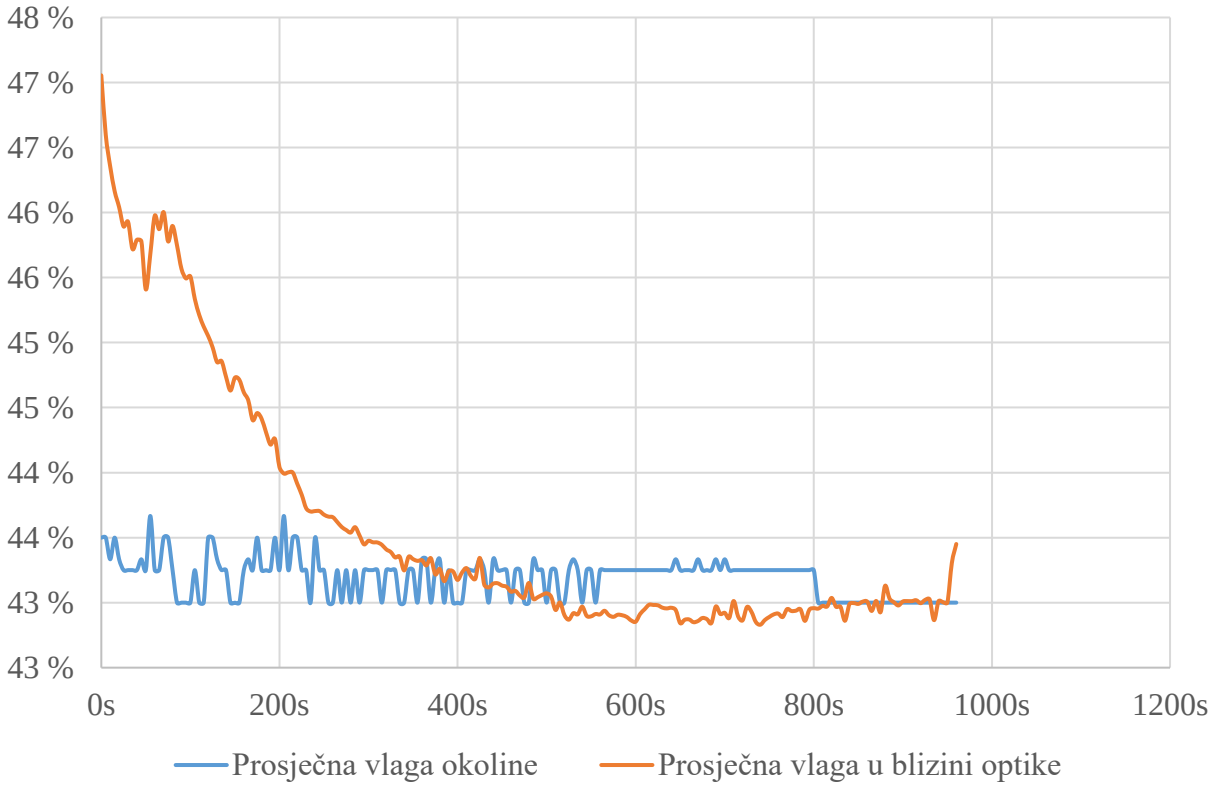
4.2. Rezultati testiranja

4.2.1. Optika pri temperaturi približnoj temperaturi okoline

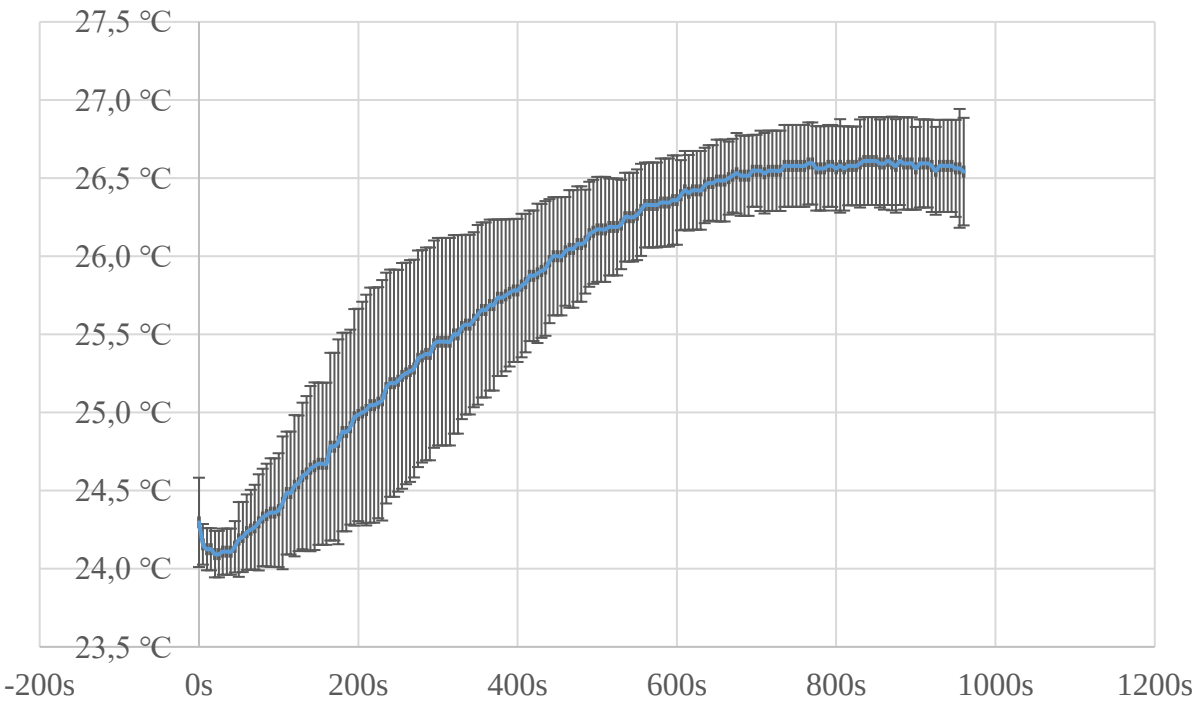
Prvi pokus prikazuje slučaj kada je teleskopska optika blizu temperature okoline. Na grafu 4.2. može se vidjeti da je temperatura optike dosegla temperaturu okoline u 200 sekundi. Na drugom grafu 4.1. prikazano je kako relativna vlaga sa zagrijavanjem teleskopske optike pada. Nakon 400 sekundi teleskopska optika toplija je od temperature okoline i temperature zraka u blizini optike, a vlaga u blizini optike manja je od vlage u okolini.



Slika 4.2. Prosječne temperature



Slika 4.1. Prosječna vlaga

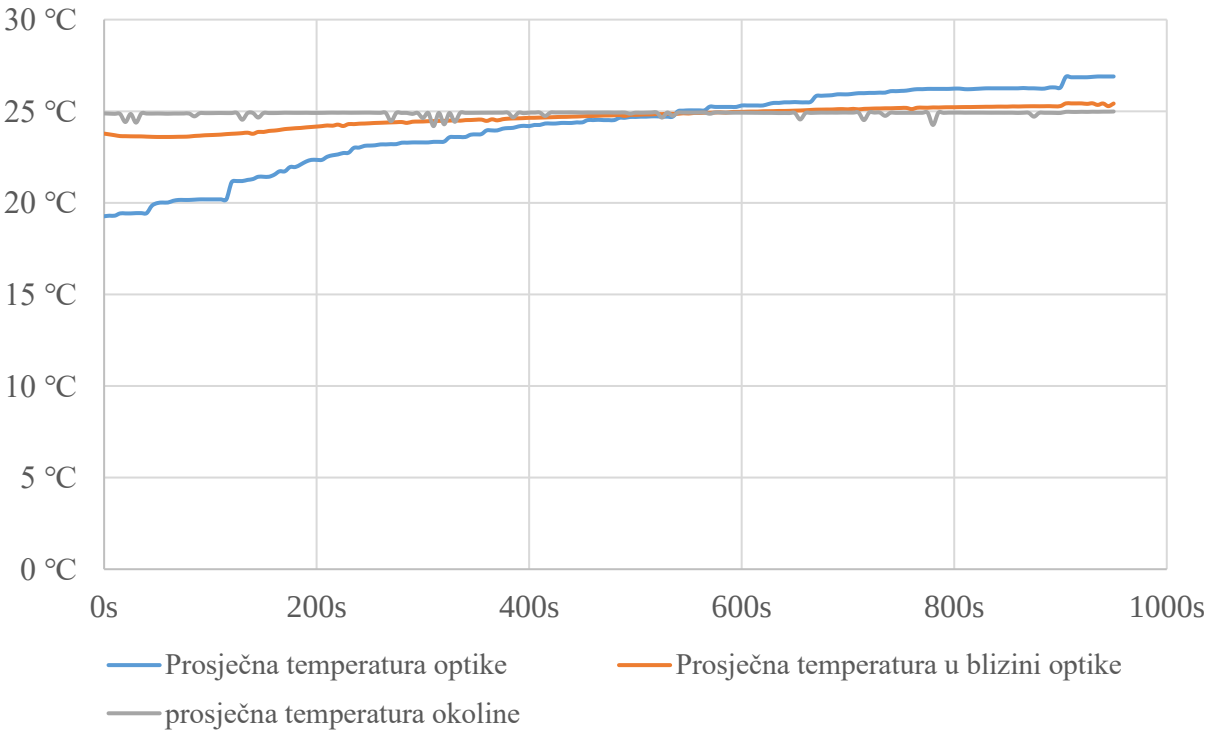


Slika 4.3. Varijacija temperature optike

Temperatura u blizini optike veća je od temperature okoline zato što se tijekom uzastopnih testova okolina zagrijala te se na grafu 4.3. može vidjeti kako potrebno vrijeme za zagrijavanje teleskopske optike može varirati ovisno o početnoj temperaturi optike i temperaturi okoline.

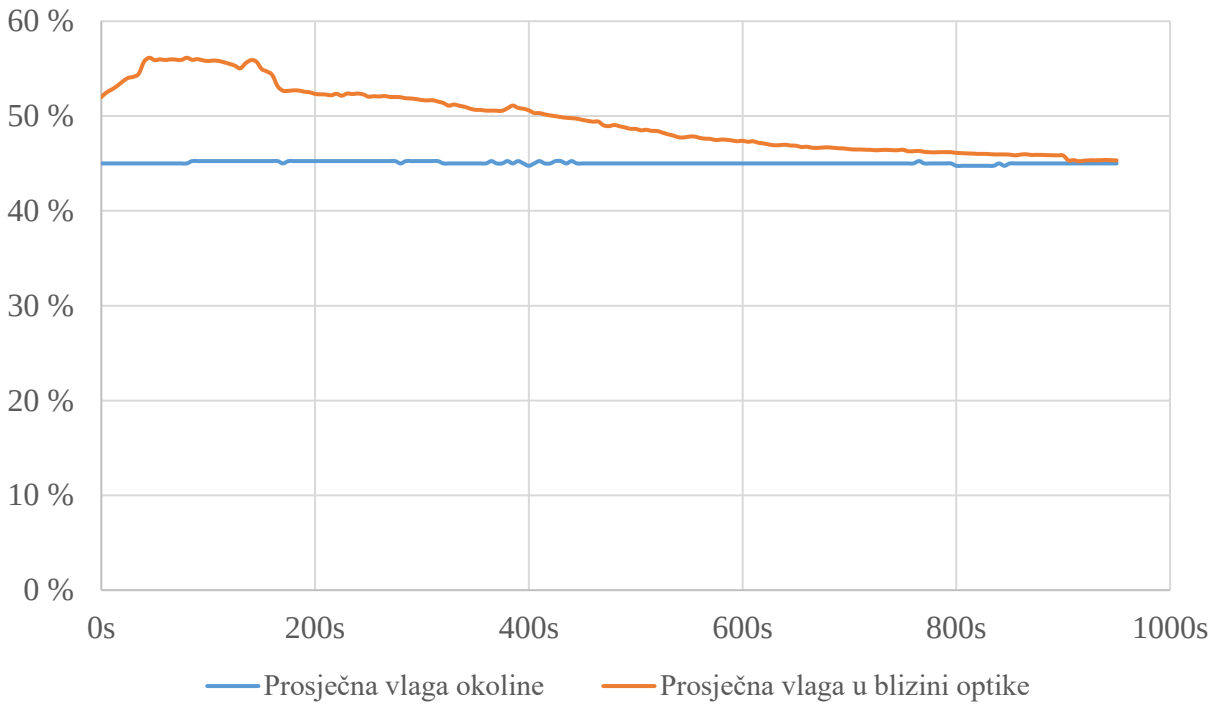
4.2.2. Realni scenarij

U ovome pokusu optika je bila 6 - 7 ℃ manja od okoline. Može se vidjeti na grafu 4.4. da je u ovome pokusu bilo potrebno 500 sekundi da temperatura optike dosegne temperaturu okoline.

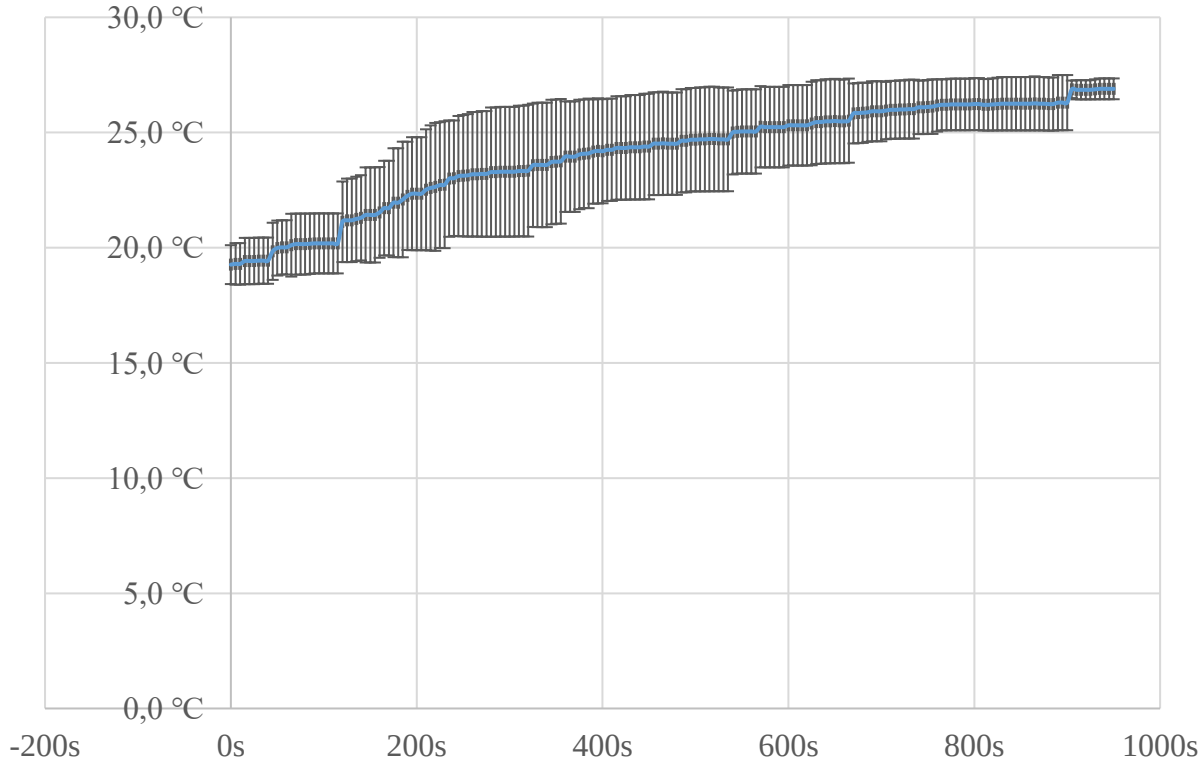


Slika 4.4. Prosječne temperature

Kako bi temperatura bila 6 - 7 ℃ manja od okoline, optika je neko vrijeme bila u hladnjaku te je na početku testiranja bilo malo kapljica na optici. Na grafu 4.5. možemo vidjeti da relativna vlaga prvo raste te kada kapljice na optici ispare, relativna vlaga u zraku počinje padati.



Slika 4.5. Prosječna vlaga

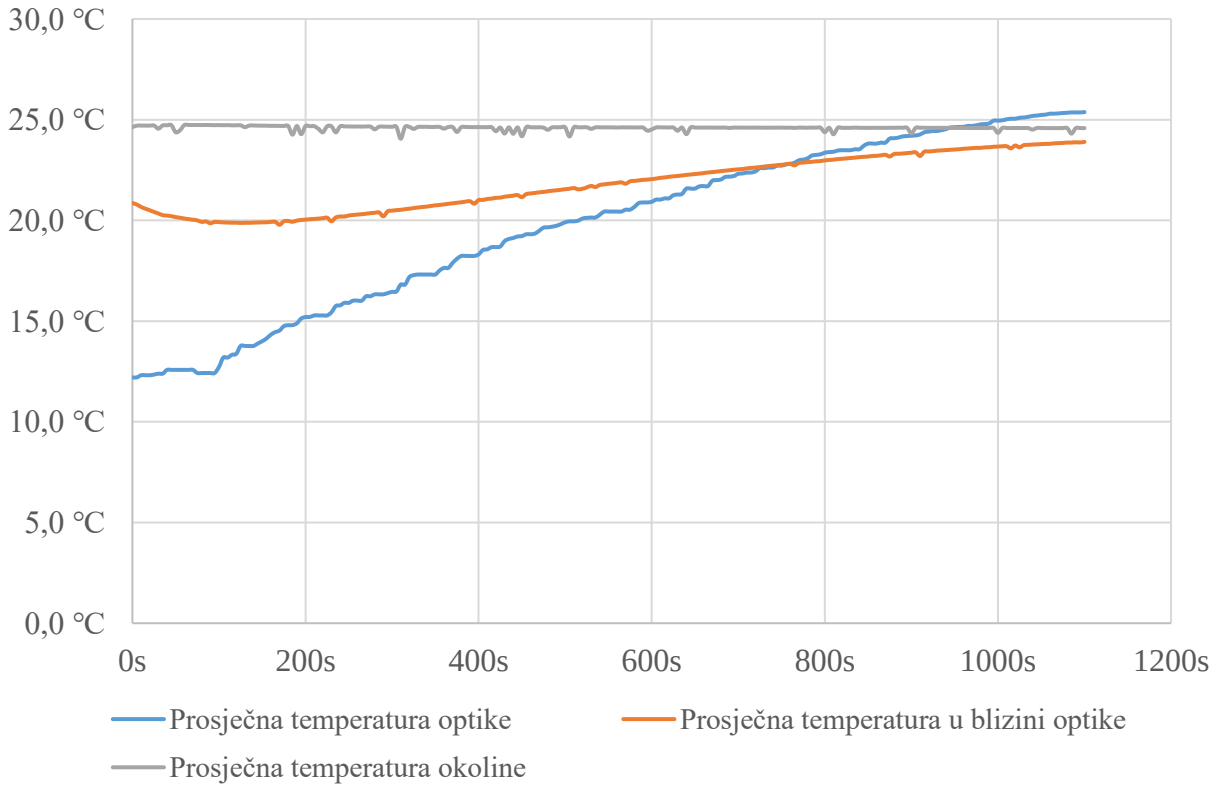


Slika 4.6. Varijacija temperature optike

Na grafu 4.6. prikazano je variranje temperature optike koje ovisi o početnoj temperaturi optike i temperaturi zraka u blizini optike.

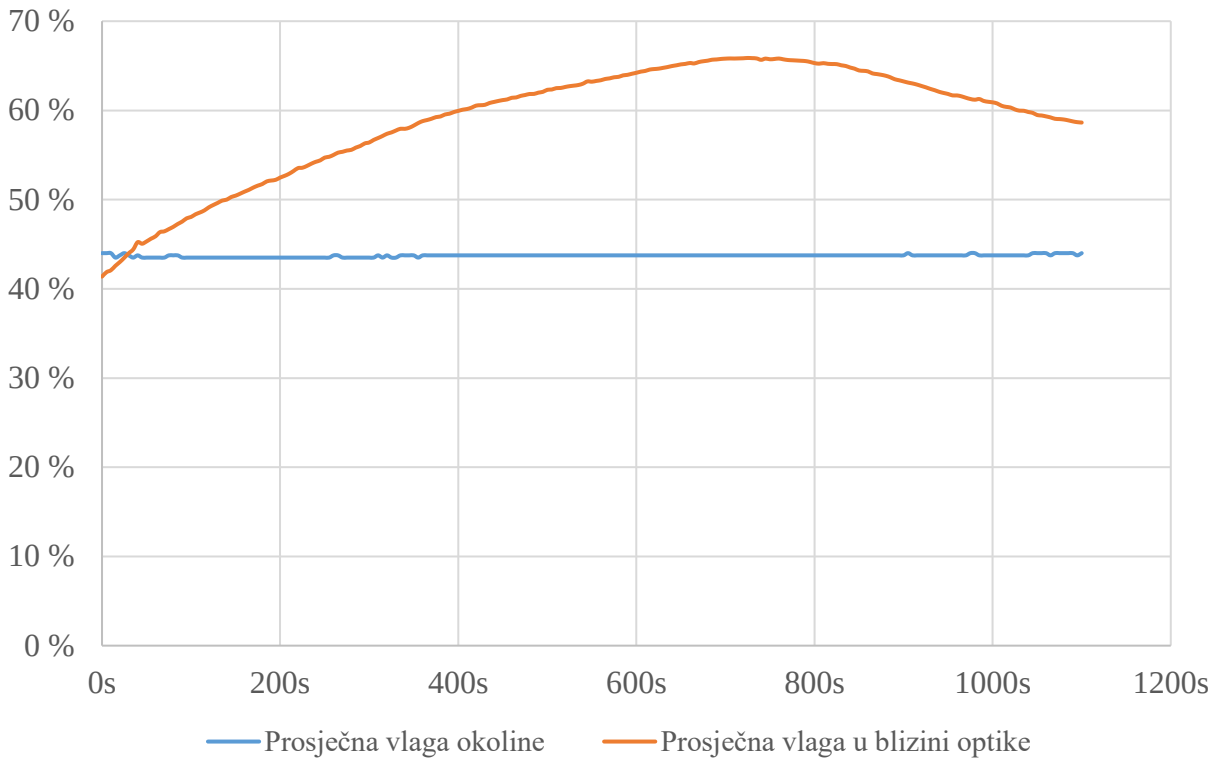
4.2.3. Veće odstupanje od temperature okoline

U ovome pokusu početna temperatura optike bila je 12 ℃ te je razlika između okoline i teleskopske optike gotovo 14 ℃. Može se vidjeti na grafu 4.7. da je teleskopskoj optici bilo potrebno 750 sekundi da postigne temperaturu u blizini optike te gotovo 200 sekundi više da dostigne temperaturu okoline.



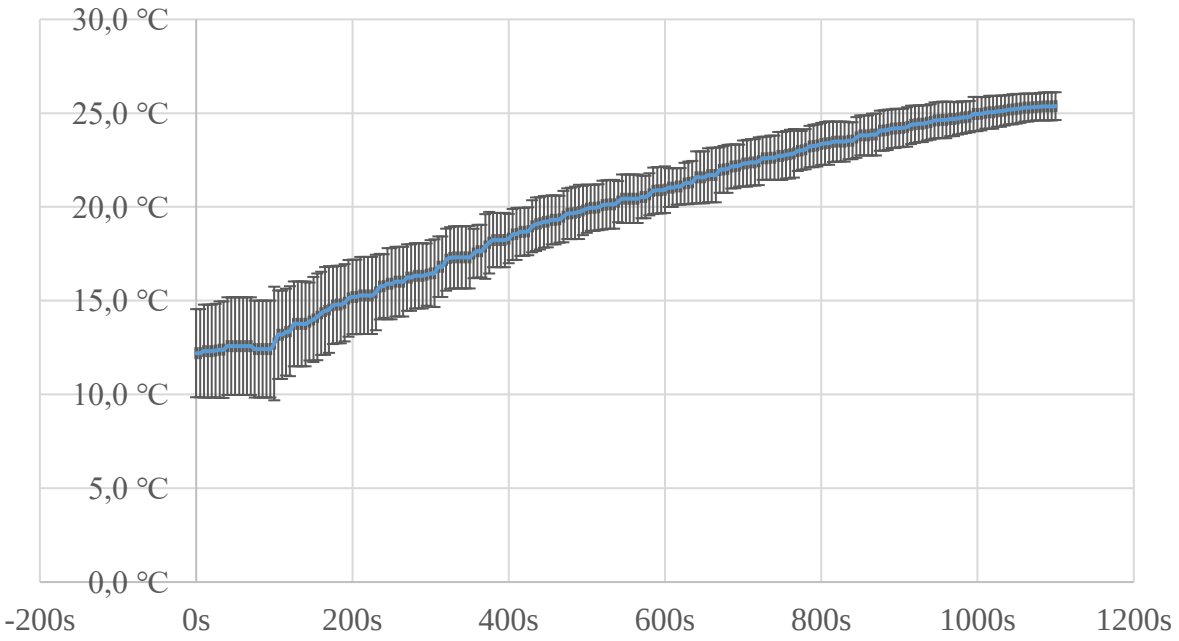
Slika 4.7. Prosječne temperature

Kako je za ovaj pokus bilo potrebno ohladiti optiku u zamrzivaču, odmah pri početku testa bilo je kondenziranih kapljica na površini koje su vremenom isparile. Na grafu 4.9. može se vidjeti da je relativna vlaga rasla dok kondenzirana vlaga nije isparila te nakon toga vidi se pad relativne vlage. Ista pojava mogla se vidjeti u prošlom pokusu.



Slika 4.9. Prosječna vlaga

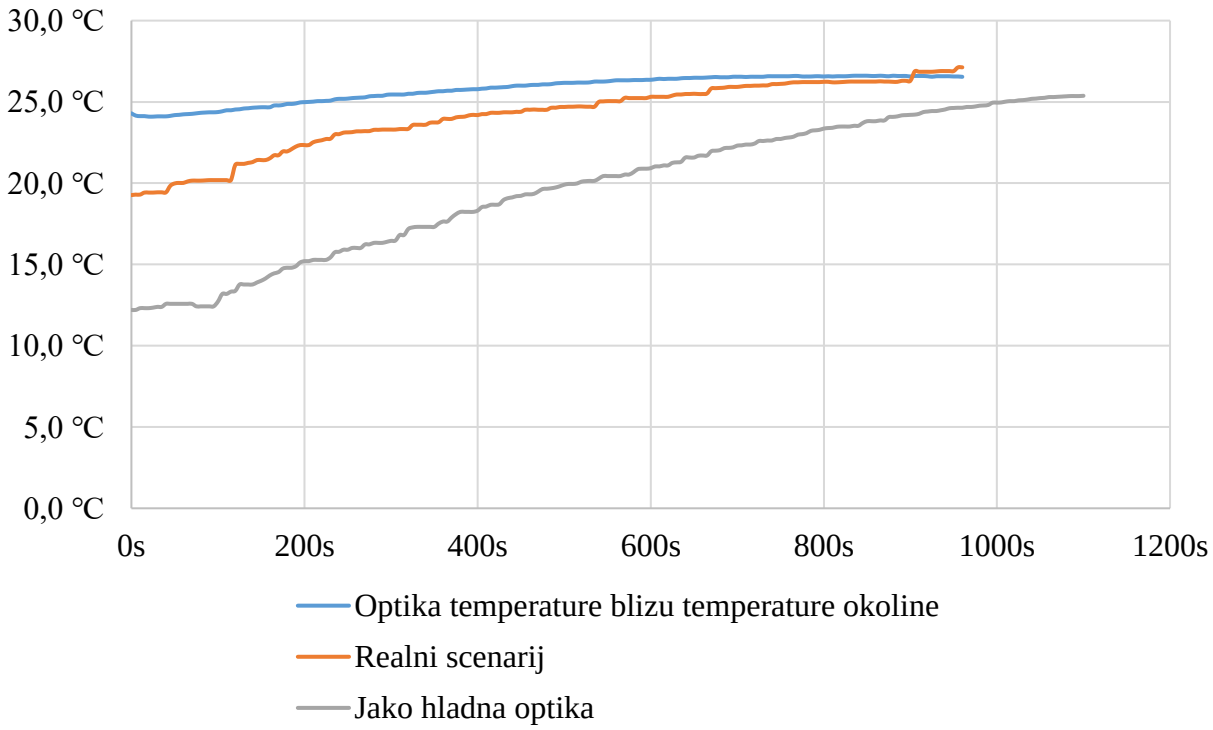
Na grafu 4.8. prikazana je varijacija temperature optike. Može se vidjeti da je variranje puno manje nego u prvim dvama pokusima. Zaključuje da su varijacije manje zato što su razlike u početnim temperaturama optike bile manje. To je postignuto tako što je optika bila u zamrzivaču dok nije postigla najnižu moguću temperaturu.



Slika 4.8. Varijacija temperature optike

4.2.4. Usporedba rezultata

Na slici 4.10. može se vidjeti koliko je prosječno vremena bilo potrebno da se optika ugrije. U svim trima testovima temperatura okoline bila je na 25 °C. Može se zaključiti da što je hladnija optika, to će joj duže trebati da se ugrije. Kada se optika krene grijati, može se primijetiti da se hladnija optika naglije zagrijava. Na početku drugog i trećeg testa kondenzirala se vlaga na površini optike, no s rastom temperature počela je isparavati. Kada je temperatura optike dosegla 19 °C, ni u jednom pokusu više nije bilo kapljica na površini.



Slika 4.10. Prosječne temperature optike u sva tri pokusa

5. ZAKLJUČAK

U završnom radu napravljen je sustav za zagrijavanje teleskopske optike koji se sastoji od ESP32 mikroupravljača, senzora, grijača i pripadajućih algoritama. Testiranjem se pokazalo kako sustav zagrijavanjem teleskopske optike uspješno uklanja kondenziranu vlagu. Iako pri manjim razlikama u temperaturi nije potrebno zagrijavanje, upotrebom ovakvog sustava u potpunosti uklanja se mogućnost kondenziranja vodene pare na teleskopskoj optici.

Uvidom u grafove rezultata testiranja može se primijetiti da temperatura optike nastavlja rasti i nakon postizanja željene temperature. To se događa zato što je grijač puno veće temperature nego zrak i nakon što se grijač isključi potrebno je više vremena da se ohladi. Tijekom tog vremena grijač nastavlja grijati optiku te se može zaključiti da optika bude 1 - 2 °C toplija nego temperatura okoline.

LITERATURA

[1] Kondenzacija, Hrvatska enciklopedija, mrežno izdanje, Leksikografski zavod Miroslav Krleža, 2021., https://www.enciklopedija.hr/natuknica.aspx?id=32731, pristup: 01.09.2021.

[2] Rosište, Hrvatska enciklopedija, mrežno izdanje, Leksikografski zavod Miroslav Krleža, 2021., https://www.enciklopedija.hr/Natuknica.aspx?ID=53391, pristup: 01.09.2021.

[3] Alan Macrobert, Dealing with dew: dew heaters, dew shields and more, 15.10.2008., https://skyandtelescope.org/astronomy-equipment/equipment-diy/dealing-with-dew/, pristup: 14.09.2021.

[4] Teleskopi za početnike, https://www.astroucionica.hr/teleskopi-za-pocetnike-sve-sto-trebate-znati/#Katadiopterski_teleskopi, pristup: 15.09.2021.

[5] How do telescopes work, https://spaceplace.nasa.gov/telescopes/en/, pristup: 15.09.2021.

[6] Adam Block, How do professional observatories deal with dew?, 27.12.2016., https://astronomy.com/magazine/ask-astro/2016/12/observatories-deal-with-dew, pristup: 14.09.2021.

[7] Alister Ling, Dew: what to do about it, The Royal Astronomical Society of Canada, http://adsabs.harvard.edu/pdf/1986JRASC..80L...9L, pristup: 15.09.2021.

[8] Peter Danek, Why Are Observatories Built On Mountain Tops, https://telescopeguides.com/why-are-observatories-built-on-mountain-tops/, pristup: 15.09.2021.

[9] Jackson Landers, The Coldest, Driest, Most Remote Place on Earth Is the Best Place to Build a Radio Telescope, https://www.smithsonianmag.com/smithsonian-institution/coldest-driest-place-earth-best-place-radio-telescope-180961495/, pristup: 15.09.2021.

[10] Discoveries - Why a Space Telescope?, https://www.nasa.gov/content/discoveries-why-a-space-telescope, pristup: 15.09.2021.

[11] What is the Hubble Space Telescope?, https://www.nasa.gov/audience/forstudents/k-4/stories/nasa-knows/what-is-the-hubble-space-telecope-k4.html, pristup: 1.09.2021.

[12] IRFZ44NPbF Product Data Sheet, https://www.infineon.com/dgdl/irfz44npbf.pdf?fileId=5546d462533600a40153563b3a9f220d, pristup: 04.09.2021.

[13] JavaScript Live / Dynamic Charts & Graphs, https://canvasjs.com/html5-javascriptdynamic-chart/, pristup: 09.09.2021.

[14] The WebSocket API,

https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API, pristup: 08.09.2021.

[15] FreeRTOS - ESP32, https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html, pristup: 06.09.2021.

SAŽETAK

Cilj rada bio je objasniti problem zamagljivanja na teleskopskoj optici te dizajnirati i napraviti sustav koji rješava problem. U radu je prikazano kako pomoću ESP32 mikroupravljača napraviti sustav koji senzorima prikuplja potrebne informacije pomoću kojih na nekoliko načina može regulirati temperaturu teleskopske optike i spriječiti zamagljivanje na njenoj površini. Na kraju rada prikazani su rezultati testiranja sustava iz kojih se može vidjeti ponašanje teleskopske optike i okoline pri različitim temperaturama.

Ključne riječi: ESP32, kondenzacija, mikroupravljač, teleskopska optika

ABSTRACT

A system for dew control on telescope optics

The aim of this project was to design and create a system for dew control on telescope optics. This paper shows how to make a system with ESP32 microcontroller which collects required information with sensors and regulates temperature of telescope optics to prevent condensation of water vapor on its surface. At the end of the paper there are test results of system that shows how telescope optics and its surrounding behave at different temperatures.

Keywords: condensation, ESP32, microcontroller, telescope optics

PRILOZI I DODACI

Prilog P.3.1.

ESP32 DevKitC - VE mikroupravljač je korišten za izradu ovog sustava. Može koristiti Wi - Fi i Bluetooth te ima ugrađenu Wi - Fi antenu. Sadrži 39 pinova od kojih su 34 ulazno - izlazni pinovi.

Tehničke karakteristike:

- Radni napon 3.3V
- Maksimalna frekvencija 240MHz
- SRAM – 520 KB
- Podržani protokoli za komunikaciju: SPI, I2C, I2S, CAN, UART
- Brza memorija 8MB

Prilog P.3.2.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Grijač</title>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Document</title>
</head>
<body>


    <!--Odabir načina rada-->
    <div class="radio-group mode" id="work_mode">

            <input type="radio" id="automatic_mode" name="work_mode"
            value="auto" checked onclick="workmodechange(this)">
            <label class="radio-button" for="automatic_mode">Auto</label>
            <input type="radio" id="manual_mode" name="work_mode"
            value="manual" onclick="workmodechange(this)">
            <label class="radio-button"
            for="manual_mode">Odabir temperature</label>
            <input type="radio" id="heateron_mode" name="work_mode"
            value="heaterOn" onclick="workmodechange(this)">
            <label class="radio-button"
            for="heateron_mode">grijač uključen</label>
            <input type="radio" id="manual_mode_PWM" name="work_mode"
            value="manualPWM" onclick="workmodechange(this)">
            <label class="radio-button"
            for="manual_mode_PWM">Odabir PWM</label>
    </div>
```

```
<div class="work_mode_display">
    <p id="auto">Odabran je automatski način rada.
    <br>Sustav će sam regulirati temperaturu optike</p>
    <div id="manual">
        <label for="manual_temperature">Odaberite željenu temperaturu
        optike</label><br>
        <input type="range" id="manual_temperature"
        name="manual_temperature" min="0" max="40"
        oninput="this.nextElementSibling.value = this.value"
        onchange="manualValueChanged()">
        <output>24</output>
    </div>
    <p id="heaterOn">Grijač je uključen dok ne promijenite način rada</p>
    <div id="manualPWM">
        <label for="manual_PWM">Odaberite željeni PWM signal</label><br>
        <input type="range" id="manual_PWM" name="manual_PWM" min="0"
        max="255" oninput="this.nextElementSibling.value = this.value"
        onchange="manualValueChangedPWM()">
        <output>130</output>
    </div>

</div>
<h3 id="controller"></h4>
<!--Tablica za prikaz trenutnih vrijednosti sa senzora-->
<table class="rezultati">
    <caption><h1>Trenutne vrijednosti mjerenja</h1></caption>
    <tr>
        <th>Senzor na optici</th>
        <th>Senzor u okolini</th>
        <th>Senzor u blizini optike</th>
    </tr>
    <tr>
        <td rowspan="3" id="DS18S20temp">°C</td>
        <td id="BMP280temp">°C</td>
        <td rowspan="2" id="BME280temp">°C</td>
    </tr>
    <tr>
        <td id="DHT11humidity">%</td>

    </tr>
    <tr>
        <td id="BMP280pressure">Pa</td>
        <td id="BME280humidity">%</td>
    </tr>
</table>

<!--Odabir željenog grafa-->
<div class="radio-group graphSelection">

    <input type="radio" id="selectTemp" name="graphSelect"
    value="temp" checked  onclick="graphSelect(this)">
    <label class="radio-button" for="selectTemp">Show temperature</label>
    <input type="radio" id="selectHum" name="graphSelect"
    value="humidity1" onclick="graphSelect(this)">
    <label class="radio-button" for="selectHum">Show humidity</label>
    <input type="radio" id="selectPressure" name="graphSelect"
      value="pressure" onclick="graphSelect(this)">
    <label class="radio-button"
      for="selectPressure">Show pressure</label>
```

```
    </div>
    <div class="graphContainer">
        <div id="temp"></div>
        <div id="humidity1"></div>
        <div id="pressure"></div>
    </div>


<script src="canvasjs.min.js"></script>
    <script src="script.js"></script>
</body>
</html>
```

Prilog P.3.3.

```
/*Pozadina i pozicioniranje sadržaja na stranici*/
body {
    background: linear-
gradient(to left, #333 5%,#eee 10%,  #eee 90%, #333 95%);
    display: grid;
    row-gap: 25px;
    grid-template-columns: 10% auto 10%;
    grid-template-rows: auto;
    grid-template-areas:
    ". workmode ."
    ". workmodeoptions ."
    ". table ."
    ". graphSelection ."
    ". graf .";
}


/*Izgled izbornika*/
.mode{grid-area: workmode;}
.radio-group{

    display: flex;
    flex-direction: row;
    justify-content: center;
    justify-self: center;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.25);}
.radio-group input{ display: none;}
.radio-group label{
    font-size: 14px;
    padding: 8px 14px;
    font-family: sans-serif;
    color: #ffffff;
    cursor: pointer;
    background-color: grey;

}
.radio-group label:not(:last-of-type){
    border-radius: 1px solid #006B56;
}
.radio-group input:checked + label{
    background-color: teal;
}

/*Uređivanje izbornika za izbor načina rada*/
.work_mode_display{
    grid-area: workmodeoptions;
    justify-self: center;
}
#auto, #manual, #heaterOn, #manualPWM{
    display: none;
}
#controller{display:block;
    grid-area: statusMessage;}


/*Grafovi*/
.graphSelection{grid-area: graphSelection;
```

```
justify-self: center;}
.graphContainer{grid-area: graf;
justify-self: center;
width: 80%;
}
#temp1, #humidity1, #temp2, #humidity2, #temp3{display: none;
    height: 450px;
    }

/*Tablica za prikaz temperature i vlage*/
table {
    border: 1px solid black;
    border-collapse: collapse;
}
h3{text-align: center;}
.rezultati{ grid-area: table; }
tr, th, td {
    border: 1px solid black;
    border-collapse: collapse;
    text-align: center;
    padding: 10px;
}

/*Promjene koje se primjenjuju kada je web stranica uža od 650 piksela*/
@media screen and (max-width: 650px) {
    body{
        grid-template-columns:0% auto 0%!important;
        background:#eee ;

    }
    .radio-group{
        display: flex;
        flex-wrap: wrap;
        flex-direction: row;
        justify-content: center;
        justify-self: center;
        box-shadow: 0 0 15px rgba(0, 0, 0, 0.25);
    }
    .rezultati {
        justify-self: center;
        width: 80%;
    }
    .graphSelection {
        width: 80%;
    }
    .graphContainer {
        width: 90% !important;
        justify-content: center !important;
        justify-self: center !important;
        align-items: center !important;
    }
    #temp1, #humidity1, #temp2, #humidity2, #temp3{
        width: 90% !important;
    }
    #temp1 > div, #humidity1 > div, #temp2 > div, #humidity2 > div, #temp3 >
div {
        width: 90% !important;

    }}
```

Prilog P.3.4.

```
var gateway = `ws://${window.location.hostname}/ws`;
var url="ws://192.168.4.1:1337"
var currentWorkMode;
var currentGraph;
var currentGraphObject;
var webSocket;
//Broj koji server daje klijentu. Samo klijent 0 može mijenjati način rada
//grijača, ostali klijenti samo mogu pratiti vrijednosti sa senzora
var clientNumber;

//Varijable koje spremaju vrijednosti koje server pošalje klijentu
var BMP280temp = [];
var BME280temp=[];
var DS18S20temp=[];
var DHT11humidity=[];
var BME280humidity=[];
var BMP280pressure=[];

//Funkcija koja se izvodi pri učitavanju stranice. Postavlja početni graf i
//izgled stranice
function settingvariables(){

    currentGraph=document.getElementById("temp");
    currentGraph.style.display='block';
    currentGraphObject=chartTemperature;
    wsConnect(url);
    currentWorkMode=document.getElementById("auto");
    currentWorkMode.style.display='block';

}
//Definiranje funkcija koje će se otvoriti pri otvaranju i zatvaranju
//WebSocket konekcije sa serverom, te funkcije kada klijent dobije poruku od
//servera ili se dogodi greška
function wsConnect(url){
  webSocket = new WebSocket(url);
  webSocket.onopen = function(evt) {onOpen(evt) };
  webSocket.onclose = function(evt) {onClose(evt) };
  webSocket.onmessage = function(evt) {onMessage(evt) };
  webSocket.onerror = function(evt) {onError(evt) };
}

function onOpen(evt){
  console.log("Connected");
}

function onClose(evt){
  console.log("Disconnected");
  //Ako se veza zatvori pokušaj se ponovno povezati za 2 sekunde
  setTimeout(function(){wsConnect(url)}, 2000);
}

function onMessage(evt){
  if(evt.data.startsWith('{')){
    putData(evt.data);
  }
  else if(evt.data.startsWith('CL')){
    clientNumber=evt.data.charAt(2);
```

```
    if(clientNumber==0){document.getElementById("controller").innerHTML="Vi u
pravljate sustavom";}
    else{

      document.getElementById("controller").innerHTML="Samo prvi klijent može
 upravljati sustavom";
      currentWorkMode.style.display='none';
      document.getElementById("work_mode").style.display='none';
    }
  }
  else{
    alert(evt.data);
  }
}

function onError(evt){
  console.log("ERROR: "+evt.data);
}

//Funkcija za slanje podataka serveru
function SendToServer(message){
  console.log(message);
  webSocket.send(message);
}

//Funkcija koja pri odabiru načina rada šalje uputu serveru
function workmodechange(radiobutton){
  if(clientNumber==0){
    currentWorkMode.style.display='none';
    currentWorkMode=document.getElementById(radiobutton.value);
    currentWorkMode.style.display='block';
    if(radiobutton.value=="auto"){
      SendToServer("WM0");
    }
    else if(radiobutton.value=="manual"){
      SendToServer("WM1"+document.getElementById("manual_temperature").value)
;
    }
    else if(radiobutton.value=="heaterOn"){
      SendToServer("WM2");
    }
    else if(radiobutton.value=="manualPWM"){
      SendToServer("WM3"+document.getElementById("manual_PWM").value);
    }
  }
}
//Funkcija koja pri promjeni željene temperature šalje serveru novu
//vrijednost
function manualValueChanged(){
  SendToServer("WM1"+document.getElementById("manual_temperature").value);
}
//Funkcija koja pri promjeni vrijednosti PWM signala šalje serveru novu
//vrijednost
function manualValueChangedPWM(){
  SendToServer("WM3"+document.getElementById("manual_PWM").value);
}
//Funkcija koja omogućava promjenu grafa (trenutni graf sakrije dok odabrani
//graf prikaže)
function graphSelect(graph){
  currentGraph.style.display='none';
```

```
  currentGraph=document.getElementById(graph.value);
  currentGraph.style.display='block';
  if(graph.value=='temp'){currentGraphObject=chartTemperature;}
  else if(graph.value=='humidity1'){currentGraphObject=chartHumidity;}
  else if(graph.value=='pressure'){currentGraphObject=chartPressure;}
  currentGraphObject.render();
}


  //Grafovi
  //Graf za temperaturu
  var chartTemperature = new CanvasJS.Chart("temp", {
    zoomEnabled: true,
    title :{
      text: "Temperatura"
    },
    axisY: {
      title: "Temperatura (°C)",
      suffix: " °C"
    },

    legend:{
      cursor: "pointer",
      fontSize: 16,

    },

    data: [{
      name: "BMP280 (senzor u okolini)",
      type: "line",
      dataPoints: BMP280temp,
      showInLegend: true
    },
    {name: "BME280 (senzor u blizini optike)",
      type: "line",
      dataPoints: BME280temp,
      showInLegend: true
    },
    {name: "DS18S20 (senzor na optici)",
      type: "line",
      dataPoints: DS18S20temp,
      showInLegend: true
    }
  ]

  });



//Graf za vlagu
var chartHumidity = new CanvasJS.Chart("humidity1", {
  zoomEnabled: true,
  title :{
    text: "Vlaga"
  },
  axisY: {
    title: "Vlaga zraka (%)",
    suffix: " %"
  },
  legend:{
```

```
    cursor: "pointer",
    fontSize: 16,

  },

  data: [{
    name: "DHT11 (senzor u okolini)",
    type: "line",
    dataPoints: DHT11humidity,
    showInLegend: true
  },
  {name: "BME280 (senzor u blizini optike)",
    type: "line",
    dataPoints: BME280humidity,
    showInLegend: true
  }
]

});

//Graf za tlak zraka
var chartPressure = new CanvasJS.Chart("pressure", {
  zoomEnabled: true,
  title :{
    text: "Tlak zraka"
  },
  axisY: {
    title: "Tlak zraka (Pa)",
    suffix: " Pa"
  },

  legend:{
    cursor: "pointer",
    fontSize: 16,
  },

  data: [{
    name: "BME280 (senzor u okolini)",
    type: "line",
    dataPoints: BMP280pressure,
    showInLegend: true
  }
]

});


  var xVal = 0;
  var yVal1 = 100;
  var yVal2 = 100;
  var yVal3 = 100;
  var updateInterval = 1000;
  var currentDataLenght=0;
  var dataLength = 1000;

  //Kada server pošalje podatke sa senzora ovom funkcijom se oni dodaju u
//graf i tablicu
  function putData(data){
    var JSONdata=JSON.parse(data);
    BMP280temp.push({x: xVal,y: JSONdata.BMP280.temperature});
```

```
    BME280temp.push({x:xVal, y: JSONdata.BME280.temperature});
    DS18S20temp.push({x:xVal, y: JSONdata.DS18S20.temperature});
    DHT11humidity.push({x:xVal, y: JSONdata.DHT11.humidity});
    BME280humidity.push({x: xVal, y: JSONdata.BME280.humidity});
    BMP280pressure.push({x: xVal,y: JSONdata.BMP280.pressure});
    if (currentDataLenght > dataLength) {
      BMP280temp.shift();
      BME280temp.shift();
      DS18S20temp.shift();
      DHT11humidity.shift();
      BME280humidity.shift();
      BMP280pressure.shift();
    }
    document.getElementById("DS18S20temp").innerHTML=DS18S20temp[DS18S20temp.
length-1].y+" °C";
    document.getElementById("BMP280temp").innerHTML=BMP280temp[BMP280temp.len
gth-1].y+" °C";
    document.getElementById("BME280temp").innerHTML=BME280temp[BME280temp.len
gth-1].y+" °C";
    document.getElementById("DHT11humidity").innerHTML=DHT11humidity[DHT11hum
idity.length-1].y+" %";
    document.getElementById("BME280humidity").innerHTML=BME280humidity[BME280
humidity.length-1].y+" %";
    document.getElementById("BMP280pressure").innerHTML=BMP280pressure[BMP280
pressure.length-1].y+" Pa";
    currentDataLenght++;
    currentGraphObject.render();
    xVal++;
  }

//Pri promjeni veličine prozora preglednika prilagođava se veličina grafa
function adjustgraph(){
  chartTemperature.render();
  chartHumidity.render();
  chartPressure.render();
}

window.onresize=adjustgraph;
window.onload=settingvariables();
```

Prilog P.3.5.

```
#include <Arduino.h>
#include <WiFi.h>
#include <AsyncTCP.h>
#include <ESPAsyncWebServer.h>
#include <WebSocketsServer.h>
#include "SPIFFS.h"
#include "FS.h"
#include <OneWire.h>
#include <DallasTemperature.h>
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <DHT.h>
#include "SparkFunBME280.h"
#include <ArduinoJson.h>

//Stvaranje AsyncWebServer objekta na portu 80
AsyncWebServer server(80);
//Stvaranje WebSocket objekta
WebSocketsServer webSocket = WebSocketsServer(1337);
char msg_buff[100];
//Odabir pina i definiranje senzora DHT11
#define DHTPIN 5
#define DHTTYPE DHT11
DHT dht(DHTPIN, DHTTYPE);
//Odabir pina i definiranje potrebnih objekata za rad senzora DS18S20
#define DS18S20PIN 4
OneWire oneWire(DS18S20PIN);
DallasTemperature DS18S20sensor(&oneWire);
//Definiranje pinova za I2C komunikaciju s BME280 i BMP280
#define SDA_0 21
#define SCL_0 22

#define SDA_1 32
#define SCL_1 33

TwoWire I2C_0 = TwoWire(0);
TwoWire I2C_1 = TwoWire(1);
BME280 mySensorA;
BME280 mySensorB;

//Definiranje pina, frekvencije i rezolucije za PWM signal kojime se upravlja
//grijač
#define PWM_PIN   15
#define PWM1_Ch    0
#define PWM1_Res   8
#define PWM1_Freq  1000

//Zadaci za upravljanje grijačem
TaskHandle_t manualRegulation=NULL;
TaskHandle_t automaticRegulation=NULL;
//Trenutna očitana vrijednost sa senzora
float TBMEtemp,TBMPtemp,TDS18S20temp, TDHT11humidity, TBMEhumidity, TBMPpress
ure;
//Željena temperatura leće i željena snaga PWM
int desiredTemperature, desiredPWM;
//Brojač pomoću kojeg se svakih dvije minute sprema vrijednost temperature BM
//P280 u BMP280tempM
```

```
int brojac=24;
float BMP280tempM;
//Funkcija za slanje podataka svim spojenim klijentima. Maksimalno 3 klijenta
//mogu dobivati podatke.
void sendToAllClients(String message){
  for(int i=0;i<3;i++){
    webSocket.sendTXT(i, message);
  }
}

//Funkcija u kojoj se postavljaju adrese za BME280 i BMP280, te ESP32
//započinje komunikaciju sa senzorima
void initializeSensors(){

  //BMP280 i BME280
  I2C_0.begin(SDA_0 , SCL_0 , 100000 );
  I2C_1.begin(SDA_1 , SCL_1 , 100000 );
  mySensorA.settings.commInterface = I2C_MODE;
  mySensorA.setI2CAddress(0x77);


  mySensorA.setReferencePressure(101200);
  if(mySensorA.beginI2C(I2C_0) == false) Serial.println("Sensor A connect
   failed");

  mySensorB.settings.commInterface = I2C_MODE;
  mySensorB.setI2CAddress(0x76);


  if(mySensorB.beginI2C(I2C_1) == false) Serial.println("Sensor B connect
   failed");
  //DHT11
  dht.begin();

  //DS18S20
  DS18S20sensor.begin();
}

//Funkcija za prikupljanje podataka sa senzora i stvaranje JSON objekta kojeg
//se šalje klijentu
void GetMeasures(void * parameter){

  for(;;){
  //Spremanje izmjerenih vrijednosti
  float temp=0;
  temp=(float)mySensorB.readTempC();
  TBMEtemp=temp;
  temp=(float)mySensorB.readFloatHumidity();
  TBMEhumidity=temp;
  DS18S20sensor.requestTemperatures();
  temp=(float)DS18S20sensor.getTempCByIndex(0);
  if(temp!=(-127) && temp!=85){TDS18S20temp=temp;}
  temp=(float)dht.readHumidity();
  TDHT11humidity=temp;
  temp=(float)mySensorA.readTempC();
  TBMPtemp=temp;
  temp=(float)mySensorA.readFloatPressure();
  TBMPpressure=temp;

  //Unos izmjerenih vrijednosti u JSON koji se šalje klijentima
```

```
  DynamicJsonDocument doc(1024);
  JsonObject BME280JSON=doc.createNestedObject("BME280");
  BME280JSON["temperature"]=TBMEtemp;
  BME280JSON["humidity"]=TBMEhumidity;
  JsonObject DS18S20sensorJSON  = doc.createNestedObject("DS18S20");
  DS18S20sensorJSON["temperature"]= TDS18S20temp;
  JsonObject DHT11JSON= doc.createNestedObject("DHT11");
  DHT11JSON["humidity"]=TDHT11humidity;
  JsonObject BMP280JSON=doc.createNestedObject("BMP280");
  BMP280JSON["temperature"]=TBMPtemp;
  BMP280JSON["pressure"]=TBMPpressure;
  String message;
  serializeJson(doc,message);
  Serial.println(message);
  webSocket.broadcastTXT(message);
  if(brojac==24){
    BMP280tempM=TBMPtemp;
    brojac=0;
  }
  brojac++;
  //Zaustavljanje izvođenja ove funkcije te oslobađanje jezgre koja izvršava
//ovu funkciju na 5 sekundi
  vTaskDelay(5000 / portTICK_PERIOD_MS);

  }
}


//Ime i lozinka za pristup Wi- Fi konekciji
const char* ssid = "ESP32sustav";
const char* password = "lozinka123";

//Inicijalizacija SPIFF sustava
void initializeSPIFFS(){
  if(!SPIFFS.begin(true)){
        Serial.println("SPIFFS Mount Failed");
        return;
    }
}

//Inicijalizacija Wi - Fi mreže. Postavljanje da se ESP32 mikroupravljač
//ponaša kao pristupna točka te imenovanje mreže
void initWiFi(){
Serial.println("Configuring access point...");
WiFi.softAP(ssid, password);
  IPAddress myIP = WiFi.softAPIP();
  Serial.print("AP IP address: ");
  Serial.println(myIP);
}

//Funkcija koja uključuje grijač te grije optiku dok ona ne postigne željenu
//temperaturu koju je klijent odredio
void Manual(void* pvParameters){
  for(;;){
  if(TDS18S20temp < (float)desiredTemperature){
    ledcWrite(PWM1_Ch, 130);
  }
  else{
    ledcWrite(PWM1_Ch, 0);
  }
```

```
  vTaskDelay(6500 / portTICK_PERIOD_MS);
  }


}
//Funkcija za automatsko kontroliranje temperature. Grijač se uključuje u
//slučaju da je optika hladnija od okoline ili zraka u blizini optike.
//Kada su promjene male te razlika temperature u blizini optike i
//okoline manja od 2 stupnja optika se regulira prema senzoru u blizini
//optike (BME280)
//U slučaju kada se događaju veće promjene
//temperature optika se regulira prema senzoru u okolini (BMP280)
void Automatic(void* pvParameters){
  for(;;){
  if(((TBMEtemp-TBMPtemp)<1 && (TBMEtemp-TBMPtemp)>(-1)) && (TBMPtemp-
BMP280tempM)<2){
    if(TBMEtemp>TDS18S20temp){
      ledcWrite(PWM1_Ch, 130);
    }
    else{
      ledcWrite(PWM1_Ch, 0);
    }
  }
  else{
    if(TBMPtemp>TDS18S20temp){
      ledcWrite(PWM1_Ch, 130);
    }
    else{
      ledcWrite(PWM1_Ch, 0);
    }
  }
  Serial.print("Auto mode ");
vTaskDelay(6000 / portTICK_PERIOD_MS);
  }
}

//Funkcija koja upravlja WebSocket komunikacijom između klijenta i servera
void onWebSocketEvent(uint8_t client_num, WStype_t type, uint8_t * payload, s
ize_t lenght){

  switch (type)
  {

  case WStype_DISCONNECTED:
    Serial.printf("[%u] Disconnected!\n", client_num);
    break;

  //Kada se uspostavi WebSocket komunikacija sa klijentom, server mu šalje
 //broj kojim se kasnije određuje može li klijent upravljati sustavom
  case WStype_CONNECTED:{
    IPAddress ip = webSocket.remoteIP(client_num);
    Serial.printf("[%u] Connectuion from: ", client_num);
    Serial.println(ip.toString());
    sprintf(msg_buff,"CL%d",client_num);
    webSocket.sendTXT(client_num,msg_buff);
    break;}


//U slučaju da klijent pošalje zahtjev za promjenu načina rada grijača,
//zaustavlja se trenutni način  rada i pokreće novo odabrani način
```

```
    case WStype_TEXT:
    {
      Serial.printf("[%u] Received text: %s\n", client_num, payload);

      if(payload[0]=='W' && payload[1]=='M'){
        //Korisnik želi da sustav samostalno regulira temperaturu optike
        if(payload[2]=='0'){
          ledcWrite(PWM1_Ch, 0);
          vTaskSuspend(manualRegulation);
          vTaskResume(automaticRegulation);
        }
        //Korisnik želi da optika bude određene temperature
        else if(payload[2]=='1'){
          ledcWrite(PWM1_Ch, 0);
          sprintf(msg_buff,"%s",payload);
          String value=String(msg_buff).substring(3);
          desiredTemperature=atoi(value.c_str());
          vTaskSuspend(automaticRegulation);
          vTaskResume(manualRegulation);
        }
        //Korisnik želi da grijač bude uključen bez obzira na temperaturu
          //optike i okoline
        else if(payload[2]=='2'){
          vTaskSuspend(automaticRegulation);
          vTaskSuspend(manualRegulation);
          ledcWrite(PWM1_Ch, 130);}
        }
        //Korisnik je odabrao željeni PWM
        else if(payload[2]=='3'){
          sprintf(msg_buff,"%s",payload);
          String value=String(msg_buff).substring(3);
          desiredPWM=atoi(value.c_str());
          vTaskSuspend(automaticRegulation);
          vTaskSuspend(manualRegulation);
          ledcWrite(PWM1_Ch, desiredPWM);
        }

      break;}

    case WStype_BIN:
    case WStype_ERROR:
    case WStype_FRAGMENT_TEXT_START:
    case WStype_FRAGMENT_BIN_START:
    case WStype_FRAGMENT:
    case WStype_FRAGMENT_FIN:
    default:
      break;
    }
}

void setup() {
  Serial.begin(115200);
  //Pozivanje funkcija za inicijalizaciju
  initializeSPIFFS();
  initializeSensors();
  delay(2000);
  initWiFi();
  //PWM inicijalizacija
  ledcAttachPin(PWM_PIN, PWM1_Ch);
  ledcSetup(PWM1_Ch, PWM1_Freq, PWM1_Res);
```

```
  //Postavljanje pripadajućih datoteka za HTTP_GET zahtjeve
  server.on("/", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/index.html");
  });

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

  server.on("/script.js", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/script.js", "text/javascript");
  });

  server.on("/canvasjs.min.js", HTTP_GET, [](AsyncWebServerRequest *request){
    request->send(SPIFFS, "/canvasjs.min.js", "text/javascript");
  });


  server.begin();
  webSocket.begin();
  webSocket.onEvent(onWebSocketEvent);

//Stvaranje zadatka koji prikuplja mjerenja sa senzora
  xTaskCreatePinnedToCore(
    GetMeasures,      // Funkcija koja se izvodi
    "JSON",
    8000,
    NULL,
    2,                  // Prioritet zadatka
    NULL,
    0          // Jezgra koja obavlja zadatak
);

  //Stvaranje zadataka za upravljanje grijačem, te zaustavljanje istih.
  //Pokreću se po potrebi kada klijent zatraži željeni način rada
  xTaskCreate(Manual, "Manual", 4000, NULL, 0, &manualRegulation);
  vTaskSuspend(manualRegulation);
  xTaskCreate(Automatic,"Automatic", 4000, NULL, 1, &automaticRegulation);
  vTaskSuspend(automaticRegulation);

}

void loop() {
  //Obrada sljedećeg zahtjeva od klijenta
  webSocket.loop();
}
```

Prilog P.3.6.

Ovo je JavaScript biblioteka koja se koristi za izradu grafova. Preuzeta je s poveznice https://canvasjs.com/assets/script/canvasjs.min.js.